Diss. ETH No. 28091

A GENERAL FRAMEWORK FOR HIGH-RESOLUTION ROBOTIC TACTILE SENSING: DESIGN, SIMULATION, AND LEARNING

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES of ETH Zurich

(Dr. sc. ETH Zurich)

presented by

Carmelo Sferrazza
M.Sc. ETH in Robotics, Systems, and Control

born on 14 January 1993
citizen of Italy

accepted on the recommendation of

Prof. Dr. Raffaello D'Andrea, examiner
Prof. Dr. Rebecca Kramer-Bottiglio, co-examiner
Dr. Katherine J. Kuchenbecker, co-examiner

2022

A general framework for high-resolution robotic tactile sensing: design, simulation, and learning

Carmelo Sferrazza

Institute for Dynamic Systems and Control
ETH Zurich
2022

Institute for Dynamic Systems and Control
ETH Zurich
Switzerland

Abstract

In order to fulfill their potential in the manufacturing and retail sectors of the modern world, autonomous machines need to be able to perceive and react to contact with their surroundings, both to enhance their capabilities, as well as to increase operational safety. This thesis investigates solutions to the contact sensing problem of robotic systems, pivoting on the development of a vision-based tactile sensing principle that provides rich information upon physical interaction with the environment. The sensors based on such a principle are low-cost, scalable to large surfaces and straightforward to manufacture. However, they do not directly measure physical quantities, but rather provide raw data in the form of what are generally known as tactile images. In this work, a machine learning-based data processing framework is presented to address three main requirements, namely, sensing accuracy, efficiency, and generalization across tasks and contact conditions.

State-of-the-art sensing accuracy, at a spatial resolution comparable to that of the human fingertip, is achieved through a deep neural network that maps the raw tactile images to the three-dimensional force distribution applied to the sensing surface, which provides a compact and generic representation of the contact state. In fact, the force distribution contains information about the location and the intensity of shear and pressure forces, as well as about the shape and the number of the possibly distinct contact regions. In addition, it provides an interpretable physical quantity that is shown to be very practical for planning higher-level robotic tasks.

The size of the neural network is kept compact to ensure real-time inference. However, in the context of data-driven methods, efficiency is also a concern with regard to training data requirements. In this thesis, accurate finite element-based simulations enable the synthetic generation of raw tactile data under a variety of contact conditions. The same simulations also yield appropriate force distribution labels, which are otherwise not possible to collect with currently existing commercial force sensors. Hence, the deep neural network is entirely trained with synthetic data, avoiding the need for real-world data collection. A strategy is then presented that facilitates a seamless transfer of the inference model from simulation to reality, retaining high sensing accuracy. In addition, the model transfers across sensors of the same type without further training.

The simulation training facilitates data collection across different scenarios, such as the contact with arbitrarily shaped objects or the combination of shear and pressure interactions. An appropriate choice of learning architecture shows generalization capabilities when applied to contact conditions not present in the training dataset. Beyond the pure sensing task, a proof-of-concept robotic system is presented that fully leverages the versatility of the tactile sensor. The system achieves dynamic manipulation of objects with unknown physical properties solely through the use of tactile feedback fed to a closed-loop control policy trained with a deep reinforcement learning algorithm.

In a separate part, this thesis discusses a different research topic, where past experience data are employed to improve the trajectory tracking performance of autonomous systems. This is achieved by estimating unmodeled disturbances over different trials, and including them in the formulation of a computationally efficient model predictive control framework. The approach is demonstrated on two flying vehicle applications, namely, on a vehicle powered with electric ducted fans and controlled through thrust vectoring, and on a quadcopter that aims to balance a pendulum rod during flight.

Riassunto

Per potere sfruttare a pieno il loro potenziale nel settore manifatturiero e in quello della vendita al dettaglio, nel mondo moderno le macchine autonome dovranno essere in grado di percepire e reagire al contatto con l'ambiente circostante, sia per migliorare le loro funzionalità, che per aumentare la sicurezza operativa. Questa tesi studia le soluzioni al problema della percezione del contatto dei sistemi robotici, facendo perno sullo sviluppo di un principio di misurazione tattile basato sull'acquisizione di immagini ricche di informazioni sull'interazione fisica con l'ambiente. I sensori basati su tale principio sono a basso costo, scalabili su grandi superfici e semplici da produrre. Tuttavia, questi non misurano direttamente delle quantità fisiche, ma piuttosto forniscono dati grezzi sotto forma di quelle che sono generalmente note come immagini tattili. In questa tesi, viene presentato un framework di elaborazione dei dati, basato sull'apprendimento automatico, che mira a soddisfare tre requisiti fondamentali: l'accuratezza di misurazione, l'efficienza, e la generalizzazione per varie mansioni e scenari di contatto.

Un'accuratezza di misurazione all'avanguardia, con una risoluzione spaziale paragonabile a quella dei polpastrelli dell'uomo, è ottenuta attraverso una rete neurale artificiale profonda che mappa alle immagini tattili grezze la distribuzione tridimensionale delle forze applicate alla superficie di misurazione, fornendo una rappresentazione compatta e generica dello stato del contatto. Infatti, la distribuzione di forza contiene informazioni sulla posizione e l'intensità delle forze normali e di taglio, così come sulla forma e il numero delle regioni di contatto, che possono anche essere molteplici. Inoltre, fornisce una quantità fisica interpretabile che si dimostra essere molto utile per i robot per la pianificazione di compiti di più alto livello.

La dimensione della rete neurale è mantenuta compatta per garantire un'inferenza in tempo reale. Tuttavia, nel contesto dei metodi basati sull'apprendimento automatico, l'efficienza deve essere garantita anche per quanto riguarda i requisiti dei dati di addestramento. In questa tesi, accurate simulazioni basate sul metodo a elementi finiti permettono la generazione sintetica di dati tattili grezzi per un'ampia varietà di scenari di contatto. Le stesse simulazioni producono anche le appropriate etichette di distribuzione di forza, che non sono altrimenti possibili da raccogliere con i sensori di forza attualmente in commercio. Quindi, la rete neurale profonda è interamente addestrata con dati sintetici, evitando la necessità di raccogliere dati nel mondo reale. Viene inoltre presentata una strategia che garantisce un agevole trasferimento del modello di inferenza dalla simulazione alla realtà, mantenendo un'elevata accuratezza di misurazione. Inoltre, il modello può essere trasferito tra sensori dello stesso tipo senza ulteriore addestramento.

L'addestramento in simulazione facilita la raccolta di dati in diversi scenari, come in quelli in cui il contatto avviene con oggetti di forma arbitraria o nel caso di interazioni che combinano forze normali e di taglio. Una scelta appropriata dell'architettura di ap-

prendimento mostra inoltre capacità di generalizzazione quando applicata a condizioni di contatto non presenti tra i dati di addestramento. Al di là del puro compito di misurazione, viene presentato un sistema robotico proof-of-concept che sfrutta pienamente la versatilità del sensore tattile. Il sistema effettua con successo la manipolazione dinamica di oggetti con proprietà fisiche sconosciute avendo a disposizione solo il feedback tattile, che viene fornito a un sistema di controllo addestrato con un algoritmo di apprendimento per rinforzo profondo.

In una parte separata, questa tesi discute un diverso argomento di ricerca, dove dati raccolti durante precedenti campagne sperimentali sono impiegati per migliorare le prestazioni di sistemi autonomi per quanto riguarda il controllo di traiettoria. Ciò si ottiene stimando attraverso esperimenti ripetuti i disturbi non considerati nel modello del sistema, e quindi includendoli nella formulazione di un algoritmo di controllo predittivo efficiente dal punto di vista computazionale. L'approccio è dimostrato tramite due applicazioni a veicoli volanti: su un veicolo alimentato con ventole intubate elettriche e controllato attraverso la vettorizzazione della spinta, e su un quadrirotore che mira a bilanciare un'asta durante il volo.

Acknowledgments

All the people I have met during my PhD have contributed in many ways to this thesis, with their diverse thoughts, suggestions, and feedback.

My deepest gratitude goes to my advisor, Raff, for his guidance and continuous support. I have had the perfect conditions and flexibility to shape my research project, and a great part of it is due to your trust, patience, and motivation, especially during the first months of my PhD. I have learned so much from you, and it has been invaluable to always be able to count on your direct and precious advice. Thank you for the opportunities you have offered, and for being an inspiration with your approach to research.

I would like to thank Prof. Ken Goldberg, Prof. Rebecca Kramer-Bottiglio, and Dr. Katherine Kuchenbecker, for agreeing to act as co-examiners, and for their willingness to spend considerable time reviewing this thesis. I am also grateful to Prof. Sebastian Trimpe and Prof. Melanie Zeilinger for their advice and support.

Being surrounded by brilliant colleagues at the Institute for Dynamic Systems and Control (IDSC) has been the highlight of my PhD, and their striving for excellence a natural motivation. I have been fortunate to have started my PhD at the same time as Matthias, who has been my office-mate since day one. Thank you for the everyday feedback exchanges, for all the stimulating discussions, for sharing the teaching duties of Recursive Estimation, and for the many hundreds of lunches together. I would also like to thank Michael for bringing me to IDSC for my Master's thesis. You were an amazing supervisor, and looking back, your passion for research truly inspired me to pursue a PhD. Having you later as a colleague was a great help in getting started with the group, and I have learned a lot from our work together. Thank you Rajan and Weixuan, for the late lunches and the street food adventures. And thank you Tony, for being always willing to help with anything. I am also thankful to my senior colleagues, Dario, Max, Mike, and Robin, for being very welcoming, and for passing on the group's culture. I would also like to thank Adam from the Institute for Mechanical Systems, for the wonderful collaboration and for the many insights you gave me about continuum mechanics. I owe a huge thank you to all the stellar students that I have supervised at ETH, for your efforts and passion, for making my project yours.

The support staff at IDSC do an impressive job in keeping things running. I would like to thank Helen for being a savior in literally everything, and for proofreading the million pages of papers and documents. And thank you Katharina for the administrative support when starting my time at IDSC. I am also thankful for the invaluable technical support by Michi, Matthias Mueller, Mac and Dani. And thank you Marcus, both for your help and the weekend get-togethers.

I cannot thank enough everyone in my big family. I want to thank my father, for teaching me that there is always time for the important things in life. And thank you

mom for unconditionally backing all my decisions. I would also like to say a heartfelt thank you to my sisters, for being at the same time both my role models and my first supporters. I am also grateful to all my friends, for making any place in the world feel like at home.

Finally, no words could express my gratitude to Margarita. You have been the one to look up to, my strength during the tough and crazy times, my joy when celebrating our achievements. Thank you for your endless love and support.

Contents

Preface

This thesis documents the research carried out by the author during his doctoral studies under the supervision of Professor Raffaello D'Andrea at the Institute for Dynamic Systems and Control at ETH Zurich between January 2017 and December 2021.

The work is presented in the form of a cumulative thesis: its main content consists of seven self-contained research articles (of which four are journal articles and three are conference contributions) that have been published during the doctoral studies.

The work is divided into two parts: the design of a novel vision-based tactile sensor, and the development of a data processing framework are presented in Part A, while Part B deals with the implementation of a learning-based trajectory tracking approach within a computationally efficient model predictive control framework. These two parts address different problems and research topics, but they both leverage data to enhance the capabilities of autonomous systems.

The articles are put into context by three introductory chapters, which are structured as follows: Chapter 1 introduces and motivates this work, including the problems considered, related work, and the approaches used. Chapter 2 describes the key contributions of the research papers included in this thesis and how the individual papers relate to each other. Chapter 3 then provides a discussion of potential extensions and new directions of this research.

1

Introduction

Humans rely on their sense of touch as the primary means to physically interact with the environment. Robots, conversely, have so far relied on their sense of vision to both plan and execute the majority of their interactive tasks, such as those involving grasping and manipulation [1]–[3]. As opposed to what happens in humans, many of these tasks are often approached by robots with minimal or absent contact feedback, leveraging the rich and accurate stream of visual information provided by modern cameras. However, contact feedback remains of the utmost importance when it comes to reducing uncertainty during interactions with small or fragile objects [4], for fine manipulation tasks [5], or in conditions where visual information deteriorates or is insufficient [6], such as when coping with the occlusions naturally caused by a grasping motion.

Aiming to address such needs, tactile sensing research has focused on providing machines with rich contact information upon interaction with the environment. However, decades of advancements [7] have not yet produced a comprehensive solution such as the one represented by cameras and image sensors for the computer vision community. One of the main difficulties lies in reliably obtaining distributed information without wiring thousands of sensors over the entire robot body. Moreover, processing a stream of tactile data into meaningful measurements is particularly challenging, as it is generally infeasible to accurately model contact between surfaces in real time, especially when this involves soft materials. In fact, such materials are typically desired at the interface of the sensor with the external world, to increase friction, compliance and conformation to the surface of the objects that a robot is picking or manipulating.

In addition to enhancing robots' autonomy, the advancement of an artificial sense of touch shows promise to benefit numerous different applications, spanning from the entertainment industry [8] to the medical domain, such as in the development of smart prosthetics [9].

The gap between the artificial senses of vision and touch has inspired researchers [10]–[12] to leverage the high resolution of cameras to convert contact stimuli into a tactile image [13], by capturing the deformation of a soft sensing surface. In this context, Part A of this thesis first describes the design of such a vision-based tactile sensor, which features high spatial resolution and is particularly suited to cover surfaces of arbitrary shape, while keeping manufacturing complexity to a minimum. This part also discusses the development of a data-driven framework that avoids the need for real-time modeling to process the tactile images and extract distributed physical quantities. The framework

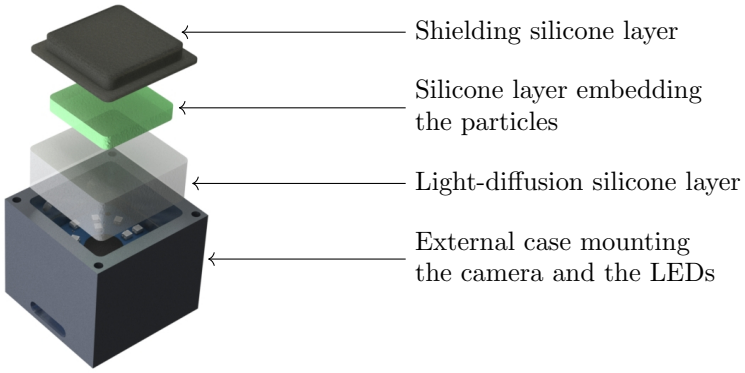


Figure 1.1 Exploded view of the sensor discussed in this thesis.

focuses on efficiency and exhibits strong generalization power. In addition, the appendix discusses a scalability application of the tactile sensing technique, followed by a dynamic manipulation application that leverages the features of the sensor.

Part B of this thesis stems from of a different research project and is unrelated to the research done in the context of tactile sensing. However, data are also at the core of this part, where measurements from previous experience are processed to compensate for unmodeled repeatable disturbances in a model predictive control setting. The approach presented is particularly efficient both in terms of sampling frequency and memory requirements.

The context for each part is presented below, while the contributions made in the thesis (and specifically the contributions of the papers in this thesis) are discussed in Chapter 2.

1.1 Data-driven, vision-based tactile sensing

Among the several tactile sensing principles [14]–[18] that have been developed during the last decades, optical (or vision-based) tactile sensors benefit from low cost, high resolution and ease of wiring, while retaining the softness of their sensing surface. The vision-based tactile sensor presented in this thesis (see Fig. 1.1) aims to maximize the information contained in the tactile images by mixing thousands of spherical particles within an elastomer, which is placed on top of a compact camera surrounded by LEDs. The deformation of the sensing surface upon contact with external bodies induces the motion of the particles, which is tracked by the internal camera, providing information about the forces causing such deformation. As opposed to other approaches in the literature [19]–[21], the denseness of the particle patterns is especially effective at creating strain information at each pixel of the tactile image, and their randomness further simplifies manufacture. The simplicity of the fabrication technique enables a straightforward design of sensing surfaces with different shapes and geometry, depending on the application requirements.

While vision-based tactile sensors provide rich, qualitatively interpretable tactile images, extracting relevant physical quantities from such images is challenging, as this gen-

erally requires modeling the behavior of the soft materials involved, as well as the optics of the internal camera. As a consequence, most of the literature has focused on the estimation of low-dimensional quantities, such as resulting forces [22] and contact locations [23]. This thesis focuses on the estimation of the three-dimensional contact force distribution, to retain versatile and high-resolution information. In fact, a variety of contact quantities may be extracted from the force distribution, including the resulting forces and contact regions. In addition, this representation naturally scales to scenarios with multiple points of contact, and yields a physical abstraction from the images, which facilitates robotics applications. Finally, as opposed to reconstructing the surface deformation [24], the force distribution exactly encodes the shape of the contact regions, whereas the elastic material may instead deform even in the regions surrounding those actually in contact.

Data processing pipelines in the literature have either been addressed by employing heuristics and model simplifications [12], [25], [26], or supervised learning [22]. In particular, the latter strategy has recently been of growing interest in the literature, even outside the principles based on vision [27], [28], as it generally exhibits accurate force estimation without compromising real-time execution. However, the main concerns raised towards such learning-based approaches are data efficiency and cross-task generalization. The image processing framework proposed in this thesis leverages computer vision and deep learning algorithms to predict the three-dimensional force distribution from the tactile images. First, an automated strategy to systematically collect real-world data is investigated. Then, a solution is proposed to generate very accurate training data in a simulation based on hyperelastic material models and including the camera projection, entirely avoiding the need for experimental data collection. The choice of the training scenarios and the learning architecture yield generalization across contact with a variety of objects, as shown in Fig. 1.2. A seamless sim-to-real transfer enables the pipeline to achieve high accuracy on real-world images, being able to predict the force distribution in real time at a frequency of over 100 Hz on the single core of a standard laptop CPU.

The transfer of data-driven tactile sensing models across different fabricated instances of the same sensor has never been investigated in detail, although it is widely recognized to be a crucial feature with regard to the scalability of these sensors. In the context of vision-based tactile sensors, a limiting factor for their integration with robotic platforms has also been their footprint, due to the thickness of their internal optical unit, and their scalability to larger surfaces. In this thesis, a transfer learning strategy is proposed to transfer knowledge across sensors with limited real-world data requirements. Then, the simulated world is leveraged to propose another strategy that does not require any additional samples to transfer the neural network maps to any sensors of the same type. Finally, a multi-camera sensor is presented in the appendix that exhibits reduced thickness and a larger sensing surface. In this context, a scalable framework is proposed to reuse information across different regions of the same surface.

The development of many sensing principles and devices, which has been a consequence of the years invested in researching a comprehensive sensing solution, has resulted in software and algorithms that are tailored to the specific sensors employed [29]–[31].

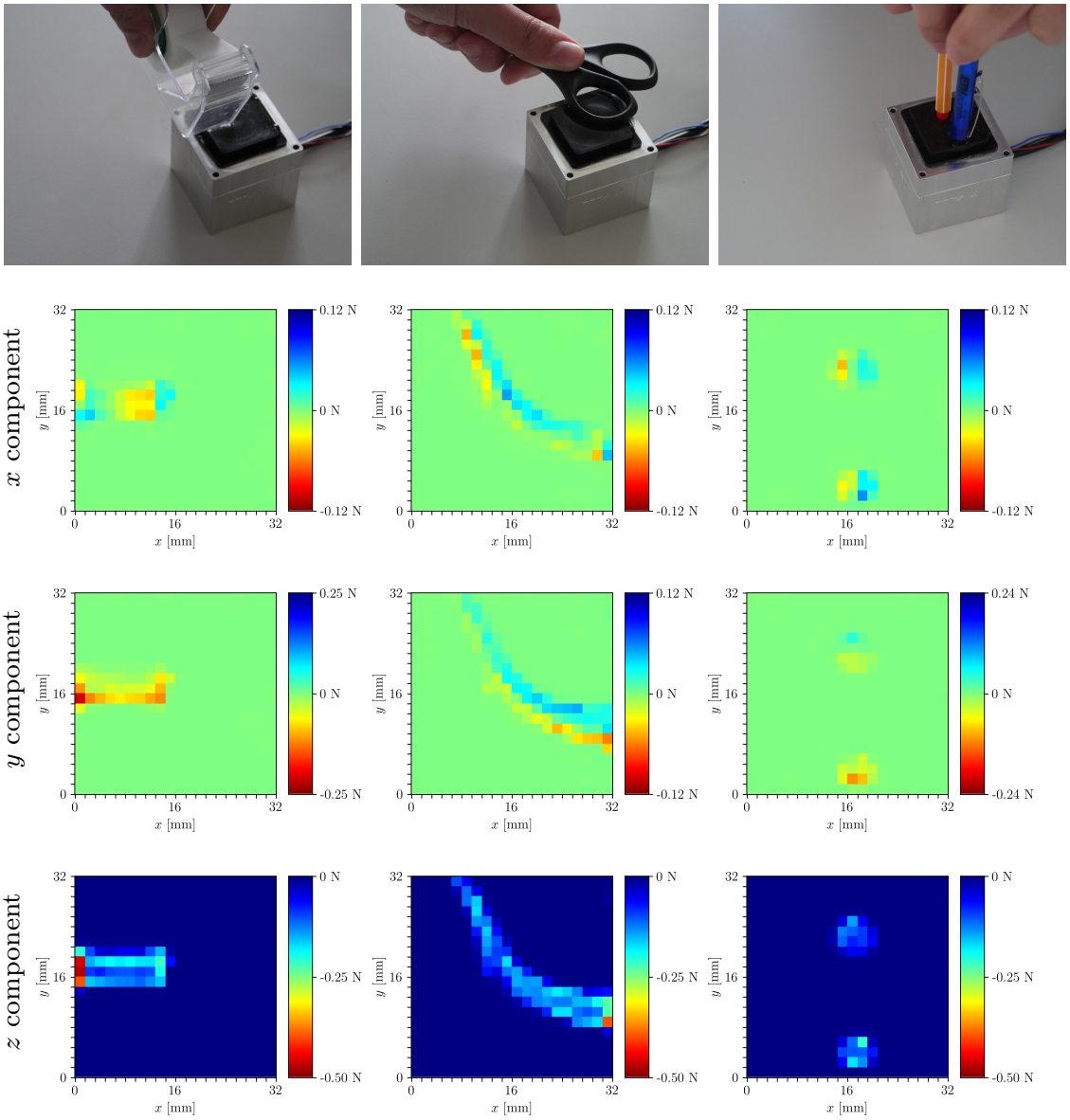


Figure 1.2 Examples of the predicted 3D force distribution for different contact objects and conditions. The distribution is estimated at a finite number of discrete bins.

The emphasis of this work on estimating general physical quantities, as opposed to directly using the raw camera output, targets this reusability issue in the tactile sensing research community. An example is presented in the appendix with a dynamic manipulation task, where a reinforcement learning policy (entirely trained on a novel simulator, and deployed in the real-world) is able to swing up rods similar to a classical inverted pendulum scenario. Rather than being attached to a pivot point, the pendulum is controlled by a parallel-jaw gripper that delicately adjusts the gripping force to achieve the task, uniquely based on the tactile feedback in the form of the force distribution. In fact, the entire framework was developed at the level of the forces, without relying on any other information from the images, which greatly increases efficiency. In addition, this shows promise to facilitate the direct transfer of such an approach to similar systems em-

ploying different tactile sensing principles, provided that they can estimate distributed forces. In this regard, the strategy presented in this thesis to generate ground truth force distributions is also generally applicable and not limited to vision-based tactile sensors.

1.2 Computationally efficient learning-based model predictive control

Model predictive control (MPC) [32] is a powerful feedback control strategy that is particularly suited to handle system constraints. This strategy has become a standard in the process industry [33] and is now seeing a growing interest across a wide range of different fields. However, the fact that it is based on repeatedly solving an optimization problem in real time has prevented its application to systems that exhibit fast dynamics. In this context, several techniques have been developed to reduce the computational burden of MPC. While most of them rely on an early truncation of the prediction horizon [34], with a consequent reduction of the predictive power, others have suggested parametrizing the states and inputs of the system with basis functions [35], to retain high prediction power and considerably reduce the number of optimization variables at the same time. In this thesis, a parametrized MPC strategy is extended to the trajectory tracking case, where an autonomous system aims to track non-equilibrium motions over time. While the approach relies on a linearized model of the system of interest, nonlinearities and other unmodeled disturbances are estimated at each trajectory trial through a Kalman filter, and included in the model for the subsequent trials. This has the effect of considerably improving the tracking performance, while being suitable for real-time and onboard control, in addition to showing a considerable reduction of the memory needed to store and load trajectories. Two different variations of the approach proposed are discussed in this work, each in the context of a different flying vehicle application. First, the approach is simplified through some heuristics to track trajectories flown with a vehicle [36] consisting of three electric ducted fans mounted on a lightweight frame. MPC is applied onboard the vehicle, which is steered through two flaps attached at each of the fan outlets to redirect the airflow and perform thrust vectoring. Then, a more general approach is applied to a quadcopter system that aims to balance a pendulum while flying aggressive trajectories, refining and improving its performance over subsequent trials.

2

Contributions

This chapter summarizes the scientific contributions for each of the papers that constitute this thesis. In total, four journal publications and three peer-reviewed conference proceedings are discussed.

In addition, a peer-reviewed conference proceeding and a journal publication are placed in the concluding appendix section. These two contributions stemmed from Master's thesis projects directly supervised by the author. Responsibilities included defining the student project, including the idea and an appropriate scope, recruiting the student, constantly discussing and supporting the research project, introducing the student to the testbeds, and supporting them in writing the resulting contributions.

Furthermore, a list of other contributions such as results from unpublished student projects and outreach activities are provided in this chapter.

2.1 Part A: Data-driven, vision-based tactile sensing

[P1] C. Sferrazza and R. D'Andrea, "Design, Motivation and Evaluation of a Full-Resolution Optical Tactile Sensor", *Sensors*, vol. 19, no. 4: 928, 2019

This article describes the design of a vision-based tactile sensing principle that exhibits ease of manufacture and scalability to surfaces with arbitrary shapes. A random pattern of spherical particles (or markers) is embedded within a soft material, and an internal camera is employed to track the motion of such particles upon interaction with external bodies. The placement of the particles across the entire volume of the sensing elastomer is motivated with a simplified analysis based on a linear elastic model, where such placement exhibits an advantageous trade-off between sensor threshold and robustness to noise. In fact, the particles closer to the surface contribute to lowering the minimum detectable force, while at the same time information is diversified by placing the markers at different depths. A deep neural network is trained on experimental data to process the tactile images in a supervised learning fashion. Specifically, an automatic machine is programmed to collect thousands of datapoints by pressing a needle-shaped indenter against the sensing surface at different locations and depths. Optical flow features (with respect to an image at rest) are collected for each of such indentations, while a force sensor measures the

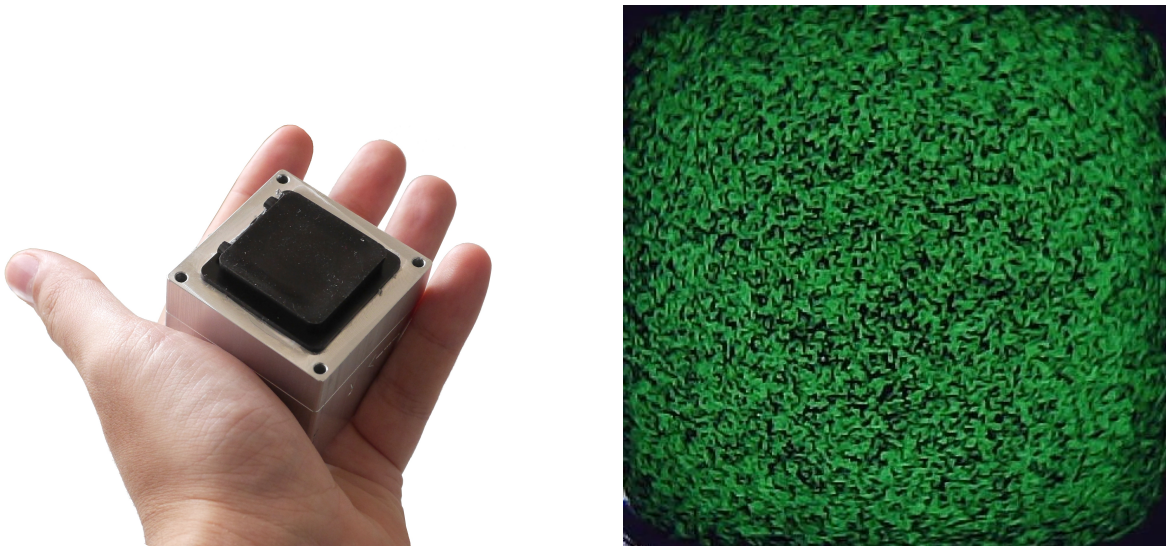
corresponding normal force. Then, a discretized representation of the normal contact force distribution is derived from the total force measurement for the case considered, and this is used to label the corresponding features. The resulting sensing pipeline reconstructs the force distribution for the simple case with high accuracy and in real time. The spatial resolution and sensing range achieved are comparable to those of the human fingertip. A comparative study shows double higher location accuracy when leveraging the strain information at all pixels, rather than relying on sparse marker displacements. For this reason, starting from [P2], the particles' size is reduced to increase their quantity and further exploit the dense information (see Fig. 2.1).

[P2] C. Sferrazza and R. D'Andrea, "Transfer learning for vision-based tactile sensing", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 7961–7967

When employing data-driven models to process the images of vision-based tactile sensors, such models do not generally transfer across different realizations of the same sensor. The most common approach is to collect a new dataset and retrain the model for each sensor realization. In [P1], optical flow features are extracted from the images and averaged within small image regions before being fed to a neural network, in order to minimize the influence of the specific particle distribution. Nevertheless, misalignments in the camera placement and differences between the cameras' parameters lead to considerable loss in accuracy when evaluating the neural network on a sensor realization different to that used to collect the training data. This paper proposes to augment the original network architecture with a calibration layer that can be trained by freezing the original network weights. This leads to a reduction of the training data requirements, with the model accuracy being retained across sensors by retraining the new layer with only 10% of the original data otherwise needed for a new sensor realization.

[P3] C. Sferrazza, A. Wahlsten, C. Trueeb, and R. D'Andrea, "Ground Truth Force Distribution for Learning-Based Tactile Sensing: A Finite Element Approach", *IEEE Access*, vol. 7, pp. 173 438–173 449, 2019

While machine learning approaches have been largely employed to process raw tactile data, these have generally been limited to the estimation of low-dimensional quantities such as total contact forces or locations. The estimation of more general and informative quantities such as the 3D contact force distributions has been impaired by the lack of commercially available sensors to label the raw tactile data with ground truth for such quantities. In this context, the work in [P1] and [P2] is limited to indentations with needle-shaped objects. Such works rely on the assumption that the contact surface lies entirely within one of the discrete bins where the force distribution is measured, and use an external force-torque sensor to compute appropriate single-entry labels for the image features captured during data collection. This article, conversely, deals with cases where the contact surface spans multiple of such bins. Therefore, a method is presented to provide ground truth force distributions by extracting them in a finite element simulation



(a) Sensor protoype **(b)** Particles image

Figure 2.1 A prototype of the tactile sensor employed is shown in (a). The dense spread of green particles is captured by the internal camera at a state of zero force in (b).

of the sensor's materials. The soft materials' hyperelastic models are obtained through state-of-the-art characterization experiments. The models are then evaluated in an indentation setup (considerably different from the material characterization tests) against a commercial force sensor, showing high accuracy in all the 3D components, superior to that achieved by linear elastic models. Thousands of labels are collected in the finite element environment within a scalable, parallelized framework, where indentations with a finger-shaped indenter are commanded to replicate those made in the real-world data collection on the sensor described in [P1], [P2]. The optical-flow features collected from such a sensor are matched to the synthetic labels, and the resulting *mixed-source* dataset is then used to train a neural network. This achieves high accuracy on the indenter used for training and is suitable for real-time 3D force distribution inference. While this article derives a data-processing mapping specific to the sensing technique discussed in this thesis, the label generation strategy is directly transferable to the majority of soft tactile sensors, independent of the underlying principle.

[P4] C. Sferrazza, T. Bi, and R. D'Andrea, "Learning the sense of touch in simulation: a sim-to-real strategy for vision-based tactile sensing", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020

Data-driven approaches such as those in [P1]–[P3] generally require a time-consuming and complex experimental data collection. Although accurate labels are collected in simulation in [P3], they still require matching to features extracted from real-world images. In addition, these approaches do not necessarily generalize well to contact conditions considerably different than those seen during training. This paper addresses both these

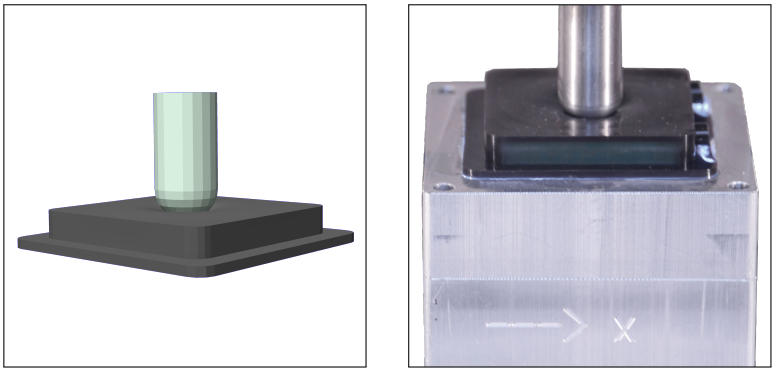
	[P1]	[P2]	[P3]	[P4]	[P5]
Training indenter	Needle	Needle	Finger-shaped	Finger-shaped	21 different indenters
Type of training data	Normal indentations	Normal indentations	Normal indentations	Normal indentations	Normal, shear, 3D indentations
Source of training data	Real-world	Real-world	Real-world features, simulated labels	Simulation	Simulation
Spatial resolution (bin size)	1.68 mm	3.55 mm	1.6 mm	1.6 mm	1.6 mm
Features employed	Dense optical flow	Dense optical flow	Dense optical flow	Dense optical flow	Pixel intensities
Neural network	Fully connected	Fully connected	Fully connected	Fully convolutional	Fully convolutional
Generalization	Only valid for one indenter, only normal indentations	Only valid for one indenter, only normal indentations	Only valid for one indenter, only normal indentations	Valid across several indenters, only normal indentations	Valid across generic indenters, across generic contact conditions
Transfer across sensor's realizations	No	Yes, via retraining with a portion of real-world data	No	Yes, without additional data, provided that the camera is calibrated	Yes, without additional data, provided that the camera is calibrated
Sensing frequency (CPU)	60 Hz	60 Hz	40 Hz	50 Hz	120 Hz

Table 2.1 The table provides a comparison between the different models described in the tactile sensing papers included in this thesis. Only the best model for each paper is shown in the table.

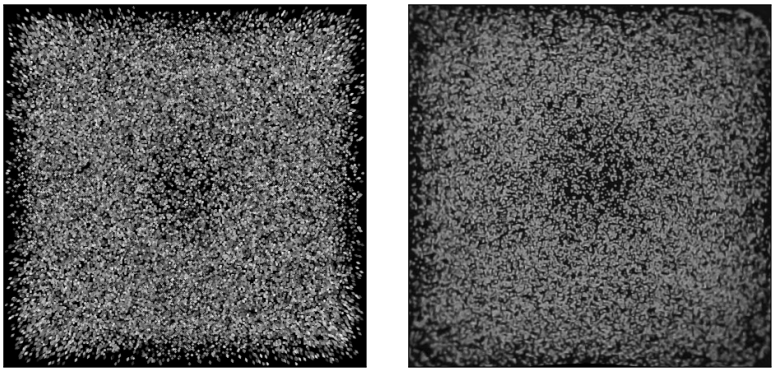
issues, by simulating the remaining part of the pipeline, that is, the camera projection, greatly reducing the data collection efforts. Specifically, a simplified camera model is introduced in simulation, based on which synthetic optical flow features are computed directly from the material deformation and matched to force distribution labels computed in the same simulation environment, where thousands of indentations are made on the simulated sensor. In the real-world, an accurate camera model is extracted through camera calibration and employed to remap the captured images as if they were shot by the camera used in simulation. Then, the optical flow features are extracted from such remapped images. Although the same finger-shaped indenter as in [P3] is used for training, the fully-convolutional neural network (based on an encoder-decoder architecture), which is entirely trained with synthetic features, retains high accuracy on real-world data and generalizes to unseen contact conditions, such as contact with multiple bodies. Moreover, the remapping strategy has the additional benefit of directly transferring the model across different sensor realizations that share the same elastomer geometry and mechanical properties, without the need for additional training.

[P5] C. Sferrazza and R. D'Andrea, "Sim-to-Real for High-Resolution Optical Tactile Sensing: From Images to Three-Dimensional Contact Force Distributions", *Soft Robotics*, 2021

The models trained in [P1]–[P4] all consider indentations made perpendicular (normal) to the sensing surface. However, shear-dominant interactions are of the utmost importance



(a) Simulated indentation **(b)** Real-world indentation



(c) Simulated image **(d)** Real-world image

Figure 2.2 Comparison between simulated and real world. The image in (c) was generated as described in [P5], while the one in (d) was obtained by converting a real-world image to grayscale.

for monitoring phenomena such as slippage. In addition, the optical flow simulation in [P4] relies on a simplification that is only valid for a limited deformation range, which is violated when applying large shearing motions or making contact with large indenters. In this context, this article replaces the optical flow simulator with the generation of synthetic tactile images (see Fig. 2.2), by projecting fictitious spherical particles through the simplified simulation camera model. A dataset comprising combinations of pressure and shear-dominant indentations, made with a variety of different indenters, is generated in such simulation, greatly simplifying the efforts otherwise needed to build complex hardware setups to collect such diverse data. Optical flow features are extracted from the generated images, and a tailored neural network is trained entirely with synthetic data to predict the three-dimensional contact force distribution. This is then compared to a similar network that directly learns from the pixels of the simulated images. Once it is deployed to the real world through the same remapping strategy as in [P4], the pixel-based network outperforms the strategy based on optical flow, both in terms of accuracy and inference speed (as the computation of the optical flow is avoided), reaching a sensing frequency of 120 Hz. The diverse training dataset further improves the generalization power of the network on generic contact conditions, including those featuring strong shear motions.

2.2 Part B: Computationally efficient learning-based model predictive control

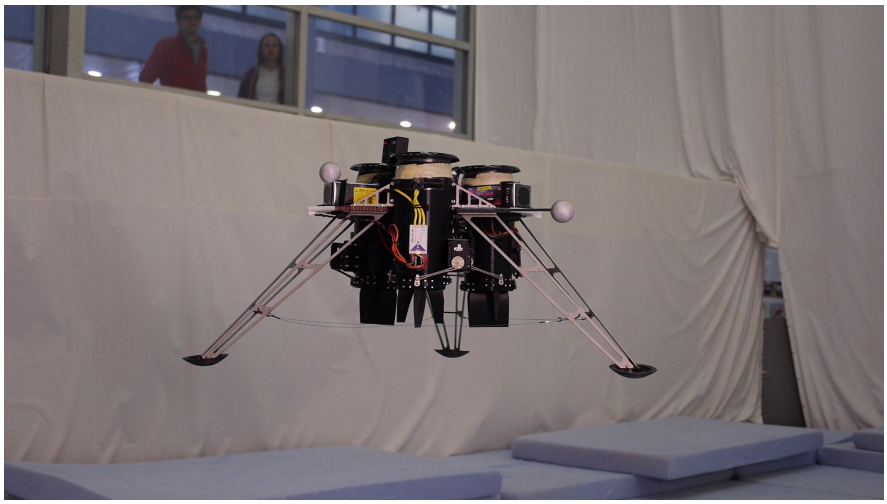
[P6] C. Sferrazza, M. Muehlebach, and R. D'Andrea, "Trajectory tracking and iterative learning on an unmanned aerial vehicle using parametrized model predictive control", in *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 2017, pp. 5186–5192

This paper describes a computationally efficient MPC framework for tracking trajectories with an unmanned aerial vehicle (shown in Fig. 2.3) that has input constraints. The state and input trajectories are parametrized with linear combinations of basis functions, with the aim to approximate an infinite-horizon optimal control problem. The generation of dynamically feasible trajectories is discussed, where reference trajectories are obtained for all states and inputs by solving an optimization problem that accounts for the desired trajectory requirements and system constraints. The basis functions used for the parametrization have finite polynomial order, and a consequently limited representation power. To address this issue, the trajectories are split over a series of smaller intervals, and shifted in a region of the state-space, chosen heuristically, where they better approximate the desired requirements. A learning strategy is included in the framework to improve the tracking performance by accounting for the unmodeled repeatable disturbances estimated over time. The approach is tested onboard a real system, running on an embedded computer, showing accurate tracking while respecting the system's constraints.

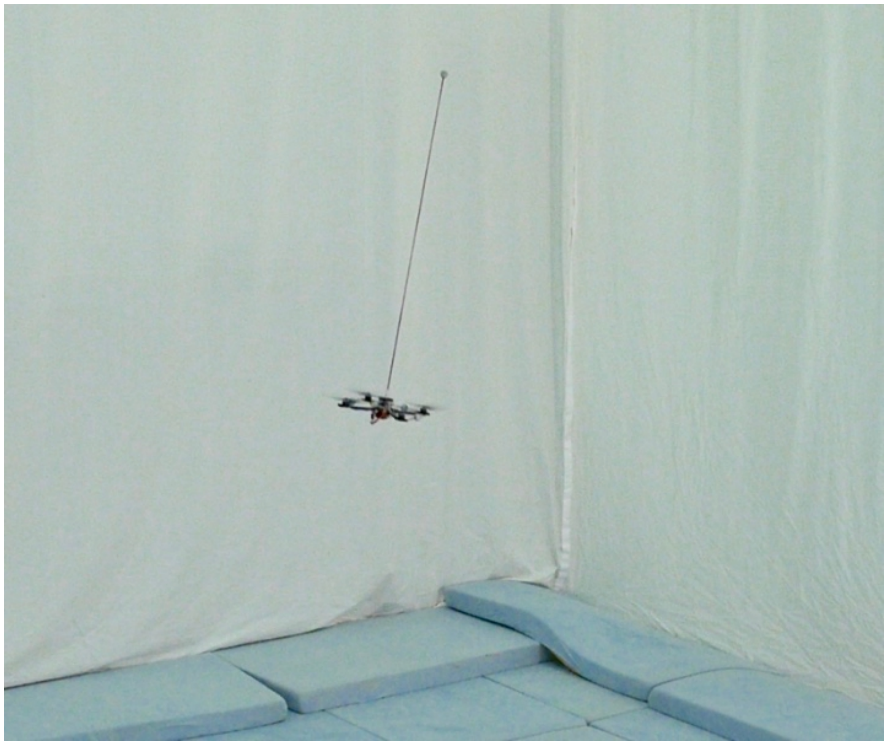
[P7] C. Sferrazza, M. Muehlebach, and R. D'Andrea, "Learning-based parametrized model predictive control for trajectory tracking", *Optimal Control Applications and Methods*, vol. 41, no. 6, pp. 2225–2249, 2020

In [P6], the generation of feasible trajectories for the parametrized MPC scheme is performed after a heuristic step, and the repeatable disturbances that are used to augment the system's model are computed in a further separate step. This creates dependencies between the different steps and makes tuning the framework challenging. In contrast, this article proposes to solve an all-in-one optimization problem, where the feasible trajectories are computed in the same step as the repeatable disturbances, providing a simpler way to trade off the accuracy of the generated reference trajectories (with respect to the desired requirements) and the quality of the estimated disturbances. The optimization problem structure is leveraged to reduce the number of optimization variables and constraints, and to exhibit faster execution times. Not only the states, but also the inputs and the disturbances are naturally expressed in a coordinate frame where the basis functions employed for the parametrization retain a higher representation power. This in particular leads to a better performance in coping with higher-order disturbances. An analysis of the space requirements for storing the trajectories shows that the framework leads to a considerably more memory-efficient solution compared to classical MPC approaches. In

the article, a real-world application of this approach to a flying inverted pendulum (see Fig. 2.3) with state and input constraints is presented, showing that the learning can quickly and effectively compensate for unmodeled effects and nonlinearities.



(a) Vehicle powered with three electric ducted fans [36]



(b) Flying inverted pendulum system

Figure 2.3 The two systems employed for the experiments presented in [P6] and [P7], respectively.

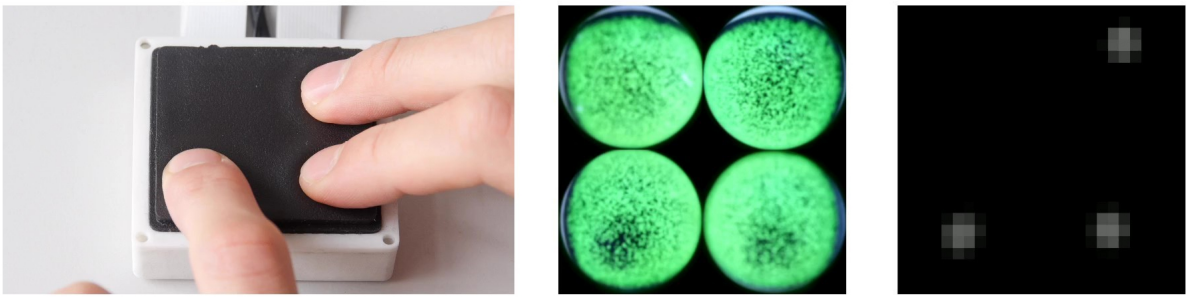
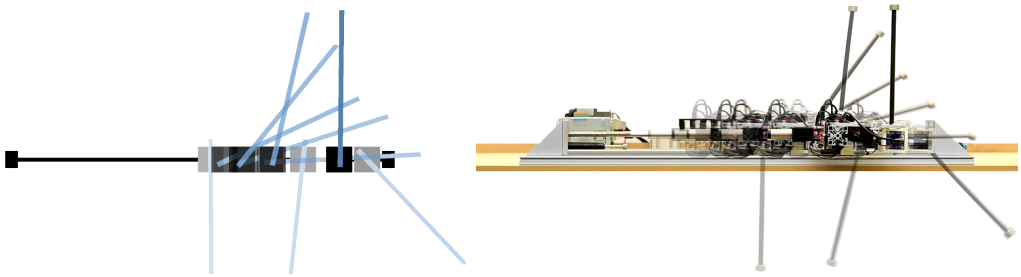


Figure 2.4 The multi-camera sensor presented in [R1] can detect distinct contact regions. In the normal force distribution depicted in the rightmost figure, brighter regions indicate higher pressure. The frame was extracted from [V5].

2.3 Appendix: Related publications

[R1] C. Trueeb, C. Sferrazza, and R. D'Andrea, "Towards vision-based robotic skins: a data-driven, multi-camera tactile sensor", in *Proceedings of the IEEE International Conference on Soft Robotics (RoboSoft)*, 2020, pp. 333–338

The bulkiness of their internal camera is considered to be the main limitation of vision-based tactile sensors. Additionally, cameras typically need to be placed at a certain distance from the sensing gel on these devices to retain sufficient focus and field of view. This paper relies on the same sensing principle as in [P1], but proposes the use of multiple small embedded cameras placed next to each other to cover a larger field of view. Equipped with close-focus lenses, the cameras are placed at only 4 mm from the sensing gel, achieving an overall sensor thickness of 17 mm and only using commodity components. With a sensing surface of 49×51 mm, the sensor is (at the time of writing) both among the thinnest and with the widest sensing surface in the camera-based category. A similar data processing strategy as in [P3] is employed, where a mixed-source training dataset is collected with a finger-shaped indenter, with 3D contact force distribution labels computed via finite element simulations. Image differences between the current frame and one taken before deformation are mapped to force distributions with a neural network architecture tailored to the use of multiple cameras, which aims to reuse the learned weights over different regions of the sensing surface. Such a learning architecture, which is implemented in real time on an embedded computer equipped with a GPU, exhibits modularity features. In fact, a scalability experiment shows promise of possibly training the core part of a data processing network on only a subset of a large surface, therefore reducing the data requirements and training times on the rest of the surface. In addition, the same training technique may also be employed when replacing a defective camera on a large tactile skin (provided that this is made possible without having to replace the gel).



(a) Simulation **(b)** Reality

Figure 2.5 In [R2], tactile control policies are learned in simulation and directly deployed to the corresponding real-world system without further adaptation. Swing-up motions in simulation and reality are depicted in (a) and (b), respectively.

[R2] T. Bi, C. Sferrazza, and R. D'Andrea, "Zero-shot sim-to-real transfer of tactile control policies for aggressive swing-up manipulation", *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5761–5768, 2021

While robotic systems equipped with tactile sensors have achieved manipulation tasks with feedback control policies that monitor contact and are able to adapt to changes in object properties [37], [38], the majority of these examples do not require a high degree of dynamicism. This article discusses an aggressive manipulation task, where a parallel-jaw gripper moved by a linear motor aims to swing poles up to a vertical position. The aim of the system is to achieve the task by solely monitoring the feedback from tactile sensors embedded at the gripper's fingers, without any external visual sensing. The task resembles a cart-pole application, but without a mechanically fixed pivot point, with the pole free to escape the grip at any time. The dome-shaped, vision-based tactile sensor employed is based on the principle presented in [P1]. The sensor outputs the three-dimensional contact force distribution based on a simulation-trained data processing framework such as that described in [P5]. To achieve the task, the control policy implicitly estimates the physical properties of the pole, which are unknown to the policy beforehand, and adjusts the gripping distance as well as the acceleration of the cart. In this article, the strategy builds upon a novel simulator of the considered system that is based on the finite element method and leverages the output of the tactile sensor, reasoning at the level of the forces and avoiding the need to simulate the optics of the internal camera. The simulator runs at 360 Hz on a single CPU core, being particularly suited for training deep reinforcement learning policies. Such a policy is trained entirely on the simulator with a privileged learning approach [39], taking the optimal actions based on an history of sensor observation. Finally, leveraging dynamics randomization [40], the policy is directly deployed on the real system without further adaptation, achieving the swing-up of poles that differ significantly in their physical attributes.

2.4 List of publications

Publications in this thesis

Publications comprising the two main parts of this thesis.

[P1] C. Sferrazza and R. D'Andrea, "Design, Motivation and Evaluation of a Full-Resolution Optical Tactile Sensor", *Sensors*, vol. 19, no. 4: 928, 2019.

[P2] C. Sferrazza and R. D'Andrea, "Transfer learning for vision-based tactile sensing", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 7961–7967.

[P3] C. Sferrazza, A. Wahlsten, C. Trueeb, and R. D'Andrea, "Ground Truth Force Distribution for Learning-Based Tactile Sensing: A Finite Element Approach", *IEEE Access*, vol. 7, pp. 173 438–173 449, 2019.

[P4] C. Sferrazza, T. Bi, and R. D'Andrea, "Learning the sense of touch in simulation: a sim-to-real strategy for vision-based tactile sensing", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[P5] C. Sferrazza and R. D'Andrea, "Sim-to-Real for High-Resolution Optical Tactile Sensing: From Images to Three-Dimensional Contact Force Distributions", *Soft Robotics*, 2021.

[P6] C. Sferrazza, M. Muehlebach, and R. D'Andrea, "Trajectory tracking and iterative learning on an unmanned aerial vehicle using parametrized model predictive control", in *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 2017, pp. 5186–5192.

[P7] C. Sferrazza, M. Muehlebach, and R. D'Andrea, "Learning-based parametrized model predictive control for trajectory tracking", *Optimal Control Applications and Methods*, vol. 41, no. 6, pp. 2225–2249, 2020.

Related publications

Publications related to Part A of this thesis, and included in the appendix.

[R1] C. Trueeb, C. Sferrazza, and R. D'Andrea, "Towards vision-based robotic skins: a data-driven, multi-camera tactile sensor", in *Proceedings of the IEEE International Conference on Soft Robotics (RoboSoft)*, 2020, pp. 333–338. **Best Paper Award Winner**.

[R2] T. Bi, C. Sferrazza, and R. D'Andrea, "Zero-shot sim-to-real transfer of tactile control policies for aggressive swing-up manipulation", *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5761–5768, 2021.

Additional publications

Publications that are not directly related to those included in this thesis. They describe the application of data-driven, vision-based sensing to a pneumatic actuator, by placing an internal camera to track the elongation state while controlling a soft robot arm.

[Ad1] P. Werner, M. Hofer, C. Sferrazza and R. D'Andrea, "Vision-Based Proprioceptive Sensing: Tip Position Estimation for a Soft Inflatable Bellow Actuator", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 8889–8896.

[Ad2] M. Hofer, C. Sferrazza and R. D'Andrea, "A Vision-based Sensing Approach for a Spherical Soft Robotic Arm", *Frontiers in Robotics and AI*, vol. 8, 2021.

2.5 Student supervision

Several students were supervised as part of the author's doctoral studies. Below is a complete list of student projects, sorted by the type of project performed.

Master's thesis

The master's thesis is a six-month, full-time research project.

[MT1] Pietro Griffa, "Tactile-Enabled Robotic Grasping", 2021.

[MT2] Thomas Bi, "Learning Aggressive Tactile Swing-Up Maneuvers", 2020. **Awarded ETH Medal and Willi Studer Prize for the Best Master's Degree**.

[MT3] Camill Trueeb, "An End-to-End Approach to Multi-Camera Tactile Sensing", 2019. **ETEL Master Award Winner**.

[MT4] Yipai Du, "Learning Dynamical Features for Vision-based Tactile Sensors", Exchange student from KTH Stockholm, 2019.

[MT5] Benita Nortmann, "A State-Dependent Approach to Learning-Based Model Predictive Control", Exchange student from Imperial College London, 2019.

Master semester project

The master semester project is a semester-long, part-time research project.

[SP1] Peter Werner, "Trajectory Generation for Tactile-Enabled Robotic Manipulation", 2021.

[SP2] Felix Schmitt-Koopmann, "Few-Shot Learning for a Tactile Sensor", 2021.

[SP3] Thomas Bi, "Generation of Optical Flow from Finite Element Simulations: Applications to Tactile Sensing", 2020.

[SP4] Camill Trueeb, "Automated FEA Simulations to Provide Ground Truth for Tactile Sensing", 2018.

[SP5] Laura Maria Gasser, "Feature Engineering for an Optical Tactile Sensor", 2018.

[SP6] Zhejun Zhang, "Improving the Trajectory Tracking of a Parametrized MPC Approach", Jointly supervised with Michael Muehlebach, 2017.

Bachelor's thesis

The bachelor's thesis is a three-month, full-time research project.

[BT1] Peter Werner, "Time of Flight and Camera Based Sensing for an Air-Driven Linear Soft Actuator", Jointly supervised with Matthias Hofer, 2019. **Awarded SGA (Swiss Society for Automatic Control) prize**.

Internship

Internships are full-time practical projects lasting one to three months.

[In1] Anil Parsi, "Improving the design of a vision-based tactile sensor", 2018.

[In2] Xueying Xie, "Data synchronization for tactile sensing applications", 2018.

2.6 Outreach

Talks

Note that the talks at scientific conferences corresponding to the publications [P2], [P4], [P6], [R1] are not listed.

Academic:

Sep. 2021	*Haptic Intelligence Department* (Max Planck Institute for Intelligent Systems).
July 2021	*Autonomy Talk* (ETH Zurich), `https://youtu.be/gFfN3U95GyM`.
July 2021	*Coffee Talk, Automatic Control Laboratory* (ETH Zurich).
June 2021	*Robotics Institute Seminar* (University of Toronto), `https://youtu.be/xGIseD5tvYk`.
June 2021	*Seminar, Automation Lab* (UC Berkeley).
June 2021	*Mechanical and Aerospace Engineering Seminar* (UCLA).
May 2021	*Seminar, Institute for Data Science in Mechanical Engineering* (RWTH Aachen University).
Mar. 2020	*Autonomy Talk* (ETH Zurich).

(a) WORLD.MINDS **(b)** TEDxZurich (live demo) **(c)** TEDxZurich

Figure 2.6 The tactile sensor was presented and demonstrated at several venues. (a) shows a snapshot of the talk at the 2019 WORLD.MINDS Annual Symposium, in front of a live audience of 300 people. The YouTube recording of the TEDxZurich talk (see (b) and (c)), featuring a live demonstration, counts more than 40,000 views.

General public:

Nov. 2020	*TEDxZurich* (Zurich), `https://youtu.be/IXKovDtgD_8`.
Dec. 2019	*WORLD.MINDS Annual Symposium* (Zurich), `https://youtu.be/ece2Fl6a5fY`.
May 2019	*AI Night, House of Electronic Arts* (Basel).

Demonstrations

During the period of this thesis, the author was involved in demonstrating the vision-based tactile sensor and other research projects of the Institute for Dynamic Systems and Control at general public events.

Sep. 2020	Winterthur	Swiss Science Center Technorama
Jan. 2019	Davos	World Economic Forum

In addition to the above, demonstrations were also conducted during various talks and for lab visitors (ranging from primary school students to distinguished professors).

Blog posts and online articles

The following general audience articles were published on online communication platforms.

[On1] C. Sferrazza, "Robots that feel by seeing", *Robohub*, 2021.

[On2] C. Sferrazza, "The significance of (online) public talks", *ETH Ambassadors*, 2020.

Videos

The following videos were created as an addition to research articles and for consumption by the general public, demonstrating some of the research results.

[V1] P. Griffa, C. Sferrazza, and R. D'Andrea, *Leveraging distributed contact force measurements for slip detection: a physics-based approach*, Sep. 2021. [Online]. Available: `https://youtu.be/YeotGbKVWcY`.

[V2] C. Sferrazza and R. D'Andrea, *Sim2real for high-resolution optical tactile sensing: From images to 3D contact force distributions*, Sep. 2021. [Online]. Available: `https://youtu.be/dvOk2XrSmLE`.

[V3] T. Bi, C. Sferrazza, and R. D'Andrea, *Zero-shot sim-to-real transfer of tactile control policies for aggressive swing-up manipulation*, Apr. 2021. [Online]. Available: `https://youtu.be/In4jkaHzJLc`.

[V4] C. Sferrazza, T. Bi, and R. D'Andrea, *Learning the sense of touch in simulation: a sim-to-real strategy for vision-based tactile sensing*, Mar. 2020. [Online]. Available: `https://youtu.be/dDTga9PgWS0`.

[V5] C. Trueeb, C. Sferrazza, and R. D'Andrea, *Towards vision-based robotic skins: a data-driven, multi-camera tactile sensor*, Oct. 2019. [Online]. Available: `https://youtu.be/lbavqAlKl98`.

[V6] C. Sferrazza, A. Wahlsten, C. Trueeb, and R. D'Andrea, *Ground truth force distribution for learning-based tactile sensing: a finite element approach*, Sep. 2019. [Online]. Available: `https://youtu.be/9A-cONrsiOg`.

[V7] C. Sferrazza and R. D'Andrea, *Transfer learning for vision-based tactile sensing*, Mar. 2019. [Online]. Available: `https://youtu.be/CdYK5I6Sccw`.

[V8] C. Sferrazza and R. D'Andrea, *Design, motivation and evaluation of a full-resolution optical tactile sensor*, Feb. 2019. [Online]. Available: `https://www.mdpi.com/1424-8220/19/4/928/s1`.

[V9] C. Sferrazza, M. Muehlebach, and R. D'Andrea, *Learning based parametrized model predictive control for trajectory tracking*, Oct. 2018. [Online]. Available: `https://youtu.be/-E4znjVDCyA`.

Selected media coverage

The tactile sensing research was featured in various international news media.

- *Machine learning helps researchers build low-cost tactile sensor*, The Robot Report, March 2020.
- *Allowing robots to feel*, ETH News, March 2020.
- *Sensor skin could give robot grippers a delicate touch*, New Atlas, March 2020.
- *A deep learning-based method for vision-based tactile sensing*, Tech Xplore, March 2020.
- *Robotic Skin Sees When (and How) You're Touching It*, Hackaday, November 2019.
- *Tactile sensor could enable soft robot skins*, Fierce Electronics, November 2019.

- *A multi-camera optical tactile sensor that could enable vision-based robotic skins*, Tech Xplore, November 2019.
- *The Best Machine Learning Research of 2019 So Far*, Open Data Science on Medium.com, June 2019.

3

Future work

This chapter provides an overview of potential future work based on the research presented in this thesis.

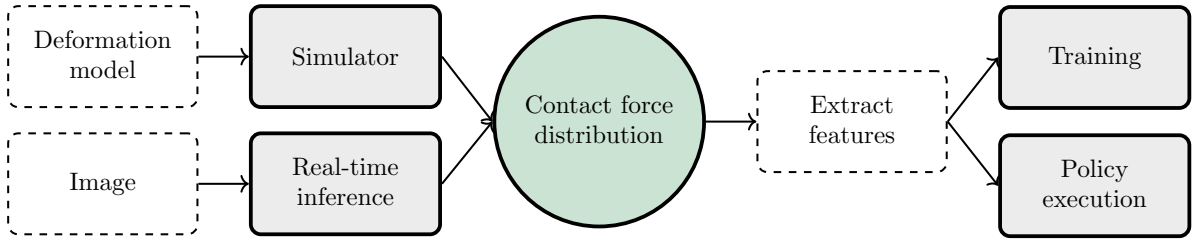
Data-driven, vision-based tactile sensing

The data processing technique described in [P5] is efficient, versatile and accurate. The strategy is based on building a very accurate offline simulator where the pipeline is trained, and a seamless sim-to-real transfer is then achieved without further training. It fully leverages the sensing principle, which is well suited to be simulated, since its pattern's randomness naturally requires processing algorithms that are robust to pattern variations, as those introduced when generating synthetic images. However, unmodeled effects still contribute to an observable sim-to-real gap, which should be addressed to further improve the sensing accuracy. In this regard, strategies that explicitly aim to address distributional shifts in the data from simulation and reality may be investigated to steer the training in a direction that filters out the differences and closes the gap between these two worlds. Such a strategy would further leverage the estimation of the force distribution in providing an abstraction that bypasses mismatches between simulated and real images.

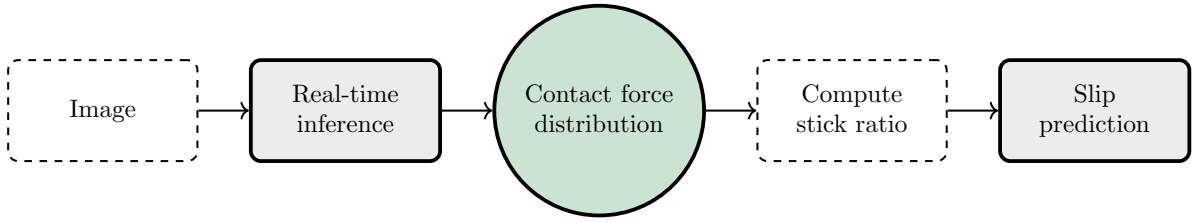
Tactile estimation and control for dexterous manipulation

Systematically condensing high-resolution tactile information into a state representative of the system considered is a very relevant but relatively unexplored field. In [R2], the control strategy pivots on the knowledge that pole orientation and total force information are essential to perform the swing-up task. Therefore, such quantities are extracted from the force distribution and used as inputs to the control policy. However, for generic manipulation tasks, it may be more challenging to identify such key information beforehand. Therefore, strategies based on autoencoders and representation learning [41] may be investigated to automatically extract such features from the force distribution readings depending on the task of interest.

Alternatively, simpler learning strategies for the swing-up task may be developed if an estimator was able to accurately characterize the pole and its pose from the real-world measurements. Future work in this direction will focus on the design of Bayesian filters to estimate the full state of the pole, by incorporating the torque, angular and position information encoded in the force distribution readings. On generic manipulation tasks,



(a) Swing-up pipeline



(b) Slip prediction pipeline

Figure 3.1 The figure shows two examples where the abstraction provided by the force distribution facilitates the development of higher-level tasks. The swing-up task discussed in [R2] is shown in (a), where the top row represents the classical scenario during training. Through a simplified deformation model, the simulator computes the force distributions at each step; total force and angular information are extracted from the force distribution and employed for the training of the policy that will then be employed in the real world. In the bottom row of (a), the real-world pipeline is shown: the force distribution is predicted from the current image, the same features as in simulation are then extracted and fed as inputs to the policy that computes the optimal action to achieve the task. For such an application, the force distribution abstraction enables the simulator to reason at the level of the forces, rendering a technique that is independent of the underlying sensing principle, and avoiding the need to simulate the camera optics, which leads to a considerable improvement in computational efficiency. In (b), the slip prediction pipeline presented in [42] is outlined. The mentioned abstraction makes it possible to neglect the deformation model and directly enables the development of the pipeline by computing the stick ratio from the force distribution. This is then thresholded to provide a straightforward rule for detecting slippage.

pose estimation may further be facilitated by the presence of distinct features, which are instead absent on the smooth surface of the poles.

Slip detection and grasp adjustment based on distributed force measurements

Humans are able to maintain a firm grasp of objects by constantly monitoring slippage and refining the necessary gripping force without damaging such objects. Similarly, tactile sensors promise to estimate slippage during robotic manipulation tasks to achieve safe and reliable operations. The three-dimensional force fields provided by the sensor discussed in this thesis enable the estimation of the stick ratio, defined as the ratio between the sticking and the slipping regions of the sensing surface in contact with an external object. This ratio provides an indication of incipient slip, which can be employed to anticipate the actual slippage of the object. In the recent work [42] originating from [MT1], the stick ratio was computed by locally applying the Coulomb friction law at different locations over the sensing surface. This led to a considerably improved performance in slip prediction

compared to globally applying the Coulomb friction law, especially due to the ability of coping with rotational slippage. Such results may be employed to develop strategies that adjust the gripping force in real-time, or perform a regrasp based on the available slippage information.

Note that the force distribution readings are fully leveraged in this approach. Specifically, this application shows a further benefit compared to those achieved in the swing-up task in [R2], where the developed simulator leverages the sensing output to bypass the need to model the camera projection. In fact, for the slip detection task [42], the force distribution provides a means to additionally bypass the deformation of the sensor, purely reasoning on the estimated force field (see Fig. 3.1).

Large-scale fabrication

To facilitate the widespread adoption of vision-based tactile sensors outside of research labs, the fabrication of these devices on a large scale should be addressed. This is a particularly unexplored area, especially when it comes to sensors that cover large and complex surfaces. However, the rapid evolution of camera technology, which has led to unprecedented miniaturization results, and the development of novel soft material fabrication techniques [43] open new avenues for exploration in this field. On the data processing side, approaches tailored to process incoming data from larger surfaces through the use of multiple cameras will be necessary, such as that presented as a proof-of-concept in [R1].

References for Chapters 1-3

[1] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-End Training of Deep Visuomotor Policies", *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[2] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies", *Science Robotics*, vol. 4, no. 26, 2019.

[3] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic Grasping of Novel Objects using Vision", *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 157–173, 2008.

[4] M. R. Tremblay and M. R. Cutkosky, "Estimating friction using incipient slip sensing during a manipulation task", in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1993, pp. 429–434.

[5] A. Billard and D. Kragic, "Trends and challenges in robot manipulation", *Science*, vol. 364, no. 6446, eaat8414, 2019.

[6] H. Dang, J. Weisz, and P. K. Allen, "Blind grasping: Stable robotic grasping using tactile feedback and hand kinematics", in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 5917–5922.

[7] M. H. Lee, "Tactile Sensing: New Directions, New Challenges", *The International Journal of Robotics Research*, vol. 19, no. 7, pp. 636–643, 2000.

[8] N. Heravi, W. Yuan, A. M. Okamura, and J. Bohg, "Learning an Action - Conditional Model for Haptic Texture Generation", in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 11 088–11 095.

[9] Y. Wu, Y. Liu, Y. Zhou, Q. Man, C. Hu, W. Asghar, F. Li, Z. Yu, J. Shang, G. Liu, *et al.*, "A skin-inspired tactile sensor for smart prosthetics", *Science Robotics*, vol. 3, no. 22, 2018.

[10] M. Ohka, Y. Mitsuya, K. Hattori, and I. Higashioka, "Data conversion capability of optical tactile sensor featuring an array of pyramidal projections", in *Proceedings of the IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 1996, pp. 573–580.

[11] N. J. Ferrier and R. W. Brockett, "Reconstructing the Shape of a Deformable Membrane from Image Data", *The International Journal of Robotics Research*, vol. 19, no. 9, pp. 795–816, 2000.

[12] K. Kamiyama, H. Kajimoto, N. Kawakami, and S. Tachi, "Evaluation of a vision-based tactile sensor", in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, 2004, pp. 1542–1547.

[13] K. Shimonomura, "Tactile Image Sensors Employing Camera: A Review", *Sensors*, vol. 19, no. 18: 3933, 2019.

[14] Y.-L. Park, C. Majidi, R. Kramer, P. Bérard, and R. J. Wood, "Hyperelastic pressure sensing with a liquid-embedded elastomer", *Journal of Micromechanics and Microengineering*, vol. 20, no. 12: 125029, 2010.

[15] H. B. Muhammad, C. M. Oddo, L. Beccai, C. Recchiuto, C. J. Anthony, M. J. Adams, M. C. Carrozza, D. W. L. Hukins, and M. C. L. Ward, "Development of a bioinspired MEMS based capacitive tactile sensor for a robotic finger", *Sensors and Actuators A: Physical*, vol. 165, no. 2, pp. 221–229, 2011.

[16] J. A. Fishel and G. E. Loeb, "Sensing tactile microvibrations with the BioTac – Comparison with human sensitivity", in *Proceedings of the IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, 2012, pp. 1122–1127.

[17] J.-S. Heo, J.-H. Chung, and J.-J. Lee, "Tactile sensor arrays using fiber Bragg grating sensors", *Sensors and Actuators A: Physical*, vol. 126, no. 2, pp. 312–327, 2006.

[18] L. Zou, C. Ge, Z. J. Wang, E. Cretu, and X. Li, "Novel Tactile Sensor Technology and Smart Tactile Sensing Systems: A Review", *Sensors*, vol. 17, no. 11: 2653, 2017.

[19] W. Yuan, R. Li, M. A. Srinivasan, and E. H. Adelson, "Measurement of shear and slip with a GelSight tactile sensor", in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 304–311.

[20] A. Yamaguchi and C. G. Atkeson, "Combining finger vision and optical tactile sensing: Reducing and handling errors while cutting vegetables", in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2016, pp. 1045–1051.

[21] B. Ward-Cherrier, N. Pestell, L. Cramphorn, B. Winstone, M. E. Giannaccini, J. Rossiter, and N. F. Lepora, "The TacTip Family: Soft Optical Tactile Sensors with 3D-Printed Biomimetic Morphologies", *Soft Robotics*, vol. 5, no. 2, pp. 216–227, 2018.

[22] W. Yuan, S. Dong, and E. H. Adelson, "GelSight: High-Resolution Robot Tactile Sensors for Estimating Geometry and Force", *Sensors*, vol. 17, no. 12: 2762, 2017.

[23] N. F. Lepora and B. Ward-Cherrier, "Superresolution with an optical tactile sensor", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 2686–2691.

[24] S. Dong, W. Yuan, and E. H. Adelson, "Improved GelSight tactile sensor for measuring geometry and slip", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 137–144.

[25] D. Ma, E. Donlon, S. Dong, and A. Rodriguez, "Dense Tactile Force Estimation using GelSlim and inverse FEM", in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5418–5424.

[26] N. Kuppuswamy, A. Castro, C. Phillips-Grafflin, A. Alspach, and R. Tedrake, "Fast Model-Based Contact Patch and Pose Estimation for Highly Deformable Dense-Geometry Tactile Sensors", *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1811–1818, 2019.

[27] Y. S. Narang, K. Van Wyk, A. Mousavian, and D. Fox, "Interpreting and Predicting Tactile Signals via a Physics-Based and Data-Driven Framework", in *Proceedings of the Robotics: Science and Systems Conference (RSS)*, 2020.

[28] H. Lee, H. Park, G. Serhat, H. Sun, and K. J. Kuchenbecker, "Calibrating a Soft ERT-Based Tactile Sensor with a Multiphysics Model and Sim-to-real Transfer Learning", in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 1632–1638.

[29] Y. She, S. Wang, S. Dong, N. Sunil, A. Rodriguez, and E. Adelson, "Cable Manipulation with a Tactile-Reactive Gripper", 2020.

[30] C. Wang, S. Wang, B. Romero, F. Veiga, and E. Adelson, "SwingBot: Learning Physical Features from In-hand Tactile Exploration for Dynamic Swing-up Manipulation", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5633–5640.

[31] N. F. Lepora, A. Church, C. De Kerckhove, R. Hadsell, and J. Lloyd, "From pixels to Percepts: Highly Robust Edge Perception and Contour Following using Deep Learning and an Optical Biomimetic Tactile Sensor", *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2101–2107, 2019.

[32] M. Morari and J. H. Lee, "Model predictive control: past, present and future", *Computers & Chemical Engineering*, vol. 23, no. 4, pp. 667–682, 1999.

[33] M. L. Darby and M. Nikolaou, "MPC: Current practice and challenges", *Control Engineering Practice*, vol. 20, no. 4, pp. 328–342, 2012.

[34] M. Alamir and G. Bornard, "Stability of a truncated infinite constrained receding horizon scheme: the general discrete nonlinear case", *Automatica*, vol. 31, no. 9, pp. 1353–1356, 1995.

[35] M. Muehlebach and R. D'Andrea, "Parametrized infinite-horizon model predictive control for linear time-invariant systems with input and state constraints", in *Proceedings of the American Control Conference (ACC)*, IEEE, 2016, pp. 2669–2674.

[36] M. Muehlebach and R. D'Andrea, "The Flying Platform - A testbed for ducted fan actuation and control design", *Mechatronics*, vol. 42, pp. 52–68, 2017.

[37] H. Yousef, M. Boukallel, and K. Althoefer, "Tactile sensing for dexterous in-hand manipulation in robotics – A review", *Sensors and Actuators A: Physical*, vol. 167, no. 2, pp. 171–187, 2011.

[38] Z. Kappassov, J.-A. Corrales, and V. Perdereau, "Tactile sensing in dexterous robot hands – Review", *Robotics and Autonomous Systems*, vol. 74, pp. 195–220, 2015.

[39] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by Cheating", in *Proceedings of the Conference on Robot Learning (CoRL)*, 2020, pp. 66–75.

[40] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization", in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1–8.

[41] M. Tschannen, O. Bachem, and M. Lucic, "Recent advances in autoencoder-based representation learning", *Bayesian Deep Learning Workshop at NeurIPS*, 2018.

[42] P. Griffa, C. Sferrazza, and R. D'Andrea, "Leveraging distributed contact force measurements for slip detection: a physics-based approach enabled by a data-driven tactile sensor", *arXiv preprint arXiv:2109.11504*, 2021.

[43] M. A. Bell, K. P. Becker, and R. J. Wood, "Injection Molding of Soft Robots", *Advanced Materials Technologies*, p. 2 100 605, 2021.

Part A

Data-driven, vision-based tactile sensing

consisting of publications

[P1] C. Sferrazza and R. D'Andrea, "Design, Motivation and Evaluation of a Full-Resolution Optical Tactile Sensor", *Sensors*, vol. 19, no. 4: 928, 2019

[P2] C. Sferrazza and R. D'Andrea, "Transfer learning for vision-based tactile sensing", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 7961–7967

[P3] C. Sferrazza, A. Wahlsten, C. Trueeb, and R. D'Andrea, "Ground Truth Force Distribution for Learning-Based Tactile Sensing: A Finite Element Approach", *IEEE Access*, vol. 7, pp. 173 438–173 449, 2019

[P4] C. Sferrazza, T. Bi, and R. D'Andrea, "Learning the sense of touch in simulation: a sim-to-real strategy for vision-based tactile sensing", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020

[P5] C. Sferrazza and R. D'Andrea, "Sim-to-Real for High-Resolution Optical Tactile Sensing: From Images to Three-Dimensional Contact Force Distributions", *Soft Robotics*, 2021

Paper P1

Design, motivation and evaluation of a full-resolution optical tactile sensor

Carmelo Sferrazza and Raffaello D'Andrea

Abstract

Human skin is capable of sensing various types of forces with high resolution and accuracy. The development of an artificial sense of touch needs to address these properties, while retaining scalability to large surfaces with arbitrary shapes. The vision-based tactile sensor proposed in this article exploits the extremely high resolution of modern image sensors to reconstruct the normal force distribution applied to a soft material, whose deformation is observed on the camera images. By embedding a random pattern within the material, the full resolution of the camera can be exploited. The design and the motivation of the proposed approach are discussed with respect to a simplified elasticity model. An artificial deep neural network is trained on experimental data to perform the tactile sensing task with high accuracy for a specific indenter, and with a spatial resolution and a sensing range comparable to the human fingertip.

Published in *Sensors*.

1. Introduction

Relatively small image sensors nowadays provide very high resolutions and wide dynamic ranges. Together with cost effectiveness, these factors have made cameras a comprehensive solution for providing robots with a sense of vision. Moreover, in particular in recent years, the use of machine learning for computer vision problems has contributed to the accomplishment of numerous challenging tasks in the field (see, for example, [1]–[3]).

The benefits of artificial vision systems, from the points of view of both hardware and developed algorithms, can be exploited in different domains, as is the case for robotic tactile sensing systems. Although the fundamental importance of the sense of touch for interacting with the environment has been shown in both humans (see [4]), and robots (see [5]), finding a sensing solution that yields satisfactory performance for various types of interactions and tasks is still an open problem. By using a camera to monitor various physical properties, such as the change in light intensity, which are related to the deformation of a soft material subject to external forces, it is possible to design algorithms which reconstruct the force distribution with high resolution.

This article describes the design of a sensor (shown in Figure 1) that consists of a camera that tracks the movement of spherical markers within a gel, providing an approximation of the strain field inside the material. This information is exploited to reconstruct the normal external force distribution that acts on the surface of the gel. The use of a camera intrinsically renders a very high spatial resolution, given by the extensive number of pixels composing the images. A specific choice of relevant features leverages the full resolution of the camera, with the sensor's spatial resolution not limited by the number of markers.

Moreover, this article provides a theoretical analysis of the elastic model of the material, which evaluates different marker layouts. This analysis indicates that the presence of markers at different depths within the gel yields a higher robustness to errors in the marker tracking, while retaining a small sensor threshold.

The map between the marker displacements and the normal force distribution is modeled with a neural network, which is trained on a vast number of images. These images are collected while an automatically controlled milling machine presses an indenter's tip against the sensor's surface at different locations. During this procedure, ground truth force measurements are provided by a force torque (F/T) sensor. The proposed approach is also discussed with respect to transfer learning applications in the author's work in [6], where preliminary results show that it is possible to greatly reduce the required training data and the training times.

The resulting pipeline runs in real-time on a standard laptop (dual-core, 2.80 GHz), predicting the normal force distribution from the image stream at 60 Hz.

1.1 Related work

As highlighted in [7], among the reasons that have delayed the widespread deployment of tactile sensors on robotic systems compared to vision sensors, the lack of a tactile analog to optical arrays is a result of the inherent complexity of interpreting the information

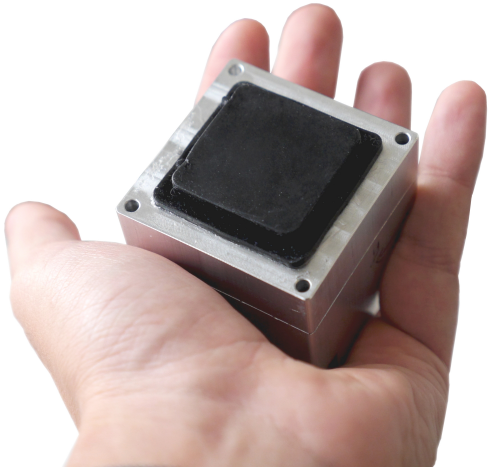


Figure 1. The tactile sensor presented in this article.

obtained via physical contact.

In the literature, different categories of tactile sensors focus on obtaining this information using different principles (see, for example, [8], where the electrical resistance of an elastomer is related to the pressure exerted on it, and [9], where an array of zinc oxide nanorods generates a voltage signal whose amplitude is proportional to the normal force applied). A detailed description of the main classes of tactile sensors for robotic applications is provided in [10].

Optical (or vision-based) tactile sensors are a large class of devices that exploit various light-related principles, which describe how different properties change with the stress applied to a contact surface. Several examples are described in the literature, based on principles such as photometric stereo [11], total internal reflection [12], and reflected light intensity [13]. Vision-based tactile sensors only marginally affect the observed sensor's surface, and therefore do not alter the softness of the contact area. This is a main requirement for tactile sensors that interact with the environment [7]. In fact, a soft material has the advantage of compliance, friction and conformation to the surfaces it interacts with, which for example are crucial properties for manipulation tasks.

Several approaches proposed for optical tactile sensing are based on tracking a series of markers on the sensor's surface (see [14]). The movement of these markers is directly related to the strain field of the material, and can therefore be used to reconstruct the external force distribution on the surface. For example, in [15], a numerical method which captures a linear elastic model of the material is used to reconstruct the force distribution from marker displacements.

The choice of the marker layout in vision-based tactile sensors is investigated in various works. A symmetrical pattern is exploited in [16] to generalize tactile stimuli to new orientations. In [17], an analytical model approximately reconstructs the force distribution from the displacement of spherical markers. These markers are placed over two different depth layers within the sensor's surface, and this layout shows a better robustness to

noise compared to the case in which all markers are placed at the same depth.

To overcome the limitations imposed by the assumptions made on the material model, whose full nonlinear description is highly complex to derive, machine learning approaches have been explored for tactile sensing. By training a learning algorithm with an extensive amount of data, it is possible to reconstruct the force applied to the sensor's surface. For example, in [18], a deep neural network is used to estimate the total contact force that an optical tactile sensor with markers printed on the surface is exerting on several objects. The need for training data is fulfilled in different ways in the literature, for instance with robot manipulators [5] or with other automatic machines [19].

This article presents an optical tactile sensor that is based on tracking spherical markers, which are randomly spread over the entire three-dimensional volume of its soft surface. The sensor presented here does not rely on tracking a specific marker layout (as in [16], [17]) and is provided with standard lighting conditions (opposed to [18]), facilitating the deployment to larger surfaces (through multiple cameras) of arbitrary shapes. In fact, the proposed strategy only requires the presence of distinct features, which move according to the material deformation, therefore greatly simplifying manufacture.

Conversely to most of the cited approaches, the features engineered for the proposed supervised learning algorithm can be chosen in a way that the information at all pixels is processed, independent of the pattern choice, therefore actually exploiting the full resolution of the camera. Moreover, the proposed design and strategy can be easily extended to different types of forces and interactions.

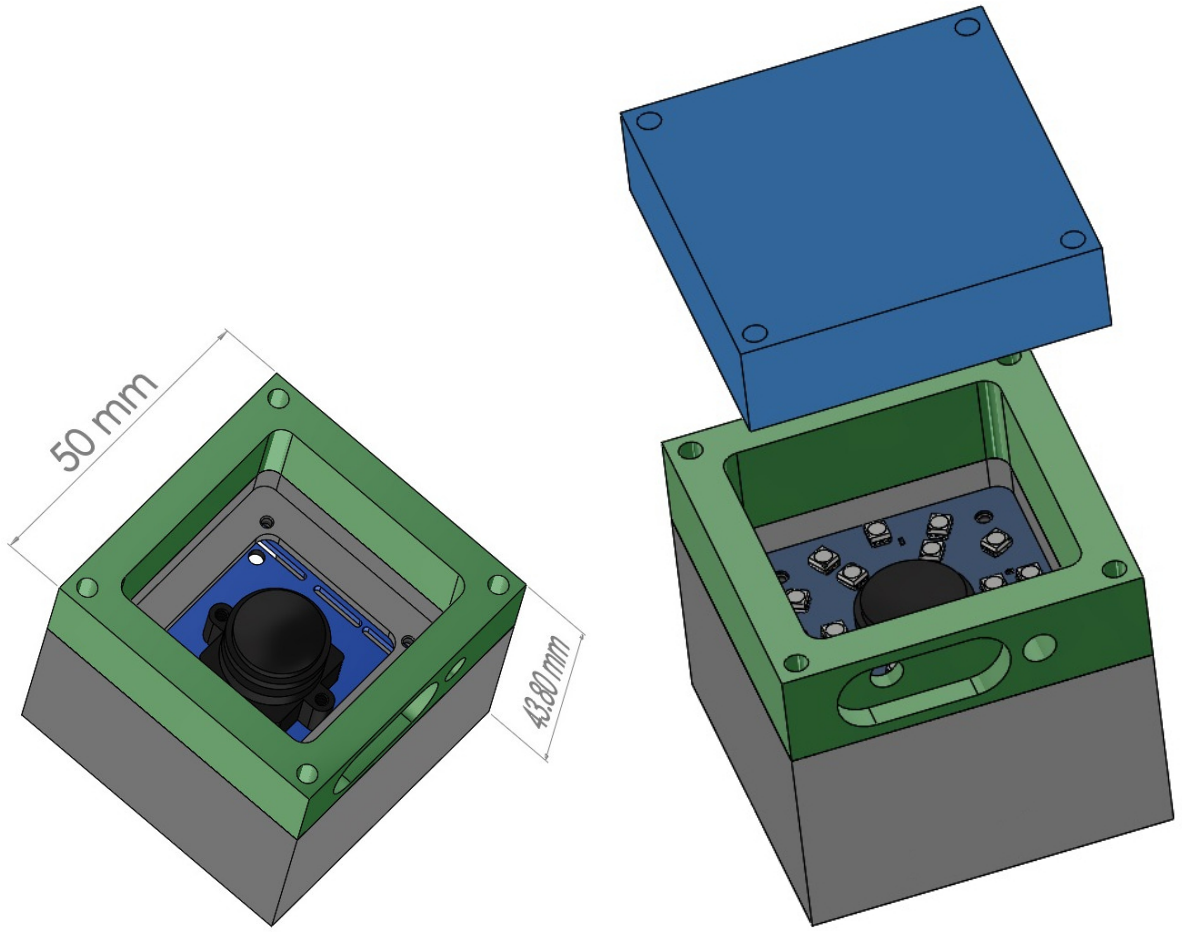
1.2 Outline

The tactile sensor is presented in Section 2, where its design and the production procedure are discussed. Theoretical analyses of the sensor threshold and robustness are explained in Section 3. Section 4 describes a procedure for training data collection and the related data processing for the generation of ground truth labels. Two feature engineering strategies, both suitable for real-time execution, are presented in Section 5. The learning architecture, which captures a map between the engineered features and the generated ground truth labels, is discussed in Section 6. Results and comparisons are shown in Section 7. Finally, Section 8 draws the conclusions of the article.

2. Design and production

To track the spherical markers within the soft material, a fisheye RGB camera (2.24 megapixels ELP USBFHD06H) was placed inside a mold, as shown in Figure 2a, and equipped with a board that controls two rings of LEDs (see Figure 2b), which provide constant illumination. The camera can read image frames up to 100 fps at a resolution of 640×480 pixels.

The soft materials employed in the sensor production were all degassed in a vacuum (to remove air bubbles, which form during mixing) and poured through cavities in the mold.



(a) Camera attached to the mold

(b) Parts for casting the stiff layer

(c) Second lid

(d) Third lid

Figure 2. CAD drawings showing the different parts of the mold: (**a**) the camera was attached to the mold, and (**b**) the LED board was placed around the lens. Three lids (blue (**b**); red (**c**); and yellow (**d**)) were interchanged for the various production steps. The colors in this figure are only for ease of visualization, and do not correspond to the colors of the real tactile sensor.

The mold was placed on one of its sides (that is, with the camera lens pointing sideways) during production. Three different lids were used to perform the three respective steps of the production procedure, which is explained in the following. The design proposed here differs from the one employed in [6], where a bottom-up procedure yielded imperfections on the sensor's top surface.

A first lid closes the mold, and a relatively stiff transparent silicone layer (ELASTOSIL® RT 601 RTV-2, mixing ratio 7:1, shore hardness 45A) was then poured into the mold through a side cavity, which is shown in Figure 2b. After the silicone cured, the lid was removed and replaced with the second one, shown (in red) in Figure 2c, which had an indent of 30 × 30 × 4.5 mm. Spherical fluorescent green particles (shown in Figure 3) with a diameter of 500–600 μm were mixed with a very soft silicone gel (Ecoflex™ GEL, mixing ratio 1:1, shore hardness 000-35) and poured into the mold filling the empty indent. After this soft layer was also cured, the second lid was replaced with the third one, shown (in yellow) in Figure 2d, with an indent that left an empty section around the gel of varying thickness (depending on the section) between 1 mm and 1.5 mm. A black silicone layer (ELASTOSIL® RT 601 RTV-2, mixing ratio 25:1, shore hardness 10A) was then poured through the cavity on the third lid. Figure 4 shows a schematic cross-sectional view of the three silicone layers, and an example of the resulting tactile sensor is shown in Figure 1.

The stiff layer, which was poured first, served as a base for the softer materials that were placed on top of it, and as a spacer between the camera and the region of interest. All the materials employed have comparable refraction indexes, therefore preventing unwanted reflections from the LEDs. The spherical particles have a density that is close to the density of the gel they are mixed with. Together with the viscosity of the material, this is crucial to obtain a homogeneous spread of the markers over the entire depth of the gel. A detailed discussion on this point can be found in Appendix A. The black silicone layer added consistency to the extremely soft gel, which also tended to stick on contact, and provided a shield against external light disturbances. The thickness of the sensor from the base of the camera to the top of the surface is 37 mm.

3. Motivation

Besides ease of manufacture and portability to surfaces with arbitrary shapes, the approach presented here showed theoretical benefits in a simplified scenario. This section presents first order analyses of two different properties of the proposed sensor: the robustness to noise in the marker displacements and the sensor threshold. Here, the threshold is defined as the minimum force that leads to a detectable change in the camera image. To this purpose, simplified models of the camera and the material were considered. Since the resulting expressions were dependent on the marker distribution and on the displacements observed, Monte Carlo simulations (i.e., based on repeated random sampling) were performed for different external forces and marker layouts, and the results are discussed. In particular, four layout classes were considered:

Figure 3. A concentration of the spherical fluorescent green markers, compared to the size of a regular pen cap.

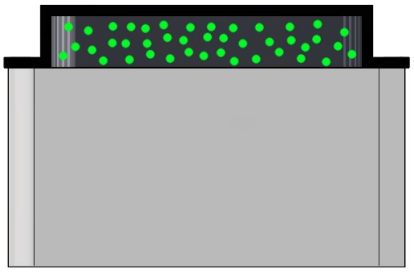


Figure 4. Schematic cross-sectional view of the resulting soft materials, shown here on top of each other. For ease of visualization, the base stiff layer is shown in semi-transparent light gray. The green particles were embedded in a soft layer, which was covered by an additional black layer.

Single layer at 1 mm: The markers were randomly distributed on a single layer (with an horizontal section of 30 mm × 30 mm) placed at a depth of 1 mm from the sensor's surface.

Single layer at 2 mm: The markers were randomly distributed on a single layer (with an horizontal section of 30 mm × 30 mm) placed at a depth of 2 mm from the sensor's surface.

Single layer at 6 mm: The markers were randomly distributed on a single layer (with an horizontal section of 30 mm × 30 mm) placed at a depth of 6 mm from the sensor's surface.

Homogeneous spread between 1 mm and 6 mm: The markers were randomly distributed over a depth range of 1–6 mm from the sensor's surface (covering an horizontal section of 30 mm × 30 mm).

In the reminder of this article, vectors are expressed as tuples for ease of notation, with dimension and stacking clear from the context.

3.1 Model

A semi-infinite linear elastic half-space model was assumed for the surface material. This model has shown limited loss of accuracy in the force reconstruction task for indentations in the proximity of the origin of the half-space (see [20]), and provides a tractable analytical solution. Although a simple scenario with a single rubber layer was considered, the physical parameters of the conducted simulations were chosen to approximate the first-order behavior of the sensor presented in Section 2.

Three Cartesian axes, x, y and z, defined the world coordinate frame. These were centered at the origin of the half-space, placed on the top surface. The z-axis was positive in the direction entering the material, and the remaining two axes spanned the horizontal surface. Let $s_j := (x_j, y_j, z_j)$ be the position of a marker j before the force is applied, for $j = 0, \ldots, N_\mathrm{m} - 1$, where N_m is the number of markers within the gel. Given a point force applied at the origin (this model can be easily extended to the entire force distribution, as shown in [17]), the three-dimensional displacement Δs_j of the marker j can be derived from the Bousinnesq and Cerruti solutions (see [21]), as,

$$\Delta s_j = H_j F, \tag{1}$$

where $F \in \mathbb{R}^3$ is the applied force vector, and,

$$H_j = \frac{1+\nu}{2\pi E R_j} \begin{bmatrix} \frac{R_j^2 + x_j^2}{R_j^2} + (1-2\nu)\frac{R_j^2 + R_j z_j - x_j^2}{(R_j + z_j)^2} & \frac{x_j y_j}{R_j^2} - (1-2\nu)\frac{x_j y_j}{(R_j + z_j)^2} & \frac{x_j z_j}{R_j^2} - (1-2\nu)\frac{x_j}{R_j + z_j} \\ \frac{x_j y_j}{R_j^2} - (1-2\nu)\frac{x_j y_j}{(R_j + z_j)^2} & \frac{R_j^2 + y_j^2}{R_j^2} + (1-2\nu)\frac{R_j^2 + R_j z_j - y_j^2}{(R_j + z_j)^2} & \frac{y_j z_j}{R_j^2} - (1-2\nu)\frac{y_j}{R_j + z_j} \\ \frac{x_j z_j}{R_j^2} + (1-2\nu)\frac{x_j}{R_j + z_j} & \frac{y_j z_j}{R_j^2} + (1-2\nu)\frac{y_j}{R_j + z_j} & \frac{z_j^2}{R_j^2} + 2(1-\nu) \end{bmatrix}, \tag{2}$$

where ν and E are the Poisson's ratio and the Young's modulus of the material, respectively, and $R_j := \sqrt{x_j^2 + y_j^2 + z_j^2}$.

A pinhole camera model (see [22] (p. 49)) was assumed, with square pixels and the optical center projected at the origin of the image coordinate frame, on the z-axis of the world frame. The image plane was parallel to the sensor's surface.

The displacement $\Delta p_j \in \mathbb{R}^2$, as observed by the camera and expressed in pixels in the image frame, was computed as the difference between the marker positions in the image after and before the force is applied. This leads to,

$$\begin{aligned} \Delta p_j &= K_{\mathrm{a},j}(s_j + \Delta s_j) - K_{\mathrm{b},j} s_j \\ &= (K_{\mathrm{a},j} - K_{\mathrm{b},j}) s_j + K_{\mathrm{a},j} H_j F, \end{aligned} \tag{3}$$

where

$$K_{\mathrm{a},j} = \begin{bmatrix} \frac{f}{d-(z_j+H_{j,3}F)} & 0 & 0 \\ 0 & \frac{f}{d-(z_j+H_{j,3}F)} & 0 \end{bmatrix}, \tag{4}$$

$$K_{\mathrm{b},j} = \begin{bmatrix} \frac{f}{d-z_j} & 0 & 0 \\ 0 & \frac{f}{d-z_j} & 0 \end{bmatrix}, \tag{5}$$

f is the focal length of the camera (in pixels), d is the distance of the optical center from the origin of the world coordinate frame, and $H_{j,i}$ represents the ith row of the H_j matrix, for $i = 1, 2, 3$. Note that, for small d, which is usually the case for state-of-the-art optical tactile sensors, the difference between $K_{\mathrm{a},j}$ and $K_{\mathrm{b},j}$ is not negligible. The expression in Equation (3) can be rearranged, as explained in Appendix B, as,

$$\Delta p_j = \underbrace{\left\{ \frac{\Delta p_j}{d-z_j} H_{j,3} + \frac{f}{(d-z_j)^2} \begin{bmatrix} x_j \\ y_j \end{bmatrix} H_{j,3} + \frac{f}{d-z_j} \begin{bmatrix} H_{j,1} \\ H_{j,2} \end{bmatrix} \right\}}_{:=P_j(\Delta p_j)} F = P_j(\Delta p_j) \cdot F, \tag{6}$$

where $P_j \in \mathbb{R}^{2\times 3}$ depends on the displacement Δp_j. For force sensing applications, once the marker displacement Δp_j is observed in the image, the equality in Equation (6) represents an underdetermined system of two equations with three unknowns (the components of F). To find a unique solution, and to improve the reconstruction error in the presence of noise, more equations can be provided by tracking the remaining markers, yielding,

$$\Delta p := \begin{bmatrix} \Delta p_0 \\ \vdots \\ \Delta p_{N_\mathrm{m}-1} \end{bmatrix} = \underbrace{\begin{bmatrix} P_0(\Delta p_0) \\ \vdots \\ P_{N_\mathrm{m}-1}(\Delta p_{N_\mathrm{m}-1}) \end{bmatrix}}_{:=P(\Delta p)} \cdot F = P(\Delta p) \cdot F, \tag{7}$$

with $P \in \mathbb{R}^{2N_\mathrm{m}\times 3}$. The force F can then be reconstructed as,

$$F = P(\Delta p)^\dagger \cdot \Delta p, \tag{8}$$

where the apex † indicates the Moore–Penrose pseudo-inverse matrix. The matrix P depends on the different Δp_j, therefore the equality in Equation (8) indicates a nonlinear dependence between F and Δp.

The parameters used for the Monte Carlo simulations described in the following subsections are summarized in Table 1. In particular, the Young's modulus was chosen as the linear combination of the resulting values for the soft layers described in Section 2, according to the conversions introduced in [23] (note, however, that this value is only a multiplicative factor in Equation (2), and does not have a large effect on the resulting

trends in the simulations).

Symbol	Value	Description
ν	0.4999	Poisson's ratio
E	0.1 MPa	Young's modulus
f	256 pixels	Focal length
d	20 mm	z-coordinate of the camera optical center
N_f	1000	Number of drawn force samples
N_c	1000	Number of drawn configurations for each class

Table 1. Simulation parameters.

3.2 Robustness to noise

To a certain extent, the learning algorithm presented in Section 6 aims to approximate a map similar to the one in Equation (8), which relates the observed marker displacements to the applied external force. For the purpose of analyzing how the output of this map reacts to a perturbation δp of the displacements Δp, a first order linearization was performed around the unperturbed state. Therefore, the force corresponding to the perturbed displacements can be expressed as,

$$F\Big|_{\Delta p+\delta p} \approx P(\Delta p)^\dagger \cdot \Delta p + J(\Delta p) \cdot \delta p, \tag{9}$$

where J is the appropriate Jacobian matrix, containing the derivative terms of the map with respect to Δp. Using the properties of the norm,

$$\left\| F\Big|_{\Delta p+\delta p} - F\Big|_{\Delta p} \right\|_2 \approx \|J(\Delta p)\delta p\|_2 \leq \underbrace{\|J(\Delta p)\|_F}_{:=\kappa(\Delta p)} \|\delta p\|_2, \tag{10}$$

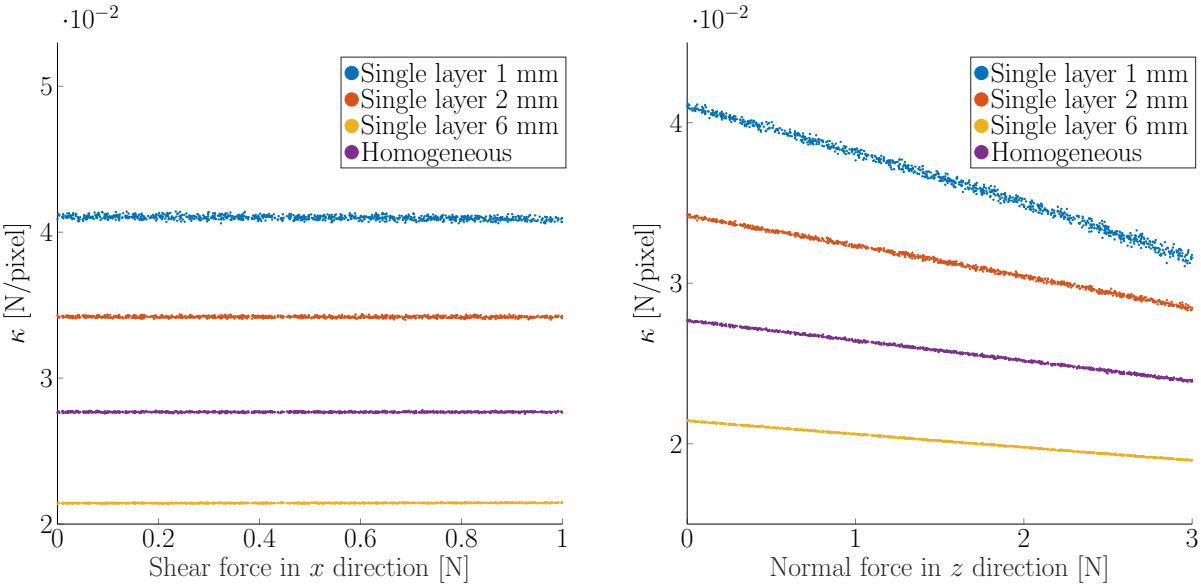
where the L^2-norm and the Frobenius norm are used accordingly. The value that $\kappa \in \mathbb{R}$ takes is often referred to as the *absolute condition number* of a function (see [24] (p. 221)), and it provides a bound on the slope at which a function changes given a change in the input. In this analysis, it quantified the sensitivity of the map in Equation (8) to noise in the observed displacements. Note that the noise considered here might stem from both image noise and errors in the estimation of the marker displacements (e.g., errors in the optical flow estimation, see Section 5).

The Monte Carlo simulations were performed as follows: (1) N_f force samples were randomly drawn. Each of these samples represented a three-dimensional concentrated force vector, which was placed at the origin of the world coordinate frame. (2) For each force sample, N_c marker configurations were randomly drawn for each layout class, and the resulting marker displacements were projected to the image frame. (3) Therefore, the re-

sulting value of κ was computed and averaged for the N_c different random marker configurations (in the same layout class).

Figure 5 shows the average magnitude of κ for different types of forces. For ease of visualization, the results presented here considered the application of pure shear (along one axis) or pure normal force, but similar conclusions were obtained for generic force vectors. The figures indicate how the map in Equation (8) was more robust for configurations with the markers placed closer to the camera, i.e., deeper in the soft material. The layout class with an homogeneous spread of markers between 1 mm and 6 mm showed higher robustness compared to the case of all markers placed at 1 mm or 2 mm depth (but was lower when compared to the class with the markers placed at 6 mm depth). Note that there were two counteracting effects in the model considered, that is, markers closer to the camera exhibited smaller displacements, but these displacements underwent higher amplification when projected to the image.

The sensor's robustness to noise was also evaluated in the same simulation framework by perturbing the observed marker displacements after projecting these to the image frame. Zero-mean Gaussian noise with variance σ_p^2 was added to the observed Δp, and the force was reconstructed using Equation (8). The resulting root mean squared error (RMSE) for varying σ_p (averaged over the different simulations) is shown in Figure 6, showing that, considering higher order terms, neglected in the linearization, led to the same trend as in Figure 5.



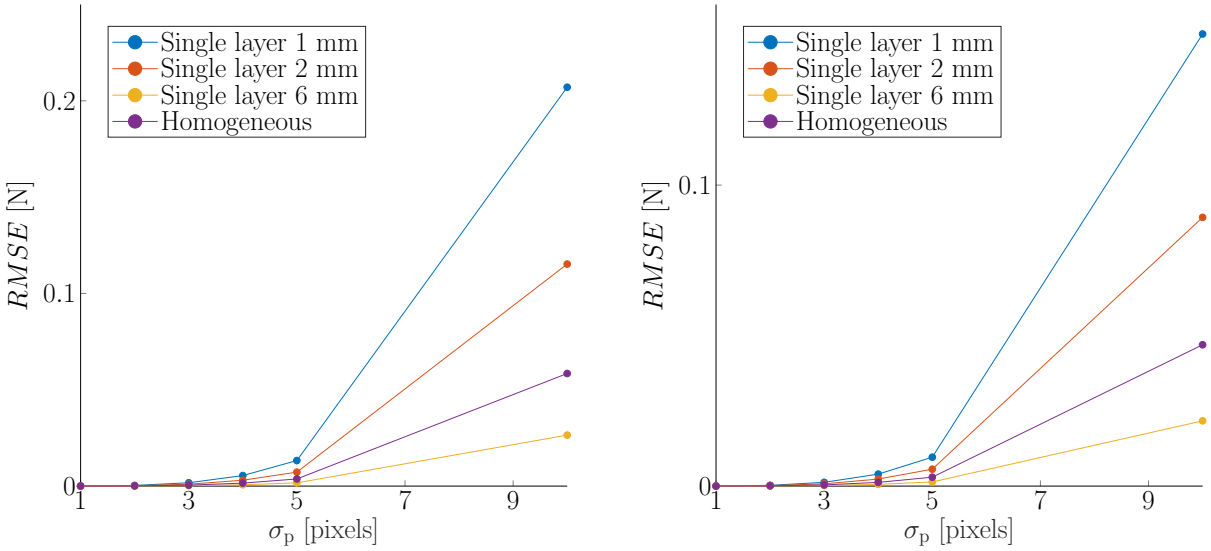
(a) Absolute condition number for shear force

(b) Absolute condition number for normal force

Figure 5. The plots show the resulting absolute condition number κ for various shear (**a**) and normal (**b**) force samples, averaged over multiple configurations in the same class, for the different layout classes.

3.3 Sensor threshold

The tactile sensor threshold is given by the minimum force that generates a noticeable change in the image, i.e., when a marker is observed at different pixel coordinates. In the



(a) Error for shear force

(b) Error for normal force

Figure 6. Tactile sensing applications are simulated by drawing shear (**a**) and normal (**b**) force samples, computing the resulting marker displacements, and projecting these to the image frame. Therefore, Gaussian noise with variance σ_p^2 was added to the observed displacements, and the force was reconstructed through the map in Equation (8). The resulting averaged root mean squared error is shown in these plots for different σ_p.

worst case scenario (a similar analysis applies to other scenarios), each marker center (or feature) of interest is at the center of a pixel. In this case, a marker needs to move by at least half a pixel to be observed at a different position in the image. Therefore, during the simulations described in Section 3.2, the component-wise maximum among the observed marker displacements was computed (that is, the maximum component of the observed Δp vector). For each force sample, this quantity was averaged over the N_c random marker configurations. The sensor threshold was then estimated as the minimum magnitude of the (shear or normal) force that yields a displacement greater than half a pixel.

Table 2 shows the resulting sensor threshold for pure normal and shear forces, respectively, for the different classes of layouts. In the simplified scenario considered, the configurations with the markers placed closer to the sensor's top surface exhibited a smaller threshold, while the layout class with homogeneous spreads of markers at different depths retained a threshold comparable to the layout class with a single layer at 1 mm depth. Summarizing, the analyses presented here show that spreading the markers homogeneously at different depths (in this example between 1 mm and 6 mm) might yield an advantageous trade-off between robustness to noise in the observed marker displacements and sensor threshold. In particular, compared to the case of all markers at a depth of 1 mm, this layout class showed higher robustness to noise while retaining a comparable sensor threshold (which for this example was about 2–3 times smaller than the layout with a single layer at 6 mm depth).

Layout Class	Sensor Threshold (Shear)	Sensor Threshold (Normal)
Single layer at 1 mm	0.013 N	0.035 N
Single layer at 2 mm	0.027 N	0.059 N
Single layer at 6 mm	0.056 N	0.084 N
Homogeneous spread	0.018 N	0.048 N

Table 2. Sensor threshold for various layout classes.

4. Ground truth data

In the context of supervised learning, each data point is associated with a label that represents the ground truth of the quantity of interest. In the approach presented here, a label vector representing the normal force distribution is assigned to the image that the camera captures when this force is applied. In this section, the procedure followed to collect the data, which were then processed and used to train the learning architecture presented in Section 6, is described. The generation of ground truth label vectors is then discussed.

4.1 Data collection

The training data were automatically collected by pressing an indenter's tip against the sensor gel by means of a milling and drilling machine (Fehlmann PICOMAX 56 TOP).

The machine was equipped with a three-axis computer numerical control, which enabled precise motion (in the order of 10^{-3} mm) of a spindle in the three-dimensional operating space. A spherical-ended cylindrical indenter (40 mm long, with a diameter of 1.2 mm) was mounted on a state-of-the-art six-axis F/T sensor (ATI Mini 27 Titanium) through a plexiglass plate. The F/T sensor was attached to the spindle and used to measure the force applied by the indenter, which was pressed against the gel at different depths and positions. An image of this setup is shown in Figure 7.

The milling machine outputted a digital signal, which was positive when the needle reached the commanded pressure position. The signal was read by a standard laptop, which was also responsible for reading the F/T sensor data and the camera stream. The data were synchronized on the laptop by extracting the image frames that were recorded (and cropped to 480 $\times$ 480 pixels) when the digital signal was positive. These images were then matched to the corresponding normal force value, which was sent by the F/T sensor.

4.2 Data labels

The neural network architecture presented in Section 6 requires the labels to be expressed as real vectors. To this purpose, the sensor surface was divided into n bins, each of these representing a different region. Therefore, a corresponding n-dimensional label vector embedded the force that was applied to each of these regions. Since the force was applied

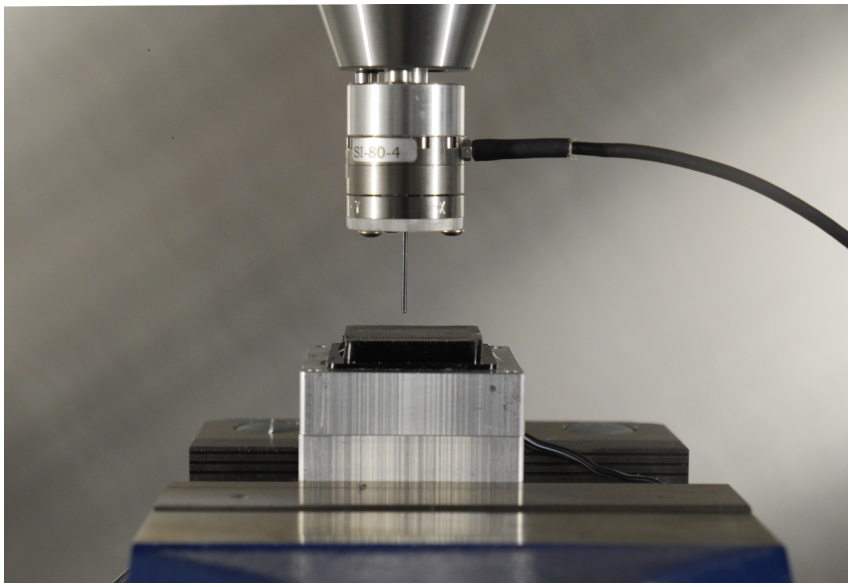


Figure 7. The image shows the data collection setup in the automatic milling machine. The F/T sensor is the cylindrical device connected with the cable in the upper part of the figure.

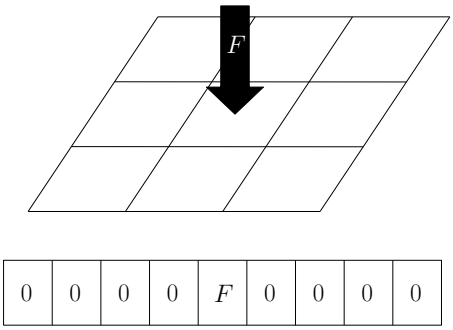


Figure 8. The scheme shows an example of a ground truth label vector. The indenter applies a force F to the center of the surface, which is part of a specific bin. The corresponding central vector component is then set to F, while the remaining components are set to zero.

to the gel with the relatively small spherical indenter used for the training data collection, this was simplified as a point force at the center of the contact. The point of application of the force was mapped to one of the n surface bins, and the value of the force read by the F/T sensor was assigned to the label vector component that represented the appropriate bin. All remaining components were then set to zero, to indicate a zero force distribution in those regions. A scheme of this procedure is shown in Figure 8.

Note that the number of surface bins n provides an indication of the spatial resolution of the sensor (measured in millimeters), which is different from the camera resolution that was fixed at 480×480 pixels in the scenario considered here. A larger n provides a finer spatial resolution, at the expense of a higher dimensional map between images and force distribution, which is generally more complex to estimate.

5. Optical flow features

Extracting meaningful features by processing the images has the benefit of reducing the required amount of training data and the training times.

In a soft material, the strain field provides information about the force applied to the surface, which generates the material deformation [25]. In the approach presented here, the spherical markers' displacement rendered an approximation of the strain field, which could be obtained by means of optical flow techniques. In fact, the input features to the supervised learning architecture presented in Section 6 were derived from the estimated optical flow.

This section discusses two approaches to the feature engineering problem, based on sparse and dense optical flow, respectively [26].

5.1 Sparse optical flow

Sparse optical flow methods are based on tracking the movement of a set of keypoints. In particular, the Lucas Kanade algorithm (see [27]) solves the optical flow equations relying on the assumption that pixels in a small neighborhood have the same motion.

For the tactile sensor proposed in this paper, the centers of the fluorescent markers represented a natural choice as keypoints. However, identifying these keypoints required a detection phase with the gel surface at rest. The proposed algorithm is based on the watershed transformation for image segmentation (see [28]). The different steps are explained in Figure 9.

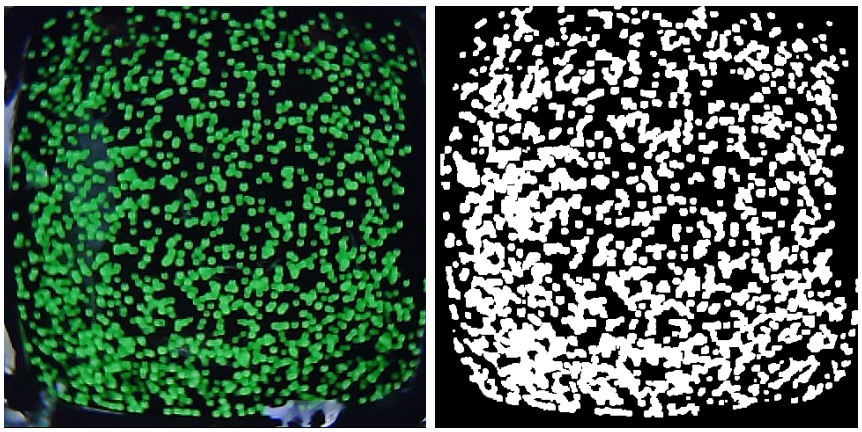
Once the keypoints were chosen, they were tracked in the following frames using Lucas Kanade optical flow. The resulting flow was represented as a tuple of magnitude and angle with respect to a defined axis. Similar to the approach explained for generating the label vectors, the image was divided into a uniform grid of m regions. Note that m does not necessarily have to be equal to n. The markers were assigned to the appropriate resulting bins, depending on their location in the image. The features were then defined by the following average quantities, for each image bin $i = 1, \dots, m$,

$$d_{\text{avg},i} = \frac{1}{N_i} \sum_{j=0}^{N_i-1} d_{ij} \tag{11}$$

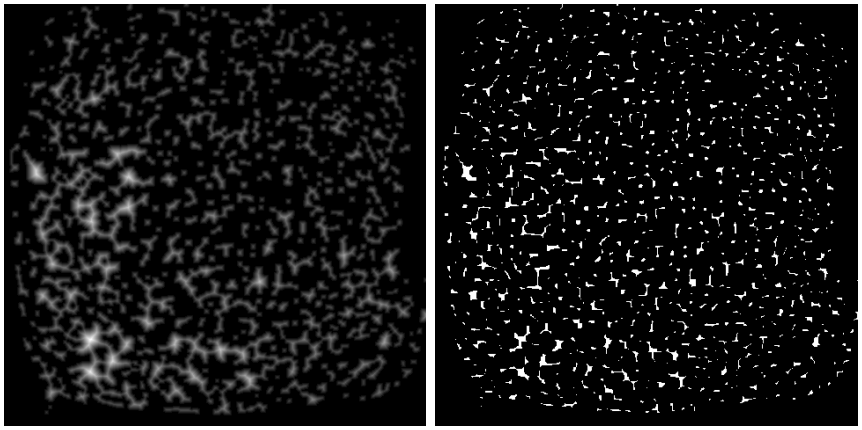
$$\alpha_{\text{avg},i} = \text{atan2}\left(\sum_{j=0}^{N_i-1} \sin(\alpha_{ij}), \sum_{j=0}^{N_i-1} \cos(\alpha_{ij}) \right), \tag{12}$$

where d_{ij} and α_{ij} represent the magnitude and angle tuple relative to the displacement of the marker j in the region i, and N_i is the number of markers assigned to the region i. The average defined in Equation (12) is often referred as the circular mean of the angles (see [30] (p. 106) for an explanation).

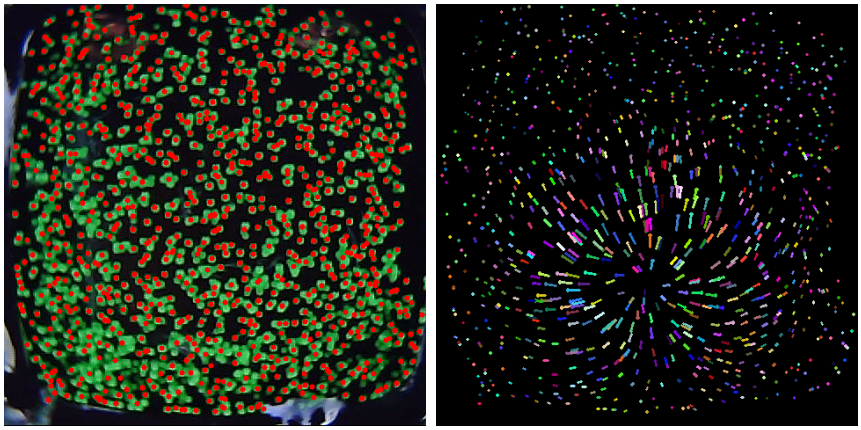
Under the assumptions that the markers were homogeneously distributed over the



(a) Original image **(b)** Thresholded markers



(c) Distance transform **(d)** Peaks of the distance transform



(e) Chosen keypoints **(f)** Sparse optical flow

Figure 9. The original image (**a**) is thresholded to extract the green color of the markers (**b**). The distance transform (see [29] for the definition) is computed (**c**), and its local peaks are extracted (**d**). The watershed algorithm is finally applied to segment the original image into the different markers. The centers of the resulting contours, in red (**e**), are chosen as the keypoints to track through Lucas Kanade optical flow in the following frames (**f**).

entire volume of the gel, these average quantities provided an approximation of the optical flow at each of the bin centers. The feature vector for each image was therefore obtained by appropriately stacking the two average quantities for each of the m regions, resulting in a $2m$-dimensional feature space. Note that the averages also provided invariance of the features to the marker pattern, provided that the same material and design were retained.

5.2 Dense optical flow

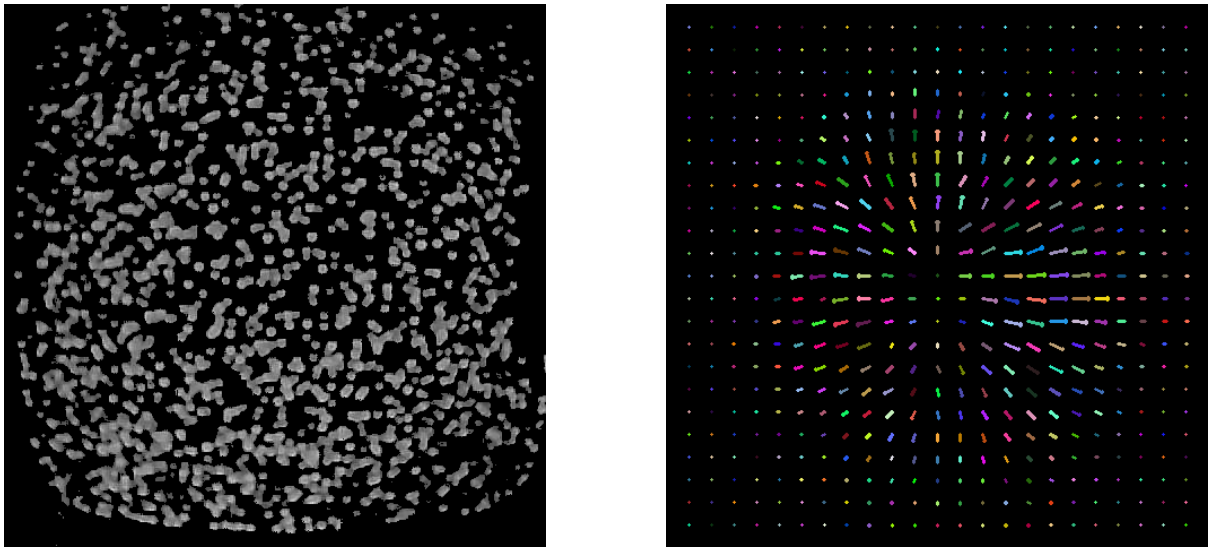
Compared to sparse optical flow, dense optical flow methods are more accurate, at the expense of a higher need for computational resources [31]. Moreover, they do not rely on a preliminary detection phase, which may introduce further inaccuracies (see, for example, the lower-left part of Figure 9e). In fact, rather than tracking a set of features over subsequent frames, they aim at estimating the motion at each pixel of the image. The Dense Inverse Search (DIS) optical flow algorithm (see [32]) reconstructs the motion through the computation of the flow at different image scales, and exploits an inverse search technique to obtain a considerable increase in speed. Therefore, rather than relying on the detection of keypoints, the DIS algorithm reconstructs the optical flow from any trackable distinct pattern. Moreover, it exploits the full resolution of the camera (distinct from the spatial resolution of the tactile sensor), rendering motion information at each pixel of the image. As a consequence, the spatial resolution of the tactile sensor is not limited by the number of markers. Note that a specific marker layout (e.g., a uniform grid) would not simplify the image processing and the reconstruction of the dense optical flow, and an eventual regular spacing could cause the loss of information at certain pixels. Conversely, the random spread of markers facilitates manufacture and does not impact the deployment of the sensor to applications where a larger contact surface with arbitrary shape is required.

To match the required format of the DIS algorithm, the original image was processed. The green regions in the image were thresholded and the remaining non-green regions were replaced with black pixels, removing external light disturbances and material imperfections. Finally, the resulting image was converted to gray-scale. An example of the masked image and the computed dense optical flow are shown in Figure 10.

As in the case of the sparse keypoint tracking, the resulting flow was represented as tuples of magnitude and angle, in this case for each pixel. The pixels were then assigned to m image bins. Redefining N_i for the dense optical flow case as the number of pixels assigned to the region i, the averages defined in Equations (11) and (12) were computed, to compose a set of $2m$ features.

6. Learning architecture

The problem of reconstructing the normal force distribution from images can be formulated as a multiple multivariate regression problem, that is, a regression problem where



(a) Masked image **(b)** Dense optical flow

Figure 10. The original image is processed by replacing non-green regions with black pixels (**a**). The DIS algorithm computes the dense optical flow (**b**) on the resulting image. Note that the flow is estimated at each pixel, and a subsampled version is shown in (**b**) for ease of visualization.

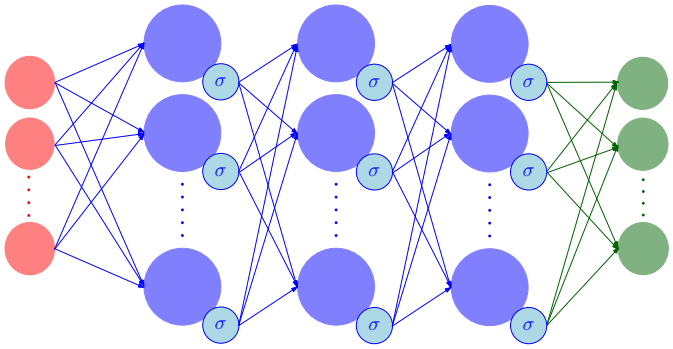


Figure 11. The figure shows a schematic representation of the DNN architecture, neglecting the bias terms. The $2m$ input layer neurons (in red) enter the hidden layers (in blue), which have sigmoid activation functions. The output layer, with n outputs, is shown in green. Note that $2m$ and n are the number of features and surface bins, respectively.

both input features and labels are multi-dimensional vectors. A multi-output deep neural network (DNN) provides a single function approximation to the underlying map, which exploits the interrelations among the different inputs and outputs. Similar to the work in [6], the approach proposed in this article presents a feedforward DNN architecture (see Figure 11), with three fully connected hidden layers of width 1600, which apply a sigmoid function to their outputs. The input data to the network were the $2m$ (sparse or dense) optical flow features described in Section 5, while the output vectors provided an estimate of the normal force applied at each of the n bins the tactile sensor surface was divided into, as described in Section 4.

The architecture weights were trained with RMSProp with Nesterov momentum

(known in the literature as Nadam, see [33]), with training batches of 100 samples and a learning rate of 0.001. The solver aimed at minimizing an average root mean squared error ($aRMSE$), defined according to [34] as,

$$aRMSE = \frac{1}{n}\sum_{i=0}^{n-1}\sqrt{\frac{\sum_{l=0}^{N_{\text{set}}-1}\left(y_i^{(l)} - \hat{y}_i^{(l)}\right)^2}{N_{\text{set}}}}, \tag{13}$$

where $y_i^{(l)}$ and $\hat{y}_i^{(l)}$ denote the ith true and predicted label vector component, respectively, for the lth sample in the considered dataset portion (i.e., training, validation, and test sets), which contains N_{set} data points.

Twenty percent of each dataset was retained as a test set, against which the performance was evaluated. Ten percent of the remaining data points were randomly chosen as a validation set. Dropout regularization (at a 10% rate) was employed at each hidden layer during training, which terminated when the loss computed on the validation set did not decrease for 50 consecutive epochs.

Due to the very sparse structure of the label vectors considered in the experiments presented here (see Figure 8), two evaluation metrics were computed on the test set, in addition to the $aRMSE$, which are more intuitive for this application. For $l = 0, 1, \ldots, N_{\text{set}}-1$, the label component of the ground truth vector where the magnitude of the applied force is maximum was computed,

$$k_l = \underset{i=0,1,\ldots,n-1}{\arg\max} |y_i^{(l)}|. \tag{14}$$

The same quantity was computed for the estimated label vector,

$$\hat{k}_l = \underset{i=0,1,\ldots,n-1}{\arg\max} |\hat{y}_i^{(l)}|. \tag{15}$$

Denoting $c(k)$ as the location of the center of a surface bin k on the horizontal plane, a distance metric was introduced,

$$d_{\text{loc}} = \frac{1}{N_{\text{set}}}\sum_{l=0}^{N_{\text{set}}-1}\|c(k_l) - c(\hat{k}_l)\|_2. \tag{16}$$

Finally, an error computed on the maximum components of the true and estimated

label vectors was defined as,

$$RMSE_{\mathrm{mc}} = \sqrt{\frac{1}{N_{\mathrm{set}}} \sum_{l=0}^{N_{\mathrm{set}}-1} \left(y_{k_l}^{(l)} - \hat{y}_{\hat{k}_l}^{(l)} \right)^2}. \tag{17}$$

The metric in Equation (16) provides an indication of how close the location of the maximum estimated force was to the real location of the force applied by the indenter described in Section 4. The metric in Equation (17) indicates how accurate (in the magnitude) the estimation of this force was.

7. Results

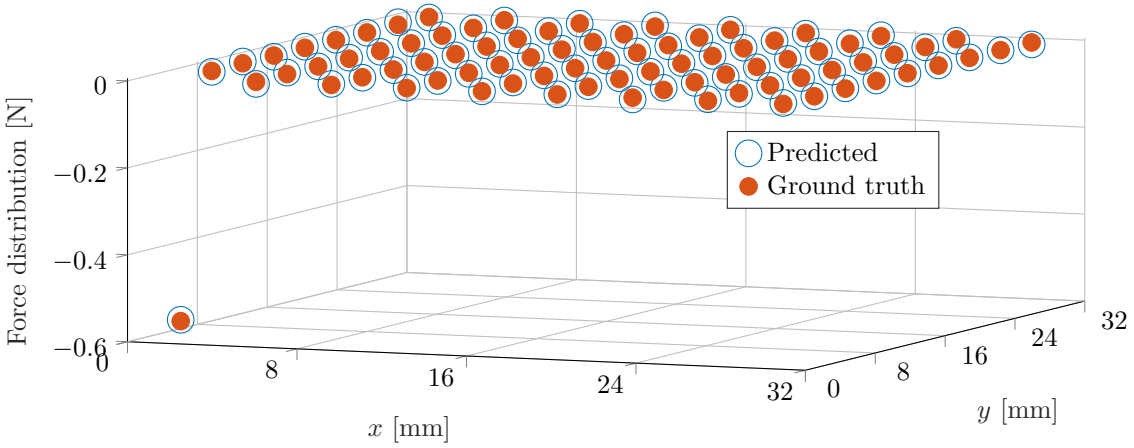
The tactile sensor's performance was evaluated on a dataset collected on a test indenter, as described in Section 4. The images (acquired at a resolution of 640 × 480 pixels) were cropped to a region of interest of 480 × 480 pixels, which covered the gel surface. A total of 10,952 data points were recorded by commanding the needle to reach various depths (up to 2 mm, for a maximum force of 1 N) at different positions on the surface, which were defined by an equally-spaced grid (0.75 mm between adjacent points). Note that other ranges of force could be similarly covered by replacing the Ecoflex™ GEL with another material with different hardness.

An example of the prediction of the normal force distribution is shown in Figure 12. The learning architecture was evaluated for different values of m and n, and for features based on sparse and dense optical flow. The results, greatly outperforming the authors' previous work in [6], are shown in Figure 13.

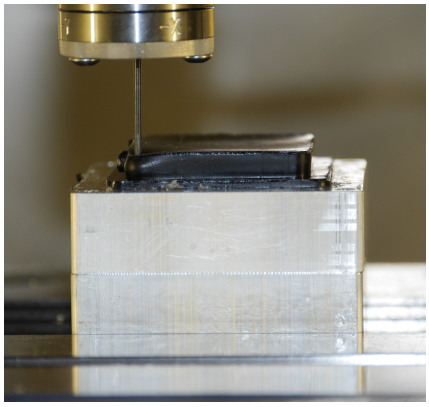
The plots show that, in terms of the $aRMSE$, which is the loss actually optimized during training, the dense optical flow outperformed the sparse optical flow in estimating the force distribution, in both the finer and coarser spatial resolution cases (determined by n). However, this metric did not provide a reliable comparison across different spatial resolutions, since it was influenced by the zero values in the label vectors, which were considerably more in the finer resolution case.

As shown in Figure 13b, the $RMSE_{\mathrm{mc}}$ decreased by increasing the number of averaging regions in the image, and was lower for the dense optical flow case, considering the same number of surface bins. Estimating the force distribution with a finer spatial resolution slightly degraded the performance according to this metric, and might require a different network architecture and more training data, due to the higher dimension of the predicted output.

The use of the information at all pixels, provided by the dense optical flow, resulted in a better accuracy in estimating the location of the applied force (indicated by d_{loc}), compared to the sparse optical flow. In fact, the performance obtained with finer resolution and dense optical flow features was comparable to the case with coarser resolution and



(a) Prediction



(b) Indentation

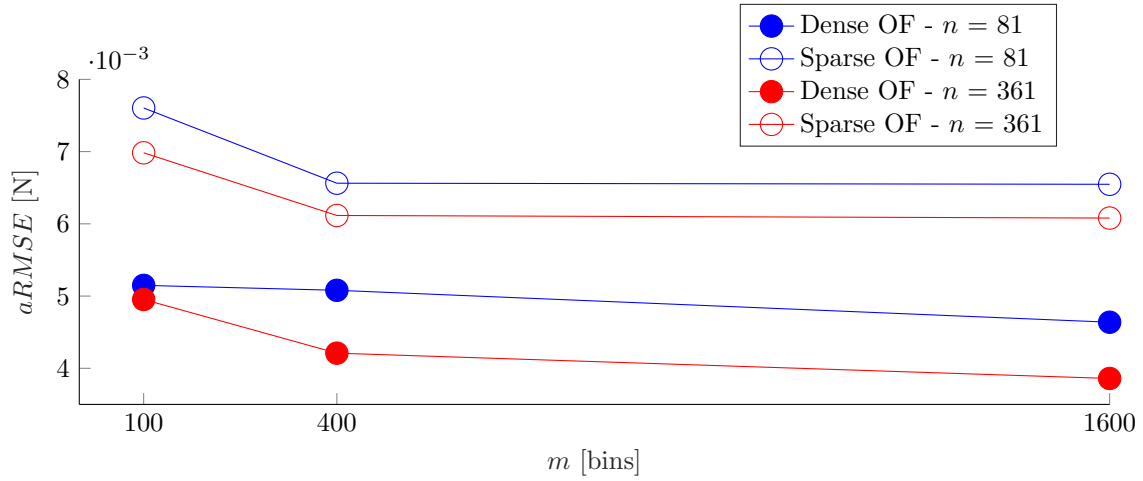
Figure 12. (**a**) The prediction of the force distribution for a sample in the test set (for a number of image bins $m = 1600$, surface bins $n = 81$ and dense optical flow features); and (**b**) the corresponding indentation. High accuracy was achieved at each bin, including the corners of the gel, which underwent a deformation that generally differed from the rest of the surface when subject to force (due to boundary effects).

sparse optical flow features. Moreover, excessively increasing the number of image bins m might degrade the performance for the sparse optical flow case, since some averaging regions might not be covered by a sufficient number of detected keypoints.

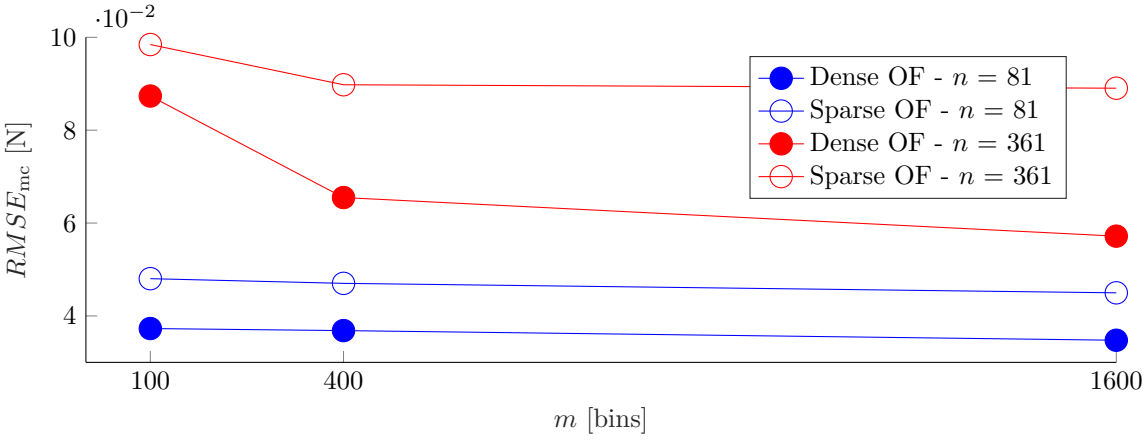
8. Conclusions

This article has discussed the design of and the motivation for a novel vision-based tactile sensor, which uses a camera to extract information on the force distribution applied to a soft material. Simulation results have shown the benefits of the proposed strategy. The experimental results presented in this paper (see Supplementary video[1]) have

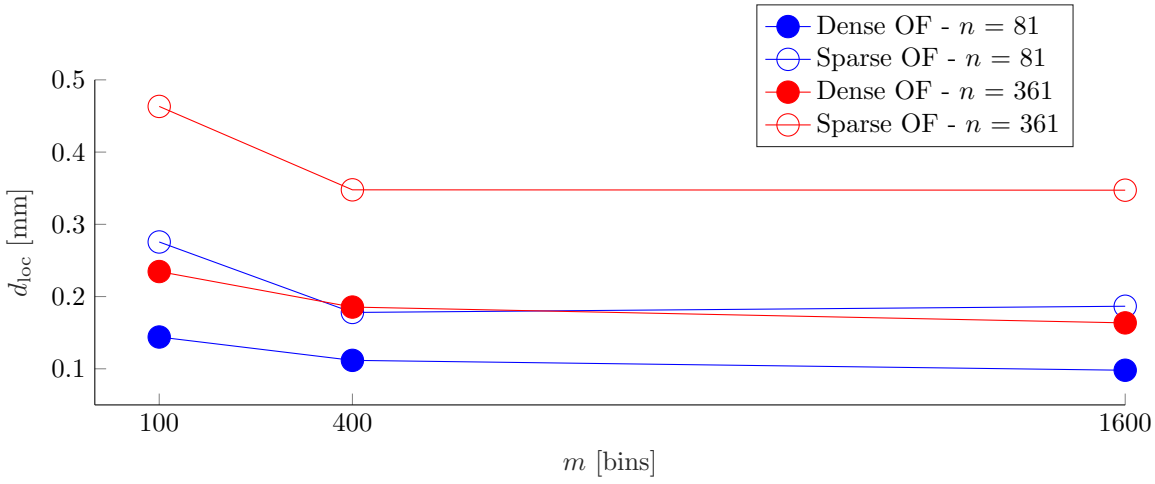
[1]Supplementary video: `https://www.mdpi.com/1424-8220/19/4/928/s1`



(a) Resulting $aRMSE$



(b) Resulting $RMSE_{\text{mc}}$



(c) Resulting d_{loc}

Figure 13. The plots show the resulting metrics (defined in Equations (13), (17) and (16)) for various values of image bins m and surface bins n, and for dense and sparse optical flow (OF) features. The resolution of the ground truth F/T sensor is 0.06 N.

shown high accuracy in reconstructing the normal force distribution applied by a small spherical indenter. Although in the experimental setup considered an approximation to a point indenter has been applied, the proposed learning algorithm does not explicitly use knowledge of this information (that is, the point force and the zero valued regions are indistinctly predicted through the same architecture). In fact, the representation of the force distribution introduced in this article (embedded in appropriate label vectors) is suitable to represent generic normal forces, including multi-point contacts and larger indentations. Moreover, this representation can be readily extended to shear forces, by appropriate vector concatenation. Future work will investigate the cases not considered in this article, to address the limitations introduced by the point force approximation. To this purpose, the approach proposed here will need to be evaluated for arbitrary force distributions on a wider dataset, with various indenters and multiple types of contact. These steps will be aimed towards the generalization of the current approach to touch with objects of arbitrary shapes (e.g., for robotic grasping tasks).

Dense optical flow features, whose application in the context of vision-based tactile sensors is novel (to the authors' knowledge), exhibit higher performance compared to (sparse) keypoint tracking and retain fast execution times. The use of these features leverages the randomized marker layout, since (as highlighted in Section 5) the spacing required for regular grid patterns, as for example in [35], might instead cause a decrease in accuracy in the estimation of the dense optical flow.

The spatial resolution ($n = 361$ corresponds to surface bins with a side of 1.58 mm) and sensing range achieved are comparable to the human skin (see [36] for a reference), and the resulting pipeline runs in real-time at 60 Hz.

A. Choice of the particles' density

The cure time of the Ecoflex™ GEL, which is mixed with the markers, is about 2 h at a room temperature of 23 °C. However, after pouring the material into the mold, the cure is accelerated with mild heat (up to 80 °C in an oven), resulting in cure times of about 10 min. During this time, a density difference between the spherical particles and the material might lead to sinking or floating of the markers.

Assuming that a particle has higher density than the fluid (a symmetrical analysis applies to the opposite case), the dynamics of a single marker in the material is described by the following differential equation,

$$\rho_{\mathrm{p}} V_{\mathrm{p}} \ddot{x}_{\mathrm{p}}(t) = \rho_{\mathrm{p}} V_{\mathrm{p}} g - \rho_{\mathrm{f}} V_{\mathrm{p}} g - F_{\mathrm{D}}(t), \tag{18}$$

where $t \geq 0$ denotes the time in seconds; x_{p} indicates the position of the particle (positive in the direction of gravity); ρ_{p} and ρ_{f} denote the density of the particle and the fluid, respectively; V_{p} is the volume of the particle; and g is the gravitational acceleration.

Given the low Reynolds number of the considered application (low speed and very small particle diameter), the drag force F_{D} is described through the Stoke's law, that is,

$$F_{\mathrm{D}}(t) = 6\pi\eta R_{\mathrm{p}}\dot{x}_{\mathrm{p}}(t), \tag{19}$$

where η is the dynamic viscosity of the material and R_{p} is the radius of the particle.

The solution to the differential equation in Equation (18) is,

$$x_{\mathrm{p}}(t) = (\rho_{\mathrm{p}} - \rho_{\mathrm{f}})V_{\mathrm{p}}g\left[t + \frac{\rho_{\mathrm{p}}V_{\mathrm{p}}}{b}\left(\exp\left(-\frac{bt}{\rho_{\mathrm{p}}V_{\mathrm{p}}}\right) - 1\right)\right] + x_{\mathrm{p},0}, \tag{20}$$

with $b := 6\pi\eta R_{\mathrm{p}}$, and $x_{\mathrm{p},0}$ the starting position of the marker. The parameters of the setup presented in this article are shown in Table 3, and lead to a change in position of about 0.35 mm in 10 min. Considering that the mold lays on one of its sides during the material cure, this change does not affect the marker distribution to a large extent. Moreover, for simplicity, the drastic increase in dynamic viscosity (until hardening) that the material experiences during the cure was not considered in this analysis (which considers η constant). In practice, the dynamic viscosity increase contributes to considerably slow down the particle, further reducing the resulting displacement to a negligible value.

Symbol	Value	Description
ρ_{p}	1020 Kg/m^3	Density of a marker
ρ_{f}	980 Kg/m^3	Density of the material
V_{p}	0.0654 mm^3	Volume of a marker
g	9.81 m/s^2	Gravitational acceleration
η	9.3 Pa·s	Dynamic viscosity of the material
R_{p}	0.25 mm	Radius of a marker

Table 3. Physical parameters.

B. Model derivation

From Equations (4) and (5), the difference between the camera matrices after and before displacement is,

$$K_{\mathrm{a},j} - K_{\mathrm{b},j} = \begin{bmatrix} \frac{fH_{j,3}F}{(d-z_j-H_{j,3}F)(d-z_j)} & 0 & 0 \\ 0 & \frac{fH_{j,3}F}{(d-z_j-H_{j,3}F)(d-z_j)} & 0 \end{bmatrix}. \tag{21}$$

Therefore, Equation (3) becomes,

$$\begin{aligned}\Delta p_j &= \frac{f H_{j,3} F}{(d - z_j - H_{j,3}F)(d - z_j)} \begin{bmatrix} x_j \\ y_j \end{bmatrix} + \frac{f}{d - z_j - H_{j,3}F} \begin{bmatrix} H_{j,1}F \\ H_{j,2}F \end{bmatrix} \\ &= \frac{1}{(d - z_j - H_{j,3}F)(d - z_j)} \left\{ f H_{j,3} F \begin{bmatrix} x_j \\ y_j \end{bmatrix} + f(d - z_j) \begin{bmatrix} H_{j,1}F \\ H_{j,2}F \end{bmatrix} \right\}. \end{aligned} \tag{22}$$

Assuming that a marker is never at the same location as the optical center, Equation (22) can be rearranged as,

$$(d - z_j - H_{j,3}F)(d - z_j)\Delta p_j = f H_{j,3} F \begin{bmatrix} x_j \\ y_j \end{bmatrix} + f(d - z_j) \begin{bmatrix} H_{j,1}F \\ H_{j,2}F \end{bmatrix} \tag{23}$$

$$\left[(d - z_j)^2 - H_{j,3}F(d - z_j)\right] \Delta p_j = f H_{j,3} F \begin{bmatrix} x_j \\ y_j \end{bmatrix} + f(d - z_j) \begin{bmatrix} H_{j,1}F \\ H_{j,2}F \end{bmatrix} \tag{24}$$

$$(d - z_j)^2 \Delta p_j = H_{j,3}F(d - z_j)\Delta p_j + f H_{j,3} F \begin{bmatrix} x_j \\ y_j \end{bmatrix} + f(d - z_j) \begin{bmatrix} H_{j,1}F \\ H_{j,2}F \end{bmatrix}$$

$$\Delta p_j = \left\{ \frac{\Delta p_j}{d - z_j} H_{j,3} + \frac{f}{(d - z_j)^2} \begin{bmatrix} x_j \\ y_j \end{bmatrix} H_{j,3} + \frac{f}{d - z_j} \begin{bmatrix} H_{j,1} \\ H_{j,2} \end{bmatrix} \right\} F,$$

which is the same expression as in Equation (6).

Acknowledgments

The authors would like to thank Michael Egli, Anil Parsi and Marc-Andrè Corzillius for their support in the manufacturing of the sensor.

References

[1] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection", *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[2] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies", *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[3] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps", *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[4] G. Westling and R. S. Johansson, "Factors influencing the force control during precision grip", *Experimental brain research*, vol. 53, no. 2, pp. 277–284, 1984.

[5] R. Calandra, A. Owens, M. Upadhyaya, W. Yuan, J. Lin, E. H. Adelson, and S. Levine, "The feeling of success: Does touch sensing help predict grasp outcomes?", *Proceedings of Machine Learning Research*, vol. 78, pp. 314–323, 2017.

[6] C. Sferrazza and R. D'Andrea, "Transfer learning for vision-based tactile sensing", *arXiv preprint arXiv:1812.03163v1*, 2018.

[7] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016.

[8] M. Shimojo, A. Namiki, M. Ishikawa, R. Makino, and K. Mabuchi, "A tactile sensor sheet using pressure conductive rubber with electrical-wires stitched method", *IEEE Sensors Journal*, vol. 4, no. 5, pp. 589–596, 2004.

[9] B. P. Nabar, Z. Celik-Butler, and D. P. Butler, "Self-powered tactile pressure sensors using ordered crystalline zno nanorods on flexible substrates toward robotic skin and garments", *IEEE Sensors Journal*, vol. 15, no. 1, pp. 63–70, 2015.

[10] Z. Kappassov, J.-A. Corrales, and V. Perdereau, "Tactile sensing in dexterous robot hands", *Robotics and Autonomous Systems*, vol. 74, pp. 195–220, 2015.

[11] M. K. Johnson, F. Cole, A. Raj, and E. H. Adelson, "Microgeometry capture using an elastomeric sensor", *ACM Transactions on Graphics*, vol. 30, no. 4, 46:1–46:8, 2011.

[12] M. Koike, S. Saga, T. Okatani, and K. Deguchi, "Sensing method of total-internal-reflection-based tactile sensor", *Proceedings of the IEEE World Haptics Conference*, pp. 615–619, 2011.

[13] L. S. Lincoln, S. J. M. Bamberg, E. Parsons, C. Salisbury, and J. Wheeler, "An elastomeric insole for 3-axis ground reaction force measurement", *Proceedings of the IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics*, pp. 1512–1517, 2012.

[14] B. Winstone, G. Griffiths, C. Melhuish, T. Pipe, and J. Rossiter, "TACTIP — tactile fingertip device, challenges in reduction of size to ready for robot hand integration", *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, pp. 160–166, 2012.

[15] D. Ma, E. Donlon, S. Dong, and A. Rodriguez, "Dense tactile force distribution estimation using gelslim and inverse fem", *arXiv preprint arXiv:1810.04621*, 2018.

[16] B. Ward-Cherrier, L. Cramphorn, and N. F. Lepora, "Exploiting sensor symmetry for generalized tactile perception in biomimetic touch.", *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1218–1225, 2017.

[17] K. Kamiyama, H. Kajimoto, N. Kawakami, and S. Tachi, "Evaluation of a vision-based tactile sensor", *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1542–1547, 2004.

[18] W. Yuan, S. Dong, and E. H. Adelson, "GelSight: High-resolution robot tactile sensors for estimating geometry and force", *Sensors*, vol. 17, no. 12, pp. 2762–2782, 2017.

[19] M. Y. Chuah and S. Kim, "Improved normal and shear tactile force sensor performance via least squares artificial neural network (LSANN)", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 116–122, 2016.

[20] K. Kamiyama, K. Vlack, T. Mizota, H. Kajimoto, K. Kawakami, and S. Tachi, "Vision-based sensor for real-time measuring of surface traction fields", *IEEE Computer Graphics and Applications*, vol. 25, no. 1, pp. 68–75, 2005.

[21] R. B. Hetnarski and J. Ignaczak, *Mathematical theory of elasticity*. CRC Press, 2004.

[22] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[23] A. Mix and A. Giacomin, "Standardized polymer durometry", *Journal of Testing and evaluation*, vol. 39, no. 4, pp. 696–705, 2011.

[24] M. T. Heath, *Scientific computing*. McGraw-Hill New York, 2002.

[25] K. L. Johnson, *Contact mechanics*. Cambridge university press, 1987.

[26] D. Fleet and Y. Weiss, "Optical flow estimation", *Handbook of Mathematical Models in Computer Vision*, N. Paragios, Y. Chen, and O. Faugeras, Eds., 2006.

[27] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision", *Proceedings of the International Joint Conference on Artificial Intelligence*, 1981.

[28] S. Beucher, "The watershed transformation applied to image segmentation", *Scanning Microscopy Supplement 6*, pp. 299–324, 1992.

[29] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions", *Theory of computing*, vol. 8, no. 1, pp. 415–428, 2012.

[30] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[31] J. Wulff and M. J. Black, "Efficient sparse-to-dense optical flow estimation using a learned basis and layers", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 120–130, 2015.

[32] T. Kroeger, R. Timofte, D. Dai, and L. Van Gool, "Fast optical flow using dense inverse search", *Proceedings of the European Conference on Computer Vision*, pp. 471–488, 2016.

[33] T. Dozat, "Incorporating Nesterov momentum into Adam", 2015.

[34] H. Borchani, G. Varando, C. Bielza, and P. Larrañaga, "A survey on multi-output regression", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 5, no. 5, pp. 216–233, 2015.

[35] A. Yamaguchi and C. G. Atkeson, "Combining finger vision and optical tactile sensing: Reducing and handling errors while cutting vegetables", *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 1045–1051, 2016.

[36] J. Dargahi and S. Najarian, "Human tactile perception as a standard for artificial tactile sensing—a review", *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 1, no. 1, pp. 23–35, 2004.

Paper P2

Transfer learning for vision-based tactile sensing

Carmelo Sferrazza and Raffaello D'Andrea

Abstract

Due to the complexity of modeling the elastic properties of materials, the use of machine learning algorithms is continuously increasing for tactile sensing applications. Recent advances in deep neural networks applied to computer vision make vision-based tactile sensors very appealing for their high-resolution and low cost. A soft optical tactile sensor that is scalable to large surfaces with arbitrary shape is discussed in this paper. A supervised learning algorithm trains a model that is able to reconstruct the normal force distribution on the sensor's surface, purely from the images recorded by an internal camera. In order to reduce the training times and the need for large datasets, a calibration procedure is proposed to transfer the acquired knowledge across multiple sensors while maintaining satisfactory performance.

Published in *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

1. Introduction

The importance of the sense of touch in humans has been repeatedly shown to be fundamental even when all other sensing modalities are available, see for example [1] and [2]. Similarly, artificial tactile sensors can be of great help to robots for manipulation tasks, and in general for their interaction with the environment and with humans.

The aim of robotic tactile sensing systems is to make a robot capable of sensing the different types of forces, temperature changes and vibrations that its surface is subject to. However, compared to the human tactile sensing system, the state-of-the-art tactile sensors often address only a fraction of these capabilities [3], and are generally only tailored to specific tasks.

In the last decade, as a result of the increased use of machine learning, computer vision has progressed dramatically. This has enabled the use of cameras to sense the force distribution on soft surfaces, by means of the deformation that elastic materials undergo when subject to force. In this respect, acquisition devices that sense a variety of properties (i.e. force, vibrations) via direct physical contact, using a vision sensor to infer these properties from the change in light intensity or refractive index, are often referred to as optical tactile sensors. In particular, one class of optical tactile sensors relies on tracking the motion of markers distributed over a deformable surface to reconstruct the force distribution, see for example [4] and the references therein.

The approach discussed in this article is based on the sensor presented in [5]. The sensor exploits the information provided by the movement of spherical markers that are randomly distributed within the volume of a three-dimensional silicone gel. An off-the-shelf camera is used to track the pattern created by these markers inside the gel. In this way, an approximation of the strain field is obtained. Training data is collected with an automatic procedure, where the spherical tip of a needle is pressed against the gel at different positions and depths. During this procedure, the camera is used to collect images of the resulting pattern, while the ground truth normal force is measured with a force torque (F/T) sensor.

A neural network is trained with the labeled data to reconstruct the normal force distribution over the deformable sensor's surface. In order to reduce the training data necessary to achieve satisfactory performance on a different gel (with different hardness and marker distribution), a calibration layer is added to the existing architecture, and trained on a considerably smaller dataset. The resulting architecture achieves an accuracy comparable to the one attained by the original model on the larger dataset.

1.1 Related work

Sensing the location and the type of the different forces that a body is subject to is a key requirement to enable safe human-robot interaction. However, tactile sensing research is still far from achieving the same sort of comprehensive solution that cameras, with their high resolution and relatively small size and low cost, represent for visual sensing. Various categories of tactile sensors have been proposed in the literature, monitoring and

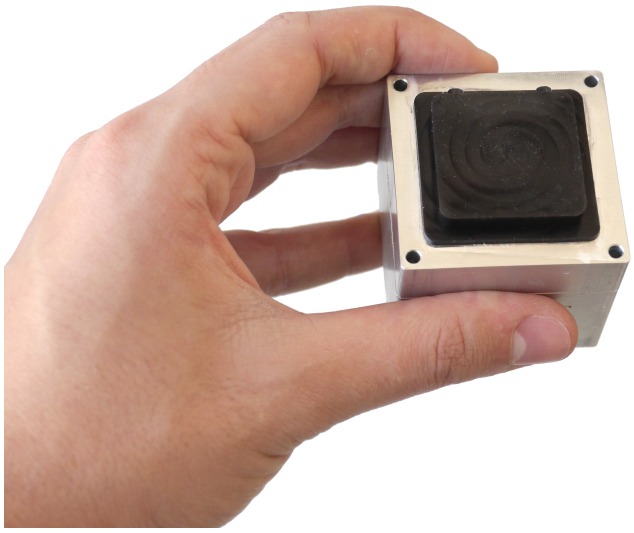


Figure 1. The experimental setup used for the evaluation presented in this article.

exploiting different properties of materials that change under contact with another body, of which an overview is given in [3] and [6].

Compared to other categories (e.g. tactile sensing arrays exploiting changes in capacitance or resistance), soft optical (or vision-based) tactile sensors combine low cost, ease of manufacture and minimal wiring. They intrinsically provide high spatial resolution (stemming from the high resolution of modern image sensors) and preserve the softness of their surface. These advantages come with the drawback of a larger size, which can be overcome by the use of multiple cameras (with a smaller footprint), as conceptually described in [7].

Vision-based tactile sensors generally observe the behavior of certain light quantities to infer the forces applied to the elastic surface of the sensor. As an example, in [8], a tactile sensor that uses the principle of total internal reflection is proposed, while in [9] photometric stereo is used to reconstruct the contact shape with a piece of elastometer through the use of differently colored lights.

In [10], a camera tracks the motion of spherical markers positioned in a grid pattern within a transparent elastic material. An analytical model is used to reconstruct the applied force distribution by approximating the problem with a semi-infinite elastic half-space. In particular, it is experimentally measured how the spread of the markers over two layers at different depths improves the robustness to errors in the computation of the displacement of these markers. A biologically-inspired design is presented and analyzed in [11] and [12], with markers arranged in a special pattern below the surface. The average displacement of these markers is related to the shear and normal forces applied to the contact surface.

However, the complexity of modeling the elastic properties of soft materials makes the derivation of a map from the monitored properties (i.e. marker displacement, changes in

light intensity) to force distribution very challenging, especially when a sensor is interfaced with different types of objects. For this reason, machine learning algorithms have recently been applied to the force reconstruction problem, see for instance [13] and [14].

In [4], a deep neural network is used to estimate the contact force on different types of objects, by using the same sensor as in [9] with the addition of a printed layer of markers on the surface. In [15], based on an automatic data collection procedure, a neural network estimates the force on a footpad while moving over certain trajectories.

Compared to most of the approaches cited, the method presented in [5] and discussed here does not rely on a special pattern of markers to be tracked by the camera, due to the particular choice of the features extracted from the images. In fact, the random distribution of the markers simplifies manufacture and makes this sensor suitable to adapt to arbitrary shapes and scalable to large surfaces through the use of multiple cameras. The chosen representation of the normal force distribution can handle different types of indenters and multiple points of contact. Moreover, it can easily be extended to sense shear forces. The algorithms proposed here provide an estimate of the normal force distribution at 60 Hz on a standard laptop (dual-core CPU, 2.80 GHz).

Transfer learning, see [16] for a survey, is an important topic to address when it becomes relevant to speed up learning. As an example, a data alignment mechanism is proposed in [17] to transfer the task knowledge between two different robot architectures. In the context of learning-based tactile sensing, collecting a dataset might be time consuming. The proposed design and modeling enable the transfer of the knowledge acquired on one tactile sensor across different gels with different marker patterns. To this purpose, a calibration procedure is presented in this paper, which considerably reduces the amount of data required for learning, as well as the training time.

1.2 Outline

The sensor design is discussed in Section 2. The training data collection procedure is described in Section 3 and the postprocessing of the data into features for the learning algorithm is presented in Section 4. In Section 5, the neural network architecture is explained and experimental results are presented. The calibration procedure is described in Section 6, while conclusions are drawn in Section 7.

2. Sensor design

The tactile sensor discussed here, schematically shown in Fig. 2, is made of a silicone gel (with a squared horizontal section of 32x32 mm) with spherical markers spread over its volume and a camera underneath to track the motion of these markers. The camera is an ELP USBFHD06H, equipped with a fisheye lens that has an angle of view of 180 degrees. The image frames are acquired at a resolution of 640x480 pixels and cropped to a region of interest of 440x440 pixels. The camera frame rate is 60 fps.

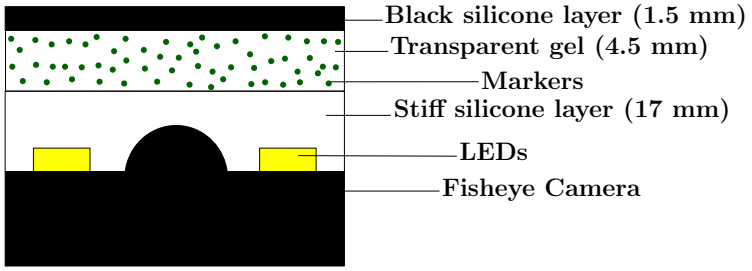


Figure 2. The full thickness of the experimental setup from the base of the camera to the surface is approximately 38 mm. The thickness of the different silicone layers is indicated in the figure.

The soft silicone gel is produced in a three-layer structure as described in [5], with the difference that smaller fluorescent green polyethylene microspheres (with a diameter of 150 to 180 μm) are used as markers to provide strain information at a finer spatial resolution. An indication of the thickness of the different layers is shown in Fig. 2.

The randomness of the marker positions over the gel's volume facilitates production. Conversely to sensing techniques that require a specific pattern in the marker distribution, the sensor discussed here is directly adaptable to any required shape.

Note that the design proposed shows a proof of concept and its dimensions have not been optimized. Nevertheless, the size is suitable for deployment to a robot gripper, as shown in [18] with a vision-based sensor of similar dimensions. The current sensor thickness is mainly determined by the fisheye lens and the stiff layer. The thickness of these components can be traded off against the field of view of the camera, i.e., smaller cameras with lower angles of view and placed closer to the sensor's surface would result in reduced thickness but smaller gel coverage. In this respect, the use of a higher number of cameras can increase the field of view while retaining a limited thickness. Alternatively, applications with very thin structures, e.g. robotic fingers, can be addressed by the use of mirrors [19]. In the context of robot skins, a coarser spatial resolution is generally required in sections of the body where space is of less relevance, i.e. humanoid robot trunk or arms, where tactile sensors are generally used for collision detection. As an example, the camera might be placed inside the robot trunk at a greater distance from the surface to cover a larger portion of the body.

3. Data collection

The neural network architecture needs a considerable amount of training data to be able to predict the normal force distribution with satisfactory accuracy. In order to automatize the data collection procedure, a precision milling and drilling machine (Fehlmann PICOMAX 56 TOP) with 3-axis computer numerical control (CNC) is used. A F/T sensor (ATI Mini40) is attached to the spindle of the machine to assign ground truth force labels to the data. A plexiglass plate, connected to the F/T sensor, serves as a holder for a spherical-ended cylindrical indenter that is approximately 40 mm long with a diameter

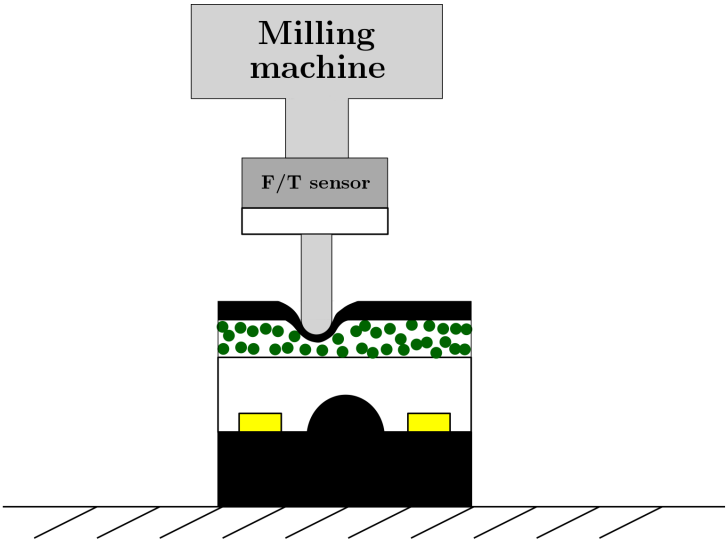


Figure 3. The indenter is controlled by a CNC milling machine and it is pressed against the gel while the data from the camera and the F/T sensor are recorded.

of 1.2 mm. The tactile sensor is clamped to the machine base, and the indenter is pressed and released over the entire surface of the gel at different depths to span a certain range of normal force values. The F/T readings are recorded while the camera streams the pictures at a fixed frequency. The milling machine provides a 24 V digital signal that is used to synchronize the data from the different sources. Fig. 3 shows the data collection setup.

4. Feature engineering

The task of reconstructing the normal force distribution on the surface of the tactile sensor can be formulated as a supervised learning problem, mapping the information obtained from the camera images to the applied force. To this purpose, the selection of relevant features is of great importance. Similarly, choosing an appropriate representation of the force distribution, that is, the labels of the supervised learning algorithm, is crucial.

In order to have a flexible representation, suitable to span the force fields generated by multiple objects with arbitrary shapes, the approach presented in [5] and used in this paper consists of the discretization of the gel's surface in n smaller bins, assigning a force value to each of these. The resulting n values for each data point are stacked together in a n-dimensional vector, which represents the label in that instance.

In the special case of the data collected with the automatic procedure described in Section 3, this vector has a very sparse structure. In fact, assuming that the surface in contact with the indenter lies entirely inside one bin, the value of the normal force applied by the tip of the needle on the gel's surface is encoded at the vector's component representing the bin that contains the point of application. The remaining vector components

are filled with zeros. An example of a label vector obtained from the automatic data procedure is shown in Fig. 4. Note that the previous assumption might be violated for indentation cases at the boundaries of the bins. In these cases, the representation above approximates the normal force distribution by assigning the entire measured force to the bin containing the indentation center.

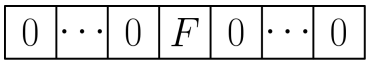


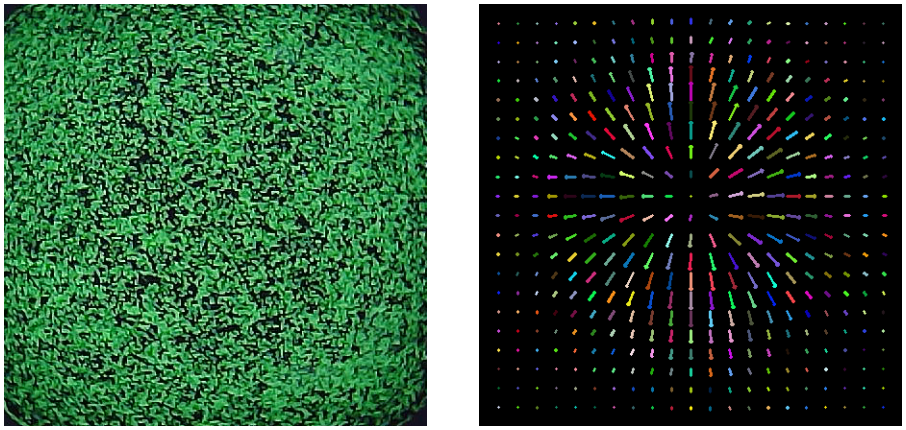
Figure 4. For the spherical indenter described in Section 3, the resulting normal force distribution is nonzero only at the bin that includes the indentation position, where it takes the value measured by the F/T sensor, here indicated with F.

With regard to the input to the force reconstruction model, the images are processed before being fed to the learning architecture for training or prediction. Conversely to directly taking the image pixels as features, extracting meaningful information from each image and creating a more compact set of features results in lower data requirements and shorter training times. In addition, having features that are likely to be invariant under the same force distribution, when extracted on separate gels with different marker patterns, is highly desirable. In this respect, this paper discusses an approach based on dense optical flow [20].

Dense optical flow algorithms aim to estimate the motion at each pixel of the image. The algorithm based on Dense Inverse Search (DIS), see [21], approaches this task by reconstructing the flow at multiple scales in a coarse-to-fine fashion. DIS employs an efficient search of correspondences, which renders the approach suitable to run in real-time. With respect to the application proposed here, the algorithm is independent of the marker type and distribution, since it only requires a trackable distinct pattern, which in the tactile sensor discussed in this paper is in fact given by the spherical markers.

The original RGB image is converted to grayscale, and the optical flow algorithm is applied. An example of the computed flow is shown in Fig. 5.

The resulting flow is stored at each pixel as tuples of magnitude and an angle with respect to a fixed axis. As in [5], the image plane is then divided into a grid of m regions with equal area and each optical flow tuple is assigned to one of these regions, depending on its position in the image. For each region $i = 1, \ldots, m$, the average magnitude and direction of the optical flow are computed. These average quantities have the advantage of implicitly storing position information (in fact, they are an approximation of the optical flow at each of the regions' centers). Furthermore, provided that a distinct pattern is available at each region, they are independent of the markers' distribution over the entire volume. Therefore, they are designed to be invariant when the same forces are applied to a different gel with the same material properties but not necessarily with the same marker pattern. The two average quantities for each of the m regions are chosen as the $2 \times m$ features, which are the input to the neural network architecture that is presented in Section 5.



(a) Original image **(b)** Optical flow

Figure 5. The RGB camera image, captured at rest in (a), is first converted to grayscale. The dense optical flow, shown in (b) at subsampled locations for an example indentation, is then computed with respect to the image at rest using the DIS-based algorithm.

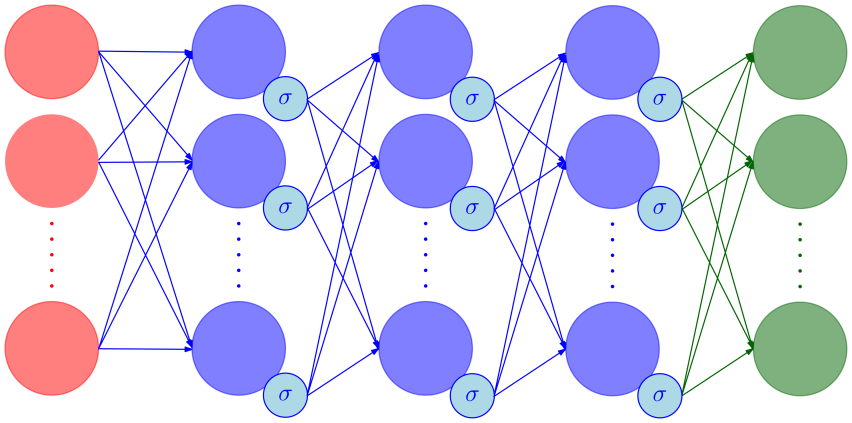


Figure 6. The DNN architecture. In red the input layer (with $2 \times m$ placeholders), in blue the hidden layers with the logistic activation functions, indicated with σ, in green the output layer (with n neurons), which has an identity activation function to perform the regression task. The biases are not shown in the figure.

5. Model training

5.1 Learning architecture

A deep neural network (DNN) is designed to estimate the normal force distribution applied to the tactile sensor's surface, given the features extracted from the optical flow. A feedforward architecture with three fully connected hidden layers, all using a logistic function as the activation unit, is chosen. The input layer reads the $2 \times m$ features described in Section 4, while the output layer returns the predicted n-dimensional output, which assigns a force value to each of the sensor's surface bins. A scheme summarizing the chosen architecture, which exhibits relatively fast training times, is shown in Fig. 6.

The loss used to train the DNN is the average root mean squared error ($aRMSE$, see

[22]), that is,

$$aRMSE = \frac{1}{n}\sum_{i=0}^{n\text{-}1}\sqrt{\frac{\sum_{l=0}^{N_{\text{set}}\text{-}1}\left(y_i^{(l)} - \hat{y}_i^{(l)}\right)^2}{N_{\text{set}}}}, \tag{1}$$

where $y_i^{(l)}$ and $\hat{y}_i^{(l)}$ are the i-th components of the true and the predicted l-th label vector, respectively, and N_{set} is the number of samples in the set that is being evaluated (i.e. training, validation, test sets). The optimization method used to train the network is RMSProp with Nesterov momentum (known as Nadam, see [23]). In order to prevent overfitting, dropout layers, see [24], are added after each of the hidden layers and are used during the training phase. Moreover, a portion of the training data is selected as a validation set, and the loss computed on this set is used to early stop the optimization when this loss does not decrease for N_{es} consecutive epochs.

5.2 Evaluation

A dataset is collected using the automatic procedure presented in Section 3. The needle is pressed on the square surface at a set of positions described by a grid with equal spacings of 0.75 mm. The tip of the indenter reaches eight different depth levels, from 0.25 mm to 2 mm. This procedure gives 10952 data points, with the normal force measured by the F/T sensor up to 1 N.

The other parameters used in this experiment are summarized in Table 1.

Symbol	Value	Description
m	1600	# of averaging regions in the image
n	81	# of bins that grid the sensor surface
-	200	training batch size
-	0.001	learning rate
N_{es}	50	early stopping parameter
-	0.15	dropout rate
-	(800,400,400)	hidden layers' size

Table 1. Parameters used to train the DNN architecture.

Before training, 20% of the dataset is put aside as a test set, that is subsequently used to evaluate the results.

Given the sparse nature of the label vectors, the evaluation of the results is discussed with respect to some quantities (additionally to the $aRMSE$) that are particularly intuitive for this application. These measures have been introduced in [5] and capture the performance of the sensor in predicting the correct location and magnitude of the force applied with the needle used for this experiment. For $l = 0, 1, \ldots, N_{\text{set}} - 1$, the following indexes are computed,

$$k_l = \underset{i=0,1,\ldots,n-1}{\arg\max} |y_i^{(l)}|, \qquad \hat{k}_l = \underset{i=0,1,\ldots,n-1}{\arg\max} |\hat{y}_i^{(l)}|.$$

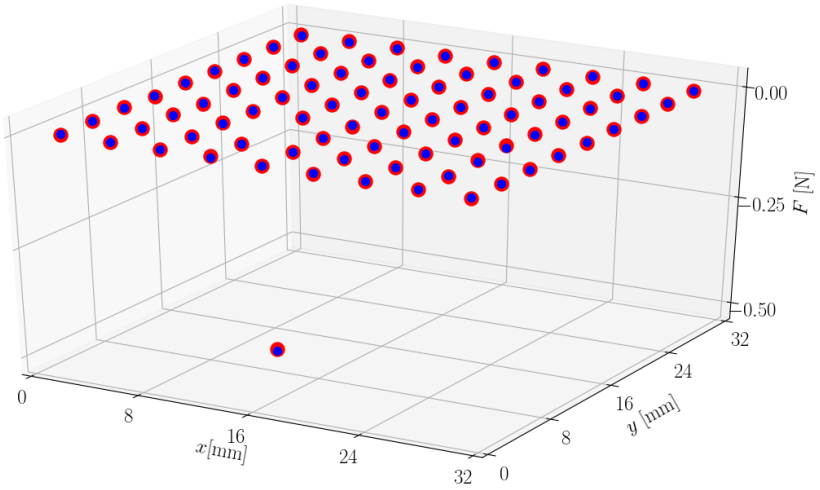


Figure 7. The predicted normal force distribution (in red) for a sample in the test set, compared to the ground truth (in blue). x and y are the two Cartesian axes spanning the gel's surface, with the origin at one corner. The sign of the normal force F is defined to be negative when directed towards the camera.

An error metric based on the distance between the maximum components of the true and estimated label vectors is then introduced as,

$$d_{\text{loc}} = \frac{1}{N_{\text{set}}} \sum_{l=0}^{N_{\text{set}}-1} \|c(k_l) - c(\hat{k}_l)\|_2, \tag{2}$$

where $c(k)$ denotes the location of the center of the bin k on the surface. Similarly, an error that only considers the magnitude of the maximum components of the true and estimated label vectors is defined as,

$$RMSE_{\text{mc}} = \sqrt{\frac{1}{N_{\text{set}}} \sum_{l=0}^{N_{\text{set}}-1} \left(y_{k_l}^{(l)} - \hat{y}_{\hat{k}_l}^{(l)}\right)^2}. \tag{3}$$

The results of the trained model evaluated on the test set are summarized in Table 2. An example of the predicted normal force distribution is shown in Fig. 7.

Criterion	Value	Unit
$aRMSE$	0.009	[N]
d_{loc}	0.107	[mm]
$RMSE_{\text{mc}}$	0.065	[N]

Table 2. Evaluation of the trained model on the test set.

6. Calibration

Collecting large datasets for each sensor is time consuming, even if automatized as described in Section 3. Furthermore, a model that has been trained to reconstruct the normal force distribution on a particular gel may not yield the same prediction performance on another gel. Despite the sensor design and the invariance of the chosen features, there are still other variables that are specific for each sensor and might not have been accounted for.

Modeling the gel as a linearly elastic half-space, the magnitude of the 3D elastic displacement u of a marker is related to the concentrated normal force F applied to the sensor's surface by a proportional factor, that is,

$$u = h(E)F, \tag{4}$$

where h depends on the hardness of the material, through its Young's modulus E. A formal derivation of this fact is given by the Bousinnesq solution [25, p. 50]. However, the resulting hardness of the gel's surface is sensitive to the actual percentage of the two components that are mixed together to produce it. When the model trained on one gel is applied to a gel with a different hardness (i.e. mixing ratio), an appropriate transformation of the marker displacements observed in the images is necessary to attain comparable performance. Nevertheless, it is not straightforward to perform this operation due to the influence of the camera model on this transformation and to the presence of noise introduced by the camera acquisition and the optical flow computation.

In addition, the relative position of the gel with respect to the camera is crucial. In fact, the sensitivity of a marker's position in the image (with respect to a fixed image coordinate frame) to changes in the distance z from the lens is inversely proportional to the square of z. That is, from the pinhole camera equations (see [26, p. 49]),

$$p \propto \frac{1}{z} \Rightarrow \frac{\partial p}{\partial z} \propto \frac{1}{z^2}, \tag{5}$$

where p is the distance of the marker in the image from the origin of the image coordinate frame. Therefore, given the small distance of the markers from the fisheye lens, relatively small differences in the thickness of the materials across multiple gels (that are introduced during the production and assembly of the sensor) result in considerably different observed displacements.

Moreover, if a different camera is used, its intrinsic parameters have an effect on how the marker displacements are projected onto the image plane. This problem can be solved by undistorting the images, a procedure that may however introduce other inaccuracies in compensating for the considerable distortion of the lens.

Considering that the differences mentioned are all at the level of the features, a preprocessing fully connected layer with a rectified linear unit (ReLU) activation function is

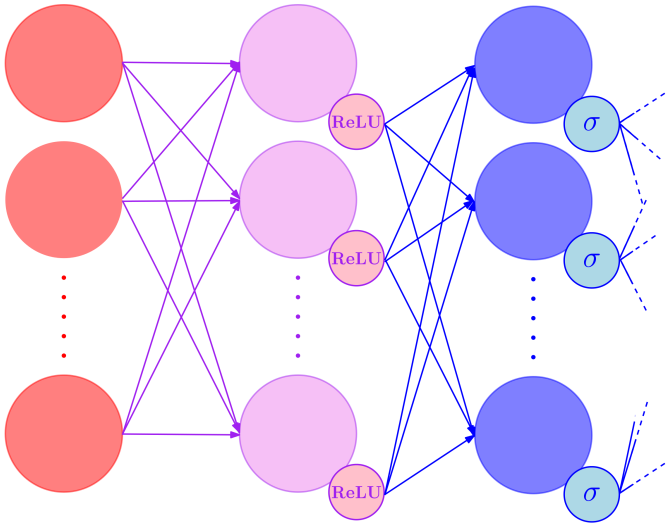


Figure 8. The input layer is directed towards a calibration layer (in violet) with $2 \times m$ neurons, which is then connected to the hidden layers of the original architecture.

added to the previously trained DNN architecture between the input and the first hidden layer. The weight matrix corresponding to this preprocessing layer is initialized with the identity, which leads to a considerable speed up in learning. In fact, the differences across sensors are expected to contribute to rather small deviations from a diagonal structure of the weight matrix.

A considerably smaller number of data points is collected on a new gel (over a coarser grid), and the augmented architecture (of which a snippet is shown in Fig. 8) is then trained on this dataset by freezing all the weights apart from the ones belonging to the added preprocessing layer. In this way, the training time and the data requirements are greatly reduced, while retaining comparable performance to the one obtained in Section 5. The parameters used for this procedure are summarized in Table 3 and the results are shown in Fig. 9, for different sizes of the training set.

Symbol	Value	Description
-	64	size of training batches
-	0.0001	learning rate
N_{es}	200	early stopping parameter
-	0.05	dropout rate

Table 3. Parameters used to train the calibration layer

Note that the success of this rather simple calibration technique is mainly due to the choice of the features described in Section 4. As opposed to learning the force reconstruction task directly from the pixel values, the averaged optical flow features are in fact invariant across different gels, except for the alignment and scaling factors mentioned above.

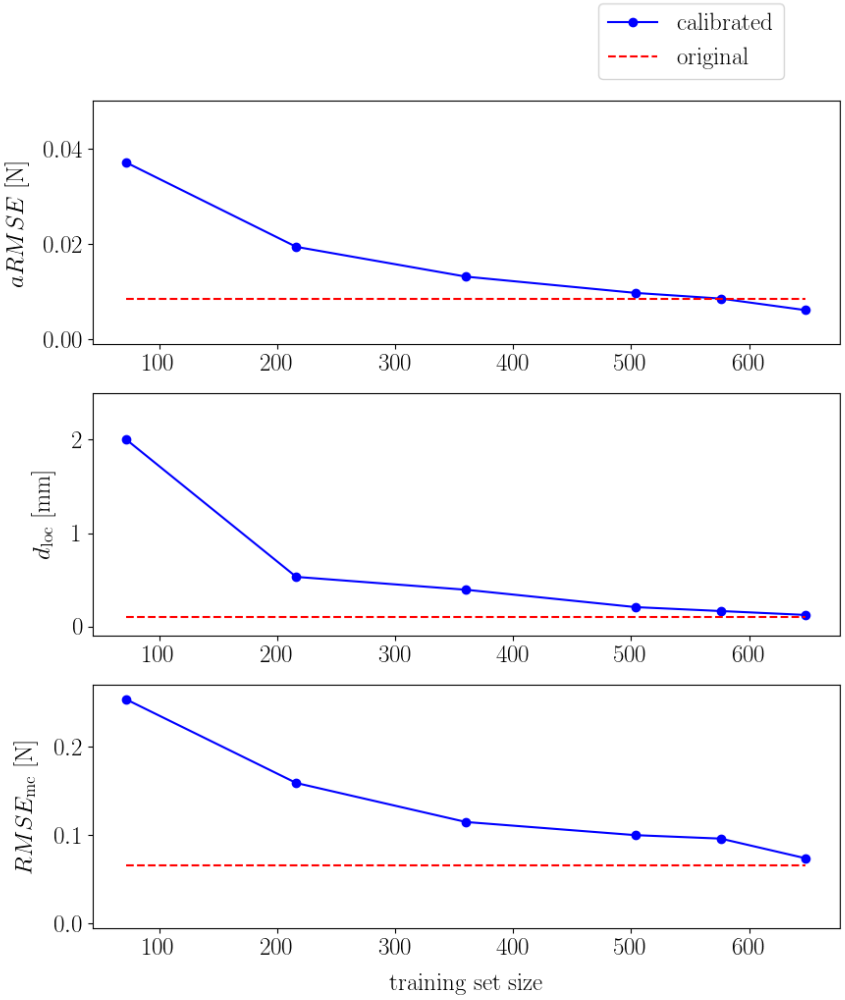


Figure 9. A dataset of 800 samples is collected on the sensor that undergoes the calibration procedure. Portions of different sizes are used as training sets, while the remaining data serve as a test set for evaluation. The different metrics show that the augmented architecture (in blue) attains comparable performance to the experiment presented in Section 5 (in red), where the network was trained with a much larger dataset (7884 data points). Training both the original and the augmented architecture with the smaller dataset from scratch yields a substantially inferior performance, and the results are therefore not shown in this plot. The same applies to predicting the normal force distribution on the new gel with the network trained in Section 5 (on the first gel, before calibration).

The algorithms presented in this paper yield a real-time execution of 60 Hz, with the implementation not particularly optimized for efficiency. An overview of the time needed for a single execution of the main steps of the pipeline is shown in Table 4.

7. Conclusion

An approach to sense the normal force distribution on a soft surface has been discussed. A learning algorithm has been applied to a vision-based tactile sensor, which is inexpensive

Component	Time (ms)
Image acquisition and cropping	1
Optical flow computation	9
Feature generation	2
Prediction	2

Table 4. Average times for real-time pipeline

and easy to manufacture. The proposed strategy is scalable to arbitrary surfaces and therefore suitable for robot skin applications.

The results (see the video accompanying this paper[1]) show that the sensor can reconstruct the normal force distribution applied with a test indenter after being trained on an automatically collected dataset. Note that the learning problem discussed here is a multiple multivariate regression, which maps multi-dimensional feature vectors to multi-dimensional label vectors. The DNN architecture is able to predict the force applied to each of the surface bins, which can either be zero or the one actually applied with the point indenter. Conversely to regression techniques that separately predict each output, a feedforward neural network can capture the interrelations between the different output label components. Since the network does not directly use knowledge of the point indentation, this strategy is therefore appealing for more general contacts (e.g. with larger indenters or multiple contacts) that will be the subject of future work.

The tactile sensing pipeline proposed here estimates the static force distribution from the images captured by a camera. The training datasets employed in this paper capture measurements taken after the indentation has reached steady-state. Nevertheless, dynamic indentations are reconstructed with a limited loss of accuracy, as shown in the accompanying video. As an example, a detailed analysis of hysteresis effects has the potential of further improving the predictions in both the loading and unloading phases.

In order to speed up the learning and the training data collection, the variations across different sensors of the same type have been identified at the level of the features. Therefore, this paper has proposed a calibration procedure that accordingly modifies the input to the learning architecture, thus transferring the knowledge across different gels.

Acknowledgments

The authors would like to thank Michael Egli and Marc-André Corzillius for their support in the manufacturing of the sensor, and Laura Gasser for her contribution to the feature engineering.

[1]Video: `https://youtu.be/CdYK5I6Sccw`

References

[1] G. Westling and R. S. Johansson, "Factors influencing the force control during precision grip", *Experimental brain research*, vol. 53, no. 2, pp. 277–284, 1984.

[2] J. B. F. van Erp and H. A. H. C. van Veen, "Touch down: The effect of artificial touch cues on orientation in microgravity", *Neuroscience Letters*, vol. 404, no. 1-2, pp. 78–82, 2006.

[3] R. S. Dahiya, G. Metta, M. Valle, and G. Sandini, "Tactile sensing - from humans to humanoids", *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 1–20, 2010.

[4] W. Yuan, S. Dong, and E. H. Adelson, "GelSight: High-resolution robot tactile sensors for estimating geometry and force", *Sensors*, vol. 17, no. 12: 2762, 2017.

[5] C. Sferrazza and R. D'Andrea, "Design, motivation and evaluation of a full-resolution optical tactile sensor", *Sensors*, vol. 19, no. 4: 928, 2019.

[6] M. I. Tiwana, S. J. Redmond, and N. H. Lovell, "A review of tactile sensing technologies with applications in biomedical engineering", *Sensors and Actuators A: Physical*, vol. 179, pp. 17–31, 2012.

[7] A. Yamaguchi and C. G. Atkeson, "Combining finger vision and optical tactile sensing: Reducing and handling errors while cutting vegetables", in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2016, pp. 1045–1051.

[8] M. Koike, S. Saga, T. Okatani, and K. Deguchi, "Sensing method of total-internal-reflection-based tactile sensor", in *Proceedings of the IEEE World Haptics Conference*, 2011, pp. 615–619.

[9] M. K. Johnson, F. Cole, A. Raj, and E. H. Adelson, "Microgeometry capture using an elastomeric sensor", *ACM Transactions on Graphics*, vol. 30, no. 4, 46:1–46:8, 2011.

[10] K. Kamiyama, H. Kajimoto, N. Kawakami, and S. Tachi, "Evaluation of a vision-based tactile sensor", in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, 2004, pp. 1542–1547.

[11] C. Chorley, C. Melhuish, T. Pipe, and J. Rossiter, "Development of a tactile sensor based on biologically inspired edge encoding", in *Proceedings of the International Conference on Advanced Robotics*, 2009, pp. 1–6.

[12] B. Winstone, G. Griffiths, C. Melhuish, T. Pipe, and J. Rossiter, "TACTIP — tactile fingertip device, challenges in reduction of size to ready for robot hand integration", in *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, 2012, pp. 160–166.

[13] O. Kroemer, C. H. Lampert, and J. Peters, "Learning dynamic tactile sensing with robust vision-based training", *IEEE Transactions on Robotics*, vol. 27, no. 3, pp. 545–557, 2011.

[14] M. Meier, F. Patzelt, R. Haschke, and H. J. Ritter, "Tactile convolutional networks for online slip and rotation detection", in *Proceedings of the International Conference on Artificial Neural Networks*, vol. 9887, 2016, pp. 12–19.

[15] M. Y. Chuah and S. Kim, "Improved normal and shear tactile force sensor performance via least squares artificial neural network (LSANN)", in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2016, pp. 116–122.

[16] S. J. Pan, Q. Yang, *et al.*, "A survey on transfer learning", *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[17] B. Bocsi, L. Csató, and J. Peters, "Alignment-based transfer learning for robot models", in *Proceedings of the International Joint Conference on Neural Networks*, 2013, pp. 1–7.

[18] S. Dong, W. Yuan, and E. H. Adelson, "Improved gelsight tactile sensor for measuring geometry and slip", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 137–144.

[19] E. Donlon, S. Dong, M. Liu, J. Li, E. Adelson, and A. Rodriguez, "Gelslim: A high-resolution, compact, robust, and calibrated tactile-sensing finger", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 1927–1934.

[20] D. Fleet and Y. Weiss, "Optical flow estimation", in *Handbook of Mathematical Models in Computer Vision*, N. Paragios, Y. Chen, and O. Faugeras, Eds., Springer, 2006.

[21] T. Kroeger, R. Timofte, D. Dai, and L. Van Gool, "Fast optical flow using dense inverse search", in *Proceedings of the European Conference on Computer Vision*, 2016, pp. 471–488.

[22] H. Borchani, G. Varando, C. Bielza, and P. Larrañaga, "A survey on multi-output regression", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 5, no. 5, pp. 216–233, 2015.

[23] T. Dozat, "Incorporating Nesterov momentum into Adam", 2015.

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[25] K. L. Johnson, *Contact mechanics*. Cambridge university press, 1987.

[26] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

Paper P3

Ground truth force distribution for learning-based tactile sensing: a finite element approach

Carmelo Sferrazza, Adam Wahlsten, Camill Trueeb and Raffaello D'Andrea

Abstract

Skin-like tactile sensors provide robots with rich feedback related to the force distribution applied to their soft surface. The complexity of interpreting raw tactile information has driven the use of machine learning algorithms to convert the sensory feedback to the quantities of interest. However, the lack of ground truth sources for the entire contact force distribution has mainly limited these techniques to the sole estimation of the total contact force and the contact center on the sensor's surface. The method presented in this article uses a finite element model to obtain ground truth data for the three-dimensional force distribution. The model is obtained with state-of-the-art material characterization methods and is evaluated in an indentation setup, where it shows high agreement with the measurements retrieved from a commercial force-torque sensor. The proposed technique is applied to a vision-based tactile sensor, which aims to reconstruct the contact force distribution purely from images. Thousands of images are matched to ground truth data and are used to train a neural network architecture, which is suitable for real-time predictions.

Published in *IEEE Access*.

1. Introduction

A growing number of applications require robots to interact with the environment [1] and with humans [2]. The use of soft materials for robotics applications [3] introduces intrinsic safety during interactive tasks [4]. In addition, precise estimation of contact forces is crucial for effective operation without damaging the robot's surroundings, e.g., for manipulation of fragile objects [5].

Modeling the interaction of soft materials with generic objects is highly complex. As a consequence, several tactile sensing strategies leverage the use of machine learning algorithms to map sensory feedback to the corresponding quantities of interest, e.g., contact forces, shape and materials, see [6]–[8]. These maps are generally retrieved by means of supervised learning techniques, which fit a model to a large amount of labeled data, i.e., sensory data paired with the corresponding ground truth.

However, the estimation of the full contact force distribution purely from data is limited by the lack of a ground truth source that does not alter the interaction between the soft material and the objects in contact. This article aims to provide a systematic way of labeling data with ground truth for the three-dimensional force distribution, which is obtained in simulation through the finite element method (FEM).

The approach is evaluated on a vision-based tactile sensor, originally presented in [9], which uses a camera to track spherical particles within a transparent gel. Hyperelastic models of the sensor's materials are retrieved from state-of-the-art material characterization tests, which are fully independent of the evaluation experiments. A label vector representing the ground truth force distribution is assigned to each image collected during an automatic indentation procedure. The total contact force also retrieved from the FEM simulations shows a satisfactory agreement with the measurements obtained from a commercial force-torque (F/T) sensor.

The dataset generated with the strategy proposed here is then used to train a deep neural network (DNN) architecture [10], which maps optical flow features to the contact force distribution. The evaluation of this strategy is carried out in a specific indentation setup, with the resulting pipeline running in real-time on the CPU of a standard laptop computer (dual-core, 2.80 GHz) at 40 Hz.

1.1 Related work

In recent decades, tactile sensing research has shown the potential of providing robots with the sense of touch, exploited both singly [11] or in combination with vision [12].

Among the various categories of tactile sensors, see [13], [14] for a survey, vision-based (or optical) tactile sensors are based on optical devices that monitor properties related to the contact between the sensor's surface (generally soft) and the environment. Among the advantages of this type of tactile sensors are high resolution, low cost, ease of manufacture and the preservation of the surface softness.

One category of optical tactile sensors uses a camera to track sparse markers within a soft, transparent gel, which deforms when subject to external forces, see for example

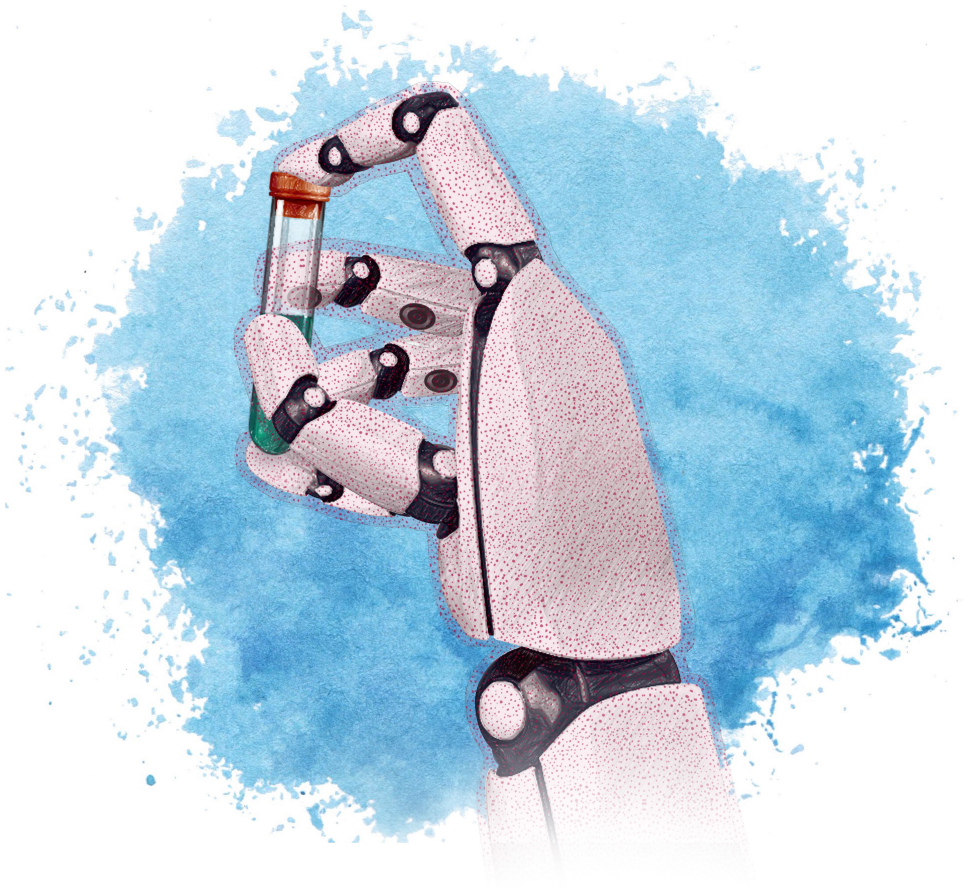


Figure 1. The tactile sensing technique presented in [9] is suitable to cover large surfaces of arbitrary shape and dimension. A concept of a robotic hand covered with cameras and a transparent gel embedding a spread of red particles is shown in this figure.

[15], [16]. Other optical devices are able to provide information about the contact with the environment, as shown with the use of dynamic vision sensors [17] and depth cameras [18]. The sensor used here for the evaluation of the proposed approach is based on an RGB camera (which retains a small size) that tracks a dense spread of particles within a soft gel, see for example Fig. 1. This design is presented in [9] and shows performance advantages over sparse marker tracking, and ease of manufacture, without any assumptions about the surface shape.

Converting tactile information to quantities, such as the force distribution, which are of high relevance for many robotics tasks (e.g., grasping or manipulation), is not trivial. In fact, the main complexity is introduced by the absence of a generally valid closed-form model, which maps the deformation of a hyperelastic material to the external forces applied to it. The use of data-driven techniques aims to overcome this problem, approximating this map with a model learned from a collection of past data. In [19], an optical tactile sensor that exploits photometric stereo and markers painted on its soft surface is used to reconstruct the total contact force by means of a neural network architecture. In [20], an array of light emitters and receivers is placed below a soft gel

to create tactile information, which is then provided to machine learning algorithms that reconstruct the location and the depth of an indentation, as well as the type of the employed indenter. Although these techniques generally require large datasets, transfer learning techniques can reuse information extracted across different sensors, as shown in [21].

The FEM [22] is a powerful numerical technique that provides approximate solutions of boundary value problems arising in engineering, by subdividing a large system into many smaller parts (called elements). One of the widespread applications of this technique is the analysis of the behavior of soft materials under various loading conditions. In [23], the silicone gel pad of an optical tactile sensor is modeled as a linear elastic material, and the FEM is used to compute the stiffness matrix that approximates the relation between external forces and displacements of the sensor's material. Based on reconstructed surface displacements, this matrix is then used to compute an estimate of the force distribution applied to the sensor. FEM simulations of a flexible 3D shape sensor are used in [24] to optimize design parameters. Furthermore, these simulations show the uniqueness of a strain-to-shape mapping for the case considered. In [25], theoretical justifications based on FEM equations are provided to reconstruct the external forces applied to soft bodies using information about their deformation.

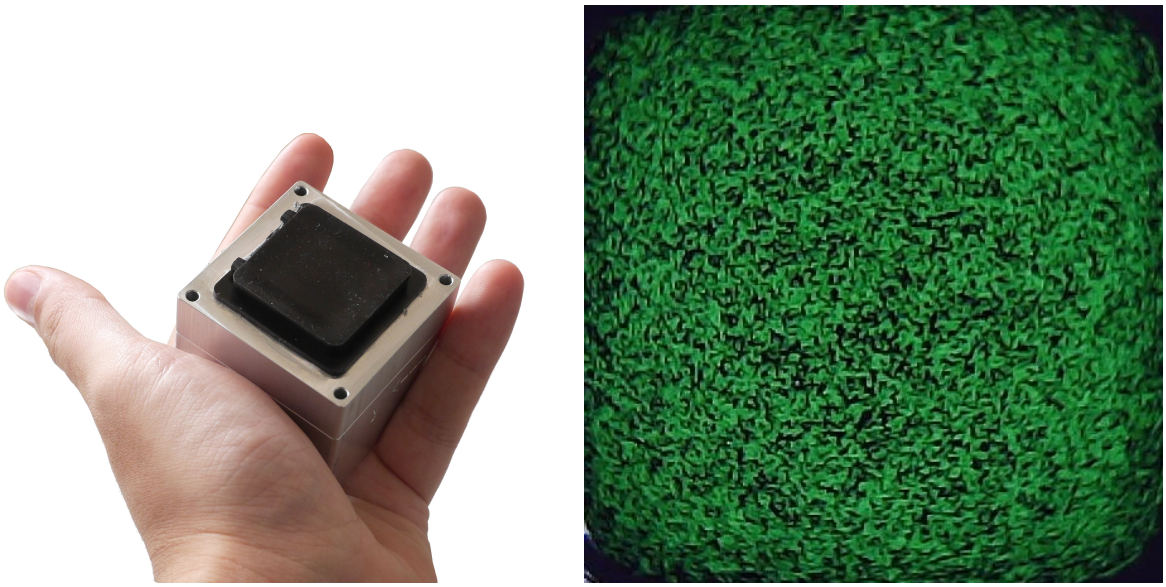
The strategy followed in this article exploits FEM simulations to obtain ground truth data for the contact force distribution applied to the soft surface of a tactile sensor. The lack of ground truth data has so far prevented the development of learning-based tactile sensors that predict the full force distribution, limiting them to the estimation of simpler quantities, e.g., the resultant force and its location, and the depth of a contact. The hyperelastic models identified capture the full material behavior, including nonlinearities, rendering highly accurate simulations. Images collected in experiments on a vision-based tactile sensor are matched to the ground truth and used to train a DNN that reconstructs the force distribution with high accuracy and in real-time.

1.2 Outline

The sensing principle and the hardware used for the evaluation are presented in Section 2, while the material characterization is discussed in Section 3. In Section 4, the dataset generation is described, from the collection of images to the approach proposed for assigning ground truth labels. The learning algorithm and the results are presented in Section 5. Section 6 concludes the article with a brief discussion.

2. Hardware

The approach discussed in this article for generating ground truth labels is evaluated on a vision-based tactile sensor. The tactile sensing strategy is presented in [9], and is based on tracking the movement of spherical particles, which are randomly spread within a soft



(a) Sensor protoype **(b)** Particles image

Figure 2. The prototype of the sensor (designed for desktop testing) is shown in (a). The dense spread of green particles is captured by the camera placed inside the aluminum part. The resulting RGB image at a state of zero force is shown in (b).

gel placed in front of a camera. The prototype used for the experiments and the camera image at a state of zero force are shown in Fig. 2.

The soft material is produced in a three-layer structure, as depicted in Fig. 3. From the bottom (which touches the camera lens) to the top surface, the following materials are used: 1) a stiffer layer (ELASTOSIL® RT 601 RTV-2, mixing ratio 7:1, shore hardness 45A); 2) the soft gel (Ecoflex™ GEL, mixing ratio 1:1, shore hardness 000-35) embedding the particles, which comprise 1.96 % of the layer volume; 3) a black surface layer (ELASTOSIL® RT 601 RTV-2, mixing ratio 25:1, shore hardness 10A). After curing, the complete sensor is placed in an oven at 60 °C for 8 hours. This step has the effect of reducing the aging of the materials, which is discussed in further detail in Section 3.

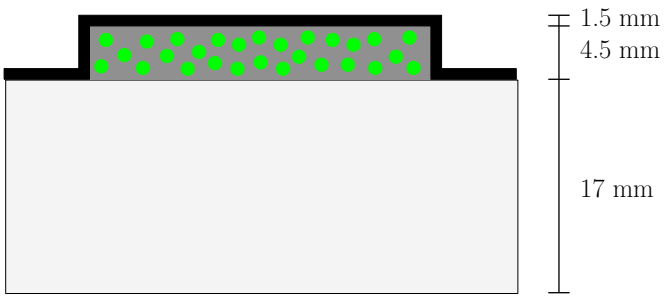


Figure 3. A scheme of the three-layer structure that composes the soft material. The thickness of the different layers is shown in the figure above. This structure yields a top surface of 32x32 mm.

3. Material characterization

Finite element analysis (FEA) of arbitrary contact interactions with the sensor's soft surface requires material models that account for geometrical and material nonlinearities. Soft elastomers are often modeled as hyperelastic materials [26], and finding a suitable model formulation and corresponding parameters generally necessitates experimental data from both uniaxial and biaxial stress states [27], [28]. To this end, a large-strain multiaxial characterization of the two most compliant materials, the Ecoflex GEL and the Elastosil 25:1, is performed. Samples of both materials are tested in uniaxial tension (UA), pure shear (PS), and equibiaxial tension (EB) based on previously described protocols [28]. The bottom layer of Elastosil with the mixing ratio 7:1 is considerably stiffer than the soft adjacent Ecoflex GEL, see Section 3.5, and is therefore modeled as rigid in the subsequent FEA.

3.1 Sample preparation

Thin material sheets of each elastomer are prepared as described in Section 2, and cast to a nominal thickness of 0.5 mm. Test pieces are cut to obtain gauge dimensions (length × width) of 40 mm × 10 mm for UA, 10 mm × 60 mm for PS, and a diameter of 30 mm for membrane inflation tests (EB). The test pieces of Ecoflex GEL used for the mechanical characterization do not contain the spherical particles, i.e. the pure material behavior is tested. An ink pattern is applied to the sample surface to facilitate optical strain analysis [28]. After each experiment, the sample thickness h_0 is measured on cross-sections cut from the central region using a confocal microscope (LSM 5 Pascal, Carl Zeiss AG) with a 10× objective in brightfield mode.

3.2 Mechanical testing

UA and PS tests are performed on a tensile testing set-up (MTS Systems) consisting of horizontal hydraulic actuators, 50 N force sensors, and a CCD-camera (Pike F-100B, Allied Vision Technologies GmbH) equipped with a 0.25× telecentric lens (NT55-349, Edmund Optics Ltd.) that captures top-view images of the deforming test piece. Displacement-controlled monotonic tests are performed up to a specified nominal strain (Ecoflex GEL: 200 %; Elastosil 25:1: 100 %) at a nominal strain rate of 0.3 %/s. The strain-rate dependence is analyzed in an additional UA test, where the sample is loaded cyclically with strain rates increasing from 0.1 %/s up to 10 %/s.

An EB state of tension is realized in a pressure-controlled membrane inflation test (see [28] for details). Briefly, a thin, circular sample is clamped on top of a hollow cylinder and inflated by means of a syringe pump (PhD Ultra, Harvard Apparatus), while a pressure sensor (LEX 1, Keller AG) measures the inflation pressure p. Top and side-view images are recorded with CCD cameras (GRAS-14S5C-C, Point Grey Research), and the image sequences are used for evaluating the in-plane deformation at the apex and the apex radius of curvature r, respectively.

All experiments are performed at room temperature and on the same day as completed curing. The mechanical properties of soft elastomers are known to change with aging [28], [29], a process attributed to additional, thermally activated cross-linking [29]. To assess the influence of aging, additional UA test pieces of the same sheets were kept at room temperature and tested several weeks after fabrication.

3.3 Experimental data analysis

Since nominal strains computed from the clamp displacements are prone to errors due to sample slippage [30], the local in-plane principal stretches in the center of the test-piece, λ_1 and λ_2, are computed from the top-view image sequences using a custom optical flow-tracking algorithm [28]. The principal stretch in thickness direction is calculated by assuming material incompressibility, i.e., $\lambda_3 = 1/(\lambda_1\lambda_2)$. In the UA and PS configurations, the Cauchy stress in loading direction is evaluated as $\sigma = F\lambda/(w_0 h_0)$, where F are the measured force values, $\lambda := \lambda_1$ is the principal stretch in loading direction, and w_0 is the reference width of the test piece. For inflation tests, the measured inflation pressure and apex radius of curvature can be used to approximate the equibiaxial Cauchy stress at the apex as $\sigma = pr/(2h_0\lambda_3)$, which holds for $h_0 \ll r$ [31].

3.4 Constitutive models

The experimental data are used to fit the parameters of a hyperelastic, incompressible Ogden model [31], for which the strain-energy density per unit reference volume reads

$$W = \sum_{k=1}^{K} \frac{\mu_k}{\alpha_k} \left(\lambda_1^{\alpha_k} + \lambda_2^{\alpha_k} + \lambda_3^{\alpha_k} - 3\right), \ \lambda_1\lambda_2\lambda_3 = 1. \tag{1}$$

The material parameters μ_k, α_k must satisfy the constraint $\mu_k\alpha_k > 0$, $k = 1, 2, \ldots, K$, and can be used to calculate the corresponding Young's modulus as $E = (1+\nu)\sum_{k=1}^{K} \mu_k\alpha_k$, with $\nu = 0.5$ being the Poisson's ratio of an isotropic incompressible material. The principal Cauchy stresses immediately follow from (1) as (see [31], p. 571)

$$\sigma_i = \sum_{k=1}^{K} \mu_k \lambda_i^{\alpha_k} - q, \quad i = 1, 2, 3, \tag{2}$$

where q is an arbitrary hydrostatic pressure arising due to the incompressibility constraint, whose value depends on the boundary conditions. By specializing (2) to the three experimentally considered load cases (see [31]), the analytical formulas were used to minimize the squared error between the model and the experiments using the minimization routine `fmincon` available in MATLAB (R2018b, The MathWorks, Inc.). Ogden models of order $K = 2$ were found to provide the best description of the data for both materials compared to neo-Hookean or Mooney–Rivlin formulations; the resulting parameter sets are reported in Table 1.

Material	μ_1 [kPa]	α_1 [-]	μ_2 [kPa]	α_2 [-]
Ecoflex GEL	7.9652	1.2769	0.3093	3.5676
Elastosil 25:1	85.1168	2.8991	−0.0020	−8.2915

Table 1. Material parameters of Ogden's model for the Ecoflex GEL and the Elastosil 25:1

3.5 Results

The individual stress-stretch curves for each sample of the two elastomers tested are reported in Fig. 4, together with the sample averages and the model predictions. Both models identified provide an excellent description of the mechanical behavior over the whole range of deformation for all three load cases. The additional UA tests suggest a negligible influence of both strain rate and shelf time (over 5 weeks) on the mechanical behavior of the Elastosil 25:1 for the rates and times tested. However, the Ecoflex GEL shows a dependence on both strain rate and aging, see Fig. 12, in the Appendix A. These dependencies, as well as potential softening phenomena upon cyclic loading (Mullins effect), are neglected in the hyperelastic model.

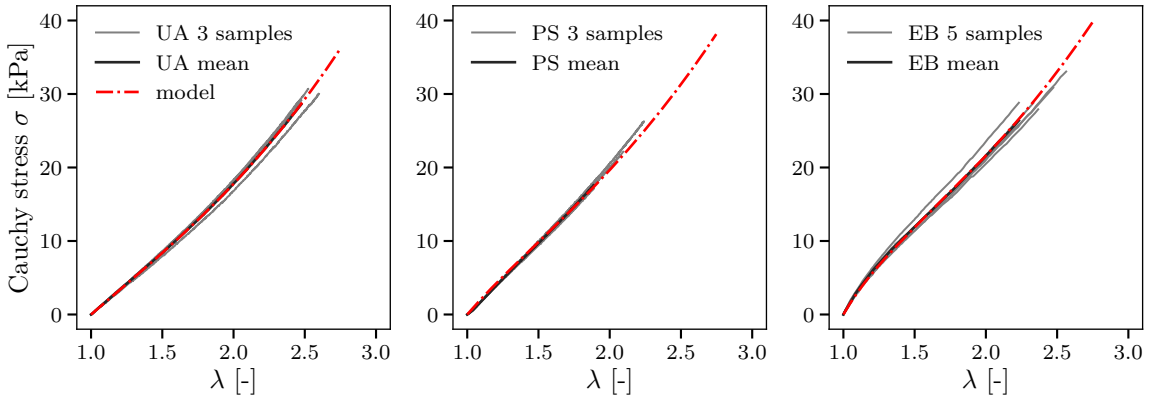
Corresponding Young's moduli (calculated using the Ogden model coefficients) of the Ecoflex GEL and the Elastosil 25:1 are 16.9 kPa and 370.2 kPa, respectively. For comparison, the Young's modulus of the stiffer Elastosil 7:1, determined by microindentation tests (FT-MTA02, FemtoTools AG), is found to be 0.97 MPa, i.e. more than 50 times stiffer than the Ecoflex GEL.

4. Generating a dataset

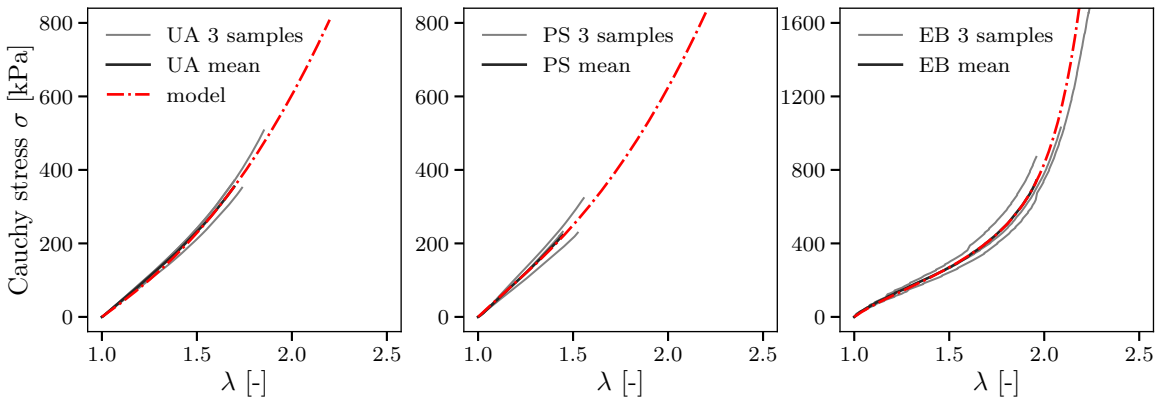
The task of mapping the information extracted from the images to the applied contact force distribution is formulated here as a supervised learning problem. This requires a training dataset composed of input features (here retrieved from the images) and the respective ground truth labels (here obtained from finite element simulations, using the material models derived in Section 3). These labels represent the quantities of interest in the inference process (e.g., the contact force distribution). The following subsections describe in detail each of the components of the dataset.

4.1 Features

In order to perform a large number of indentations within a feasible time, an automatic milling and drilling machine (Fehlmann PICOMAX 56 TOP) is used to press an indenter against the soft surface of the tactile sensor at different locations and depths. The machine is equipped with fast and precise motion control (up to 10^{-3} mm). In the experiments presented here, a stainless steel spherical-ended cylindrical indenter is used. The indenter has a diameter of 10 mm and is attached to the spindle of the milling machine, together



(a) Mechanical behavior of the Ecoflex GEL



(b) Mechanical behavior of the Elastosil 25:1

Figure 4. Stress-stretch response of the Ecoflex GEL (a) and the Elastosil 25:1 (b) in, from left to right, uniaxial tension (UA), pure shear (PS), and equibiaxial tension (EB), together with corresponding hyperelastic model predictions. Note the different scales in (a) and (b), in particular the significantly stiffer equibiaxial response of the Elastosil 25:1.

with a six-axis F/T sensor (ATI Mini 27 Titanium). The experimental data collection setup is shown in Fig. 5.

A total of 13,448 vertical indentations were performed, on an horizontal grid with a regular spacing of 0.55 mm, at various depths (with a maximum depth of 2 mm). The RGB images of the particle spread are captured once the indentation has reached the commanded position. Five images were collected for each indentation in order to improve the robustness to image noise. The F/T sensor's measurements of the total vertical and horizontal (over two perpendicular axes) contact force were recorded.

The optical flow field is extracted from the images (converted to grayscale) through an algorithm based on Dense Inverse Search [32]. The magnitude and the direction of the field are then averaged in m image regions of equal area, as described in [9]. The tuples of magnitude and direction for each of these regions yield a set of $2 \times m$ features for each

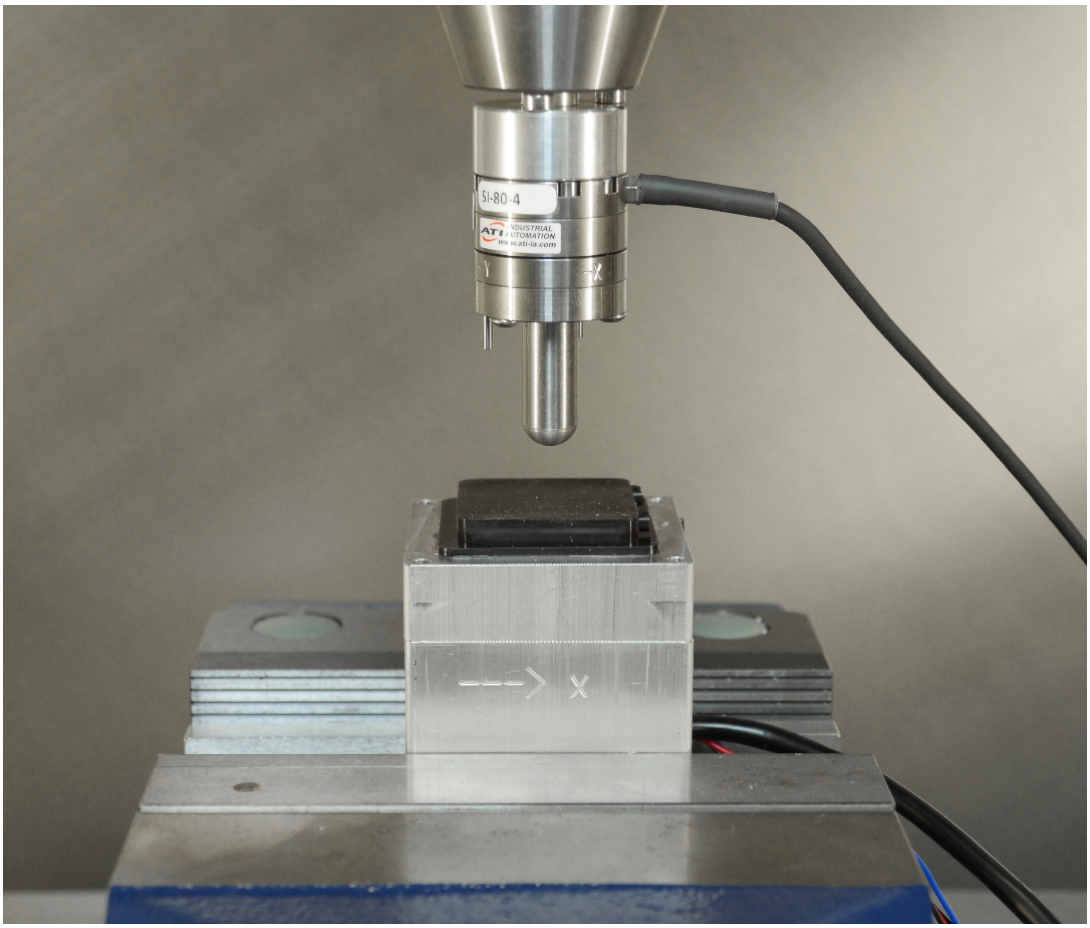
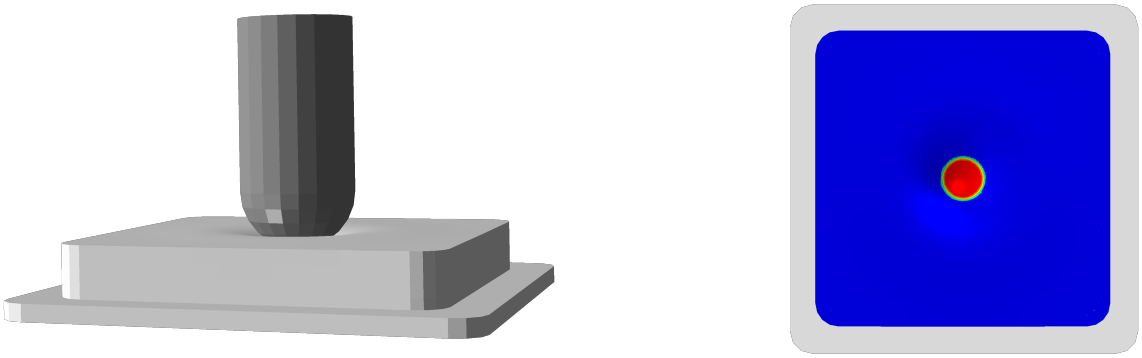


Figure 5. The experimental data collection setup is shown above. The indenter and the F/T sensor (connected through the cable on the top right) are attached to the spindle of an automatic milling machine.



(a) Sample indentation **(b)** Contact pressure distribution

Figure 6. The result of a sample FEM indentation in Abaqus is shown in this figure. The indenter and the gel are modeled to reflect their actual material and geometric properties, see (a). An example of the resulting contact pressure distribution (top view) is shown in (b), where the colors are mapped to the pressure magnitude (from zero, in blue, to the maximum, in red).

data point.

The readings from the F/T sensor are used to assess the quality of the ground truth labels, as described in the next subsection. The range of forces recorded in this procedure spans up to 1.7 N in the vertical direction and 0.15 N in each of the horizontal axes. Note that the large difference in magnitude between the vertical and horizontal forces is mainly due to the symmetry of the indentations, which leads to the cancellation of the symmetric contributions to the total horizontal force, with the exception of the regions close to the edges of the surface.

4.2 Labels

Although the F/T sensor provides the total contact force in each direction, it does not provide any information about the force distribution over the contact surface. The force distribution renders a compact representation of various contact aspects for generic indentations. In fact, it encodes information about the contact area and forces applied to the surface, even in the case of interactions with objects of complex geometries or when multiple and distinct contact points are present. An example application, in which both the contact area and the total contact force are necessary, is presented in [33], where the interaction with a specific object is exploited. The contact force distribution is obtained in this article through FEA, which essentially simulates the indentation experiments performed with the milling machine, see for example Fig. 6.

The FEM simulations are carried out in Abaqus/Standard [34]. The geometry of the two top layers of the sensor is modeled as shown in Fig. 3, and material properties are assigned to each layer as described in Section 3, using the implementation of Ogden's model provided by Abaqus. Note that this neglects the influence of the spherical particles, as the model was derived from tests on the pure Ecoflex GEL. Assuming rigid particles, the ratio between the Ecoflex–particle composite modulus E_c and the Young's modulus E of the pure Ecoflex GEL can be estimated using Eshelby inclusion theory [35], $E_\mathrm{c}/E = 1/(1 - 5\phi/2) \approx 1.05$ for a particle volume fraction $\phi = 0.0196$. The spherical-ended indenter is modeled as an analytical rigid shell. The finite element mesh is composed of linear tetrahedral elements (C3D4H) and hexahedral elements with reduced integration (C3D8RH). Both element types are used with hybrid formulation as appropriate for incompressible materials. A local mesh refinement is performed at the contact and at the interface of the materials, with a characteristic element size of 0.3 mm. Tie constraints are applied at the material interface to enforce the same displacement of the nodes in contact. The bottom nodes are fixed, reflecting the interface with the much stiffer bottom layer (Elastosil 7:1).

The contact between the top surface and the indenter is modeled as a hard contact and discretized with a surface-to-surface method. The friction coefficient between the indenter and the top layer is estimated by letting a block of Elastosil 25:1 rest on an inclined stainless steel plate. The maximum tilt angle θ before the block begins to slide is recorded with an external camera, and the static friction coefficient μ_0 is calculated from static equilibrium as $\mu_0 = \tan\theta$. This procedure yields a friction coefficient of 0.45. The

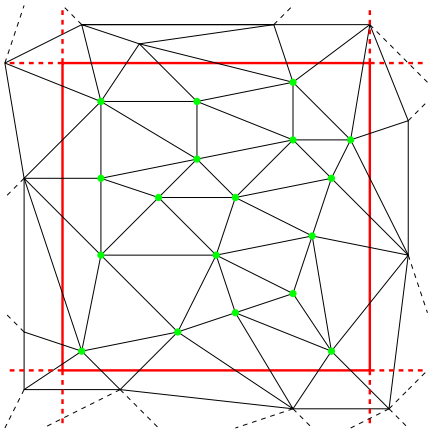


Figure 7. The procedure to discretize the force distribution is sketched in this figure. The sensor's surface is discretized into n bins of equal size, with the boundaries shown in red. The surface mesh used for the FEA is shown in black in the undeformed state. Each node is assigned to the bin that contains it, as is the case for the green nodes contained in the bin shown in solid red. For each indentation, the resulting forces at the nodes assigned to the same bin are summed along each axis to determine the three label vector components for the corresponding bin.

friction coefficient was assumed constant, a possible dependence on the contact pressure was neglected.

The FEM simulations generate the normal and shear components of the contact force distribution resulting from each indentation. Note that both the normal and shear forces acting at each node are generally 3D vectors. As an example, the normal force that stems from a pure vertical indentation is not necessarily vertical, as a consequence of the material deformation (although the vertical component generally has the largest magnitude).

The normal and shear force distributions are discretized by summing the respective nodal forces inside n surface bins, as shown in Fig. 7. The resulting 3D force for each of these bins is used as a ground truth label with $3 \times n$ components for each data point.

This procedure is applied to assign ground truth labels to the 13,448 indentations described in the previous subsection. Since there are no readily available sensors that measure the full contact force distribution with high spatial resolution and without altering the sensor's soft surface, the quality of the labels is evaluated by comparing the components of the total force resulting from the FEM simulations with the ones measured by the F/T sensor. Note that the total force components can be obtained from the FEM simulations by summing the contact force values of all the n bins, or simply summing the force values at all nodes of the surface mesh used for the FEA. The resulting root-mean-square error on the ground truth ($\mathrm{RMSE}_{\mathrm{GT}}$) for the entire dataset is reported in Table 2 for each component. x and y are the horizontal axes, and z is the vertical axis, which is positive pointing from the camera towards the top surface. The resulting errors are comparable to the F/T sensor's resolution, shown in the table as a reference. In Fig. 8, the plots show the agreement on the z component of the total force between the F/T sensor's readings and the results from the FEA for two of the indentation locations. The good agreement between the F/T measurements and the FEM simulations

further justifies the simplifying assumptions taken in the material characterization and the FEM modeling. Additionally, the same plots show that using a linear elastic material model and neglecting geometric nonlinearities (i.e., NLgeom flag in Abaqus) lead to a considerable performance loss for large deformations.

Axis	RMSE_{GT}	F/T resolution
x	0.02 N	0.03 N
y	0.02 N	0.03 N
z	0.06 N	0.06 N

Table 2. Total force agreement (FEA vs F/T sensor)

Although the FEM simulations can be time consuming to carry out, depending on the accuracy required, most of the operations are highly parallelizable, as for example, the several indentations. This makes it possible to exploit cluster computers or GPUs to reduce the time consumption. The simulations presented here are carried out on the Euler cluster of ETH Zurich.

Note that the strategy presented above provides the ground truth for the full contact force distribution under no assumptions on the specific tactile sensing technique. It is therefore not limited to use on vision-based devices, but more generally on data-driven approaches to the force reconstruction task.

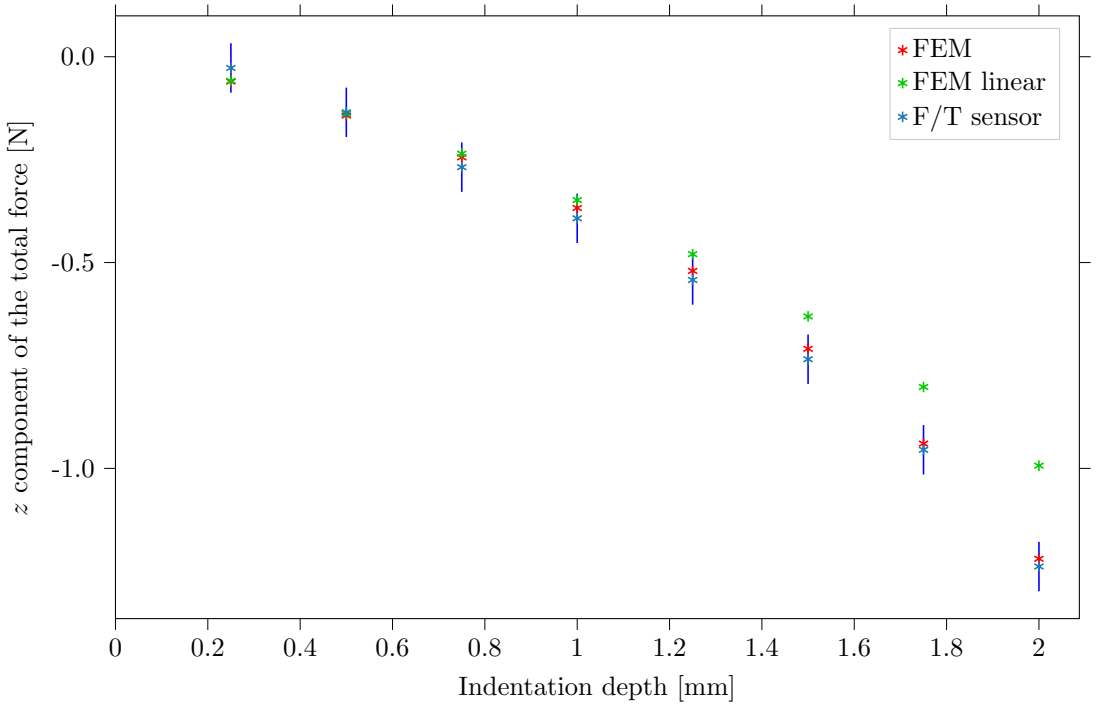
5. Neural network training

5.1 Learning architecture

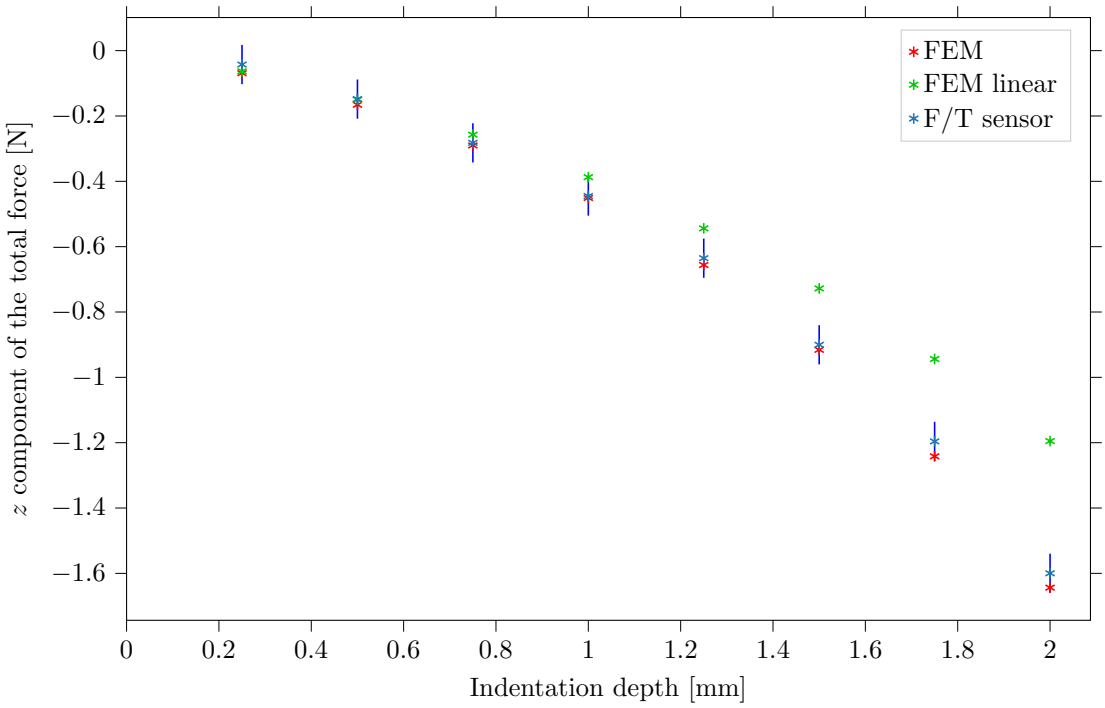
A feedforward DNN architecture (see Fig. 9) is used to address the supervised learning task of reconstructing the full contact force distribution from the features extracted from the images. An input layer with $2 \times m$ units represents the image features described in Section 4.1 (a tuple of averaged optical flow magnitude and direction for each of the chosen image regions). Similarly, an output layer with $3 \times n$ units represents the discretized force distribution applied to the surface of the sensor (a three-dimensional force vector for each of the discrete surface bins).

Three fully connected hidden layers with a sigmoid activation function are used to model the map between the inputs and the outputs. Dropout layers are used after each of the hidden layers during the training phase. Twenty percent of the dataset is used as a test set, while the remaining data are used for training. The architecture is trained with PyTorch[1] by minimizing the mean squared error (MSE) through the Adam optimizer, see [36]. The remaining parameters chosen for the optimization, as well as the size of each layer, are summarized in Table 3. Note that the spatial resolution of the tactile sensor is

[1] `www.pytorch.org`



(a) Center indentation



(b) Corner indentation

Figure 8. The plots above show the agreement on the total vertical contact force between the measurements obtained from the F/T sensor (in blue) and the FEM simulations (in red). The results from the simulations are accurate for indentations at the center of the surface (a) and close to the corners (b) (5 mm from each of the edges) for different indentation depths. The F/T sensor readings are shown with ±0.06 N bars, representing the resolution of the F/T sensor. In green, the results obtained using a linear elastic model (as opposed to the hyperelastic model described in Section 3) and neglecting geometric nonlinearities are shown.

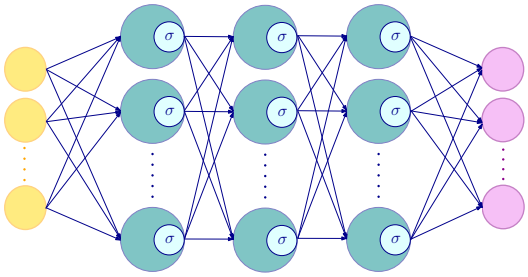


Figure 9. A diagram of the learning architecture used to predict the 3D contact force distribution. In yellow the input layer, representing the image features, in cyan the hidden layers, and in magenta the output layer, representing the discretized force distribution.

determined by the size of the surface bins, which have a side of 1.6 mm, comparable to the spatial resolution of the human fingertip [37]. However, a finer resolution may yield additional discrimination capabilities, i.e., for sensing an object's roughness or texture.

In contrast to architectures that directly learn from the pixels (e.g., convolutional neural networks), the extraction of the optical flow features described in Section 4.1 yields a relatively shallow model, reducing the training times and the data requirements. Additionally, the use of these features makes it possible to efficiently transfer the model across different sensors, as shown in previous work [21]. However, these features do not exploit the information about particles placed at different distances from the camera.

Symbol	Value	Description
m	1600	# of averaging image regions
n	400	# of discrete surface bins
-	(800, 600, 400)	hidden layers' size
-	1E-4	learning rate
-	400	training batch size
-	0.1	dropout rate

Table 3. DNN parameters

5.2 Results

After training, the quality of the DNN predictions is evaluated on the test set. Additionally to the root-mean-square error (RMSE) on the entire test set, the sparse RMSE on the non-zero values of the FEM ground truth is also computed as

$$\text{RMSES} := \sqrt{\frac{1}{|\mathcal{I}|} \sum_{(i,l)\in\mathcal{I}} \left(f_i^{(l)} - \hat{f}_i^{(l)} \right)^2},$$

where $f_i^{(l)}$ and $\hat{f}_i^{(l)}$ are the i-th components of the ground truth and the predicted label,

respectively, for the l-th sample in the test set, and,

$$\mathcal{I} := \left\{ (i,l) \in \{0,\dots,3n-1\} \times \{0,\dots,N_{\text{set}}-1\} \mid f_i^{(l)} \neq 0 \right\},$$

with N_{set} the number of samples in the test set. This metric emphasizes the prediction performance in the location where the contact is expected.

Moreover, the RMSE on the total force is estimated for both the cases, in which the ground truth is provided either by the FEM simulations ($\text{RMSET}_{\text{FEM}}$) or the F/T sensor ($\text{RMSET}_{\text{F/T}}$). The resulting errors from the predictions on the test set are summarized in Table 4 for each axis. The values in the last row are affected by both the errors introduced by the FEM modeling and the DNN predictions. Note that it is only possible to compute the metrics for the force distribution (first two rows) in relation to the FEM simulations (the F/T sensor only provides total forces, without specific information about the force distribution). As a reference, the ranges of force provided by the ground truth labels are summarized in Table 5. Examples of the predicted contact force distribution are shown in Fig. 10 and Fig. 11.

The resulting DNN is deployed on the dual-core laptop computer introduced in Section 1. The entire pipeline yields real-time predictions on an image stream of 40 frames per second, as shown in the experiments available in the video[2] attached to this article. The parallelization of both the optical flow algorithm and the neural network prediction step is not exploited here, but it may be leveraged on commercially available embedded computers provided of GPUs.

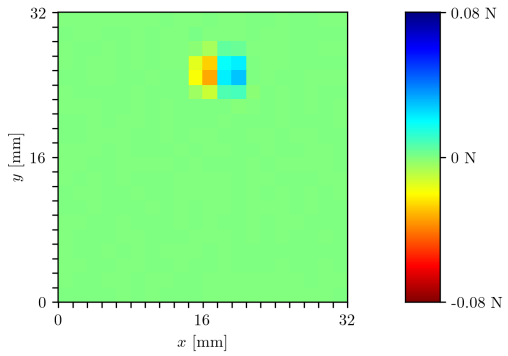
Metric	x	y	z
RMSE	0.001 N	0.001 N	0.003 N
RMSES	0.007 N	0.006 N	0.016 N
$\text{RMSET}_{\text{FEM}}$	0.004 N	0.004 N	0.045 N
$\text{RMSET}_{\text{F/T}}$	0.025 N	0.021 N	0.082 N

Table 4. Resulting errors on force distribution and total force

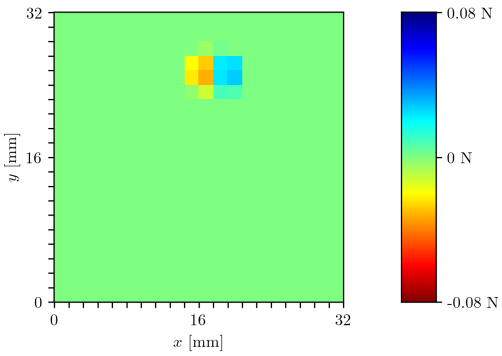
Quantity	x	y	z
Force per bin	[-0.06,0.06] N	[-0.06,0.06] N	[-0.15,0] N
Total force	[-0.13,0.13] N	[-0.13,0.13] N	[-1.66,0] N

Table 5. Range of ground truth forces

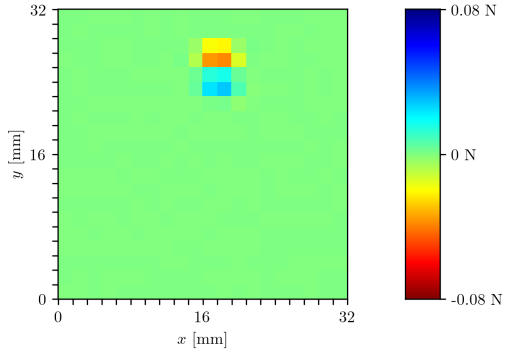
[2]Video: `https://youtu.be/9A-cONrsiOg`



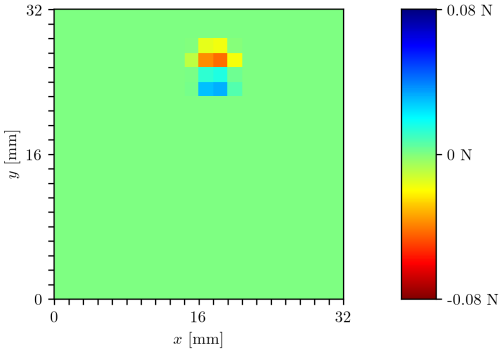
(a) x component of the predicted force distribution



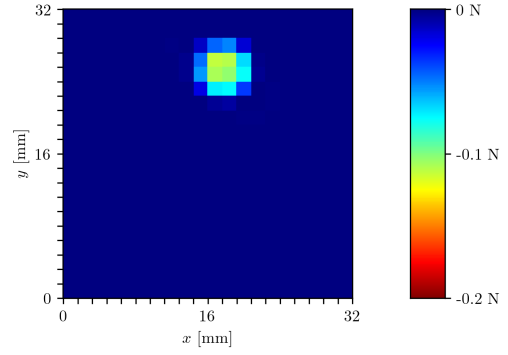
(b) x component of the ground truth force distribution



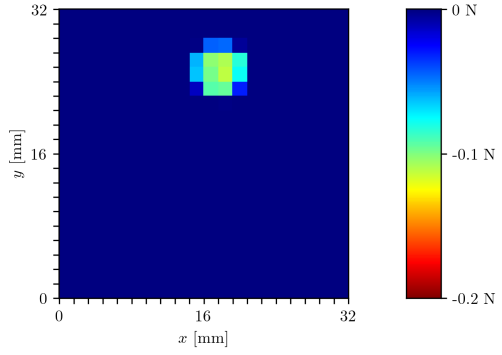
(c) y component of the predicted force distribution



(d) y component of the ground truth force distribution

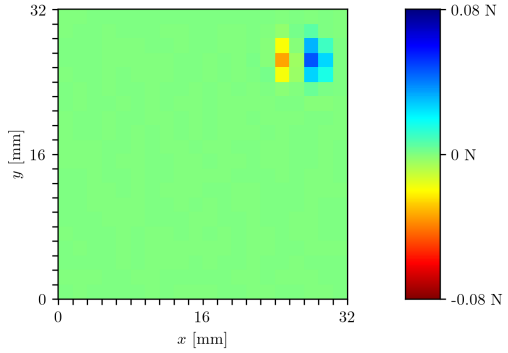


(e) z component of the predicted force distribution

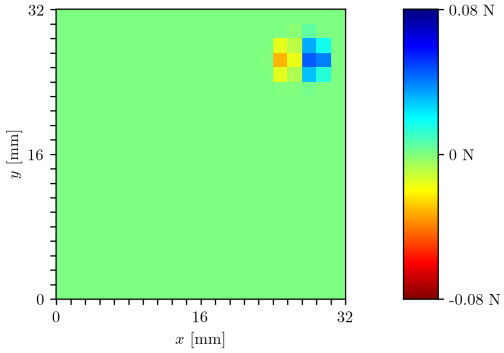


(f) z component of the ground truth force distribution

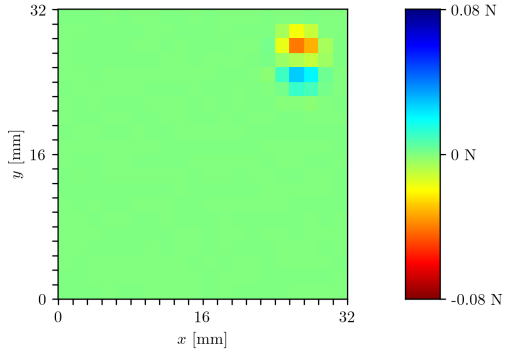
Figure 10. The plots above show the predicted (left) and ground truth (right) 3D contact force distribution applied to the top surface of the tactile sensor for an indentation in the test set. Note that the axes are defined as in Section 4, that is, with two perpendicular horizontal axes x and y, aligned with two of the top surface edges, and a vertical axis z, which is positive pointing from the camera towards the top surface.



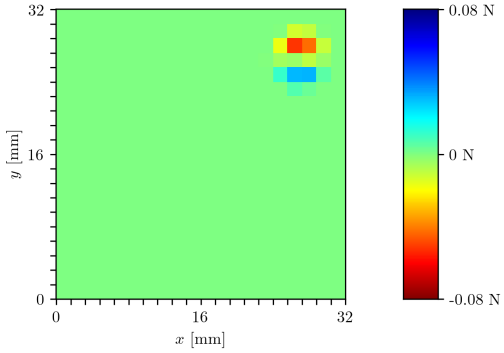
(a) x component of the predicted force distribution



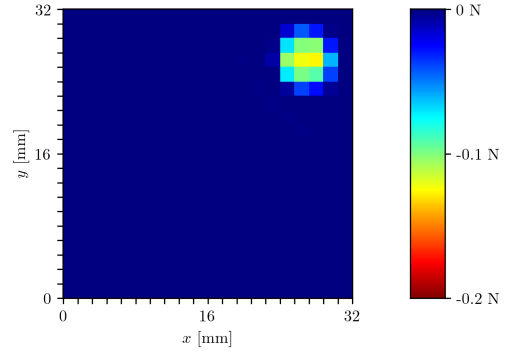
(b) x component of the ground truth force distribution



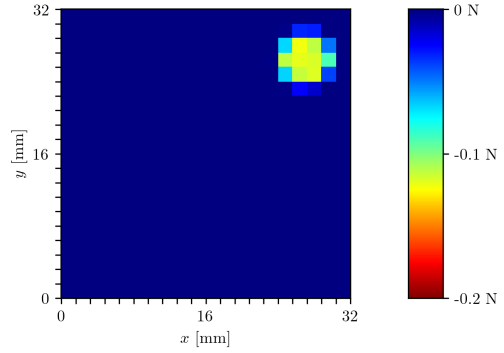
(c) y component of the predicted force distribution



(d) y component of the ground truth force distribution



(e) z component of the predicted force distribution



(f) z component of the ground truth force distribution

Figure 11. Similarly to Figure 10, the plots above show the predicted (left) and ground truth (right) 3D contact force distribution applied to the top surface of the tactile sensor for an indentation in the test set. In this case, the indentation was centered at 5 mm from the top and right edges, with the indenter's radius being 5 mm. It can be noted a larger asymmetry in the force distribution, affected by the stiffer edges of the gel.

6. Conclusion

This article has presented a strategy to provide ground truth contact force distribution for learning-based tactile sensing. The approach has been evaluated on a vision-based tactile sensor, which is based on tracking particles spread within a soft gel. After the characterization of the hyperelastic materials, which provides accurate material models for FEA, a large number of real indentations and corresponding simulations have been performed to generate a dataset that includes image features and ground truth for the 3D contact force distribution. Although the material characterization was performed with considerably different tests and setup (i.e., UA, PS, EB tests) than the indentations considered in the evaluation, the total forces recorded in the experiments are comparable to the ones determined in simulation, showing the generalization potential of the approach proposed. Note that due to the fact that the simulation labels are assigned to real data obtained from experimental indentations, the experimental setup needs to be carefully arranged. As an example, the alignment of the tactile sensor with the reference axes of the milling machine used for the data collection is crucial for obtaining good performance.

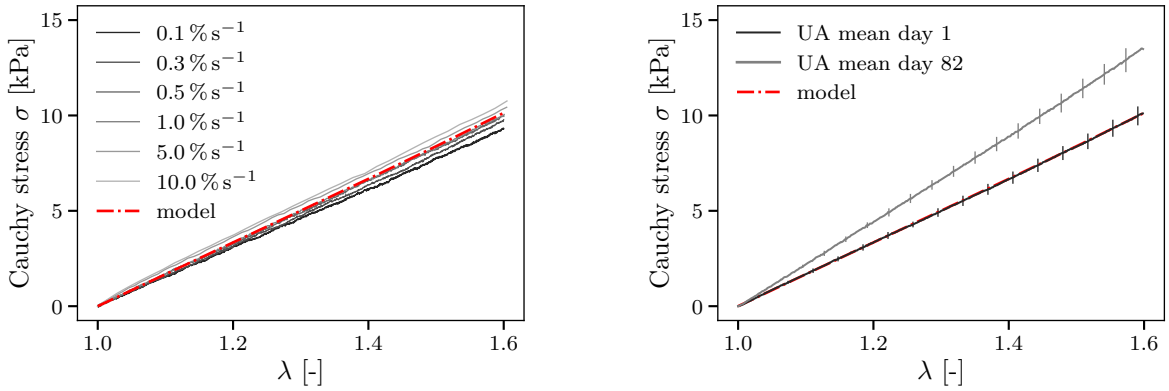
As shown in Section 5, the dataset generated with the strategy proposed in this article can be used to train a DNN for accurate reconstruction of the force distribution applied to the surface of the tactile sensor. Although in these experiments the DNN has been trained and evaluated on a sample indenter, the techniques presented here are directly applicable to generic shapes and indentations, including multiple and distinct contact areas. However, this would likely require the collection of a dataset under various contact conditions, involving interactions with complex objects. Therefore, the generalization capabilities of this strategy will be object of future work.

A. Strain-rate and shelf-time dependent properties of Ecoflex GEL

The additional experimental data on the rate-dependence and the aging effect on the mechanical behavior of the Ecoflex GEL are shown in Fig. 12. While the strain-rate dependence remains relatively low over the three decades analyzed (Fig. 12a), a significant stiffening after 9 weeks of storage at room temperature is evident due to material aging (Fig. 12b). Longer curing times and higher curing temperatures may be used to approach the final, curing-independent material properties [28], [29].

Acknowledgments

The authors would like to thank Michael Egli for the manufacturing support and Francesco Filotto for his insights about the generation of the ground truth labels. The work of



(a) Strain-rate dependence in uniaxial tension for the Ecoflex GEL

(b) Age-dependent mechanical properties of the Ecoflex GEL

Figure 12. Uniaxial tension (UA) tests showing (a) strain-rate and (b) shelf-time (aging) dependence on the mechanical properties of Ecoflex GEL.

A. Wahlsten was supported by the Swiss National Science Foundation under Grant No. 179012.

References

[1] J. Mahler and K. Goldberg, "Learning Deep Policies for Robot Bin Picking by Simulating Robust Grasping Sequences", in *Proceedings of the 1st Annual Conference on Robot Learning*, vol. 78, PMLR, 2017, pp. 515–524.

[2] C. D. Kidd and C. Breazeal, "Robots at Home: Understanding Long-Term Human-Robot Interaction", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2008, pp. 3230–3235.

[3] H. Shen, "Meet the soft, cuddly robots of the future", *Nature*, vol. 530, no. 7588, pp. 24–26, 2016.

[4] H. Abidi and M. Cianchetti, "On Intrinsic Safety of Soft Robots", *Frontiers in Robotics and AI*, vol. 4:5, 2017.

[5] S. L. Gorniak, V. M. Zatsiorsky, and M. L. Latash, "Manipulation of a fragile object", *Experimental Brain Research*, vol. 202, no. 2, pp. 413–430, 2010.

[6] S. Decherchi, P. Gastaldo, R. S. Dahiya, M. Valle, and R. Zunino, "Tactile-Data Classification of Contact Materials Using Computational Intelligence", *IEEE Transactions on Robotics*, vol. 27, no. 3, pp. 635–639, 2011.

[7] N. Wettels and G. E. Loeb, "Haptic Feature Extraction from a Biomimetic Tactile Sensor: Force, Contact Location and Curvature", in *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, IEEE, 2011, pp. 2471–2478.

[8] O. Kroemer, C. H. Lampert, and J. Peters, "Learning Dynamic Tactile Sensing With Robust Vision-Based Training", *IEEE transactions on robotics*, vol. 27, no. 3, pp. 545–557, 2011.

[9] C. Sferrazza and R. D'Andrea, "Design, Motivation and Evaluation of a Full-Resolution Optical Tactile Sensor", *Sensors*, vol. 19, no. 4:928, 2019.

[10] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT press, 2016.

[11] H. Dang and P. K. Allen, "Learning Grasp Stability", in *Proceedings of the IEEE International Conference on Robotics and Automation*, IEEE, 2012, pp. 2392–2397.

[12] R. Calandra, A. Owens, D. Jayaraman, J. Lin, W. Yuan, J. Malik, E. H. Adelson, and S. Levine, "More Than a Feeling: Learning to Grasp and Regrasp Using Vision and Touch", *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3300–3307, 2018.

[13] M. H. Lee and H. R. Nicholls, "Tactile sensing for mechatronics—a state of the art survey", *Mechatronics*, vol. 9, no. 1, pp. 1–31, 1999.

[14] R. S. Dahiya, G. Metta, M. Valle, and G. Sandini, "Tactile Sensing—From Humans to Humanoids", *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 1–20, 2010.

[15] A. Yamaguchi and C. G. Atkeson, "Combining Finger Vision and Optical Tactile Sensing: Reducing and Handling Errors While Cutting Vegetables", in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, IEEE, 2016, pp. 1045–1051.

[16] B. Ward-Cherrier, N. Pestell, L. Cramphorn, B. Winstone, M. E. Giannaccini, J. Rossiter, and N. F. Lepora, "The TacTip Family: Soft Optical Tactile Sensors with 3D-Printed Biomimetic Morphologies", *Soft Robotics*, vol. 5, no. 2, pp. 216–227, 2018.

[17] A. Rigi, F. Baghaei Naeini, D. Makris, and Y. Zweiri, "A Novel Event-Based Incipient Slip Detection Using Dynamic Active-Pixel Vision Sensor (DAVIS)", *Sensors*, vol. 18, no. 2:333, 2018.

[18] A. Alspach, K. Hashimoto, N. Kuppuswamy, and R. Tedrake, "Soft-bubble: A highly compliant dense geometry tactile sensor for robot manipulation", in *Proceedings of the 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, IEEE, 2019, pp. 597–604.

[19] W. Yuan, S. Dong, and E. H. Adelson, "GelSight: High-Resolution Robot Tactile Sensors for Estimating Geometry and Force", *Sensors*, vol. 17, no. 12:2762, 2017.

[20] P. Piacenza, E. Hannigan, C. Baumgart, Y. Xiao, S. Park, K. Behrman, W. Dang, J. Espinal, I. Hussain, I. Kymissis, and M. Ciocarlie, "Touch Sensors with Overlapping Signals: Concept Investigation on Planar Sensors with Resistive or Optical Transduction", *CoRR*, vol. abs/1802.08209, 2019. arXiv: `1802.08209`. [Online]. Available: `http://arxiv.org/abs/1802.08209`.

[21] C. Sferrazza and R. D'Andrea, "Transfer learning for vision-based tactile sensing", *CoRR*, vol. abs/1812.03163, 2019. arXiv: `1812.03163`. [Online]. Available: `http://arxiv.org/abs/1812.03163`.

[22] D. V. Hutton, *Fundamentals of Finite Element Analysis*. McGraw-Hill, 2004.

[23] D. Ma, E. Donlon, S. Dong, and A. Rodriguez, "Dense Tactile Force Distribution Estimation using GelSlim and inverse FEM", in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 5418–5424.

[24] T. L. T. Lun, K. Wang, J. D. L. Ho, K.-H. Lee, K. Y. Sze, and K.-W. Kwok, "Real-Time Surface Shape Sensing for Soft and Flexible Structures Using Fiber Bragg Gratings", *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1454–1461, 2019.

[25] Z. Zhang, J. Dequidt, and C. Duriez, "Vision-Based Sensing of External Forces Acting on Soft Robots Using Finite Element Method", *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1529–1536, 2018.

[26] R. W. Ogden, *Non-linear elastic deformations*. Ellis Horwood Ltd., 1984.

[27] P. Steinmann, M. Hossain, and G. Possart, "Hyperelastic models for rubber-like materials: consistent tangent operators and suitability for Treloar's data", *Archive of Applied Mechanics*, vol. 82, no. 9, pp. 1183–1217, 2012.

[28] R. Hopf, L. Bernardi, J. Menze, M. Zündel, E. Mazza, and A. E. Ehret, "Experimental and theoretical analyses of the age-dependent large-strain behavior of Sylgard 184 (10:1) silicone elastomer", *Journal of the Mechanical Behavior of Biomedical Materials*, vol. 60, pp. 425–437, 2016.

[29] V. Placet and P. Delobelle, "Mechanical properties of bulk polydimethylsiloxane for microfluidics over a large range of frequencies and aging times", *Journal of Micromechanics and Microengineering*, vol. 25, no. 3, 2015.

[30] L. Bernardi, R. Hopf, A. Ferrari, A. E. Ehret, and E. Mazza, "On the large strain deformation behavior of silicone-based elastomers for biomedical applications", *Polym. Test.*, vol. 58, pp. 189–198, 2017.

[31] R. W. Ogden, "Large deformation isotropic elasticity — on the correlation of theory and experiment for incompressible rubberlike solids", in *Proceedings of the Royal Society of London. A. Mathematical, Physical and Engineering Sciences*, vol. 326, 1972, pp. 565–584.

[32] T. Kroeger, R. Timofte, D. Dai, and L. Van Gool, "Fast Optical Flow using Dense Inverse Search", in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, 2016, pp. 471–488.

[33] K. Nozu and K. Shimonomura, "Robotic bolt insertion and tightening based on in-hand object localization and force sensing", in *Proceddings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, IEEE, 2018, pp. 310–315.

[34] Dassault Systèmes, *Abaqus/Standard User's Manual, Version 6.14*, English, Simulia, 2014.

[35] J. D. Eshelby, "The determination of the elastic field of an ellipsoidal inclusion, and related problems", *Proc. R. Soc. London. Ser. A. Math. Phys. Sci.*, vol. 241, no. 1226, pp. 376–396, 1957.

[36] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization", in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[37] J. Dargahi and S. Najarian, "Human tactile perception as a standard for artificial tactile sensing—a review", *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 1, no. 1, pp. 23–35, 2004.

Paper P4

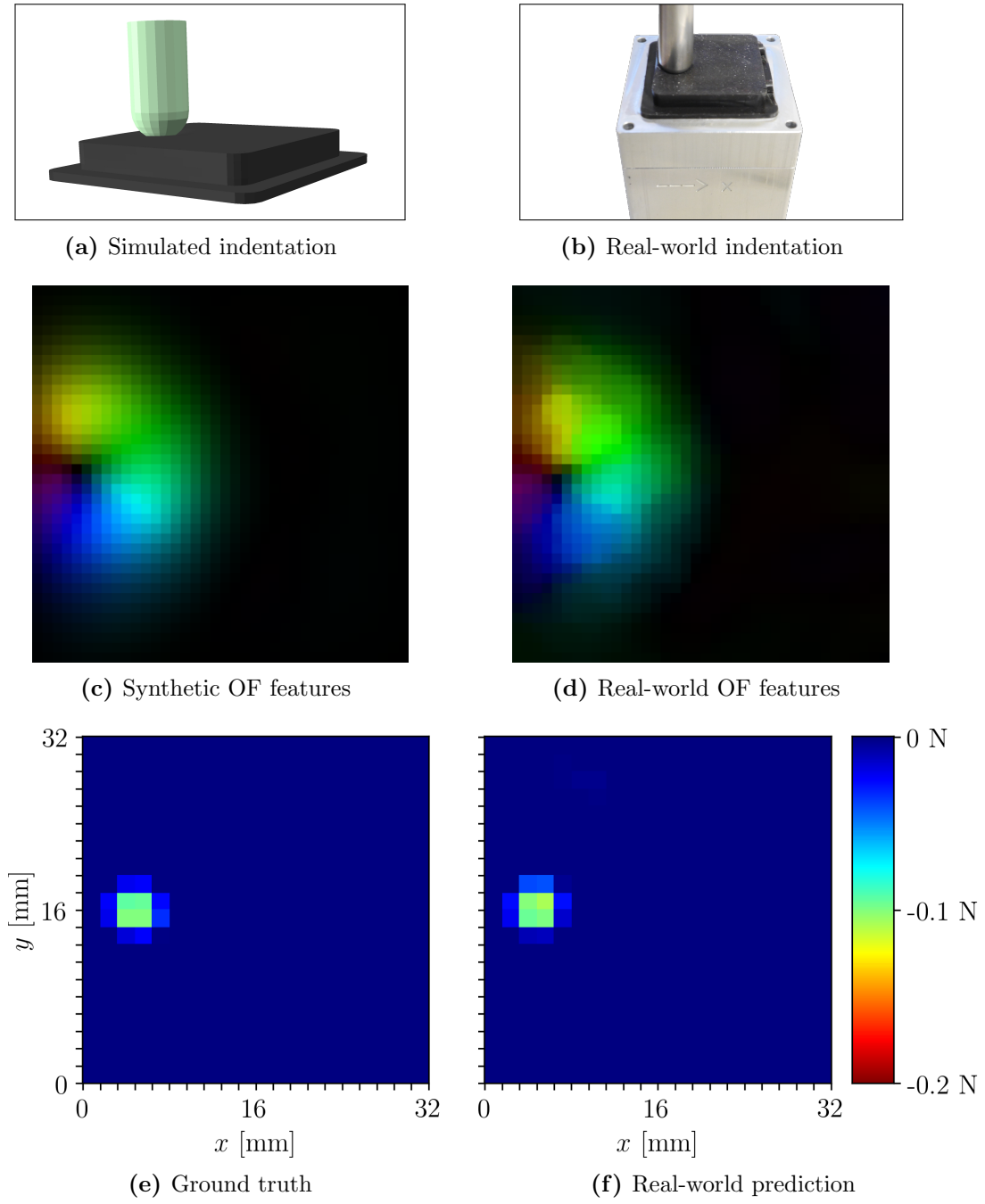
Learning the sense of touch in simulation: a sim-to-real strategy for vision-based tactile sensing

Carmelo Sferrazza, Thomas Bi and Raffaello D'Andrea

Abstract

Data-driven approaches to tactile sensing aim to overcome the complexity of accurately modeling contact with soft materials. However, their widespread adoption is impaired by concerns about data efficiency and the capability to generalize when applied to various tasks. This paper focuses on both these aspects with regard to a vision-based tactile sensor, which aims to reconstruct the distribution of the three-dimensional contact forces applied on its soft surface. Accurate models for the soft materials and the camera projection, derived via state-of-the-art techniques in the respective domains, are employed to generate a dataset in simulation. A strategy is proposed to train a tailored deep neural network entirely from the simulation data. The resulting learning architecture is directly transferable across multiple tactile sensors without further training and yields accurate predictions on real data, while showing promising generalization capabilities to unseen contact conditions.

Published in *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*.



(a) Simulated indentation **(b)** Real-world indentation

(c) Synthetic OF features **(d)** Real-world OF features

(e) Ground truth **(f)** Real-world prediction

Figure 1. In this work, a fully synthetic dataset (see (a), (c) and (e)) is generated to train an artificial neural network, which aims to reconstruct the three-dimensional contact force distribution applied to the soft surface of a vision-based tactile sensor from optical flow (OF) features. The networks exhibits accurate predictions on real-world data (see (b), (d) and (f)). Note that in (e) and (f) only the vertical component of the contact force distribution is shown.

1. Introduction

Understanding physical contact with the environment is a crucial requirement for the safe and reliable operation of robots interacting with their surroundings. As an example,

a robot that aims to grasp objects in a cluttered box benefits from sensory feedback about the contact with these objects, in order to infer the quality of the grasp and correct its behavior [1], [2]. Research on tactile sensing focuses on providing such feedback, generally by processing the information about the deformation of a soft sensing surface when interacting with external bodies.

Among the various quantities of interest, the distribution of the contact forces applied to the sensing surface offers high versatility in terms of tasks and conditions. In fact, the contact force distribution retains information about the total force applied and it densely encodes the surface patches in contact with external objects. Additionally, its distributed nature provides a representation that generalizes to various contact conditions, i.e., interaction with a generic number of objects, sensing surfaces of arbitrary shape and size. However, the complexity of accurately modeling soft materials has hindered the development of sensors that can accurately reconstruct the contact force distribution in real-time, especially in the case of large deformations of the soft sensing surface. As a matter of fact, when soft materials, e.g. rubber, deform beyond a limited linear elastic region, the stress-strain relation becomes nonlinear [3]. As a result, accurately mapping the information about the material deformation to the contact forces generating it becomes challenging and often computationally infeasible in real-time for general cases.

In order to overcome this limitation, machine learning approaches have been proposed to obtain a data-driven model, which approximates the mapping of interest and yields appropriate inference times. A drawback in these approaches is the large amount of data typically required for each sensor produced and their limited generalization capabilities when applied to data not seen during the training time. This paper proposes a strategy to train a deep neural network with data obtained via highly accurate, state-of-the-art simulations of a vision-based tactile sensor, exploiting hyperelastic material models and an ideal camera projection. The network is directly deployed to a real-world application, where the images are transformed to the reference simulation camera model. In this way, the network does not need to be trained for each sensor, but only the appropriate camera calibration parameters need to be extracted. The real-world evaluation shows an accurate transfer from simulation to reality (sim-to-real, or sim2real), and a refinement strategy is proposed to further improve the real-world performance with a single indentation. In the case considered in this paper, the training dataset only consists of simulations of single spherical indentations. However, the model deployed in reality shows promising generalization capabilities to multiple contact conditions and to indenters of different shapes. The sensing pipeline presented here runs on the CPU of a standard laptop computer (dual-core, 2.80 GHz) at 50 Hz.

1.1 Related work

The estimation of contact forces using tactile sensors has been largely investigated in the context of contact with single objects, where the force magnitude and location, or the object shape is of interest. Several approaches have been developed for resistive [4], barometric measurement-based [5], capacitive [6], and optical [7] tactile sensors, either using

model-based or data-driven methods. Compared to these categories, optical (or vision-based) tactile sensors exhibit very high resolution, low cost and ease of manufacture, at the expense of non-trivial data processing.

In the context of vision-based tactile sensors, the reconstruction of the contact force distribution was first addressed in [8]. The soft material was modeled as an infinite, linear elastic half-space and a closed-form solution was proposed to map the deformation of the material to the contact force distribution. In [9], a technique based on the finite element method (FEM) was proposed to estimate the force distribution in real-time, given the assumption of linear elasticity. The same assumption was used in [10] to formulate an optimization-based method that estimated the surface patches in contact with external objects. While data-driven techniques are potentially suitable to overcome the complexity of modeling soft materials without introducing assumptions only valid for small deformations, their application to the estimation of distributed quantities has mainly been prevented by the lack of a ground truth source, which is crucial for supervised learning approaches. However, in [11] a strategy based on offline finite element simulations was recently proposed to address this problem and provide ground truth labels for generic data-driven tactile sensors. A hyperelastic model was shown to outperform a linear elastic formulation also for rather small deformations.

The major drawback of learning-based approaches to tactile sensing, either estimating total forces or distributed quantities, lies in their high data requirements. To partially address this issue, a transfer learning approach was proposed in [12] to reduce the amount of data needed to transfer a learning architecture across the different sensors produced. Recently, simulation approaches have been investigated for different sensing principles in order to perform most of the training phase with synthetic data. In [13], a simulation model was presented to generate raw data for a sensor based on barometric measurements, with the location and magnitude of the force applied at a contact point as inputs. Tactile images for an optical sensor based on multi-color LEDs were generated in simulation in [14] for various deformations of the sensing surface. Binary tactile contacts were simulated in the context of a grasping scenario in [15], while a sim-to-real approach was investigated in [16] for the estimation of conductivity on the surface of a tactile sensor based on electrical impedance tomography.

This work aims to provide a simulation strategy to generate an entire supervised learning dataset for a vision-based tactile sensor, with the objective of estimating the full contact force distribution from real-world tactile images. The sensor was first presented in [17] and is based on a camera that tracks a random spread of particles within a soft, transparent material to infer information about the forces applied to the sensing surface. In the proposed design, all pixels of the camera provide informative data, which can be leveraged via a machine learning architecture aiming to reconstruct the three-dimensional contact force distribution with high accuracy, as shown in [11]. As opposed to [11], where simulated ground truth labels for the force distribution were matched to real-world images, this work proposes to also generate the tactile images in simulation and to use the entire synthetic dataset in a supervised learning fashion. To this purpose,

deformation data were generated via FEM simulations based on hyperelastic material models and fed through an ideal pinhole camera model. For real-world deployment, the tactile images obtained on a real tactile sensor were transformed to the pinhole reference model, by employing state-of-the-art calibration methods. In this way, not only can the model trained in simulation be deployed to real-world sensors, but it can also be easily transfered across multiple instances of the sensors produced, provided that the calibration model has been extracted.

1.2 Outline

The sensing principle and the hardware used to evaluate the approach presented here are discussed in Section 2. The generation of synthetic tactile images and ground truth labels are described in Section 3. In Section 4, the transformation applied to real-world data is presented, as well as the procedure to obtain an accurate camera projection model. The results are discussed in Section 5, while Section 6 draws the conclusions and gives an outlook on future work. In the remainder of this paper, vectors are expressed as tuples for ease of notation, with dimension and stacking clear from the context.

2. Sensing principle

The sensor employed in this paper follows the principle introduced in [17]. Three soft silicone layers are poured on top of an RGB fisheye camera (ELP USBFHD06H), surrounded by LEDs. From the bottom: a stiff transparent layer (ELASTOSIL® RT 601 RTV-2, mixing ratio 7:1, shore hardness 45A), which serves as a spacer and for light diffusion; a very soft transparent layer (Ecoflex™ GEL, ratio 1:1, shore hardness 000-35), which embeds a spread of randomly distributed polyethylene particles (microspheres with a diameter of 150 to 180 μm); a black layer (ELASTOSIL® RT 601 RTV-2, ratio 25:1, shore hardness 10A), which is more resistant to repeated contact than the middle layer and shields the sensor from external light. An exploded view of the sensor layers is depicted in Fig. 2. The volume of the gel containing the particles is 30×30×4.5 mm, while the joint volume of the particle layer and the black layer is 32×32×6 mm.

When the soft sensing surface is subject to force, the material deforms and displaces the particles tracked by the camera. This motion generates different patterns in the images, which can be processed to extract information about the contact force distribution causing the deformation.

3. Learning in simulation

The contact force distribution is modeled here in a discretized fashion, by dividing the square sensing surface in $n \times n$ bins of equal area. Three matrices F_x^G, F_y^G, F_z^G of size

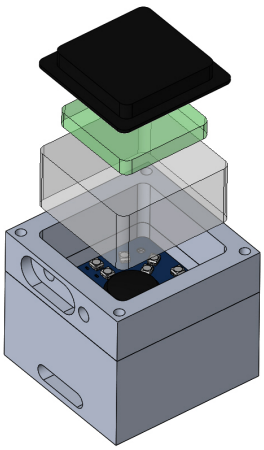


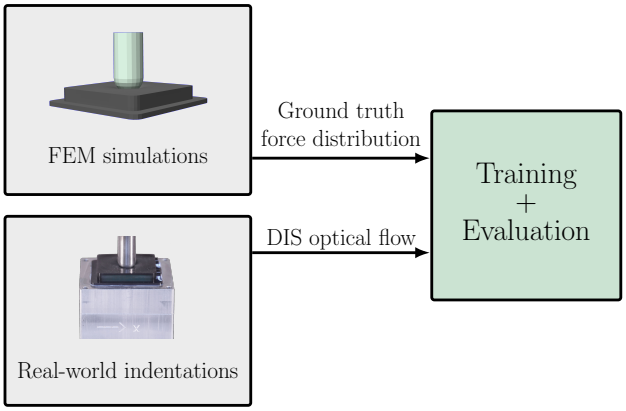
Figure 2. An exploded view of the tactile sensor employed in this paper is shown in the figure above.

$n \times n$ represent the force distribution in the gel coordinate system, where the origin is placed at one of the surface corners, x^G and y^G are aligned with two perpendicular surface edges and z^G is the vertical axis, pointing from the camera towards the surface, as shown in Fig. 4. Each matrix element represents the respective force component applied at the respective bin.

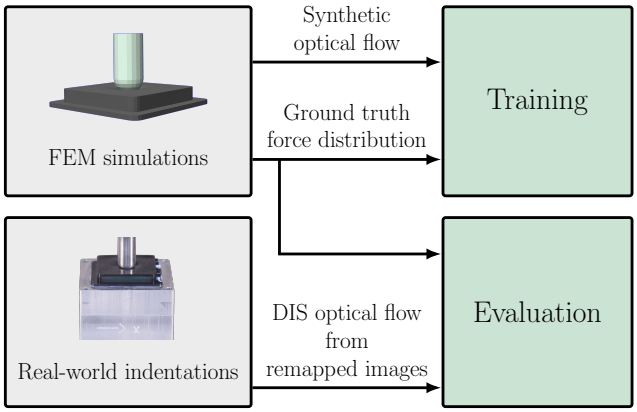
The reconstruction of the force distribution can be posed as a supervised learning problem, even if there are no readily available commercial sensors that can measure the ground truth, i.e., the three-dimensional contact force distribution applied to soft materials, without altering the sensing surface. In fact, in [11] it was shown how highly accurate ground truth force distributions can be obtained by means of FEM simulations, where the corresponding nodal forces are summed within each bin to obtain the force distribution matrices. In [11], hyperelastic models of the soft materials employed were obtained via state-of-the-art characterization methods and used to simulate thousands of indentation experiments. In order to create a supervised learning dataset (see Fig. 3), the resulting ground truth labels were then matched to optical flow features extracted via an algorithm based on dense inverse search (DIS [18]). These features were obtained by replicating in reality the same indentation experiments. Conversely, in this paper the optical flow features, which are the input to the learning algorithm, are obtained as well from the simulated indentations and together with the ground truth labels contribute to generating a fully synthetic training dataset, greatly reducing the data collection efforts. The sim-to-real transfer is evaluated on real-world indentations performed at sampled surface locations, as discussed in Section 4.

3.1 Generating optical flow features

The DIS optical flow algorithm estimates the displacement of the particles at each pixel between a reference frame (with the gel at rest) and the current frame. In order to



(a) Dataset generation as in [11]



(b) Dataset generation proposed here

Figure 3. As shown in (a), the strategy proposed in [11] was based on the collection of real-world images, from which optical flow features were extracted. Ground truth labels were obtained in simulation and assigned to these features to train a supervised learning architecture. The resulting architecture was then evaluated on a portion of this dataset, not used during training. Here the training dataset is fully generated in simulation by extracting synthetic optical flow features from FEM indentations. In this step, a reference pinhole camera model is employed. The sim-to-real transfer is evaluated on a dataset composed of real optical flow features, which are computed after the real-world (distorted) images are remapped to the reference pinhole model.

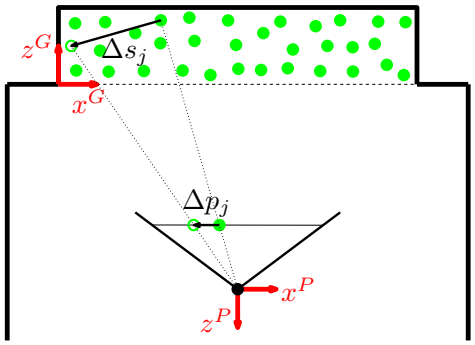


Figure 4. The figure depicts the coordinate systems employed, here shown in the two-dimensional case. The displacement vectors are retrieved from the FEM simulations in the gel coordinate system (superscript G) and transformed to the pinhole coordinate system (superscript P). Finally, the projected pixel displacements are computed as described in (5)-(6).

replicate this functionality on the simulated indentations, the undeformed gel volume containing the particles is first sampled on a fine uniform grid of points. In the remainder of this paper, the positions of these points with the gel at rest are denoted as *undeformed locations*, while their positions after deformation are denoted as *deformed locations*. For each indentation, the respective displacement vectors are assigned to the undeformed locations, via an inverse distance weighted interpolation [19] of the displacement field obtained from the FEM simulations. This step ensures that 3D displacement vectors are estimated at uniformly spaced locations, since the FEM mesh is generally refined at sections of interest. Each of these displacement vectors emulates the motion that a particle located at the respective undeformed location at rest would experience during an indentation. Within a specific indentation, this means that a particle with an initial location in the gel coordinate system at

$$s_j^G := (x_j^G, y_j^G, z_j^G) \tag{1}$$

is expected to move to a deformed location $s_j^G + \Delta s_j^G$, where

$$\Delta s_j^G := (\Delta x_j^S, \Delta y_j^S, \Delta z_j^S) \tag{2}$$

represents the corresponding displacement vector, for $j = 0, \ldots, N_\mathrm{s} - 1$, with N_s the number of sampled locations. In order to simulate an optical flow estimation, the displacement vectors are projected to the image plane. This involves a coordinate transformation, from the gel coordinate system (aligned with the simulation coordinate system) to the image, employing an ideal pinhole camera model [20, p. 49], as depicted in Fig. 4. To this purpose, a displacement vector and its corresponding undeformed location are first transformed to the 3D pinhole camera coordinate system as

$$\Delta s_j^P = R^{GP} \Delta s_j^G, \tag{3}$$

$$s_j^P = R^{GP} s_j^G + t^{GP}, \tag{4}$$

where R^{GP} and $t^{GP} := (t_x^{GP}, t_y^{GP}, t_z^{GP})$ are the corresponding rotation matrix and a translation vector, respectively, comprising the reference camera's extrinsic parameters. The choice of these parameters is further discussed in Section 4. The resulting pixel displacement and the projection of the corresponding undeformed location on the image, i.e., $\Delta p_j := (\Delta u_j, \Delta v_j)$ and $p_j := (u_j, v_j)$, respectively, are then computed as

$$\Delta p_j = K_{\mathrm{a},j}(s_j^P + \Delta s_j^P) - K_{\mathrm{b},j} s_j^P, \tag{5}$$

$$p_j = K_{\mathrm{b},j} s_j^P, \tag{6}$$

with

$$K_{\mathrm{a},j} = \frac{1}{z_j^P + \Delta z_j^P} \begin{bmatrix} f & 0 & u_\mathrm{c} \\ 0 & f & v_\mathrm{c} \end{bmatrix}, \tag{7}$$

$$K_{\mathrm{b},j} = \frac{1}{z_j^P} \begin{bmatrix} f & 0 & u_\mathrm{c} \\ 0 & f & v_\mathrm{c} \end{bmatrix}, \tag{8}$$

where $|f|$ is the focal length and the camera center coordinates $u_\mathrm{c}, v_\mathrm{c}$ are set at the image center. Since the particle layer has a square horizontal section of 30×30 mm, the image region of interest is set as a square of (arbitrary) dimension 440×440 pixels and the focal length is equally chosen for both the image coordinates as

$$f := \frac{440}{30} t_z^{GP}, \tag{9}$$

to exactly fill the image with the particle layer. Note that f is negative, due to the definition of the pinhole camera coordinate system.

In order to create a compact set of features, the image is divided into $m \times m$ regions of equal area. The pixel displacements are assigned to the image regions, based on the coordinates of the corresponding p_j. Then, the average of these displacements within each image region (i, l), for $i = 0, \ldots, m-1$ and $l = 0, \ldots, m-1$, is computed for both components as

$$\overline{\Delta u_{il}} = \frac{1}{\|w_{il}\|} \sum_{j \in \mathcal{J}_{il}} w_j \Delta u_j, \tag{10}$$

$$\overline{\Delta v_{il}} = \frac{1}{\|w_{il}\|} \sum_{j \in \mathcal{J}_{il}} w_j \Delta v_j, \tag{11}$$

where $\mathcal{J}_{il} \subseteq \{0, \ldots, N_\mathrm{s} - 1\}$ is the set of the displacement indices assigned to the region (i, l), w_j are averaging weights and $\|w_{il}\| := \sum_{j \in \mathcal{J}_{il}} w_j$.

The weights are introduced to account for occlusions occurring in real-world images. In fact, since the synthetic optical flow emulates the displacement of the particles in reality, one must consider the fact that on real images some of the particles might be occluded by the particles closer to the camera. For this reason, each projected displacement is weighted in (10)-(11) with the probability of both its deformed and undeformed locations being visible in the image, that is,

$$w_j := \rho_j \sigma_j, \tag{12}$$

where ρ_j is the probability that a particle located (in the gel coordinate system) at s_j^G is visible in the image frame taken with the gel at rest, and σ_j is the probability that a

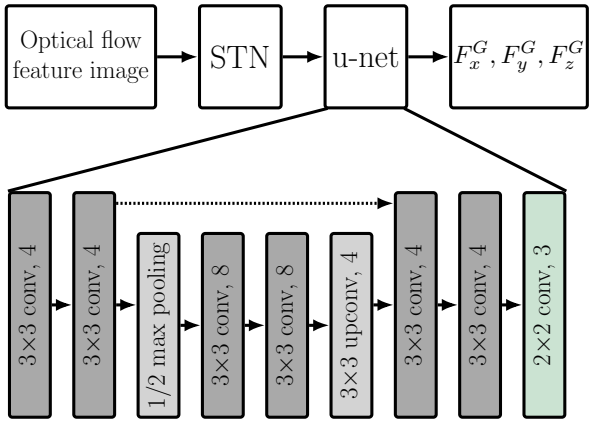
particle located at $s_j^G + \Delta s_j^G$ is visible in the image frame taken after deformation, that is, at the time the optical flow is being computed. Note that here the two visibility events in the respective frames have been considered to be independent, which might not be a valid assumption for some special cases (e.g., for particles placed on the camera optical axis during centered vertical indentations). However, this simplifies the derivation and has proved to be a reasonable assumption in practice. The two probability values for each weight can be computed via Monte Carlo simulations, assuming that the density of the particles does not considerably change during an indentation. This is done by randomly drawing 100 different particle configurations and projecting them to the image plane using (4) and (6). Assuming that a spherical particle is projected to a circle in the image, the radius r of the projected circle is computed as shown in the Appendix, and a particle is considered as occluded if its center in the image is covered by any other circle generated by a particle closer to the camera. Note that in the calculation of the weights, the particle configurations are sampled using the known particle-to-silicone ratio. The resulting particles are generally less than N_s, which is chosen to be large enough to enhance robustness to numerical noise in the FEM results.

The probability of a visible particle is approximated in discrete 3D bins, to which particles are assigned depending on their position within the gel, dividing the number of visible particles in the bin by the total number of particles assigned to the respective bin. This is done for each particle layer configuration and the resulting probabilities are averaged over the 100 configurations. Both the values ρ_j and σ_j are retrieved from this probability discretization, depending on the locations s_j^G and $s_j^G + \Delta s_j^G$ for the j-th displacement vector. Note that this is a valid procedure for computing σ_j only if the density of the particles is constant over an indentation. This assumption is justified by the large number of particles spread within the gel at varying depths. However, this approximation becomes more severe for large deformations, when the particles tend not to spread homogeneously.

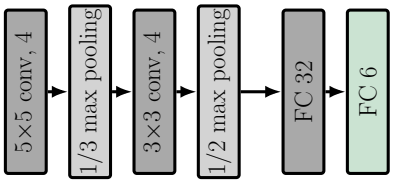
3.2 Learning architecture

The learning task is addressed here as an image-to-image translation, also known as pixel-wise regression. In fact, the quantities obtained from (10)-(11) are rearranged to form an image-like tensor with two channels each with $m \times m$ elements. Similarly, the three matrices F_x^G, F_y^G, F_z^G representing the force distribution are grouped in three channels each with $n \times n$ elements. In the following, the case subject of evaluation, i.e., $m = 40$, $n = 20$, will be considered.

In this paper, the neural network architecture is largely inspired by u-net [21], a well-known architecture widely employed for image segmentation tasks. The original version in [21] exhibited a fully convolutional structure, with a contracting path to extract context from the image patches and a symmetric expanding path (via upsampling) to assign a label to each pixel. Additionally, high resolution information was fed to the upsampled layers to perform pixel-wise regression. Here, the blocks inspired by u-net are placed after a spatial transformer network (STN [22]), which learns an affine transformation of the



(a) Learning architecture and u-net blocks



(b) The STN localization network

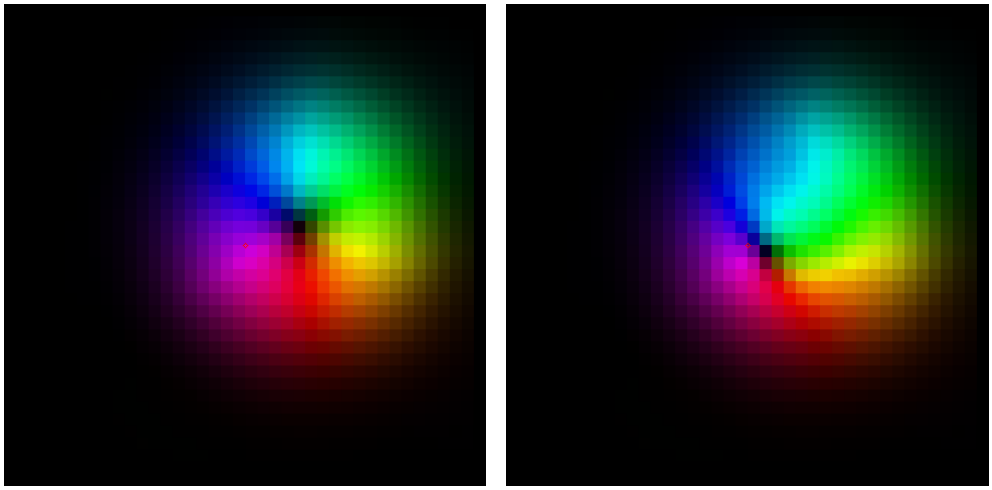
Figure 5. The learning architecture is built upon an STN part and a slimmer version of u-net. For ease of visualization, some abbreviations have been introduced above. For instance, the label "3×3 conv, 4" indicates a convolutional layer with four output channels and a 3×3 kernel, while "1/2 max pooling" refers to a maximum pooling layer, which subsamples the input to half of its original size. Finally, "3×3 upconv, 4" represents an upconvolutional layer, which doubles the input size, and "FC 32" denotes a fully connected layer with 32 units. In (a), the dashed arrow indicates the concatenation of the high resolution content with the upsampled information. For all 3×3 convolutional layers, unitary zero-padding and a stride of 1 were used to retain the input size, as opposed to the last layer, where no padding and a stride of 2 halve the input to obtain 20×20 force distribution matrices. In (b), the *localization network* of the STN is shown, which learns a 6D affine transformation. No padding and a stride of 1 were used for the convolutional layers. In both (a) and (b), batch normalization and rectified linear unit activations were used at all convolutional and fully connected layers, with the exception of the respective output layers (in green).

input features conditioned to the input itself, with the purpose of aligning the optical flow with the contact force distribution. The architecture is depicted in Fig. 5. Note that the STN block only transforms the input image using the learned affine transformation, retaining the initial input size.

In order to close the sim-to-real gap, the synthetic optical flow features are perturbed during training via elastic deformation noise (see Fig. 6), which has been proven to be especially suitable for pixel-wise regression tasks [23].

4. Real data adjustment

In real-world applications, the pinhole camera model does not capture the full camera projection, which exhibits lens distortion and various non-idealities. Camera calibration



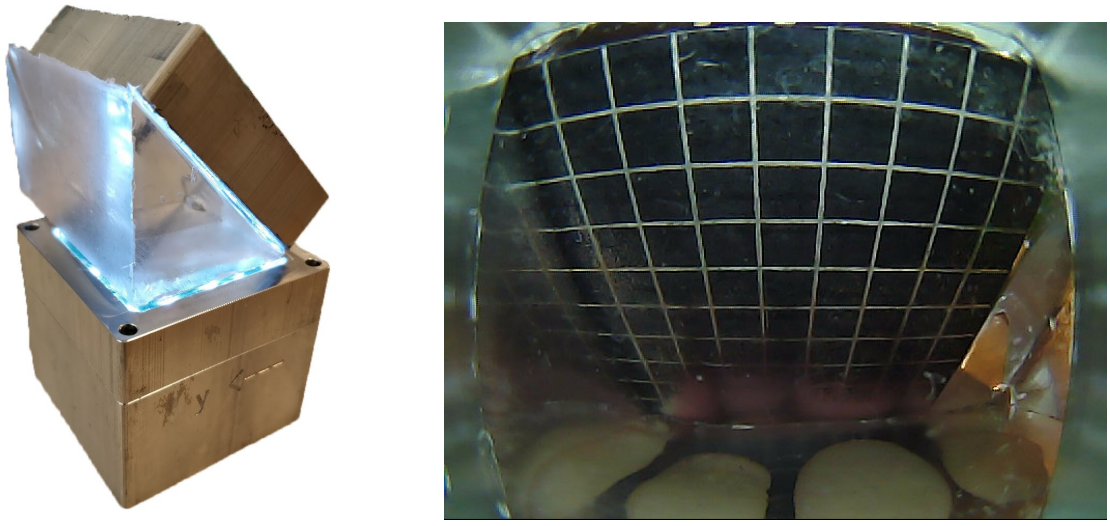
(a) Original OF features **(b)** Deformed OF features

Figure 6. In this figure, an example of the synthetic optical flow (OF) features before (a) and after (b) elastic deformation is shown. The color represents direction, while darker regions represent smaller displacements.

techniques aim to find the actual camera model from real images taken with the camera of interest. In this section, a procedure is proposed that enables the deployment of the neural network trained in simulation as described in Section 3 (assuming a pinhole projection) to a real-world sensor with a generic camera model. First, the employed camera calibration technique is presented. Then, an algorithm to remap the real world images to the pinhole reference model introduced in Section 3 is described.

4.1 Camera calibration

Given the large field of view of the fisheye lens employed, a calibration technique that accounts for the lens distortion is required to accurately match the camera projection. The strategy presented in [24] was employed in this paper, since it is tailored to omni-directional and fisheye cameras and enables straightforward calibration via a MATLAB toolbox. However, given the fact that in the application discussed here the camera is surrounded by silicone, the different refraction index with respect to air causes the light rays to deviate. Although the calibration method presented in [24] does not account for these refraction effects, shooting the calibration images directly through the same silicone medium (the stiffer rubber described in Section 2) yielded accurate results. One of the calibration images is shown in Fig. 7. As a result of the calibration, the toolbox provides a function denoted as `world2cam`, which accounts for the intrinsic parameters and projects a 3D point in the coordinate system of the real-world camera to the corresponding pixel in the image. By feeding an image in which the origin of the calibration pattern is aligned with the FEM coordinate system, the toolbox also outputs the extrinsic parameters of interest, R^{GC} and t^{GC}. These parameters represent a transformation from the FEM (or gel) coordinate system to the coordinate system of the real-world camera (see Fig. 8).



(a) Calibration procedure **(b)** Calibration image

Figure 7. Six calibration images were used to obtain the real-world camera model. The images were taken before casting the two upper silicone layers, by placing a grid pattern at different positions, with only a transparent silicone medium of different shapes between the camera and the pattern. The calibration using the toolbox described in [24] obtained a subpixel reprojection error.

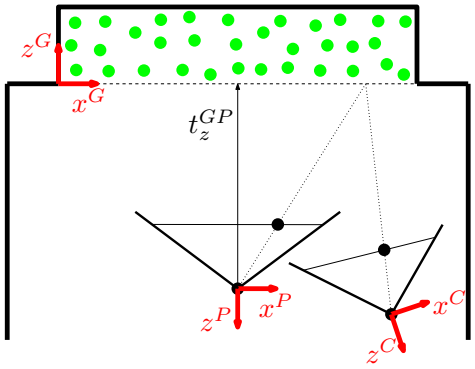
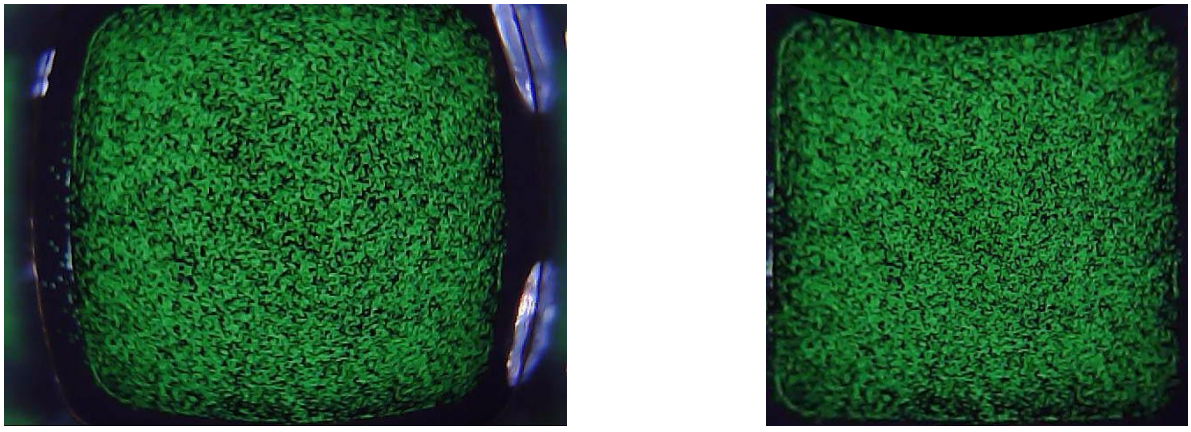


Figure 8. The remapping procedure is shown in this figure. A new pinhole camera image is generated by filling each pixel with the corresponding real image pixel, found by reflecting the appropriate image ray on the bottom of the particle layer. Note that this approximation is exact when the origins of the pinhole (superscript P) and real (superscript C) camera coordinate systems coincide.

4.2 Images remapping

The camera model obtained via calibration is employed to remap the images as if they were shot with the reference pinhole camera described in Section 3. In this way, the neural network trained on the simulation data obtained via the pinhole model can be deployed to real tactile sensors with different intrinsic and extrinsic parameters, provided that they share the same gel geometry and mechanical properties.

The procedure is based on reprojecting the pixels in the distorted image onto the pinhole image via the corresponding 3D world points. Since the images provide 2D information, only the direction of the corresponding 3D points can be retrieved. To overcome



(a) Original image **(b)** Remapped image

Figure 9. As shown in this figure, the remapping procedure removes most of the image distortion (see (a)), with the particle layer captured as a square in (b).

this limitation, the vertical coordinate of the 3D points is considered to be fixed and known. Since the majority of the particles visible in the image are the ones closer to the camera, this vertical coordinate is set to the lowest point $z^P := t_z^{GP}$ of the silicone layer containing the particles. Fig. 8 depicts the remapping procedure.

For each pixel $p := (u, v)$ in the pinhole image, the respective 3D world approximation $s^P := (x^P, y^P, t_z^{GP})$ is retrieved inverting the pinhole projection (see (6)) as

$$x^P = \frac{t_z^{GP}}{f}(u - u_\mathrm{c}), \qquad y^P = \frac{t_z^{GP}}{f}(v - v_\mathrm{c}). \tag{13}$$

The 3D point is then transformed to the coordinate system of the real-world camera via the appropriate rotation and translation,

$$s^C = R^{GC}\left(R^{GP}\right)^{-1}\left(s^P - t^{GP}\right) + t^{GC}. \tag{14}$$

Note that the reference pinhole extrinsic parameters R^{GP} and t^{GP}, which can be arbitrarily chosen, are set in the vicinity of the expected R^{GC} and t^{GC}, respectively, to limit the impact of the approximation introduced above. These parameters depend on the design and assembly of the real-world tactile sensors. Finally, the corresponding pixel in the real-world image is retrieved via the `world2cam` function. An example of a remapped image is shown in Fig. 9.

5. Results

A fully synthetic dataset including 13,448 vertical indentations over the entire sensing surface was generated as described in Section 3. A stainless steel, spherical-ended cylin-

drical indenter with a diameter of 10 mm was modeled and used for all the indentations. The simulations were of the same type as in [11], where additional details are given.

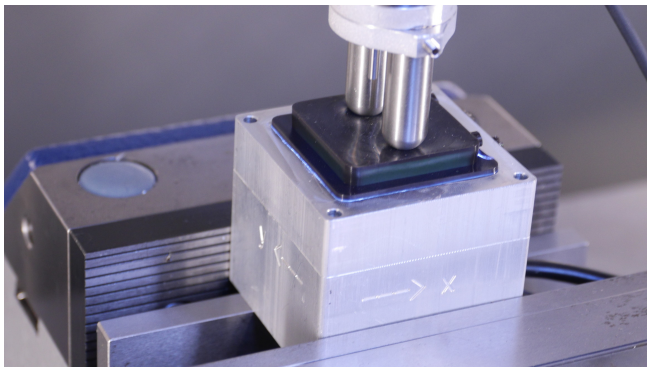
The same setup was prepared in reality, where an equal indenter was attached to the spindle of an automatically controlled milling machine (Fehlmann PICOMAX 56 TOP). In this real-world scenario 200 indentations were performed, where the images were collected and matched to simulated ground truth labels, as described in [11]. These indentations were in the same range as the synthetic data and spanned a depth of 2 mm and normal forces up to 1.7 N. The simulation and real-world scenarios are depicted in Fig. 3.

The neural network was trained on the fully synthetic dataset, by minimizing the mean squared error via the Adam optimizer [25], as implemented in Pytorch. The network was then evaluated on the real-world data, composed of real images and respective synthetic labels. As described in Section 4, the real-world images were remapped to the reference pinhole camera frame via the calibrated camera model, using both the intrinsic and extrinsic parameters. Additionally, a strategy was implemented to compensate for the mismatches introduced during production, assembly and calibration. In fact, in the current setup the exact alignment of the calibration pattern with the gel frame, which is necessary to retrieve the extrinsic parameters, is challenging. In order to take these deviations into account, a single real-world indentation taken in the center of the sensor at a depth of 1.25 mm was used to perform a local refinement of t^{GC}. This refinement was performed via a grid search around t^{GC}, by minimizing the mean squared error between the synthetic and real-world optical flow features. The magnitude of the resulting deviation was 0.7 mm, which is compatible with the hypothesis that this error was mainly introduced during the placement of the calibration pattern. A sample prediction on real-world data after refinement is shown in Fig. 1. The evaluation results on the full real-world dataset are shown in Table 1 for the cases with and without refinement. The RMSE rows show the root mean squared error on the respective components of the 3D contact force distribution. Additionally, the root mean squared error on the total force applied is shown, denoted as RMSET. Note that the total force components can be computed by summing the appropriate force values across all the surface bins. The performance after the one-point refinement is comparable to the resolution of commercial force sensors, as was the case in [11] using real-world training images. The remaining gap is in large part due to artifacts introduced by the DIS algorithm (see Fig. 1) and to the modeling approximations. The convolutional nature of the learning architecture presented here exhibits promising generalization capabilities. In fact, the network was only trained on vertical indentations with a single, spherical-ended indenter, but exhibits sensible predictions for contacts with multiple bodies and objects of different shapes, as shown in Fig. 10. The real-time prediction at 50 Hz on a standard laptop CPU is shown in the video[1] attached to this paper, which also includes the prediction of the horizontal force distribution.

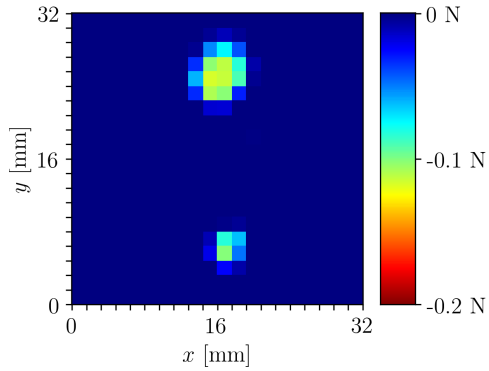
[1]Video: `https://youtu.be/dDTga9PgWS0`

Metric	F_x^G	F_y^G	F_z^G
RMSE	0.002 N	0.002 N	0.005 N
RMSET	0.032 N	0.043 N	0.150 N
RMSE (refined)	0.001 N	0.001 N	0.004 N
RMSET (refined)	0.032 N	0.041 N	0.131 N

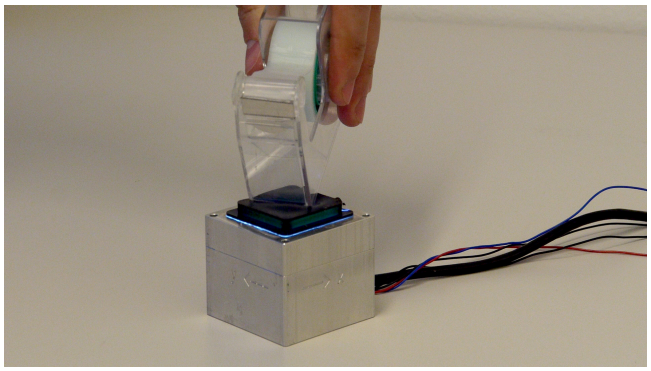
Table 1. Resulting errors on force distribution and total force



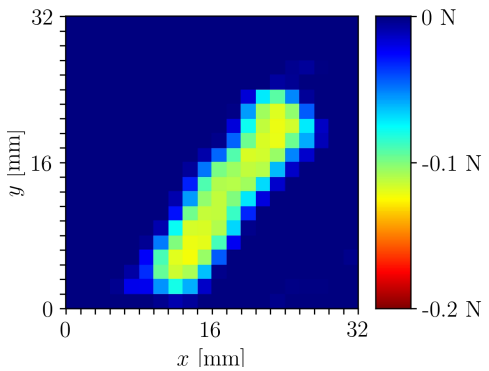
(a) Multi-contact indentation



(b) F_z^G prediction



(c) Generic-shape indentation



(d) F_z^G prediction

Figure 10. In (a), the indentation experiment involving multiple contact points is shown. The network fully trained on simulated single indentations detects both the distinct contact locations (see (b)), as well as the different pressure intensity (the two indenters have indeed a difference in length of 1 mm). Additionally, (c) and (d) show an example of the generalization of the network when sensing the contact with an object of a different shape than the one seen during training.

6. Conclusion

In this paper, a strategy has been presented to train an artificial neural network, which aims to reconstruct the three-dimensional contact force distribution applied to the soft surface of a vision-based tactile sensor. The generation of a fully synthetic dataset enables the training of the network in simulation, exhibiting accurate sim-to-real transfer when evaluated on real-world data. Additionally, the convolutional structure of the network facilitates generalization to a variety of contact conditions.

The remaining errors can mainly be explained by two factors: the discrepancies be-

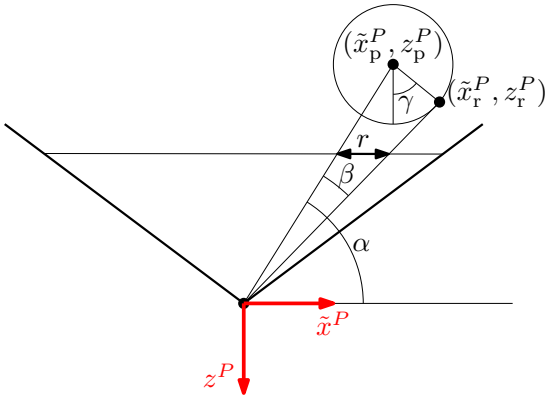


Figure 11. As shown in this figure, the pixel radius r on the image is computed by taking the difference between the pixel coordinate of the particle center and the projection of the point where the image ray is tangent to the particle.

tween real and synthetic optical flow features; and the fact that the elastic deformation noise injected during training and found to be essential for the sim-to-real transfer may excessively deteriorate the information relating to shear forces, which are rather small in the vertical indentation setup. Future work will focus on both these issues, by investigating appropriate noise characteristics that aim to emulate the imperfections of the real optical flow, and by augmenting the dataset with contact conditions that exhibit higher shear forces.

A. Appendix

Let $s_\mathrm{p}^P := \left(x_\mathrm{p}^P, y_\mathrm{p}^P, z_\mathrm{p}^P\right)$ be the position of a particle of radius R in the pinhole camera frame. In general, a spherical particle is projected onto an ellipse in the image plane, which has its major axis in the plane containing the camera optical axis and the camera ray passing through the center of the sphere. A proof of this fact can be found in [26]. Therefore, the fraction r of the major axis can be computed by a 2D analysis on this plane, see Fig. 11, via the following steps:

$$\tilde{x}_\mathrm{p}^P = \sqrt{\left(x_\mathrm{p}^P\right)^2 + \left(y_\mathrm{p}^P\right)^2}, \tag{15}$$

$$\alpha = \arctan\left(-\frac{z_\mathrm{p}^P}{\tilde{x}_\mathrm{p}^P}\right), \quad \beta = \arcsin\left(\frac{R}{\sqrt{\left(\tilde{x}_\mathrm{p}^P\right)^2 + \left(z_\mathrm{p}^P\right)^2}}\right),$$

$$\gamma = \alpha - \beta,$$

$$\tilde{x}_\mathrm{r}^P = \tilde{x}_\mathrm{p}^P + R\sin\gamma, \qquad z_\mathrm{r}^P = z_\mathrm{p}^P + R\cos\gamma,$$

$$r = \left| f\left(\frac{\tilde{x}_\mathrm{r}^P}{z_\mathrm{r}^P} - \frac{\tilde{x}_\mathrm{p}^P}{z_\mathrm{p}^P}\right)\right|. \tag{16}$$

In Section 3, considering the small size of the particles, the projected ellipse is approximated as a circle with radius r, computed as in (16).

Acknowledgments

The authors would like to thank Michael Egli for his support in the sensor manufacture.

References

[1] R. Calandra, A. Owens, D. Jayaraman, J. Lin, W. Yuan, J. Malik, E. H. Adelson, and S. Levine, "More Than a Feeling: Learning to Grasp and Regrasp Using Vision and Touch", *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3300–3307, 2018.

[2] S. Dong and A. Rodriguez, "Tactile-Based Insertion for Dense Box-Packing", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 7953–7960.

[3] P. A. L. S. Martins, R. M. Natal Jorge, and A. J. M. Ferreira, "A Comparative Study of Several Material Models for Prediction of Hyperelastic Properties: Application to Silicone-Rubber and Soft Tissues", *Strain*, vol. 42, no. 3, pp. 135–147, 2006.

[4] K. Weiß and H. Worn, "The Working Principle of Resistive Tactile Sensor Cells", in *Proceedings of the IEEE International Conference Mechatronics and Automation*, vol. 1, 2005, pp. 471–476.

[5] B. Sundaralingam, A. S. Lambert, A. Handa, B. Boots, T. Hermans, S. Birchfield, N. Ratliff, and D. Fox, "Robust Learning of Tactile Force Estimation through Robot Interaction", in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 9035–9042.

[6] J. M. Romano, K. Hsiao, G. Niemeyer, S. Chitta, and K. J. Kuchenbecker, "Human-Inspired Robotic Grasp Control With Tactile Sensing", *IEEE Transactions on Robotics*, vol. 27, pp. 1067–1079, 2011.

[7] W. Yuan, S. Dong, and E. H. Adelson, "GelSight: High-Resolution Robot Tactile Sensors for Estimating Geometry and Force", *Sensors*, vol. 17, no. 12: 2762, 2017.

[8] K. Kamiyama, H. Kajimoto, N. Kawakami, and S. Tachi, "Evaluation of a Vision-based Tactile Sensor", in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, 2004.

[9] D. Ma, E. Donlon, S. Dong, and A. Rodriguez, "Dense Tactile Force Estimation using GelSlim and inverse FEM", in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2019, pp. 5418–5424.

[10] N. Kuppuswamy, A. Castro, C. Phillips-Grafflin, A. Alspach, and R. Tedrake, "Fast Model-Based Contact Patch and Pose Estimation for Highly Deformable Dense-Geometry Tactile Sensors", *IEEE Robotics and Automation Letters*, 2019.

[11] C. Sferrazza, A. Wahlsten, C. Trueeb, and R. D'Andrea, "Ground Truth Force Distribution for Learning-Based Tactile Sensing: A Finite Element Approach", *IEEE Access*, vol. 7, pp. 173 438–173 449, 2019.

[12] C. Sferrazza and R. D'Andrea, "Transfer learning for vision-based tactile sensing", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 7961–7967.

[13] P. Ruppel, Y. Jonetzko, M. Görner, N. Hendrich, and J. Zhang, "Simulation of the SynTouch BioTac Sensor", in *Proceedings of the International Conference on Intelligent Autonomous Systems*, Springer, 2018, pp. 374–387.

[14] D. F. Gomes, A. Wilson, and S. Luo, "GelSight Simulation for Sim2Real Learning", in *ICRA ViTac Workshop*, 2019.

[15] B. Wu, I. Akinola, J. Varley, and P. K. Allen, "MAT: Multi-Fingered Adaptive Tactile Grasping via Deep Reinforcement Learning", in *Proceedings of the Conference on Robot Learning (CoRL)*, 2019.

[16] H. Park, H. Lee, K. Park, S. Mo, and J. Kim, "Deep Neural Network Approach in Electrical Impedance Tomography-based Real-time Soft Tactile Sensor", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 7447–7452.

[17] C. Sferrazza and R. D'Andrea, "Design, Motivation and Evaluation of a Full-Resolution Optical Tactile Sensor", *Sensors*, no. 4: 928, 2019.

[18] T. Kroeger, R. Timofte, D. Dai, and L. Van Gool, "Fast Optical Flow using Dense Inverse Search", in *Proceedings of the European Conference on Computer Vision*, Springer, 2016, pp. 471–488.

[19] L. Mitas and H. Mitasova, "Spatial interpolation", in *Geographical Information Systems: Principles, Techniques, Management and Applications*, P. A. Longley, M. F. Goodchild, D. J. Maguire, and D. W. Rhind, Eds., John Wiley & Sons, 1999.

[20] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2010.

[21] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, 2015, pp. 234–241.

[22] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial Transformer Networks", in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 2017–2025.

[23] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis", in *Proceedings of the International Conference on Document Analysis and Recognition*, 2003, pp. 958–963.

[24] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A Toolbox for Easily Calibrating Omnidirectional Cameras", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006, pp. 5695–5701.

[25] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization", in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[26] D. S. Wokes and P. L. Palmer, "Perspective Projection Of A Spheroid Onto An Image Plane", 2008.

Paper P5

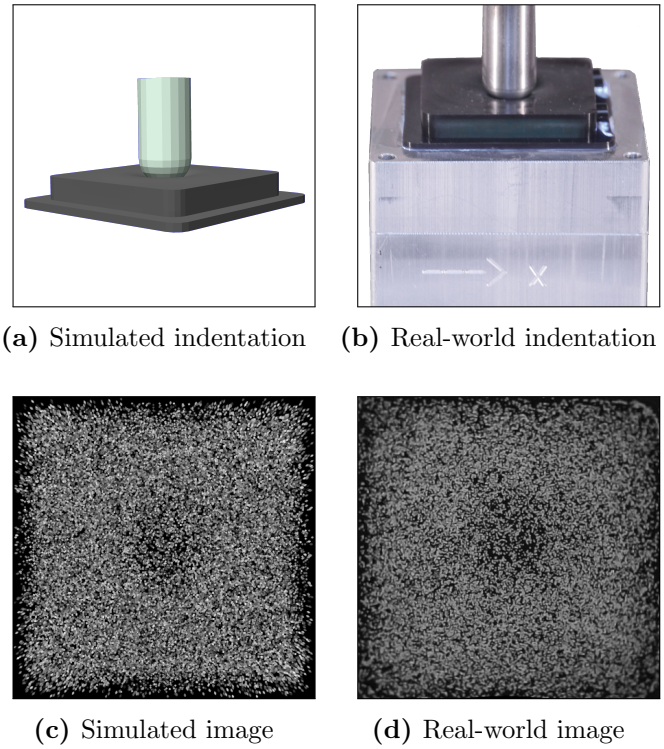
Sim-to-real for high-resolution optical tactile sensing: From images to three-dimensional contact force distributions

Carmelo Sferrazza and Raffaello D'Andrea

Abstract

The images captured by vision-based tactile sensors carry information about high-resolution tactile fields, such as the distribution of the contact forces applied to their soft sensing surface. However, extracting the information encoded in the images is challenging and often addressed with learning-based approaches, which generally require a large amount of training data. This article proposes a strategy to generate tactile images in simulation for a vision-based tactile sensor based on an internal camera that tracks the motion of spherical particles within a soft material. The deformation of the material is simulated in a finite element environment under a diverse set of contact conditions, and spherical particles are projected to a simulated image. Features extracted from the images are mapped to the 3D contact force distribution, with the ground truth also obtained via finite-element simulations, with an artificial neural network that is therefore entirely trained on synthetic data avoiding the need for real-world data collection. The resulting model exhibits high accuracy when evaluated on real-world tactile images, is transferable across multiple tactile sensors without further training, and is suitable for efficient real-time inference.

Published in *Soft Robotics*.



(a) Simulated indentation **(b)** Real-world indentation

(c) Simulated image **(d)** Real-world image

Figure 1. This work builds upon the generation of training images in simulation for a data-driven, vision-based tactile sensor based on the tracking of a spread of particles.

1. Introduction

Research on vision-based (or optical) tactile sensors aims to provide robots with high-resolution information about contact with external objects. However, while the images stemming from the various optical tactile sensing principles are intuitive and to some extent interpretable by human observations, the extraction of accurate physical quantities is challenging. In this regard, the complexity of mapping the information extracted from the images to the corresponding contact conditions mainly results from the fact that accurate modeling techniques for soft materials are generally not suitable for real-time applications. Additionally, previous research has predominantly focused on the estimation of low-dimensional quantities (e.g., total contact force, center of contact), which may be sufficient for a limited range of tasks, but not for generic applications, as is the case for tasks that involve arbitrary points of contact.

The work discussed in this article targets both these topics, proposing a data-driven approach to reconstruct the three-dimensional distribution of the contact forces applied to the soft surface of a vision-based tactile sensor. The sensing strategy was presented in the authors' previous work [1] and is based on the tracking of particles randomly spread within a soft gel. The use of data bypasses the need for modeling techniques with real-time guarantees, but as opposed to classical data-driven strategies, here the data necessary for training the learning architecture at the core of the method are entirely generated in

simulation. Furthermore, the estimation of the contact force distribution directly yields both the total contact force (i.e., the component-wise integral of the force distribution) and the contact locations (i.e., the surface patches where the contact pressure is nonzero), and is additionally suitable to represent generic contact conditions with arbitrary points of contact, therefore providing high versatility across several tasks.

The main contributions of this work are the following:

- It details a method to simulate the images captured by a vision-based tactile sensor [1], starting from simulations based on the finite element method [2] (FEM).
- It outlines two strategies to generate simulated datasets comprising tactile image features and labels. These strategies differ from the one presented in the authors' previous work [3], as they relax a small deformation assumption and simplify the transfer from simulation to reality. The datasets collected for this work comprise a variety of contact conditions, producing high shear and pressure forces with indenters of different shapes and sizes.
- It describes a tailored learning architecture, based on u-net [4], which can be trained entirely with simulated data obtained offline via high-fidelity FEM-based simulations. When evaluated on real-world tactile sensors, the architecture yields high accuracy in the reconstruction of the force distribution, achieving real-time inference up to a speed of 120 Hz.

1.1 Related work

In recent years, a number of tactile sensing principles [5] have been developed to address the needs of the robotics community. Among these, vision-based tactile sensors [6] employ standard cameras [7] or optical devices [8], [9] to infer the deformation of a soft membrane and obtain information about the contact with external objects that causes the deformation. This category of tactile sensors generally benefits from high resolution and ease of wiring, and its straightforward manufacture enables fast prototyping for robotic systems. Although the bulkiness of their sensing unit is the main limitation of such approaches, recent works have proposed compact solutions that exploit embedded cameras [10]–[14] or mirrors [15].

The sensory feedback provided by tactile sensors typically requires further processing, as it does not directly translate to the physical quantities of interest for robotic tasks. In this regard, model-based methods [16], [17] often rely on strong modeling assumptions (e.g., linear elasticity of the materials) to solve the processing task in an approximate fashion, while data-driven methods [18]–[20] aim to compute offline a mapping from raw data to the quantities of interest, in order to preserve accuracy while ensuring real-time inference.

While most of the literature has primarily focused on the estimation of low-dimensional physical quantities (e.g., total forces), recently several works have shifted the focus towards the estimation of distributed quantities, which aim to provide high-resolution tac-

tile fields for a wide range of tasks. In the context of vision-based sensors, the estimation of the contact patches [17] has been proposed, and the reconstruction of the contact force distribution has been discussed, both in a model-based [16] and a data-driven [18] fashion. Additionally, various approaches have been proposed outside the vision-based domain, with regard to the estimation of the deformation field [21] and the pressure distribution [22].

As a result of the possibility of collecting and generating accurate data offline, data-driven approaches generally exhibit smaller estimation errors than model-based methods [8]. However, their bottleneck often lies in the fact that they require large amounts of training data and they do not often generalize well when employed in unseen contact conditions. In order to address the issue of data efficiency, a number of works have focused on generating training data in simulation to extract a model that retains its accuracy when employed in the real world. Examples of such sim-to-real (or sim2real) transfers can be found in the literature for edge prediction [23] and the estimation of the contact pressure [22] and the deformation field [24], [25]. In previous work, a sim-to-real approach was presented to estimate the 3D force distribution [3] for a limited range of scenarios.

This article presents two different methods to generate a dataset to train a data-driven approach entirely via FEM simulations, with the aim to reconstruct the three-dimensional contact force distribution applied to a vision-based tactile sensor. Image features were extracted from the tactile images generated in simulation, and mapped to three matrices representing the components of the force vectors applied over the soft sensing surface. The mapping was obtained via a tailored neural network architecture, which is able to capture various contact conditions as high shear and pressure forces, as well as indentations with flat or round objects. Additionally, high accuracy was retained on real-world data and the real-time speed could be more than doubled compared to previous work [3].

1.2 Outline

The sensing strategy and the hardware are described in Section 2.1, while the method to generate tactile images and extract the related features is presented in Section 2.2. Starting from the generated dataset, Sections 2.3 and 3 describe the learning pipeline and the evaluation on simulated and real-world data, respectively. Final remarks and an outlook are included in Section 4.

2. Materials and methods

2.1 Hardware

The tactile sensor employed in this work is based on a camera that tracks particles randomly distributed within a deformable material. The fabrication follows previous work [1], and is detailed in Section 1 of the supplementary material. The sensing surface amounts

to a rectangular prism of 32×32×6 mm. The soft materials have been characterized previously [18] as hyperelastic materials following uniaxial, pure shear and equibiaxial tension tests. The resulting second-order Ogden models [26] were employed for the FEM simulations discussed in the following sections.

2.2 Dataset generation

Supervised learning is a natural data-driven way of processing sensory feedback and mapping raw data to the quantities of interest. In the context of vision-based tactile sensing, formulating the task in a supervised learning manner involves two crucial preliminary steps: I) the choice of appropriate features to condense the information contained in the images; II) the formalization of finite-dimensional labels representing the quantities of interest. Additionally, the availability of data necessary to train suitable learning architectures needs to be considered when addressing the formulation of the problem. In this work, training data were generated entirely in a finite element simulation environment with the objective of avoiding real-world data collection and maximizing the variability of the contact conditions without the need for complex hardware setups. A further advantage of collecting contact data in simulation is the possibility of extracting high-resolution tactile fields [18], which are otherwise not possible to measure with the commercially available commodity sensors. This work aimed to estimate the three-dimensional force distribution, which is a condensed representation of several contact quantities. In fact, the force distribution encodes both the contact locations, which can be obtained by thresholding the normal component, and the total contact forces, which can be obtained by integrating the distribution over the sensing surface. As opposed to the deformation field, the contact patches are exactly encoded in the force distribution, while the deformation field can, for example, show deformation also where no contact is applied, as a result of the elasticity of the soft material. Additionally, from the force distribution it is possible to compute the torques acting on the contact object, and all these properties remain valid for contact with multiple or arbitrary objects.

An FEM simulation environment was created in Abaqus/Standard [28], details of this are provided in Section 2 of the supplementary material and in a previous work [18]. Two training datasets were built by performing indentations in such an FEM environment with the 21 different indenters shown in Fig. 2. The indentation trajectories were performed by either moving the indenter vertically and then purely horizontally, or by prescribing indenter motions from different angles followed by random perturbations in the vicinity of the first indentation. A total of 3300 indentation trajectories (each comprising 50 indentation steps) were executed in simulation, with total forces up to 16 N in the vertical direction and up to 5 N in each of the horizontal directions. For each step of these trajectories, the contact force distribution and the displacement field were extracted at the nodes of a mesh refined around the contact between the indenter and the soft material. These quantities were further processed to compose two sets of features and labels, as described in Section 1 for the displacement field and Section 2 for the force distribution. Since the training dataset was entirely generated in simulation, two test datasets were

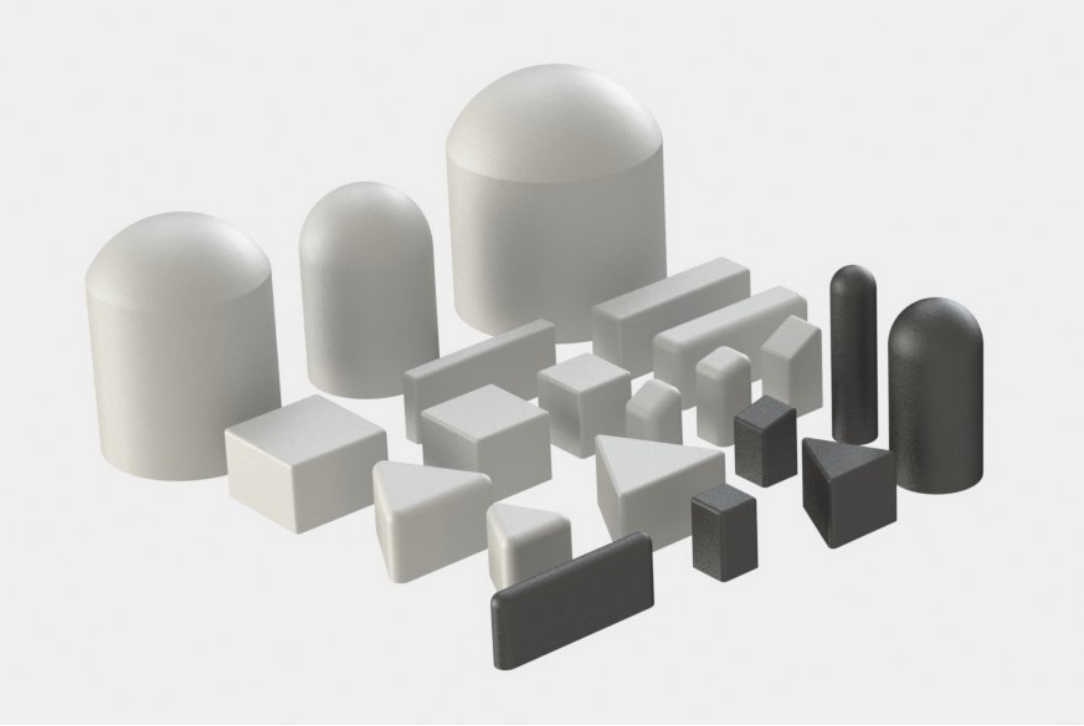


Figure 2. The figure shows the indenters used to collect the training data in the FEM simulations. Real-world realizations of the black indenters were used to collect the test data in reality. Note that the indentation surfaces correspond to the top surfaces in the figure. The sharp corners of the indenters were smoothed out to avoid a known singularity in the flat-punch indentation experiment [27].

collected in reality as described in Section 3 to verify the sim-to-real transfer and the real-world performance.

1) Training features In this article, two different methods to extract image features are compared. The resulting types of features are denoted in the following as optical flow features and raw features, respectively. The starting points of both methods are the images captured by the internal camera, and for training purposes, these images were entirely generated in simulation. The soft materials were modeled in the FEM simulations as described in Section 2.1 of this manuscript and in Section 2 of the supplementary material. Highly accurate models were obtained for the same materials via state-of-the-art characterization experiments in previous work [18], where these models were also validated against a force-torque sensor. The Ogden model parameters used there were also employed in this work. A static friction coefficient of 0.9 was used, as it proved accurate for the indenters employed (see the experiments performed in Section 2 of the supplementary material).

A gel coordinate system (see Fig. 3(a)) was defined by placing the origin at one of the bottom corners of the layer containing the particles, the z axis pointing towards the upper surface, and the x and y axes aligned with two of the horizontal edges. For each indentation step performed in simulation, the FEM provides the displacement field of

the soft layer that comprises the particles. This displacement field is provided at the discrete nodes of the FEM mesh. For such nodes, also the initial position (at rest, before deformation) is known. In order to generate the dataset, a random distribution of particles was sampled for each indentation step, and an inverse distance weighted scheme [29] was used to interpolate the displacement field at the corresponding particle location s_j^G, for $j = 0, ..., N_\mathrm{p} - 1$, where N_p is the number of particles and the superscript G indicates the gel coordinate system. The 3D displacement of the j-th particle is denoted in the following as Δs_j^G. The strategy followed was to project the particles to the image plane using an ideal pinhole camera model [30], and only account for the camera's non-idealities at a later stage [3], as described in Section 3. Therefore, as depicted in Fig. 3, the position s_j^G and the respective displacement Δs_j^G were first transformed from the gel coordinate system to the 3D pinhole camera coordinate system (indicated by the superscript P) as

$$s_j^P = R^{GP} s_j^G + t^{GP}, \tag{1}$$

$$\Delta s_j^P = R^{GP} \Delta s_j^G \tag{2}$$

where the rotation matrix R^{GP} and the translation vector $t^{GP} := (t_x^{GP}, t_y^{GP}, t_z^{GP})$ are the pinhole camera's extrinsic parameters. These parameters could be chosen arbitrarily, but they were actually chosen to be close to the real-world camera's extrinsic parameters, as discussed in Section 3. The pinhole image resolution was arbitrarily set to be 440×440 pixels, and although the focal length could also be chosen arbitrarily in this step, in order to exactly capture the region where the particle layer (which has a square horizontal section of 30×30 mm) is visible, this was set for both the image coordinates as

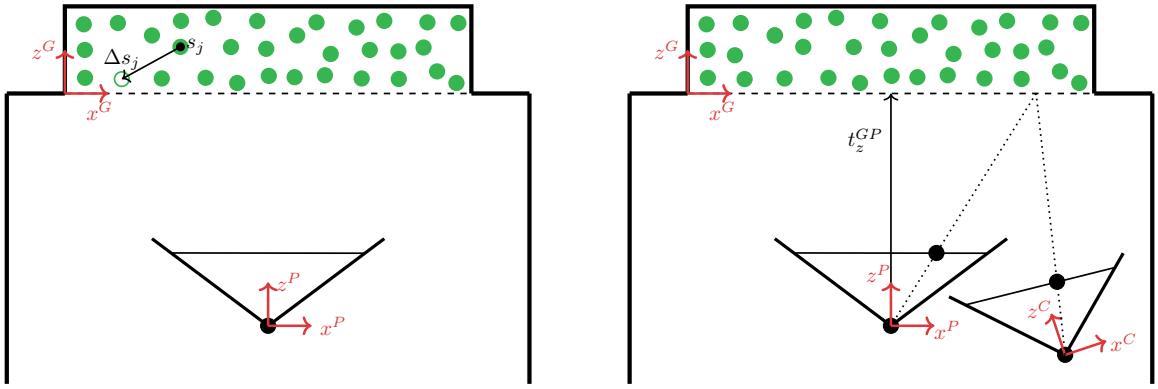
$$f := \frac{440}{30} t_z^{GP}. \tag{3}$$

The projection of the spherical particle centered at s_p^P via the pinhole camera model results in an ellipse on the image plane [31]. The derivation of the center, the axis lengths and the orientation of each ellipse can be found in Section 3 of the supplementary material. The ellipses can then be drawn using the drawing functionality of OpenCV[1].

For each indentation step, one image at rest (projecting all the particles, i.e., by setting $s_\mathrm{p}^P = s_j^P$ for the j-th particle) and one image after deformation (setting $s_\mathrm{p}^P = s_j^P + \Delta s_j^P$ for the j-th particle) were generated. The images were initialized with black pixels, and each ellipse was drawn with a random RGB color to perturb the data with additional variability. The images were then converted to grayscale in a second step. An example of a simulated image is shown in Fig. 1(c). In order to further increase the training robustness, the number of the particles within the gel was slightly perturbed at each indentation step. Training features were then extracted from the images via two different methods:

1. Optical flow features: For each indentation step, the dense optical flow between

[1] https://opencv.org/



(a) Displacement of a particle **(b)** Remapping of a pixel

Figure 3. The drawings show the definition of the three coordinate systems used throughout the article: the gel coordinate system (superscript G), the pinhole camera coordinate system (superscript P), and the real-world camera coordinate system (superscript C). In (a), an example of 3D displacement of a particle originally placed at s_j is depicted. In (b), a pixel in the pinhole camera is mapped to the corresponding pixel in the real-world camera.

the image at rest and the image after deformation was computed using an algorithm based on Dense Inverse Search [32]. The per-pixel flow was then subsampled performing an average pooling in a grid of 88×88 bins. The two Cartesian components of the optical flow resulted in two matrices, which were concatenated into a two-channel matrix. This method differs from previous work [3], where optical flow features were directly computed from the FEM displacement field, assuming that the density of the particles remained constant during an indentation. In reality, this is not the case for large indentations, as the particles tend to spread radially under pressure, and the method presented here can cope with such conditions.

2. Raw features: The two images for each indentation step were subsampled to 88×88 pixels and concatenated into a two-channel image, which was directly fed to the training algorithm.

2) Training labels The same set of labels described in the following was assigned to each set of features to compose two separate training datasets. For each indentation step, the FEM simulations provide the three-dimensional contact force distribution at the surface nodes of the FEM mesh. Dividing the surface into a grid of 20×20 bins [18], the force components at the nodes falling inside a bin were summed to obtain a 20×20 three-channel matrix, representing the training label for the corresponding indentation step datapoint. Examples of ground truth labels are shown in Fig. 8. In this work, a node was assigned to a certain bin depending on its initial position before deformation, in order to simplify the binning at the boundaries of the gel, which can vary with deformation. As an alternative, it would also be possible to assign the nodes to the bins according to the position after deformation, by introducing an adaptive binning strategy at the boundaries of the grid.

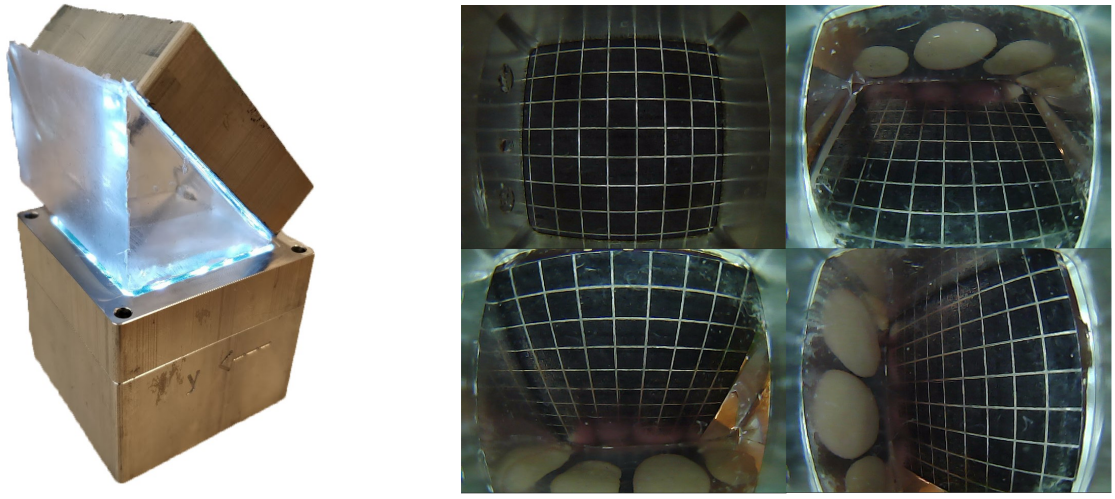
3) Test dataset In order to evaluate the real-world performance of the models described in Section 2.3, 1100 test datapoints were collected in an experimental setup, using a programmable milling machine (Fehlmann PICOMAX 56 TOP) to make vertical and shear-dominant indentations with the six black indenters shown in Fig. 2, as well as multi-contact indentations with two spherically-ended indenters placed at different heights. The resulting test dataset induced total forces up to 4.5 N in the vertical direction and up to 3.8 N in each of the horizontal directions. These ranges differ from the training data ranges, which also included data inducing larger strains and where the material model fit was less accurate. Such higher-strain data showed improved generalization in the learning and for this reason were included in the training dataset.

During the test data collection procedure, the images taken by the real-world camera were recorded. Since the models were trained with features obtained from images generated via a pinhole camera projection, a further procedure was needed to account for the camera's non-idealities on real-world images [3]. This procedure is denoted as remapping and it essentially maps the pixels from a real-world image (converted to grayscale) to the pixels of an image of the same scene as if it was taken from the ideal pinhole camera used for the training dataset. The remapping procedure requires two main steps:

1. Calibration step: during fabrication, seven images of a grid pattern were shot through a silicone medium, see Fig. 4. In this way, it is possible to account for the refractive index of the soft materials. Using a fisheye camera calibration toolbox [33], the images were used to obtain both the extrinsic parameters R^{GC} and t^{GC} of the real-world camera as well as a transformation function from the actual camera 3D coordinate system to the real-world image. The extraction of the extrinsic parameters was achieved by providing the calibration toolbox with a calibration image where the origin of the grid pattern coincided with the origin of the gel coordinate system.

2. Interpolation step: for each pixel in the fictitious pinhole image, the corresponding pixel in the real-world image was obtained, via a procedure sketched in Fig. 3(b). For this step, the pixels were assumed to be placed approximately at a fixed z coordinate in the pinhole camera coordinate system, set here with the bottom of the gel layer. The details of the interpolation procedure are further detailed in Section 4 of the supplementary material.

   As shown in Fig. 3(b), the approximation introduced above has a smaller effect when the pinhole extrinsic parameters R^{GP} and t^{GP} are close to the real-world camera extrinsic parameters R^{GC} and t^{GC}, respectively. As mentioned in Section 1, since the pinhole extrinsic parameters can be set arbitrarily, these were indeed chosen to be close to the expected real-world extrinsic parameters to limit the impact of the approximation. While the calibration parameters are fixed across images of the same camera and can be computed offline, the interpolation step needs to be performed for each image.

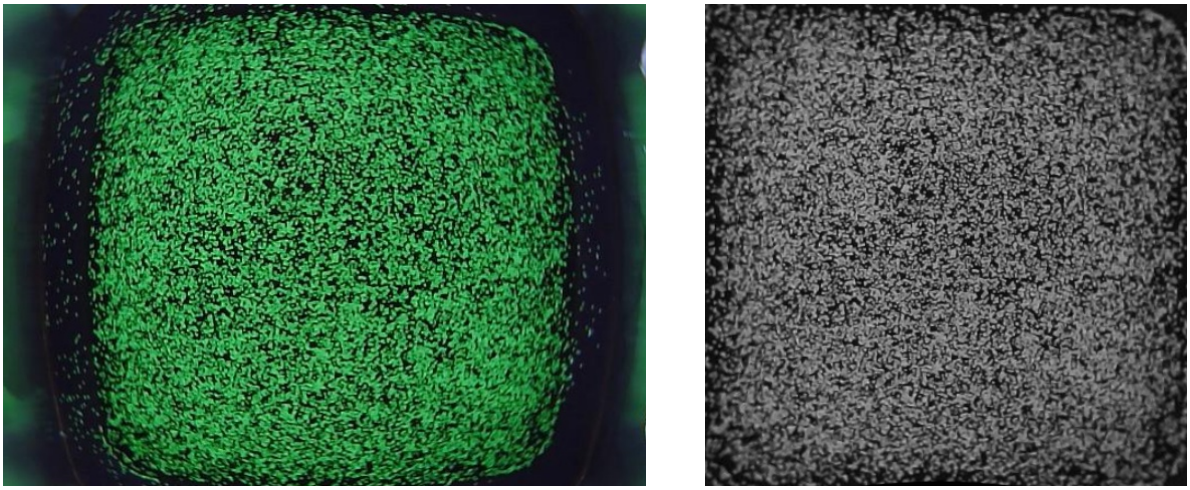


(a) Calibration setup **(b)** Sample calibration images

Figure 4. The calibration images, examples of which are shown in (b), were shot through a silicone medium during fabrication, in order to account for the refraction index of the soft materials. As shown in (a), this was done straight after casting the first layer, by placing additional silicone parts between the first layer and a grid pattern attached to an aluminum surface.

The extrinsic parameters obtained during calibration are very sensitive to the exact placement of the grid pattern for the corresponding calibration image. This requires pressing the grid pattern against the silicone medium just enough to remove the air in the middle without penetrating the soft material, which is challenging to achieve in reality. Therefore, a grid search (in the submillimeter range) was performed in the vicinity of the translation vector t^{GC}, in order to make the particles in a sample remapped image taken at rest match the entire image frame. For this, after a series of dilation and erosion steps, a bounding box around the pixels can be easily computed using OpenCV and compared to the frame boundaries. A refined, remapped image is shown in Fig. 5, where lens distortion effects and misalignments were successfully compensated for.

After remapping, the same image features described in Section 1 were extracted from the images. Since no real-world sensor can provide ground truth contact force distributions, these were extracted in simulation as described in Section 2 and in previous work [18], and assigned to the corresponding features to compose two test datasets. Note that since the real-world camera non-idealities can be compensated in the remapping step, which does not affect training, this enables the transfer of models trained on the pinhole data across multiple instances of fabricated sensors, provided that the camera calibration is performed as described above. The remapping procedure described here aims to compensate only for the camera mismatches and does not serve as a calibration for the FEM model, which was independently characterized in previous work [18], as further detailed in the supplementary material.



(a) Original image **(b)** Image before feature extraction

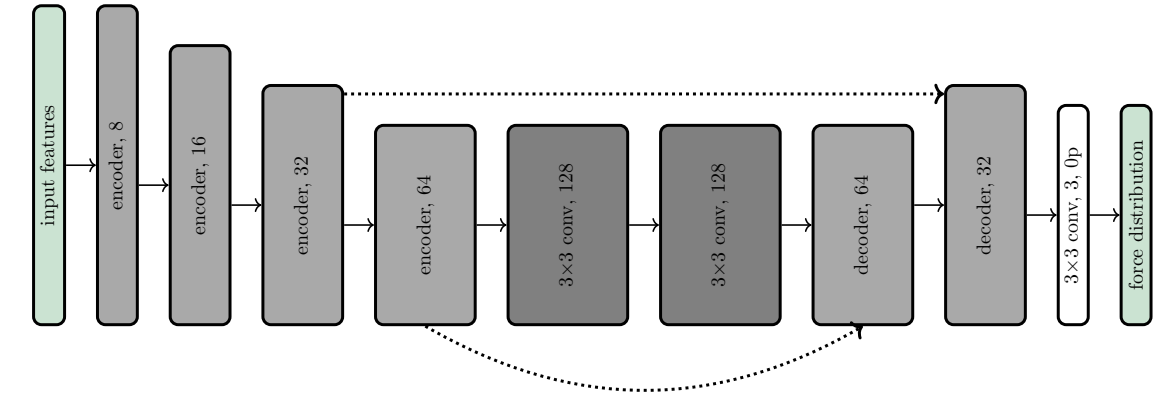
Figure 5. The original image taken from the real-world camera, shown in (a), was converted to grayscale and remapped as if it was taken from the ideal pinhole camera. A refinement procedure was applied to account for inaccuracies introduced during calibration. The resulting image in (b) shows the particle layer in its actual squared geometry, covering the entire image frame.

2.3 Learning architecture

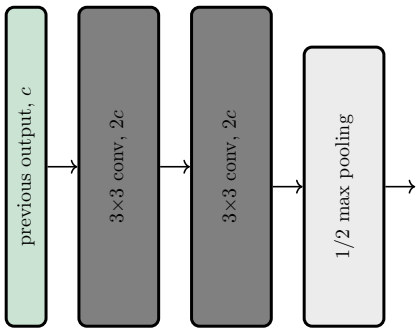
The same learning architecture was employed for both training datasets, that is, on those containing optical flow features and raw features, respectively. The architecture consists of a convolutional neural network, designed as a lightweight version of u-net [4], and tailored to the estimation of the force distribution from tactile features. In fact, this estimation problem can be formulated as an image-to-image translation [34] (known also as pixel-wise regression). A sketch of the architecture is shown in Fig. 6. The neural network exhibits an encoder-decoder structure, where feature information is first increased in the contraction step by doubling the channels between each pooling operation. In the decoding step, the force distribution is then computed through upconvolutions and concatenations of high-resolution features extracted during the contraction step. As a result, the architecture has the effect of both capturing context and enabling precise localization.

3. Results

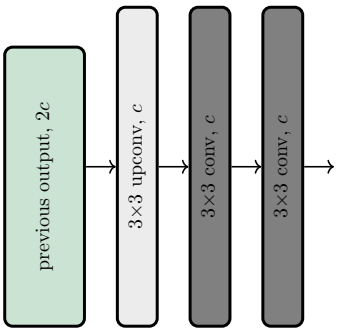
The learning architecture was trained twice from scratch using I) the training dataset comprising averaged optical flow features and discretized force distribution labels, and II) the training dataset comprising raw image features and discretized force distribution labels. Both datasets were generated entirely in simulation and both sets of training features contained two-channel 88×88 matrices (or images), as described in Section 1. The architecture was trained with the AdamW optimizer [35] by minimizing a mean-squared loss (normalized by the maximum value per channel) with a learning rate of



(a) The learning architecture



(b) An encoder block



(c) A decoder block

Figure 6. In (a), a diagram of the learning architecture is shown. The encoder and decoder blocks are summarized in (b) and (c), respectively. All the blocks in green serve as placeholders. "3×3 conv, c" indicates a convolutional layer with a 3×3 filter size and c output channels, while "3×3" upconv, c" indicates an upconvolution that doubles the input size. The dotted lines (omitted in (b) and (c)) indicate the concatenation of an earlier layer output with upsampled information. After each convolutional layer, with the exception of the white one before the final output, batch normalization and rectified linear units were employed. "0p" indicates no padding. Where not indicated, all convolutional filters have unit zero-padding and unit stride.

1e-3 and a batch size of 256. During training, the datasets were randomly augmented by appropriately flipping the features and labels, exploiting the symmetry of the gel geometry and the pinhole camera projection. For the raw-feature dataset, the images were additionally augmented by perturbing the image brightness and adding salt-and-pepper noise.

After training in PyTorch[2], the models were converted to the ONNX format, and used in real-time via the ONNX Runtime framework[3]. This generally led to a 4x inference speed-up on the CPU of a standard laptop (dual-core, 2.80 GHz), compared to the inference in PyTorch.

The performance of both trained models was evaluated on the corresponding synthetic validation datasets, picked randomly as the 20% of the indentation trajectories in the ap-

[2]https://pytorch.org/
[3]https://www.onnxruntime.ai/

	RMSE [N]			RMSET [N]			Range of total forces [N]		
	x	y	z	x	y	z	x	y	z
Optical-flow (sim)	0.006	0.006	0.013	0.187	0.164	0.577	-5.0 – 5.0	-5.0 – 5.0	-16.0 – 0
Raw-feature (sim)	0.006	0.005	0.012	0.120	0.132	0.314	-5.0 – 5.0	-5.0 – 5.0	-16.0 – 0
Optical-flow (real)	0.006	0.007	0.018	0.190	0.230	0.914	-3.2 – 3.2	-3.8 – 3.8	-4.5 – 0
Raw-feature (real)	0.005	0.007	0.014	0.267	0.296	0.362	-3.2 – 3.2	-3.8 – 3.8	-4.5 – 0

Table 1. The table shows the error metrics of the trained models on the validation datasets extracted in simulation and the test datasets collected in reality, for both the cases where optical flow features and raw features were used as inputs.

	MAE [N]			SDAE [N]		
	x	y	z	x	y	z
Optical-flow (bin)	0.001	0.001	0.004	0.006	0.007	0.018
Raw-feature (bin)	0.001	0.001	0.003	0.005	0.007	0.014
Optical-flow (total)	0.103	0.110	0.645	0.159	0.202	0.648
Raw-feature (total)	0.099	0.107	0.238	0.248	0.275	0.273

Table 2. The table shows additional error metrics on the real-world test sets in terms of the absolute errors for bin-wise and total force predictions, namely the mean absolute error (MAE), and the standard deviation of the absolute errors (SDAE).

propriate training dataset. Additionally, the models were evaluated on the corresponding real-world test dataset described in Section 3. Table 1 summarizes the results based on two different error metrics for each force component: I) RMSE, that is the root-mean-squared error on the respective component of the force distribution, II) RMSET, that is the root-mean squared error on the respective component of the total force, which was obtained by summing the force distribution over all the bins. The range of total forces in the corresponding dataset is also shown in the table. In addition, Table 2 reports the mean and the standard deviation of the absolute errors, for the bin-wise and total force predictions on the real-world test data.

As the numerical results indicate, there is a slight difference in accuracy between the horizontal and vertical components of the predictions. This may be explained by the fact that during the vertical indentations, the shear forces were rather small or close to zero. More importantly, the raw-feature model outperformed the optical-flow model in most of the metrics on the corresponding real-world test dataset. In fact, while in practice the location accuracy for both models was similar, the optical-flow features tend to mitigate the differences across indenters under real-world noise, therefore resulting in inaccurate force predictions. On the other hand, overall the raw-feature model showed a better transfer from simulation to reality, especially retaining a considerably higher accuracy in the vertical component. In addition to the difference in accuracy, the model trained on the raw image features does not require the extraction of the optical flow, which was the bottleneck for the model trained on optical flow features. As a result, since the model inference only takes about two milliseconds, the whole raw-feature pipeline (including the image acquisition and remapping) runs in real-time at 120 Hz on CPU, compared to the

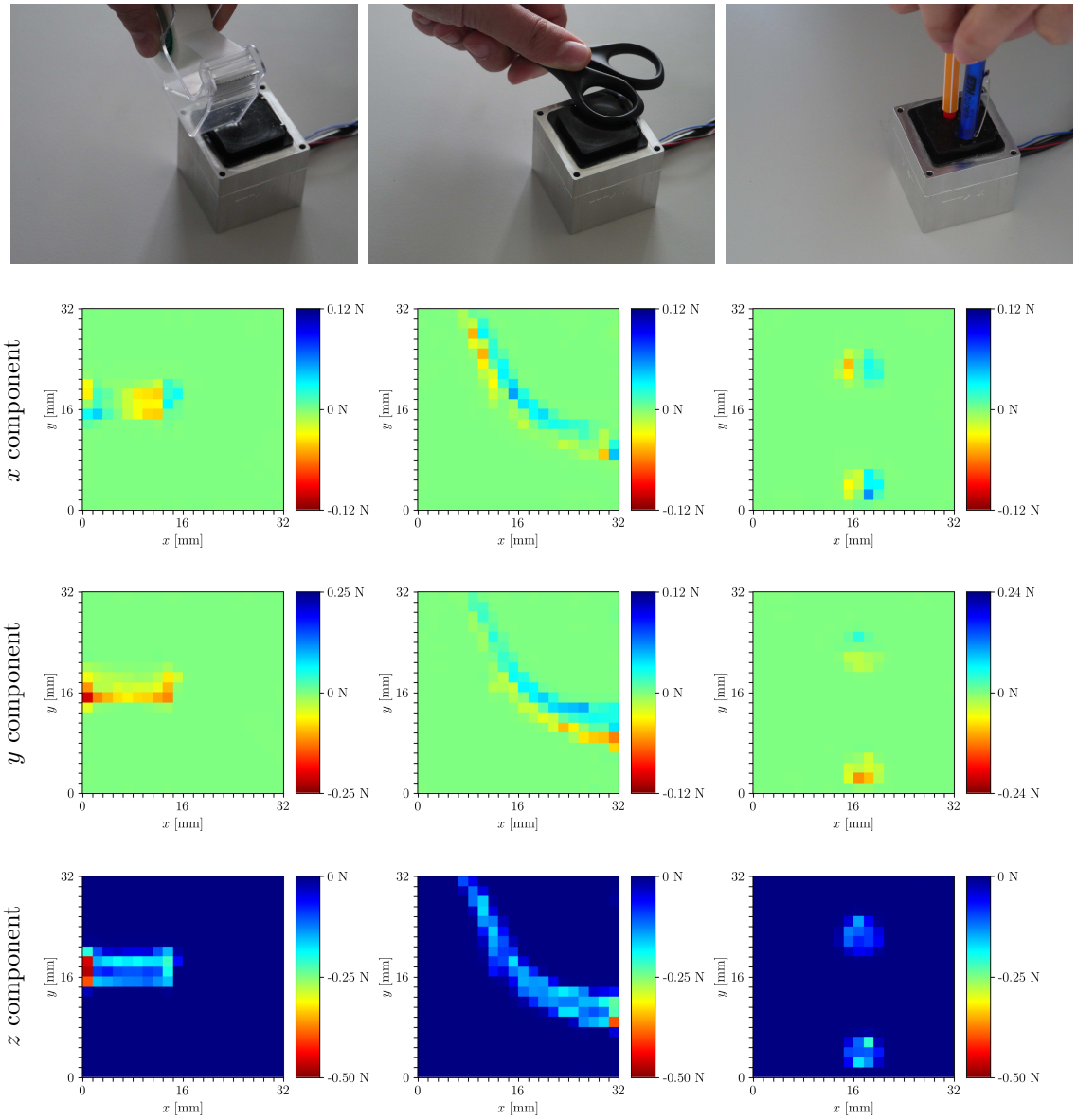


Figure 7. The figures show sensible predictions for the different contact conditions that are shown in the first row. The x, y, and z components of the predicted force distributions are shown in the second, third, and fourth rows, respectively. In the first column, the tape dispenser was initially pressed against the gel and then translated to induce higher shear forces in the negative y direction. In the second column, the contact with an object that differs significantly from those in the training set is shown, while the third column shows the contact with multiple bodies (not included in the training data), the lower of which is laterally translated as shown by the asymmetrical shear component in the y direction.

50 Hz of the optical-flow pipeline.

The real-time performance of the raw-feature pipeline is shown in the supplementary video[4], where contact conditions with arbitrary objects were explored (see Fig. 7). The estimation of the force distributions on samples in the test set with the model trained

[4]Supplementary video: `https://youtu.be/dvOk2XrSmLE`

on raw features are shown in Fig. 8. Additional results and comparisons are available in Section 5 of the supplementary material.

4. Conclusion

This work has discussed strategies to simulate the images captured by a vision-based tactile sensor. Starting from FEM simulations, the displacement field was processed to generate training features for a supervised learning architecture that mapped these features to contact force distribution labels. The resulting models are directly transferable across multiple instances of real-world sensors, since the training procedure does not make use of real-world images. Two different strategies were compared, with the model obtained from raw features outperforming a model based on optical flow features for both real-world accuracy and inference speed. In addition to providing a physical quantity directly interpretable across robotic tasks, the extraction of accurate force distributions also provides an abstraction from the image pixels that bypasses the remaining mismatch between real and simulated images.

Since this work aimed to provide a comparison between the two approaches, the same input and output sizes were employed for both strategies. However, given the gain in prediction speed, the raw-feature approach may be extended to use higher-resolution features or to predict the force distribution on a finer grid by trading off the sensing frequency. As shown in Table 1, a gap still remains between simulation and reality, which could be addressed by explicitly addressing the domain transfer problem. This issue will be the subject of future work.

The simulator described in this work provides highly accurate force distribution labels to train learning-based models suitable for real-time inference. However, the simulator itself is not running in real-time, due to the computational complexity of the finite element method. While this was not in the scope of this work, different simulation techniques can trade off accuracy to achieve real-time capabilities and become suitable to warm-start the training of tactile policies in simulation, as detailed in a related work [36].

A. Fabrication

The soft materials are arranged in three layers, as shown in Fig. 9(a), and were poured (after degassing in a vacuum chamber) on top of the camera (ELP USBFHD06H with a fisheye lens) and the surrounding LEDs with the mold lying on one side, i.e., with the camera pointing sideways, as shown in Fig. 9(b). The fabrication strategy followed previous work [1], but it is presented extensively here to provide additional details for reproducibility:

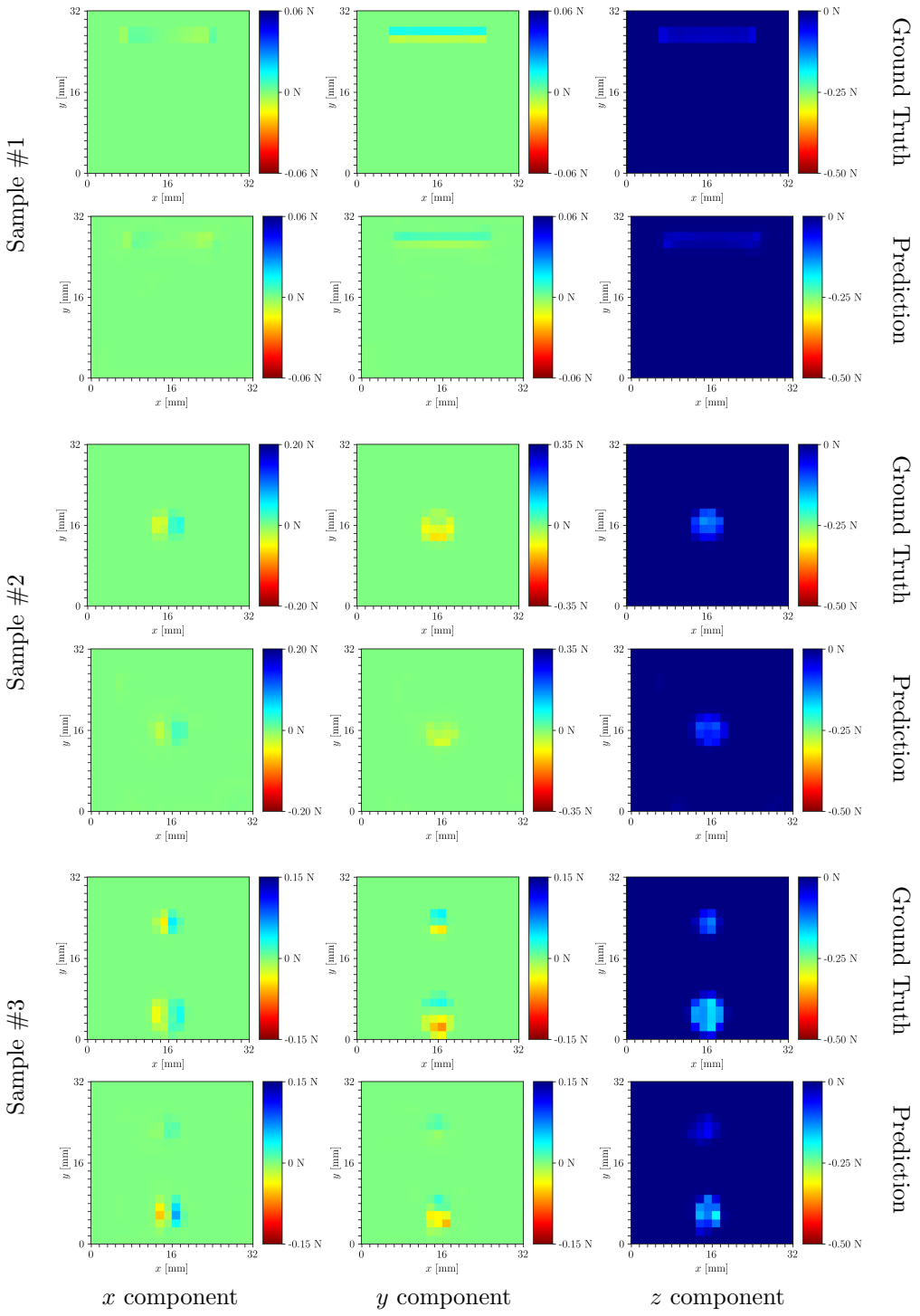
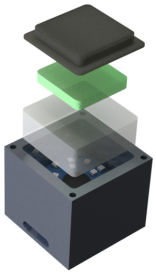
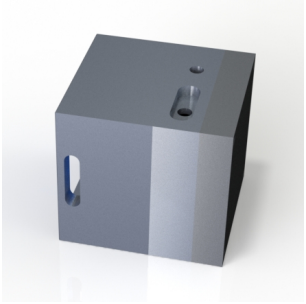


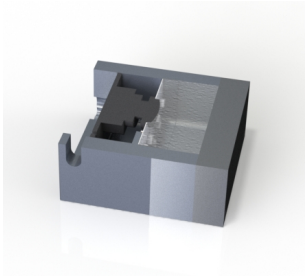
Figure 8. The figures show the ground truth (first, third, and fifth rows) and predicted (second, fourth, and sixth rows) force distribution components (x in the first column, y in the second column, and z in the third column) for different samples and indenters in the real-world test dataset. Predictions were made with the raw-feature model. The first two rows show vertical indentation; the third and fourth rows show a shear-dominant indentation, and the last two rows a multi-contact indentation.



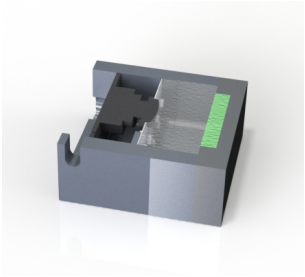
(a) Exploded view of the three-layer structure



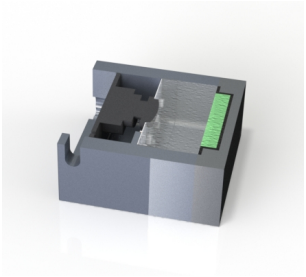
(b) Mold assembled before casting the first layer



(c) Cross-section after casting the first layer



(d) Cross-section after casting the second layer



(e) Cross-section after casting the third layer

Figure 9. The figure details the sensor's fabrication. The soft materials are arranged in a three-layer structure (see (a)) on top of the camera and the LEDs, and were poured into the mold from the side, through lateral cavities such as those shown in (b). Three different lids (see (c)-(e)) were employed for each of the soft layers.

1. A first layer of Elastosil® RT 601 RTV-2 (mixing ratio 7:1, shore hardness 45A) was poured into the mold, closed with a first lid (see Fig. 9(c)). The mold was then placed into an oven at 80 °C for 20 minutes for curing. This layer serves as a stiff base and facilitates light diffusion.

2. A release agent (Mann Ease Release™ 200) was sprayed before assembling the second lid, shown in Fig. 9(d). Then, a layer of Ecoflex™ GEL (mixing ratio 1:1, very soft, with shore hardness 000-35), mixed with green, fluorescent spherical particles (with a diameter of 150 to 180 μm), was poured into the mold. The mold was finally placed into an oven at 80 °C for 20 minutes for curing.

3. Finally, a layer of Elastosil® RT 601 RTV-2 (mixing ratio 25:1, shore hardness 10A), mixed with black silicone color (Elastosil® Color Paste FL), was poured into the mold, closed with a third lid, shown in Fig. 9(e). The mold was then placed into an oven at 80 °C for 45 minutes for curing. This layer is stiffer than the Ecoflex GEL, and shields the sensor from damage and light disturbances.

4. After removing the last lid, the sensor was placed back in the oven at 60 °C for 8 hours. This step has been shown to reduce stiffening caused by the aging of the materials [18].

The two soft upper layers amount to a rectangular prism of 32×32×6 mm.

B. The FEM simulation environment

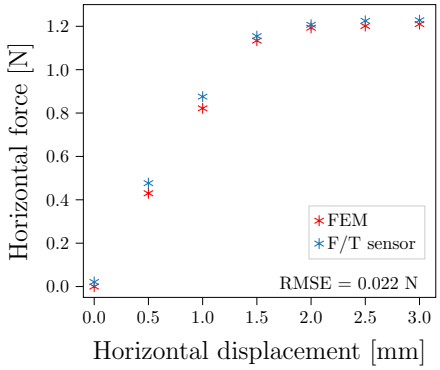
The FEM simulation environment was created in Abaqus/Standard [28] following previous work [18], where the Ecoflex GEL and the black Elastosil layer have both been characterized as hyperelastic materials using second-order Ogden models [26]. Given the large difference in hardness, the stiff base layer was considered rigid [18] in the FEM simulations discussed in the article. The contact between the top surface and the indenters was modeled as a hard contact and discretized with a surface-to-surface method. The basic Coulomb friction model available in Abaqus was employed, where the friction coefficient was assumed to be constant and was used as a tuning parameter, as described in the following.

The material characterization followed state-of-the-art techniques based on uniaxial tension, pure shear, and equibiaxial tension tests. The models obtained required no further calibration for the simulations described in this article. In fact, the characterization tests were entirely independent of the evaluation experiments carried out in this work. In order to verify the consistency of the Ogden models and the related FEM simulations, the previous work [18] also showed an accurate total force agreement when the same vertical indentation experiments were performed both in the FEM environment and the real world, where the total force was obtained from the readings of a commercial six-axis F/T sensor. Such verification experiments were augmented here to test the total force accuracy also in the case of shear-predominant or multi-contact indentations. The results of these experiments are shown in Fig. 10, where the total force resulting from FEM indentations was compared with the force measured by an F/T sensor (ATI Mini27 Titanium with an horizontal resolution of 0.03 N and a vertical resolution of 0.06 N) when repeating the same indentations in the real-world. The real-world experiments were carried out by mounting the F/T sensor and the appropriate indenters to the spindle of a controllable milling machine. The friction coefficient was tuned to a value of 0.9 using a single 3D-printed rough indenter in the experiment shown in (a). However, the remaining experiments showed generalization to different indentation shapes (see (b)) and a limited loss of accuracy for indenters of a different and smoother material, such as stainless steel (see (c)). In addition, the friction coefficient has a limited influence on the accuracy of the vertical component of the total force, as shown for the multi-contact experiment performed in (d) with stainless steel indenters.

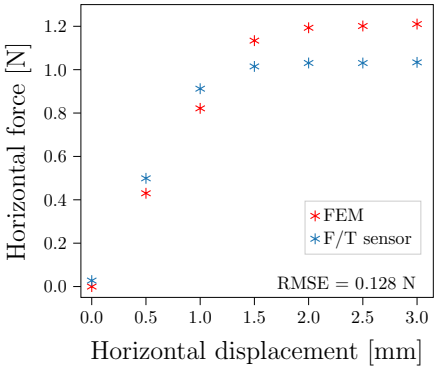
The indentation trajectories collected in the FEM environment, which were employed to generate training data, were randomized as follows:

1. First, for each trajectory, one of the 21 indenters modeled in simulation was selected randomly and translated to a random horizontal position over the sensing surface with a randomized orientation.

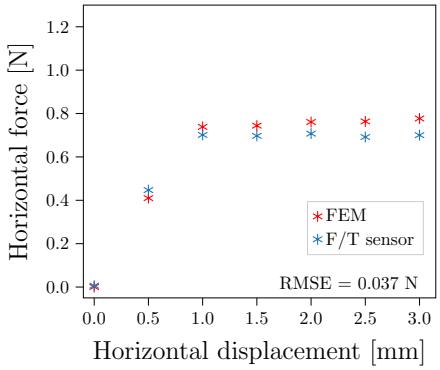
2. Then, 80% of the time, a vertical indentation followed by randomized horizontal translations was simulated. In the remaining 20% of the time, random 3D displacements were directly simulated in the vicinity of the initial position. Such a split



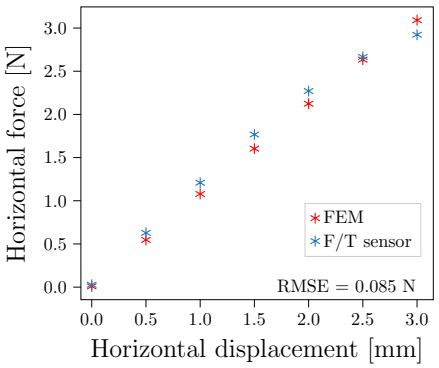
(a) Shear with spherical indenter (3DP)



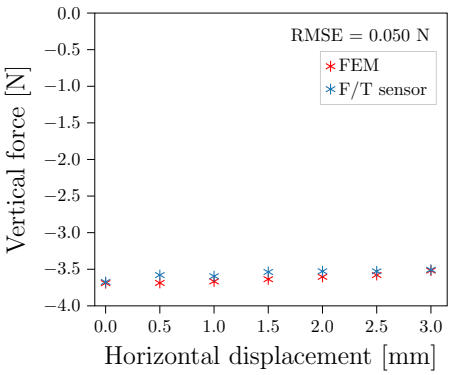
(b) Shear with spherical indenter (SS)



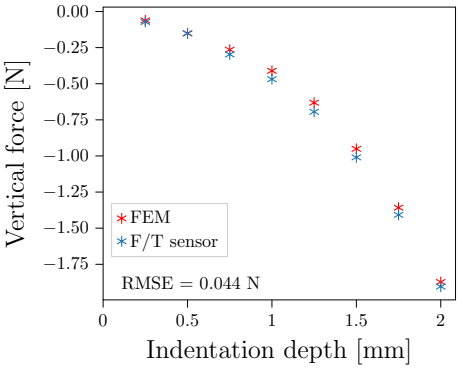
(c) Shear with square flat indenter (3DP)



(d) Shear with triangular flat indenter (3DP)



(e) Shear with triangular flat indenter (3DP)



(f) Pressure with two spherical indenters (SS)

Figure 10. The plots show the agreement (measured by the root-mean-square error, RMSE) between the total force resulting from FEM indentations (in red) and the readings of an F/T sensor (in blue) when repeating the same indentations in a controlled experimental scenario. While the FEM simulations are performed assuming a fixed friction coefficient, the real-world experiments employed indenters that were either 3D-printed (3DP) or made of stainless steel (SS). In (a), a 3DP spherically-ended indenter was first pressed 2 mm down in the center of the sensing surface. Then it was laterally sheared in one direction, and the horizontal force was recorded at discrete steps. In (b), the same experiment was repeated with an SS indenter with the same geometry. In (c), the shear experiment was carried out with a square flat-ended indenter at a depth of 1 mm, and in (d) with a triangular flat-ended indenter at a depth of 2 mm. The corresponding vertical force for the same experiment as in (d) is shown in (e), where a slight decrease in force was detected during shear from both the FEM simulation and the F/T sensor. In (f), two SS spherically-ended indenters (both attached to the same F/T sensor in the real world) were employed to make vertical indentations on the sensing surface, with the total vertical force recorded at discrete steps. Since the two indenters had a constant difference in height of 1.1 mm, the first four steps (up to 1 mm depth) resulted from contact with only one of the indenters, while the remaining steps (after 1 mm) resulted from a double indentation.

in the training trajectories aimed to favor the typical robotic manipulation case, where shear motion happens after a vertical grasp. Each indentation trajectory was split into 50 steps, with the maximum intra-step displacement constrained to 0.1 mm to facilitate convergence. The maximum depth reached by the indenters was 2 mm, while the maximum lateral displacement from the start of the indentation was 3 mm. Static steps were employed, therefore neglecting time-dependent material effects, which are however limited, as shown in Fig. 13.

C. Projection of a particle onto the image plane

The projection of the spherical particle centered at $s_{\mathrm{p}}^P := \left(x_{\mathrm{p}}^P, y_{\mathrm{p}}^P, z_{\mathrm{p}}^P\right)$ via the pinhole camera model results in an ellipse on the image plane [31], see Fig. 11(a). The pixel length r of the major axis of each ellipse can be computed via the projection formulas in the plane containing the camera's optical axis and the camera ray passing through the center of the spherical particle. An example projection in this plane is shown in Fig. 11(b). The coordinate $\tilde{x}_{\mathrm{p}}^P$ can be computed from the horizontal coordinates of the center of the particle as,

$$\tilde{x}_{\mathrm{p}}^P = \sqrt{\left(x_{\mathrm{p}}^P\right)^2 + \left(y_{\mathrm{p}}^P\right)^2}. \tag{4}$$

Then, from the figure, it follows that:

$$\alpha = \arctan\left(\frac{z_{\mathrm{p}}^P}{\tilde{x}_{\mathrm{p}}^P}\right), \quad \beta = \arcsin\left(\frac{R}{\sqrt{\left(\tilde{x}_{\mathrm{p}}^P\right)^2 + \left(z_{\mathrm{p}}^P\right)^2}}\right), \tag{5}$$

$$\gamma = \alpha - \beta, \tag{6}$$

where R is the radius of the sphere. The pixel length r of the major axis can then be computed as:

$$\tilde{x}_{\mathrm{r}}^P = \tilde{x}_{\mathrm{p}}^P + R\sin\gamma, \tag{7}$$

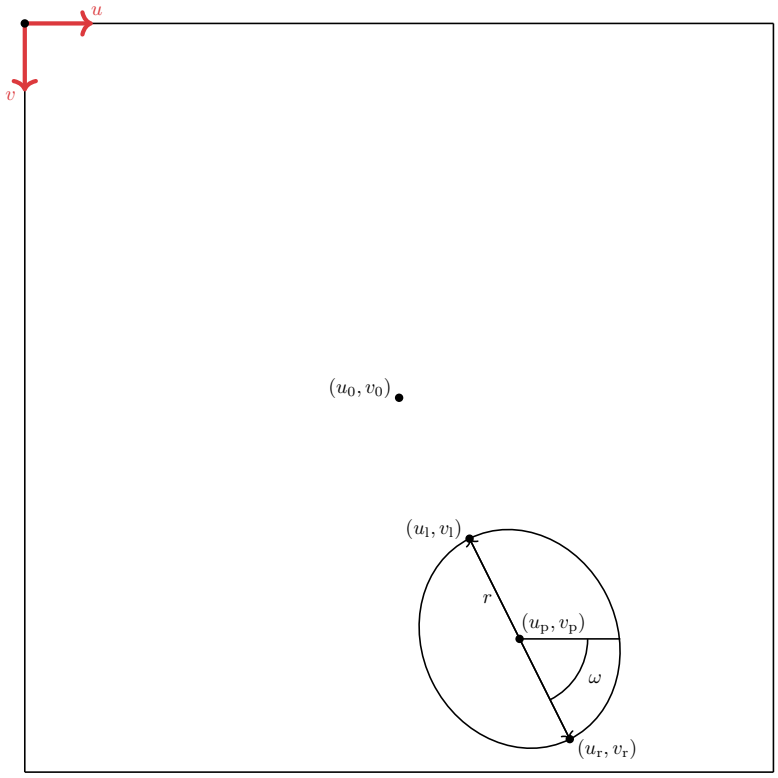
$$z_{\mathrm{r}}^P = z_{\mathrm{p}}^P - R\cos\gamma, \tag{8}$$

$$\tilde{x}_{\mathrm{l}}^P = \tilde{x}_{\mathrm{p}}^P - R\sin(\gamma + 2\beta), \tag{9}$$

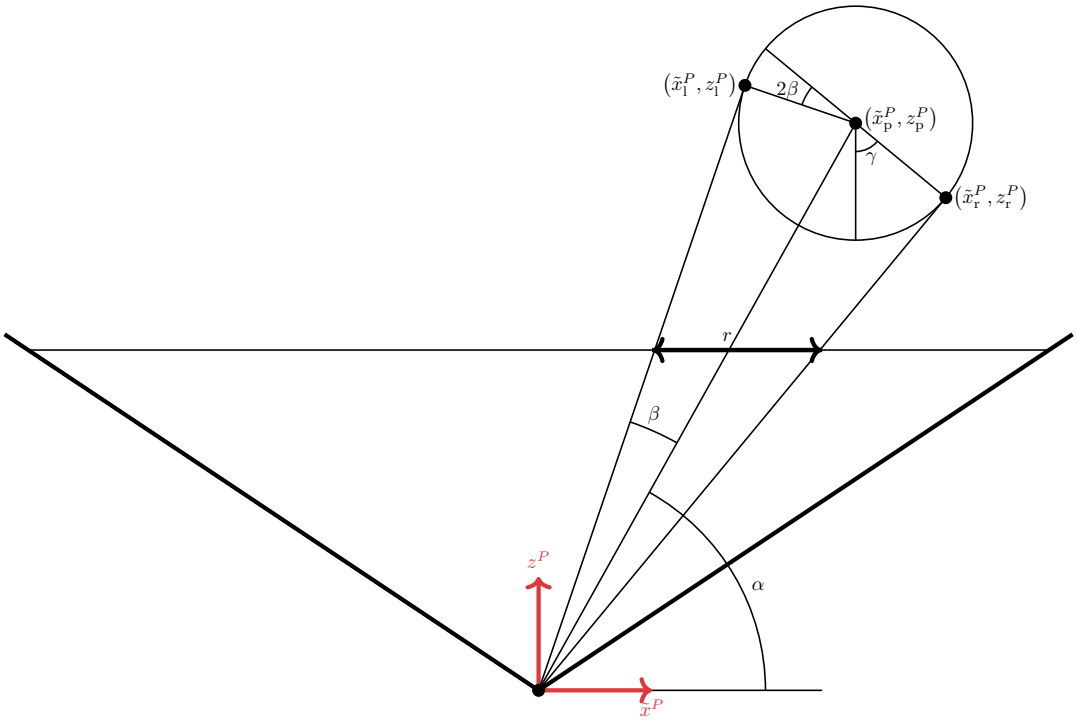
$$z_{\mathrm{l}}^P = z_{\mathrm{p}}^P + R\cos(\gamma + 2\beta), \tag{10}$$

$$r = \left| f\left(\frac{\tilde{x}_{\mathrm{r}}^P}{z_{\mathrm{r}}^P} - \frac{\tilde{x}_{\mathrm{l}}^P}{z_{\mathrm{l}}^P}\right)\right|. \tag{11}$$

The orientation of the ellipse on the image plane is fully determined by the horizontal



(a) Projection of a sphere onto the image plane



(b) 2D view of the projection of a sphere

Figure 11. The figures show that the projection of a sphere corresponds to an ellipse (see (a)) in the image plane. The length r of the major axis of this ellipse can be computed via 2D geometry in the plane that contains the optical axis and the ray passing through the camera and the center of the sphere.

position of the particle, and can therefore be computed trivially as:

$$\omega = \arctan2\left(y_{\mathrm{p}}^{P}, x_{\mathrm{p}}^{P}\right). \tag{12}$$

Additionally, the center of the ellipse can be computed by observing that $(u_{\mathrm{r}}, v_{\mathrm{r}})$ and $(u_{\mathrm{l}}, v_{\mathrm{l}})$ correspond to the projection of $(\tilde{x}_{\mathrm{r}}^{P}, z_{\mathrm{r}}^{P})$ and $(\tilde{x}_{\mathrm{l}}^{P}, z_{\mathrm{l}}^{P})$, respectively, onto the image plane:

$$x_{\mathrm{r}}^{P} = \tilde{x}_{\mathrm{r}}^{P}\cos\omega, \quad y_{\mathrm{r}}^{P} = \tilde{x}_{\mathrm{r}}^{P}\sin\omega, \tag{13}$$

$$u_{\mathrm{r}} = f\frac{x_{\mathrm{r}}^{P}}{z_{\mathrm{r}}^{P}} + u_0, \quad v_{\mathrm{r}} = f\frac{y_{\mathrm{r}}^{P}}{z_{\mathrm{r}}^{P}} + v_0, \tag{14}$$

$$x_{\mathrm{l}}^{P} = \tilde{x}_{\mathrm{l}}^{P}\cos\omega, \quad y_{\mathrm{l}}^{P} = \tilde{x}_{\mathrm{l}}^{P}\sin\omega, \tag{15}$$

$$u_{\mathrm{l}} = f\frac{x_{\mathrm{l}}^{P}}{z_{\mathrm{l}}^{P}} + u_0, \quad v_{\mathrm{l}} = f\frac{y_{\mathrm{l}}^{P}}{z_{\mathrm{l}}^{P}} + v_0, \tag{16}$$

where (u_0, v_0) are the coordinates of the pinhole image center.

Therefore, the pixel coordinates of the center of the ellipse are:

$$u_{\mathrm{p}} = \frac{u_{\mathrm{r}} + u_{\mathrm{l}}}{2}, \quad v_{\mathrm{p}} = \frac{v_{\mathrm{r}} + v_{\mathrm{l}}}{2}. \tag{17}$$

Finally, noting that the pixel length of the minor axis of the ellipses does not vary with the horizontal coordinates of the sphere [31], this length can be computed for a trivial case, that is, when the center of a particle lies on the optical axis (i.e., $x_{\mathrm{p}}^{P} = y_{\mathrm{p}}^{P} = 0$). The same formulas as in (4)-(11) can be employed, since for this special case the projection results in a circle, where both the major axis and the minor axis of the ellipse correspond to the diameter. Using the center, the axis lengths and the orientation of each ellipse, these can be drawn using the drawing functionality of OpenCV[5].

D. Remapping

As shown in Fig. 12, for a pixel $p := (u, v)$ in the image plane of the pinhole camera, a 3D point $s^P := (x^P, y^P, t_z^{GP})$ was retrieved using the pinhole projection equations as:

$$x^P = \frac{t_z^{GP}}{f}(u - u_0), \tag{18}$$

$$y^P = \frac{t_z^{GP}}{f}(v - v_0). \tag{19}$$

[5]https://opencv.org/

The 3D point was then converted to the coordinate system of the real-world camera, indicated with the superscript C, through the corresponding rotation and translation operations:

$$s^C = R^{GC}\left(R^{GP}\right)^{-1}\left(s^P - t^{GP}\right) + t^{GC}. \tag{20}$$

The corresponding pixel in the real-world image was then retrieved via the transformation function obtained from the calibration toolbox.

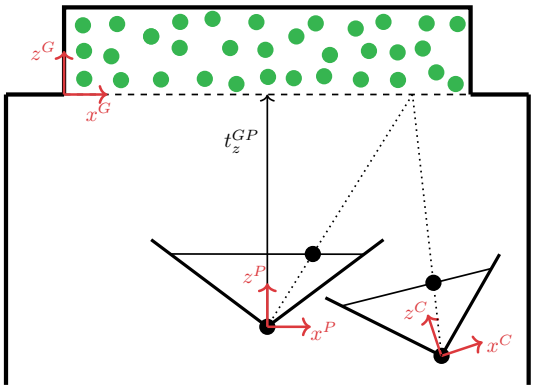


Figure 12. In the figure, a pixel in the pinhole camera is mapped to the corresponding pixel in the real-world camera.

E. Supplementary results

This section presents supplementary results and illustrations in addition to those in the main article. Fig. 14 compares optical flow examples obtained for the same indentation in simulation and reality. In Fig. 13, a programmable milling machine was employed to make indentations using two of the test indenters, and the total force recorded with an F/T sensor was compared with the real-time prediction of the neural network presented in the main article.

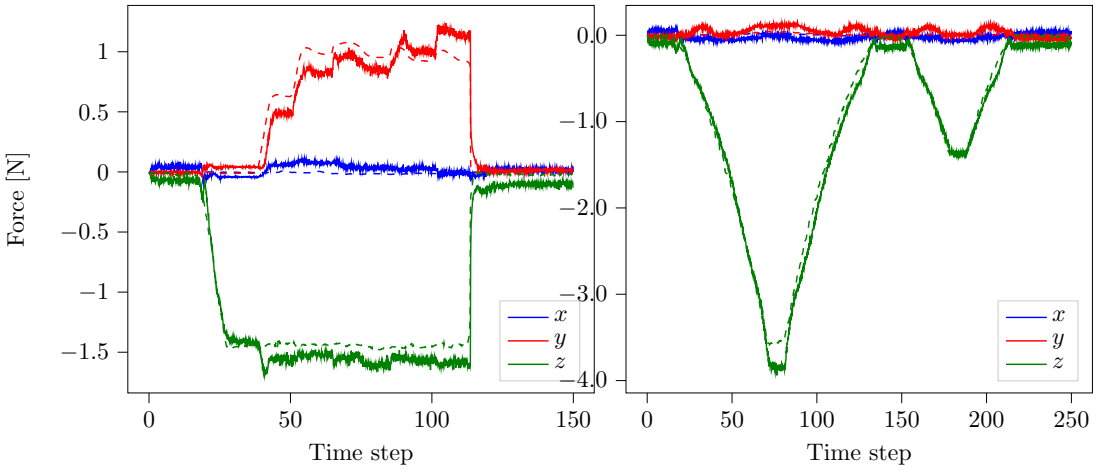
Table 3 and Table 4 show in detail the different error metrics listed by data subgroups for the real-world test dataset, depending on the type of indentation or the indenter employed. Both tables are based on metrics computed using the raw-feature model described in the main article. It can be noted that shear-dominant indentations may present a decrease in accuracy in the prediction of the z component of the force. This is partly due to the fact that shear-dominant data generally showed higher noise, as a small misalignment in the indenter mounting may lead to considerably different behavior of the material during the shearing trajectory. In addition, as shown in the main article, the model tended to generalize to multi-contact indentations. However, the performance for such contact conditions may be further improved by including a portion of multi-contact data in the

		Vertical	Shear-dominant	Multi-contact
RMSE	x	0.003	0.010	0.004
	y	0.005	0.011	0.004
	z	0.013	0.018	0.013
RMSET	x	0.034	0.548	0.088
	y	0.057	0.606	0.061
	z	0.277	0.465	0.488
MAE (bin)	x	0.001	0.002	0.001
	y	0.001	0.002	0.001
	z	0.002	0.004	0.002
MAE (total)	x	0.022	0.333	0.060
	y	0.035	0.342	0.046
	z	0.168	0.368	0.345
SDAE (bin)	x	0.003	0.010	0.004
	y	0.004	0.011	0.004
	z	0.012	0.017	0.012
SDAE (total)	x	0.025	0.436	0.065
	y	0.045	0.500	0.040
	z	0.221	0.284	0.345
Range (total)	x	-0.2–0.2	-3.2–3.2	-0.3–0.3
	y	-0.6–0.6	-3.8–3.8	-0.3–0.3
	z	-4.5–0	-3.8–0	-3.5–0

Table 3. The table reports in detail the different error metrics (using a raw-feature model) for each of the three types of real-world indentations, that is, vertical, shear-dominant, and multi-contact indentations. The abbreviations are defined as in the main article. The Newton unit was omitted here for all the values.

training dataset. Among the six indenters employed, the results generally showed a correlation between the range of forces and the errors recorded. In addition, it turned out to be very challenging to accurately align the tilted-plane indenter (which is the indenter that shows a deeper side in Fig. 2 of the main article) with the reference system of the gel for data collection. For this reason, shear data were not collected with such an indenter.

Furthermore, Fig. 15 shows that a diverse dataset is crucial for generalization. The samples in the figure correspond to the first two in Fig. 8 of the main article, but the predictions were made with the network trained in previous work [3]. This network was only trained with vertical indentations made with a spherically-ended indenter. To evaluate the generalization, the first two rows show a vertical indentation made with a cylindrical indenter, while the third and fourth rows show a shear-dominant indentation made with a spherically-ended indenter. While the network in previous work [3] showed sensible predictions for some indenters different from the one used for training, the figure shows how, in contrast, the network does not generalize well to light pressure conditions with the cylindrical indenter. In particular, the network predicted the typical force profile for a spherically-ended indenter. In addition, the shear-dominant indentation was also mispredicted, with the y component of the force distribution predicted as symmetrical, which



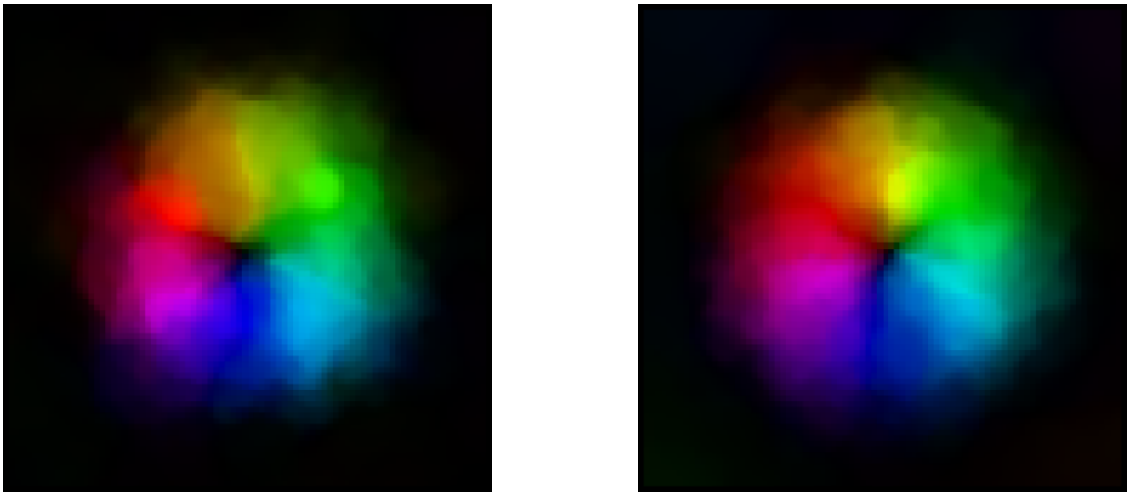
(a) Shear with cylindrical indenter **(b)** Pressure with triangular flat indenter

Figure 13. The plots compare the total force computed from the predictions of the neural network (solid lines) against the readings of an F/T sensor (dashed lines) for each of the three force components. In (a), a thin cylindrical indenter was first pressed against the sensing surface up to a depth of 1 mm, then it was laterally displaced in the y direction in discrete steps up to 3 mm, and finally lifted. In (b), two pressure cycles were executed with a triangular flat indenter, first up to 2 mm, then up to 1 mm. Although the network was only trained on static data, the predictions accurately capture the force trends in both the figures. The main inaccuracies can be observed for considerably larger deformations (where the material characterization in previous work [18] showed larger variance) with the triangular flat indenter, or during the unloading phase (where the material showed mild relaxation effects).

		Spherical large	Triangular	Square	Cylindrical	Spherical small	Tilted-plane
RMSE	x	0.003	0.009	0.007	0.007	0.004	0.002
	y	0.003	0.008	0.007	0.014	0.003	0.002
	z	0.008	0.022	0.015	0.021	0.008	0.014
RMSET	x	0.194	0.411	0.306	0.494	0.082	0.034
	y	0.129	0.449	0.293	0.644	0.027	0.047
	z	0.154	0.478	0.231	0.498	0.082	0.664
MAE (bin)	x	0.001	0.002	0.001	0.001	0.001	0.001
	y	0.001	0.002	0.001	0.003	0.001	0.001
	z	0.001	0.005	0.003	0.005	0.001	0.003
SDAE (bin)	x	0.004	0.009	0.007	0.007	0.003	0.002
	y	0.003	0.008	0.007	0.014	0.003	0.002
	z	0.007	0.021	0.014	0.020	0.008	0.013
MAE (total)	x	0.080	0.197	0.128	0.219	0.030	0.023
	y	0.057	0.208	0.142	0.336	0.021	0.037
	z	0.111	0.379	0.185	0.395	0.058	0.596
SDAE (total)	x	0.177	0.361	0.278	0.443	0.076	0.025
	y	0.116	0.398	0.257	0.550	0.017	0.030
	z	0.106	0.291	0.138	0.302	0.059	0.293
Range (total)	x	-1.2–1.2	-3.2–3.2	-2.2–2.2	-3.2–3.2	-0.1–0.1	-0.01–0.01
	y	-1.2–1.2	-3.2–3.2	-2.2–2.2	-3.8–3.8	-0.1–0.1	-0.01–0.01
	z	-1.7–0	-4.3–0	-2.0–0	-4.6–0	-1.0–0	-1.3–0

Table 4. The table reports in detail the different error metrics (using a raw-feature model) for each of the six real-world indenters employed to collect the test dataset (and shown in the main article). The Newton unit was omitted here for all the values.

is the typical force profile in a vertical indentation.



(a) Simulation **(b)** Reality

Figure 14. Comparison of the optical flow obtained in simulation (a) versus reality (b). The color represents the direction, while darker regions represent smaller displacements.

Acknowledgments

The authors would like to thank Michael Egli and Matthias Mueller for their support in the sensor manufacture, and Thomas Bi for the discussions on the sensing pipeline.

References

[1] C. Sferrazza and R. D'Andrea, "Design, Motivation and Evaluation of a Full-Resolution Optical Tactile Sensor", *Sensors*, vol. 19, no. 4:928, 2019.

[2] D. V. Hutton, *Fundamentals of Finite Element Analysis*. McGraw-Hill, 2004.

[3] C. Sferrazza, T. Bi, and R. D'Andrea, "Learning the sense of touch in simulation: a sim-to-real strategy for vision-based tactile sensing", in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2020.

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015, pp. 234–241.

[5] R. S. Dahiya, G. Metta, M. Valle, and G. Sandini, "Tactile Sensing—From Humans to Humanoids", *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 1–20, 2010.

[6] K. Shimonomura, "Tactile Image Sensors Employing Camera: A Review", *Sensors*, vol. 19, no. 18:3933, 2019.

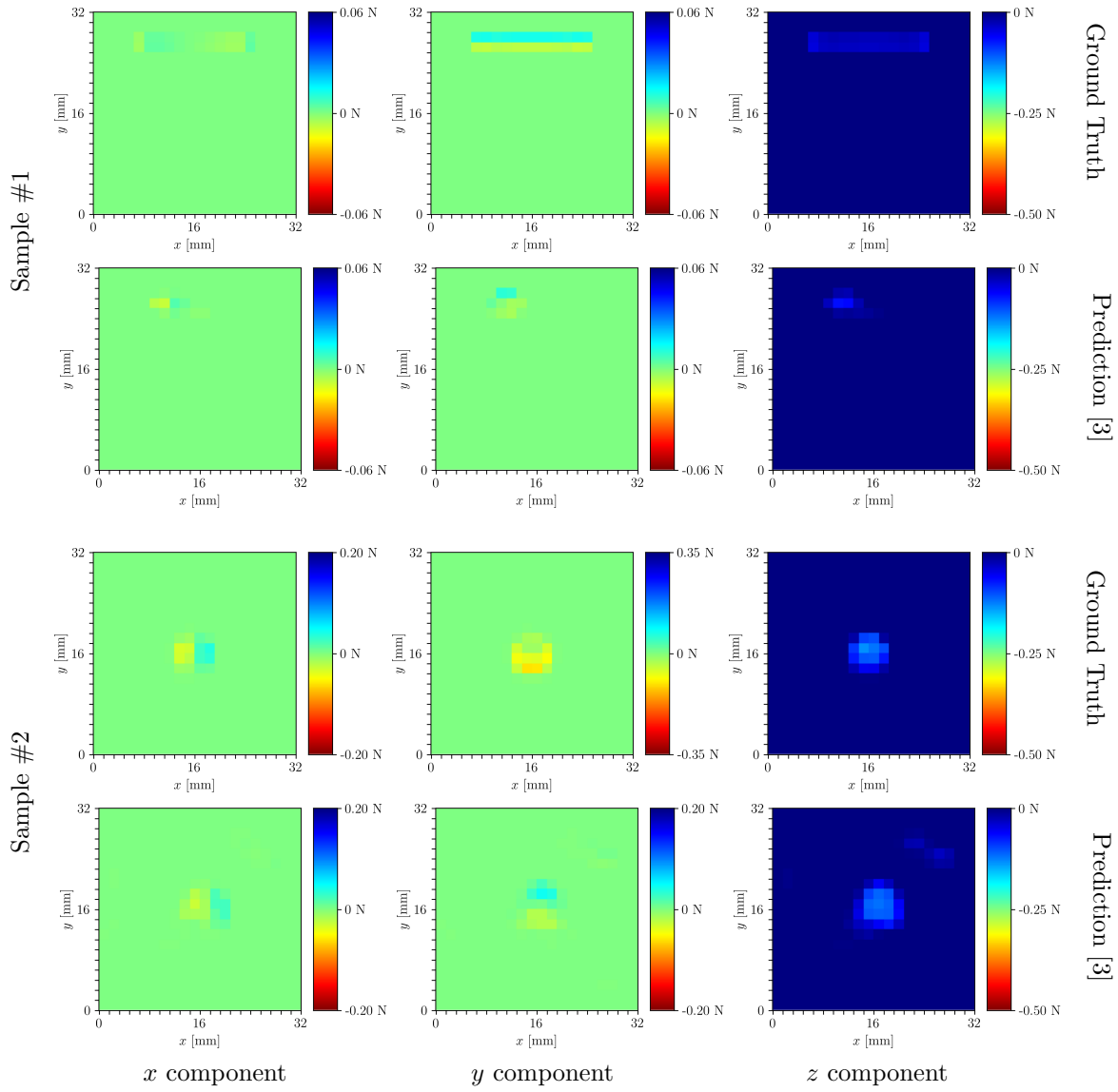


Figure 15. The figures show the ground truth (first and third rows) and predicted (second and fourth rows) force distribution components (x in the first column, y in the second column, and z in the third column) for the first two samples shown in Fig. 8 in the main article, collected with two different indenters in the real world. Predictions were made with the model trained in previous work [3], where only vertical indentations made with a spherically-ended indenter were contained in the training dataset. The first two rows show a vertical indentation with a cylindrical indenter; the third and fourth rows show a shear-dominant indentation with a spherically-ended indenter. Note how the model trained in previous work [3] does not generalize well to such cases.

[7] B. Ward-Cherrier, N. Pestell, L. Cramphorn, B. Winstone, M. E. Giannaccini, J. Rossiter, and N. F. Lepora, "The TacTip Family: Soft Optical Tactile Sensors with 3D-Printed Biomimetic Morphologies", *Soft robotics*, vol. 5, no. 2, pp. 216–227, 2018.

[8] F. Baghaei Naeini, D. Makris, D. Gan, and Y. Zweiri, "Dynamic-Vision-Based Force Measurements Using Convolutional Recurrent Neural Networks", *Sensors*, vol. 20, no. 16:4469, 2020.

[9] A. C. Abad and A. Ranasinghe, "Visuotactile Sensors with Emphasis on GelSight Sensor: A Review", *IEEE Sensors Journal*, vol. 20, no. 14, pp. 7628–7638, 2020.

[10] C. Trueeb, C. Sferrazza, and R. D'Andrea, "Towards vision-based robotic skins: a data-driven, multi-camera tactile sensor", in *Proceedings of the IEEE International Conference on Soft Robotics*, 2020, pp. 333–338.

[11] M. Lambeta, P.-W. Chou, S. Tian, B. Yang, B. Maloon, V. R. Most, D. Stroud, R. Santos, A. Byagowi, G. Kammerer, D. Jayaraman, and R. Calandra, "DIGIT: A Novel Design for a Low-Cost Compact High-Resolution Tactile Sensor With Application to In-Hand Manipulation", *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 3838–3845, 2020.

[12] A. Padmanabha, F. Ebert, S. Tian, R. Calandra, C. Finn, and S. Levine, "OmniTact: A Multi-Directional High Resolution Touch Sensor", in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2020, pp. 618–624.

[13] B. Romero, F. Veiga, and E. Adelson, "Soft, Round, High Resolution Tactile Fingertip Sensors for Dexterous Robotic Manipulation", in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2020, pp. 4796–4802.

[14] D. F. Gomes, Z. Lin, and S. Luo, "GelTip: A Finger-shaped Optical Tactile Sensor for Robotic Manipulation", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 9903–9909.

[15] E. Donlon, S. Dong, M. Liu, J. Li, E. Adelson, and A. Rodriguez, "GelSlim: A High-Resolution, Compact, Robust, and Calibrated Tactile-sensing Finger", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 1927–1934.

[16] D. Ma, E. Donlon, S. Dong, and A. Rodriguez, "Dense Tactile Force Estimation using GelSlim and inverse FEM", in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2019, pp. 5418–5424.

[17] N. Kuppuswamy, A. Castro, C. Phillips-Grafflin, A. Alspach, and R. Tedrake, "Fast Model-Based Contact Patch and Pose Estimation for Highly Deformable Dense-Geometry Tactile Sensors", *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1811–1818, 2019.

[18] C. Sferrazza, A. Wahlsten, C. Trueeb, and R. D'Andrea, "Ground Truth Force Distribution for Learning-Based Tactile Sensing: A Finite Element Approach", *IEEE Access*, vol. 7, pp. 173 438–173 449, 2019.

[19] W. Yuan, S. Dong, and E. H. Adelson, "GelSight: High-Resolution Robot Tactile Sensors for Estimating Geometry and Force", *Sensors*, vol. 17, no. 12: 2762, 2017.

[20] P. Piacenza, K. Behrman, B. Schifferer, I. Kymissis, and M. Ciocarlie, "A Sensorized Multicurved Robot Finger With Data-Driven Touch Sensing via Overlapping Light Signals", *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 5, pp. 2416–2427, 2020.

[21] Y. S. Narang, K. Van Wyk, A. Mousavian, and D. Fox, "Interpreting and Predicting Tactile Signals via a Physics-Based and Data-Driven Framework", in *Proceedings of Robotics: Science and Systems*, 2020.

[22] H. Lee, H. Park, G. Serhat, H. Sun, and K. J. Kuchenbecker, "Calibrating a Soft ERT-Based Tactile Sensor with a Multiphysics Model and Sim-to-real Transfer Learning", in *2020 IEEE International Conference on Robotics and Automation*, 2020, pp. 1632–1638.

[23] Z. Ding, N. F. Lepora, and E. Johns, "Sim-to-Real Transfer for Optical Tactile Sensing", in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2020, pp. 1639–1645.

[24] D. F. Gomes, A. Wilson, and S. Luo, "GelSight Simulation for Sim2Real Learning", in *ICRA ViTac Workshop*, 2019.

[25] Y. Wang, W. Huang, B. Fang, and F. Sun, "Elastic Interaction of Particles for Robotic Tactile Simulation", *arXiv preprint arXiv:2011.11528*, 2020.

[26] R. W. Ogden, "Large deformation isotropic elasticity – on the correlation of theory and experiment for incompressible rubberlike solids", in *Proceedings of the Royal Society of London. A. Mathematical, Physical and Engineering Sciences*, vol. 326, 1972, pp. 565–584.

[27] M. Ciavarella, D. Hills, and G. Monno, "The influence of rounded edges on indentation by a flat punch", *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 212, no. 4, pp. 319–327, 1998.

[28] Dassault Systèmes, *Abaqus/Standard User's Manual, Version 2019*, English, Simulia, 2019.

[29] L. Mitas and H. Mitasova, "Spatial interpolation", in *Geographical Information Systems: Principles, Techniques, Management and Applications*, P. A. Longley, M. F. Goodchild, D. J. Maguire, and D. W. Rhind, Eds., John Wiley & Sons, 1999.

[30] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2010.

[31] D. S. Wokes and P. L. Palmer, "Perspective Projection Of A Spheroid Onto An Image Plane", *SIAM Journal on Imaging Sciences*, 2008.

[32] T. Kroeger, R. Timofte, D. Dai, and L. Van Gool, "Fast Optical Flow using Dense Inverse Search", in *Proceedings of the European Conference on Computer Vision*, 2016, pp. 471–488.

[33] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A Toolbox for Easily Calibrating Omnidirectional Cameras", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 5695–5701.

[34] S. Kaji and S. Kida, "Overview of image-to-image translation by use of deep neural networks: denoising, super-resolution, modality conversion, and reconstruction in medical imaging", *Radiological Physics and Technology*, vol. 12, no. 3, pp. 235–248, 2019.

[35] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization", in *Proceedings of the International Conference on Learning Representations*, 2019.

[36] T. Bi, C. Sferrazza, and R. D'Andrea, "Zero-shot sim-to-real transfer of tactile control policies for aggressive swing-up manipulation", *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5761–5768, 2021.

Part B

Computationally efficient learning-based model predictive control

consisting of publications

[P6] C. Sferrazza, M. Muehlebach, and R. D'Andrea, "Trajectory tracking and iterative learning on an unmanned aerial vehicle using parametrized model predictive control", in *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 2017, pp. 5186–5192

[P7] C. Sferrazza, M. Muehlebach, and R. D'Andrea, "Learning-based parametrized model predictive control for trajectory tracking", *Optimal Control Applications and Methods*, vol. 41, no. 6, pp. 2225–2249, 2020

Paper P6

Trajectory tracking and iterative learning on an unmanned aerial vehicle using parametrized model predictive control

Carmelo Sferrazza, Michael Muehlebach and Raffaello D'Andrea

Abstract

A parametrization of state and input trajectories is used to approximate an infinite-horizon optimal control problem encountered in model predictive control. The resulting algorithm is discussed with respect to trajectory tracking, including the problem of generating feasible trajectories. In order to account for unmodeled repeatable disturbances an iterative learning scheme is applied, and as a result, the tracking performance can be improved over consecutive trials. The algorithm is applied to an unmanned aerial vehicle and shown to be computationally efficient, running onboard at a sampling rate of 100 Hz during the experiments.

Published in *Proceedings of the 2017 IEEE Conference on Decision and Control.*

1. Introduction

Model Predictive Control (MPC) is an established control strategy for addressing challenging control problems, often including input and state constraints, see for example [1] and references therein. It is based on the following procedure: at each time step, an optimal control problem is solved and the first portion of the resulting input trajectory is applied to the system. This yields an implicit feedback law providing robustness against disturbances and modeling errors.

Due to the fact that an optimal control problem has to be solved at each time step, MPC is computationally demanding, particularly when applied to systems with fast dynamics. A common approach to reduce the computational complexity is to discretize the dynamics and truncate the prediction horizon. This leads naturally to a trade-off between computation and prediction horizon. However, the truncation of the prediction horizon might require the introduction of a terminal cost and terminal state constraints to preserve stability, see [1], [2].

In contrast, the MPC approach presented in [3] retains an infinite prediction horizon by parametrizing input and state trajectories with decaying basis functions. Provided that the resulting trajectories fulfill the constraints for all times, this leads to inherent closed-loop stability and recursive feasibility guarantees. Moreover, by choosing suitable basis functions, the resulting MPC formulation tends to have fewer optimization variables and therefore exhibits small execution times, as reported in [4].

This article extends the approach presented in [3] to trajectory tracking and discusses the application to an unmanned aerial vehicle. In addition, an iterative learning scheme is incorporated in order to reject repeatable disturbances.

1.1 Related Work

Due to the increase in computational power and the availability of dedicated optimization routines, see for example [5], [6], MPC has been applied to a wide range of dynamic systems. The vast majority of MPC controllers found in the literature are based on a discrete-time finite-horizon formulation, an overview of which is given in [1]. In [7] and [8, Ch. 3, Ch. 6], an alternative formulation is proposed, where the finite differences of the control inputs (in discrete time) or the time derivative of the inputs (in continuous time) are parametrized with so-called Laguerre or Kauz basis functions. Although similar basis functions are used herein, their approach is different due to the fact that a finite prediction horizon is retained, and that the control inputs are not parametrized directly. Moreover, the state variable is eliminated, whereas we encode the dynamics as an equality constraint that may or may not be eliminated, potentially leading to a sparser optimization problem.

In the following, we point out some applications of MPC to real-world systems, which we find relevant for our work. This includes the control of unmanned aerial vehicles with MPC, and trajectory tracking or iterative learning in combination with MPC.

In [9], the application of MPC to a thrust-vectored flight control experiment with a ducted fan actuation is presented. Input and state constraints are included and the region

of attraction of the MPC controller is shown to be larger than that of the corresponding linear quadratic regulator.

Different strategies have been presented to tackle the problem of trajectory tracking with MPC. In [10], a successive linearization approach has been applied to the discrete-time infinite-horizon MPC strategy. This method has been shown to be more efficient than a nonlinear MPC formulation, while maintaining a comparable performance, when applied to a mobile robot. Guarantees for recursive feasibility have been presented in [11] for constrained trajectory tracking problems through the use of time-varying terminal regions. In [12], the tracking control problem of underactuated vehicles is addressed by allowing an asymptotic tracking error. This provides a means to compute the terminal set and the terminal control law that guarantee asymptotic convergence of the position of the vehicle to a tube centered around the desired path. A direct multiple-shooting approach is introduced in [13] for solving optimal control problems encountered in MPC. The approach is evaluated in simulation, by performing extreme maneuvers with a flying vehicle. In [14], a MPC framework is implemented and used for trajectory tracking of a quadrotor, where the dynamics are modeled as a set of piecewise affine systems given by different operating points. An application of MPC to the trajectory tracking of a formation of flying vehicles is presented in [15].

Unlike most of the approaches summarized above, we do not discretize the dynamics, but parametrize input and state trajectories with exponentially decaying basis functions. As a result, we retain an infinite prediction horizon, and obtain an optimization problem with relatively few variables. The numerical effectiveness of our approach will be demonstrated by the fact that the resulting MPC controller achieves a sampling time of 100 Hz on an embedded computer (Gumstix DuoVero COM).

Trajectory tracking can often be improved by incorporating learning approaches. Iterative Learning Control (ILC), see [16], provides a way to iteratively improve the system model available to the controller over multiple trials. An overview of ILC can be found in [17]. In [18], a time-varying Kalman filter is used to estimate the repeatable disturbances along a trajectory, which might stem from unmodeled system dynamics and uncertainties in the physical parameters. We will use a similar approach to identify the repeatable disturbances, and incorporate them into our MPC formulation.

In [19] and [20], learning approaches are included in an MPC framework. In [21], deep learning is combined with MPC for guided policy search, where MPC is used to generate data at training time. The method is evaluated with simulations of a quadrotor's flight. In [22], an underlying control sequence is included as a (deficient) reference to be improved for the predictive tracking control. At each iteration, the input sequence is corrected by performing a learning update. A similar strategy has been proposed in [23], where the MPC performance is improved by feeding back the control errors from previous iterations, based on the concept of repetitive control.

Applications of ILC in combination with MPC have been shown on a pH plant, see [24], and in [25], where learning-based MPC has been applied on a quadrotor that has been trained to catch a ball.

In our approach, we augment our system's model with repeatable disturbances, which we parametrize using the same basis functions as for representing the input and the states. As a consequence, these disturbances can be naturally incorporated in our MPC formulation. In that way, the system's model is updated over consecutive trials, improving the accuracy of the predictions used for MPC.

1.2 Outline

The parametrized MPC problem is presented in Section 2. Input and state trajectories are approximated through a linear combination of Laguerre functions, and a constraint sampling strategy is proposed. Section 3 discusses the problem of generating a trajectory that is parametrized by the given basis functions. The online trajectory tracking is explained in Section 4. In Section 5, an iterative learning scheme is incorporated. Simulation and experimental results obtained from flights with an unmanned aerial vehicle are shown in Section 6. Concluding remarks are made in Section 7.

2. Parametrized MPC

In order to approximate the infinite-horizon optimal control problem that will be used in our MPC approach, we will represent state and input trajectories as linear combinations of Laguerre functions, that is,

$$\tilde{x}(t) := (I_n \otimes \tau(t))^{\mathsf{T}} \eta_x, \quad \tilde{u}(t) := (I_m \otimes \tau(t))^{\mathsf{T}} \eta_u, \tag{1}$$

with $\tau(t) := (\tau_1(t), \tau_2(t), \ldots, \tau_s(t))$, for all $t \in [0, \infty)$, and

$$\tau_i(t) := \sqrt{2\lambda} \exp(-\lambda t) \sum_{k=0}^{i-1} \binom{i-1}{k} \frac{(-1)^k}{k!} (2\lambda t)^k, \tag{2}$$

where $\eta_x \in \mathbb{R}^{ns}$, $\eta_u \in \mathbb{R}^{ms}$ are the parameter vectors, λ is the exponential decay, n and m describe the state respectively the input dimension, and $\otimes$ denotes the Kronecker product. This approach has been previously presented in [3], where additional motivation and examples are included. For ease of notation, vectors are expressed as n-tuples, with dimension and stacking clear from the context, i.e. $\tau(t) = (\tau_1(t), \tau_2(t), \ldots, \tau_s(t)) \in \mathbb{R}^s$.

It can be shown that the basis functions $\tau(t)$ satisfy the following properties,

$$\dot{\tau}(t) = M_\lambda \tau(t), \quad \tau(t) = e^{M_\lambda t} \tau(0), \quad \forall t \in [0, \infty), \tag{3}$$

where

$$M_\lambda = \begin{bmatrix} -\lambda & 0 & \cdots & 0 \\ -2\lambda & -\lambda & & \vdots \\ \vdots & & \ddots & 0 \\ -2\lambda & \cdots & -2\lambda & -\lambda \end{bmatrix} \in \mathbb{R}^{s\times s}, \tag{4}$$

which will be used in a later stage.

As discussed in [3], by representing input and state trajectories with $\tilde{x}$ and $\tilde{u}$ according to (1), the constrained infinite-horizon linear-quadratic optimal regulator problem can be approximated as

$$\inf_{\eta_x,\eta_u} \frac{1}{2}\left\{\eta_x^\mathsf{T}\left(Q\otimes I_s\right)\eta_x + \eta_u^\mathsf{T}\left(R\otimes I_s\right)\eta_u\right\} \tag{5}$$

$$\text{s.t.} \quad A_x\eta_x + B_u\eta_u = 0, \tag{6}$$

$$(I_n\otimes\tau(0))^\mathsf{T}\eta_x = x_0, \tag{7}$$

$$F\eta_u \in \mathcal{U} := [u_{\min}, u_{\max}]^s, \tag{8}$$

where $u_{\min}\in\mathbb{R}^m$ and $u_{\max}\in\mathbb{R}^m$ are the lower and upper bounds on the control input $u(t)$, $Q\in\mathbb{R}^{n\times n}$ is positive definite ($Q\succ 0$), $R\in\mathbb{R}^{m\times m}$ is positive definite ($R\succ 0$),

$$F := \begin{bmatrix} \mathrm{I}_m\otimes\tau(t_1)^\mathsf{T} \\ \mathrm{I}_m\otimes\tau(t_2)^\mathsf{T} \\ \vdots \\ \mathrm{I}_m\otimes\tau(t_s)^\mathsf{T} \end{bmatrix} \in \mathbb{R}^{ms\times ms}, \tag{9}$$

and

$$A_x := A\otimes I_s - I_n\otimes M_\lambda^\mathsf{T}, \quad B_u := B\otimes I_s, \tag{10}$$

where $A\in\mathbb{R}^{n\times n}$ is the system matrix and $B\in\mathbb{R}^{n\times m}$ is the matrix through which the inputs enter the system. The suboptimality of this approximation can be quantified, as discussed in [26].

The cost function in (5) matches the quadratic cost

$$\int_0^\infty \frac{1}{2}\left\{\tilde{x}(t)^\mathsf{T}Q\tilde{x}(t) + \tilde{u}(t)^\mathsf{T}R\tilde{u}(t)\right\}\mathrm{d}t. \tag{11}$$

In addition, as shown in [3], the constraints (6) and (7) imply that the system's

dynamics are fulfilled exactly, that is,

$$\dot{\tilde{x}}(t) = A\tilde{x}(t) + B\tilde{u}(t), \quad \tilde{x}(0) = x_0, \quad \forall t \in [0, \infty). \tag{12}$$

The constraint (8) represents a box constraint on the inputs, where the input constraints are relaxed, and are only enforced at the specific time instants t_i, $i = 1, \ldots, s$. The choice of these time instants is discussed in [4]. The above formulation can be generalized to the case of arbitrary linear constraints on the states and the inputs. However, restricting the input (and possibly state) constraints to be box constraints enables an efficient implementation of the optimization routine for solving the optimization problem (5), as highlighted in [4].

If the resulting optimal input trajectory violates the constraint $u_{\min} \leq u(t) \leq u_{\max}$ for all times $t \in [0, \infty)$, for instance in between the time instants t_i, $i = 1, \ldots, s$, the theoretic stability and recursive feasibility guarantees are no longer valid in general. In practice however, this sampling strategy often results in constraint satisfaction for all times, and enables to solve (5) efficiently with the Generalized Fast Dual Gradient (GFDG) method, see [4].

3. Trajectory generation

We propose to generate feasible trajectories for all states and inputs of the given system by solving an optimization problem similar to (5). This process includes a transformation of the desired trajectories from the physical space, which we denote by $\bar{x}_{\text{des}}(t)$, to the parameter space.

In particular, we propose solving an optimization problem for finding feasible trajectories (compatible with the dynamics and fulfilling the constraints) that are the closest possible fit to $\bar{x}_{\text{des}}(t)$ (in the weighted L^2-sense), while keeping the control effort as small as possible.

The Laguerre functions that are used in our parametrized MPC approach have limited polynomial order, and as such they would not be suitable to perform a single fit over the entire trajectory horizon. In order to tackle this problem, we split the entire trajectory into N smaller intervals of length T. In order to not lose the predictive power of the MPC approach, we solve the trajectory generation problem over a prediction window containing a number N_{pred} of these intervals, covering a time horizon of length $N_{\text{pred}}T$. As a result, this approach improves the accuracy of the fit. In practice, a trade-off between the prediction horizon $N_{\text{pred}}T$ and the accuracy of the fit has to be found.

Moreover, the basis functions are decaying to zero and as such, they are unable to capture steady-state deviations. Provided that these steady-state offsets correspond to equilibrium points of the dynamics, they are included in the trajectory generation problem. More precisely, for each interval j we may introduce the offsets $x_{\text{b},j}$, that are chosen

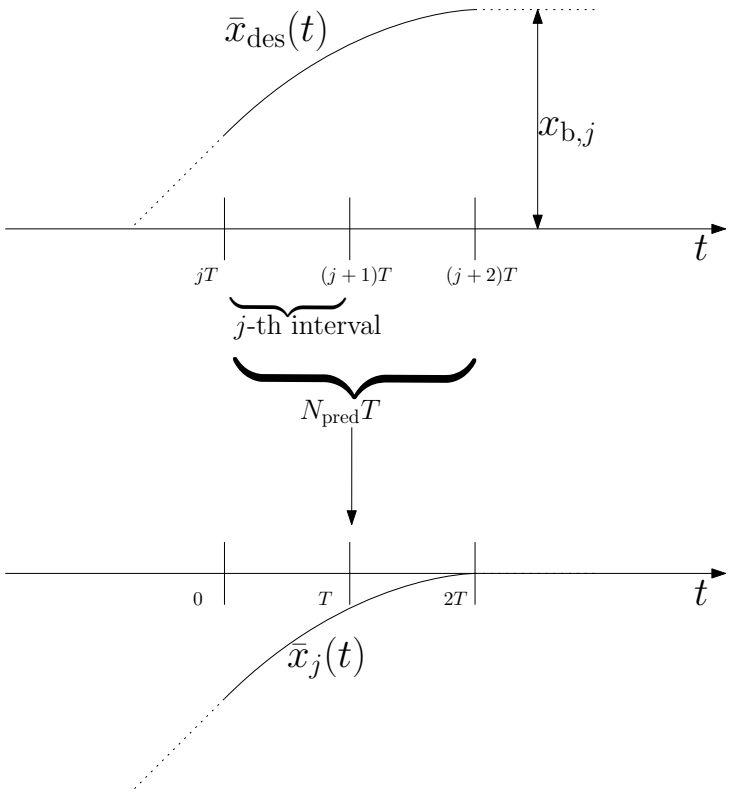


Figure 1. Offset and prediction window for a single state, with $N_{\mathrm{pred}} = 2$. The curve is shifted by the offset $x_{\mathrm{b},j}$, such that at the end of the prediction window the state will be at zero. The shifted curve is used for solving the trajectory generation problem in (20) and the resulting parameters $\eta_{\mathrm{ref},j}$ are stored for the j-th interval. The procedure is repeated for all the intervals $j = 0, \ldots, N-1$.

such that $\bar{x}_{\mathrm{des}}((j+N_{\mathrm{pred}})T) - x_{\mathrm{b},j} = 0$, and shift the desired trajectory by these offsets, leading to the shifted trajectories $\bar{x}_j(t) := \bar{x}_{\mathrm{des}}(t+jT) - x_{\mathrm{b},j}$. These are defined over the intervals $j = 0, \ldots, N-1$, which are used as a starting point for the trajectory generation problem. The procedure is illustrated in Figure 1.

We generate the feasible trajectories, represented by $\eta_{\mathrm{ref},j} := (\eta_{x,\mathrm{ref},j}, \eta_{u,\mathrm{ref},j})$, by minimizing, for the j-th interval,

$$\int_0^{N_{\mathrm{pred}}T} \frac{1}{2}\Big\{\Delta_{x,j}(t)^{\mathsf{T}}\bar{Q}\Delta_{x,j}(t) + \tilde{u}_j(t)^{\mathsf{T}}\bar{R}\tilde{u}_j(t)\Big\}\mathrm{d}t \tag{13}$$

with respect to $\eta_{\mathrm{ref},j}$, where $\bar{Q} \succeq 0$ and $\bar{R} \succ 0$ are suitable tuning matrices,

$$\Delta_{x,j}(t) := \bar{x}_j(t) - (I_n \otimes \tau(t))^{\mathsf{T}}\eta_{x,\mathrm{ref},j}, \tag{14}$$

and

$$\tilde{u}_j(t) := (I_m \otimes \tau(t))^{\mathsf{T}}\eta_{u,\mathrm{ref},j}. \tag{15}$$

Using the properties of the Kronecker product, and eliminating the terms not depending on $\eta_{\text{ref},j}$, (13) can be rearranged as,

$$\frac{1}{2}\left\{\eta_{x,\text{ref},j}^{\mathsf{T}}(\bar{Q}\otimes J_s)\eta_{x,\text{ref},j}+\eta_{u,\text{ref},j}^{\mathsf{T}}(\bar{R}\otimes J_s)\eta_{u,\text{ref},j}\right\}-\int_0^{N_{\text{pred}}T}\bar{x}_j(t)^{\mathsf{T}}\bar{Q}(I_n\otimes\tau(t))^{\mathsf{T}}\eta_{x,\text{ref},j}\mathrm{d}t,$$

where

$$J_s:=\int_0^{N_{\text{pred}}T}\tau(t)\tau(t)^{\mathsf{T}}\mathrm{d}t. \tag{16}$$

The matrix J_s can be computed efficiently considering that

$$M_\lambda J_s=\int_0^{N_{\text{pred}}T}M_\lambda\tau(t)\tau(t)^{\mathsf{T}}\mathrm{d}t=\int_0^{N_{\text{pred}}T}\dot{\tau}(t)\tau(t)^{\mathsf{T}}\mathrm{d}t, \tag{17}$$

and integrating by parts results in

$$M_\lambda J_s=\left[\tau(N_{\text{pred}}T)\tau(N_{\text{pred}}T)^{\mathsf{T}}-\tau(0)\tau(0)^{\mathsf{T}}\right]-\int_0^{N_{\text{pred}}T}\tau(t)\dot{\tau}(t)^{\mathsf{T}}\ \mathrm{d}t \tag{18}$$

$$=\left[\tau(N_{\text{pred}}T)\tau(N_{\text{pred}}T)^{\mathsf{T}}-\tau(0)\tau(0)^{\mathsf{T}}\right]-J_sM_\lambda^{\mathsf{T}}. \tag{19}$$

This leads to the Lyapunov equation,

$$M_\lambda J_s+J_sM_\lambda^{\mathsf{T}}-\left[\tau(N_{\text{pred}}T)\tau(N_{\text{pred}}T)^{\mathsf{T}}-\tau(0)\tau(0)^{\mathsf{T}}\right]=0$$

that gives J_s as a unique solution. Existence and uniqueness of the solution J_s of the above Lyapunov equation is guaranteed, since the triangular matrix M_λ is negative definite, see [27, p. 114].

Therefore, the complete problem at each interval j reduces to

$$\inf_{\substack{\eta_{x,\text{ref},j}\\ \eta_{u,\text{ref},j}}}\frac{1}{2}\left\{\eta_{x,\text{ref},j}^{\mathsf{T}}(\bar{Q}\otimes J_s)\eta_{x,\text{ref},j}+\eta_{u,\text{ref},j}^{\mathsf{T}}(\bar{R}\otimes J_s)\eta_{u,\text{ref},j}\right\}$$

$$-\int_0^{N_{\text{pred}}T}\bar{x}_j(t)^{\mathsf{T}}\bar{Q}(I_n\otimes\tau(t))^{\mathsf{T}}\eta_{x,\text{ref},j}\mathrm{d}t \tag{20}$$

$$\text{s.t.}\quad A_x\eta_{x,\text{ref},j}+B_u\eta_{u,\text{ref},j}=0, \tag{21}$$

$$(I_n\otimes\tau(0)^{\mathsf{T}})\eta_{x,\text{ref},j}+x_{\text{b},j}$$

$$=(I_n\otimes\tau(T)^{\mathsf{T}})\eta_{x,\text{ref},j-1}+x_{\text{b},j-1}, \tag{22}$$

$$(I_m\otimes\tau(0)^{\mathsf{T}})\eta_{u,\text{ref},j}=(I_m\otimes\tau(T)^{\mathsf{T}})\eta_{u,\text{ref},j-1}, \tag{23}$$

$$F\eta_{u,\text{ref},j}\in\mathcal{U}. \tag{24}$$

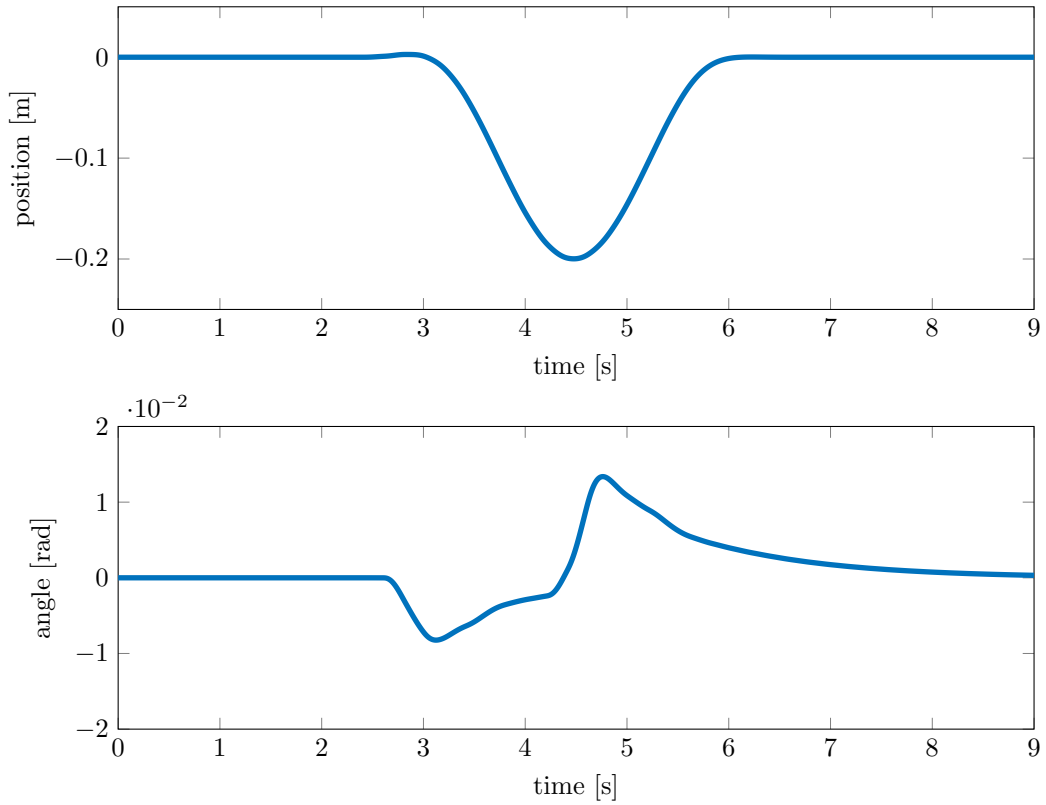


Figure 2. Trajectory generation results. The upper plot shows a sinusoidal pulse in the x-position for the Flying Platform, an unmanned aerial vehicle to be presented in Section 6. The lower plot shows the corresponding pitch (Euler angle) over the same horizon. This state was not specified among the desired ones, so the algorithm has the freedom of finding an appropriate feasible trajectory.

In case parts of the desired state trajectory are not provided, the corresponding value in $\bar{Q}$ should be set to zero (or very close to zero) for the corresponding state. In this way, a higher flexibility will be left to the algorithm in order to find a feasible trajectory.

The constraints (22) and (23) are introduced in order to preserve the continuity of the states and the inputs, such that the trajectory defined by $\eta_{\text{ref,j}}$ starts exactly where the trajectory defined by $\eta_{\text{ref,j-1}}$ ends. In a similar way, both constraints are replaced for the interval $j = 0$ by

$$(I_n \otimes \tau(0)^{\mathsf{T}})\eta_{x,\text{ref},0} = \bar{x}_{\text{des}}(0) - x_{\text{b},0}. \tag{25}$$

The results of the problem on two of the states of the system that we will present in Section 6 are shown in Figure 2.

4. Online trajectory tracking

Once a trajectory has been generated offline, it can be tracked online by the MPC con-

troller. At each time step, a counter is used to detect the current interval and load the appropriate $\eta_{\text{ref},j}$ and $x_{\text{b},j}$. For the time instants within the intervals, the parameters have to be shifted in time by pre-multiplying a delay matrix to the current $\eta_{\text{ref},j}$. Considering a single state x, and given the sampling time T_s and the time instant kT_s, $k = 0, \ldots, K-1$, we have, according to (3),

$$\tau(kT_s) = e^{M_\lambda kT_s}\tau(0). \tag{26}$$

Within the j-th interval, this leads to

$$\begin{aligned} x(kT_s) = \tau(kT_s)^\mathsf{T}\eta_{x,\text{ref},j} &= \tau(0)^\mathsf{T}e^{M_\lambda^\mathsf{T} kT_s}\eta_{x,\text{ref},j} \\ &= \tau(0)^\mathsf{T}\eta_{x,\text{ref,d},j}. \end{aligned} \tag{27}$$

The above formula (27) extends naturally to multiple states and inputs, by virtue of the Kronecker product. Therefore, the corresponding shifted parameters are calculated as

$$\eta_{\text{ref,d},j} = (I_{n+m} \otimes e^{kM_\lambda^\mathsf{T} T_s})\eta_{\text{ref},j}. \tag{28}$$

We will drop the subscript '$_\text{d}$' from now on to simplify notation. As a result, in the j-th interval, the following problem is solved online:

$$\begin{aligned} \inf_{\eta_x,\eta_u} \frac{1}{2}\Big\{ & (\eta_x - \eta_{x,\text{ref},j})^\mathsf{T} (Q \otimes I_s) (\eta_x - \eta_{x,\text{ref},j}) \\ & + (\eta_u - \eta_{u,\text{ref},j})^\mathsf{T} (R \otimes I_s) (\eta_u - \eta_{u,\text{ref},j}) \Big\} \\ \text{s.t.} \quad & A_x\eta_x + B_u\eta_u = 0, \\ & (I_n \otimes \tau(0)^\mathsf{T})\eta_x = x_0 - x_{\text{b},j}, \\ & F\eta_u \in \mathcal{U}, \end{aligned} \tag{29}$$

where x_0 is an estimate of the current position. The problem can be solved with the GFDG method as proposed in Section 2, using the change of variables $\eta'_x = \eta_x - \eta_{x,\text{ref},j}$, $\eta'_u = \eta_u - \eta_{u,\text{ref},j}$.

5. Trajectory tracking with iterative learning

In order to account for the repeatable disturbances, we implement an iterative learning scheme. It consists of a Kalman Filter, see [28, Ch. 8], for estimating these repeatable disturbances, which are then included in the proposed MPC framework. The use of a Kalman Filter is motivated by the fact that it is a well-established technique, easily

tunable, and provides a means to incorporate prior information.

We model the disturbances as additive noise that enters the dynamics through the matrix $G \in \mathbb{R}^{n \times n_v}$,

$$\dot{x}(t) = Ax(t) + Bu(t) + Gv(t), \quad \forall t \in [0, \infty), \tag{30}$$

where $v(t) \in \mathbb{R}^{n_v}$ is the disturbance vector. In analogy to the system's state and input, which are modeled using $\tilde{x}$ and $\tilde{u}$, the disturbance $v(t)$ is assumed to have the form

$$\tilde{v}(t) := (I_{n_v} \otimes \tau(t))^{\mathsf{T}} \eta_v. \tag{31}$$

As a result, the equality constraint capturing the system dynamics in (29) is reformulated as

$$A_x \eta_x + B_u \eta_u + G_v \eta_v = 0, \quad G_v := G \otimes I_s. \tag{32}$$

Note that, as remarked earlier, trajectories $\tilde{x}(t)$, $\tilde{u}(t)$ and $\tilde{v}(t)$ satisfying (32) and the initial condition $\tilde{x}(0) = x_0$, fulfill the equations of motion (30) exactly.

We discretize the augmented system's model in (30) as (assuming a zero-order hold sampling),

$$x[k+1] = A_{\mathrm{d}} x[k] + B_{\mathrm{d}} u[k] + G_{\mathrm{d}} v[k], \tag{33}$$

where $k = 0, 1, \ldots$, A_{d}, B_{d} and G_{d} are the discrete-time matrix representation of (30) with sampling time T_s, and $x[k] = x(kT_s)$. We aim at estimating the disturbance trajectory $v[k]$ based on the data available from the previous trials.

As we are interested in capturing the repeatable parts of the disturbance, we use the following model to describe the evolution of $v[k]$ over different trials,

$$v[k]^{i+1} = v[k]^i + q[k]^i, \quad q[k]^i \sim \mathcal{N}(0, Q_{\mathrm{KF}}), \tag{34}$$

where $q[k]^i$ denotes the process noise that is assumed to be normally distributed with zero mean and variance Q_{KF}, and the superscript i refers to the trial number. Thus, the prediction step of the Kalman filter is given by

$$\hat{v}_{\mathrm{p}}[k]^{i+1} = \hat{v}_{\mathrm{m}}[k]^i, \quad P_{\mathrm{p}}[k]^{i+1} = P_{\mathrm{m}}[k]^i + Q_{\mathrm{KF}}, \tag{35}$$

where $\hat{v}_{\mathrm{p}}[k]^i$ and $P_{\mathrm{p}}[k]^i$ indicate the expected value and the variance of the random variable $v[k]^i$ conditioned on the trajectory data up to the $(i-1)$-th trial, while $v_{\mathrm{m}}[k]^i$ and $P_{\mathrm{m}}[k]^i$ indicate the expected value and the variance of the same random variable conditioned on

the trajectory data up to the i-th trial.

During the execution of a trajectory, the actual states and inputs are recorded and are used to update the estimate of $v[k]$. We use (33) to formulate the measurement equation,

$$\underbrace{x[k+1]^i - A_\mathrm{d}x[k]^i - B_\mathrm{d}u[k]^i}_{:=z[k]^i} = \quad G_\mathrm{d}v[k]^i + n[k]^i,$$

with $n[k]^i \sim \mathcal{N}(0, R_\mathrm{KF})$. The measurement update equations can be expressed as

$$P_\mathrm{m}[k]^i = \left((P_\mathrm{p}[k]^i)^{-1} + G_\mathrm{d}^\mathsf{T} R_\mathrm{KF}^{-1} G_\mathrm{d}\right)^{-1} \tag{36}$$

$$\hat{v}_\mathrm{m}[k]^i = \hat{v}_\mathrm{p}[k]^i + P_\mathrm{m}[k]^i G_\mathrm{d}^\mathsf{T} R_\mathrm{KF}^{-1}(z[k]^i - G_\mathrm{d}\hat{v}_\mathrm{p}[k]^i). \tag{37}$$

Once the Kalman filter updates are performed, the current estimated disturbance trajectory $v_\mathrm{m}[k]^i$ is transformed to the continuous-time domain through a zero-order hold, yielding $v(t)^i$. The latter is transformed to the parameter space by performing a curve fitting that minimizes a regularized L^2-distance. Again, the Laguerre functions are not suitable for fitting a curve over a large interval, as they only have few degrees of freedom. To tackle this problem, we use a similar approach as in the trajectory generation, and split the trajectory $v(t)^i$ in N_v pieces of length T_v. In order to simplify notation, we present this fitting procedure for one-dimensional disturbances, but it can be easily extended to the multi-dimensional case. Introducing the quantities $v_j(t)^i := v(t + jT_v)^i$, the resulting fitting procedure (for each interval j) reduces to,

$$\eta_{v,j}^i := \arg\min_\eta \int_0^{T_v} \frac{1}{2}\left\{ \left(v_j(t)^i - \tau(t)^\mathsf{T}\eta\right)^2 + r\eta^\mathsf{T}\dot{\tau}(t)\dot{\tau}(t)^\mathsf{T}\eta \right\}\mathrm{d}t, \tag{38}$$

where the first integrand penalizes the squared distance to the disturbance trajectory, while the second term performs a regularization in order to increase the smoothness of the resulting trajectory $\tau(t)^\mathsf{T}\eta_{v,j}^i$. The regularization is controlled with the tuning parameter $r \in \mathbb{R}$, $r \geq 0$.

The optimization problem (38) can be solved analytically leading to

$$\eta_{v,j}^i = \left(J_v + rM_\lambda J_v M_\lambda^\mathsf{T}\right)^{-1} \int_0^{T_v} v_j(t)^i \tau(t)\mathrm{d}t, \tag{39}$$

where

$$J_v := \int_0^{T_v} \tau(t)\tau(t)^\mathsf{T}\mathrm{d}t \tag{40}$$

is computed in a similar way as in (16)-(18).

The algorithm that runs at each trajectory execution is summarized by the pseudo-code given in Algorithm 1.

Algorithm 1 Pseudo-code for disturbances estimation.

Result: Estimate η_v^i corresponding to the i-th trial
Record $x[k]^i$, $u[k]^i$ over the entire trajectory;
Update the Kalman filter state $v[k]^i$;
Split the interpolated $v(t)^i$ over N_v intervals $\rightarrow v_j(t)^i$;
for *each interval* j **do**
  Fit $\eta_{v,j}^i$ through $v_j(t)^i$;
end
Merge all the estimates and store a lifted vector $\eta_v^i = \left(\eta_{v,0}^i, \eta_{v,1}^i \cdots\right)$

After the i-th trial, the current estimate of the disturbance parameters is used to replan a feasible trajectory. This is done by solving again the generation problem in (20), replacing (21) with

$$A_x \eta_{x,\mathrm{ref},j} + B_u \eta_{u,\mathrm{ref},j} + G_v \eta_{v,j}^i = 0, \tag{41}$$

to include the estimated disturbances in the system's dynamics. In a similar way, in the online trajectory tracking problem the equality constraint in (29) is replaced by

$$A_x \eta_x + B_u \eta_u + G_v \eta_{v,j}^i = 0. \tag{42}$$

The vector $\eta_{v,j}^i$ is shifted in time in a similar way as described by (28).

6. Experimental results

6.1 Hardware and software

The Flying Platform is a flying machine consisting of three electrical ducted fans mounted on a lightweight frame, see [29]. Two flaps are attached to each fan in order to control the air flow and perform thrust vectoring. A PX4FMU[1] is used as a flight controller for actuation and measuring the angular velocities with the built-in gyroscope. The PX4 communicates through a serial bus with a Gumstix DuoVero Zephyr computer-on-module (COM)[2], equipped with a dual-core ARM Cortex-A9 that runs at 1 GHz. The COM

[1] `www.pixhawk.org`
[2] `www.gumstix.org`

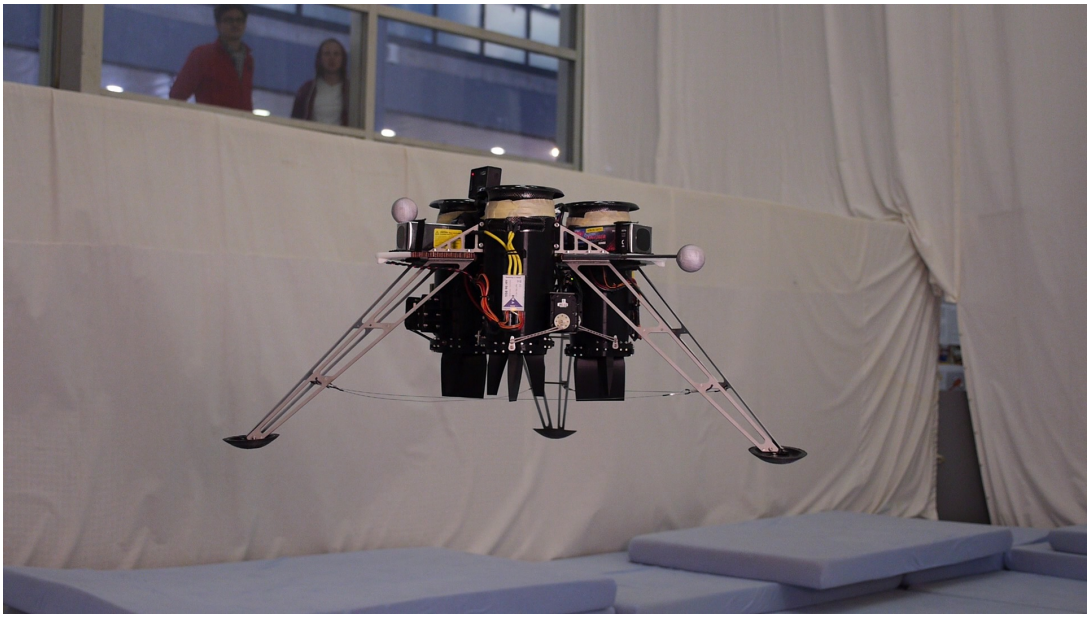


Figure 3. The Flying Platform during flight.

disposes of 1GB RAM. Position and attitude information is provided by a Vicon[3] motion capture system. Translational velocities are estimated through off-board state estimation techniques. The Flying Platform receives the data computed off-board, that is, the actual position, attitude and (linear) velocity, through a wireless communication.

The parametrized MPC routine and the iterative learning are implemented on the Gumstix COM, that runs a Linux-based operating system.

6.2 Model

A first-principles model is linearized about hover. In order to be invariant to different yaw set-points, the linearization is carried out in a yaw-fixed body coordinate system. The Flying Platform model has 12 states, that is, position, translational velocities, attitude, and angular velocities, and 9 inputs (3 per fan), which are given by $T_{z,i}$ (vertical thrusts) and $T_{x,i}$, $T_{y,i}$ (horizontal components, in perpendicular directions to each flap), with $i = 1, 2, 3$. All the input saturations can be approximated as box constraints, as shown in [4].

6.3 Results

We choose $s = 5$, and an exponential decay $\lambda = 5\text{s}^{-1}$. The following matrices are chosen for the learning, with $n_v = 12$,

$$G = I_{n_v}, \quad Q_{\text{KF}} = 10^{-1} \cdot I_{n_v}, \quad R_{\text{KF}} = 10^{-5} \cdot I_n.$$

[3] `www.vicon.com`

parameter	value	description
T	0.05 s	interval length in trajectory generation
N_{pred}	25	prediction horizon for tracking (multiples of T)
r	0.1	regularization in learning fit
T_v	1 s	interval length in learning fit

Table 1. Tuning parameters.

Our choice of G doesn't specify any weights on how the disturbances enter the system. The choice of Q_{KF} leaves some freedom to our assumption of invariance of the disturbances over different trials, while the low magnitude of the values in R_{KF} emphasizes the considerable trust we give to the measurements.

The following tuning matrices are chosen for the controller presented in (29), which runs at a sampling frequency of 100 Hz,

$$\begin{aligned} Q &= \operatorname{diag}\big(\overbrace{200,200,30}^{\text{position}},\overbrace{10,10,10}^{\text{lin. velocity}},\overbrace{40,40,10}^{\text{attitude}},\overbrace{10,10,10}^{\text{ang. velocity}},5\big) \\ R &= 10^{-3}\cdot\operatorname{diag}\big(\underbrace{1.7,0.85,15}_{T_{x,1},T_{y,1},T_{z,1}},\underbrace{1.7,0.85,15}_{T_{x,2},T_{y,2},T_{z,2}},\underbrace{1.7,0.85,15}_{T_{x,3},T_{y,3},T_{z,3}}\big)\,. \end{aligned}$$

The remaining tuning parameters are summarized in Table 1. The desired task is to track a circle with a diameter of 20 cm in 6 s. Six consecutive trials have been performed. The resulting tracking performance is shown in Figure 4.[4]

Due to the estimation of the disturbances η_v, the predictions of the MPC are more accurate as the number of trials increases, which improves the tracking performance. The repeatable disturbances are most likely due to asymmetries in the mass distribution of the real system.

7. Conclusion

A parametrized MPC approach has been extended to the trajectory tracking case, and a learning scheme has been introduced. To this extent, the inherent limitations of the Laguerre functions that are used to parametrize the state and input trajectories are addressed by introducing multiple prediction intervals. Still, accounting for large constant disturbances remains challenging due to the decaying nature of the basis functions and requires careful tuning. The same applies to disturbances that quickly change in time, due to the bounded derivative of the basis functions. The method has been tested on an unmanned aerial vehicle, showing satisfactory tracking performance, and achieving a

[4] A video showing some of the experiments can be found at the following link: `https://youtu.be/GgIwrnoNvTY`.

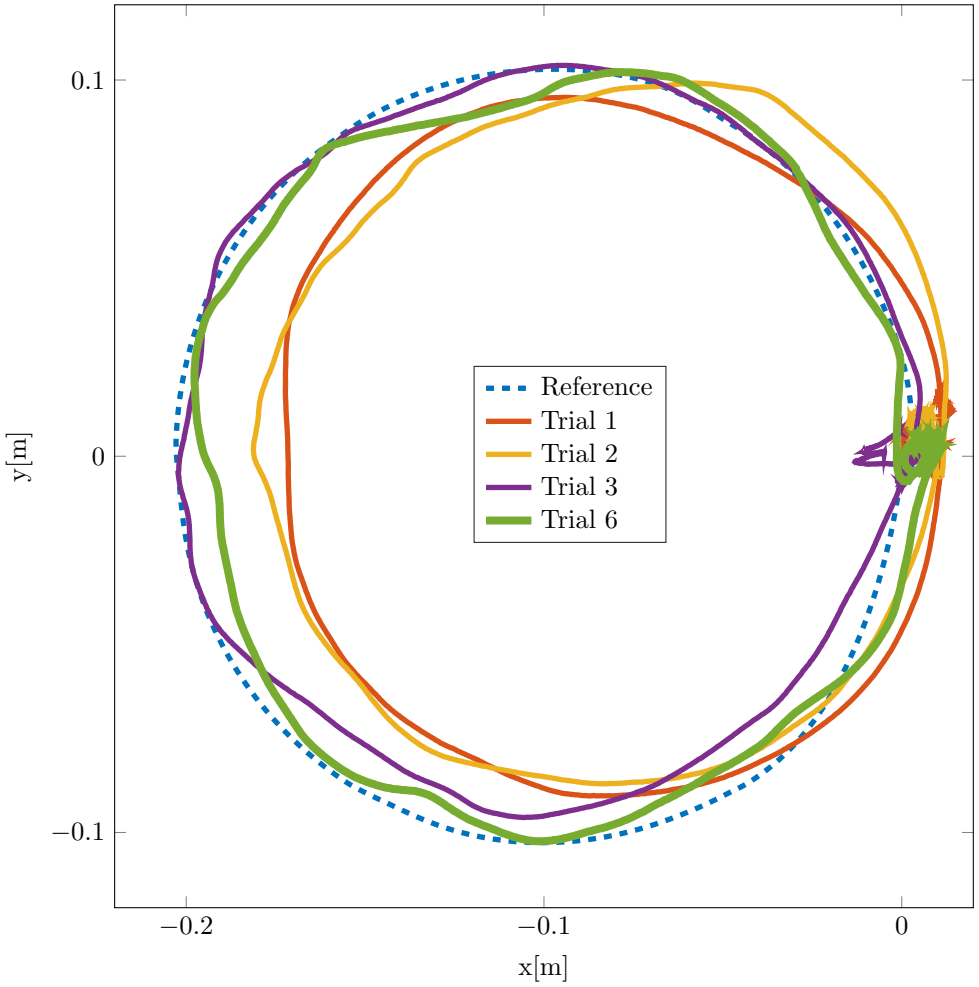


Figure 4. Trajectory tracking of a circle (counter-clockwise direction, starting from the origin). The feasible (parametrized) trajectory matches the desired trajectory in the xy-plane, and is therefore not shown. The system improves the accuracy of the predictions used for MPC, increasing the tracking performance over subsequent trials, before reaching a steady-state after six trials.

sampling frequency of 100 Hz. The experiments have also shown how the system improves in performing the task over subsequent trials.

Acknowledgments

The authors would like to thank Marc-Andrè Corzillius, Michael Egli, Tobias Meier and Lukas Fröhlich for their contribution to the development of the Flying Platform. This work was supported by ETH-grant ETH-48 15-1.

The experiments of this research were carried out in the Flying Machine Arena. A list of present and past participants is available at `http://flyingmachinearena.org/people/`.

References

[1] M. Morari and J. H. Lee, "Model predictive control: Past, present and future", *Computers & Chemical Engineering*, vol. 23, no. 4-5, pp. 667–682, 1999.

[2] M. Alamir and G. Bornard, "Stability of a truncated infinite constrained receding horizon scheme: The general discrete nonlinear case", *Automatica*, vol. 31, no. 9, pp. 1353–1356, 1995.

[3] M. Muehlebach and R. D'Andrea, "Parametrized infinite-horizon model predictive control for linear time-invariant systems with input and state constraints", *American Control Conference*, pp. 2669–2674, 2016.

[4] M. Hofer, M. Muehlebach, and R. D'Andrea, "Application of an approximate model predictive control scheme on an unmanned aerial vehicle", *International Conference on Robotics and Automation*, pp. 2952–2957, 2016.

[5] Y. Wang and S. Boyd, "Fast model predictive control using online optimization", *Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, 2009.

[6] A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones, "Efficient interior point methods for multistage problems arising in receding horizon control", *Conference on Decision and Control*, pp. 668–674, 2012.

[7] L. Wang, "Continuous time model predictive control design using orthogonal functions", *International Journal of Control*, vol. 74, no. 16, pp. 1588–1600, 2001.

[8] L. Wang, *Model Predictive Control System Design and Implementation Using MATLAB*. Springer, 2009.

[9] W. B. Dunbar, M. B. Milam, R. Franz, and R. M. Murray, "Model predictive control of a thrust-vectored flight control experiment", *IFAC World Congress*, vol. 35, no. 1, pp. 355–360, 2002.

[10] F. Kühne, J. M. G. da Silva Jr., and W. F. Lages, "Mobile robot trajectory tracking using model predictive control", *Latin American Robotics Symposium*, 2005.

[11] T. Faulwasser and R. Findeisen, "A model predictive control approach to trajectory tracking problems via time-varying level sets of Lyapunov functions", *Conference on Decision and Control and European Control Conference*, pp. 3381–3386, 2011.

[12] A. Alessandretti, A. P. Aguiar, and C. N. Jones, "Trajectory-tracking and path-following controllers for constrained underactuated vehicles using model predictive control", *European Control Conference*, pp. 1371–1376, 2013.

[13] S. Gros, R. Quirynen, and M. Diehl, "Aircraft control based on fast non-linear MPC & multiple-shooting", *Conference on Decision and Control*, pp. 1142–1147, 2012.

[14] K. Alexis, G. Nikolakopoulos, and A. Tzes, "On trajectory tracking model predictive control of an unmanned quadrotor helicopter subject to aerodynamic disturbances", *Asian Journal of Control*, vol. 16, no. 1, pp. 209–224, 2014.

[15] B. Vanek, T. Péni, J. Bokor, and G. Balas, "Practical approach to real-time trajectory tracking of UAV formations", *American Control Conference*, pp. 122–127, 2005.

[16] Z. Bien and J. X. Xu, *Iterative Learning Control: Analysis, Design, Integration and Applications*. Kluwer Academic Publishers, 1998.

[17] Y. Wang, F. Gao, and F. J. Doyle III, "Survey on iterative learning control, repetitive control, and run-to-run control", *Journal of Process Control*, vol. 19, no. 10, pp. 1589–1600, 2009.

[18] A. Schoellig and R. D'Andrea, "Optimization-based iterative learning control for trajectory tracking", *European Control Conference*, pp. 1505–1510, 2009.

[19] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, "Provably safe and robust learning-based model predictive control", *Automatica*, vol. 49, no. 5, pp. 1216–1226, 2013.

[20] N. Amann, D. H. Owens, and E. Rogers, "Predictive optimal iterative learning control", *International Journal of Control*, vol. 69, no. 2, pp. 203–226, 1998.

[21] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search", *International Conference on Robotics and Automation*, pp. 528–535, 2016.

[22] E. J. Adam and A. H. González, "Iterative learning - MPC: An alternative strategy", in *Frontiers in Advanced Control Systems*, G. L. de Oliveira Serra, Ed., InTech, 2012, ch. 9.

[23] K. K. Tan, S. N. Huang, T. H. Lee, and A. Tay, "Disturbance compensation incorporated in predictive control system using a repetitive learning approach", *Systems & Control Letters*, vol. 56, no. 1, pp. 75–82, 2007.

[24] J. R. Cueli and C. Bordons, "Iterative nonlinear model predictive control. Stability, robustness and applications", *Control Engineering Practice*, vol. 16, no. 9, pp. 1023–1034, 2008.

[25] P. Bouffard, A. Aswani, and C. Tomlin, "Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results", *International Conference on Robotics and Automation*, pp. 279–284, 2012.

[26] M. Muehlebach and R. D'Andrea, "Approximation of continuous-time infinite-horizon optimal control problems arising in model predictive control", *Conference on Decision and Control*, pp. 1464–1470, 2016.

[27] K. J. Åström and R. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008.

[28] D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley, 2006.

[29] M. Muehlebach and R. D'Andrea, "The Flying Platform - A testbed for ducted fan actuation and control design", *Mechatronics*, vol. 42, pp. 52–68, 2017.

Paper P7

Learning-based parametrized model predictive control for trajectory tracking

Carmelo Sferrazza, Michael Muehlebach and Raffaello D'Andrea

Abstract

The article is concerned with the tracking of non-equilibrium motions with model predictive control. It proposes to parametrize input and state trajectories of a dynamic system with basis functions to alleviate the computational burden in model predictive control. As a result of the parametrization, an optimization problem with fewer variables is obtained, and the memory requirements for storing the reference trajectories are reduced. The article also discusses the generation of feasible reference trajectories that account for the system's dynamics, as well as input and state constraints. In order to cope with repeatable disturbances, which may stem from unmodeled dynamics for example, an iterative learning procedure is included. The approach relies on a Kalman filter that identifies the repeatable disturbances based on previous trials. These are then included in the system's model available to the model predictive controller, which compensates them in subsequent trials. The proposed approach is evaluated on a quadcopter, whose task is to balance a pole, while flying a predefined trajectory.

Published in *Optimal Control Applications and Methods*.

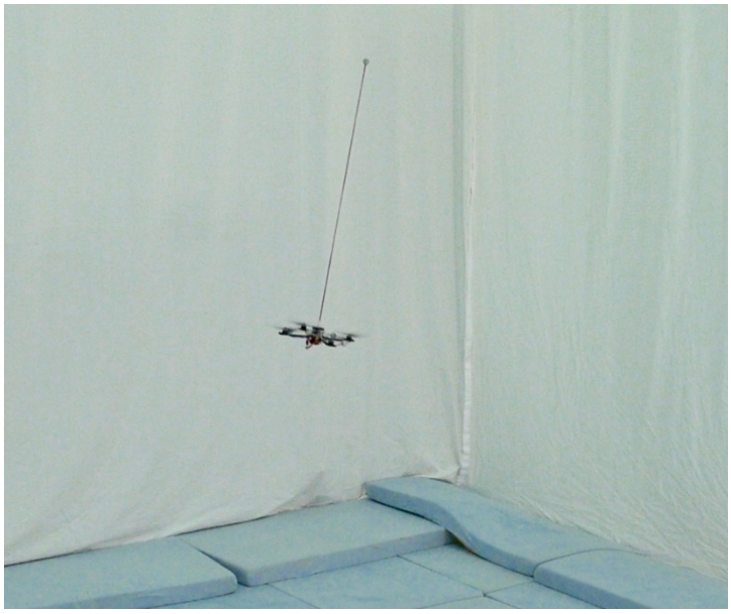


Figure 1. The flying inverted pendulum system used for the experiments presented in this article.

1. Introduction

Model predictive control (MPC) is an effective and popular control strategy for systems that have input and state constraints. There are numerous examples where model predictive control has been successfully applied in practice, see for example the works of Richalet et al. [1], Borrelli et al. [2], Geyer et al. [3], Papafotiou et al. [4]. The underlying idea is to repeatedly solve an optimal control problem that takes input and state constraints into account, and generates optimal input and state trajectories that guide the system from its current state to a desired state (often the origin). In order to be robust against modeling errors and disturbances, only a small portion of the input trajectory is applied to the system, before resolving the optimal control problem, subject to the actual position as an initial condition [5].

As a consequence, the underlying optimal control problem often needs to be solved very quickly in order to achieve a sufficiently small sampling time. Even if the optimization is warm-started, the resulting computational load is often substantial and represents a bottleneck of MPC. To reduce the computational load, the optimal control problem is often simplified, for example by using a coarser system model with fewer dimensions, and/or by reducing the prediction horizon. However, both the simplified system model and the smaller prediction horizon may degrade the resulting closed-loop performance and may even lead to instability.

The parametrization of the input and state trajectories with basis functions is an approach that avoids the discretization and the truncation of the time horizon. For the regulator problem, it has been shown in the work of Muehlebach and D'Andrea [6] that the resulting MPC controller asymptotically stabilizes the origin and is recursively feasi-

ble (without the need for a terminal constraint and a terminal cost). This article discusses the extension of such a parametrized MPC strategy from the regulator problem to the tracking problem. In fact, this step is non-trivial, particularly with regard to practical implementation. In addition, a strategy to identify and account for repeatable disturbances is presented, in case the same trajectory is executed multiple times. The implementation of the resulting learning-based MPC control strategy on a flying inverted pendulum system (see Figure 1) is discussed.

Related work: In the work of Dunbar et al. [7] a testbed for a ducted-fan actuated flying vehicle is controlled with MPC. The authors discuss the regulator problem and it is shown that the region of attraction of the MPC controller is larger than that of the related linear quadratic regulator.

Various works focus on reducing the computational complexity of MPC, see for example move-blocking strategies [8] or tailored MPC solvers [9], [10]. The works of Wang [11], Rossiter and Wang [12], and Khan and Rossiter [13] propose parametrizing the input trajectories with Laguerre or Kauz basis functions. The potential benefits of the parametrization in terms of performance and extension of the feasibility regions are discussed.

The article by Faulwasser and Findeisen [14] proposes an extension of MPC from the regulation problem to trajectory tracking. A nonlinear and continuous-time MPC formulation is presented, including time-varying terminal set constraints that guarantee recursive feasibility and closed-loop stability. Stephens et al. [15] discuss the design and implementation of an MPC controller to track trajectories of an industrial machine tool servo drive. The authors suggest extending the reference trajectory, for example via first-order hold, and are able to reduce the input horizon (they distinguish between the input and the prediction horizons), such that they can apply explicit MPC. In the work of Neunert et al. [16], a general nonlinear control strategy is presented, which uses time-varying feedforward and feedback terms. The resulting cascaded control structure includes a low-level controller that can be executed at very high rates, while MPC is used at a higher-level for updating the parameters of the low-level controller. The research of Kamel et al. [17] compares a linear and a non-linear MPC controller for trajectory tracking with a hexacopter. The authors conclude that the nonlinear MPC controller outperforms the linear one for the experiments conducted. In the work of Mueller and D'Andrea [18], a diminishing horizon MPC controller is introduced to track interception trajectories with a quadcopter. The trajectories generated at each sampling step are obtained by optimizing a quadratic program that is tailored to the quadcopter's dynamics and is based on conservative bounds on the quadcopter's jerk. Compared to the approach proposed in this article, this results in a comparable computational effort. However, the algorithm does not provide convergence guarantees and does not adapt to the environment.

In order to cope with modeling errors a learning-based predictive controller is presented in the work by Bouffard et al. [19]. The dynamics are assumed to be affine and time-invariant and an extended Kalman filter is used to learn, respectively update, the dynamics. As a result, the authors demonstrate offset-free tracking and discuss the im-

provements of a step response compared to the nominal MPC controller. In the work of Wang et al. [20], iterative learning control is included in an MPC framework to control multi-phase batch processes. Two different strategies are proposed to design an updating law using model predictive control, which can intrinsically deal with constraints. A reference-free approach is presented in the work of Rosolia and Borrelli [21], where a nonlinear MPC controller improves the performance over repetitive tasks when the reference trajectory is not known. The design of terminal constraints at each iteration is proposed to guarantee performance improvement and stability requirements.

However, the approaches cited mostly rely on a discretized formulation of the dynamics and/or a piece-wise constant control input. The approach followed in this article is different: It aims to reduce the number of degrees of freedom and alleviate the computational burden of MPC by parametrizing input and state trajectories with basis functions. In addition to requiring less memory (see Section 3) and reducing computation, inherent (without the addition of terminal constraints) closed-loop stability and recursive feasibility results can be demonstrated when regulating linear time-invariant systems [6], [22]. This parametrization is originally presented in the work of Muehlebach and D'Andrea [6], where simulation results show the regulation of a linear time-invariant system to equilibrium, without addressing the problem of constraint satisfaction. In the work of Muehlebach et al. [22], a strategy to guarantee constraint satisfaction is implemented and the resulting algorithm is shown to run at 100 Hz on an embedded platform, regulating the system to equilibrium. In the work of Sferrazza et al. [23], a heuristic strategy has been proposed to apply this idea to the trajectory tracking problem. In this article, non-equilibrium motions are successfully tracked with a general framework that also accounts for the generation of feasible trajectories. This approach is demonstrated in experiments, where a quadrotor balances an inverted pendulum. In order to cope with the system's nonlinearities and unmodeled dynamics, an iterative learning approach that identifies the repeatable disturbances and augments the model accordingly is applied. The computation related to the learning can be done offline, thus not altering the complexity of the MPC algorithm during the tracking of a trajectory. The trajectory generation problem is formulated here as a single optimization over the entire trajectory duration. This differs from the work of Sferrazza et al. [23], where a greedy approach was used to simplify the optimization. The main differences between the two approaches are emphasized throughout this article.

Outline: Section 2 introduces the different building blocks of the proposed control strategy and shows how they are connected: In Subsection 2.1 the MPC controller that tracks the generated reference trajectories is presented. Subsection 2.2 describes how the reference trajectories are generated, and the identification of repeatable disturbances is explained in Subsection 2.3. The benefits in terms of memory usage are elaborated in Section 3. Section 4 presents experimental results, where a quadrotor balances an inverted pendulum while flying a three-dimensional figure-eight trajectory. The article concludes in Section 5 with remarks.

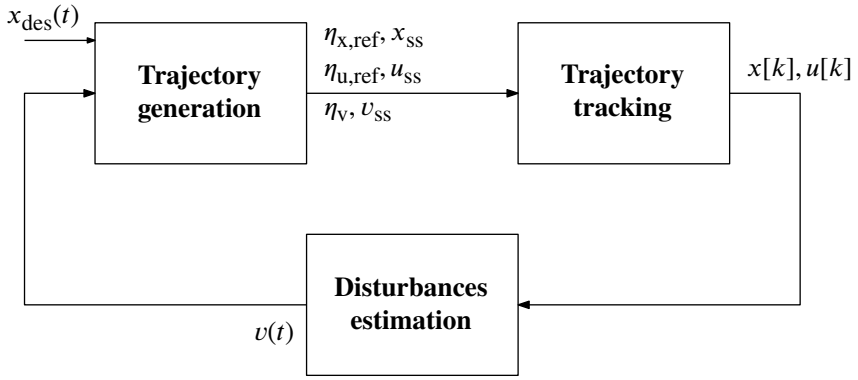


Figure 2. At each trial, the generated trajectory is followed by means of a trajectory tracking controller. The state and input rollouts are used to estimate the repeatable disturbances after the trial completion, and are then taken into account in the next generation of the trajectories to improve the tracking performance. Note that in this diagram $x_{\text{des}}(t)$ and $v(t)$ denote the state and disturbance trajectories in continuous time for the entire trajectory duration. Similarly $x[k]$ and $u[k]$ denote the state and input rollouts in discrete time for the entire trajectory duration.

2. Method

The following section proposes a parametrized MPC approach to generate and track non-equilibrium motions of a dynamic system. It is assumed that the system can be roughly approximated by linear time-invariant dynamics. A scheme of the proposed framework is shown in Figure 2 and consists of the following building blocks: The *trajectory generation* procedure takes a desired trajectory, $x_{\text{des}}(t)$, and, based on an estimate of the repeatable disturbances $v(t)$, computes feasible reference trajectories for the state and input. These trajectories are parametrized by linear combinations of basis functions and include steady-state offsets (as indicated by the variables $\eta_{\text{x,ref}}, \eta_{\text{u,ref}}, \eta_{\text{v}}, x_{\text{ss}}, u_{\text{ss}}$, and v_{ss}). The *trajectory tracking* procedure implements an (online) MPC controller for tracking the reference trajectories. The actual state and input trajectories are recorded, and after execution of the trajectory, the *disturbance estimation* procedure updates the estimate $v(t)$ of the repeatable disturbances. In the first iteration, the disturbance estimate $v(t)$ is typically initialized with zero, unless some prior knowledge is available.

The following subsections describe each individual block in detail. In the remainder of this article, vectors are expressed as tuples, with dimension and stacking clear from the context.

2.1 Trajectory tracking

In the following subsection the trajectory tracking procedure is described. As indicated in Figure 2, its aim is to track feasible reference trajectories, given a disturbance estimate.

All trajectories are parametrized with basis functions, that is,

$$\tilde{x}(t) = (I_n \otimes \tau(t))^\top \eta_{\mathrm{x}} + x_{\mathrm{ss}} \tag{1}$$
$$\tilde{x}_{\mathrm{ref}}(t) = (I_n \otimes \tau(t))^\top \eta_{\mathrm{x,ref}} + x_{\mathrm{ss}} \tag{2}$$
$$\tilde{u}(t) = (I_m \otimes \tau(t))^\top \eta_{\mathrm{u}} + u_{\mathrm{ss}} \tag{3}$$
$$\tilde{u}_{\mathrm{ref}}(t) = (I_m \otimes \tau(t))^\top \eta_{\mathrm{u,ref}} + u_{\mathrm{ss}} \tag{4}$$
$$\tilde{v}(t) = (I_{n_{\mathrm{v}}} \otimes \tau(t))^\top \eta_{\mathrm{v}} + v_{\mathrm{ss}}, \tag{5}$$

for all $t \in [0, \infty)$, where $\tilde{x}$ and $\tilde{u}$ denote the planned state and input trajectories, and $\tilde{x}_{\mathrm{ref}}$, $\tilde{u}_{\mathrm{ref}}$, and $\tilde{v}$ the reference and disturbance trajectories that are obtained from the trajectory generation routine and are fixed during trajectory tracking. The parameter vectors of the planned state and input trajectories, which are subject to the (online) optimization, are denoted by $\eta_{\mathrm{x}} \in \mathbb{R}^{ns}$ and $\eta_{\mathrm{u}} \in \mathbb{R}^{ms}$, the parameters of the reference state, the reference input, and the disturbance trajectory are denoted by $\eta_{\mathrm{x,ref}} \in \mathbb{R}^{ns}$, $\eta_{\mathrm{u,ref}} \in \mathbb{R}^{ms}$, and $\eta_{\mathrm{v}} \in \mathbb{R}^{n_{\mathrm{v}}s}$, and the steady-state offsets on the state, the input, and the disturbances are denoted by $x_{\mathrm{ss}} \in \mathbb{R}^n$, $u_{\mathrm{ss}} \in \mathbb{R}^m$, and $v_{\mathrm{ss}} \in \mathbb{R}^{n_{\mathrm{v}}}$. The integers n, m, and n_{v} describe the dimension of the state, the input, and the disturbances, and $\otimes$ refers to the Kronecker product. In contrast to the work of Sferrazza et al. [23], steady-state offsets are also included in the inputs and the disturbances. In addition, these offsets are decision variables in the optimization problem introduced in Subsection 2.2. This makes it possible to capture large steady-state disturbances and gives the trajectory generation algorithm the flexibility to trade off the placement of these offsets during the optimization. The basis functions $\tau(t) := (\tau_1(t), \tau_2(t), \ldots, \tau_s(t)) \in \mathbb{R}^s$, where s refers to the number of basis functions, are assumed to satisfy the following standing assumptions:

H1 The basis functions $\tau_1, \tau_2, \ldots, \tau_s$ are linearly independent.

H2 The basis functions satisfy the following first order differential equation $\dot{\tau}(t) = M\tau(t)$ for all $t \in [0, \infty)$, where the matrix $M \in \mathbb{R}^{s \times s}$ is Hurwitz.

Assumption H1 is required to guarantee that the resulting optimization problems have unique solutions. The motivation of Assumption H2 is twofold: First, if both H1 and H2 are fulfilled, linear time-invariant dynamics $\dot{\tilde{x}}(t) = A\tilde{x}(t) + B\tilde{u}(t) + G\tilde{v}(t)$ are fulfilled if and only if the corresponding parameter vectors satisfy

$$(I_n \otimes M^\top)\eta_{\mathrm{x}} = (A \otimes I_s)\eta_{\mathrm{x}} + (B \otimes I_s)\eta_{\mathrm{u}} + (G \otimes I_s)\eta_{\mathrm{v}},$$

where A, B and G are the system's matrices. A proof of this statement can be found in the work of Muehlebach and D'Andrea [6]. Second, if Assumption H2 is fulfilled, it implies that the basis functions are able to capture arbitrary time shifts. More precisely, given the trajectory $\tilde{x}(t)$ parametrized by the basis functions, the shifted version of this

trajectory, $\tilde{x}(t+\Delta t)$ can be parametrized with the same basis functions,

$$\tilde{x}(t+\Delta t) = (I_n \otimes \tau(t))^\top \underbrace{(I_n \otimes \exp(M^\top \Delta t))\eta_\mathrm{x}}_{\hat{\eta}_\mathrm{x}}, \tag{6}$$

for all $t \in [0, \infty)$, where $\Delta t \in [0, \infty)$ is any time shift. This is particularly important for fast online execution, where the optimization at the next time step can be warm-started with the solution obtained at the current time. The time-shift property also implies closed-loop stability and recursive feasibility of the resulting MPC algorithm in the regulation case (that is for $\eta_{\mathrm{x,ref}} = 0$, $\eta_{\mathrm{u,ref}} = 0$) [6].

Examples for basis functions that fulfill Assumptions H1 and H2 are exponentially decaying polynomials or exponentially decaying harmonics. In the remainder, exponentially decaying polynomials of the form

$$\tau(t) \in \exp(-\lambda t)\mathrm{span}(1, t, t^2, \ldots, t^{s-1})$$

will be used, which are obtained by choosing M as

$$M = \begin{pmatrix} -\lambda & 0 & \ldots & 0 \\ -2\lambda & -\lambda & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -2\lambda & -2\lambda & \ldots & -\lambda \end{pmatrix}. \tag{7}$$

The steady-state offsets x_ss, u_ss, v_ss, obtained from the trajectory generation procedure describe an equilibrium, that is,

$$0 = Ax_\mathrm{ss} + Bu_\mathrm{ss} + Gv_\mathrm{ss}.$$

The aim of the trajectory tracking procedure is to repeatedly plan state and input trajectories that start from the current state of the system and follow the reference trajectories $\tilde{x}_\mathrm{ref}$, $\tilde{u}_\mathrm{ref}$ as closely as possible. The reference trajectories $\tilde{x}_\mathrm{ref}$ and $\tilde{u}_\mathrm{ref}$ will ultimately converge to x_ss and u_ss (since $\lim_{t\to\infty} \tau(t) = 0$), hence the name steady-state offsets. This leads to the following optimization problem

$$\min_{\eta_\mathrm{x}, \eta_\mathrm{u}} \frac{1}{2} \int_0^\infty \left\{ (\tilde{x}(t) - \tilde{x}_\mathrm{ref}(t))^\top Q (\tilde{x}(t) - \tilde{x}_\mathrm{ref}(t)) + (\tilde{u}(t) - \tilde{u}_\mathrm{ref}(t))^\top R (\tilde{u}(t) - \tilde{u}_\mathrm{ref}(t)) \right\} \mathrm{d}t \tag{8}$$

$$\begin{aligned} \text{s.t. } & \dot{\tilde{x}}(t) = A\tilde{x}(t) + B\tilde{u}(t) + G\tilde{v}(t), \tilde{x}(0) = x_0 \\ & b_\mathrm{min} \leq C_\mathrm{x}\tilde{x}(t) + C_\mathrm{u}\tilde{u}(t) \leq b_\mathrm{max}, \forall t \in [0, \infty), \end{aligned} \tag{9}$$

where $Q \succeq 0$ and $R \succ 0$ are matrices of appropriate dimension. The problem is repeatedly solved subject to the current state x_0 as an initial condition (as common in MPC). Note that for simplicity, only a linear constraint on the state and input trajectories is included in the optimization. Slightly more general linear constraints including derivatives or integrals of the state and input trajectories can be dealt with in an analogous way.

Feedback control is achieved by repeatedly solving (8) and applying the first portion of the input $\tilde{u}(t)$, $t \in [0, T_\mathrm{s})$ to the system, where T_s denotes the sampling time. The reference trajectories $\tilde{x}_\mathrm{ref}$ and $\tilde{u}_\mathrm{ref}$, as well as the disturbances $\tilde{v}$ are adapted according to the time that has elapsed. Thus, at time $t = T_\mathrm{s}$, the reference and disturbance trajectories in (8) are modified to $\tilde{x}_\mathrm{ref}(t+T_\mathrm{s})$, $\tilde{u}_\mathrm{ref}(t+T_\mathrm{s})$, and $\tilde{v}(t+T_\mathrm{s})$. According to (6), this amounts to a simple multiplication of the parameter vectors $\eta_\mathrm{x,ref}$, $\eta_\mathrm{u,ref}$, and η_v with the matrix

$$I \otimes \exp(M^\top T_\mathrm{s}), \tag{10}$$

where the dimension of the identity matrix I is chosen accordingly.

For orthonormal basis functions[1], the above optimization problem reduces to [6]

$$\min_{\eta_\mathrm{x},\eta_\mathrm{u}} \frac{1}{2}(\eta_\mathrm{x} - \eta_\mathrm{x,ref})^\top (Q \otimes I_s)(\eta_\mathrm{x} - \eta_\mathrm{x,ref}) + \frac{1}{2}(\eta_\mathrm{u} - \eta_\mathrm{u,ref})^\top (R \otimes I_s)(\eta_\mathrm{u} - \eta_\mathrm{u,ref}) \tag{11}$$

$$\text{s.t. } (I_n \otimes M^\top)\eta_\mathrm{x} = (A \otimes I_s)\eta_\mathrm{x} + (B \otimes I_s)\eta_\mathrm{u} + (G \otimes I_s)\eta_\mathrm{v}, \tag{12}$$

$$(I_n \otimes \tau(0))^\top \eta_\mathrm{x} + x_\mathrm{ss} = x_0 \tag{13}$$

$$b_\mathrm{min} \leq F_\mathrm{x}\eta_\mathrm{x} + F_\mathrm{u}\eta_\mathrm{u} + D_\mathrm{x}x_\mathrm{ss} + D_\mathrm{u}u_\mathrm{ss} \leq b_\mathrm{max}. \tag{14}$$

The linear equality constraint (12) imposes the dynamics and (13) the initial condition. The linear inequality constraint (14) describes the input and state constraints. These are obtained by sampling the constraint (9). The values of the matrices F_x, F_u, D_x and D_u may change slightly for different applications, and are given in Appendix C for the system considered in Section 4.

The optimization therefore simplifies to solving the quadratic program (11), subject to the optimization variables $\eta_\mathrm{x} \in \mathbb{R}^{ns}$ and $\eta_\mathrm{u} \in \mathbb{R}^{ms}$. Due to the parametrization with the basis functions, the quadratic program (11) typically has few optimization variables and can be solved efficiently. As highlighted before, by exploiting the time-shift property of the basis functions (see (6)), the problem can be warm-started with the solution from the previous time step.

2.2 Trajectory generation

The goal of the trajectory generation is to generate feasible input, state and disturbance trajectories, such that the state trajectory follows the desired trajectory (given to the algorithm) as closely as possible. We assume that the desired trajectory might not nec-

[1]Without loss of generality, linear independent basis functions can always be made orthonormal. Choosing M according to (7) yields orthonormal basis functions.

essarily prescribe the motion for all states. For example, the desired trajectory could only contain a position reference that should be followed, while finding the appropriate reference trajectories for the velocities is left to the algorithm. The feasible trajectories satisfy the system's dynamics, as well as the input and state constraints.

The feasible trajectory is generated by solving an optimization problem, where the desired trajectory is given in the time domain. The resulting trajectories are parametrized by $\eta_{\mathrm{x,ref}}$, $\eta_{\mathrm{u,ref}}$, and η_v, and the steady-state offsets x_{ss}, u_{ss}, and v_{ss}, as indicated in Figure 2.

Due to the finite number of basis functions, only trajectories with a finite length can be approximated well. In order to overcome this limitation, the trajectory's duration is split into N smaller intervals of length T, and different linear combinations of basis functions are used to approximate the trajectory in each of these intervals, as shown in Figure 3. Therefore, the trajectory tracking problem (11) is modified such that, depending on the time that has elapsed, the reference trajectory in the corresponding interval is tracked, i.e. $\eta_{\mathrm{x,ref}}$, $\eta_{\mathrm{u,ref}}$, η_{v}, x_{ss} and u_{ss} are replaced by the corresponding $\eta_{\mathrm{x,ref},j}$, $\eta_{\mathrm{u,ref},j}$, $\eta_{\mathrm{v},j}$, $x_{\mathrm{ss},j}$ and $u_{\mathrm{ss},j}$, for $j = 0, 1, \ldots, N-1$. A practical advantage of this approach is the possibility of aborting the trajectory at any time during execution, leading the system to a safe equilibrium position (represented by the current interval's steady-state offsets). In that case, provided that the constraints are satisfied for all times (a sampling strategy is applied in this article), closed-loop stability and recursive feasibility are guaranteed [6], [22]. The approach is particularly suited to smooth desired trajectories, where the approximations at two consecutive intervals generally overlap for a certain time (see Figure 3). However, this might not be the case for trajectories with sharp changes at the time of switching between two intervals.

In order to find a feasible approximation that is closest to the desired trajectory in the L^2-sense, while penalizing the input and fitting the estimated disturbances, the set of reference parameters and steady-state offsets is chosen to minimize

$$\sum_{j=0}^{N-1} \Bigg\{ \int_0^T \frac{1}{2} \Big[\Delta_{\mathrm{x},j}(t)^\top \bar{Q} \Delta_{\mathrm{x},j}(t) + \tilde{u}_{\mathrm{ref},j}(t)^\top \bar{R} \tilde{u}_{\mathrm{ref},j}(t) + r \dot{\tilde{u}}_{\mathrm{ref},j}(t)^\top \dot{\tilde{u}}_{\mathrm{ref},j}(t) + \Delta_{\mathrm{v},j}(t)^\top \bar{V} \Delta_{\mathrm{v},j}(t) \Big] \mathrm{d}t \Bigg\}, \tag{15}$$

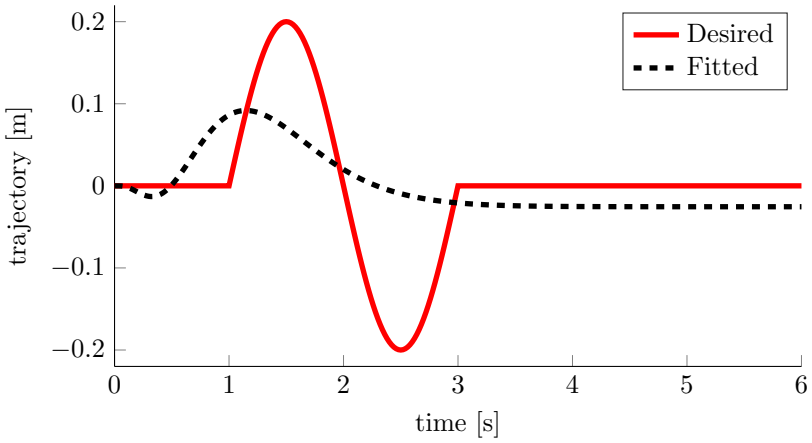
where r controls the smoothness of the input trajectories, $\bar{Q} \succeq 0$, $\bar{R} \succ 0$ and $\bar{V} \succeq 0$ are suitable symmetric tuning matrices, and,

$$\Delta_{\mathrm{x},j}(t) := x_j(t) - \left[(I_n \otimes \tau(t))^\top \eta_{\mathrm{x,ref},j} + x_{\mathrm{ss},j} \right], \tag{16}$$

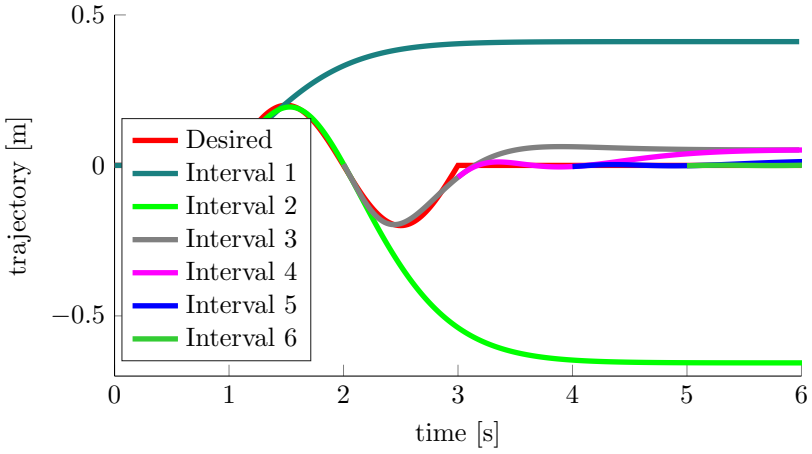
$$\tilde{u}_{\mathrm{ref},j}(t) := (I_m \otimes \tau(t))^\top \eta_{\mathrm{u,ref},j} + u_{\mathrm{ss},j}, \tag{17}$$

$$\Delta_{\mathrm{v},j}(t) := v_j(t) - \left[(I_{n_{\mathrm{v}}} \otimes \tau(t))^\top \eta_{\mathrm{v},j} + v_{\mathrm{ss},j} \right], \tag{18}$$

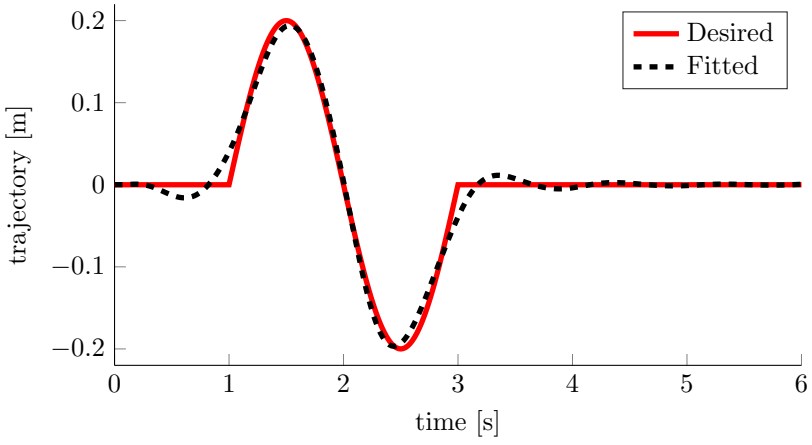
for $t \in [0, T]$. The desired state trajectory at the intervals $j = 0, 1, \ldots, N-1$ is represented



(a) Generation with one interval



(b) Multiple interval trajectories



(c) Generation with multiple intervals

Figure 3. Influence of multiple intervals in the trajectory generation procedure. Because of the limited representation power of the basis functions, it is not possible to fit arbitrary long trajectories, see (a). To overcome this problem, it is proposed to split the trajectory over multiple intervals, with different parametrizations for each of these, see (b). Merging the different intervals' trajectories results in a more accurate fit, as shown in (c).

by $x_j(t) := x_{\text{des}}(t + jT)$, whereas $v_j(t) := v(t + jT)$ represents the disturbances divided over the different intervals. Defining the optimization variables through the following auxiliary quantities,

$$z_{\text{x,ref}} := (\eta_{\text{x,ref},0}, x_{\text{ss},0}, \eta_{\text{x,ref},1}, x_{\text{ss},1}, \dots) \tag{19}$$

$$z_{\text{u,ref}} := (\eta_{\text{u,ref},0}, u_{\text{ss},0}, \eta_{\text{u,ref},1}, u_{\text{ss},1}, \dots) \tag{20}$$

$$z_{\text{v}} := (\eta_{\text{v},0}, v_{\text{ss},0}, \eta_{\text{v},1}, v_{\text{ss},1}, \dots), \tag{21}$$

and eliminating the terms that do not depend on these variables, simplifies (15) to

$$\frac{1}{2}\left[z_{\text{x,ref}}^\top \tilde{Q} z_{\text{x,ref}} + 2f_{\text{x}}^\top z_{\text{x,ref}} + z_{\text{u,ref}}^\top \tilde{R} z_{\text{u,ref}} + z_{\text{v}}^\top \tilde{V} z_{\text{v}} + 2f_{\text{v}}^\top z_{\text{v}}\right], \tag{22}$$

with $\tilde{Q}$, $\tilde{R}$, $\tilde{V}$, f_{x} and f_{v} defined in Appendix A.

In addition, the dynamics need to be fulfilled for the generated trajectories, and the steady-state offsets are required to be equilibria. This is encoded, for $j = 0, \dots, N-1$, as

$$(I_n \otimes M^\top)\eta_{\text{x,ref},j} = (A \otimes I_s)\eta_{\text{x,ref},j} + (B \otimes I_s)\eta_{\text{u,ref},j} + (G \otimes I_s)\eta_{\text{v},j} \tag{23}$$

$$Ax_{\text{ss},j} + Bu_{\text{ss},j} + Gv_{\text{ss},j} = 0. \tag{24}$$

In order to ensure that the state, the input, and the disturbances are continuous over subsequent intervals, the following constraints are added, for $j = 1, \dots, N-1$,

$$(I_n \otimes \tau(0)^\top)\eta_{\text{x,ref},j} + x_{\text{ss},j} - (I_n \otimes \tau(T)^\top)\eta_{\text{x,ref},j-1} - x_{\text{ss},j-1} = 0 \tag{25}$$

$$(I_m \otimes \tau(0)^\top)\eta_{\text{u,ref},j} + u_{\text{ss},j} - (I_m \otimes \tau(T)^\top)\eta_{\text{u,ref},j-1} - u_{\text{ss},j-1} = 0 \tag{26}$$

$$(I_{n_{\text{v}}} \otimes \tau(0)^\top)\eta_{\text{v},j} + v_{\text{ss},j} - (I_{n_{\text{v}}} \otimes \tau(T)^\top)\eta_{\text{v},j-1} - v_{\text{ss},j-1} = 0. \tag{27}$$

The constraints (25)-(27) prevent discontinuities in the actuation when the trajectory tracking controller switches reference parameters among consecutive intervals. In case the disturbances stem from the system's nonlinearities, (27) encodes the natural fact that the disturbances are continuous. Without loss of generality, the trajectories are enforced to start at zero (other cases can be dealt with in a similar way, up to a translation) with the following constraints,

$$(I_n \otimes \tau(0)^\top)\eta_{\text{x,ref},0} + x_{\text{ss},0} = 0 \tag{28}$$

$$(I_m \otimes \tau(0)^\top)\eta_{\text{u,ref},0} + u_{\text{ss},0} = 0 \tag{29}$$

$$(I_{n_{\text{v}}} \otimes \tau(0)^\top)\eta_{\text{v},0} + v_{\text{ss},0} = 0. \tag{30}$$

Finally, the state and input constraints are imposed, yielding for $j = 0, \ldots, N-1$,

$$b_{\min} \leq F_{\mathrm{x}}\eta_{\mathrm{x,ref},j} + D_{\mathrm{x}}x_{\mathrm{ss},j} + F_{\mathrm{u}}\eta_{\mathrm{u,ref},j} + D_{\mathrm{u}}u_{\mathrm{ss},j} \leq b_{\max}. \tag{31}$$

Defining,

$$z := \left(z_{\mathrm{x,ref}}, z_{\mathrm{u,ref}}, z_{\mathrm{v}}\right), \tag{32}$$

the complete optimization problem that is solved to generate the reference set of parameters and steady state offsets is rearranged as,

$$\inf_{z} \quad \frac{1}{2}z^{\top}Hz + f^{\top}z \tag{33}$$

$$\text{s.t. } A_{\mathrm{eq}}z = 0 \tag{34}$$

$$z_{\min} \leq A_{\mathrm{in}}z \leq z_{\max},$$

with H, f defined in Appendix A, and where A_{eq}, A_{in}, $z_{\min}$ and $z_{\max}$ are defined through appropriate concatenations of the constraints (23)-(31). As mentioned in the previous subsection, in contrast to the work of Sferrazza et al. [23], the steady-state offsets are included among the decision variables.

In the authors' experience, using an active-set solver [24], the optimization problem (33) can be solved more efficiently when the equality constraints are eliminated. This is done by means of QR factorization of A_{eq}, as,

$$A_{\mathrm{eq}} = P_{\mathrm{A}}R_{\mathrm{A}}^{\top}Q_{\mathrm{A}}^{\top} = P_{\mathrm{A}}\begin{bmatrix}R_{\mathrm{A},1}^{\top} & \mathbf{0}\end{bmatrix}\begin{bmatrix}Q_{\mathrm{A},1}^{\top} \\ Q_{\mathrm{A},2}^{\top}\end{bmatrix} = P_{\mathrm{A}}R_{\mathrm{A},1}^{\top}Q_{\mathrm{A},1}^{\top}, \tag{35}$$

where Q_{A}, $Q_{\mathrm{A},1}$ and $Q_{\mathrm{A},2}$ are orthogonal matrices, R_{A} and $R_{\mathrm{A},1}$ are upper triangular matrices, P_{A} is a permutation matrix such that the magnitude of the elements on the main diagonal of R_{A} is decreasing, and $\mathbf{0}$ is the zero matrix of appropriate dimensions. From the orthogonality of Q_{A}, it follows that by replacing,

$$z = Q_{\mathrm{A},2}\hat{z}, \tag{36}$$

where $\hat{z}$ is an arbitrary vector of appropriate dimension, (34) is satisfied,

$$A_{\mathrm{eq}}z = P_{\mathrm{A}}R_{\mathrm{A},1}^{\top}\underbrace{Q_{\mathrm{A},1}^{\top}Q_{\mathrm{A},2}}_{=0}\hat{z} = 0. \tag{37}$$

Therefore, according to (36), the optimization problem in (33) can be reformulated as,

$$\begin{aligned} \inf_{\hat{z}} \quad & \frac{1}{2}\hat{z}^\top Q_{\mathrm{A},2}^\top H Q_{\mathrm{A},2}\hat{z} + f^\top Q_{\mathrm{A},2}\hat{z} \\ \text{s.t. } & z_{\min} \leq A_{\mathrm{in}} Q_{\mathrm{A},2}\hat{z} \leq z_{\max}. \end{aligned} \tag{38}$$

The problem in (38) has generally considerably less optimization variables and constraints than the one in (33), and it exhibited smaller solution times in the setup used by the authors.

2.3 Kalman filter

The repeatable disturbances of the system are estimated with a Kalman filter, and are provided at each trial to the optimization problem described in Subection 2.2. The filter aims at fusing prior and measurement information. Its state is denoted by $v[k]$, for $k = 0, 1, \ldots$, where the square brackets indicate the discrete-time indexing, $v[k] := v(kT_\mathrm{s})$, with T_s being the sampling time.

The process model is chosen to express the expected repeatability of the estimated disturbances over subsequent trials as,

$$v[k]^i = v[k]^{i-1} + n[k]^i, \tag{39}$$

where i is the current trial, and the process noise $n[k]^i$ is assumed to be zero-mean Gaussian, with variance Q_{KF}. Therefore, the Kalman filter's prior update is,

$$\hat{v}_\mathrm{p}[k]^i = \hat{v}_\mathrm{m}[k]^{i-1} \tag{40}$$

$$P_\mathrm{p}[k]^i = P_\mathrm{m}[k]^{i-1} + Q_{\mathrm{KF}}, \tag{41}$$

where $\hat{v}_\mathrm{p}^i[k]$ and $P_\mathrm{p}[k]^i$ express the current belief (that is the expected value and variance) of $v[k]^i$, conditioned on the measurements up to the $(i-1)$-th trial, and $\hat{v}_\mathrm{m}^i[k]$ and $P_\mathrm{m}[k]^i$ express the current belief (again, mean and variance) of $v[k]^i$, given the measurements up to the i-th trial. The information provided by the deviations from the system's dynamics at each trial of the trajectory tracking are incorporated in the measurement model of the Kalman filter. Given the discretized dynamics,

$$x[k+1] = A_\mathrm{d} x[k] + B_\mathrm{d} u[k] + G_\mathrm{d} v[k], \tag{42}$$

where A_d, B_d and G_d are the discrete-time system's matrices, the actual state and input trajectories, respectively $x[k]^i$ and $u[k]^i$, are recorded during the i-th trial, and used to

compute these deviations as,

$$G_{\mathrm{d}} v[k]^i = \underbrace{x[k+1]^i - A_{\mathrm{d}} x[k]^i - B_{\mathrm{d}} u[k]^i}_{:=z[k]^i}$$

In order to increase the signal-to-noise ratio, the disturbance vector is held constant for N_{KF} discrete time steps. In this way, at each trial, N_{KF} measurements subsequently contribute to update a single disturbance estimate value. Assuming that $z[k]^i$ is corrupted by zero-mean Gaussian noise with variance R_{KF}, the measurement update is performed as described in Algorithm 2. The current disturbance estimate is transformed to a continuous time signal $v(t)$ through a zero-order hold, and is used as an input to the trajectory generation problem, as described in Subsection 2.2. In contrast to the work of Sferrazza et al. [23], the parametrized approximation describing $v(t)$ is obtained by optimizing (15), providing a trade-off between the placement of the various offsets and the fit of the reference trajectories.

Algorithm 2 Pseudo-code for measurement update.

Initialize: $k = 0$
while *k is smaller than the trajectory length* **do**
  Initialize: $\hat{v}_{\mathrm{m}}[k]^i = \hat{v}_{\mathrm{p}}[k]^i, P_{\mathrm{m}}[k]^i = P_{\mathrm{p}}[k]^i$
  for $l \in \{0, 1, \ldots, N_{KF} - 1\}$ **do**
    `// Apply the measurements` $z[k+l]^i$ `in sequential order`
    $P_{\mathrm{m}}[k]^i = \left((P_{\mathrm{m}}[k]^i)^{-1} + G_{\mathrm{d}}^\top R_{\mathrm{KF}}^{-1} G_{\mathrm{d}}\right)^{-1}$
    Update $\hat{v}_{\mathrm{m}}[k]^i$ with $\hat{v}_{\mathrm{m}}[k]^i + P_{\mathrm{m}}[k]^i G_{\mathrm{d}}^\top R_{\mathrm{KF}}^{-1}(z[k+l]^i - G_{\mathrm{d}} \hat{v}_{\mathrm{m}}[k]^i)$
  end
  $\hat{v}_{\mathrm{m}}[k + N_{\mathrm{KF}} - 1]^i = \hat{v}_{\mathrm{m}}[k + N_{\mathrm{KF}} - 2]^i = \cdots = \hat{v}_{\mathrm{m}}[k]^i$; `// The disturbance vector is held constant`
  $P_{\mathrm{m}}[k + N_{\mathrm{KF}} - 1]^i = P_{\mathrm{m}}[k + N_{\mathrm{KF}} - 2]^i = \cdots = P_{\mathrm{m}}[k]^i$; `// for` $N_{\mathtt{KF}}$ `discrete time steps`
  $k = k + N_{\mathrm{KF}}$
end

3. Memory considerations

The parametrization of the system's states, inputs and disturbances is particularly advantageous when applied to trajectory tracking. As a matter of fact, the reference trajectories need to be stored in the memory and loaded during the task execution, as is the case with a standard approach, see for example the work of Kühne et al. [25]. When these trajectories are represented in discrete time, a reference needs to be available for all the quantities of interest at each sampling step. Therefore, the space requirements amount

to,

$$c \cdot (n + m + n_{\mathrm{v}}) \frac{t_{\mathrm{f}}}{T_{\mathrm{s}}}, \tag{43}$$

where c is the space needed to store a floating point number, t_{f} is the trajectory duration, and T_{s} is the sampling time. Conversely, by using the parametrization introduced in Subsection 2.1, a set of parameters and steady-state offsets represents the reference trajectory at each interval. Therefore the space requirements are,

$$c \cdot (n + m + n_{\mathrm{v}})(s + 1) \frac{t_{\mathrm{f}}}{T}, \tag{44}$$

where T is the length of the intervals the trajectory is divided into. Space requirements are reduced through the parametrization when,

$$T > (s + 1) T_{\mathrm{s}}. \tag{45}$$

Introducing a space-efficiency coefficient,

$$\nu_{\mathrm{s}} = \frac{T}{(s + 1) T_{\mathrm{s}}}, \tag{46}$$

the condition in (45) can be expressed as,

$$\nu_{\mathrm{s}} > 1. \tag{47}$$

In the application presented in Section 4, a space-efficiency coefficient of about 8.33 is achieved, meaning that almost an order of magnitude less memory is required compared to a standard approach. Note that this also greatly affects the dimension of the trajectory generation problem. As pointed out in the book by Bentley [26], space efficiency often has positive effects on run time. In fact, a smaller program is likely to fit into faster levels of the memory hierarchy [27], which generally have limited size. These effects are even more relevant when a collection of several trajectories need to be available on embedded systems with relatively small random access memory. In this case, the accessing times stemming from the need to load the trajectory data from disk storage can be reduced by a compact representation of the reference trajectories, as described in this article.

4. Experimental results

The proposed approach is evaluated on a quadcopter, with the task of balancing a pole while flying a predefined trajectory. This section presents the setup and the experimental results.

4.1 Hardware

The experiments are conducted on a quadcopter which uses the frame, motors and motor controllers of an AscTec Hummingbird. The quadcopter is equipped with custom electronics and a circular plate that serves as a base for the pole. The physical parameters of the experimental setup are summarized in Table 1. An onboard controller, running at 1 kHz on a PX4FMU flight controller, tracks the desired angular rates and the desired total thrust, which are commanded by the parametrized MPC controller. The latter runs on a standard desktop computer that communicates with the quadcopter's flight controller through wireless communication and achieves a sampling time of 0.02 s. The experiments were conducted in the Flying Machine Arena [28], where an external motion tracking system provides measurements of the system's state.

Symbol	Value	Description
m_q	0.5 Kg	mass of the quadrotor
d_q	0.17 m	length of one quadrotor arm
m_p	0.038 Kg	mass of the pole
d_p	0.584 m	half length of the pole
f_min	0.616 N	minimum rotor thrust
f_max	3.862 N	maximum rotor thrust
$\alpha_\mathrm{rel,max}$	30°	maximum relative roll angle
$\beta_\mathrm{rel,max}$	30°	maximum relative pitch angle

Table 1. Physical parameters.

4.2 Model

The quadcopter's state is represented by its position, that is, r_x, r_y and r_z, the corresponding linear velocities v_x, v_y and v_z, and the xyz-Euler angles, that is, α, β and γ. The pole is represented by its roll and pitch angles, respectively α_p and β_p, and the corresponding angular velocities $\omega_{\mathrm{p},\alpha}$ and $\omega_{\mathrm{p},\beta}$. The pole is assumed to be very thin and its rotation about its axis of symmetry is not modeled. The quadcopter's angular rates ω_α, ω_β and ω_γ, and the difference between the total thrust f_tot and the hovering thrust $f_{\mathrm{tot},0}$ are regarded as the control inputs for the MPC controller. These are then sent to the onboard controller, where they are tracked with a very high bandwidth [28]. Summarizing,

the complete state and input vectors are,

$$x = (r_{\mathrm{x}}, r_{\mathrm{y}}, r_{\mathrm{z}}, v_{\mathrm{x}}, v_{\mathrm{y}}, v_{\mathrm{z}}, \alpha, \beta, \gamma, \alpha_{\mathrm{p}}, \beta_{\mathrm{p}}, \omega_{\mathrm{p},\alpha}, \omega_{\mathrm{p},\beta}), \tag{48}$$

$$u = (\omega_\alpha, \omega_\beta, \omega_\gamma, f_{\mathrm{tot}} - f_{\mathrm{tot},0})\,. \tag{49}$$

The system's matrices are derived from a simple first principles model (see Appendix B for a detailed derivation), which considers the quadrocopter and the pole to be rigid bodies. Friction between the pendulum and the quadrocopter, drag of the pendulum and the quadrocopter, as well as any other aerodynamic forces, are neglected (the propellers are assumed to provide a certain thrust aligned with the symmetry axis of the quadrocopter). The nonlinear model is linearized about hover, which yields

$$A = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & k_1 & 0 & 0 & -k_2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -k_1 & 0 & 0 & k_2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -k_3 & 0 & 0 & k_3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -k_3 & 0 & 0 & k_3 & 0 & 0
\end{bmatrix}, B = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{m_{\mathrm{q}}+m_{\mathrm{p}}} \\
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}, \tag{50}$$

where k_1, k_2 and k_3 are constants (see Appendix B), m_{q} is the mass of the quadcopter, and m_{p} is the mass of the pole. The matrix G, which models how the disturbances affect the system's dynamics, is chosen as the identity matrix of dimension n_{v}.

Actuation constraints are imposed on the thrust f_l of a single rotor, that is, for $l = 0, 1, 2, 3$,

$$f_{\min} \leq f_l \leq f_{\max}. \tag{51}$$

State constraints on the relative angles between the pole and the quadcopter ensure sufficient contact between the two bodies, and are expressed as,

$$|\alpha - \alpha_{\mathrm{p}}| \leq \alpha_{\mathrm{rel,max}} \tag{52}$$

$$|\beta - \beta_{\mathrm{p}}| \leq \beta_{\mathrm{rel,max}}. \tag{53}$$

These constraints can be reduced to the form in (14), as shown in Appendix C.

4.3 Results

The tuning matrices for the trajectory tracking are chosen as follows,

$$Q = \operatorname{diag}\big(\overbrace{10,\ 10,\ 10,}^{\text{position}}\ \overbrace{1,\ 1,\ 1,}^{\text{lin. velocity}}\ \overbrace{1,\ 1,\ 10,}^{\text{attitude}}\ \overbrace{0.3,\ 0.3,}^{\text{pend. angles}}\ \overbrace{0.15,\ 0.15}^{\text{pend. ang. vel.}}\big)$$
$$R = \operatorname{diag}\big(\underbrace{0.05,\ 0.05,\ 0.5,}_{\omega_\alpha,\omega_\beta,\omega_\gamma}\ \underbrace{0.015}_{f_{\text{tot}}}\big).$$

The matrices for the trajectory generation are,

$$\bar{Q} = 0.01 \operatorname{diag}\big(\overbrace{10^6,\ 10^6,\ 10^6,}^{\text{position}}\ \overbrace{1,\ 1,\ 1,}^{\text{lin. velocity}}\ \overbrace{1,\ 1,\ 10^3,}^{\text{attitude}}\ \overbrace{10^{-4},\ 10^{-4},\ 1,\ 1}^{\text{pend. angles \& ang. vel.}}\big)$$
$$\bar{R} = \operatorname{diag}\big(\underbrace{1,\ 1,\ 1,}_{\omega_\alpha,\omega_\beta,\omega_\gamma}\ \underbrace{0.01}_{f_{\text{tot}}}\big),$$
$$\bar{V} = 10^4 \operatorname{diag}\big(300,\ 300,\ 100,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 100,\ 100,\ 1,\ 1\big)$$

The desired trajectory $x_{\text{des}}(t)$ is only provided for r_{x}, r_{y} and r_{z}, and is equal to zero for all the other quantities, therefore leaving the solver the flexibility to choose feasible trajectories for these states. The matrices Q, R, $\bar{Q}$, $\bar{R}$ and $\bar{V}$ are chosen to be positive definite matrices, which implies that the trajectory tracking problem in (8) and the trajectory generation problem in (33) are strictly convex quadratic programs. This ensures convergence of both programs to a feasible solution (if a solution exists) when solved with an active set method [29].

A "prerun" and a "postrun" trajectory's portions are generally added at the start and end of the trajectory, see Figure 3, by padding the desired trajectory with the initial and final state, respectively, for an appropriate time. Both the prerun and the postrun ensure a smoother transition with the hovering phase, which precedes and follows each execution. In addition, the postrun portion serves as a penalty for steering the system to the final state (note that the trajectory generation does not impose a hard constraint on the final state).

The Kalman filter's matrices used to estimate the system's disturbances are,

$$Q_{\text{KF}} = 10^{-2} \operatorname{diag}\big(1,\ 1,\ 1,\ 10,\ 10,\ 10,\ 0.01,\ 0.01,\ 1,\ 1,\ 1,\ 1,\ 1\big)$$
$$R_{\text{KF}} = 10^{-4} \operatorname{diag}\big(1,\ 1,\ 1,\ 0.1,\ 0.1,\ 0.1,\ 100,\ 100,\ 1,\ 1,\ 1,\ 1,\ 1\big).$$

The Kalman filter's state is initialized with zero mean and the following variance,

$$P_0 = 10^{-2} I_{n_{\text{v}}}. \tag{54}$$

The remaining scalar parameters are summarized in Table 2.

Symbol	Value	Description
T_s	0.02 s	sampling time
n_v	13	dimension of the disturbance vector
s	5	number of basis functions
λ	4 s^{-1}	basis function decay
T	1 s	interval length
N_KF	10	Kalman filter parameter
r	0.1	regularization parameter

Table 2. Tuning parameters.

Both the trajectory generation and the online trajectory tracking are solved using qpOASES [24], which implements an active set strategy. The trajectory generation program is solved on average in 250 ms, while the trajectory tracking controller runs at 50 Hz.

A 12 s trajectory representing a three-dimensional eight is encoded in $x_\mathrm{des}(t)$, and the parametrization obtained through the trajectory generation procedure provides a feasible approximation, as shown in Figure 4. As a result, the space-efficiency coefficient introduced in (46) is equal to,

$$\nu_\mathrm{s} \approx 8.33. \tag{55}$$

This trajectory is repeatedly tracked by the parametrized MPC controller introduced in Subsection 2.1, and the disturbances are updated after each trial, as explained in Subsection 2.3. The resulting tracking performance is shown in Figure 6 and Figure 7[2]. The framework presented in this article builds upon the scenario that trajectories are repetitively tracked, and that the system's disturbances can be captured by the basis functions used for the parametrization. In fact, the iterative learning algorithm especially needs to compensate for tracking errors when the quadcopter tilts and changes direction, that is, when the model linearized about hover is not very accurate. The fact that the system improves its tracking performance in these regions over subsequent trials shows that this framework is able to account for unmodeled nonlinearities, provided that the above assumptions are met. Compared to the heuristic approach of Sferrazza et al. [23], the use of steady-state offsets enables the proposed strategy to cope with trajectories and disturbances with larger magnitude (even at the end of an interval), as shown in the example in Appendix D. Together with the fact that a different system is used for the experiments, this facilitates the tracking of considerably more aggressive trajectories.

The disturbance estimate shows a higher variance towards the end of the trajectory as a consequence of the particular choice of modeling the system's disturbance as time-

[2]A video showing an experiment can be found at the following link: `https://youtu.be/-E4znjVDCyA`

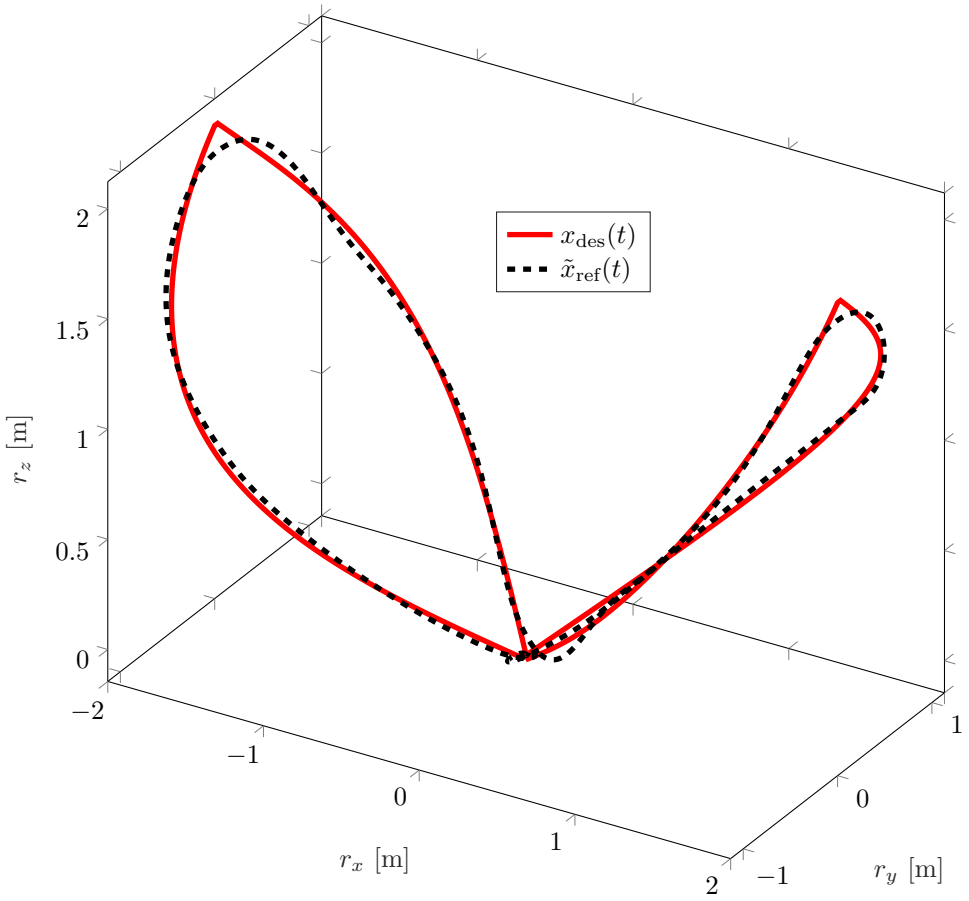


Figure 4. Result of the trajectory generation procedure. A three-dimensional figure-eight (in solid red), with sharp edges, is approximated by a feasible trajectory (in dashed black) that is parametrized with basis functions and steady-state offsets.

varying trajectories. In fact, an incorrect disturbance estimate in an earlier part of the trajectory propagates in time affecting the later parts (it might even drive the system to a state where the current disturbance estimate is not a valid approximation). However, after multiple trials, the disturbance estimates converge, see for example Figure 8, which alleviates this effect.

The quadratic cost for the different trials, defined through the matrices Q and R, is shown in Figure 5. Due to the fact that the disturbances are not modeled as a function of the state, they are subject to change when the system explores different regions in the state space, i.e. during the first trials, when the algorithm has not yet converged. Therefore, the procedure might first exhibit an increase in the cost, which corresponds to this initial exploration phase, before reaching an overall lower steady-state cost. However, the aforementioned choice of modeling the disturbances as time-varying trajectories has the benefit of not increasing the complexity of the online trajectory tracking MPC controller.

In the experiment presented, the algorithm only requires 5 trials to reach the steady-state cost, and it exhibits little fluctuation in the remaining executions. Although in this example the system always executed the same task under the same conditions, the

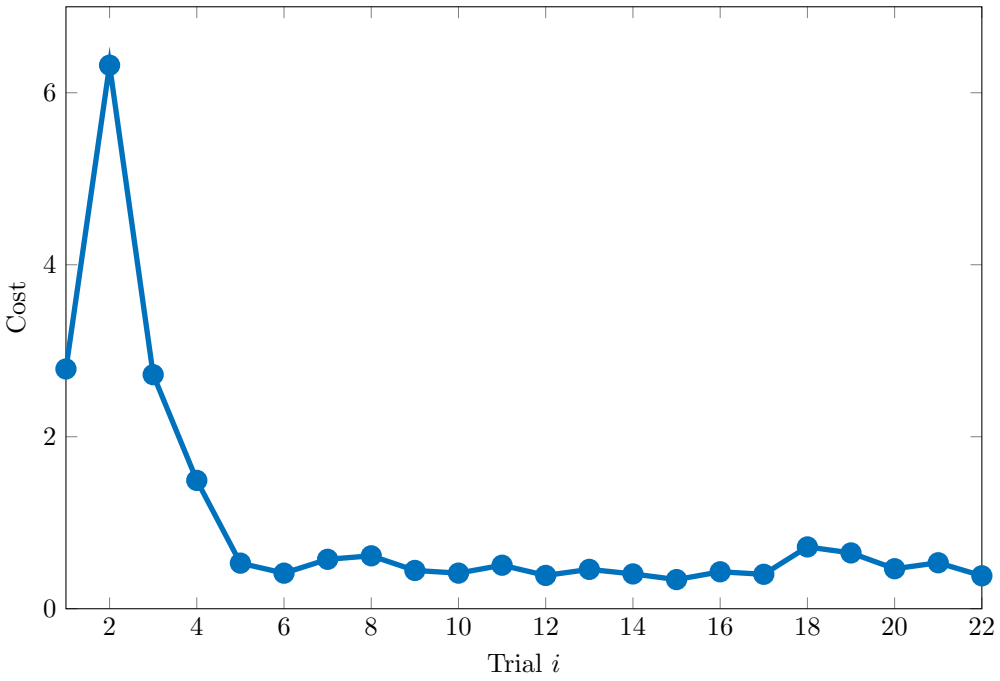


Figure 5. This plot shows the resulting trajectory tracking cost over the 22 trials conducted. The proposed method estimates the repeatable disturbances over the entire trajectory. After an initial exploration phase, the cost converges towards a value that is about three times lower than the initial one.

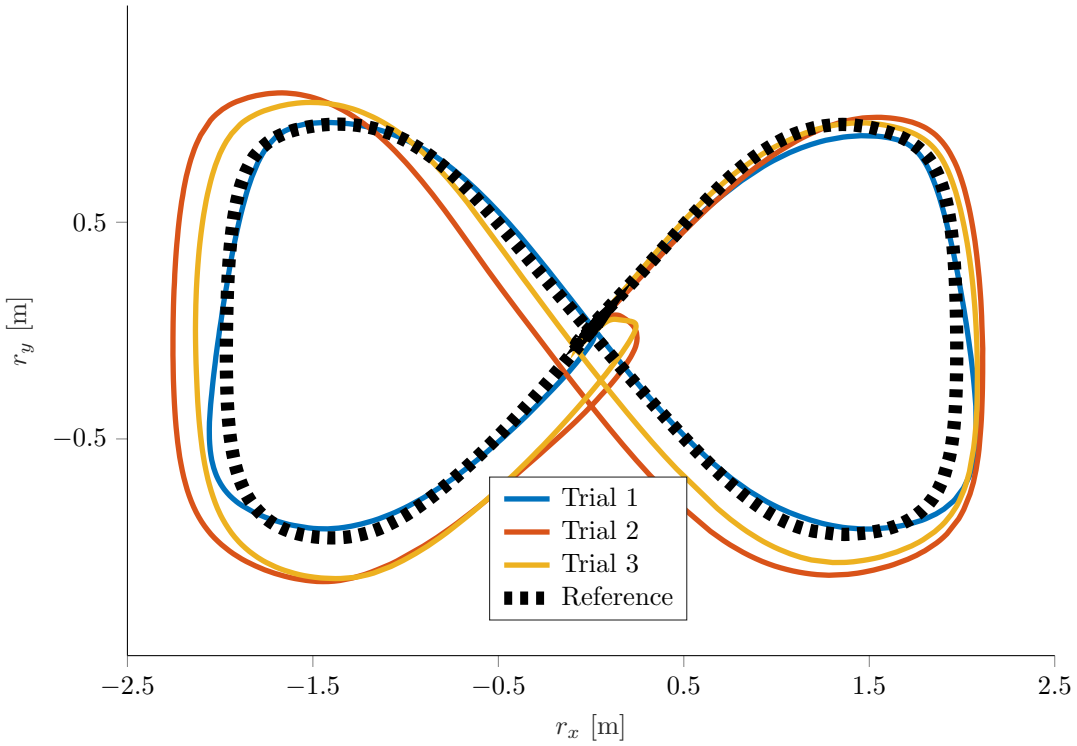
fast learning rate and the fact that the learning algorithm keeps running after each trial (i.e. no early stopping is required) enables the possibility of responsively adapting the disturbance estimate to new situations, e.g. if the quadcopter is subject to a different load.

5. Conclusion

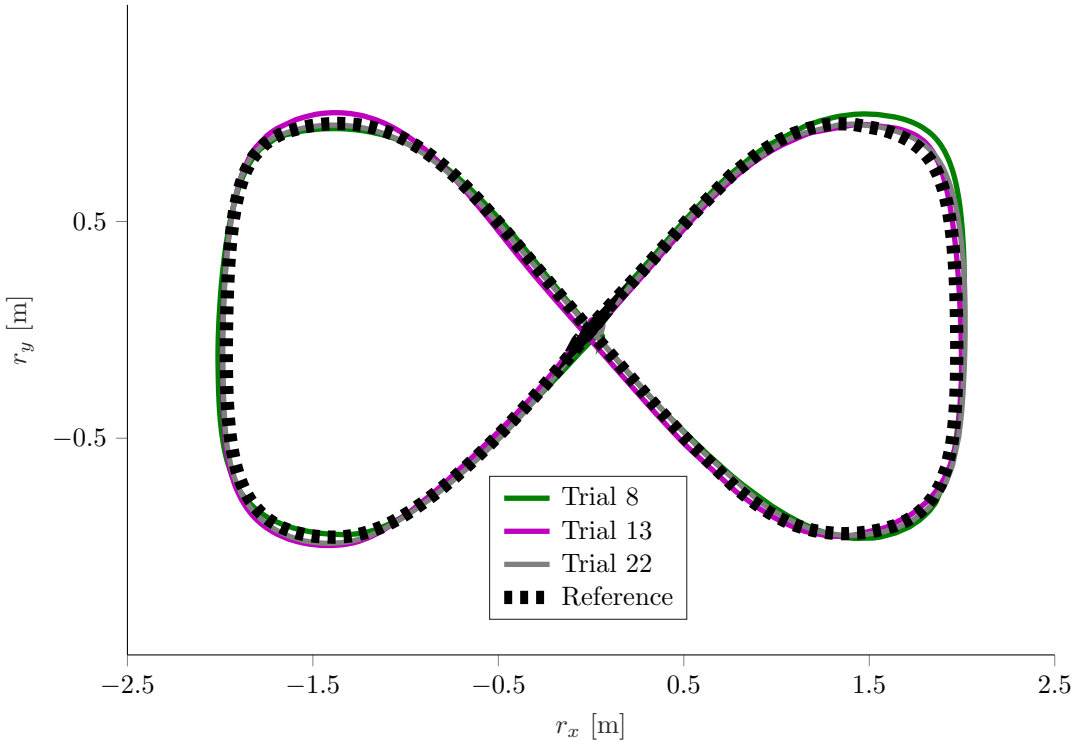
This article presented a method that enables the use of parametrized model predictive control for trajectory tracking. A systematic framework that can capture and compensate for unmodeled dynamics has been developed. Experimental results indicate that 1) the tracking performance indeed improves when executing the same experiment, and 2) the resulting control scheme brings advantages in terms of computation and storage compared to a traditional MPC approach.

A. Mathematical derivation of the parametrized MPC matrices

In the following, the cost function (15) is rearranged by splitting the integral and expand-

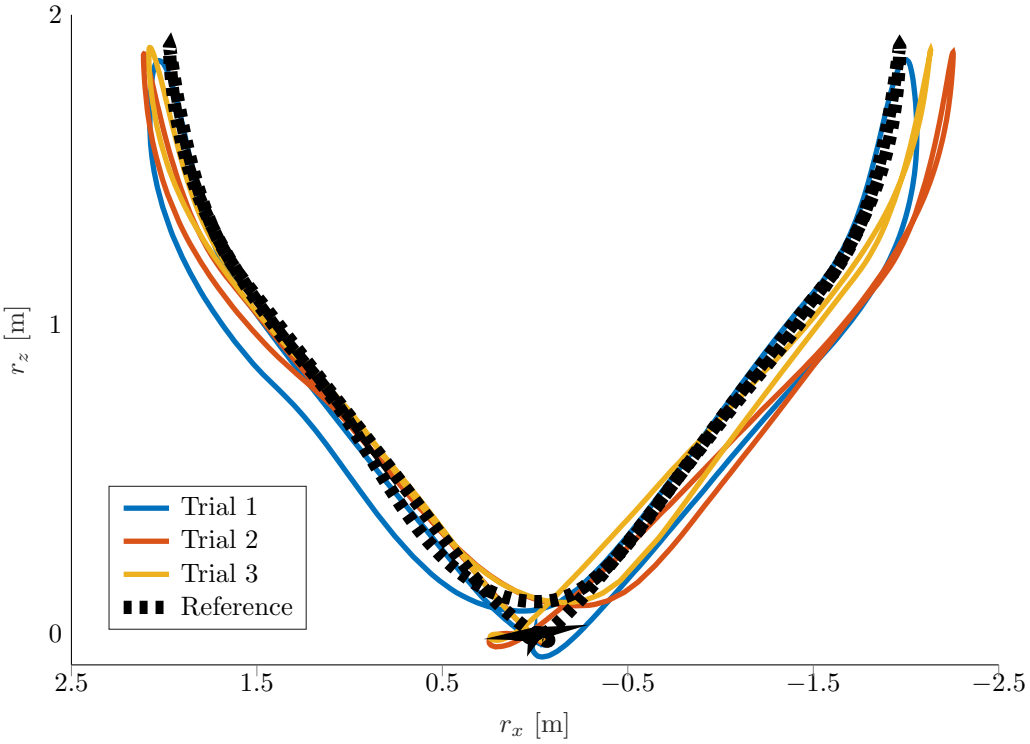


(a) First trials

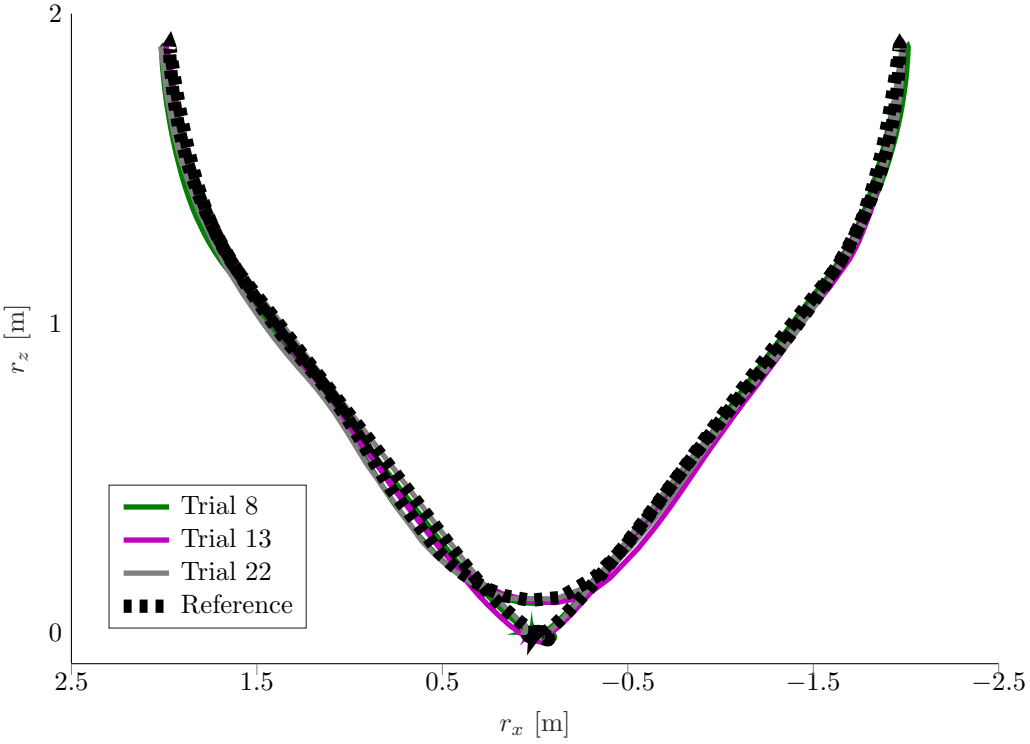


(b) Selected trials

Figure 6. Trajectory tracking performance, top view. The feasible reference trajectory (in dashed black) is tracked for 22 subsequent trials. The disturbance estimate improves over the trials, yielding better tracking performance. The first three trials are shown in (a), while some of the other trials are shown in (b).

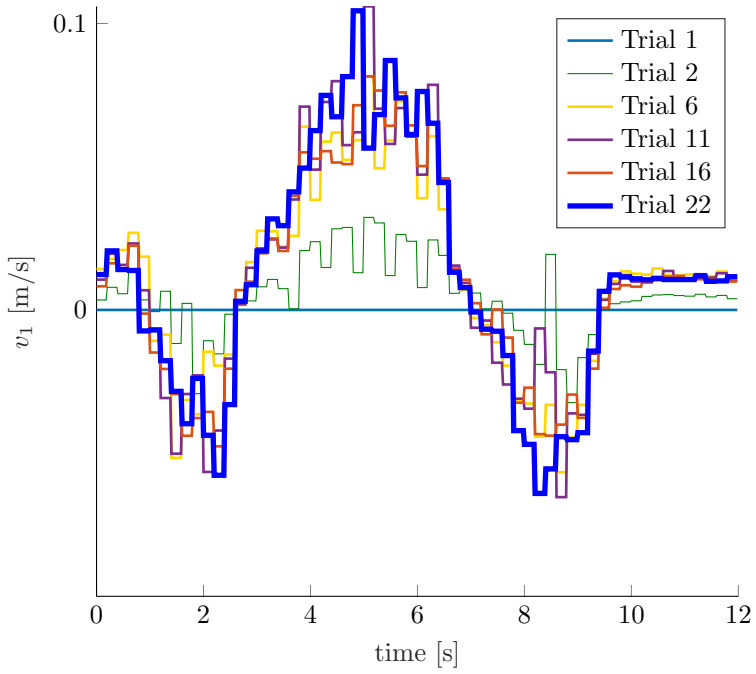


(a) First trials

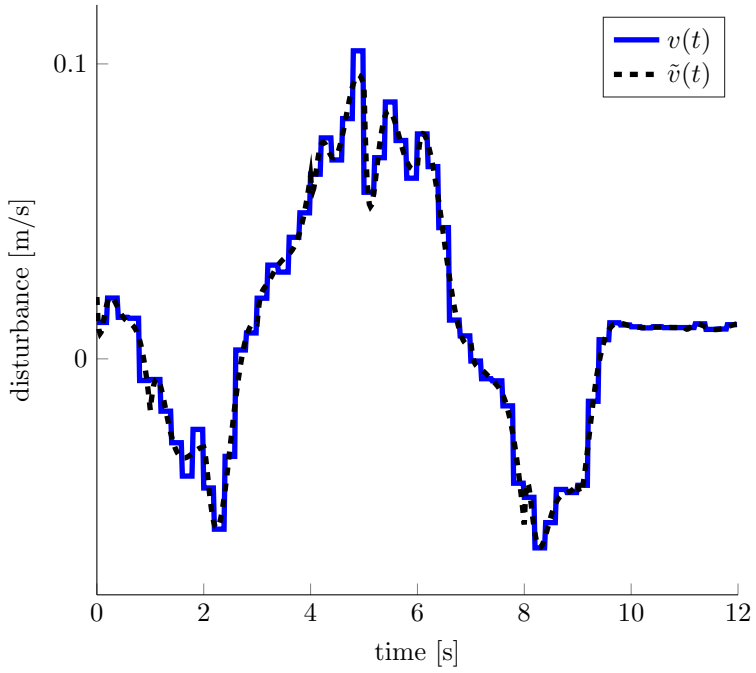


(b) Selected trials

Figure 7. Trajectory tracking performance, side view. The first three trials are shown in (a), while some of the other trials are shown in (b).



(a) Disturbance estimate over different trials



(b) Last estimate fit

Figure 8. The above plots show how a single disturbance's component is estimated and approximated. In (a), the first disturbance vector's component is shown over different trials. Given the choice of the matrix G, this component is added to the equation of motion that describes the dynamics of the state r_{x}. The plot (a), where the last estimate is shown in blue (thick), indicates convergence of the disturbance trajectory. In (b), the last estimate is compared with the approximation (parametrized by the basis functions) obtained from the trajectory generation procedure.

ing the different terms. The terms regarding the system's state are expanded as,

$$\begin{aligned}\int_0^T \frac{1}{2}\left[\Delta_{\mathrm{x},j}(t)^\top \bar{Q}\Delta_{\mathrm{x},j}(t)\right]\mathrm{d}t = \int_0^T \frac{1}{2}\Big\{ & x_j(t)^\top \bar{Q} x_j(t) \\ & + \left[\left(I_n \otimes \tau(t)\right)^\top \eta_{\mathrm{x,ref},j} + x_{\mathrm{ss},j}\right]^\top \bar{Q}\left[\left(I_n \otimes \tau(t)\right)^\top \eta_{\mathrm{x,ref},j} + x_{\mathrm{ss},j}\right] \\ & - 2x_j(t)^\top \bar{Q}\left[\left(I_n \otimes \tau(t)\right)^\top \eta_{\mathrm{x,ref},j} + x_{\mathrm{ss},j}\right]\Big\}\mathrm{d}t \\ = \int_0^T \frac{1}{2}\Big\{ & x_j(t)^\top \bar{Q} x_j(t) + x_{\mathrm{ss},j}^\top \bar{Q} x_{\mathrm{ss},j} \\ & + \eta_{\mathrm{x,ref},j}^\top \left(I_n \otimes \tau(t)\right) \bar{Q} \left(I_n \otimes \tau(t)\right)^\top \eta_{\mathrm{x,ref},j} - 2x_j(t)^\top \bar{Q} x_{\mathrm{ss},j} \\ & + x_{\mathrm{ss},j}^\top \bar{Q} \left(I_n \otimes \tau(t)\right)^\top \eta_{\mathrm{x,ref},j} + \eta_{\mathrm{x,ref},j}^\top \left(I_n \otimes \tau(t)\right) \bar{Q} x_{\mathrm{ss},j} \\ & - 2x_j(t)^\top \bar{Q} \left(I_n \otimes \tau(t)\right)^\top \eta_{\mathrm{x,ref},j}\Big\}\mathrm{d}t. \end{aligned} \tag{56}$$

Neglecting the first term in the sum, which does not depend on the optimization variables, and defining $z_{\mathrm{x,ref},j} := (\eta_{\mathrm{x,ref},j}, x_{\mathrm{ss},j})$, (56) can be rewritten as,

$$\frac{1}{2}\left[z_{\mathrm{x,ref},j}^\top \tilde{Q}_j z_{\mathrm{x,ref},j} + 2f_{\mathrm{x},j}^\top z_{\mathrm{x,ref},j}\right], \tag{57}$$

with,

$$\begin{aligned}\tilde{Q}_j &= \int_0^T \begin{bmatrix} \left(I_n \otimes \tau(t)\right) \bar{Q} \left(I_n \otimes \tau(t)\right)^\top & \left(I_n \otimes \tau(t)\right) \bar{Q} \\ \bar{Q}\left(I_n \otimes \tau(t)\right)^\top & \bar{Q} \end{bmatrix} \mathrm{d}t \\ &= \begin{bmatrix} \bar{Q} \otimes J_s & \int_0^T \left(I_n \otimes \tau(t)\right) \bar{Q}\,\mathrm{d}t \\ \int_0^T \bar{Q}\left(I_n \otimes \tau(t)\right)^\top \mathrm{d}t & \bar{Q}T \end{bmatrix}\end{aligned}$$

$$f_{\mathrm{x},j} = -\int_0^T \begin{bmatrix} \left(I_n \otimes \tau(t)\right) \bar{Q} x_j(t) \\ \bar{Q} x_j(t) \end{bmatrix} \mathrm{d}t, \tag{58}$$

where

$$J_s := \int_0^T \tau(t)\tau(t)^\top \mathrm{d}t, \tag{59}$$

which can be computed as the solution to the following Lyapunov equation, as shown in

the work of Sferrazza et al. [23],

$$MJ_s + J_s M^\top - \left[\tau(T)\tau(T)^\top - \tau(0)\tau(0)^\top\right] = 0. \tag{60}$$

The terms regarding the system's disturbances are expanded in the same way, defining $z_{\mathrm{v},j} := (\eta_{\mathrm{v},j}, v_{\mathrm{ss},j})$, as,

$$\frac{1}{2}\left[z_{\mathrm{v},j}^\top \tilde{V}_j z_{\mathrm{v},j} + 2 f_{\mathrm{v},j}^\top z_{\mathrm{v,ref},j}\right], \tag{61}$$

where

$$\tilde{V}_j = \begin{bmatrix} \bar{V} \otimes J_s & \int_0^T \left(I_{n_\mathrm{v}} \otimes \tau(t)\right) \bar{V}\, \mathrm{d}t \\ \int_0^T \bar{Q}\left(I_{n_\mathrm{v}} \otimes \tau(t)\right)^\top \mathrm{d}t & \bar{V}T \end{bmatrix}$$

$$f_{\mathrm{v},j} = -\int_0^T \begin{bmatrix} \left(I_{n_\mathrm{v}} \otimes \tau(t)\right) \bar{V} v_j(t) \\ \bar{V} v_j(t) \end{bmatrix} \mathrm{d}t. \tag{62}$$

Note that due to Assumption H2 in Subsection 2.1,

$$\dot{\tilde{u}}_{\mathrm{ref},j}(t) = \left(I_m \otimes \dot{\tau}(t)\right)^\top \eta_{\mathrm{u,ref},j} \tag{63}$$

$$= \left(I_m \otimes M\tau(t)\right)^\top \eta_{\mathrm{u,ref},j}, \tag{64}$$

which provides a means to rearrange the input terms as,

$$\begin{aligned}
&\int_0^T \frac{1}{2}\left[r\dot{\tilde{u}}_{\mathrm{ref},j}(t)^\top \dot{\tilde{u}}_{\mathrm{ref},j}(t) + \tilde{u}_{\mathrm{ref},j}(t)^\top \bar{R}\tilde{u}_{\mathrm{ref},j}(t)\right]\mathrm{d}t \\
&= \int_0^T \frac{1}{2}\Big\{ r\eta_{\mathrm{u,ref},j}^\top \left(I_m \otimes M\tau(t)\right)\left(I_m \otimes M\tau(t)\right)^\top \eta_{\mathrm{u,ref},j} \\
&+ \left[\left(I_m \otimes \tau(t)\right)^\top \eta_{\mathrm{u,ref},j} + u_{\mathrm{ss},j}\right]^\top \bar{R}\left[\left(I_m \otimes \tau(t)\right)^\top \eta_{\mathrm{u,ref},j} + u_{\mathrm{ss},j}\right] \Big\} \\
&= \int_0^T \frac{1}{2}\Big\{ r\eta_{\mathrm{u,ref},j}^\top \left(I_m \otimes M\tau(t)\tau(t)^\top M^\top\right)\eta_{\mathrm{u,ref},j} \\
&+ u_{\mathrm{ss},j}^\top \bar{R} u_{\mathrm{ss},j} + \eta_{\mathrm{u,ref},j}^\top \left(I_m \otimes \tau(t)\right)\bar{R}\left(I_m \otimes \tau(t)\right)^\top \eta_{\mathrm{u,ref},j} \\
&+ u_{\mathrm{ss},j}^\top \bar{R}\left(I_m \otimes \tau(t)\right)^\top \eta_{\mathrm{u,ref},j} + \eta_{\mathrm{u,ref},j}^\top \left(I_m \otimes \tau(t)\right)\bar{R} u_{\mathrm{ss},j} \Big\} \mathrm{d}t.
\end{aligned} \tag{65}$$

Defining $z_{\mathrm{u,ref},j} := (\eta_{\mathrm{u,ref},j}, u_{\mathrm{ss},j})$, (65) can be rewritten as,

$$\frac{1}{2} z_{\mathrm{u,ref},j}^{\top} \tilde{R}_j z_{\mathrm{u,ref},j}, \tag{66}$$

with

$$\tilde{R}_j = \begin{bmatrix} \bar{R} \otimes J_s + r I_m \otimes \left(M J_s M^{\top}\right) & \int_0^T \left(I_m \otimes \tau(t)\right) \bar{R} \, \mathrm{d}t \\ \int_0^T \bar{R} \left(I_m \otimes \tau(t)\right)^{\top} \mathrm{d}t & \bar{R} T \end{bmatrix}.$$

Summing (57), (61) and (66) over all the intervals, the cost function (15) reduces to,

$$\frac{1}{2}\left[z_{\mathrm{x,ref}}^{\top} \tilde{Q} z_{\mathrm{x,ref}} + 2 f_{\mathrm{x}}^{\top} z_{\mathrm{x,ref}} + z_{\mathrm{u,ref}}^{\top} \tilde{R} z_{\mathrm{u,ref}} + z_{\mathrm{v}}^{\top} \tilde{V} z_{\mathrm{v}} + 2 f_{\mathrm{v}}^{\top} z_{\mathrm{v}} \right] = \frac{1}{2} z^{\top} H z + f^{\top} z, \tag{67}$$

with z, $z_{\mathrm{x,ref}}$, $z_{\mathrm{u,ref}}$ and z_{v} defined as in (32), (19), (20) and (21), and,

$$\tilde{Q} = \begin{bmatrix} \tilde{Q}_0 & & \\ & \ddots & \\ & & \tilde{Q}_{N-1} \end{bmatrix}, \quad \tilde{R} = \begin{bmatrix} \tilde{R}_0 & & \\ & \ddots & \\ & & \tilde{R}_{N-1} \end{bmatrix}, \quad \tilde{V} = \begin{bmatrix} \tilde{V}_0 & & \\ & \ddots & \\ & & \tilde{V}_{N-1} \end{bmatrix} \tag{68}$$

$$H = \begin{bmatrix} \tilde{Q} & & \\ & \tilde{R} & \\ & & \tilde{V} \end{bmatrix} \tag{69}$$

$$f_{\mathrm{x}} = (f_{\mathrm{x},0}, \cdots, f_{\mathrm{x},N-1}) \tag{70}$$

$$f_{\mathrm{v}} = (f_{\mathrm{v},0}, \cdots, f_{\mathrm{v},N-1}) \tag{71}$$

$$f = (f_{\mathrm{x}}, \mathbf{0}, f_{\mathrm{v}}). \tag{72}$$

B. First principles model

For the purpose of modeling the system's dynamics, the following frames are introduced:

$\{I\}$ Inertial frame,

$\{B\}$ Quadcopter-fixed frame, with its z-axis aligned with the thrust direction (and symmetry axis of the quadcopter),

$\{C\}$ Pole-fixed frame, with its z-axis aligned with the symmetry axis of the pole.

The equations of motion are derived by using the principle of virtual power. This yields

$$m_{\mathrm{q}} {}^{I}\dot{v}_{\mathrm{q}} + m_{\mathrm{p}} {}^{I}\dot{v}_{\mathrm{p}} = R_{\mathrm{IB}} {}^{B} f_{\mathrm{tot}} + (m_{\mathrm{q}} + m_{\mathrm{p}}) {}^{I} g, \tag{73}$$

describing the position of the quadcopter and

$$ {}^{C}\Theta_{\mathrm{p}}\,{}^{C}\dot{\omega}_{\mathrm{IC}} - m_{\mathrm{p}}\,{}^{C}\tilde{r}_{\mathrm{PQ}} R_{\mathrm{IC}}^{\top}\,{}^{I}\dot{v}_{\mathrm{p}} = -{}^{C}\omega_{\mathrm{IC}} \times {}^{C}\Theta_{\mathrm{p}}\,{}^{C}\omega_{\mathrm{IC}} - m_{\mathrm{p}}\,{}^{C}\tilde{r}_{\mathrm{PQ}} R_{\mathrm{IC}}^{\top}\,{}^{I}g, \tag{74} $$

describing the attitude of the pendulum, m_{p} and m_{q} are the masses of the quadcopter and the pole, respectively, v_{p} and v_{q} are the linear velocities of the quadcopter and the pole, $R_{\mathrm{IB}} \in \mathrm{SO}(3)$ is the rotation matrix that transforms vectors from the frame B to the frame I, $R_{\mathrm{IC}} \in \mathrm{SO}(3)$ is the rotation matrix that transforms vectors from the frame C to the frame I, f_{tot} is the total thrust generated by the four rotors, g is the gravity vector, Θ_{p} is the inertia of the pole, r_{PQ} is the distance vector that goes from the pole's center of gravity (positioned at half-length of the pole) to the quadcopter's center of gravity and is expressed in the frame C, and ω_{IC} is the angular velocity vector of the C frame with respect to the inertial frame. Throughout this section, the preceding superscript denotes the frame in which a particular vector or tensor is expressed. Moreover, the "tilde" symbol on top of a vector denotes its corresponding skew symmetric matrix, i.e. $\tilde{a}$ is defined as $a \times b = \tilde{a}b$ for $a \in \mathbb{R}^3$ and for all $b \in \mathbb{R}^3$.

Note that the quadrocopter's angular rates, ${}^{B}\omega_{\mathrm{IB}}$, are considered to be control inputs for the MPC controller (with the vector's components ω_α, ω_β, ω_γ which are tracked with very high bandwidth by the onboard controller). The quadrocopter's angular rates govern the quadrocopter's attitude by

$$ \dot{R}_{\mathrm{IB}} = R_{\mathrm{IB}}\,{}^{B}\tilde{\omega}_{\mathrm{IB}}, \tag{75} $$

and enter the combined quadrocopter and pole dynamics through R_{IB} in (73).

Moreover, the fact that the quadrotor and the pole are connected yields the following constraint (expressed on acceleration level)

$$ {}^{I}\dot{v}_{\mathrm{p}} - {}^{I}\dot{v}_{\mathrm{q}} = -R_{\mathrm{IC}}\,{}^{C}\tilde{\omega}_{\mathrm{IC}}\,{}^{C}\tilde{\omega}_{\mathrm{IC}}\,{}^{C}r_{\mathrm{PQ}} - R_{\mathrm{IC}}\,{}^{C}\dot{\tilde{\omega}}_{\mathrm{IC}}\,{}^{C}r_{\mathrm{PQ}}. \tag{76} $$

Thus, the combination of (73), (74), (75), and (76) describes the entire quadrocopter-pole dynamics.

The algorithms presented in this paper rely on a linear time-invariant system model. To that extent, the nonlinear equations of motion are linearized about hover. The hover equilibrium is given by zero linear velocity and zero angular velocity of the quadcopter, the pole being at rest in the upright position, and the following constant thrust

$$ {}^{B}f_{\mathrm{tot},0} = \left(0, 0, (m_{\mathrm{q}} + m_{\mathrm{q}})g_0\right), \tag{77} $$

with $g_0 := 9.81\ \mathrm{m/s^2}$. Linearizing (76) about hover yields

$$ {}^{I}\dot{v}_{\mathrm{p}} = {}^{I}\dot{v}_{\mathrm{q}} + {}^{C}\tilde{r}_{\mathrm{PQ}}\,{}^{C}\dot{\omega}_{\mathrm{IC}}. \tag{78} $$

Linearizing (73) and including the relation (78) results in

$$(m_\mathrm{q}+m_\mathrm{p})^I\dot{v}_q + m_\mathrm{p}{}^C\tilde{r}_\mathrm{PQ}{}^C\dot{\omega}_\mathrm{IC} \approx g_0(m_\mathrm{q}+m_\mathrm{p})\,(e_\mathrm{x}\beta - e_\mathrm{y}\alpha) + ({}^Bf_\mathrm{tot} - {}^Bf_\mathrm{tot,0}), \tag{79}$$

where e_x, e_y and e_z are the standard unit vectors in $\mathbb{R}^3$, α and β are the quadcopter's roll and pitch angles. Linearizing (74) and including (78) yields

$$-m_\mathrm{p}{}^C\tilde{r}_\mathrm{PQ}{}^I\dot{v}_\mathrm{q} + \left({}^C\Theta_\mathrm{p} - m_\mathrm{p}{}^C\tilde{r}_\mathrm{PQ}{}^C\tilde{r}_\mathrm{PQ}\right){}^C\dot{\omega}_\mathrm{IC} \approx m_\mathrm{p}d_\mathrm{p}g_0\,(e_\mathrm{y}\beta_\mathrm{p} + e_\mathrm{x}\alpha_\mathrm{p})\,, \tag{80}$$

where α_p and β_p are the pole's roll and pitch angles, and d_p is the pole's half-length, i.e. $|r_\mathrm{PQ}| = d_\mathrm{p}$.

The quantities in the equations (79)-(80) relate to the ones in (48)-(49) through the following equalities,

$$^Iv_\mathrm{q} = (v_\mathrm{x}, v_\mathrm{y}, v_\mathrm{z}) \tag{81}$$

$$^C\omega_\mathrm{IC} = (\omega_{\mathrm{p},\alpha}, \omega_{\mathrm{p},\beta}, \omega_{\mathrm{p},\gamma}), \tag{82}$$

where the angular velocity $\omega_{\mathrm{p},\gamma}$ of the pole is not included in the system's state.

For the system on which the experiments are conducted, the pole's inertia matrix has the following form,

$$^C\Theta_\mathrm{p} := \mathrm{diag}\,(I_1, I_1, *)\,, \tag{83}$$

where the inertia about the symmetry axis of the pole is irrelevant, as the corresponding rotations are not included in the system's state. The A and B matrices in (50) are obtained by rearranging (79)-(80), where

$$c_\mathrm{p} := m_\mathrm{p}d_\mathrm{p} \tag{84}$$

$$\hat{I}_1 := I_1 + m_\mathrm{p}d_\mathrm{p}^2 \tag{85}$$

$$k_1 := \frac{\hat{I}_1(m_\mathrm{q}+m_\mathrm{p})g_0}{(m_\mathrm{q}+m_\mathrm{p})\hat{I}_1 - c_\mathrm{p}^2} \tag{86}$$

$$k_2 := \frac{c_\mathrm{p}^2 g_0}{(m_\mathrm{q}+m_\mathrm{p})\hat{I}_1 - c_\mathrm{p}^2} \tag{87}$$

$$k_3 := \frac{c_\mathrm{p}(m_\mathrm{q}+m_\mathrm{p})g_0}{(m_\mathrm{q}+m_\mathrm{p})\hat{I}_1 - c_\mathrm{p}^2}. \tag{88}$$

The parameters used for the experiments presented in this article are summarized in Table 3.

Symbol	Value
c_{p}	0.022 Kg m
I_1	0.008 Kg m^2
$\hat{I}_1$	0.021 Kg m^2
k_1	10.26 m/s^2
k_2	0.45 m/s^2
k_3	10.915 m/s^2

Table 3. Model parameters.

C. System's constraints

The actuation constraints (51) on the rotors need some manipulation to be expressed in the form described by (14). Introducing the vector of ones $\mathbf{1}_4 \in \mathbb{R}^4$, and stacking the different f_l, for $l = 0, 1, 2, 3$, into a vector $\bar{f} \in \mathbb{R}^4$, the constraints on the four rotors can be summarized as,

$$f_{\min}\mathbf{1}_4 \leq \bar{f} \leq f_{\max}\mathbf{1}_4. \tag{89}$$

Defining the auxiliary vector,

$$\hat{u} := (\dot{\omega}_\alpha, \dot{\omega}_\beta, \dot{\omega}_\gamma, f_{\mathrm{tot}}) = \Delta\hat{u} + \hat{u}_0, \tag{90}$$

with

$$\Delta\hat{u} := (\dot{\omega}_\alpha, \dot{\omega}_\beta, \dot{\omega}_\gamma, f_{\mathrm{tot}} - f_{\mathrm{tot},0}) \tag{91}$$

$$\hat{u}_0 := (0, 0, 0, f_{\mathrm{tot},0}), \tag{92}$$

the following relation holds,

$$\hat{u} = F\bar{f}, \tag{93}$$

with,

$$F := \begin{bmatrix} F_0 \\ \mathbf{1}_4^\top \end{bmatrix} \tag{94}$$

$$F_0 := \Theta_{\mathrm{q}}^{-1} \begin{bmatrix} 0 & d_{\mathrm{q}} & 0 & -d_{\mathrm{q}} \\ -d_{\mathrm{q}} & 0 & d_{\mathrm{q}} & 0 \\ k_{\mathrm{f}} & -k_{\mathrm{f}} & k_{\mathrm{f}} & -k_{\mathrm{f}} \end{bmatrix}, \tag{95}$$

where Θ_{q} is quadcopter's inertia matrix, d_{q} is the length of a quadcopter's arm, and k_{f} is a physical constant. For the system on which the experiments are conducted, the inertia matrix has the following shape,

$$\Theta_{\mathrm{q}} = \mathrm{diag}\left(I_{\mathrm{x}}, I_{\mathrm{y}}, I_{\mathrm{z}}\right). \tag{96}$$

The physical parameters introduced in this section are summarized in Table 4.

Symbol	Value	Description
k_{f}	0.016 m	proportional factor
I_{x}	0.003 Kg m^2	moment of inertia around x-axis
I_{y}	0.003 Kg m^2	moment of inertia around y-axis
I_{z}	0.006 Kg m^2	moment of inertia around z-axis

Table 4. Physical parameters.

With the introduced relations, (89) becomes,

$$f_{\min}\mathbf{1}_4 \leq F^{-1}\hat{u} \leq f_{\max}\mathbf{1}_4 \tag{97}$$

$$f_{\min}\mathbf{1}_4 \leq F^{-1}(\Delta\hat{u} + \hat{u}_0) \leq f_{\max}\mathbf{1}_4 \tag{98}$$

$$f_{\min}\mathbf{1}_4 - F^{-1}\hat{u}_0 \leq F^{-1}\Delta\hat{u} \leq f_{\max}\mathbf{1}_4 - F^{-1}\hat{u}_0. \tag{99}$$

Considering the approximation in (3), and the Assumption H2 in Subsection 2.1, (99) can be rearranged as,

$$f_{\min}\mathbf{1}_4 - F^{-1}\hat{u}_0 \leq F^{-1}(I_m \otimes \tau(t))^{\top} \begin{bmatrix} I_{m-1} \otimes M^{\top} & \mathbf{0} \\ \mathbf{0} & I_s \end{bmatrix} \eta_{\mathrm{u}} + F^{-1} \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} u_{\mathrm{ss}} \leq f_{\max}\mathbf{1}_4 - F^{-1}\hat{u}_0, \tag{100}$$

The above semi-infinite constraint is required to hold for all times $t \in [0, \infty)$, and is implemented by sampling at the time instances t_i, for $i = 0, \ldots, s-1$. In the experiments described in this article, $t_0 = 0$ s, $t_1 = 0.0929$ s, $t_2 = 0.3215$ s, $t_3 = 0.7164$ s, $t_4 = 1.3692$ s.

Thus, by defining

$$F_{\mathrm{u},i} := F^{-1}(I_m \otimes \tau(t_i))^\top \begin{bmatrix} I_{m-1} \otimes M^\top & \mathbf{0} \\ \mathbf{0} & I_s \end{bmatrix}, \quad \hat{F}_\mathrm{u} := \begin{bmatrix} F_{\mathrm{u},0} \\ \vdots \\ F_{\mathrm{u},s-1} \end{bmatrix}$$

$$D_{\mathrm{u},i} := F^{-1} \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}, \quad \hat{D}_\mathrm{u} := \begin{bmatrix} D_{\mathrm{u},0} \\ \vdots \\ D_{\mathrm{u},s-1} \end{bmatrix}$$

$$u_{\min,i} := f_{\min}\mathbf{1}_4 - F^{-1}\hat{u}_0$$

$$u_{\min} := \begin{bmatrix} u_{\min,0} \\ \vdots \\ u_{\min,s-1} \end{bmatrix}$$

$$u_{\max,i} := f_{\max}\mathbf{1}_4 - F^{-1}\hat{u}_0$$

$$u_{\max} := \begin{bmatrix} u_{\max,0} \\ \vdots \\ u_{\max,s-1} \end{bmatrix},$$

the constraint (100) can be approximated as

$$u_{\min} \leq \hat{F}_\mathrm{u}\eta_u + \hat{D}_\mathrm{u}u_\mathrm{ss} \leq u_{\max}.$$

Applying the same sampling strategy to the state constraints (52)-(53), yields

$$x_{\min} \leq \hat{F}_\mathrm{x}\eta_x + \hat{D}_\mathrm{x}x_\mathrm{ss} \leq x_{\max},$$

where the matrices $\hat{F}_\mathrm{x}$ and $\hat{D}_\mathrm{x}$, and the vectors $x_{\min}$ and $x_{\max}$ are defined accordingly. Consequently, the matrices F_x, F_u, D_x, and D_u, and the vectors $b_{\min}$ and $b_{\max}$ in (14) are obtained by appropriate stacking,

$$F_\mathrm{x} = \begin{bmatrix} \mathbf{0} \\ \hat{F}_\mathrm{x} \end{bmatrix}, F_\mathrm{u} = \begin{bmatrix} \hat{F}_\mathrm{u} \\ \mathbf{0} \end{bmatrix}, D_\mathrm{x} = \begin{bmatrix} \mathbf{0} \\ \hat{D}_\mathrm{x} \end{bmatrix}, D_\mathrm{u} = \begin{bmatrix} \hat{D}_\mathrm{u} \\ \mathbf{0} \end{bmatrix}, \tag{101}$$

$$b_{\min} = \begin{bmatrix} u_{\min} \\ x_{\min} \end{bmatrix}, b_{\max} = \begin{bmatrix} u_{\max} \\ x_{\max} \end{bmatrix}. \tag{102}$$

D. Example simulation for comparison to previous work

In Figure 9, simulation results are provided that compare the approach presented here to previous work [23]. The simulations are conducted on an unconstrained double integrator system, that is, $\ddot{x}(t) = u(t) + v$, where $v = 1$ m/s^2 is a constant disturbance. A ramp of 10 m is prescribed as the desired trajectory. The same parameters chosen for the experiment presented in this article are used for this example. The plots show that using the approach of Sferrazza et al. [23], the steady-state disturbance cannot be fully captured by the basis functions for the entire length of an interval. As a consequence, the approximation shows oscillations and discontinuities at the interval boundaries. This inaccurate approximation tends to reduce the smoothness of the trajectory, compared to the desired one (see the right plots in Figure 9). In contrast, the approach presented here yields an approximation that overlaps with the estimated steady-state disturbance and results in significantly smoother trajectories, as shown in the experiments presented in this article. Note that this effect becomes considerably more relevant for the tracking of aggressive trajectories on higher dimensional systems.

Acknowledgments

The experiments presented in this article were carried out in the Flying Machine Arena. A list of present and past participants is available at: `http://flyingmachinearena.org/people/`.

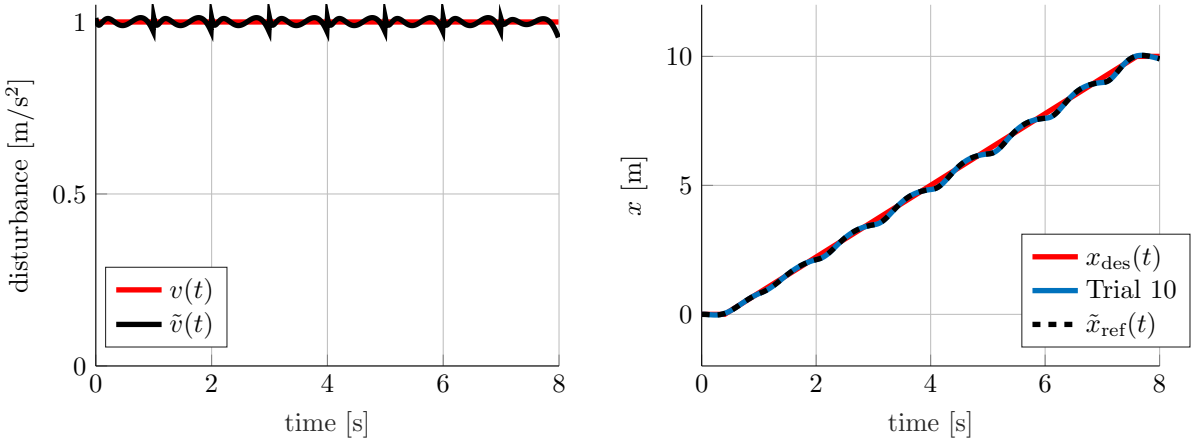
References

[1] J. Richalet, A. Rault, J. Testud, and J. Papon, "Model predictive heuristic control: Applications to industrial processes", *Automatica*, vol. 14, no. 5, pp. 413–428, 1978.

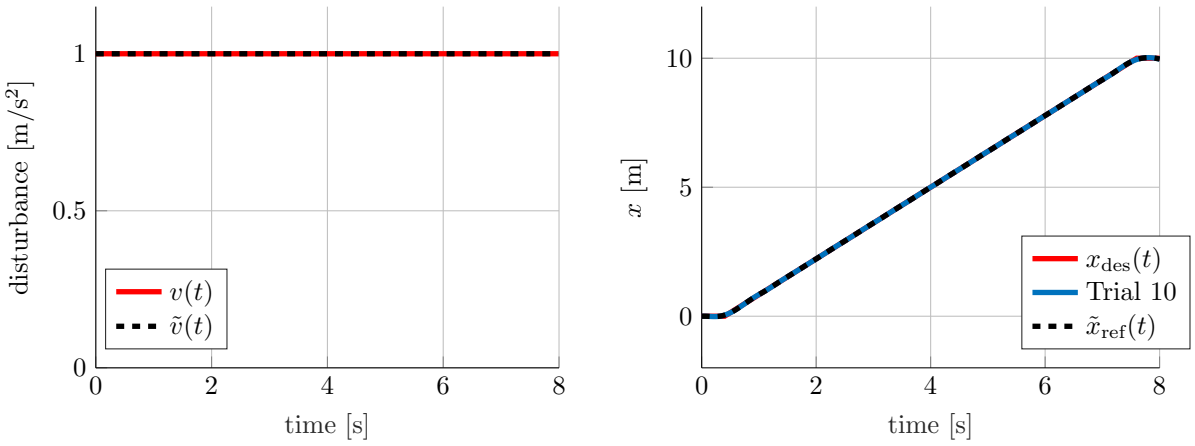
[2] F. Borrelli, P. Falcone, T. Keviczky, J. Asgari, and D. Hrovat, "MPC-based approach to active steering for autonomous vehicle systems", *International Journal of Vehicle Autonomous Systems*, vol. 3, no. 2-4, pp. 265–291, 2005.

[3] T. Geyer, G. Papafotiou, and M. Morari, "Model predictive direct torque control – part I: Concept, algorithm, and analysis", *IEEE Transactions on Industrial Electronics*, vol. 56, no. 6, pp. 1894–1905, 2009.

[4] G. Papafotiou, J. Kley, K. G. Papadopoulos, P. Bohren, and M. Morari, "Model predictive direct torque control – part II: Implementation and experimental evaluation", *IEEE Transactions on Industrial Electronics*, vol. 56, no. 6, pp. 1906–1915, 2009.



(a) Approach used in Sferrazza et al. [23]



(b) Approach used here

Figure 9. The plots on the left compare the disturbance estimation using the approach presented in the work of Sferrazza et al. [23] with the one introduced here. The disturbance captured by the Kalman filter after nine trials (in red) is approximated at each interval with exponentially decaying basis functions (in black). The plots on the right show the tracking performance of the state x at the tenth trial for the respective approaches. The generated trajectory that accounts for the current approximation of the disturbance estimate is shown in black. The actual x trajectory is shown in blue, while the desired ramp trajectory is shown in red. Note that the three trajectories overlap in (b).

[5] M. Morari and J. H. Lee, "Model predictive control: Past, present and future", *Computers & Chemical Engineering*, vol. 23, no. 4, pp. 667–682, 1999.

[6] M. Muehlebach and R. D'Andrea, "Parametrized infinite-horizon model predictive control for linear time-invariant systems with input and state constraints", *Proceedings of the American Control Conference*, pp. 2669–2674, 2016.

[7] W. B. Dunbar, M. B. Milam, R. Franz, and R. M. Murray, "Model predictive control of a thrust-vectored flight control experiment", *Proceedings of the IFAC World Congress*, vol. 35, pp. 355–360, 2002.

[8] R. Gondhalekar and J.-i. Imura, "Least-restrictive move-blocking model predictive control", *Automatica*, vol. 46, no. 7, pp. 1234–1240, 2010.

[9] A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones, "Efficient interior point methods for multistage problems arising in receding horizon control", *Proceedings of the Conference on Decision and Control*, pp. 668–674, 2012.

[10] Y. Wang and S. Boyd, "Fast model predictive control using online optimization", *IEEE Transactions on control systems technology*, vol. 18, no. 2, p. 267, 2010.

[11] L. Wang, "Model predictive control system design and implementation using MATLAB®", *Springer Science & Business Media*, 2009.

[12] J. A. Rossiter and L. Wang, "Exploiting laguerre functions to improve the feasibility/performance compromise in MPC", *Proceedings of the Conference on Decision and Control*, pp. 4737–4742, 2008.

[13] B. Khan and J. A. Rossiter, "Alternative parameterisation within predictive control: A systematic selection", *International Journal of Control*, vol. 86, no. 8, pp. 1397–1409, 2013.

[14] T. Faulwasser and R. Findeisen, "A model predictive control approach to trajectory tracking problems via time-varying level sets of Lyapunov functions", *Proceedings of the Conference on Decision and Control and European Control Conference*, pp. 3381–3386, 2011.

[15] M. A. Stephens, C. Manzie, and M. C. Good, "Model predictive control for reference tracking on an industrial machine tool servo drive", *IEEE Transactions on Industrial Informatics*, vol. 9, no. 2, pp. 808–816, 2013.

[16] M. Neunert, C. de Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear model predictive control for unified trajectory optimization and tracking", *Proceedings of the International Conference on Robotics and Automation*, pp. 1398–1404, 2016.

[17] M. Kamel, M. Burri, and R. Siegwart, "Linear vs nonlinear MPC for trajectory tracking applied to rotary wing micro aerial vehicles", *Proceedings of the IFAC World Congress*, vol. 50, no. 1, pp. 3463–3469, 2017.

[18] M. W. Mueller and R. D'Andrea, "A model predictive controller for quadrocopter state interception", *Proceedings of the European Control Conference*, pp. 1383–1389, 2013.

[19] P. Bouffard, A. Aswani, and C. Tomlin, "Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results", *Proceedings of the International Conference on Robotics and Automation*, pp. 279–284, 2012.

[20] Y. Wang, D. Zhou, and F. Gao, "Iterative learning model predictive control for multi-phase batch processes", *Journal of Process Control*, vol. 18, no. 6, pp. 543–557, 2008.

[21] U. Rosolia and F. Borrelli, "Learning model predictive control for iterative tasks. a data-driven control framework", *IEEE Transactions on Automatic Control*, vol. 63, no. 7, pp. 1883–1896, 2018.

[22] M. Muehlebach, C. Sferrazza, and R. D'Andrea, "Implementation of a parametrized infinite-horizon model predictive control scheme with stability guarantees", *Proceedings of the International Conference on Robotics and Automation*, pp. 2723–2730, 2017.

[23] C. Sferrazza, M. Muehlebach, and R. D'Andrea, "Trajectory tracking and iterative learning on an unmanned aerial vehicle using parametrized model predictive control", *Proceedings of the Conference on Decision and Control*, pp. 5186–5192, 2017.

[24] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming", *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[25] F. Künhe, J. Gomes, and W. Fetter, "Mobile robot trajectory tracking using model predictive control", *Proceedings of the latin-american robotics symposium*, 2005.

[26] J. Bentley, "Programming pearls", *Addison-Wesley Professional*, 2016.

[27] D. A. Patterson and J. L. Hennessy, "Large and fast: Exploiting memory hierarchy. in: Computer organization and design: The hardware/software interface", *Newnes*, 2013.

[28] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. D'Andrea, "A platform for aerial robotics research and demonstration: The flying machine arena", *Mechatronics*, vol. 24, no. 1, pp. 41–54, 2014.

[29] J. Nocedal and S. Wright, "Quadratic programming. in: Numerical optimization", *Springer Science & Business Media*, 2006.

Appendix

Related publications

consisting of publications

[R1] C. Trueeb, C. Sferrazza, and R. D'Andrea, "Towards vision-based robotic skins: a data-driven, multi-camera tactile sensor", in *Proceedings of the IEEE International Conference on Soft Robotics (RoboSoft)*, 2020, pp. 333–338

[R2] T. Bi, C. Sferrazza, and R. D'Andrea, "Zero-shot sim-to-real transfer of tactile control policies for aggressive swing-up manipulation", *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5761–5768, 2021

Paper R1

Towards vision-based robotic skins: a data-driven, multi-camera tactile sensor

Camill Trueeb, Carmelo Sferrazza and Raffaello D'Andrea

Abstract

This paper describes the design of a multi-camera optical tactile sensor that provides information about the contact force distribution applied to its soft surface. This information is contained in the motion of spherical particles spread within the surface, which deforms when subject to force. The small embedded cameras capture images of the different particle patterns that are then mapped to the three-dimensional contact force distribution through a machine learning architecture. The design proposed in this paper exhibits a larger contact surface and a thinner structure than most of the existing camera-based tactile sensors, without the use of additional reflecting components such as mirrors. A modular implementation of the learning architecture is discussed that facilitates the scalability to larger surfaces such as robotic skins.

Published in *Proceedings of the 2020 IEEE International Conference on Soft Robotics.*

1. Introduction

Research in whole-body tactile sensing [1] aims to provide robots with the capability of fully exploiting contact with objects to perform a wide range of tasks. As an example, humans often use both their hands and arms to transport large and heavy boxes, exploiting the feedback from their tactile receptors and the compliance of their soft skin.

The recent advances in computer vision and machine learning have drawn increasing attention towards vision-based tactile sensors, often referred to as optical tactile sensors. These sensors generally employ an optical device to provide high-resolution information about the deformation of their soft surface when subject to external forces. As an example, the motion of spherical particles embedded within a soft, transparent gel is captured by an RGB camera in [2] to render feedback about the force distribution that causes the gel's deformation. A typical drawback of the camera-based approaches is the bulkiness of their main sensing unit. Moreover, the minimum focal distance of commercial cameras usually implies the need for additional space between the camera lens and the soft gel, in which the monitored patterns are embedded, e.g., markers, particles, etc., leading to an additional increase of the overall size. Even in the case of close-focus lenses, placing the soft surface too close to the camera generally leads to a reduced field of view (FOV).

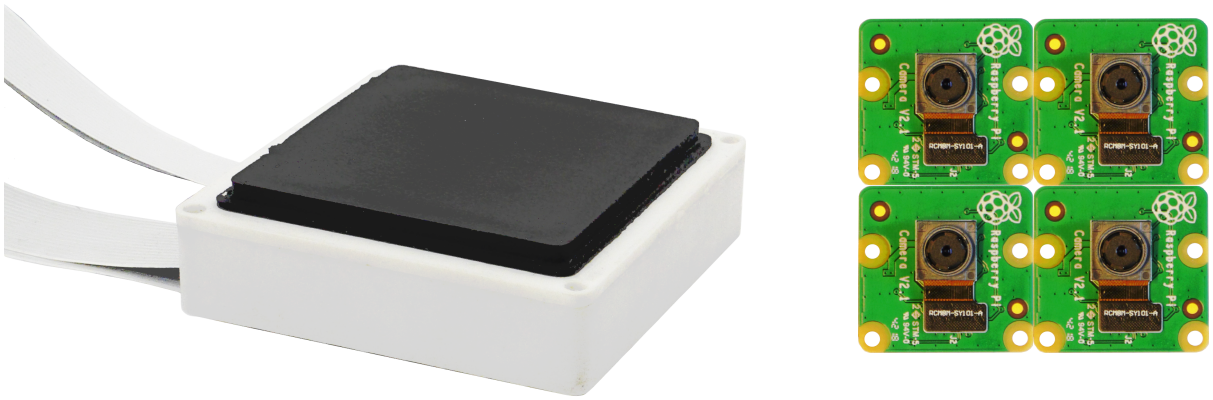


Figure 1. The tactile sensor presented in this article has a reduced thickness compared to most of the camera-based tactile sensors in the literature. In this figure, it is shown next to the four embedded camera modules placed below the soft sensor's surface, which measures 49×51 mm.

This paper proposes a multi-camera design to tackle the issues mentioned, leading to a relatively thin overall structure (about 17.5 mm, see Fig. 1) and retaining the scalability to larger surfaces. Four embedded cameras are equipped with close-focus lenses and are placed next to each other to cover an increased FOV. A deep neural network (DNN) is trained to reconstruct the three-dimensional contact force distribution applied to the sensor's surface, directly processing the pixel intensities captured on the images. The architecture employed here exhibits a modular structure to increase the software scalability

for the implementation on larger surfaces or in the case of a single camera replacement. In fact, a generalization experiment is performed by training the DNN on a subset of the available cameras. A considerably smaller part of the network is then retrained once a new camera is added, resulting in shorter training times and lower data requirements, while generalizing to the entire surface.

The DNN is deployed in real-time, leveraging the capabilities of a state-of-the-art System-on-Module, provided with an integrated GPU. The resulting sensing pipeline predicts the contact force distribution 40 times per second.

1.1 Related work

Several physical principles have been applied with the objective of providing robots with an equivalent of the human sense of touch. In fact, a number of categories of tactile sensors exist in the literature, e.g., resistive [3], piezoelectric [4] and capacitive [5]. A survey of the different categories is provided in [6]. Similarly, various examples of tactile skins using the different sensing principles and scalable to large surfaces have been described, see for example [7], [8].

Vision-based tactile sensors are based on optical devices, which track visual features related to the deformation of a soft surface. Beside RGB cameras, depth cameras [9] and dynamic vision sensors [10] have been employed in a similar manner. Optical tactile sensors show high resolution, ease of manufacture and low cost, despite a larger thickness compared to the other categories. For an overview of the different types of optical tactile sensors, see [11], [12].

In [13], the viability of an optical tactile skin is discussed. The availability of inexpensive and low power GPUs is indicated as a possible solution to enable the real-time processing of a large number of tactile images. Two cameras were mounted on each finger of a soft robotic gripper in [14] to classify the shape and size of an object. The classification is performed by means of a DNN that takes as input the concatenation of the two images. A finger-shaped gripper is presented in [15]. Tactile imprints are redirected via a mirror towards a camera to increase the sensor compactness. Two cameras are used in [16] to reconstruct the 3D displacement of inner markers in a soft tactile muscularis.

In order to overcome the complexity of interpreting the tactile information, several learning-based approaches have been applied to measure various tactile quantities. The location and depth of an indentation are reconstructed in [17] on a sensor based on an array of light emitters and receivers. In [18] a deep learning architecture estimates the total force and torque applied to a tactile sensor, which uses photometric stereo and markers painted on its surface. In [19], a neural network reconstructs the contact force distribution applied to the soft surface of a vision-based sensor. Ground truth labels are provided via the use of simulations based on the finite element method (FEM). In order to share the knowledge acquired from data across different sensors, a transfer learning approach is proposed in [20].

The approach presented here is based on four cameras placed at a short distance from the observed surface, which has a random spread of spherical particles embedded.

The choice of the components and the data-driven approach make it possible to obtain a thin structure without the use of additional reflecting components, hence simplifying manufacture. The network architecture employed is tailored to the use of multiple cameras, introducing modularity features and facilitating the scalability of the approach. The resulting pipeline reconstructs with high accuracy the contact force distribution applied by a sample indenter to the soft surface of the sensor, including the regions where the indentation is not fully covered by the FOV of a single camera.

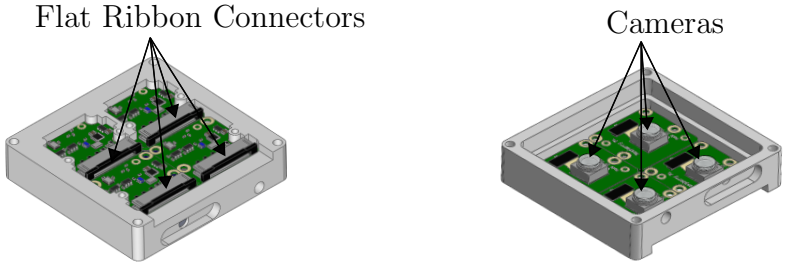
1.2 Outline

The sensing principle and the hardware used for the experiments are described in Section 2. A dimensioning analysis then discusses the possibility of further reducing the thickness of the sensor. The data collection and the learning architecture are detailed in Section 3. The results and a modularity evaluation are presented in Section 4. Remarks in Section 5 conclude the paper.

2. Sensor design

A four-camera design is first introduced in this section. Compared to previous work it shows a thinner sensor with a larger sensing surface. The outlook for a further reduction of the thickness of the design is discussed in the second part of the section.

2.1 Hardware



(a) bottom view **(b)** top view

Figure 2. The sensor's base structure accommodates four Raspberry Pi v2 camera interface boards with flat ribbon cable connectors (a) and the cameras mounted on top (b).

The optical tactile sensor is based on the tracking of unicolored particles (polyethylene microspheres with a diameter of 150 to 180μm) randomly spread within a soft, transparent silicone gel. The motion of the particles is captured by four rectangularly arranged cameras (Raspberry Pi Camera Module v2), see Fig. 2. These cameras capture 40 frames per second at a resolution of 320×240 pixels. The frames are eventually cropped and downsampled to 128×128 pixels. In order to reduce the thickness of the sensor, the default Raspberry Pi camera lenses are replaced by fisheye lenses originally mounted on

Sincerefirst SF-C7251OV-H133 cameras. The lenses are mounted onto the camera frames over distance rings, whose thickness is designed to obtain the desired focus. Finally, an LED board is placed over the camera array to provide uniform brightness.

Similarly to [2], three different silicone layers are cast onto the camera array, as shown in Fig. 3. From the bottom, the first layer is relatively stiff and adds the distance between the camera and the particles, which is necessary to improve the focus. This layer also provides additional protection for the hardware and ensures light diffusion. The second layer is the softest and contains the particles tracked by the cameras. Finally, the third layer (stiffer than the second) is cast with a black color and protects the sensor from external light sources and material damage. The same materials, mixing ratios and curing protocol as in [19] were used, for a resulting sensing surface of 49×51 mm. Each embedded camera

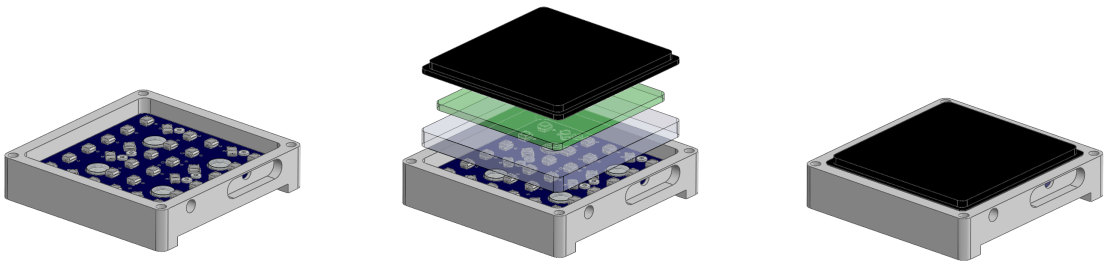


Figure 3. The cameras and an LED board are fixed to a base structure. Three silicone layers are directly poured onto the LED board and the camera lenses: A stiff transparent layer, the particle layer and a black protection layer.

is controlled by a separate, relatively inexpensive single-board computer (Raspberry Pi 3 model B+). These boards communicate with a System-on-Module (NVIDIA Jetson Nano Developer Kit), which is equipped with a 64-bit quad-core Arm Cortex-A57 CPU alongside a Maxwell GPU with 128 CUDA cores. The communication is handled by a Gigabit Ethernet switch (ANDDEAR QZ001), which enables the Jetson Nano to receive the four image streams. The Jetson Nano provides a clock source to the Raspberry Pi boards, which are synchronized through the Networking Time Protocol (NTP), to ensure contemporaneous image streams. Note that the Raspberry Pi boards and the Ethernet switch may be replaced by compact, commercially available multi-camera adapter boards for the Jetson Nano. However, drivers for these adapter boards are still under development or not easily accessible because of the relatively recent release of the Jetson Nano. This aspect has not been further investigated for the purpose of this work.

2.2 Dimensioning analysis

The design presented above exhibits an overall thickness of 17.45 mm, which is lower than most of the camera-based tactile sensors described in the literature. As an example, compared to [15], the sensor described here is slightly thinner and does not use mirrors,

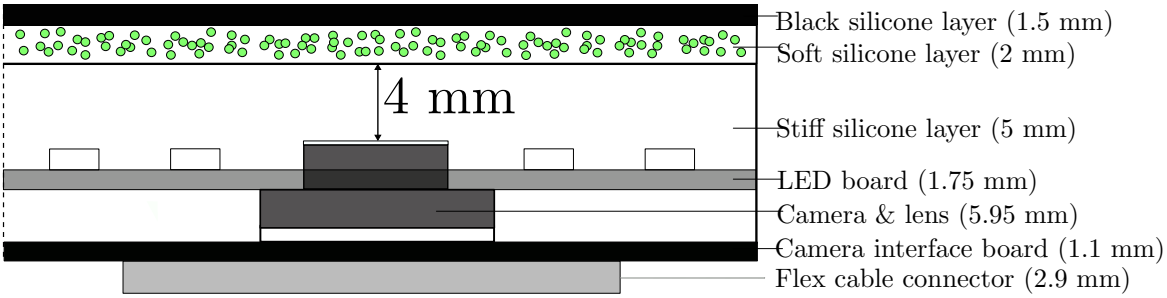


Figure 4. A side-view schematic of the sensor's structure around one of the cameras is shown in this figure. Note that the overall thickness is determined by the two upper silicone layers, the distance between the lens and the particle layer, the camera and the camera interface board, including the connector. The LED board does not contribute to the overall thickness, since it is placed around the camera lenses.

while covering a surface more than six times larger. In the following, some guidelines for further reducing the thickness of the sensor are detailed:

1. The commercial cameras employed in this work mount a flex cable connector at the bottom of their interface board, as shown in Fig. 4. A custom camera interface board, with a connector placed in the space between the interface board itself and the LED board, may reduce the thickness by 2.9 mm, leading to an overall thickness of 14.55 mm.

2. A custom camera interface board may also be placed in a different position, farther from the cameras, depending on the application. Removing the camera boards and their connectors from below the cameras would result in an overall thickness of 13.45 mm.

3. In the current design the interface boards are placed adjacent to each other in the same plane below the cameras. In order to cover a continuous surface, this requires that each camera covers a FOV of at least the size of an interface board. Moving the interface boards (as pointed out in the previous point) may additionally facilitate a closer placement of the cameras. As a consequence, this would make it possible to further reduce the distance between the lenses and the particles, while retaining a continuous surface coverage. Moreover, in this work the fisheye lenses were chosen among the commercially available solutions with a straightforward implementation. A tailored design with an accurate trade-off between the focal distance and the FOV may further reduce the overall thickness. Assuming an ideal pinhole camera model[1], the thickness is mainly limited by the size of the image sensor. Modern image sensors with a thickness of about 0.3 mm are commercially available. The smallest commodity camera module[2] inclusive of a lens has a thickness of 1.158 mm, with the possibility of focusing a surface placed at a distance of 3 mm. Such a design may already result in a tactile sensor thickness of about 5 mm.

[1] A derivation of this fact in a simplified scenario is summarized in the online appendix [21].
[2] `https://www.ovt.com/sensors/OVM6948`

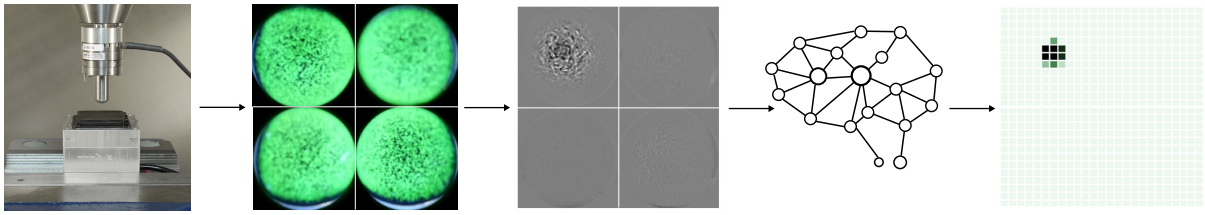


Figure 5. The sensing pipeline is shown in this figure. An indentation produces a change in the particle pattern that is visible in the difference of the pixel intensities (central image) between the current frame and a frame taken at rest. The DNN predicts the three-dimensional contact force distribution applied during the indentation. The last figure on the right shows a color visualization of the resulting F_z for each of the surface bins.

3. Method

In the following section the learning architecture is presented. First, the collection of the ground truth data is explained, then the details regarding the neural network are outlined.

3.1 Data collection

A dataset is collected following the strategy presented in [19]. Automated indentations are performed using a precision milling machine (Fehlmann PICOMAX 56 TOP) with 3-axis computer numerical control (CNC). On an evenly spaced grid, a spherically-ended cylindrical indenter with a diameter of 10 mm is pressed onto the sensor surface at different depths up to 1.5 mm. The same procedure is simulated with a finite element model in Abaqus/Standard [22], to assign ground truth labels to the images, representing the contact force distribution applied to the sensor's surface. In this regard, the surface is discretized into 650 bins of equal area. The procedure described in [19] provides the force applied to these bins, based on the FEM simulations. For each bin, three force components F_x, F_y, F_z are provided, where x and y are aligned along the two horizontal sides of the sensor's surface and centered at one of the corners, and z is the vertical axis, directed from the camera towards the sensor's surface. The resulting label vectors are assigned to the images from the four cameras for each indentation, and used in a supervised learning fashion, as described in the next subsection.

3.2 Learning Architecture

The prediction of the discretized force distribution applied to the surface of the tactile sensor is a multiple multivariate regression problem. This problem is tackled by training an end-to-end DNN that maps the images from the four cameras to the outputs of the network, that is, three force components for each of the 650 surface bins. The intensity difference images between the current frames and the respective four images taken in the undeformed surface state serve as the input to the network. An example of the resulting

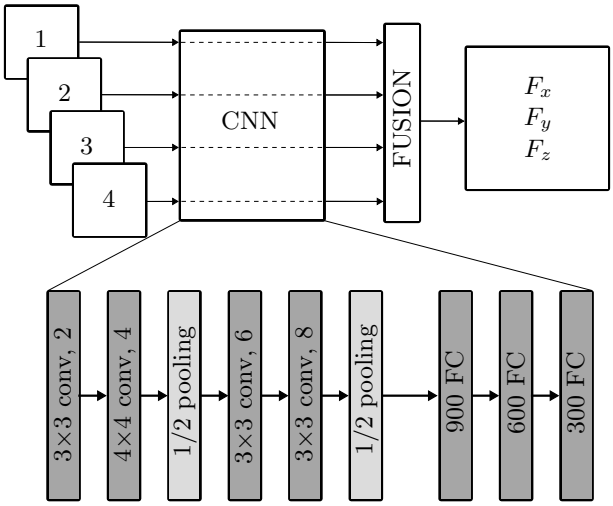


Figure 6. This figure shows the architecture of the network. Each difference image is separately fed through the same CNN, and the outputs are then combined via a fusion layer. For ease of visualization, some abbreviations have been introduced in the block diagram above. In this regard, the label "3×3 conv, 2" refers to a two-channel convolutional layer with a 3×3 kernel, while "1/2 pooling" indicates a max pooling layer, which subsamples the input to half of its original size. "900 FC" refers to a fully connected layer with 900 units.

pipeline is shown in Fig. 5.

To decouple the detection of features that are independent of the cameras' placement, the difference images from the four cameras are fed independently through a convolutional neural network (CNN). Only after this intermediate network, a fusion layer with a linear activation function combines the four different output tensors and predicts the three-dimensional discretized force distribution. Both the overall network layout and the dimensions of the different layers are shown in Fig. 6.

The CNN takes a difference image of size 128×128 as an input. Batch normalization showed a large decrease in the network training times and is used for all convolutional layers, together with rectified linear unit activation functions. A dropout rate of 0.1 is used on the fully connected (FC) layers to prevent overfitting to the training data, together with sigmoid activation functions. The root mean squared error (RMSE) is used as a loss function for the Adam optimizer [23] to train the model. 30% of the dataset was put aside for evaluation purposes.

The fact that the CNN is shared between the four cameras leads to a considerable reduction in the memory consumption, especially in view of the possible extension to larger surfaces with a higher number of cameras. Moreover, this generally leads to a smaller network size, in contrast to feeding the concatenation of the four images through a larger architecture. Smaller architectures tend to require less training data and exhibit shorter training times. Finally, the architecture presented here shows modularity features. These are discussed in Section 4.

4. Results

In this section, the performance of the multi-camera tactile sensor is evaluated. In the first part of the section, the evaluation of the learning architecture is presented. In the second part, an experiment is performed by modifying the neural network and the training procedure in order to test the modularity of the approach.

4.1 Sensor Performance

The DNN presented in Section 3.2 is trained on 70% of the full dataset. 10% of the samples are used as a validation set to apply early stopping during training. The remaining 20% is left aside for evaluation. The resulting RMSE on the force distribution is 0.00060 N, 0.00059 N, 0.0019 N for F_x, F_y, F_z, respectively, while the resulting RMSE on the total applied force (sum of the forces of all surface bins) is 0.0019 N, 0.0016 N, 0.0571 N for F_x, F_y, F_z, respectively. Note that the dataset is generated from vertical indentations, with the resulting total forces in z-direction up to 3 N, and considerably smaller total forces in the horizontal direction on most of the sensor surface. Fig. 7 shows an example prediction of the contact force distribution in z-direction and the corresponding ground truth. The model inference runs on the Jetson Nano at 86 Hz, which makes the frame capturing (40 frames per second) on the Raspberry Pi boards the bottleneck of the sensing speed. As a result, the sensing pipeline runs at a maximal prediction speed of 40 Hz. Furthermore, the fact that the four simultaneous images are fed independently through the CNN enables the detection of multiple contact points, when each camera fully captures up to one of the distinct contact patches, even if the model has only been trained with single indentations. This makes it possible to detect up to four distinct contact patches, as shown in the video[3] attached to this submission.

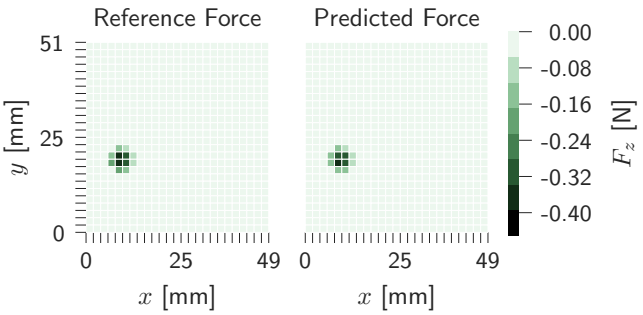


Figure 7. The figure shows the component F_z of each surface bin for an example indentation in the test set. On the left, the ground truth force distribution is shown, and on the right, the distribution predicted by the neural network.

4.2 Sensor Modularity

To evaluate the modularity of the approach, a first model is trained using only images and labels from three cameras. In a second step, the sensor is recalibrated with the training

[3]Video: `https://youtu.be/lbavqAlKl98`

data from all four cameras. The procedure is schematically shown in Fig. 8. For the calibration step, the majority of the DNN parameters are frozen, and only the last fully connected layer and the fusion layer are retrained. This serves the purpose of reducing the training times and the data requirements.

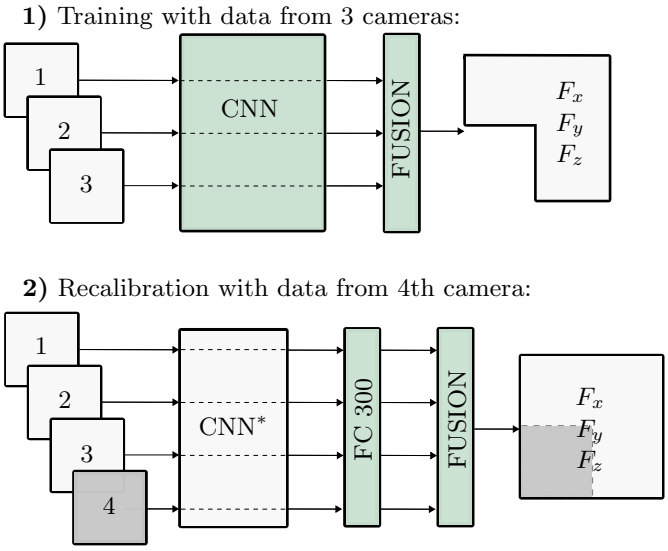
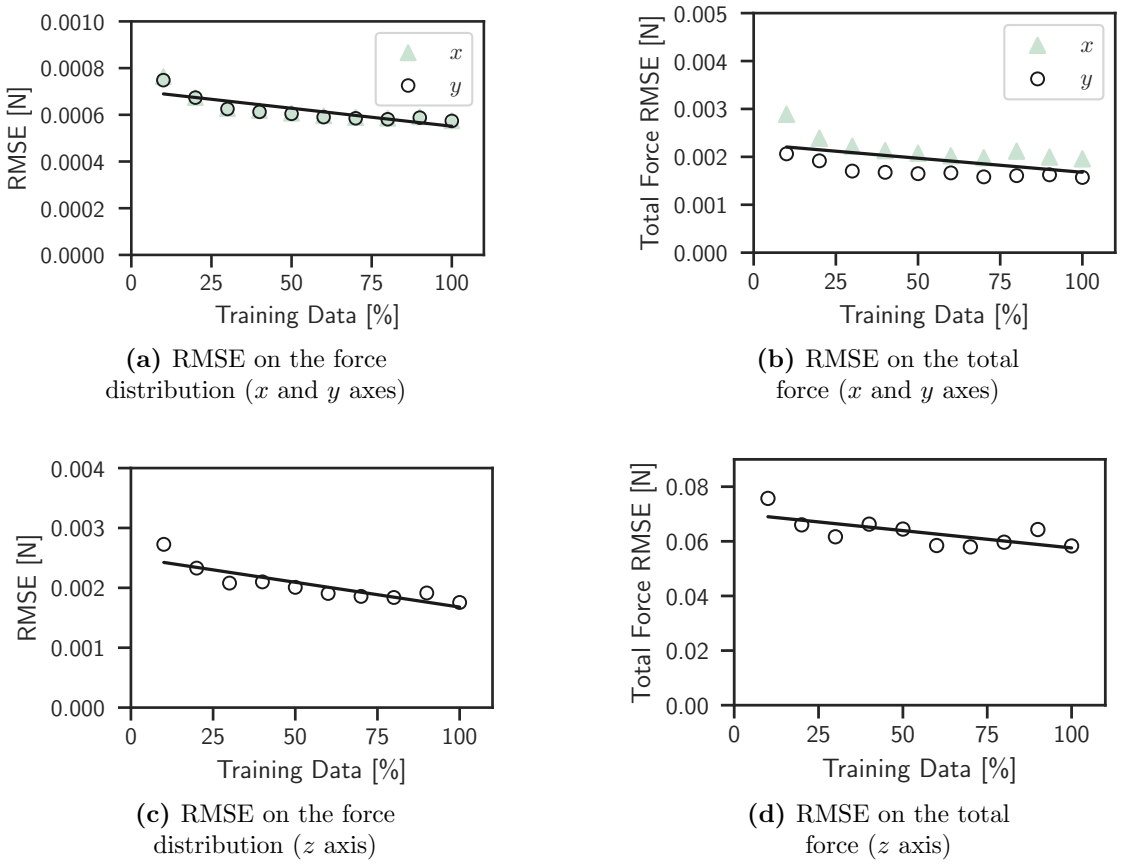


Figure 8. The model is first trained with the data from three cameras, and then extended to four cameras and recalibrated with the full dataset. In the first plot the fusion layer contains 900 units, mapping the outputs corresponding to the surface covering only three cameras. In the second plot, the size of the fusion layer is increased to 1200 units, to cover the entire surface.

As shown in Fig. 9, the recalibrated network shows comparable performance to the model trained on the whole data in Section 4.1. Moreover, the performance is retained using an even smaller portion of training data. The plots show the different error metrics as a function of the percentage of data used for training. Retraining the last two layers takes approximately 1.5 hours on the employed GPU (Nvidia TITAN X Pascal), as opposed to over 10 hours training for the whole model on the full dataset. This experiment shows promising results towards the possibility of training the most resource (both time and data) consuming part of the network on a subset of the surface, therefore reducing the data collection and the training times on the rest of the surface. This also opens the opportunity of replacing a defective camera (although not currently possible in the experimental prototype presented) on a large-scale skin, without the need to retrain the entire network.

5. Conclusion

In this paper, a multi-camera tactile sensing approach has been presented, making use of an end-to-end DNN to predict the force distribution on the sensor surface. The sensor



(a) RMSE on the force distribution (x and y axes)

(b) RMSE on the total force (x and y axes)

(c) RMSE on the force distribution (z axis)

(d) RMSE on the total force (z axis)

Figure 9. After the neural network is trained on the data from three cameras, it is recalibrated for the fourth camera with a portion of the training data from all the cameras. The resulting errors are shown above, as a function of the varying percentage of the full dataset used for training, together with a least-squares trend line.

thickness has been reduced, while at the same time the sensing surface has been extended using multiple cameras arranged in an array. A relatively inexpensive sensor design has been suggested, using Raspberry Pi cameras to capture synchronized images and a Jetson Nano Developer Kit for the model inference.

A neural network has been presented to reconstruct the contact force distribution. The modular architecture proposed here is applicable to optical tactile sensors with a larger numbers of cameras, such as vision-based robotic skins. It has been shown how the network can be efficiently recalibrated when the number of cameras is increased, without retraining any of the convolutional layers. The sensing pipeline presented here runs on an embedded computer provided with a GPU at 40 Hz. On the test dataset employed, the architecture has shown an RMSE of 0.0571 N on the total forces in the vertical direction that were collected up to 3 N.

The procedure proposed here to attach the lenses to the camera frames does not yield a very accurate focus of the images (see Fig. 5) and needs further investigation. Future work will also include the extension of this approach to various directions and shapes of the indentations, as well as a quantitative scalability analysis in the case of an increasing

number of cameras.

Acknowledgments

The authors would like to thank Michael Egli and Matthias Müller for their support in the sensor manufacture.

References

[1] V. J. Lumelsky, *Sensing, Intelligence, Motion: How Robots and Humans Move in an Unstructured World*. John Wiley & Sons, 2005.

[2] C. Sferrazza and R. D'Andrea, "Design, Motivation and Evaluation of a Full-Resolution Optical Tactile Sensor", *Sensors*, vol. 19, no. 4: 928, 2019.

[3] K. Weiß and H. Wörn, "The Working Principle of Resistive Tactile Sensor Cells", in *IEEE International Conference Mechatronics and Automation, 2005*, vol. 1, pp. 471–476.

[4] C. Li, P.-M. Wu, S. Lee, A. Gorton, M. J. Schulz, and C. H. Ahn, "Flexible Dome and Bump Shape Piezoelectric Tactile Sensors Using PVDF-TrFE Copolymer", *Journal of Microelectromechanical Systems*, vol. 17, no. 2, pp. 334–341, 2008.

[5] J. M. Romano, K. Hsiao, G. Niemeyer, S. Chitta, and K. J. Kuchenbecker, "Human-inspired Robotic Grasp Control with Tactile Sensing", *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1067–1079, 2011.

[6] R. S. Dahiya, G. Metta, M. Valle, and G. Sandini, "Tactile Sensing—From Humans to Humanoids", *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 1–20, 2009.

[7] H. Lee, K. Park, J. Kim, and K. J. Kuchenbecker, "A Large-Scale Fabric-Based Tactile Sensor Using Electrical Resistance Tomography", in *Haptic Interaction*, H. Kajimoto, D. Lee, S.-Y. Kim, M. Konyo, and K.-U. Kyung, Eds., Springer Singapore, 2019, pp. 107–109.

[8] G. Cannata, M. Maggiali, G. Metta, and G. Sandini, "An embedded artificial skin for humanoid robots", in *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 434–438.

[9] A. Alspach, K. Hashimoto, N. Kuppuswarny, and R. Tedrake, "Soft-bubble: A highly compliant dense geometry tactile sensor for robot manipulation", in *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, pp. 597–604.

[10] F. B. Naeini, A. Alali, R. Al-Husari, A. Rigi, M. K. AlSharman, D. Makris, and Y. Zweiri, "A Novel Dynamic-Vision-Based Approach for Tactile Sensing Applications", *IEEE Transactions on Instrumentation and Measurement*, 2019.

[11] K. Shimonomura, "Tactile Image Sensors Employing Camera: A Review", *Sensors*, vol. 19, no. 18, p. 3933, 2019.

[12] A. Yamaguchi and C. G. Atkeson, "Recent progress in tactile sensing and sensors for robotic manipulation: can we turn tactile sensing into vision?", *Advanced Robotics*, vol. 33, no. 14, pp. 661–673, 2019.

[13] A. Yamaguchi and C. G. Atkeson, "Optical Skin For Robots: Tactile Sensing And Whole-Body Vision", in *Workshop on Tactile Sensing for Manipulation, Robotics: Science and Systems (RSS)*, vol. 25, 2017, pp. 133–134.

[14] Y. She, S. Q. Liu, P. Yu, and E. Adelson, *Exoskeleton-covered soft finger with vision-based proprioception and exteroception*, 2019. arXiv: 1910.01287 [cs.RO].

[15] E. Donlon, S. Dong, M. Liu, J. Li, E. Adelson, and A. Rodriguez, "GelSlim: A High-Resolution, Compact, Robust, and Calibrated Tactile-sensing Finger", in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1927–1934.

[16] L. Van Duong, R. Asahina, J. Wang, *et al.*, "Development of a Vision-Based Soft Tactile Muscularis", in *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, pp. 343–348.

[17] P. Piacenza, E. Hannigan, C. Baumgart, Y. Xiao, S. Park, K. Behrman, W. Dang, J. Espinal, I. Hussain, I. Kymissis, and M. Ciocarlie, "Touch Sensors with Overlapping Signals: Concept Investigation on Planar Sensors with Resistive or Optical Transduction", *CoRR*, vol. abs/1802.08209, 2019. arXiv: 1802.08209. [Online]. Available: http://arxiv.org/abs/1802.08209.

[18] W. Yuan, S. Dong, and E. Adelson, "Gelsight: High-Resolution Robot Tactile Sensors for Estimating Geometry and Force", *Sensors*, vol. 17, no. 12, p. 2762, 2017.

[19] C. Sferrazza, A. Wahlsten, C. Trueeb, and R. D'Andrea, *Ground truth force distribution for learning-based tactile sensing: a finite element approach*, 2019. arXiv: 1909.04000 [cs.RO].

[20] C. Sferrazza and R. D'Andrea, "Transfer learning for vision-based tactile sensing", *CoRR*, vol. abs/1812.03163, 2019. arXiv: 1812.03163. [Online]. Available: http://arxiv.org/abs/1812.03163.

[21] C. Trueeb, C. Sferrazza, and R. D'Andrea, "Online appendix - On the thickness of vision-based tactile sensors", 2019. [Online]. Available: https://arxiv.org/src/1910.14526/anc/OnlineAppendix.pdf.

[22] Dassault Systèmes, *Abaqus/Standard User's Manual, v. 6.14*, English, Simulia, 2014.

[23] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization", in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

Paper R2

Zero-shot sim-to-real transfer of tactile control policies for aggressive swing-up manipulation

Thomas Bi, Carmelo Sferrazza and Raffaello D'Andrea

Abstract

This paper aims to show that robots equipped with a vision-based tactile sensor can perform dynamic manipulation tasks without prior knowledge of all the physical attributes of the objects to be manipulated. For this purpose, a robotic system is presented that is able to swing up poles of different masses, radii and lengths, to an angle of 180°, while relying solely on the feedback provided by the tactile sensor. This is achieved by developing a novel simulator that accurately models the interaction of a pole with the soft sensor. A feedback policy that is conditioned on a sensory observation history, and which has no prior knowledge of the physical features of the pole, is then learned in the aforementioned simulation. When evaluated on the physical system, the policy is able to swing up a wide range of poles that differ significantly in their physical attributes without further adaptation. To the authors' knowledge, this is the first work where a feedback policy from high-dimensional tactile observations is used to control the swing-up manipulation of poles in closed-loop.

Published in *IEEE Robotics and Automation Letters.*

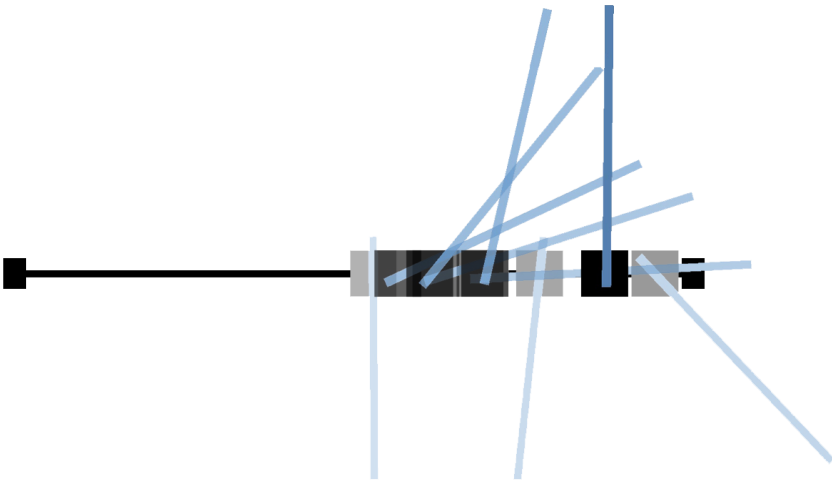
1. Introduction

Tactile sensors aim to provide robots with a sense of touch that captures information from their environment through physical contact. In this paper, the vision-based tactile sensor presented in [1] is deployed in order to demonstrate that it can provide robots with a dexterity akin to that of humans in dynamic manipulation tasks. For this purpose, a robotic system that performs swing-up maneuvers for different poles is presented (see Fig. 1). The robotic system consists of a parallel gripper, mounted to a linear motor, with two tactile sensors acting as fingers. Thereby, three key capabilities enabled by the artificial sense of touch provided by the tactile sensor are demonstrated: (i) The system is able to adapt its motion and successfully swings up poles that differ in their physical attributes (e.g. mass, length, and radius) without prior knowledge of these attributes. (ii) The system does not rely on external visual sensing; instead, the pose and attributes of the pole in contact are implicitly inferred from the tactile observations alone. (iii) The tactile observations can be processed in real-time and act as feedback for closed-loop control at 60 Hz. As a result, highly dynamic swing-up manipulations are achieved without the need for a previous in-hand exploration of the pole.

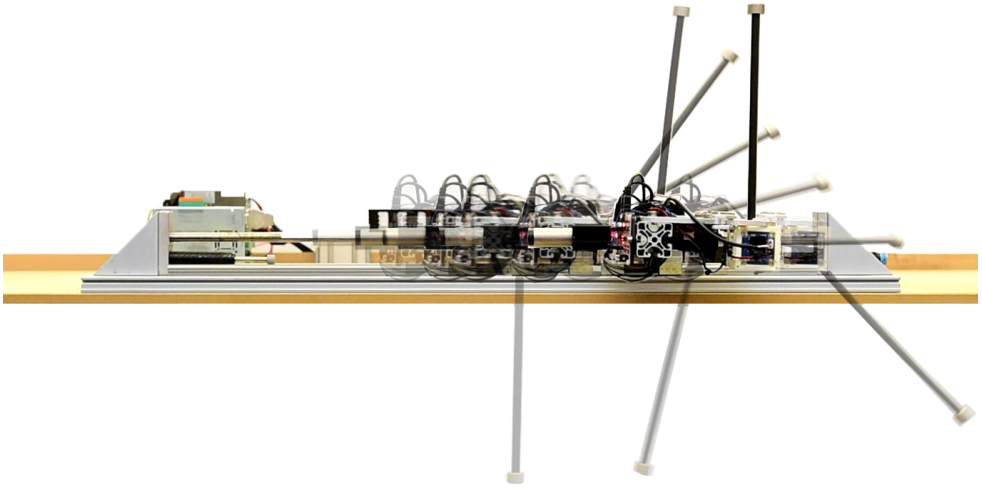
The three components that enable such adaptive dynamic swing-up manipulation are presented here. First, the high-dimensional force distribution acting on the sensor surface is directly inferred from the sensor camera images using an efficient convolutional network, which is trained on purely simulated contact interactions of the sensor with different poles. Second, a novel simulator is developed that accurately models the behaviour of the soft sensor surface when interacting with a rigid cylindrical pole. This simulation is based on combining the finite element method with state-of-the-art semi-implicit time-stepping schemes for contact resolution and runs at 360 Hz on a single core of an Intel Core i7-7700k processor. Third, a framework for learning adaptive feedback policies conditioned on a history of sensory observations is proposed. Deep reinforcement learning is utilized to train a single policy, entirely in simulation, that is able swing up different poles with unknown physical attributes. Thereby, various strategies that facilitate the sim-to-real transfer of policies learned in simulation are employed, namely *dynamics randomization* [2], and *privileged learning* [3], [4].

1.1 Related work

As reviewed in [5] and [6], many works have demonstrated how robots can leverage the sense of touch in dexterous manipulation tasks in closed-loop. For example, in [7] tactile data is used to control both the grasping force and slippage of a tactile gripper. Other examples include [8], where a pair of grippers are used to pick one end of a cable and follow it to the other end. Each gripper contains tactile sensors from which the current pose and friction forces acting on the cable can be estimated in real-time, enabling the approach to generalize to cables of different thicknesses and materials, based on a model learned from real data. A similar approach was proposed in [9]; a dual palm robotic system estimates the pose and the stick/slip behaviour of an object solely from tactile



(a) Simulation



(b) Reality

Figure 1. In this work, tactile control policies for the swing-up manipulation of poles are learned in a simulation of the tactile sensor that runs faster than real-time. When transferred to the real-world system, the policy achieves the desired behaviour without further adaptation, and swings up poles to an upright position without prior knowledge of the physical attributes of the pole.

feedback, in order to manipulate an object on a planar surface to a desired position. In [10], a deep dynamics model is learned that can predict future tactile observations based on the previous observations and actions taken. Data for the training of the dynamics model is autonomously collected on the physical system. The learned model is then used in an MPC-framework to manipulate a ball, analog stick, and 20-sided die to a desired configuration. Other learning-based approaches rely on deep reinforcement learning to find optimal control policies directly on the physical hardware. Examples include a 5-DoF arm that learns to reorient objects using a latent representation of the tactile data [11], and a robotic system that learns to type on a Braille keyboard [12].

While these approaches demonstrate robustness against external disturbances and changes in object properties, the manipulation tasks they solve generally do not require a high degree of dynamicism. For more aggressive manipulation tasks such as the swing-up manipulation of poles, feedback control based on tactile data has proven to be challenging and thus differing methods have been proposed. In [13], the sensing and manipulation are separated into two steps. First, the physical features of different poles are learned by shaking and tilting the pole in-hand and observing the tactile feedback. The learned features are then used to optimize an open-loop trajectory of a robotic arm that dynamically swings the pole up to a desired angle. The learning of the physical features, as well as the trajectory optimization, are performed end-to-end using models trained on a physically collected dataset. In [14], tactile sensing and visual tracking are combined to pivot an object to a desired angle by adjusting the gripping force exerted by a two-finger gripper. This fusion of visual and tactile information was also employed on a robotic hand in [15] to perform highly dynamic tasks such as pen spinning, ball dribbling, and ball throwing.

In this work, a unified approach is presented where aggressive swing-up maneuvers can be achieved in closed-loop from high-dimensional tactile feedback, without relying on a visual tracking system or prior in-hand exploration of the pole. Moreover, instead of relying on data collection on the physical system, as is done in the learning-based methods mentioned above, the feedback control policy is learned entirely in simulation. This removes the cost of collecting data on the physical system, which can be highly time-consuming. Additionally, challenging motions that may lead to unsafe behaviours by the physical hardware can be first explored without repercussions. This data can then be utilized to train the policy to satisfy the safety constraints that are present on the physical system. While simulators for the behaviour of tactile sensors have been developed (see e.g. [16]–[23]), the authors are not aware of any work where a simulation from first principles is utilized to learn tactile feedback control policies. Rather, the mentioned works focus on gathering supervised datasets of tactile images in simulation to train deep neural networks that can predict object position and rotation ([16], [21]), the force distribution acting on the sensor surface ([22], [23]), or the three-dimensional mesh of the object in contact ([17]).

1.2 Outline

The hardware employed for the experiments is presented in Section 2. In Section 3, the proposed methods are described. This includes the sensing approach of the tactile sensor in Section 3.1, the design of the tactile simulator in Section 3.2, and the synthesis of the swing-up control policy in Section 3.3. Results from employing the learned policy on the real-world system are presented in Section 4. Finally, Section 5 draws conclusions and gives an outlook on future work. In the remainder of this paper, vectors are expressed as tuples for ease of notation, with dimension and stacking clear from the context.

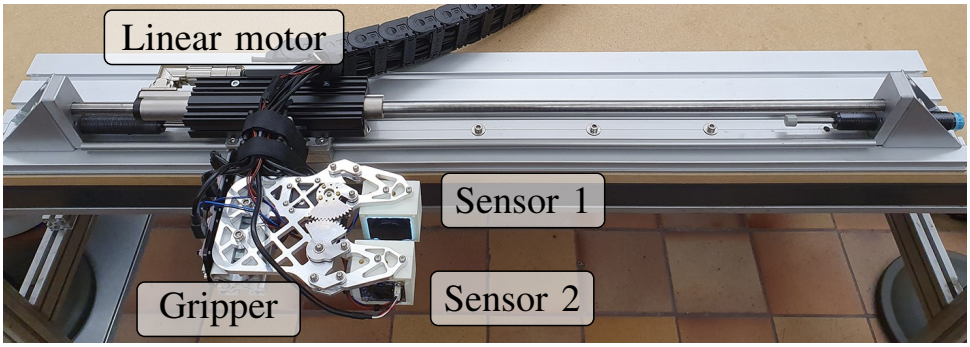


Figure 2. The robotic system presented in this paper consists of a parallel gripper comprising two tactile sensor, and a linear motor to which the gripper is mounted.

(a) Parallel gripper **(b)** Tactile sensor

Figure 3. This figure shows exploded views of the gripper (a) and the tacile sensor that acts as finger (b). A Dynamixel MX-28R controls the opening and closing of the two fingers, each of them equipped with a tactile sensor.

2. Hardware

The robotic system considered in this paper consists of three main parts; a two-finger robotic gripper where each finger comprises a tactile sensor, a linear motor (stator/slider), to which the gripper is mounted, and finally, embedded computing systems that process the sensing data and send commands to the actuators. The linear motor and tactile gripper are pictured in Fig. 2.

2.1 Tactile gripper

To enable the high-resolution gripping capability of the system, a custom 1-DoF parallel two-finger gripper was built in-house, see Fig. 3a. A Dynamixel MX-28R servo motor is used to control the distance between the two fingers with a resolution of 0.06 mm. Two

tactile sensors, placed opposite each other, act as fingers for the gripper. The sensing principle employed in this paper is based on [1]. Three soft silicone layers are poured on top of an RGB fisheye camera (ELP USBFHD06H), surrounded by LEDs. The base layer (ELASTOSIL ® RT 601 RTV-2, mixing ratio 7:1, shore hardness 45A) is stiff and transparent, and serves as a spacer. The middle layer (ELASTOSIL ®RT 601 RTV-2, ratio 25:1, shore hardness 10A) is soft and transparent, and embeds a spread of randomly distributed fluorescent green particles. A black top layer (made of the same material as the middle layer) completes the sensor and shields it from external light disturbances. The soft sensor's surface is slightly curved to provide a more anatomical grasping surface. An exploded view of the sensor layers is shown in Fig. 3b.

2.2 Linear motor

In order to achieve the translational motion of the gripper, a linear motor comprising a *stator* and a *slider* is employed. The stator (LinMot P01-23x160H-HP-R) contains the motor windings, bearings for the slider, position capture sensors and a microprocessor, and is thus able to generate motion with respect to the slider. The slider (LinMot PL01-12x850/810-HP) is a stainless steel tube and is fixed to a table so that the stator is the only moving part. The gripper is then mounted to the stator through the use of a motor flange (LinMot PF02-23x120). A motor drive (LinMot C1100-GP-XC-0S-000) controls the motion of the stator.

2.3 Embedded systems

Two embedded devices are used to control the system. First, a Raspberry Pi (RPi) runs two low-level controllers, one for the linear motor and one for the gripper. The linear motor controller tracks commanded acceleration setpoints, while the gripper controller tracks the distance between the two fingers. Second, an NVIDIA Jetson TX2, a compact embedded device with a built-in GPU, obtains the camera images from the tactile sensor at 60 Hz, pre-processes the images, and infers the force distribution. Note that this pipeline is only executed for sensor 1 (see Fig. 2). This is motivated by the fact that due to the planar nature of the system, the forces acting on sensor 2 can be assumed to be symmetrical to those acting on sensor 1. Furthermore, this reduces the computational complexity of the pipeline.

The Jetson also receives the current actuator states from the RPi. Control actions are then inferred using the proposed control policy and are communicated to the low-level controllers on the RPi that execute the commands.

3. Method

The proposed method can be divided into three different parts. First, the vision-based tactile sensor estimates the force distribution acting on its surface from its camera images.

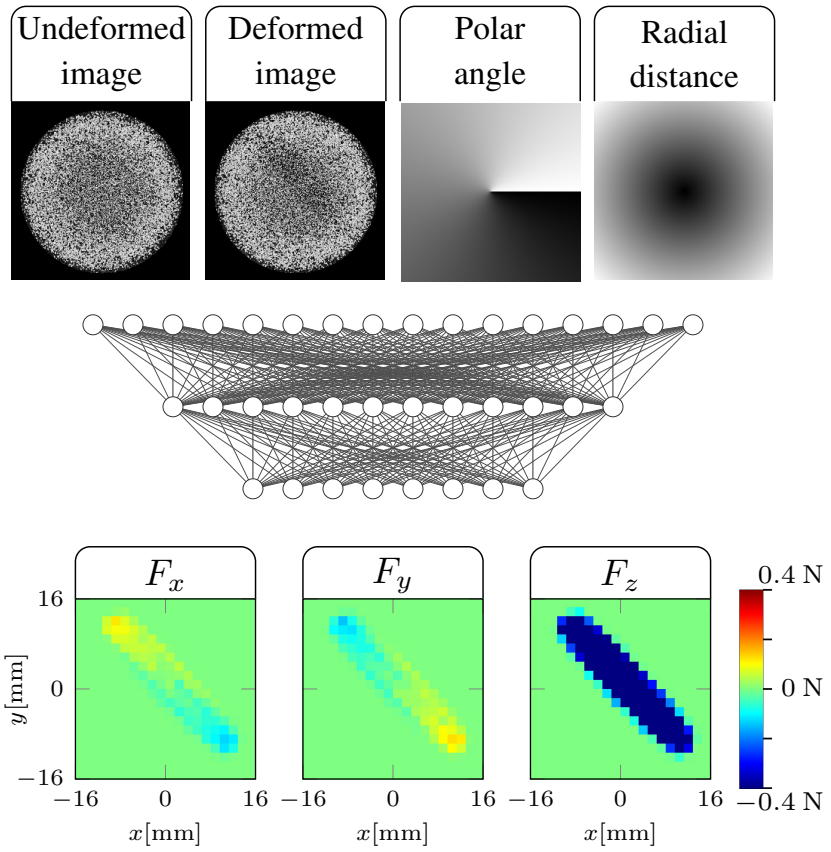


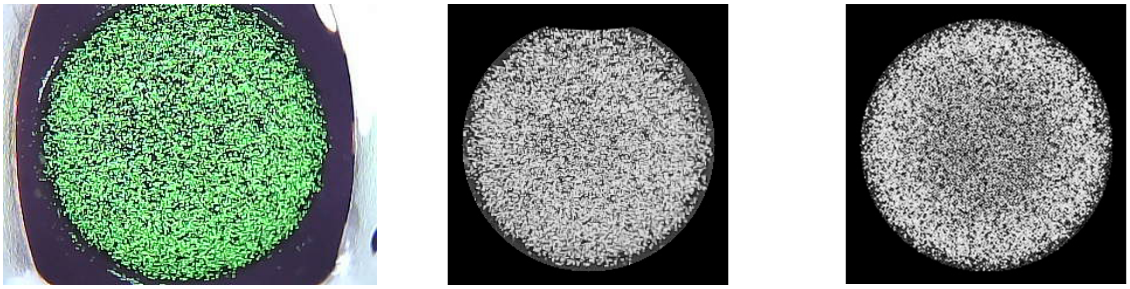
Figure 4. The features and labels of our supervised learning task. The features are the images of the particles in the undeformed and in the deformed states. Two additional channels provide the polar coordinates of the pixels. For the labels, F_x, F_y and F_z denote the x, y and z components of the discretized force distribution. The force distributions shown in the figure were collected with a pole pivoting on the sensor (as in the experiments discussed in Section 4). They show how the shape and pose of the pole manifest themselves correctly in the force distribution readings. Additionally, the lateral forces (F_x and F_y) on either side of the center of rotation point in opposite directions, which can be deducted from their change in color.

Second, a simulator for the dynamics of a pole and the given robotic system is developed. Third, a tactile feedback control policy for the swing-up manipulation is learned in the simulation using reinforcement learning.

3.1 Vision-based tactile sensing

The tactile sensor employed in this paper follows the same sensing principle as introduced in [1]. When the soft sensing surface is subject to force, the material deforms and displaces the particles tracked by the camera. This motion generates different patterns in the images. The material deformation at any point in time can thus be described by two camera images, one where no loads are applied and the material is at rest, and another at the current deformed state.

In [22], a method to generate such images in simulation is presented to train a supervised learning architecture that aims to accurately estimate the real-world 3D contact force distribution. The same approach to generate training data is employed here, using



(a) Original image **(b)** Remapped image **(c)** Simulated image

Figure 5. Camera images from the real-world sensor are preprocessed converting the original image (a) to gray-scale, and remapping the image using a calibrated camera model (b). The last image (c) corresponds to a simulated image that is used for training and is provided for comparison.

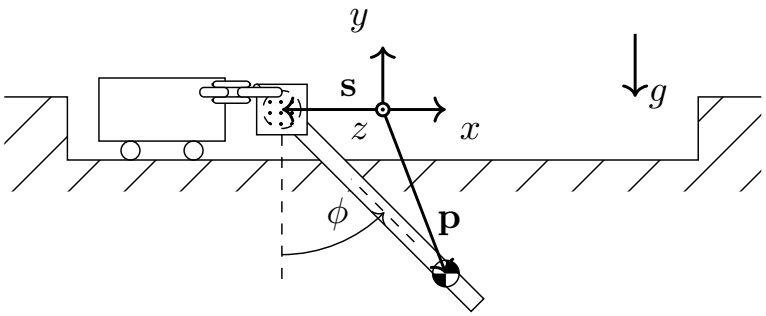
finite-element simulations of the sensor surface under various contact conditions, where hyperelastic material models for the sensor's soft materials are employed. The details of this procedure can be found in [22]. In addition to the two mentioned images per datapoint, the polar coordinates of each pixel are encoded here as two additional image channels. Explicitly incorporating such spatial location features has previously been shown to significantly improve accuracy where the location of image features is relevant for the task at hand [24]. Using a fully convolutional neural network based on ShuffleNet V2 [25], the resulting four-channel image is then mapped to accurate contact force distribution labels (see Fig. 4), with ground truth also obtained from finite element simulations [26].

On the real-world sensor, camera images are preprocessed to match those of the simulated training dataset as described in [22] and [23]. Specifically, images are converted to gray-scale and remapped using the real-world camera model (obtained via a state-of-the-art calibration technique [27]) to images of the same scene as if they were taken in the simulated world. A circular mask is then applied to remove any irrelevant image information. The results of this preprocessing procedure are illustrated in Fig. 5. On the given hardware, the force distribution for a given preprocessed camera image can be inferred in real-time in 2.5 ms.

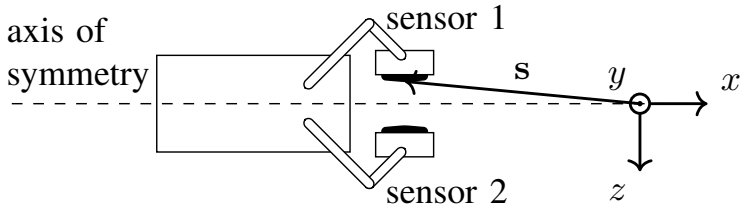
3.2 Tactile simulation

In order to achieve a fast simulator, essential for training reinforcement learning algorithms in a reasonable amount of time, a few model simplifications are introduced here. First, the material of the sensor is assumed to be linearly elastic. Second, the forces acting on the sensor surface are decoupled into two components: the forces arising due to the material deformation in the z direction, and the lateral friction forces resulting from the relative motion of the pole with respect to the sensor. A real-time finite element approach is then employed to compute the two mentioned force components.

1) Problem Statement A sketch of the system considered in this work is shown in Fig. 6, where a single coordinate system is defined. The x-axis is aligned to the moving axis of



(a) Side-view sketch of the system.



(b) Top-view sketch of the system.

Figure 6. The system can be described as a cart-pole augmented with a parallel gripper that features the two sensors.

the linear motor, denoted in the following as the cart. The y-axis points in the opposite direction to gravity, i.e. upwards. Finally, the z-axis is chosen such that all the points on sensor 1 exhibit a negative z-coordinate. The reference position $\mathbf{s} = (x_\mathrm{s},\ y_\mathrm{s},\ z_\mathrm{s})$ is chosen with the point on the curved surface of sensor 1 that is closest to the x-y plane (at rest). While there are two tactile sensors, their positions are symmetrical about the x-y plane. Hence, it suffices to only consider the position of a single sensor. Next, the orientation of the pole is defined by the angle ϕ. Lastly, $\mathbf{p} = (x_\mathrm{p},\ y_\mathrm{p},\ z_\mathrm{p})$ denotes the position of the center of mass of the pole. Note that the sensor is fixed in the y-direction ($y_\mathrm{s} = 0$) and the pole is fixed in the z-direction ($z_\mathrm{p} = 0$). These definitions are illustrated in Fig. 6. The state vector $\mathbf{x}$ is then defined as

$$\mathbf{x} = (x_\mathrm{s},\ \dot{x}_\mathrm{s},\ z_\mathrm{s},\ x_\mathrm{p},\ \dot{x}_\mathrm{p},\ y_\mathrm{p},\ \dot{y}_\mathrm{p},\ \phi,\ \dot{\phi}) \tag{1}$$

Since only the static behavior of the sensor material is analyzed, $\dot{z}_\mathrm{s}$ is not considered.

The inputs to the system are the cart acceleration and the increment in z_s between two subsequent timesteps:

$$\mathbf{u} = (\ddot{x}_\mathrm{s},\ \Delta z_\mathrm{s}) \quad . \tag{2}$$

Note that on the real system, the servo commands are mapped to Δz_s with a linear mapping identified from data.

Next, the pole is characterized as a rigid cylinder. Its radius is given by r_p, the mass

by m_p and its moment of inertia about the center of mass and along the z-axis by I_p. The length of the pole above its center of mass is given by $l_\mathrm{p,u}$ and the length below the center of mass by $l_\mathrm{p,l}$.

Given these definitions, the goal is to model the state evolution over time, i.e., $\mathbf{x}(k+1) = f(\mathbf{x}(k), \mathbf{u}(k))$, where k is the discrete time index, and f describes a functional dependency. In the following, the time index (k) will be omitted, and variables at time $(k+1)$ will be denoted by a + superscript, e.g. x^+.

2) Equations of Motion The pole is modeled as a free-body constrained to move in the x-y plane, meaning that its motion is governed by the force $F_\mathrm{p} = (F_{\mathrm{p},x},\ F_{\mathrm{p},y},\ 0)$ and torque $T_\mathrm{p} = (0,\ 0,\ T_{\mathrm{p},z})$ acting on its center of mass. Using a semi-implicit integration scheme ([28], [29]) the equations of motion are then given by

$$\dot{x}_\mathrm{s}^+ = \dot{x}_\mathrm{s} + \Delta t\ \ddot{x}_\mathrm{s} \qquad x_\mathrm{s}^+ = x_\mathrm{s} + \Delta t\ \dot{x}_\mathrm{s}^+ \tag{3}$$

$$\dot{x}_\mathrm{p}^+ = \dot{x}_\mathrm{p} + \Delta t\ \frac{F_{\mathrm{p},x}}{m_\mathrm{p}} \qquad x_\mathrm{p}^+ = x_\mathrm{p} + \Delta t\ \dot{x}_\mathrm{p}^+ \tag{4}$$

$$\dot{y}_\mathrm{p}^+ = \dot{y}_\mathrm{p} + \Delta t\ \frac{F_{\mathrm{p},y}}{m_\mathrm{p}} \qquad y_\mathrm{p}^+ = y_\mathrm{p} + \Delta t\ \dot{y}_\mathrm{p}^+ \tag{5}$$

$$\dot{\phi}^+ = \dot{\phi} + \Delta t\ \frac{T_{\mathrm{p},z}}{I_\mathrm{p}} \qquad \phi^+ = \phi + \Delta t\ \dot{\phi}^+ \tag{6}$$

$$z_\mathrm{s}^+ = z_\mathrm{s} + \Delta z_\mathrm{s} \tag{7}$$

In the following, the derivation of F_p and T_p is presented.

Both sensors are discretized using an identical mesh of $N = 576$ finite elements (nodes). Hereafter, all quantities introduced will refer to sensor 1, where the corresponding counterparts of sensor 2 are clear from the symmetrical context and are denoted using a tilde, i.e. $\tilde{\cdot}$. Then, for node i of the mesh, let (x_i, y_i, z_i) be its coordinates, and F_i the force acting on the node. Each node i in contact with the pole leads to a planar reaction force

$$F_{\mathrm{p},i} = -(F_i + \tilde{F}_i) \implies F_{\mathrm{p},i,x:y} = -2F_{i,x:y} \tag{8}$$

where the implication follows from symmetry, with the $x:y$ subscript denoting the stacked x and y components of the three-dimensional vector. Next, the gravitational force acting on the pole is denoted by $F_\mathrm{g} = \big(0,\ -m_\mathrm{p}g,\ 0\big)$, where $g = 9.81\,\mathrm{m\,s^{-2}}$. Defining $\mathbf{r}_i := \big(x_i - x_\mathrm{p},\ \ y_i - y_\mathrm{p},\ \ 0\big)$, the total force and torque acting on the pole are then

$$F_\mathrm{p} = F_g + \sum_{i\in\mathcal{S}} F_{\mathrm{p},i}\ , \qquad T_\mathrm{p} = \sum_{i\in\mathcal{S}} \mathbf{r}_i \times F_{\mathrm{p},i}. \tag{9}$$

where $\mathcal{S}$ is the set of all nodes on the surface of the sensor.

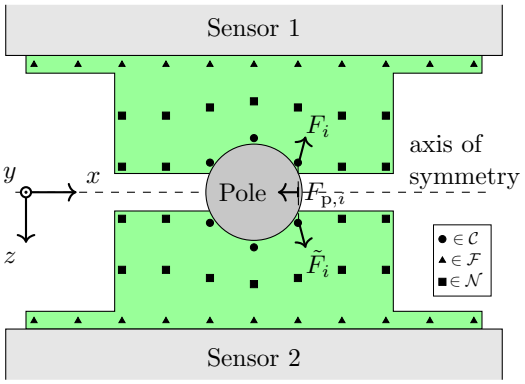


Figure 7. The nodes of the finite element mesh are assigned to three sets: $\mathcal{C}$ if in contact with the pole, $\mathcal{F}$ if constrained to not move, $\mathcal{N}$ otherwise.

As mentioned above, the contact forces are postulated to be the superposition of forces F_i^0 arising from the normal indentation of the pole into the sensor, and the lateral friction forces F_i^f, that is, $F_i = F_i^0 + F_i^{\mathrm{f}}$, which implies

$$F_{\mathrm{p},i} = \underbrace{-\left(F_i^0 + \tilde{F}_i^0\right)}_{=:F_{\mathrm{p},i}^0} \underbrace{-\left(F_i^{\mathrm{f}} + \tilde{F}_i^{\mathrm{f}}\right)}_{=:F_{\mathrm{p},i}^{\mathrm{f}}} \tag{10}$$

3) Forces Arising from Normal Indentation The forces F_i^0 are derived using the finite element theory for linearly elastic materials. Let U_i^0 be the deformation of a node i. Then, a linear relationship between the external forces and deformations is found by the finite element method as:

$$F^0 = KU^0 \tag{11}$$

where $F^0 := \left(F_1^0,\ \ldots,\ F_N^0\right)$, $U^0 := \left(U_1^0,\ \ldots,\ U_N^0\right)$, and K is the global stiffness matrix, obtained in this work in Abaqus/Standard. The system of equations in (11) can be solved by introducing the following sets, displayed in Fig. 7:

- $\mathcal{C}$: The set of all nodes that are in contact with the pole. It is the intersection of the set of nodes at the surface of the sensor (i.e., the set $\mathcal{S}$) and the set of nodes whose positions at rest collide with the pole, based on the geometric properties of the pole considered. The nodes in this set are assumed here to translate only in z-direction, and their deformation is obtained by finding the appropriate z-coordinate that intersects with the surface of the pole (see Fig. 7).

- $\mathcal{F}$: The set of all nodes that are in contact with the base layer of the sensor. Since the base layer's stiffness is much larger than the stiffness of the sensor surface, the nodes of this set are assumed to be rigid. Therefore, their deformation is set to zero, i.e., $U_i^0 = 0, \forall i \in \mathcal{F}$.

- $\mathcal{N}$: The set of nodes that are neither in contact with the base layer nor in contact with the pole. No external forces are acting on these nodes, i.e., $F_i^0 = 0, \forall i \in \mathcal{N}$.

Therefore, for a node i, once a corresponding set is identified, either the force F_i^0 or the deformation U_i^0 is known. The system (11) is then solved by using the UMFPACK library, and $F_{\mathrm{p},i}^0$ computed as in (10).

Note that here the current approach exploits the cylindrical geometry of the poles, rendering a mathematically simple intersection problem, which enables a highly efficient identification of the aforementioned sets. The extension to objects of various geometries may still be addressed efficiently by employing algorithms tailored to solve the intersection problem for generic polygons, e.g., based on the Weiler-Atherton clipping algorithm [30].

4) Lateral Friction Forces In order to find the lateral friction forces for the nodes in contact, first the case where only the friction at a single node is unknown is considered. From the solution of this case, an iterative method is utilized to solve for all friction forces in the multi-contact case.

First, it is assumed that all the friction forces except for the one at node i are known, i.e. F_j^{f} is known for all $j \neq i$. Let

$$\mathbf{v}_{i,\mathrm{rel}} := \begin{pmatrix} \dot{x}_i \\ \dot{y}_i \end{pmatrix} - \begin{pmatrix} \dot{x}_{\mathrm{s}} \\ 0 \end{pmatrix} = \begin{pmatrix} \dot{x}_{\mathrm{p}} \\ \dot{y}_{\mathrm{p}} \end{pmatrix} + \begin{pmatrix} -\mathbf{r}_{i,y} \\ \mathbf{r}_{i,x} \end{pmatrix} \dot{\phi} - \begin{pmatrix} \dot{x}_{\mathrm{s}} \\ 0 \end{pmatrix}$$

be the relative planar velocity of the point on the pole which is in contact with the node i at time k. Then, by plugging in the equations of motion (3)-(7), the relative velocity at the next timestep is found to be

$$\mathbf{v}_{i,\mathrm{rel}}^{+} = \mathbf{v}_{i,\mathrm{rel}} - \Delta t \begin{pmatrix} \ddot{x}_{\mathrm{s}} \\ 0 \end{pmatrix} + \mathbf{J}_{ii} \underbrace{F_{\mathrm{p},i,x:y}}_{F_{\mathrm{p},i,x:y}^0 + F_{\mathrm{p},i,x:y}^{\mathrm{f}}} + \sum_{j \neq i} \mathbf{J}_{ij} F_{\mathrm{p},j,x:y} \tag{12}$$

where, for generic indices a and b,

$$\mathbf{J}_{ab} = \Delta t \begin{bmatrix} \dfrac{1}{m_{\mathrm{p}}} + \dfrac{\mathbf{r}_{a,y}\mathbf{r}_{b,y}}{I_{\mathrm{p}}} & \dfrac{-\mathbf{r}_{a,y}\mathbf{r}_{b,x}}{I_{\mathrm{p}}} \\ \dfrac{-\mathbf{r}_{a,x}\mathbf{r}_{b,y}}{I_{\mathrm{p}}} & \dfrac{1}{m_{\mathrm{p}}} + \dfrac{\mathbf{r}_{a,x}\mathbf{r}_{b,x}}{I_{\mathrm{p}}} \end{bmatrix}. \tag{13}$$

In this work, Coulomb friction is assumed, and two cases are identified, where μ indicates both the static and kinetic friction coefficients. First, for the static friction case, consider the force $F_{\mathrm{p},i}^{\mathrm{f,static}}$ that takes on exactly the value to prevent motion at node i. This force can be found by setting $\mathbf{v}_{i,\mathrm{rel}}^{+} = \mathbf{0}$ in (12) and solving for $F_{\mathrm{p},i,x:y}^{f}$. If this force satisfies the

friction cone constraint, i.e.

$$\left\| F_{\mathrm{p},i}^{\mathrm{f,static}} \right\| \leq 2\mu \left| F_{i,z}^{0} \right|, \tag{14}$$

then $F_{\mathrm{p},i,x:y}^{\mathrm{f}} = F_{\mathrm{p},i,x:y}^{\mathrm{f,static}}$. The z-component is set to zero as it would eventually cancel out when considering both fingers.

If (14) is not satisfied, friction is not sufficient to prevent the motion at node i. In this case, kinetic friction is present, where the force is opposite to the direction of the velocity and is proportional to the normal component of the force. Since the velocity $\mathbf{v}_{i,\mathrm{rel}}^{+}$ and the friction force $F_{p,i}^{\mathrm{f}}$ are coupled, an approximation of the subsequent velocity is employed as

$$\hat{\mathbf{v}}_{i,\mathrm{rel}}^{+} := \mathbf{v}_{i,\mathrm{rel}} - \Delta t \begin{pmatrix} \ddot{x}_{\mathrm{s}} \\ 0 \end{pmatrix} + \mathbf{J}_{ii} F_{\mathrm{p},i,x:y}^{0} + \sum\nolimits_{j \neq i} \mathbf{J}_{ij} F_{\mathrm{p},j,x:y}$$

which is the relative velocity at the subsequent step when the effects of friction at node i are ignored. If the number of nodes is sufficiently large, this approximation is close to the true value, since the effect of the force at the single node i is small compared to the combined effect of the remaining forces at nodes $j \neq i$. Using this approximation, the kinetic friction is set to

$$F_{\mathrm{p},i,x:y}^{\mathrm{f}} = -2\mu |F_{i,z}^{0}| \hat{\mathbf{v}}_{i,\mathrm{rel}}^{+} / \left\| \hat{\mathbf{v}}_{i,\mathrm{rel}}^{+} \right\| \tag{15}$$

Given this solution to the single contact problem, the multi-node contact case is solved by repeatedly iterating over all nodes in contact until convergence, and updating the friction force at node i using the above solution, given the values at nodes $j \neq i$ of the current iteration. Then, $F_{\mathrm{p},i}$ is obtained as in (10) from $F_{\mathrm{p},i}^{0}$ and $F_{\mathrm{p},i}^{\mathrm{f}}$, and finally F_{p} and T_{p} can be computed as in (9).

3.3 Learning tactile control policies

Given a simulation of a robotic system, deep reinforcement learning (deep RL) algorithms have been successfully applied to learn sophisticated behaviours [32]. These algorithms typically depend on the Markov property of the system, i.e. they assume that the state and physical parameters that fully describe the system at a given time are available to the policy. However, for the experiments presented in Section 4, the physical parameters of the pole, e.g. the length, are unknown to the policy and the Markov property no longer holds. In deep RL, such problems are typically dealt with by using the history of observations and parametrizing the policy using recurrent neural networks which, however, can be challenging to train.

The approach employed here exploits the fact that in simulation the state and physical parameters are known. In a first stage, an *expert* policy π^{e} is learned that has access to the

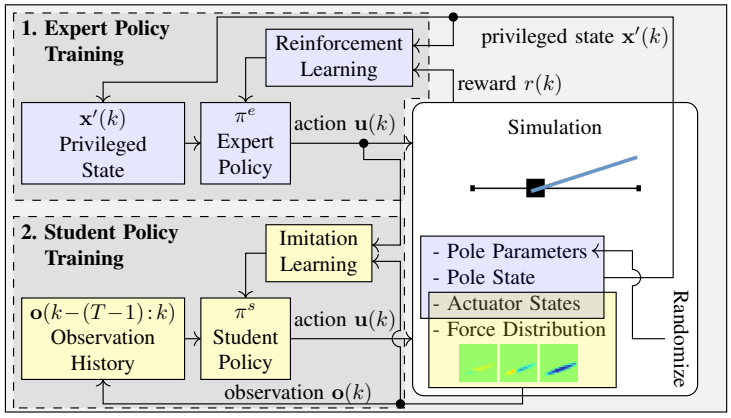


Figure 8. The figure depicts the privileged learning approach. The expert and student policy training take place in two separate steps, both performed entirely in simulation. Further, both policies are parametrized using two-layer fully connected neural networks. For the reinforcement learning of the expert policy, the SAC [31] method is employed to find the optimal policy according to (17). The student policy is then deployed to the real-world system without further adaptation.

state as well as the simulation parameters (satisfying the Markov property). In a second stage, a *student* policy π^{s} that only has access to the observations that are available on the real system is learned by imitating the behaviour of the expert policy (see Fig. 8). This is also referred to as privileged learning [3], [4].

1) State-Feedback Expert Policy In order to achieve the swing-up with a feedback policy that adapts to different poles, the expert policy is conditioned on the state $\mathbf{x}(k)$ (defined in Section 3.2), as well as the pole's physical parameters which may vary. This yields the augmented state

$$\mathbf{x}'(k) := \big(\mathbf{x}(k),\ r_{\mathrm{p}},\ m_{\mathrm{p}},\ I_{\mathrm{p}},\ l_{\mathrm{p,u}},\ l_{\mathrm{p,l}},\ \mu\big) \quad . \tag{16}$$

As a result, the policy may choose different control actions based on the features of the pole.

The goal is then to find a policy, $\pi\colon \mathbf{x}'(k) \to \mathbf{u}(k)$, that is optimal in the sense of maximizing the expected sum of future discounted rewards, i.e.

$$\pi^{\mathrm{e}} = \max_{\pi} \mathbb{E}\left(\sum_{k} \gamma^k r(\mathbf{x}'(k), \pi(\mathbf{x}'(k)))\right) \quad , \tag{17}$$

where γ is the discount factor. The reward function r is shaped to encourage low slippage and pole orientations that are close to 180°. The policy is learned using deep RL, namely the SAC [31] algorithm with the stable-baselines3 implementation [33]. The discount factor is set to $\gamma = 0.995$ while the remaining hyperparameters, as well as the policy network architecture, correspond to the default ones proposed in [31]. During training, the pole parameters $r_{\mathrm{p}},\ m_{\mathrm{p}},\ I_{\mathrm{p}},\ l_{\mathrm{p,u}},\ l_{\mathrm{p,l}}$ and μ are randomly sampled at each new

episode such that the policy learns the correct behaviour for different poles and friction ratios. This *dynamics randomization* [2] also greatly aids in the successful transfer from simulation to reality.

2) Tactile Student Policy The expert policy is conditioned on privileged knowledge, only available in simulation, and can thus not be deployed on the real system, where the pole's pose and physical attributes can only indirectly be observed through the available force distribution measurements. As a result, the student policy must be able to reason over time and implicitly recover the missing state information. First, in order to condense the sensory information into a compressed representation, an estimate of the pole's orientation $\hat{\phi}(k)$ is obtained by computing the force magnitude at each bin, thresholding the magnitudes to obtain a binary image, and finally applying a Hough line transform [34]. In addition, the total sensed normal force $F_z^{\text{tot}}(k)$ is extracted by summing the z-distribution at all bins. This is motivated by the fact that the normal force yields direct information about the friction and slippage, while the angle of the pole is the main quantity to be controlled. Then, a student policy conditioned on a history of condensed representations of the observations is learned by imitating the behaviour of the expert policy.

A condensed observation at time k is given by

$$\mathbf{o}(k) = \left(x_{\text{s}}(k),\ z_{\text{s}}(k),\ \hat{\phi}(k),\ F_z^{\text{tot}}(k) \right) \ . \tag{18}$$

Note that $x_{\text{s}}(k)$ and $z_{\text{s}}(k)$ are known for the real system, and the velocity of the cart $\dot{x}_{\text{s}}(k)$ is not included, since it can implicitly be derived from the history of $x_{\text{s}}(k)$ observations.

The student policy π^{s} is then parametrized by a neural network that maps the history of the last T condensed observations $\mathbf{o}(k-(T-1):k)$ to the control action $\mathbf{u}(k)$, where $T = 12$ is the fixed history length. The same stochastic network as proposed in [31] is used, which outputs a squashed Gaussian distribution over the control actions. This stochasticity accomplishes a desirable smoothing of the policy. The imitation of the expert policy is then posed as a supervised learning task that minimizes the negative log-probability

$$\mathcal{L} := -\log \Pr\left(\pi^{\text{s}}\left(\mathbf{o}(k-(T-1):k) \right) = \pi^{\text{e}}\left(\mathbf{x}'(k) \right) \right) \ .$$

In this work, the DAGGER [35] method is employed, where the dataset is continuously aggregated with the incoming data from the training rollouts of the student policy. Labels are obtained by querying the expert policy for the visited states. At each training iteration, the student policy is updated by performing an optimization step with batches sampled from the aggregated dataset.

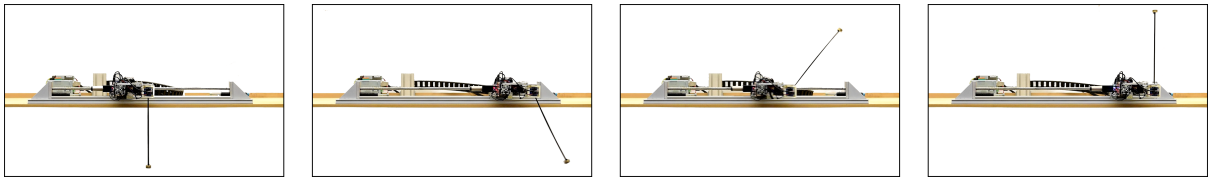


Figure 9. This figure shows a trajectory which results from employing the learned feedback control policy on the robotic system. As can be seen, the pole is dynamically swung up to an upright position.

4. Results

The validity of the methods presented is verified on the physical system, where the learned feedback policy is deployed to swing up different poles.

Feedback is crucial for this task for three reasons: i) the control actions to perform a successful swing up greatly depend on the physical parameters of the specific pole, which are assumed to be unknown to the policy in this work, ii) these control actions depend on the initial position and orientation of the pole, which is likely to differ across trials on the real system, iii) even when the physical parameters and starting pose of the pole are well known, and a trajectory is generated in simulation for such a configuration, in the authors' experience this led to swing-ups with an offset in the final angle due to slight model mismatches. Feedback is thus needed to precisely control the final angle.

Throughout the following experiments, the initial grasping of the pole is achieved by a human holding the pole between the two tactile sensors. The gripper then slowly closes its fingers until the total force applied on the sensor by the pole reaches a user-defined threshold.

The student feedback control policy is evaluated on the real-world robotic system on four different poles with masses ranging from 20 g to 38 g, lengths from 20 cm to 35 cm, and radii from 2.5 mm to 5 mm. For each pole, the control policy is run ten times and the error from 180° in the final estimated angle $\hat{\phi}$ is recorded. Experiments show that all four poles are successfully swung up to an upright position, and a mean absolute error of 4.3° is achieved. A detailed analysis of the experimental results is provided in an experimental report [36]. These results demonstrate how a single policy is able to adapt the robot's motion to perform swing-up maneuvers for a wide range of different poles without any prior knowledge of the pole's physical features, based on the feedback provided by the tactile sensor. The resulting behaviour of the policy for one of the listed poles is depicted in Fig. 9. The supplementary video[1] contains the trajectories for the remaining poles. It is vital to note that the pole shown in Fig. 9 is not contained in the distribution of poles that is used while learning either the teacher or the student policy.

Moreover, the policy is transferred directly from the simulation to the real system with no adaptation needed. This further asserts the robustness of the policy as it is able to adapt to the real system that exhibits dynamics that are not modeled in the simulation

[1] `https://youtu.be/In4jkaHzJLc`

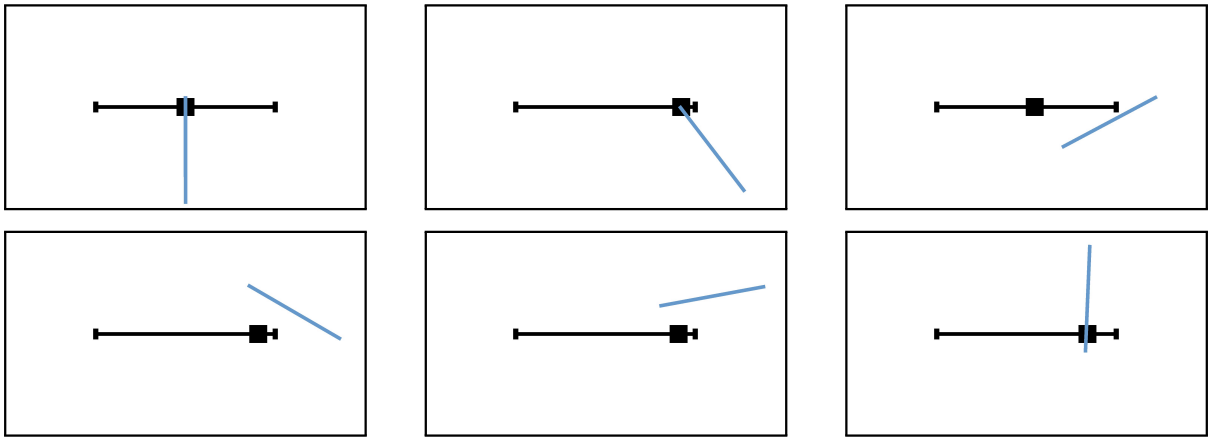


Figure 10. Using deep reinforcement learning, robust policies can be learned to achieve various tasks. Here, the reward function is shaped to encourage the throwing and catching of the pole after a rotation of 360°.

(such as dynamic effects of the sensor material, unmodeled dynamics of the actuators, and delays of the actuator commands).

5. Conclusion

In this paper, a strategy has been presented to transfer tactile control policies for the swing-up manipulation of different poles from simulation to a physical robotic system. As the simulator has been shown to closely match the dynamics of the real system, the policy learned in simulation generalizes to the real-world robotic system with no adaptation needed. Note that the system presented here does neither exploit a mechanically fixed pivot point nor directly control the rotational degree-of-freedom of the pole, but it can achieve the desired motion only through the presence of friction, whose modeling was crucial in enabling a realistic simulation.

This constitutes an important step towards a general framework to learn a wide variety of tactile manipulation tasks safely in simulation. Yet, current results have only been demonstrated for a single task on a single robotic system. Future work will focus on several aspects to further extend the generalizability of this work. In a first step, the proposed framework could be utilized to learn other pole manipulation skills on the given system, e.g. the throwing and catching of a pole. While such a policy was already successfully learned in simulation (see Fig. 10), the transfer to the physical system requires further work due to non-idealities of the hardware. For instance, when the pole is thrown in the air, it may leave the plane to which the motion is assumed to be constrained. In fact, instead of relying on the planar nature of the manipulation task, as was done in this work, the suggested simulator could be extended to handle non-planar tasks. As a result, manipulation skills for grippers that can be controlled in six degrees of freedom could also be learned. Moreover, in this paper, hand-engineered features are extracted from the tactile observations, i.e. the orientation and total normal force acting on the

Pole	r [mm]	l [mm]	m [g]	MAE [deg]	Failures
1	2.5	200	28	5.6	29% (4/14)
2	2.5	350	38	3.5	9% (1/11)
3	3.8	250	20	4.4	9% (1/11)
4	5.0	300	36	3.6	0% (0/10)
Total				**4.3**	**11.7%**

Table 1. This table lists the four poles that are used during the experimental evaluation, and specifies their physical attributes. Moreover, the mean absoulte error (MAE) and number of failures are listed.

pole. These features may not be relevant for other tasks, where learning such features end-to-end with the policy, e.g. using autoencoders, may further generalize the proposed framework.

A. Experimental setup

The student tactile control policy is deployed on the physical system without any adaptation, i.e., the policy learned in simulation is directly transferred to the physical hardware. Four different poles are used throughout the experiments. The physical attributes of each pole are listed in Table 1. For each pole, the student control policy is employed until ten successful swing-ups have been achieved. Failures, which are defined to be runs where the pole either falls or the cart is commanded to positions outside of the stroke, are also counted. During each run, the estimated orientation $\hat{\phi}$, as well as the control actions taken are recorded.

B. Supplementary results

Fig. 11 contains a plot for each of the poles. Each plot depicts the mean trajectory of the estimated orientation $\hat{\phi}$ over time for the ten successful runs. Additionally, the sample standard deviation of the ten runs is shown in the shaded blue area. As can be seen, the student control policy yields repeatable behaviour and successfully swings the poles up to an angle of 180° with small offsets of the final angle. The time to reach the upright position is roughly equal across the four poles, with an average of $\sim 0.5\,\mathrm{s}$. Additionally, there is only minimal overshoot.

Next, Fig. 12 shows the commanded cart acceleration and clamping distance for each pole averaged over the ten successful runs. As can be seen, the clamping range for poles of different radii naturally differ. It is important to note that these differences in the clamping distance arise purely from the feedback of the tactile sensor (i.e. the estimated orientation and total normal force), and that the radius is not known to the policy.

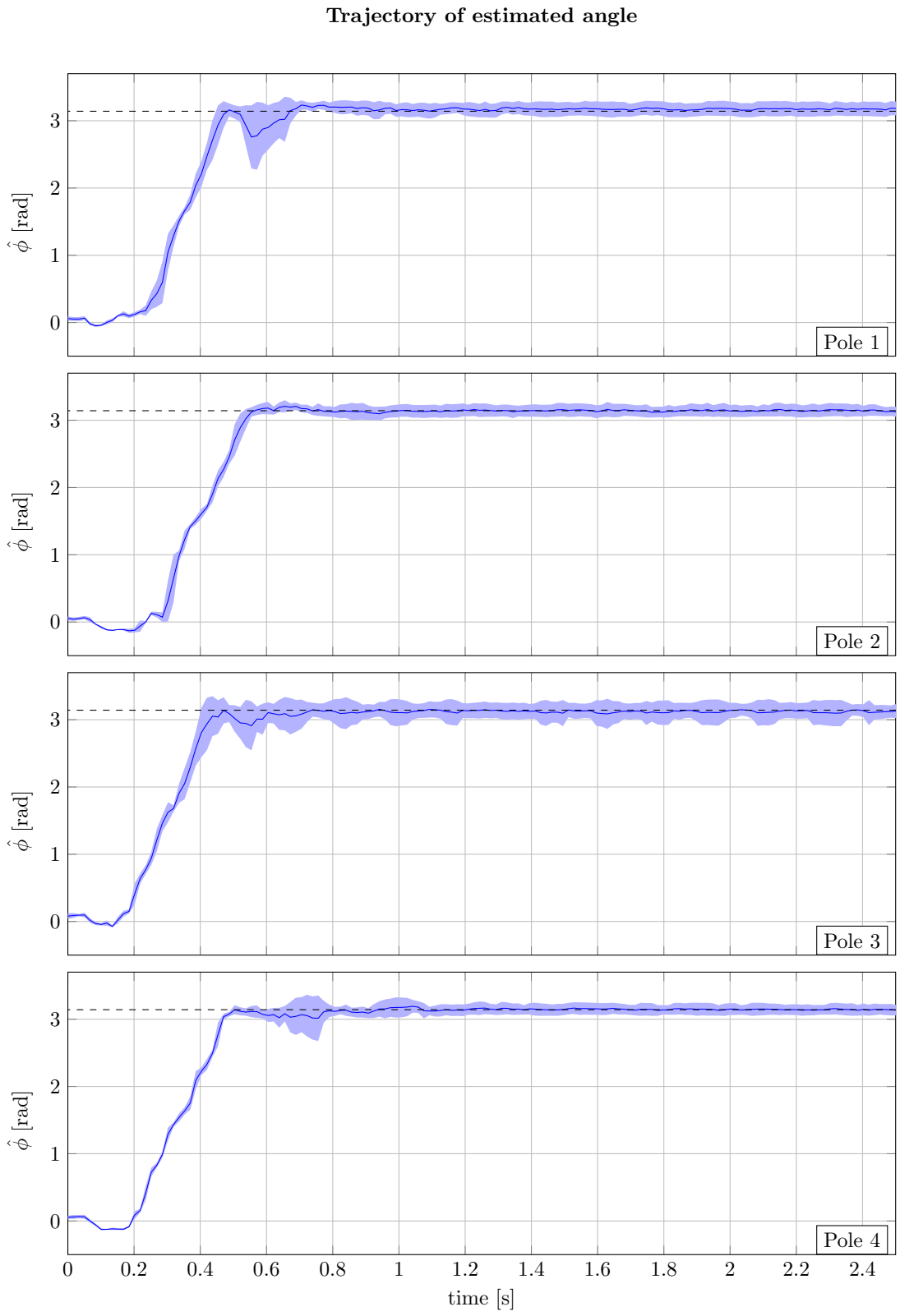


Figure 11. This figure shows the trajectory of the estimated angle $\hat{\phi}$ of the different poles when running the tactile control policy. For each pole, the control policy is run ten times. The resulting mean of the trajectory is shown by the solid blue line, while the shaded blue area depicts the standard deviation of the angle.

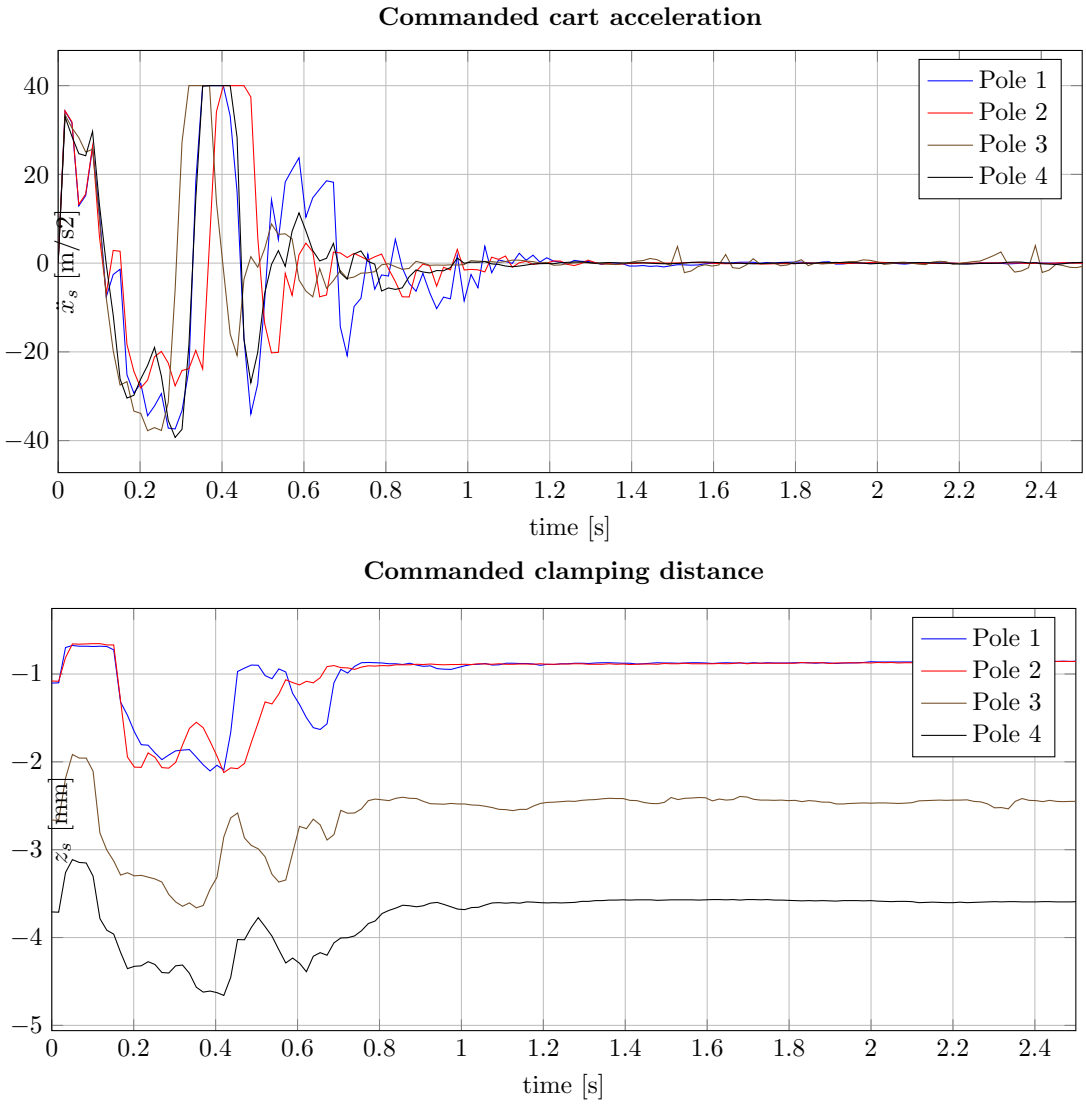


Figure 12. This figure shows the control actions taken by the robotic system for the different poles, i.e. the commanded acceleration and the commanded clamping distance of the tactile gripper. For each pole, the mean values over the ten logged runs are shown. As can be seen from the commanded clamping distance, the difference in the radii of the poles lead to large offsets in the commanded clamping distances. Moreover, even for poles of the same radii (poles 1 and 2), the clamping distance trajectories differ due to the difference in inertia.

Further, even for the two poles that share an identical radius (poles 1 and 2), there are noticeable differences in the commanded clamping distance (at around 0.5 s). This can be attributed to the fact that due to the differences in the moment of inertia, different impulses are necessary to generate a sufficient angular momentum of the pole.

Finally, Table 1 also summarizes the results and states the mean absolute error (MAE) of the final estimated angle $\hat{\phi}$ with a goal of 180°. Averaged over all the successful runs of the four poles, an error of 4.3° is observed. Moreover, the errors are consistent across the different poles. At this point, it is important to note that the estimated orientation

is based on a 20×20-dimensional sensed force distribution, and may thus be noisy and does not necessarily provide ground truth.

It can also be seen that pole 1 experiences a higher failure rate when compared to the other three poles. From observing the trajectories for pole 1, it was seen that a restitution effect takes place once the pole reaches the upright position. In other words, the pole springs back slightly once it reaches the upright position. The control policy then attempts to correct for this behaviour by accelerating the cart (see seconds $0.5 - 0.75$ in Fig. 12). This is also visible in the first plot of Fig. 11, where there is a small downward spike of the estimated angle at $\sim 0.6\,\mathrm{s}$. The high accelerations then occasionally cause a motion that is too jerky, causing the gripper to lose contact with the pole, ultimately leading to failure. Further, the fact that this restitution effect is more pronounced for pole 1, can be attributed to the fact that the pole is relatively short, and thus its center of mass is closer to the point of contact on the tactile gripper. As a result, higher forces are necessary to generate the torques necessary to achieve the desired motion.

Acknowledgments

The authors would like to thank M. Egli and M. Mueller for their contribution to the development of the system.

References

[1] C. Sferrazza and R. D'Andrea, "Design, motivation and evaluation of a full-resolution optical tactile sensor", *Sensors*, vol. 19, no. 4: 928, 2019.

[2] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization", in *IEEE Int. Conf. on Robotics and Automation*, 2018, pp. 1–8.

[3] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating", in *Conf. on Robot Learning*, 2020, pp. 66–75.

[4] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain", *Science Robotics*, vol. 5, no. 47, 2020.

[5] H. Yousef, M. Boukallel, and K. Althoefer, "Tactile sensing for dexterous in-hand manipulation in robotics—a review", *Sensors and Actuators A: physical*, vol. 167, no. 2, pp. 171–187, 2011.

[6] Z. Kappassov, J.-A. Corrales, and V. Perdereau, "Tactile sensing in dexterous robot hands", *Robotics and Autonomous Systems*, vol. 74, pp. 195–220, 2015.

[7] J. M. Romano, K. Hsiao, G. Niemeyer, S. Chitta, and K. J. Kuchenbecker, "Human-inspired robotic grasp control with tactile sensing", *IEEE Trans. on Robotics*, vol. 27, no. 6, pp. 1067–1079, 2011.

[8] Y. She, S. Wang, S. Dong, N. Sunil, A. Rodriguez, and E. Adelson, "Cable manipulation with a tactile-reactive gripper", in *Robotics: Science and Systems*, 2020.

[9] F. R. Hogan, J. Ballester, S. Dong, and A. Rodriguez, "Tactile dexterity: Manipulation primitives with tactile feedback", in *IEEE Int. Conf. on Robotics and Automation*, 2020.

[10] S. Tian, F. Ebert, D. Jayaraman, M. Mudigonda, C. Finn, R. Calandra, and S. Levine, "Manipulation by feel: Touch-based control with deep predictive models", in *IEEE Int. Conf. on Robotics and Automation*, 2019, pp. 818–824.

[11] H. Van Hoof, N. Chen, M. Karl, P. van der Smagt, and J. Peters, "Stable reinforcement learning with autoencoders for tactile and visual data", in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2016, pp. 3928–3934.

[12] A. Church, J. Lloyd, R. Hadsell, and N. F. Lepora, "Deep reinforcement learning for tactile robotics: Learning to type on a braille keyboard", *IEEE Robotics and Automation Lett.*, vol. 5, no. 4, pp. 6145–6152, 2020.

[13] C. Wang, S. Wang, B. Romero, F. Veiga, and E. Adelson, "Swingbot: Learning physical features from in-hand tactile exploration for dynamic swing-up manipulation", in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2020, pp. 5633–5640.

[14] Y. Karayiannidis, C. Smith, D. Kragic, *et al.*, "Adaptive control for pivoting with visual and tactile feedback", in *IEEE Int. Conf. on Robotics and Automation*, 2016, pp. 399–406.

[15] T. Senoo, Y. Yamakawa, S. Mizusawa, A. Namiki, M. Ishikawa, and M. Shimojo, "Skillful manipulation based on high-speed sensory-motor fusion", in *IEEE Int. Conf. on Robotics and Automation*, 2009, pp. 1611–1612.

[16] Z. Ding, N. F. Lepora, and E. Johns, "Sim-to-real transfer for optical tactile sensing", in *IEEE Int. Conf. on Robotics and Automation*, 2020.

[17] Y. Wang, W. Huang, B. Fang, and F. Sun, "Elastic interaction of particles for robotic tactile simulation", *arXiv:2011.11528*, 2020.

[18] Z. Kappassov, J.-A. Corrales-Ramon, and V. Perdereau, "Simulation of tactile sensing arrays for physical interaction tasks", in *IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics*, 2020, pp. 196–201.

[19] J. A. Joergensen, L.-P. Ellekilde, and H. G. Petersen, "RobWorkSim-an open simulator for sensor based grasping", in *41st Int. Symp. on Robotics and ROBOTIK*, VDE, 2010, pp. 1–8.

[20] S. Moisio, B. León, P. Korkealaakso, and A. Morales, "Model of tactile sensors using soft contacts and its application in robot grasping simulation", *Robotics and Autonomous Syst.*, vol. 61, no. 1, pp. 1–12, 2013.

[21] M. Bauza, E. Valls, B. Lim, T. Sechopoulos, and A. Rodriguez, "Tactile object pose estimation from the first touch with geometric contact rendering", in *Conf. on Robot Learning*, 2020.

[22] C. Sferrazza and R. D'Andrea, "Sim-to-real for high-resolution optical tactile sensing: From images to 3D contact force distributions", *arXiv preprint arXiv: 2012.11295*, 2020.

[23] C. Sferrazza, T. Bi, and R. D'Andrea, "Learning the sense of touch in simulation: A sim-to-real strategy for vision-based tactile sensing", in *IEEE Int. Conf. on Intelligent Robots and Systems*, 2020.

[24] M. Ghafoorian, N. Karssemeijer, T. Heskes, I. W. van Uden, C. I. Sanchez, G. Litjens, F.-E. de Leeuw, B. van Ginneken, E. Marchiori, and B. Platel, "Location sensitive deep convolutional neural networks for segmentation of white matter hyperintensities", *Scientific Reports*, vol. 7, no. 1, pp. 1–12, 2017.

[25] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet V2: Practical guidelines for efficient CNN architecture design", in *Proceedings of the Eur. Conf. on Computer Vision*, 2018, pp. 116–131.

[26] C. Sferrazza, A. Wahlsten, C. Trueeb, and R. D'Andrea, "Ground truth force distribution for learning-based tactile sensing: A finite element approach", *IEEE Access*, vol. 7, pp. 173 438–173 449, 2019.

[27] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A toolbox for easily calibrating omnidirectional cameras", in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006, pp. 5695–5701.

[28] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics", *IEEE Robotics and Automation Lett.*, vol. 3, no. 2, pp. 895–902, 2018.

[29] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control", in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[30] K. Weiler and P. Atherton, "Hidden surface removal using polygon area sorting", *ACM SIGGRAPH*, vol. 11, no. 2, pp. 214–222, 1977.

[31] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor", in *Int. Conf. on Machine Learning*, vol. 80, PMLR, 2018, pp. 1856–1865.

[32] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey", *The Int. Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[33] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, *Stable baselines3*, `https://github.com/DLR-RM/stable-baselines3`, 2019.

[34] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures", *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[35] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning", in *Int. Conf. on Artificial Intelligence and Statistics*, 2011, pp. 627–635.

[36] T. Bi, C. Sferrazza, and R. D'Andrea, *Zero-shot sim-to-real transfer of tactile control policies for aggressive swing-up manipulation - experimental report*, `https://arxiv.org/src/2101.02680v2/anc/ExperimentalReport.pdf`, 2021.