DISS. ETH NO. 24863

MULTI-VIEW 3D RECONSTRUCTION WITH GEOMETRY AND SHADING

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZÜRICH

(Dr. sc. ETH Zürich)

presented by

Silvano Galliani

Master of Science (M.Sc.), University of Saarland

born on 05.12.1982

citizen of Italy

accepted on the recommendation of

Prof. Dr. Konrad Schindler, examiner
ETH Zürich, Switzerland

Prof. Dr. Yasutaka Furukawa, co-examiner
Simon Fraser University, Burnaby, Canada

Prof. Dr. Gabriel Brostow, co-examiner
UCL London, United Kingdom

2018

IGP Mitteilungen Nr. 119

Multi-View 3D Reconstruction With Geometry And Shading
Silvano Galliani
Copyright © 2018, Silvano Galliani
Published by:
Institute of Geodesy and Photogrammetry
Swiss Federal Institute of Technology (ETH)
8093 Zürich, Switzerland
All rights reserved
978-3-03837-007-9

To my girls: Flora, Bianca and my boy Bruno. Thanks to my families Galliani and Arata, but a special *grazie* is for Laura, always with me.
To Konrad and all the colleagues who contributed in the creation of a relaxing but productive working environment.
Last but not least to my roommate Wilfried, who did not live to finish his PhD.

SOMMARIO

La richiesta di catturare modelli 3D di oggetti reali è sempre aumentata in passato. Al giorno d'oggi sono numerosi gli eventi che segnalano un interesse sempre più crescente in futuro: effetti speciali generati con il computer sono ampiamente utilizzati e beneficiano molto di questo tipo di dati, le stampanti 3D sono diventate più economiche e potrebbero essere presto presenti in ogni abitazione. La realtà virtuale e aumentata sono due mezzi di comunicazione simili tra loro che rivoluzioneranno la nostra vita quotidiana con lo stesso potere che ha avuto Internet o i telefoni cellulari. Difficile è prevedere il reale impatto che avranno sulla società, quello che è veramente certo è il ruolo centrale dei modelli 3D per le multiple applicazioni che verranno create. Infine, la guida autonoma trasformerà il mercato automobilistico con l'obbiettivo di ridurre il numero di incidenti mortali. Per raggiungere questo obiettivo la visione computazionale fornisce un modo affidabile e veloce per misurare la strada e le auto circostanti. Pertanto, la ricostruzione 3D è stata e sarà uno degli argomenti più importanti della visione computazionale. In questa tesi ci occupiamo del problema di ricostruzione di una scena 3D o di un oggetto partendo da immagini con una data posizione della fotocamera. Proponiamo diversi contributi alla ricostruzione di oggetti in 3D da immagini multiple facendo utilizzo della geometria e della obreggiatura. Innanzitutto ciò che proponiamo è un algoritmo molto parallelo che stima *patch* in 3D sull'oggetto utilizzando una modifica dell'idea Patchmatch: uno schema randomizzato per l'ottimizzazione di una funzione di similarità locale. Lo schema di diffusione che proponiamo consente di utilizzare un numero di thread pari alla metà del numero di pixel. Questo lo rende perfettamente adatto per essere utilizzato su GPU. Infatti noi proponiamo una implementazione efficiente che rendiamo pubblica come software open source. In secondo luogo, proponiamo di imparare una similarità con una rete neurale siamese a rami multipli. La rete risultante è invariante al numero di input al momento della predizione e può essere utilizzata per migliorare il risultato nel caso in cui ci sia rumore sull'immagine di riferimento, come le specularità, se utilizzata all'interno di un algoritmo per la ricostruzione

3D da immagini multiple. Infine proponiamo di completare una ricostruzione 3D da multiple immagini con un modello di ombreggiatura basato sull'apprendimento automatico. Utilizziamo un modello senza supervisione che utilizza una Rete Neurale Convoluzionale che fa regressione sul vettone normale alla superficie usando l'immagine come input. Ci asteniamo dall' utilizzare dati esterni, il nostro approccio è quello di utilizzare un modello che sia su misura per ogni immagine di ogni oggetto, imparando il modello senza supervision (o auto supervisione).

ABSTRACT

The demand to capture 3D models of real-world objects was always increasing in the past. Nowadays multiple are the events that signal even greater interest in the present and future: computer-generated special effects are extensively used and highly benefit from such data, 3D printing has become more affordable and could soon be present in every home. Virtual and Augmented reality are two similar media for communication which are going to revolutionize our daily life with the same power such as internet and mobile phones. It's difficult to predict the real impact on the society, what is sure is the central role of captured 3D models for the various new application that will flourish. Last but not the least autonomous driving is going to disrupt the car manufacturing business by providing self-driving car with the goal to reduce the number of deadly car accidents. To reach that goal, computer vision offers a reliable way to measure the road and the surrounding cars quickly. Therefore, 3D reconstruction has been and still is one of the most critical topics of computer vision. In this thesis, we deal with the problem of reconstructing a 3D scene or object from many colored images with a given camera position, by making use of geometry and shading.

We propose multiple contributions to multi-view 3D reconstruction. First of all, we introduce a massively parallel algorithm that fits 3D surface patches on the object by using a modification from the Patchmatch idea: a randomized scheme for the optimization of a local similarity function. Our new diffusion scheme allows using as many threads as half the number of pixels. This makes it ideally suitable to be used on GPU. Indeed we introduce an efficient implementation that is released to the public as open source software. Secondly, we use deep learning to learn a similarity function across multiple image patches with a multi-branch Siamese neural network. The resulting network is invariant at test time to the number of input and can be used to overcome noise on the reference camera, such as specularities, when used inside a multi-view-stereo pipeline. Finally, we complement a 3D reconstruction from multi-view stereo with a learned shading model. We use an unsupervised Convolutional Neural Network to regress the surface normal

from an input patch. We refrain from using external data; our approach is to use the view-specific input normal originated from our initial reconstruction as training, effectively making use of unsupervised — or self-supervised — learning.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

The main goal of computer vision is to automate to the process of understanding the world by using images. By definition images are created on the two-dimensional realm, where the sensor perceives the appearance of a three-dimensional object. But the aim is to invert this creation process by inferring the original shape that explains the image. In other words, cameras convert the 3D world to a 2D representation, while computer vision wants to invert this process and obtain a 3D knowledge of the object.

Unfortunately, this is an ill-posed for multiple factors.

First of all **perspective projection** of the object on the image distorts the original 3D scene, making it impossible, except for simple, known shapes, to reconstruct the original 3D object from a single image.

Secondly, multiple objects aligned on the line of sight between object and camera, also known as **occlusion**, destroy valuable information for the final understanding of the object.

In the same way human visual system works with two eyes, to improve the power to reconstruct and disambiguate the understanding of the world it is common to use multiple images captured simultaneously but from different point of views. The simplest approach consists of using the so-called (binocular) **stereo reconstruction** which analyzes two images to measure the scene. The displacement of the cameras, or distance of our eyes, allows to infer the 3D position of the scene by triangulating rays on the image points that corresponds to each other.

It is clear that, when using more than 2 images of a scene, it is possible to improve not only accuracy by triangulating multiple rays at the same time, but also the completeness of the reconstruction by checking the coherence of redundant observations. This is known in the literature as **multi-view stereo reconstruction**. Continuing the parallel with the human visual system, it is true that we have only two eyes, but our capability to move our head helps to clarify our perception of the world in case of doubt, effectively employing more than two images to perceive the scene, even if not captured simultaneously.

Along with stereo vision, the human visual system employs other passive cues to perceive depth. We exclude here active cues of our brain which are based on semantic and knowledge interpretation. *E.g.*, experience informs us that cars lie on top of a street and constrains their shape and dimension.

A specific cue complementary to stereo reconstruction is **shading**. It is procured by the joint interaction of shape of the object, its material properties — its color and how they reflect light — and lights illuminating it. Shading is one of the most powerful hint for our brain that helps to understand the world we live in. It does not come as a surprise that painters and illustrators utilize it to give the impression of depth on a two dimensional painting, see figure 1.

Reconstruction methods based on lighting such that Shape from Shading (SfS) try to revert the rendering process: from shading information they derive the object properties that explain its representation.

Considering that color influences the appearance of objects, textured regions cause serious problems for shading based reconstruction methods, since the decomposition of the observed brightness into color and depth is ambiguous. Instead, (multi-view) stereo relies on the abundance of unique texture on the object which facilitates the matching of corresponding 2D images of the same 3D point, for successive triangulation and extraction of depth information.

The opposite is true for homogeneous regions of the image: they confuse stereo reconstruction methods, but at the same time they are the perfect scenario for shading based reconstruction methods.

Given the advantage and drawback of both methods, in this thesis we propose to combine multi-view stereo and shading to improve the quality of the 3D reconstruction.

1.1 GOAL OF THE THESIS

This dissertation considers the reconstruction of 3D surfaces from multiple pictures of the same scene captured from different points of view. The main goal here is to obtain the 3D

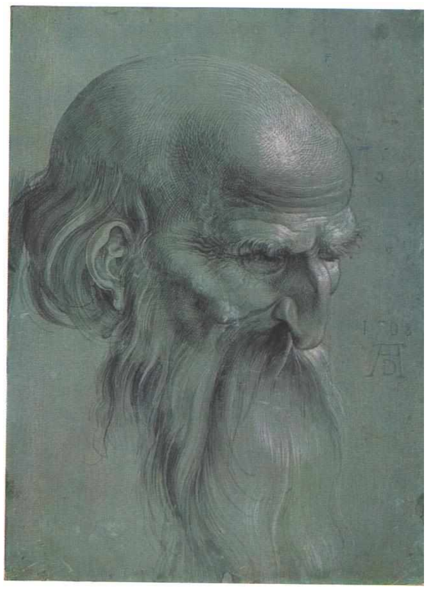


Figure 1: Painting by Albrecht Dürer showing how shading can be used to convey the depth information of an object. Source: Wikimedia Common.

information of an object using only the information inside the images and no additional restrictions or assumptions. We only assume that the position and parameters of cameras is either given or can be easily computed from Structure from Motion (SfM) methods. The corner cases where SfM would fail (lack of texture or presence of a symmetric object in the scene) might indicate a difficult dataset to compute even for successive stereo methods to be applied on the data.

The focus is not restricted only on geometric-based methods such as **Multi-View Stereo Reconstruction** (MVS) but also on additional shape reconstruction techniques such as **Shape from Shading** (SfS) which naturally complement classic feature based triangulation.

This thesis investigates the creation of a symbiosis between Multi-View Stereo Reconstruction and Shape from Shading. The two methods can be fused in such a way that SfS helps to correct the reconstruction in parts where MVS fails and vice versa. During the exploration of this connection we will tackle the following topics.

- A new Multi-View Stereo matching method. We will make use of recent advances in parallel computing to devise an algorithm, released as open source software with name Gipuma, specifically tailored for accuracy, speed and low memory consumption, see Chapter 4.

- We will analyze how to harness the potential of recent deep learning techniques to create a new similarity function across multiple images. The new approach improves accuracy and circumvents occlusion problems by considering the reference camera on par with the other views. See Chapter 5.
- We will explore how to create a reflection model which avoids any type of assumption of the scene, as opposed to what it is common from classic SfS method. It is an self supervised — or unsupervised — discriminative model specifically tailored to each object and view. The proposed shading model can be intertwined with classic stereo based methods to obtain an improved reconstruction, see Chapter 6.

1.2 RELEVANCE TO SOCIETY AND ECONOMY

The reconstruction of the 3D position of objects has always played a central role inside computer vision since more than 40 years. Multiple methods have been developed to improve the accuracy and performance of reconstruction methods.

But it is only recently, perhaps in the last 5 years, thanks to the maturity of the research and the increase of computational power, that computer vision started to make a great impact on our society. I will sketch here in particular how multi-view 3D reconstruction empowers emerging areas of society and economy. However, I will refrain from commenting on more classic applications such as visual effects for games and movies, digital mapping and land surveillance since they are not mainly based on 3D imaging.

1.2.1 *Autonomous Driving*

Autonomous driving has the potential to reduce one the most significant sources of disability and mortality worldwide: traffic accidents. That's why most major vehicle manufacturers are investing in research and development departments to build a car that exceeds or at least equates human performance. The factors into play to reach this goal are multiples, but scene understanding by means of images plays a central role.

A car must interpret the world it is living in to accelerate or steer in a particular direction in order to reduce the risk of potential danger for the passengers both in the same and on other cards. Accurate and fast depth measurements are crucial to account for maneuvers with abrupt changes of direction. This is not a trivial task first of all due to variable weather and lighting conditions that potentially interfere with the 3D reconstruction. Secondly, the strict timing requirement of the driving system makes it even more challenging. Nevertheless, current trends in depth reconstruction indicates that the technology is mature enough to employ vision inside autonomous cars.

1.2.2 *Virtual Reality*

Virtual reality, another growing business, is seen by many as the future revolution in our lives on par with the scope of personal computers, internet and mobile phones. This type of technology has the potential to disrupt our daily life in such an exponential way that becomes difficult to predict. The multi-million investments are the demonstration of how big companies compete to prevail in this market.

The immersive virtual models where the users interact has to be as similar to our reality as possible to create a lifelike experience. Rather than rely on artists to manually craft new and different 3D models, it is cheaper to replicate our real-world by automatic vision (or photogrammetric) measurements.

1.2.3 *Augmented Reality*

Augmented reality is symmetric and complementary to virtual reality. It superimposes virtual objects over our world and *augment* it with additional information. It also has the potential to become a revolutionary technology, but it appears more mature to be used for practical real-life applications than Virtual Reality. The factors limiting its mass deployments are indeed much lowers. The rendered objects are only a small fraction of the visible field of view, thus reducing the computation requirements and need for realism of the virtual object.

Also in this case, the accurate measurement of the real 3D position of the surrounding objects is crucial. By placing virtual object seamlessly inside our real-world we can partially fool the human brain that would not be disturbed by the introduction of a non-real part.

Indeed, the experience of a joint real/virtual reality would get compromised in case the surrounding world is not correctly measured. That's where vision based reconstruction plays a central role.

1.2.4 *3D printing*

With 3D printing, a recent technology, anyone with a virtual 3D model of an object can create itself a physical copy of the model at hand. It empowers users to customize products, to create new objects for personal use, or potentially to create autonomously missing component of an existing object. *E.g.*, it could duplicate a broken screw that has to fit on its nut.

For this type of application, measurement via images is crucial, it allows to replicate a physical object on the computer, its physical properties such as shape and also color. For this type of application, the high quality and fidelity of the reconstruction are top priorities while speed of the reconstruction is pushed aside. We are willing to wait more to obtain a replica of an object, but we wouldn't accept compromise on the quality of the reconstruction.

1.3 CHALLENGES

We will sketch here the major challenges to face when dealing with Multi-View Reconstruction.

While at first the problem might look straightforward, there are multiple difficulties originating first of all from the object shape and secondly from the complexity of the interactions between light sources, surface materials and object properties. The main underlining assumption in MVS is that a "point" in space appears similar from its projection on multiple cameras. It is often true, except for three specific scene properties.

1.3.1 *Specular Surfaces*

Strongly specular objects violate our assumption: the object appearance depends on the angle between camera view and light direction, meaning that specularity will be observed on one image, but not the others, see Figure 30.

The simpler specular model is the *Phong model* [96], where it becomes evident from its formulation (note V) the dependence of the final image appearance w.r.t. the view position:

$$I_{\text{Phong}} = k_d(L \cdot N) + k_s(V \cdot R)^{\alpha} \tag{1}$$

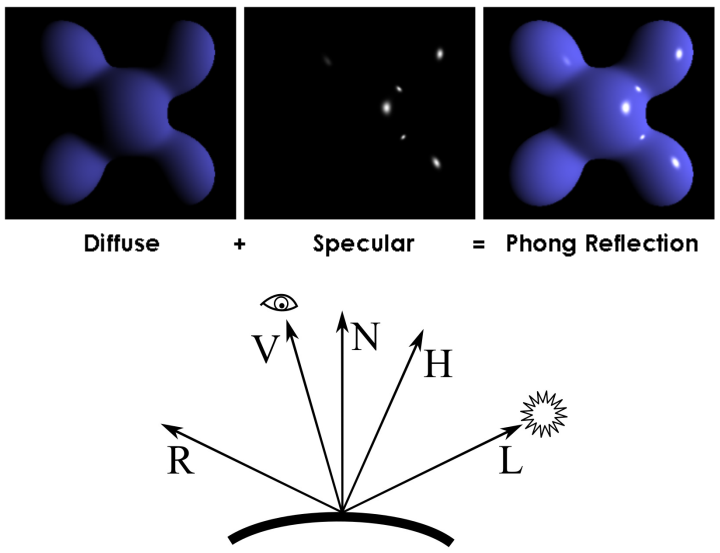


Figure 2: Top: Example of rendering of an object with specular components. Bottom: Visualisation of vectors contributing to the rendering of a Phong model. Source: Wikimedia Common.

here I_{Phong} is the image, k_d the *albedo* or intensity of the diffuse component, k_s the intensity of the specular component, L the light vector, N the vector normal to the surface, V is the view vector and R the reflection ray, see figure 2.

In order to guarantee an invariance of aspect w.r.t. the view point, it is generally assumed that the scene is mostly Lambertian, where the image appearance is assumed to not change with respect to the camera position:

$$I_{\text{Lambert}} = k_d(L \cdot N) \quad . \tag{2}$$

In practice, MVS, as opposed to two-view stereo reconstruction, benefits from the redundancy of evidence for 3D triangulation which mitigates the artifacts obtained by specularities. This assumption is validated empirically by the results obtained by MVS on real world datasets where the reflectance of the majority of surfaces is mostly Lambertian. Nevertheless, specularity is still a major source of error for stereo reconstruction methods.

1.3.2 *Homogeneous surfaces*

Another big source of error, often the most problematic, is the lack of texture on the scene, which prevents the similarity function to disambiguate the image of the same point in multiple cameras, note the holes in the reconstruction of Figure 44. This is often a problem on almost planar surfaces without texture. Indeed, on some special cases, *e.g.* on curved surfaces, it is the shading information which provides enough evidence to match the appearance of the surface point across cameras, but only in case the lights do not vary across cameras.

1.3.3 *Occlusion*

We observe occlusion when part of the object from one view is blocked from the line of sight of another view. This is one of the worse case scenarios: if **data is completely missing** on one image it is unlikely to match it on the other image, unless the occluded region is small, or the stereo algorithm is able to accumulate evidence from the other views. Occlusion problems is visible as an artifact on occlusion boundary, where, depending on the type of matching function and approach used, the shape of the object gets distorted in different ways.

1.3.4 *Solution to Problems*

To properly compensate for this type of error, the usual solution is to **regularize** the surface in a global context by imposing smoothness from nearby correct points. *E.g.*, a solution could be found by fitting a plane, or a surface which interpolates nearby points but minimizes the gradient of the resulting solution. The ideal scenario is to always rely on the data, if available, and avoid imposing arbitrary smoothness properties that might create an incorrect reconstruction. That is what we will present in Chapter 4, where we will make use of the redundant observations and refrain from "closing" holes in the reconstruction. In Chapter 6 we will see how it is possible to derive an "interpolator", both object- and view- specific, which is able to complete the surface in areas of homogeneous colour.

In Chapter 5 we will show how our proposed similarity function is able to correct for view-specific problems such as specularities and partially also for occlusion.

1.4 CONTRIBUTION

The work presented in this thesis is based on the work presented in Galliani *et al.* [41], Galliani and Schindler [42] and

Hartmann *et al.* [53]. In the following I will anticipate the most important contributions.

- **Gipuma: Massively Parallel Multi-View Stereo Reconstruction** (Chapter 4, [41]).
  We present a new, massively parallel method for high quality multiview matching. Our work builds on the Patchmatch Stereo idea [11]: starting from randomly generated 3D planes in scene space, the best-fitting planes are iteratively propagated and refined to obtain a 3D depth and normal field per view, such that a robust photo-consistency measure over all images is maximized. Our main novelties are on the one hand to formulate Patchmatch Stereo in scene space, which makes it possible to aggregate image similarity across multiple views and obtain more accurate depth maps. And on the other hand a modified, diffusion-like propagation scheme that can be massively parallelized and delivers dense multiview correspondence over ten 1.9-Megapixel images in 1.5 seconds, on a consumer-grade GPU. Our method uses a slanted support window and thus has no fronto-parallel bias; it is completely local and parallel, such that computation time scales linearly with image size, and inversely proportional to the number of parallel threads. Furthermore, it has low memory footprint (four values per pixel, independent of the depth range). It therefore scales exceptionally well and can handle multiple large images at high depth resolution. Experiments on the DTU and Middlebury multiview datasets as well as oblique aerial images show that our method achieves very competitive results with high accuracy and completeness, across a range of different scenarios.

- **Learned Multi-Patch Similarity** (Chapter 5, [53]).
  Estimating a depth map from multiple views of a scene is a fundamental task in computer vision. As soon as more than two viewpoints are available, one faces the very basic question how to measure similarity across in more than 2 image patches. Surprisingly, no direct solution exists, instead it is common to fall back to more or less robust averaging of two-view similarities. Encouraged by the success of machine learning, and in particular convolutional neural networks, we propose to learn a matching function which directly maps multiple image patches to a

scalar similarity score. Experiments on several multi-view datasets demonstrate that this approach has advantages over methods based on pairwise patch similarity.

- **Self-Supervised Normal Prediction for shading based multi-view stereo refinement** (Chapter 6 [42]).
  We present a multi-view reconstruction method that combines conventional multi-view stereo (MVS) with appearance-based normal prediction, to obtain dense and accurate 3D surface models. Reliable surface normals reconstructed from multi-view correspondence serve as training data for a convolutional neural network (CNN), which predicts continuous normal vectors from raw image patches. By training from known points in the same image, the prediction is specifically tailored to the materials and lighting conditions of the particular scene, as well as to the precise camera viewpoint. It is therefore a lot easier to learn than generic single-view normal estimation. The estimated normal maps, together with the known depth values from MVS, are integrated to dense depth maps, which in turn are fused into a 3D model. Experiments on the DTU dataset show that our method delivers 3D reconstructions with the same accuracy as MVS, but with significantly higher completeness

1.5 OUTLINE OF THE THESIS

Let us now provide a synopsis of this thesis.

Chapter 2 introduces the literature of Multi-View Stereo reconstruction related to this thesis including two main research directions. First of all we will introduce the related works on classic geometric based method for the 3D reconstruction from multiple images. Then we will present extensions making use of shading or assumption on surface normal.

Chapter 3 is pertinent to the necessary technical background needed to understand the following chapters. A complete and self-contained description of the major theoretical background would be outside the scope of this thesis. We will tackle the necessary topics to be able to understand the subsequent chapters.

We will start from camera model and stereo geometry to continue with Convolutional Neural Networks (CNN) and then we will conclude by listing multiple MVS datasets and benchmarks used with this thesis along with their properties and shortcomings.

In chapter 4 we will introduce a novel algorithm for Multi-View stereo specifically tailored to parallel computing architectures. Experiments on Middlebury and DTU dataset will validate the quality of the method. Chapter 5 concerns the computation of the similarity function across multiple views. Instead of computing the pairwise similarity of the reference camera w.r.t. the other views we propose to learn the multi-patch similarity by means of a deep convolutional network. Experiments will demonstrate the robustness of this method to noise on the reference camera such as specularities. Subsequently, in 6 we will introduce a discriminative shading model to complement and improve an initial stereo reconstruction from multiple images. Experiments on the DTU dataset show that our method delivers 3D reconstructions with the same accuracy as MVS, but with significantly higher completeness. Finally, chapter 7 closes this work by summarizing contents and giving some comments along with an outlook on potential improvements.

2 RELATED WORKS

An enormous body of literature exists on multi-view stereo reconstruction. Even if stereo methods dates back to at least 1974 [51], several new methods are proposed each year, as demonstrated by the method listed in popular binocular or multi-view stereo datasets [44, 102, 104, 106].

A big number of new multi-view stereo method are published every year, making difficult to compile a complete survey of existing multi-view stereo. Even restricting to multi-view stereo methods, it would be a formidable task to compile a complete survey of existing methods.

In this chapter, we try to review the main methods closely related to this thesis. In the first part, we introduce "classic" multi-view stereo reconstruction making use of image matching and ray triangulation to obtain the final reconstruction.

Then we introduce extensions which makes use of additional information or assumption by using normals or shading information to guide or refine the final reconstruction.

2.1 MULTI-VIEW STEREO RECONSTRUCTION

2.1.1 *Local vs. global matching.*

Successful image matching has to strike a balance between photo-consistency of the corresponding image locations and regularity (typically piecewise smoothness) of the underlying surface.

Early models usually were *local*, meaning that the correspondence computation at a given location depends only on a local neighborhood. Local methods range from simple block matching to more sophisticated approaches that avoid a strong fronto-parallel bias, either by directly warping the image to a common plane [13, 31, 43], or using an oriented matching window that adapts to the surface geometry [11, 29]. Moreover, to avoid the characteristic fattening of foreground objects, it is common to adapt either the window shape [39, 71] or the weight of pixels within the window [125] at (putative) depth discontinuities.

Later research attempted to include the correlations induced by the smoothness prior in a more principled way, which leads to *global* methods that approximately maximize an objective defined over all pixels, usually via discrete labeling, *e.g.* [32, 57, 81] or variational inference [80, 97].

Nowadays photographic images, even on mobile phones, routinely have on the order of 10 million pixels. Therefore, there is a need for matching algorithms whose complexity is low — ideally at most linear in the number of pixels. At the same time, the large image dimensions also call for algorithms that are memory-efficient, especially in the multiview case, where evidence of multiple images is exploited to create the correct match. Consequently, there has been a renewed interest in local matching algorithms. In spite of their simplicity, modern local matchers [11, 37, 99] are accurate enough to compete with their global counterparts, as demonstrated for example by the DTU [66] and KITTI [45] benchmarks.

2.1.2 *Local multiview methods.*

In their seminal work, Okutomi and Kanade [94] accumulate Sum of Squared Difference (SSD) cost values from different stereo pairs in a set of multiple images and select the depth with the lowest cumulative cost. The plane-sweeping method [22] is an early example of true multiview matching. Evidence from multiple images is accumulated on a plane that moves through the scene space along its normal. For every cell on the plane, the position with the highest support is chosen. More recently Gallup *et al.* [43] have proposed to align the plane to the dominant orientation in the scene. Hu and Mordohai [63] also start from plane-sweeping, and carefully propagate the uncertainty to exploit it during the subsequent fusion of multiple depth maps.

Furukawa and Ponce [37] relax the requirement to find a correspondence for every single pixel. Instead, they start from sparse, reliable seed patches and iteratively grow the set of point matches from there, to obtain a quasi-dense oriented point cloud. The method introduces several heuristic filters and delivers quite impressive results. Tola *et al.* [116] directly address the problem of high-resolution image sets by matching a fast descriptor between pairs of images over the epipolar line and reconstructing only points with a unique response. Campbell *et al.* [15] explicitly address the problem of ambiguous matching by considering multiple depths per point and including an unknown state in their MRF optimization.

Many authors follow the philosophy of Kang *et al.* [72]: to sidestep occlusions and aliasing along boundaries, select only a best-matching subset of all available views (in the original work 50%); to preserve discontinuities and cope with untextured areas, start with a small window around a pixel and dynamically increase it if there is a lack of texture.

2.1.3 *Points vs. surfaces.*

Multi-view stereo methods can be classified according to which representation they are based on, following the taxonomy of Seitz et al. [106]. In particular, the 3D scene can be represented by voxels, level-sets, polygon meshes, or depth maps. In this context it should be emphasized that depth maps are still a point-wise representation — triangulating every pixel in a depth map leads to a 3D point cloud, similar to those generated with RGBD sensors or laser scanners. On the contrary, the three other representations all must solve (at least implicitly) the additional step from the point cloud to the underlying surface. It may be useful for many applications but is a considerably harder and less well-defined task. Moreover, some application domains like industrial metrology or surveying, in fact, prefer 3D point clouds as a primary product. In our work we mainly aim to recover depth maps, respectively oriented point clouds. We see surface fitting as a subsequent step that is largely independent of the matching — in fact the most popular approaches [65, 75, 83] are rather agnostic about the preceding matcher, and we found the widely used Poisson method [75] to work well for our point clouds.

2.1.4 *Exhaustive vs. randomized search.*

Typically, matching algorithms require a large amount of memory, because they keep track of the cost associated with *every* possible disparity value, in order to select the most suitable one, *e.g.* [32, 57, 71, 99]. Note that for a fixed depth range the number of observable disparities grows linearly with the image resolution, too. A recent exception from the strategy of "comparing all possible disparities" is *PatchMatch Stereo* [11]. That method adopts a randomized, iterative algorithm for approximate patch matching [7], which allows one to quickly find a good solution within a vast search space without having to browse through all possibilities. The resulting low memory requirements (independent of the disparity range) make *Patchmatch Stereo* well-suited for large images or memory-constrained environments, including implementation on GPU which modify the original sequential propagation scheme [5, 6, 55, 134]. Zheng *et al.* [134] employ the Patchmatch propagation scheme for multiview reconstruction, but without considering slanted surfaces. Their focus lies on view selection when aggregating evidence over multiple cameras. A probabilistic graphical model serves to jointly address view selection and depth estimation. There are three other method that runs Patchmatch Stereo in scene space. The first [108] is applied only to pairwise stereo matching. The second [56] extend the Patchmatch scheme with an additional Kanade-Lucas-Tomasi (KLT) step to avoid unnecessary sampling iterations. Additionally, the author alternates the reconstruction with refinement of the camera position to improve the accuracy. The third [103] extend Zheng *et al.* [134] with direct estimation of slanted patches and the inclusion of a pixel-wise view selection in the graphical model.

2.1.5 *Patch Similarity Learning.*

With the rise of machine learning for computer vision problems, it has also been proposed to learn the similarity measure for (two-view) stereo. Early work still relied on hand-tuned descriptors such as SIFT, the learning served to "distort" the descriptor space so that nearby false matches get pushed apart and the distance becomes more discriminative [127]. The advent of deep learning suggested that the bottleneck might be the descriptors themselves rather than the distance metric, so it was proposed to learn similarity directly from raw images [131]. Closely related work started from separate steps for descriptor learning and metric learning, and unified them to effectively obtain a direct similarity prediction from raw image data [49] as well. An extensive study of similarity measures based on different CNN architectures is presented in [130]. That work also showed that CNN-based similarities outperform both classical descriptor spaces like SIFT [89] and other learned descriptors such as [112]. Another strategy is to learn patch descriptors, but freeze the distance measure used to compare them. The recently proposed LIFT descriptor [111, 124] is learned with the help of a Siamese network, using a loss function that ensures that descriptors of matching patches end up having low Euclidean distance, whereas non-matching descriptors have not. The learned output is a 128-dimensional descriptor vector which corresponds to the size of a SIFT descriptor vector [89] so that LIFT can serve as a drop-in replacement for SIFT or similar hand-coded descriptors in existing matching pipelines.

Yet, the learned descriptors still share the limitation of most two-view stereo methods, that similarity is measured only for image pairs, as a distance in descriptor space.

2.2 NORMALS AND SHADING IN MULTI-VIEW STEREO

2.2.1 *Normals in Multi-View Stereo.*

Many multi-view stereo methods only estimate depth, *e.g.* [22, 94, 116]. If normal vectors are required, they are found in post-processing by fitting local tangent planes to the point cloud [59, 93]. There are however a number of MVS methods that explicitly reconstruct the local tangent plane as part of their internal parametrization, and thus directly deliver surface normals on top of depth maps (respectively, 3D points). Notable examples include the well-known PMVS method [37], as well as the multi-view variant [41] of the PatchMatch stereo algorithm [11]. Methods that directly deliver normals at the reconstructed surface points naturally lend themselves to our problem. We use [41], on the one hand for its computational efficiency, and on the other hand because it provides an explicit parameter to trade off completeness *vs.* accuracy and ensure sufficiently clean training normals.

There are also methods which from the beginning constrain MVS reconstruction with strong a-priori assumptions about the surface normal. *E.g.*, Zeisl *et al.* [132] focus on indoor scenarios consisting only of horizontal floor and ceiling planes connected by vertical walls. Furukawa *et al.* [38] go even further and assume a Manhattan world [24]. At the extreme end of the spectrum (though somewhat outside the scope of our work) come model-based methods, which align the images with an existing 3D template of the object and reconstruct by deforming the template to better fit the geometric or photometric evidence, *e.g.* [77, 119].

2.2.2 *Shading Cues in Multi-View Stereo.*

The first attempts to combine multi-view geometry and shading for 3D reconstruction date back at least 30 years [10]. Since then, the topic has been somewhat overshadowed by the development of pure stereo, respectively multi-view matching, but has received constant attention [25, 36, 101]. The complex interplay between surface orientation, light sources, and surface BRDFs proved difficult to handle outside the lab, and most works focus on one of these components. Wu *et al.* [121] assume a Lambertian surface but consider general illumination, approximating the incoming illumination with spherical harmonics. Jin *et al.* [70] propose a joint variational framework for the estimation of shape, normal and a single light source, assuming a Lambertian surface with piecewise constant albedo. Haines and Wilson [48] integrate information from shading and stereo via belief propagation to estimate fine surface details. Beeler *et al.* [8] detect and eliminate ambient occlusion to improve surface estimation. Langguth *et al.* [85] combine a shading term on image gradient and stereo term in a joint optimization framework. Mauer *et al.* [92] use a variational framework to jointly estimate depth, illumination and albedo.

2.2.3 *Surface Normal Estimation.*

A number of recent works have posed surface normal prediction as a machine learning problem. Fouhey *et al.* [33] mine for distinctive, repeatedly occurring shape and appearance primitives in indoor RGB-D data, and match those primitives to new images to obtain a normal map. Later that method was augmented with shape priors for rooms and an explicit model of crease edges [34]. Ladicky *et al.* [84] directly predict normals from image features extracted in a pixel's neighborhood. They turn normal estimation into a classification problem, by clustering the normals to a discrete set of directions on the unit sphere and interpolating between neighboring directions. Instead, Eigen and Fergus [30] learn a direct regression from image to normal (alternatively also to depth or semantic label) with a multi-scale convolutional architecture.

These methods are related to ours in that they pose normal estimation as a learning problem, and in some cases also use CNNs as regression engine. Beyond this technical similarity, there are however two fundamental differences. On the one hand, our model is more specific w.r.t. illumination and reflectance: we do not learn a generic model that is supposed to cover the shading behavior of "the world", or at least of an entire dataset; rather we rely on MVS to generate sparse training data tailored to the specific image, such that for that image the prediction is more accurate, while no external training data is needed. On the other hand, our model is more generic w.r.t. geometry. We rely only on the local shading and the position in the image, but do not depend on the presence of a few vanishing directions or recurrent geometric primitives (such as for example those present in the NYU2 Dataset [110]).

Richter and Roth [100] also relax the requirement for external training data and instead use synthetic training data. They assume knowledge of the object's silhouette in the image. The distance from the silhouette is used to guess a rough initial normal map, which in turn serves to derive a quadratic approximation of the reflectance map and relight the synthetic training data appropriately.

2.2.4 *Normal extrapolation from MVS.*

Few authors have explored the idea to use an incomplete cloud of MVS points as reference for normal prediction. Xu *et al.* [123] seemingly also use the appearance around known points/normals, together with smoothness of the normal field, to fill holes in an image-based surface reconstruction. Unfortunately, no details are given in their paper. Ackermann *et al.* [1] use MVS to bootstrap photometric stereo. Instead of directly modeling lighting and reflectance, they extract per-pixel material coefficients at the MVS points and predict unknown normals by minimizing the photometric differences to the known points.

2.2.5 *Integrating Normals to Surfaces.*

Shading-based methods in most cases estimate normal vectors, which still need to be integrated to surfaces. Reconstructing a function from known gradients is a classic problem in computational geometry as well as in computer vision. Perhaps the most popular method, already employed by Horn and Brooks [60], is to solve the Poisson equation that arises as a necessary condition in variational least-squares reconstruction. In this thesis we also follow this standard approach. It has also been attempted to replace the least-squares error function by more robust norms to improve the robustness to outliers [2]. Some authors prefer to use the computationally more efficient eikonal equation [40, 58]. Further approaches include integration in the frequency domain [35], which is limited to dense vector fields; and direct line-by-line integration, which only works for noise-free data [122].

Few are the work that approach the direct integration of surface normals coming from multiple directions. The first in the literature addressing the problem were Chang *et al.* [17]. They use level sets to solve the Partial Differential Equation (PDE) resulting from the solution of a variational formulation and apply it to multi-view photometric stereo. Weinmann *et al.* [120] formulate the problem with a variational approach to reconstruct mirroring objects.

3 TECHNICAL BACKGROUND

The purpose of this chapter is to familiarize the reader with the concepts and notation used in the rest of this thesis. To this end, we start by describing the modeling of image generation, from the camera to the final image. Later we will sketch the basic concepts of 3D stereo reconstruction: search for correspondence, 3D reconstruction, homographies. We will continue by introducing Convolutional Neural Networks (CNN): basic components, insights, and optimization. Finally, we will conclude by listing multiple MVS datasets and benchmarks used with this thesis, along with their properties and shortcomings.

3.1 IMAGE GENERATION

In this section, we seek to provide a basic understanding of a camera and how it provides a mapping between the 3D world and the 2D image. For the sake of simplicity the usual model employed makes use of many simplifications to make the problem tractable. First of all, we assume the image is free from distortions caused by lenses, such as barrel or pincushion distortion. Minor geometric distortion, such as skewness of a pixel, or uneven pixel dimensions can be incorporated inside the projection model in the form of an intrinsic matrix which can be estimated during calibration. It is a well-known solved problem in computer vision; in fact, multiple methods can calibrate a camera and correct for lens distortions, *e.g.*, Heikkilä and Silvén [54], Claus and Fitzgibbon [20]. In this section, we will only sketch the basic concepts and notations, for a much more detailed and complete description of projective geometry we refer the reader to Hartley and Zisserman [52], from which the image generation and reconstruction part are taken.

3.1.1 *Pinhole Camera*

The most convenient way to represent a projective camera is to use linear algebra utilizing *homogeneous coordinates* which denote points on the image plane using the projective space P^2. In this space, a point $\mathbf{x} = (x, y)^T$ is represented as $(x, y, 1)$

which is also equivalent to (x, y, k) for $k \in R, k \neq 0$. Points with $k = 0$ represent points at infinity and do not exist in the Euclidean space.

In the same way we introduced projective space in 2D, we can do it in 3D. The projective space P^3 include all the point of the Euclidean 3D space plus the one at infinity. Similarly, as before, a 3D point $\mathbf{X} = (x, y, z)$ is represented as $(x, y, z, 1)$ or any equivalent representation (x, y, z, k) for $k \in R, k \neq 0$. For the sake of clarity, it is assumed that $k = 1$ in case an equation demands a homogeneous coordinate. We will see later how homogeneous representations of points in 2D and 3D allow us to model the pinhole camera as a linear mapping.

The goal of a camera is to map, or *image*, the 3D world onto a 2D plane, where the sensor lies. The most simple, and popular model used to represent this mapping is the **pinhole camera model**. It consists in an abstraction of a real camera but which preserves the main geometric relations between the 3D world and the final image.

It is commonly assumed that world and camera coordinates coincide. Additionally, the camera center **C** is placed at the coordinate origin, and the direction of the optical axis of the camera is the positive Z-axis. In case multiple cameras are used at the same time, the coordinate origin is at the camera center of the reference camera.

3.1.2 *Image Projection*

The **pinhole camera** performs a mapping between the 3D point $(X, Y, Z)^T$ to the point on the image plane $(fX/Z, fY/Z)^T$: a mapping from the Euclidean space R^3 to the Euclidean space R^2. The *focal length* f describes the distance of the image plane $Z = f$ to the camera center and is directly related to the *field of view* or *fov*.

When homogeneous coordinates represent world and image points the central projection is expressed very simply by a linear mapping between homogeneous coordinates:

$$\begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3}$$

Please note that equation (3) assumes that the origin of the image planes is at the principal points. For historical reasons in practice the origin lies at the bottom left of the image plane; therefore the mapping becomes:

$$\begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{pmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{4}$$

where $\mathbf{p} = (p_x, p_y)$ are the coordinates of the principal point in the image.

The full map of the pinhole camera model is almost complete, what is missing is the conversion between Euclidean coordinates and image coordinates and a scaling factor taking into account the possibility to have non-square pixels. Therefore, the projection reads as:

$$\begin{bmatrix} \alpha_x X + Ys + Zq_x \\ \alpha_y Y + Zq_y \\ Z \end{bmatrix} = K\mathbf{X} = \begin{pmatrix} \alpha_x & s & q_x & 0 \\ 0 & \alpha_y & q_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{5}$$

where $\alpha_x = fm_x$ and $\alpha_y = fm_y$ represent the focal length of the camera in terms of pixel dimensions in the x and y direction. Additionally, the added parameter s is the *skew* parameter which accounts for non-squared, skewed pixels. In practice, it is often zero, but in certain instances, it can take positive values.

In general, points in 3D space are expressed on a Euclidean frame in world coordinates, that's why we need to include rotation and translation to transform the point from the world to the camera coordinates. This can be expressed through a similarity transformation $\begin{pmatrix} R & -R\mathbf{C} \\ 0 & 1 \end{pmatrix}$ applied directly to the 3D point, where $\mathbf{C}$ represents the camera center in the world coordinate frame and R is a rotation matrix representing the rotation of the camera coordinate frame.

Finally, the *projection pipeline* reads as

$$\mathbf{x} = P\mathbf{X} = KR[I| - C]\mathbf{X} \tag{6}$$

where K is referred to as the intrinsic matrix, and $R[I| - C]$ is the extrinsic matrix.

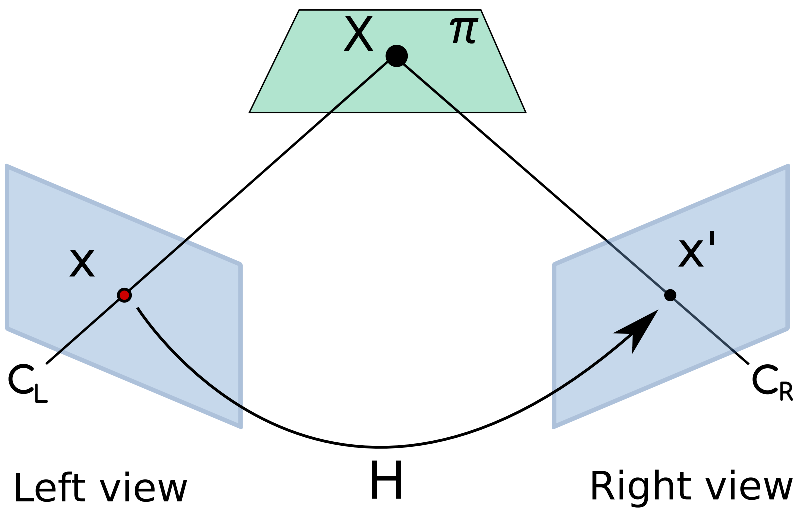


Figure 3: Homography mapping points on a plane from an image to another.

3.1.3 *Inverse Projection*

It is clear that the mapping between a 3D to a 2D point is essentially in one direction, since one dimension gets lost. Therefore, with a fixed camera, the projection of a 3D point is unique, while back-projecting a 2D point from the image correspond to an infinity of points lying on the ray passing by the point on the image and the camera center. Only with an additional knowledge of the depth $d(x, y)$ is possible to uniquely invert the projection process. The depth refers to the distance of the 3D point on this ray. More formally the ray can be expressed by connecting the camera center $C = -M^{-1}\mathbf{p}_4$ and the point at infinity $D = ((M^{-1}x)^T, 0)^T$ as follows:

$$\mathbf{d} = d\begin{pmatrix} M^{-1}\mathbf{x} \\ 0 \end{pmatrix} + \begin{pmatrix} M^{-1}\mathbf{p}_4 \\ 1 \end{pmatrix} = \begin{pmatrix} M^{-1}(d\mathbf{x} - \mathbf{p}_4) \\ 1 \end{pmatrix} \tag{7}$$

3.1.4 *Planes in Space and Homographies*

All image points of a plane are related to image points in a second view by mean of a planar homography. It is usual to say that the plane *induces* a unique homography between two views. The homography is a projective transformation that maps points from one image of the plane to another, see Figure 3. It has many practical applications such as *Image*

Rectification or *Panorama Stitching* registration, and we will see in Chapter 4 how it will be the core of our algorithm for Multi-View Stereo reconstruction.

A plane is defined as:

$$\pi^T \mathbf{X} = 0 \tag{8}$$

which express that point $\mathbf{X}$ is on plane π and where $\pi = (\mathbf{v}^T, 1)^T$.

Given the projection matrices for two views:

$$P = [I|\mathbf{0}] \qquad\qquad P' = [A\ \mathbf{a}] \tag{9}$$

then the Homography H induced by the plane is $\mathbf{x}' = H\mathbf{x}$ where:

$$H = A - \mathbf{a}\mathbf{v}^T. \tag{10}$$

Let's assume a stereo camera scenario where the first camera is aligned with the world origin:

$$P_E = K[I, 0] P'_E = K'[R|t]. \tag{11}$$

The plane has coordinate $\pi_E = (\mathbf{n}^T, d)^T$ such that points on the plane obey the relation $\mathbf{n}^T + d = 0$. The expression for the homography induced by the plane is the following. Using Equation (10) and $\mathbf{v} = \mathbf{n}/d$ the homography for the cameras $P = [I|0], P' = [R|t]$ is:

$$H = R - \mathbf{t}\mathbf{N}^T/d. \tag{12}$$

If we include also the intrinsic information on both cameras we obtain $P_E = K[I|0], P'_E = K'[R|t]$ and the final induced homography is:

$$H = K'(R - \mathbf{t}\mathbf{N}^T/d)K^{-1}. \tag{13}$$

What we obtained is a family of homography parametrized by $\mathbf{n}/d$. It is defined by the plane, and the internal and external parameters of the cameras.

3.2 IMAGE BASED 3D RECONSTRUCTION

3.2.1 *Two view geometry*

After introducing the camera model let's continue with the stereo case, where we analyze how a point in 3D is related on

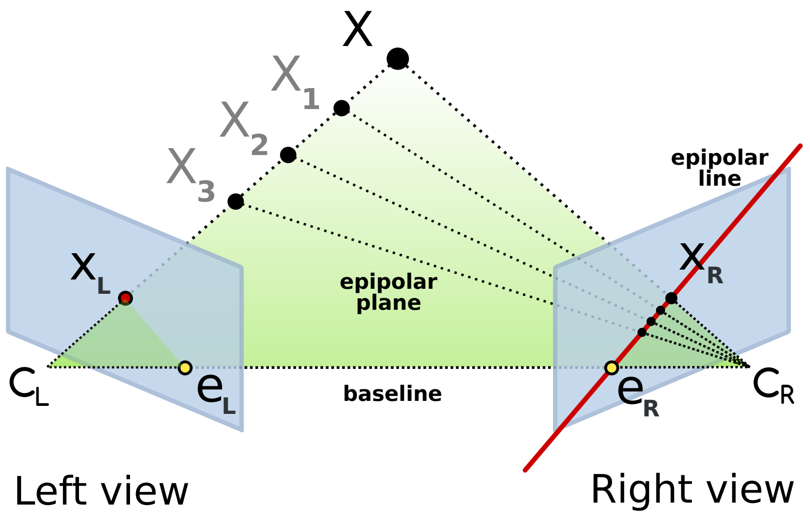


Figure 4: Epipolar geometry of a stereo image pair.

two views. This is expressed by the epipolar geometry which constraints how a point should be located when seen by two cameras.

Let's examine two cameras, the epipolar geometry describes the intrinsic projective geometry between two views.

Let's take into consideration a point $\mathbf{X}$ in space imaged in points $\mathbf{x_L}$ and $\mathbf{x_R}$ on the first and second camera respectively. The image points $\mathbf{x_L}$ and $\mathbf{x_R}$, the space point $\mathbf{X}$ and camera centers $\mathbf{C_L}$ and $\mathbf{C_R}$ lie on the same plane, see Fig. 4. The *epipolar plane* intersects both image planes in epipolar lines.

Given the point $\mathbf{x_l}$ on the first camera, its corresponding point must lie on the epipolar line of the right camera. Additionally, assuming both cameras are looking at the scene, point $\mathbf{x}$ must be in front of the cameras, therefore the point is further constrained by the epipole $\mathbf{e_R}$, the intersection point between the line connecting the two cameras, the *baseline* and the epipolar line. With those constraints, when finding the correspondence from the left to the right image, the search is limited to a 1D space. Indeed, the point in 3D can be parametrized w.r.t. the distance μ over the ray connecting $\mathbf{C_L}$ and $\mathbf{X}$:

$$\mathbf{X}(\mu) = \begin{pmatrix} M^{-1}(u\mathbf{X} - \mathbf{p}_4 \\ 1 \end{pmatrix} \tag{14}$$

Even if the literature on Multi-View Geometry (already when the number of images is more than 2 we are talking about multi-view) is vast, in the following we will not tackle directly the analytic relation of an object as seen from multiple cameras (the trifocal tensor for three views and the quadrifocal tensor for four). It will be enough to consider the 2-views epipolar geometry and extend it to a multi-camera setting.

3.3 CONVOLUTIONAL NEURAL NETWORKS

In this section, we will review the basic concepts important to understand *Convolutional Neural Networks* (CNN).

First, we will start with a basic description of a generic neural network, we will continue then with Convolutional Neural Networks. There is still a clear unbalance between the obvious practical benefits of deep neural networks and their theoretical understanding, it is nevertheless possible to point at reasons for its recent success in computer vision, by reasoning on the way convolution works on its data.

We will continue listing additional layers and their meaning for CNN network. Finally, we will sketch the optimization approach used to optimize the network parameters.

As we restrict to the important presentation to understand the following chapters, we refer to the book of Goodfellow *et al*. [46] for a more detailed overview of deep learning, from which this part is inspired.

3.3.1 *Deep Feedforward Neural Networks*

The goal of a *Deep feedforward neural networks*, also known as *feedforward neural network* or *Multi-Layer Perceptrons* (MLP), is to approximate an unknown function $\mathbf{f}^*$. Feedforward neural networks learn the parameter θ for the mapping $\mathbf{y} = f(\mathbf{x}, \theta)$ from the input $\mathbf{x}$ to the output $\mathbf{y}$ which best approximates the original function $\mathbf{f}^*$. In case $\mathbf{y}$ is defined on a discrete domain we are talking of classification, otherwise, for $\mathbf{y}$ defined on a continuous domain, the network can perform regression.

They are known as *feedforward* because data flows from the input to the output inside the direct acyclic graph (DAG) and there is no feedback connection where the outputs of the model are fed back to itself, see Fig. 5. The type of network which deals with feedback connection is known as *Recurrent Neural Network* (RNN), and it is often used in case the input data is a sequence dependent on its past, like text or videos but it will not be considered here.

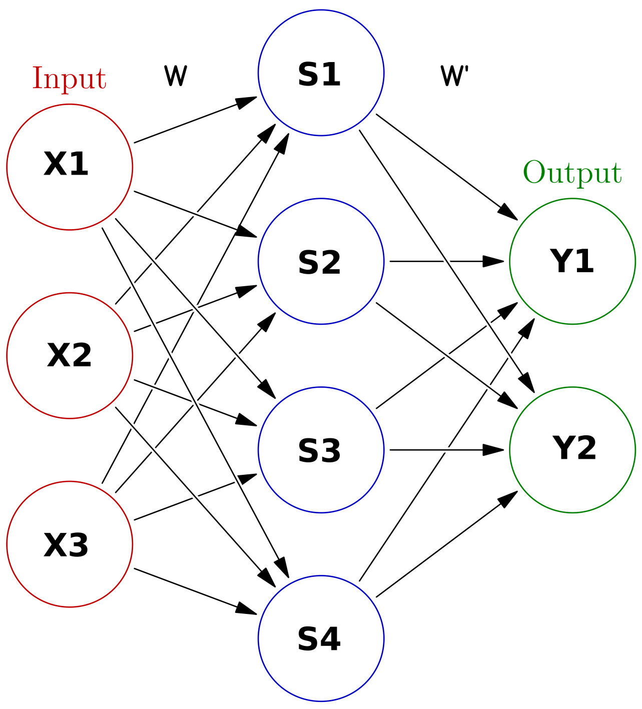


Figure 5: Graphical depiction of 2-layer neural network composed of one hidden layer of 4 units and one output layer with 2 units and three inputs. Adapted from Wikipedia.

It is known to be a network because it is composed of a combination of different functions. The overall network is then described by a direct acyclic graph which describes how the functions are composed together. For example, $f(\mathbf{x})$ might be the result of the composition of multiple functions f^1, f^2, f^3:

$$f(\mathbf{x}) = f^3(f^2(f^1(\mathbf{x}))) \tag{15}$$

This chain is a simple network where f^1 is the first *layer* f^2 the second, and f^3 the last, or *output layer*. It is a simple sequential structure that has been used by many successful neural networks. Currently, the trend is to experiment with a more elaborate architecture of the network. The amount of layers, functions in our example, determines the *depth* of the network, the more the layers, the deeper the network, that's why we are talking about **deep learning**.

NETWORK TRAINING During *training* we would like to obtain parameters θ which best approximates f^*. This is done thanks to a set of noisy observations of $f^*(x)$, the *training data* evaluated at different points. So for each input x , we have an output y. The other intermediate layers, often called *hidden layers*, are not explicitly fixed by the training data. It is the **learning algorithm** which must learn their behavior employing a training algorithm, often simply an adaptation of the popular gradient descent.

BIOLOGICAL INSPIRATION Neural network it is the most successful computing system biologically inspired. Indeed, it is composed of building blocks, or *layers*, which loosely emulate a simplified version of how the human brain is known to work, hence the name *neural*. Nevertheless, the development and research of neural network are mainly driven by advances in machine learning and engineering disciplines rather than discoveries in neuroscience.

3.3.2 *Convolutional Neural Networks*

The most successful type of network is the *Convolutional Neural Network* (CNN). In this type of network at least one layer is composed by convolutions instead of matrix multiplications.

Almost 30 years ago Lecun *et al*. [86] was the pioneer with the very first CNN architecture, refined until 1998 [87] to obtain Lenet-5, see Fig. 6. They demonstrated that CNN was able

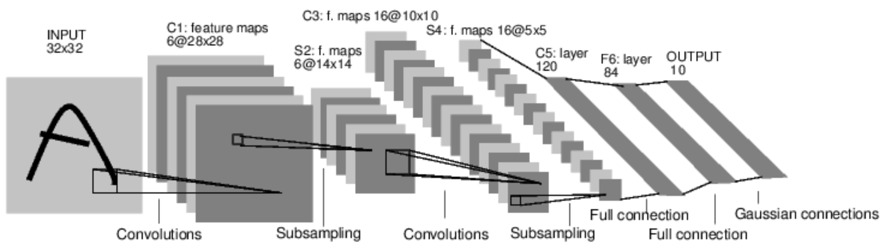


Figure 6: Original Lenet-5 network, one of the first CNN architecture with a successful application of handwritten zip code recognition, original picture from Lecun et al.[87]

to recognize successfully handwritten zip codes. Even if this application pales when compared to what CNN is capable of doing nowadays, it was nevertheless an astonishing result at the time of its publication. And more important, it became evident CNNs were able to obtain competitive results on real-world applications and were not limited to application on synthetic data inside research.

Nevertheless, it was only after 20 years, in 2010 [19] and the advent of parallel computing on video cards which allowed researchers to effectively start to explore and make use of the power of CNN. But what is considered to be the groundbreaking result, that exposed it to researchers was the work of Krizhevsky *et al.* [82]. The network architecture they proposed, named Alexnet, from the first name of the author, was composed of many more layers and won by a large margin the stiff Imagenet competition [28] for classification. From that moment on until today, computer vision community and machine learning started to explore and use CNN for different applications, often overtaking classic techniques that did not make use of a neural network. Even now what many research works try to achieve is the answer to the following question: *Can CNN beat state of-the-art even on this specific application, and which is the best architecture for our purpose*?

The answer is almost always yes: CNN proves to be tremendously successful in the field of natural language processing, image and video recognition, image segmentation, and computer vision in general. Unfortunately in 3D reconstruction CNNs struggle to obtain competing results.

Even if there are no strong theoretical results which explains why it is that successful, we can nevertheless identify three main ideas of CNN which empower neural networks:

1. sparse interactions,

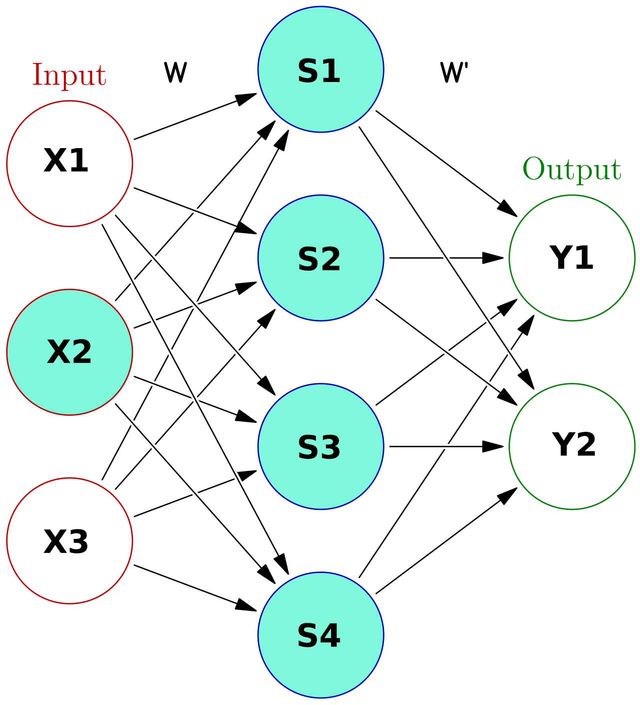
2. parameter sharing,

3. equivariant representation.

SPARSE INTERACTIONS Standard neural networks connect each layer by means of a matrix containing the parameters describing the interaction between each input unit and output unit. Every input is connected to each output, see Figure 5. Convolutional Neural Networks instead implements a sparse interaction between input and output units, due to the small support of the kernel weights see Figure 7. Indeed, usually the convolutional kernel has size between 1x1 and rarely bigger than 9x9 when compared to typical image dimension ranging from 32x32 to 512x512. The result is effectively a reduction in the number of parameters to learn for in the neural network which in turns reduce the training time and improve the statistical efficiency of the network.

PARAMETER SHARING Another unique feature of CNN is the ability to share parameters when computing the output. This is due to the fact that the same kernel is used to compute output feature by shifting it over the input, or image. In other words the kernel weights are not dependent on every location but get shared across all the image. The result is a dramatic reduction in parameter dimension as only one kernel matrix is needed for all the image as opposed to one kernel matrix for every pixel, see Figure 8. With input images of size up to 512x512 it amounts in a drastic reduction of memory requirement.

EQUIVARIANT REPRESENTATION Parameter sharing on convolution leads to another specific property: *equivariance to translation*. In functions with this type of property if we translate the input, the output changes in the same way. More formally for a function $y = f(x)$, if we shift the input by t also the output is changed in the same way: $y - t = f(x - t)$. If we think again that the same weights get applied over all the image, it becomes natural this kind of property to hold. For example, it might be useful to have a layer which detect edges on the image irrespective of where they appear. On some other cases the position when a specific feature appears of the image might be important for the final problem we are trying to solve. For example let's assume we would like to process faces in images that are already cropped to be centered on an



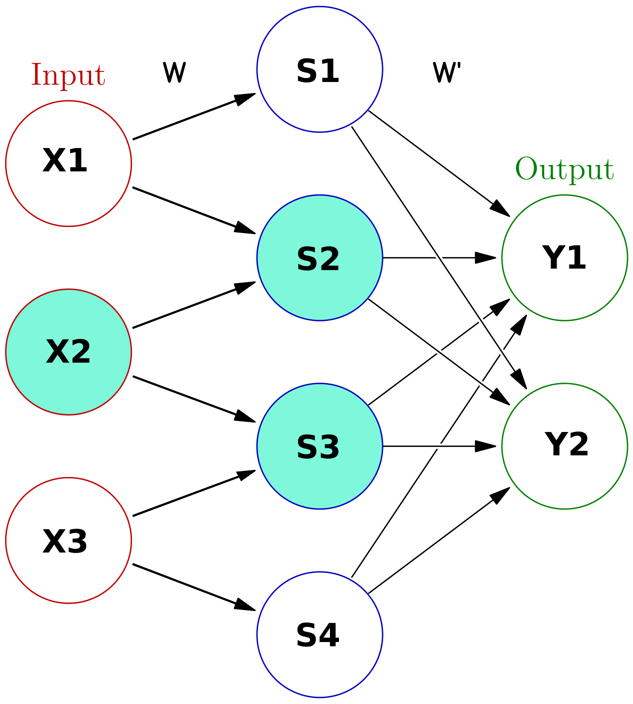
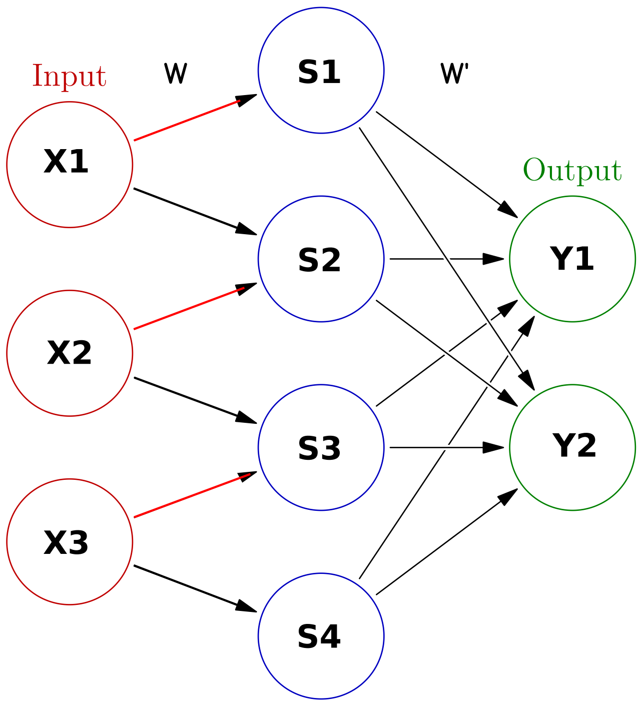


Figure 7: **Sparse connectivity** viewed from the input unit x2. **Top**: when s is formed by a matrix multiplication connectivity is dense and x2 is connected to all unit s. **Bottom**: In case s is formed by convolution with kernel width 2, connectivity is sparse and x2 gets connected only to s2 and s3. Adapted from [46]

.



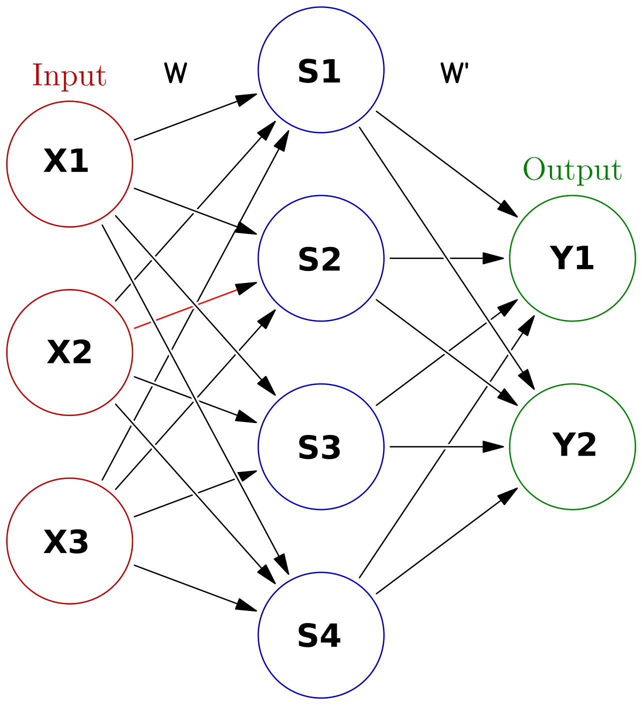


Figure 8: **Parameter sharing**: in red arrow is indicated the connection using a particular parameter in two different models. x1. **Top**: The red line is indicating the left weight of a 2-element kernel in a convolutional layer. Ad you can see the same parameter is used also on input units x1 and x2. **Bottom**: The red line indicated a specific weight of a weight matrix in a fully connected model. Since there is no sharing, the parameter is used only to connect x2 with s2. Adapted from [46]

individual face, in this case we would like to extract different features at different part of the face. A part of the network for the eyes, a part for the eyebrows and so on.

3.3.2.1 *Common Layers*

Even if neural networks have demonstrated to be able to accomplish very complex tasks, *e.g.* image caption generation [117] or text generation [47], the layer they are build with are all very simple. This is a very surprising and interesting properties of neural networks, simply by changing the order and parameters of the same building blocks we are able to obtain competing results on many applications. This makes the design of neural network very elegant, considering how basic the building blocks are. Moreover, thanks to the help of common freely available framework to create neural network, it allows almost anyone to create from scratch new networks and to prototype very quickly.

3.3.2.2 *Rectified Linear Unit*

In case we would only make use of matrix multiplications and convolutions in our network, all linear operators, we would only able to express a linear relation between the input and the output. What proved to be really powerful in neural network is the inclusion of a so-called *non-linearity* in the *activation function*, a function which breaks the linear relation between input and output and allows approximating more complex functions. Non-linearities in general proved to perform well in practice, but they also allowed to state the most popular theorem for neural networks.

The **Universal approximation theorem** proposed by Hornkit *et al.* and Cybenko *et al.* [26, 61, 62] states that a feedforward network with a linear output layer and at least one hidden layer with a finite number of neurons and any *squashing* function (an activation function under mild assumptions such as the sigmoid activation function) can approximate continuous functions on compact subsets of $\mathbf{R}^n$. In other words it states that simple neural network can represent a wide variety of functions, unfortunately it does not give any bound on the algorithmic learnability of the network parameters. While at first this result might look to have a great practical value, it says that a simple network can approximate almost any function but it could take long time to train that network, so there might not be any practical resources to train that simple network for the desired output.

While at the beginning it looked like it was the specific choice of the activation function giving the representational power to the network, in a later work Hornik showed [61] that in reality it is the feedforward architecture that give neural network the potential of being universal approximators.

A sigmoid activation function was used in the theorem because of multiple nice properties. First of all it is bounded, differentiable everywhere and with non-negative derivative at each point. In modern networks it is not used anymore because it suffers from the problem of vanishing gradients due to the small value of the derivative for low values of the input. The most popular activation function without big drawback is the *Rectified Linear Unit* (RELU). Its main benefits are to be bounded, differentiable everywhere except zero and without the vanishing gradient problem, the derivative being either 0 or 1. It is a simple operation which clips to zero any negative value of the input:

$$\mathrm{RELU}(x) = max\{0, x\}. \tag{16}$$

With the addition of this simple nonlinear operation we are able to increase the expressiveness of our network. RELU is the most common one, but many other nonlinear operations have been introduced and explored inside neural networks: i.e. $\tanh$, Leaky RELU [90], ELU [21] and SELU [78].

There is still no consensus on which one performs better, only empirical observation of different results for each type of network architecture. While some have been proved to perform better on certain kind of application and dataset usually the difference is minimal and will not affect the overall result of the network.

3.3.2.3 *Pooling*

A standard convolutional layer is composed by a convolution followed by a non-linear function such as RELU and then a pooling operator. *Pooling* is an operation which substitute the output of a net with a summary statistic of nearby output. The most common one is the *max-pooling* operation which replace the output within a rectangular window with the maximum value contained in it. Alternatively, other pooling operations have been proposed: averaging of the neighborhood as used in

Chapter 5, L_2 norm of the window or weighted average based on the distance from the central pixel.

INVARIANCE TO TRANSLATION This type of operation helps to make a representation invariant w.r.t. small variation of the input. Even if the input is translated by a small amount, within the pooled rectangle, the pooled outputs do not change. *Invariance to translation* might be a desirable characteristic depending on whether it is important for our application to detect which feature is present rather than where it is. For example when performing classification of a dog we are not interested in the position of the dog rather in understanding that there is a nose. On the other hand in case we would like to detect the pixel accurate position of the nose of the dog the pooling operation might destroy our information.

SIZE REDUCTION It is important to apply pooling to reduce the size of our network. *E.g.* if we are going to apply a max-pooling operator with a window and stride 4 we reduce the input size by a factor of $4^2 = 16$. This results in faster training time, smaller memory usage and improves the computational efficiency.

MULTI-SCALE PROPERTY Furthermore, pooling compensates for the limited receptive field of the convolution by downsampling the image. Each time pooling is applied, we practically downscale the image. In this way, even with small kernel windows, convolution inside CNNs is able to extract features from the image even if they would not be visible from the receptive field, see Figure 9. Consecutively chaining convolution and pooling effectively creates a sort of multi-scale network able to *see* the whole image, it creates what has always been used in computer vision, a *scale-space*.

UN-POOLING If instead of pooling we apply its inverse we are able to up-sample the input dimension. This allows us to create typical encoder-decoder architectures with an output dimension of the same order of magnitude of the input but retaining the multi-scale properties mentioned before.

See for example two different architectures, Fig. 11 and Fig. 12 where pooling plays an essential role and enable the convolutions, even with kernel size 3x3, to capture information from all the image.

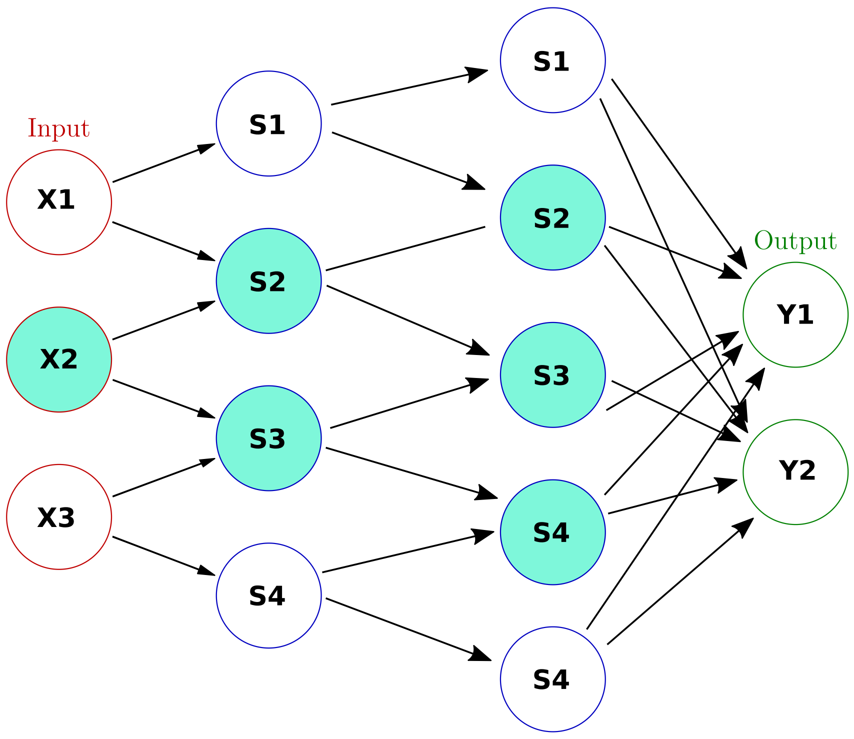


Figure 9: In this network we show the receptive field of from the input unit x2. Even if the direct connectivity of the convolution is sparse, the implicit connectivity inside the network gets later the shallow layer. When combined with pooling, unit in the deeper layer can be indirectly connected to all or most of the input units. Adapted from [46]

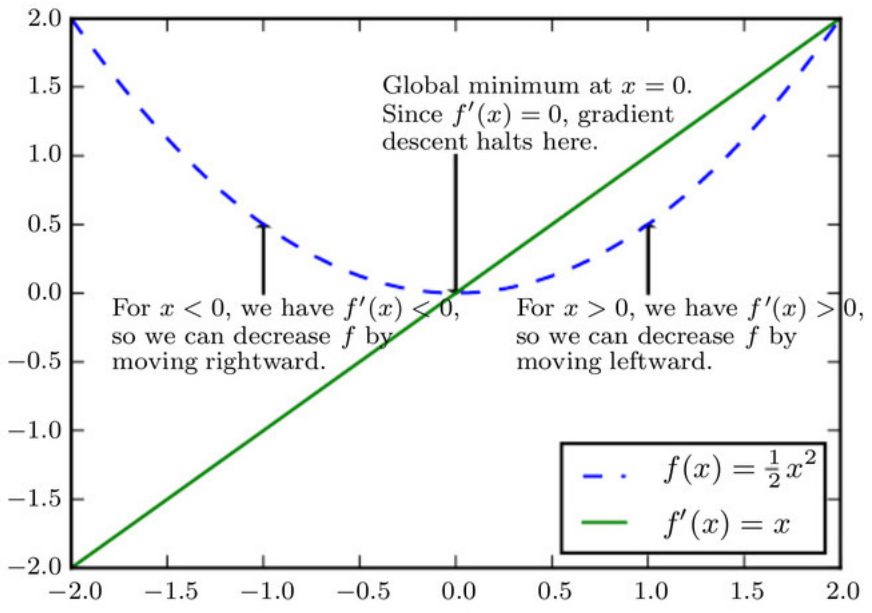


Figure 10: Gradient descent algorithm uses the derivative of a function to move the solution downhill and closer to a minimum. Source: Wikimedia.

3.3.3 *Gradient Descent*

As mentioned in Section 3.3.1 optimization is an important component of Deep Learning. Optimization refers to the task of minimizing (or maximizing) a function $f(\theta)$ by altering the value of θ. Gradient descent [16] is a technique to minimize functions by making use of the derivative of the function w.r.t. the current solution to obtain an improved solution. The algorithm start with an initial guess, an *initialization*, and proceeds iteratively in steps by updating the solution following the function downhill until a minimum is reached:

$$\theta_0 = \theta_{init} \qquad \theta_{n+1} = x - \lambda \nabla(f(\theta_n)) \tag{17}$$

where ∇ is the gradient operator and λ a parameter which tune the length of the step at each iteration, see Figure 10. For certain assumptions of the function f (for example f convex), and particular choices of λ convergence to a minimum is guaranteed.

STOCHASTIC GRADIENT DESCENT All of deep learning is powered by one popular algorithm: **stochastic gradient descent**. It shares the name with gradient descent but it includes important modification to make it working with the large training dataset typical used to obtain a good generalization.

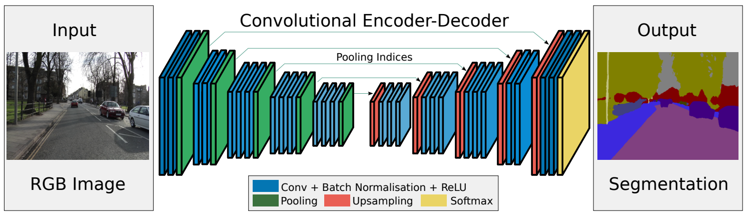


Figure 11: Architecture of Segnet by Badrinarayanan et al. [4] a network which does pixel-wise classification of images. Note the multi-scale structure of the network by means of pooling.

The insight of stochastic gradient descent is that the gradient is an expectation. Instead of using all the training samples to compute the gradient, the main idea consists of sampling uniformly the training set to build a **minibatch** of examples. Each minibatch size is typically relative small w.r.t. to the number of samples. A minibatch size is typically from 1 to few hundred. Nevertheless, each minibatch updates the current solution even if the training set might contain billions of examples. In other words the true gradient computed for the parameters θ on the training set $\nabla f(\theta, x)$ is approximated by a gradient over the minibatch set $B = \{x^1, ..., x^m\}$:

$$\theta_{n+1} = \theta_n - \lambda \nabla_{B_i} L_{B_i}(\theta, x, y) \tag{18}$$

where $L_{B_i}(\theta, x, y)$ is the loss function of the network with training pair x, y in B_i computed with parameters θ and step size λ, also known inside deep learning as **learning rate**.

Even if the shape of the function that is usually going to be optimized in deep learning is unknown, today we know that deep learning networks works very well when trained with stochastic gradient descent. The optimization algorithm does not have any guarantee to arrive at a local minimum in a few iterations, and not even that the local minimum is near the global minimum, but it often finds a low value of the cost function fast enough to be useful in practice.

3.4 MULTI-VIEW STEREO BENCHMARKS

In this section we review the most popular Multi-View Stereo datasets and benchmark used to compare different algorithm.

Science is based on progressive advancement of new works. Even though each publication should contain enough information to compare it to other existing methods, often the param-

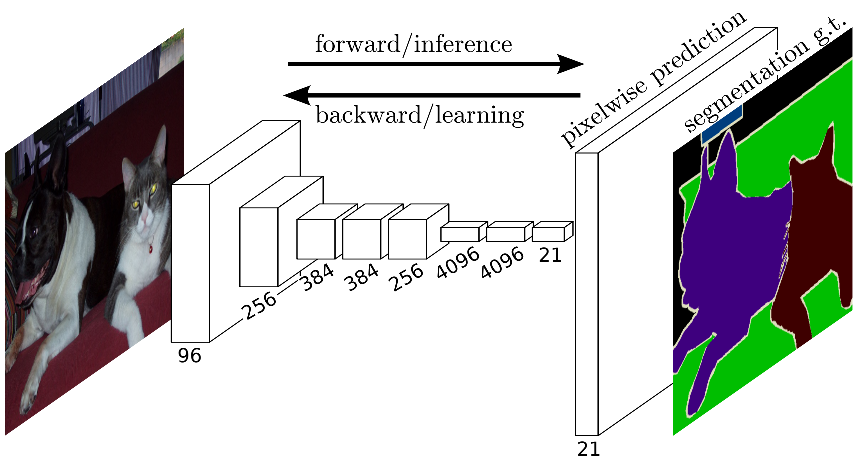


Figure 12: Architecture of a pixel-wise classification of images by Long et al. [88]. As in Fig. 11 the architecture makes use of pooling to downsize the image to capture interaction from all the image.

eters and data are so different that a fair comparison across different methods becomes very difficult. What makes it easier is the use of a single test-bed. It allows a fair comparison and speed up the job of researchers, free from the burden to re-implement other publications.

Being able to directly rely on ready-made datasets speeds up the experimental evaluation of algorithms and saves us from the error prone process of data creation. Other than enabling a more direct quantitative evaluation, a common dataset allows a community of researcher to understand in deep specific fallacies of difficulties. Therefore, researchers understand more easily the specific features or failure cases of a method.

Even if a common dataset contains per se all the ingredients to foster a fair comparison, it is the only with an impartial benchmark that we are able to truly compare and understand differences across methods. A shared benchmark provides a direct and fair error measure guaranteed by impartial "judges" which maintain an online website with a ranking of different approaches. The authors are therefore not only free from the creation of new dataset but also from the choice of an evaluation protocol, which could be otherwise crafted to favor a specific feature of the method.

We will start first with the popular *multiview Middlebury benchmark* [107], we will continue with the discontinued *Strecha benchmark* [114] and the *DTU dataset* [66]. At the end I'll de-

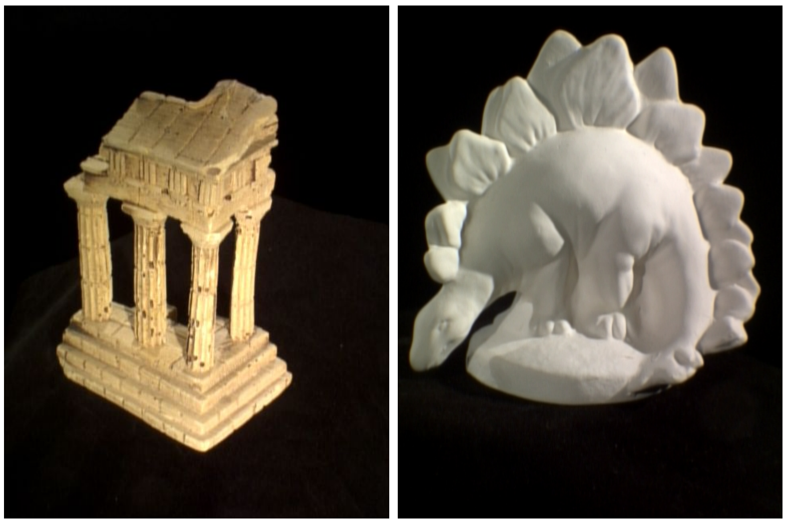


Figure 13: The two objects used to compare methods in the Middlebury Multi-View Stereo benchmark [107]. The two objects were selected to include different characteristics: sharp and smooth features.

scribe future dataset and comment the difference and feature of each one of those.

3.4.1 *Middlebury Multi-view dataset*

Seitz et al. [107] with *Middlebury Multiview stereo benchmark* are the first to propose and online benchmark for the evaluation of multi-view stereo algorithms in 2003. Even if multiple objects were proposed in their original paper, the data is composed of only two plaster reproduction of two objects: a temple and a dinosaur model, see Fig. 13, of size $10\text{cm} \times 16\text{cm} \times 8\text{cm}$ and $7\text{cm} \times 9\text{cm} \times 7\text{cm}$ respectively.

IMAGE CAPTURE Images are captured with a robotic arm which densely sample an half-sphere of one meters. Roughly 300 images at resolution 640×480 are captured per object, but images are distributed in 3 sets: *full* contains all the images, *ring* only a selection of 47 objects and *sparse ring* samples only 16 objects, see Fig. 14. This split was probably done for two recons: First of all to encourage researchers to show their results on different level of redundancy in the images. It was also probably a requirement at the time to be able to work on a reduced set of images due to the limited computing capabilities.

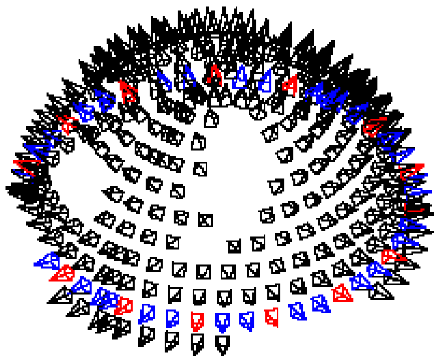


Figure 14: Position and orientation for the temple dataset. The dome is not fully covered due to shadows. The blue and red cameras indicate the set used for the *ring* dataset, while the 16 red cameras show the position for the *sparse ring* dataset

.

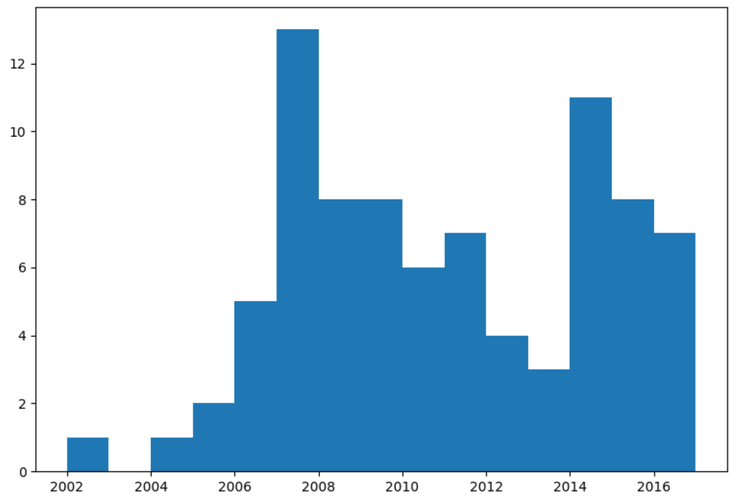


Figure 15: Frequency of submission on the Middlebury Multi-View benchmark. Even if proposed in 2006 it is still actively used.

GROUND TRUTH RECORDING Ground truth has been reconstructed by mean of Cyberware Model 15 laser stripe scanner. Per each object 200 individual scans were captured and merged together to reduce occlusions and improve the overall accuracy of the scan.

The reference model were aligned to the images in an iterative optimization approach minimizing photo-consistency between images and the reference mesh. The output of the alignment process was translation, rotation and scale, introduced to compensate for differences in calibration between images and laser scan.

COMMENTS AND SUCCESS The benchmark has been used for more than 10 years and 84 different methods have been compared and measured one against each other, see Figure 15. These numbers alone highlight the success of this benchmark, despite the limited number of objects proposed. Nevertheless, for a long period it has been the only way to directly evaluate the performance of a MVS method w.r.t. to all past methods.

3.4.2 *Strecha dataset*

Strecha *et al.* [114] proposed in 2008 a new online evaluation benchmark for MVS but specifically tailored to outdoor scenes and high resolution imagery. They proposed an evaluation of 6 different outdoor scenes: building, facades and a fountain. Each object is captured with 10 images with a resolution of 3072×2048 (roughly 6 Megapixels). The increase in dimension was a significant step forward w.r.t. Middlebury Benchmark where the resolution of images was 640×480. For this reason this dataset fostered new type of algorithms focus on speed and being able to handle large amount of data.

Nevertheless, the type of objects was too homogeneous and, despite not being captured in controlled environment, it did not provide any difficult challenge such as specularity, complex object or surface reflectance function. Images are captured manually but roughly trying to cover a circular trajectory from around the object in order limit the number of occlusions.

Unfortunately the online benchmark has been discontinued, that's why it cannot be used a reference for current evaluation of new algorithms.

Benchmark	Setting	Resolution	Online Eval.	6DoF Motion	MVS	Stereo	Video	Varying FOV
Middlebury MVS [107]	Laboratory	0.3 Mpx	✓		✓			
DTU [66]	Laboratory	2 Mpx			✓			
Strecha [114]	Buildings	6 Mpx		✓	✓			
ETH3D [105]	Varied	0.4 / 24 Mpx	✓	✓	✓	✓	✓	✓
Tanks [79]	Varied	2 / 8 Mpx	✓	✓		✓	✓	

Table 1: Comparison of existing multi-view-stereo datasets. We differentiate between different scene types (*e.g.*, staged scenes captured in a laboratory vs. synthetic scenes), whether the camera undergoes a restricted or a full 6 degrees-of-freedom (DoF) motion, whether cameras with different fields-of-view (FOV) are used, if video is used or dataset includes picture from multiple point of views(MVS).

3.4.3 *DTU dataset*

To compensate for the small image size and limited scene variability for the Middlebury Benchmark, Jensen *et al.* [67] proposed a new DTU dataset, but did not include an online evaluation, instead they release both images and ground truth data to the public. To be able to quickly capture multiple objects they made use of a programmable 6-axis robot arm. The robot provided precise camera position due to high repeatability of the robot arm. In other words they were able to program predefined positions to be used for capturing data which can be reproduced with high precision for each object. The robot was moved on concentric spheres around the object to reconstruct 80 different objects: 59 objects contained 49 images captured in a sphere with radius 50cm, while the others objects were captured with 64 images captured with an additional sphere of radius 65cm. Additionally, each image was capture with 7 different illumination, created by varying the illumination patter of multiple led on a plane. Ground truth was obtained by merging multiple scan from an accurate structured light scanner.

3.4.4 *Shortcoming and upcoming datasets*

Most of the shortcomings of the Middlebury Benchmark have been addressed by the subsequent datasets, but none of them was able to provide a decent substitute for the vision community which could be a reference for the development and measurement of advancements in MVS methods. That's why recently at the same time Schöps *et al.* [105] and Knapitsch *et al.* [79] proposed a new MVS stereo benchmark for real world scenes and high resolution imagery and video, respectively named ETH3D and Tanks. They both included stereo scenes, and video captured with a camera with 6 degree of freedom. Tanks provide only images from video without camera calibration but ETH3D include also picture from multiple point of view, in the same way as Strecha dataset, see Table 1. Furthermore, ETH3D include a stereo ring with lenses with variable field of view, to potentially emulate mobile or robotic application.

It is difficult to predict in advance whether both benchmark will become popular, nevertheless it is true that any new deep learning application is eager to make use of ready-made 3D training data. That's why both ETH3D and Tanks will provide a valuable contribution to the computer vision community.

4 MASSIVELY PARALLEL MULTIVIEW STEREOPSIS

This chapter is based on Galliani et al. *[41].*

Reconstructing dense 3D shape from multiple images has been a topic of interest in computer vision for many years. Since camera pose estimation and multiview triangulation can be considered solved (at least for images that are suitable for subsequent dense reconstruction), the problem boils down to the fundamental task of image matching, *i.e.* establishing dense correspondence between images. The majority of the literature deals with the basic stereo setup with two images, *e.g.* [64, 71, 97, 99, 102]. It is evident that using more than two viewpoints will improve both the accuracy of the reconstructed 3D points (by triangulating from more rays) and the robustness against grossly wrong matches (by checking the coherence of redundant observations). Moreover, using more than two viewpoints alleviates the occlusion problem, and can reconstruct objects more completely, *e.g.* [15, 37]. On the other hand, the multiview setup exacerbates the problem that already many successful stereo methods do not scale up to realistic image sizes of several million pixels. Nevertheless, guided by the quality metrics used in standard benchmarks such as KITTI and Middlebury, most authors concentrate on accuracy and pay limited attention to scalability and runtime performance. Many existing algorithms become impractical when moving to larger sets of high-resolution images.

In this work we present a multiview matching method that delivers dense, accurate 3D point clouds while at the same time being efficient enough to handle large images. Our goal is a fast matcher which is nevertheless very accurate. On the recent DTU benchmark, our method reaches the best compromise between accuracy and completeness (the best accuracy with 2$^{\text{nd}}$-best completeness, or the best completeness with 2$^{\text{nd}}$-best accuracy; see example in Fig. 16) still it can match ten 2-Megapixel images in less than 2 seconds on a standard desktop PC.

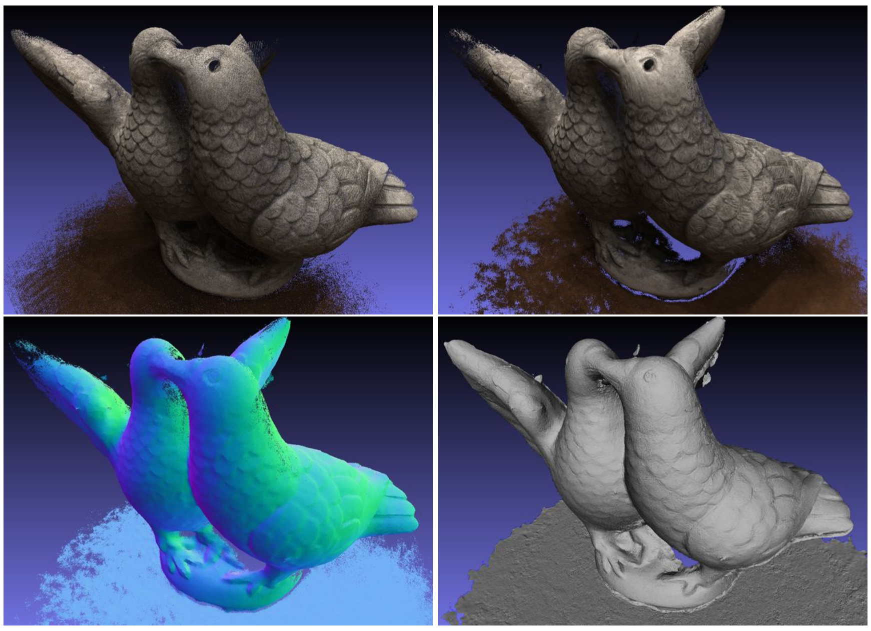


Figure 16: Results on one of the 80 evaluated objects on the DTU benchmark [66]. *Top left*: Ground truth point cloud; *top right*: reconstructed point cloud with texture; *bottom left*: color-coded surface normals; *bottom right*: reconstructed surface.

CONTRIBUTION. We present ***Gipuma***, a simple, yet powerful multiview variant of Patchmatch Stereo with a new, highly parallel propagation scheme.

Our first contribution addresses computational efficiency: standard Patchmatch is sequential in nature, since it propagates information diagonally across the image pixel-by-pixel. A little parallelisation can be achieved by procedures such as aligning the propagation direction with the image axes and running rows/columns in parallel [5, 6, 55, 134], but these still do not fully harness the capabilities of current hardware. Instead, we propose a new diffusion-like scheme that operates on half of all pixels in an image in parallel with a red-black (checkerboard) scheme. It turns out that this arguably more local propagation, which is particularly suitable for modern many-core GPUs, works as well as the standard Patchmatch procedure, while being a lot faster.

The second contribution aims for accuracy and robustness: we extend PatchMatch Stereo from a two-view to a multiview matcher to better exploit the redundancy in multiview datasets. The Patchmatch Stereo method by construction recovers also a normal in disparity space at every pixel. The starting point for our extension is the observation that one can just as well define the normals in Euclidean 3D scene space. In that case they immediately define a local tangent plane at every surface point, and thus an associated homography (respectively, a pair of slanted support windows) between any two images viewing the surface. The explicit estimation of the surface normal makes it possible to utilize plane-induced homographies when checking photo-consistency between different views. It avoids epipolar rectification and allows one to aggregate evidence over multiple images in generic configuration.

The described multiview setup still needs a reference image to fix the parametrization of the surface. Hence, we first compute depth using every image in turn as reference, and then fuse the results into one consistent 3D reconstruction. However, we prefer to carefully exploit the multiview information at the level of photo-consistency, and then use a rather basic fusion scheme to merge them into a consistent 3D point cloud. This is in contrast to some other methods that start from efficiently computable, but noisy depth maps and merge them with sophisticated fusion algorithms, which (at least implicitly) have to solve the additional problem of surface fitting [65, 83, 128].

We will show in our experiments that our implementation yields state-of-the-art multiview reconstruction on a variety of datasets.

4.1 PATCHMATCH STEREO

We start by briefly reviewing the original *Patchmatch Stereo* method [11], to set the scene for our extensions.

4.1.1 *Patchmatch for rectified stereo images.*

The core of Patchmatch stereo is an iterative, randomized algorithm to find, for every pixel $\mathbf{p}$, a plane $\pi_\mathbf{p}$ in disparity space such that the matching cost m in its local neighborhood is minimized. The cost at pixel $\mathbf{p}$ is given by a dissimilarity measure ρ, accumulated over an adaptive weight window $W_\mathbf{p}$ around the pixel. Let $\mathbf{q}$ denote the pixels in the reference image that fall within the window, and let $\pi_\mathbf{p}$ be a plane that brings each pixel $\mathbf{q}$ in correspondence with a pixel location $\mathbf{q}'_{\pi_\mathbf{p}}$ in the other image. Then the matching cost is

$$m(\mathbf{p}, \pi_\mathbf{p}) = \sum_{\mathbf{q} \in W_\mathbf{p}} w(\mathbf{p}, \mathbf{q})\, \rho(\mathbf{q}, \mathbf{q}'_{\pi_\mathbf{p}}). \qquad (19)$$

The weight function $w(\mathbf{p},\mathbf{q}) = e^{-\frac{\|I_\mathbf{p}-I_\mathbf{q}\|}{\gamma}}$ can be seen as a soft segmentation, which decreases the influence of pixels that differ a lot from the central one. We use a fixed setting $\gamma = 10$ in all experiments.

The cost function ρ consists of a weighted combination of absolute color differences and differences in gradient magnitude. More formally, for pixels $\mathbf{q}$ and $\mathbf{q}'_{\pi_\mathbf{p}}$ with colors $I_\mathbf{q}$ and $I_{\mathbf{q}'_{\pi_\mathbf{p}}}$

$$\begin{aligned} \rho(\mathbf{q},\mathbf{q}'_{\pi_\mathbf{p}}) = (1-\alpha)\cdot \min(\|I_\mathbf{q} - I_{\mathbf{q}'_{\pi_\mathbf{p}}}\|, \tau_{col}) \\ + \alpha \cdot \min(\|\nabla I_\mathbf{q} - \nabla I_{\mathbf{q}'_{\pi_\mathbf{p}}}\|, \tau_{grad}) \ , \end{aligned} \tag{20}$$

where α balances the contribution of the two terms and τ_{col} and τ_{grad} are truncation thresholds to robustify the cost against outliers. In all our experiments we set $\alpha = 0.9$, $\tau_{col} = 10$ and $\tau_{grad} = 2$.

4.1.2 *Sequential propagation.*

The Patchmatch solver initializes the plane parameters (disparity and normal) with random values. It then sequentially loops through all pixels of the image, starting at the top left corner. Good planes are propagated to the lower and right neighbors, replacing the previous values if they reduce the cost over the slanted support window. Additionally, it is proposed to also propagate planes between the two views. The propagation is interleaved with a refinement of the plane parameters (using bisection). After finishing a pass through all pixels of the image, the entire process is iterated with reversed propagation direction. Empirically, 2-3 iterations are sufficient. For optimal results the disparity image is cleaned up by *(i)* removing pixels whose disparity values are inconsistent between the two views; *(ii)* filling holes by extending nearby planes; and *(iii)* weighted median filtering.

4.1.3 *Plane parameterization.*

In Patchmatch stereo, the $\pi_\mathbf{p}$ are planes in disparity space, *i.e.* 3D points $\mathbf{P} = [x, y, disp]^\top$ must fulfill the plane equation

$$\tilde{\mathbf{n}}^\top \mathbf{P} = -\tilde{d} \quad , \quad disp = -\frac{1}{\tilde{n}_z}(\tilde{d} + \tilde{n}_x x + \tilde{n}_y y) \quad , \tag{21}$$

with normal vector $\tilde{\mathbf{n}}$ and distance $\tilde{d}$ to the origin. This definition yields an affine distortion of the support windows in the rectified setup [55].

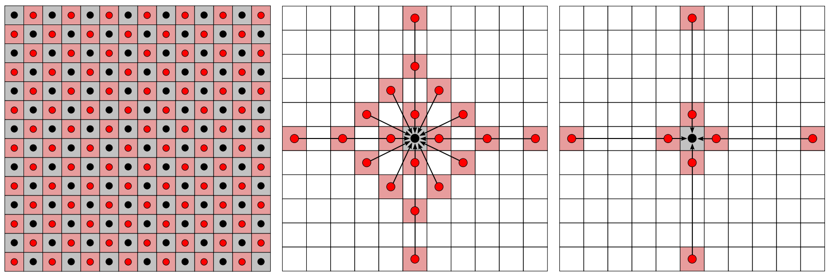


Figure 17: The propagation scheme. **Top**: Depth and normal are updated in parallel for all red pixels, using black pixels as candidates, and vice versa. **Middle**: Planes from a local neighborhood (red points) serve as candidates to update a given pixel (black). **Bottom**: Modified scheme for speed setting, using only inner and outermost pixels of the pattern.

4.2 RED-BLACK PATCHMATCH

4.2.1 *Surface normal diffusion*

The standard Patchmatch procedure is to propagate information diagonally across the image, alternating between a pass from top left to bottom right and a pass in the opposite direction. The algorithm is sequential in nature, because every point is dependent on the previous one. Although several authors have proposed a parallel propagation scheme [5, 6, 55, 134], all of them still inherited from the original Patchmatch that one propagates sequentially across the whole image.

Instead, we propose a new *diffusion*-like scheme specifically tailored to multi-core architectures such as GPU processors. We partition the pixels into a "red" and "black" group in a checkerboard pattern, and simultaneously update all black and all red ones in turn. Possible candidates for the update at a given pixel are only points in a local neighborhood that belong to the respective other (red/black) group, see Fig. 17.

The red-black (RB) scheme is a standard trick to parallelize message-passing type updating schemes, *c.f.* the red-black Gauss-Seidel method for linear equation solving. Red-black acceleration has also been proposed for Belief Propagation [32]. In fact Patchmatch can be interpreted as a form of Belief Propagation in the continuous space [9]. In contrast to these applications of the RB-scheme we look beyond the immediate neighbors. Our standard pattern uses 20 local neighbors for propagation, Fig. 17. Thanks to the larger neighborhood we

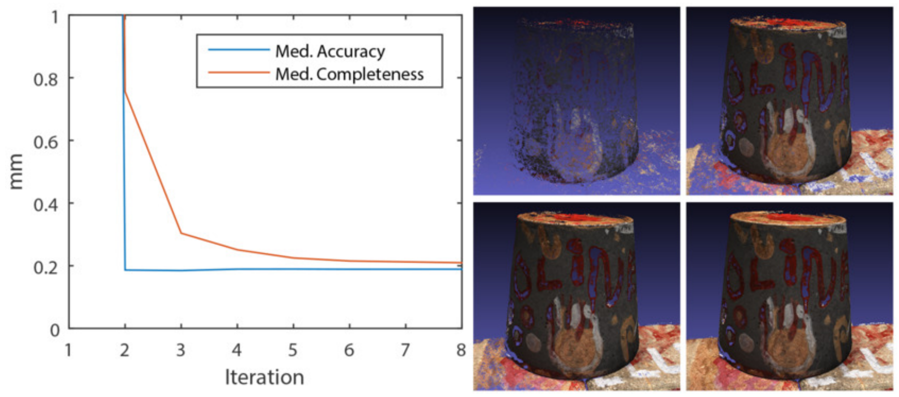


Figure 18: *Left:* Accuracy and completeness for increasing number of iterations for the object visualized on the right. *Right:* Reconstruction after iteration 2, 3, 4 and 8.

converge to a good solution already with a low number of iterations, see Fig. 18. The depicted scheme turned out to be a good compromise between the cost for each propagation step and the number of iterations needed to diffuse the information far enough. The number of iterations is fixed to 8 in all our experiments. At this point the depth map has practically converged and changes only marginally.

4.2.2 *Sparse matching cost*

We use a similar matching cost as proposed in the original Patchmatch paper [11]. The only difference is that we consider only intensity rather than color differences. The performance improvement when using RGB is tiny and in our view does not justify a threefold increase in runtime. To further speed up the computation we follow the idea of the so-called Sparse Census Transform [136] and use only every other row and column in the window when evaluating the matching cost, resulting in a 4× gain. Empirically, we do not observe any decrease in matching accuracy with this sparse cost.

The method is particularly useful for Patchmatch-type methods. Such methods require larger window sizes, because compared to the disparity a larger neighborhood is needed to reliably estimate the normal. Depending on the image scale, the necessary window size is typically at least 11x11 pixels, but can reach up to 25x25 pixels.

4.2.3 *Implementation details*

We have implemented ***Gipuma*** in CUDA, and tested it on recent gaming video cards for desktop computers. For our experiments we use the Nvidia Titan X. Images are mapped to texture memory, which provides hardware-accelerated bilinear interpolation to warp the support window between views. To limit the latency when reading from GPU memory we make extensive use of *shared memory* and cache the support window of the reference camera. We release our code as open-source software under the GPLv3 license.

RUNTIME. The runtime of our method is influenced mainly by three factors: the number of images considered for matching, the image resolution, and the size of the matching window (which in practice is roughly proportional to the image size).

For images of resolution 1600×1200 the runtime to generate a single depthmap with 10 images and window size of 25 is 14 seconds, when using our fast setting as described in Sec. 4.4.1 and windows size 15 the runtime for the same number of images is 1.5 seconds.

To generate a Middlebury depthmap from 10 views with a resolution of 640×480 the runtime is 1.5 seconds.

ALGORITHM SCALABILITY We show here how our method scales with the improvement of GPU technology. Even if it is difficult to quantify the improvement between successive generation of video cards, the (Giga) FLOPS is defined as:

$$\mathsf{FLOPS} = \mathsf{sockets} \times \frac{\mathsf{cores}}{\mathsf{socket}} \times \frac{\mathsf{cycles}}{\mathsf{second}} \times \frac{\mathsf{FLOPs}}{\mathsf{cycle}} \tag{22}$$

and provides a generic unit of measure of computing. We plot the runtime w.r.t. different generation of video cards along with their GigaFLOPS, see Fig. 19. It shows first of all that our parallel algorithm benefits directly from improvement of the underlining hardware. Then, when comparing the two plots, it shows the same trend of improvement, highlighting the proportional relation between GFLOPS and runtime of our method.

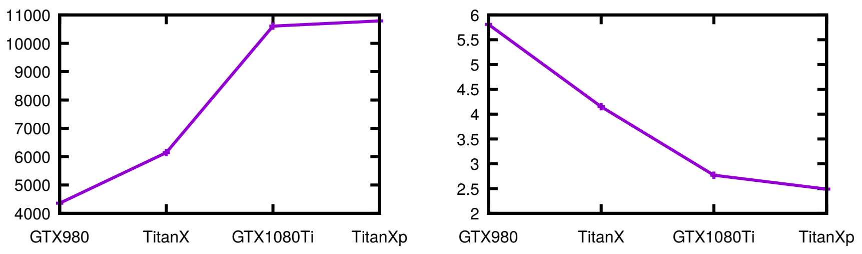


Figure 19: Comparison across different GPUs ordered by release date. Note the similar trend of GigaFLOPS (Left) w.r.t. Runtime in second (Right).

GPU	GigaFLOPS
GTX 980	4358
Titan X	6144
GTX 1080Ti	10608
Titan Xp	10790

Table 2: GigaFLOPS for GPUs used during runtime benchmark.

4.3 MULTI-VIEW EXTENSION

4.3.1 *Parametrization in scene space*

Disparity, by definition, is specific to a pair of rectified images. Instead, we propose to operate with planar patches in Euclidean scene space. This variant has several advantages. First, it avoids epipolar rectification, respectively explicit tracing of epipolar lines, which is a rather unnatural and awkward procedure in the multiview setup. Second, it delivers, as a by-product, a dense field of surface normals in 3D scene space. This can be used to improve the subsequent point cloud fusion (*e.g.* one can filter pixels with consistent depth but inconsistent normal) as well as directly provide the necessary normal used for surface reconstruction [76]. Then, it allows the data cost to directly aggregate evidence from multiple views: the cost per-pixel is computed by considering the cost of the reference camera with respect to all the other selected views.

Finally, the modification comes at little extra cost: the mapping between any two images is a plane-induced homography [52], corresponding to a 3D matrix-vector multiplication, see Fig. 20.

In the Euclidean scene-space the plane equation $\mathbf{n}^\top\mathbf{X} = -d$ holds for 3D object points $\mathbf{X} = [X, Y, Z]^\top$. Finding the object point amounts to intersecting the viewing ray with the plane in space. W.l.o.g. one can place the reference camera at the coordinate origin. With the intrinsic calibration matrix K, the depth at a pixel $\mathbf{x} = [x, y]^\top = [K|\mathbf{o}]\mathbf{X}$ is then related to the plane parameters by

$$Z = \frac{-dc}{[x-u,\ \alpha(y-v),\ c] \cdot \mathbf{n}}\ ,\ K = \begin{bmatrix} c & 0 & u \\ 0 & c/\alpha & v \\ 0 & 0 & 1 \end{bmatrix}. \tag{23}$$

where u, v is the principal point in pixels and c, $\frac{c}{\alpha}$ represent the focal length of the camera in pixels.

The image point $\mathbf{x}$ in the reference camera $K[I|\mathbf{o}]$ is then related to the corresponding point $\mathbf{x}'$ in a different camera $K'[R|\mathbf{t}]$ via the plane-induced homography, as introduced in Chapter 3:

$$H_\pi = K'(R - \frac{1}{d}\mathbf{t}\mathbf{n}^\top)K^{-1} \quad , \quad \mathbf{x}' = H_\pi\mathbf{x}\,. \tag{24}$$

INITIALIZATION. When operating in scene space, one has to take some care to ensure a correct, unbiased random initialization of the Patchmatch solver. To efficiently generate random normals that are uniformly distributed over the visible hemisphere we follow [91]. Two values q_1 and q_2 are picked from a uniform distribution in the interval $(-1, 1)$, until the two values satisfy $S = q_1^2 + q_2^2 < 1$. The mapping

$$\mathbf{n} = \left[1 - 2S\ ,\ 2q_1\sqrt{1-S}\ ,\ 2q_2\sqrt{1-S}\,\right]^\top \tag{25}$$

then yields unit vectors equally distributed over the sphere. If the projection $[u, v, c]^\top\mathbf{n}$ onto the principal ray is positive, the vector $\mathbf{n}$ is inverted.

Furthermore, one should account for the well-known fact that the depth resolution is anisotropic: even if the matching is parametrized in scene space, the similarity is nevertheless measured in image space. It follows that the measurement accuracy is approximately constant over the disparity range, respectively inversely proportional to the depth. Therefore it is advisable to uniformly draw samples from the range of possible disparities and convert them to depth values (*i.e.* supply a more densely sampled set of depths to chose from

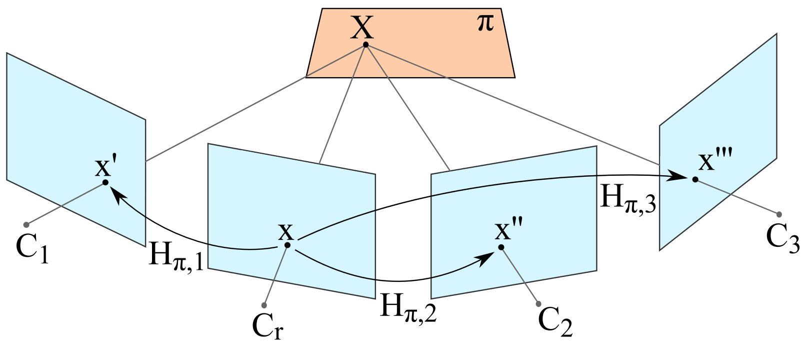


Figure 20: Multi-view setup with four cameras and homographies from reference camera C_r to three other cameras.

in the near field, where they make a difference; and a sparser set in the far field, where small variations do not produce an observable difference). For the same reason, the search interval for the plane refinement step should be set proportional to the depth.

4.3.2 *Cost computation over multiple images*

When using multiple views, the question arises how to best combine the pairwise dissimilarities between images into a unified cost. In our implementation, we only consider the pairwise similarities between the reference image and all other overlapping views, but not those between pairs of non-reference images.

VIEW SELECTION For a given reference image, we first exclude all views whose viewing directions differ from the reference image by less than α_{min} or by more than α_{max}. The two thresholds correspond to the empirical observation that baselines $< \alpha_{max}$ are too small for triangulation and lead to overly high depth uncertainty, whereas baselines $> \alpha_{max}$ have too big perspective distortions to reliably compare appearance [116]. The selection of α_{min} and α_{max} is dataset dependent.

In big datasets where the angle criteria still produces too many views, we propose to randomly pick a subset S of views within this selection only if the runtime performance is preferred over accuracy, see Sec. 4.4. When used, we set $S = 9$.

COST AGGREGATION For a specific plane π, we obtain a cost value m_i from each of the N comparisons. There are different strategies how to fuse these into a single multiview matching cost.

One possible approach is to accumulate over all n cost values, as proposed by Okutomi and Kanade [94]. However, if objects are occluded in some views, these views will return a high cost value even for the correct plane, and thereby blur the objective. In order to robustly handle such cases we follow Kang et al. [72]. They propose to include only the best 50% of all N cost values, assuming that at least half of the images should be valid for a given point. We slightly change this and instead of the fixed 50% introduce a parameter K, which specifies the number of individual cost values to be considered,

$$m_{srt} = \text{sort}_{\uparrow}(m_1 \dots m_N) \quad , \quad m_{mv} = \sum_{i=1}^{K} m_i \ . \tag{26}$$

The choice of K depends on different factors: in general, a higher value will increase the redundancy and improve the accuracy of the 3D point, but also the risk of including mismatches and thereby compromising the robustness. Empirically, rather low values tend to work better, in our experiments we use $K = 3$ or less for very sparse datasets.

4.3.3 *Fusion*

Like other multiview reconstruction schemes, we first compute a depth map for each view by consecutively treating all N views as the reference view. Then, the N depth maps are fused into a single point cloud, in order to eliminate wrong depth values and to reduce noise by averaging over consistent depth and normal estimates. Our approach follows the philosophy to generate the best possible individual depth maps, and then merge them into a complete point cloud straightforwardly. This allows and additional higher lever of parallelism: it is possible to compute independently each depthmap on a different GPU, potentially reducing the cost of depthmap computation to a factor equal to the number of GPU available.

CONSISTENCY CHECK Mismatches occur mainly in textureless regions and at occlusions, including regions outside of a camera's viewing frustum. Many such cases can be detected, because the depths estimated w.r.t. different viewpoints are not consistent with each other. To detect them, we again declare each image in turn the reference view, convert its depth map to a dense set of 3D points and reproject them to each of the

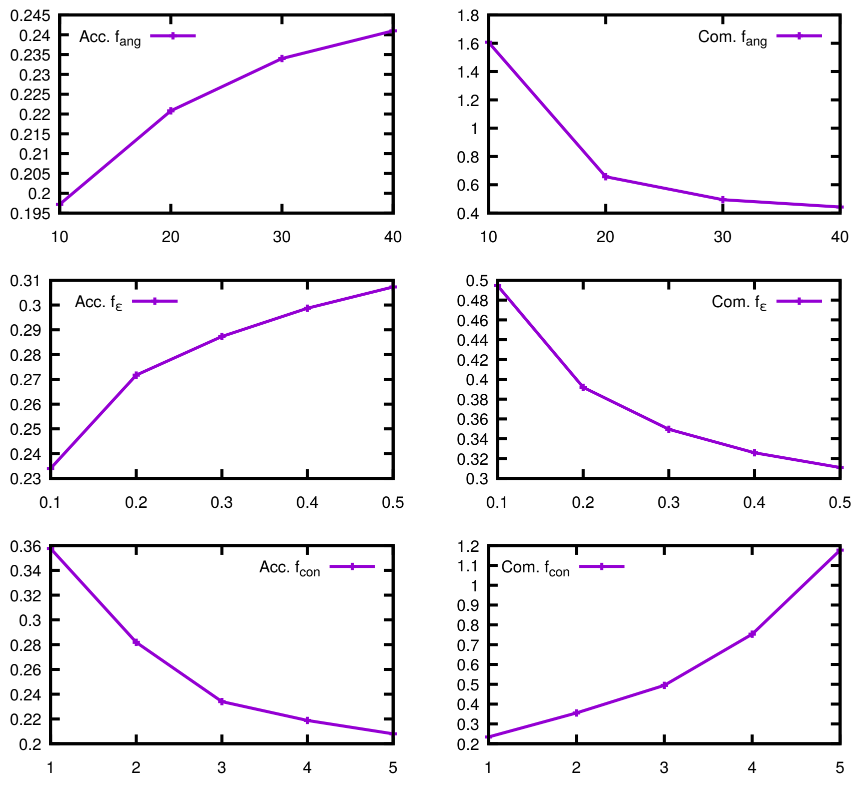


Figure 21: Behavior of depthmap fusion w.r.t. different values of $f_{ang}, f_{disp}, f_\epsilon$ see Sec. 4.3.3. Note how the consistency check step has a great impact on the completeness or accuracy of the method. **Left**: Accuracy. **Right**: Completeness. **Top**: Different values for the consistency check on the angle between close 3D points. **Middle**: Parameter test over multiple thresholds for the depth value in space. **Bottom**: Consistency enforced over different number of images.

$N - 1$ other views, resulting in a 2D coordinate $\mathbf{p}_i$ and a disparity value $\hat{d}_i$ per view. A match is considered consistent if $\hat{d}_i$ is equal to the disparity value d_i stored in the corresponding depth map, up to a tolerance of f_ϵ pixels. The threshold depends on the scale of the reconstructed scene. We further exploit the estimated surface normals and also check that the normals differ by at most f_{ang}, in our experiments set to 30°. If the depth in at least f_{con} other views is consistent with the reference view, the corresponding pixel is accepted. Otherwise, it is removed. For all accepted points the 3D position and normal are averaged directly in scene space over all consistent views to suppress noise.

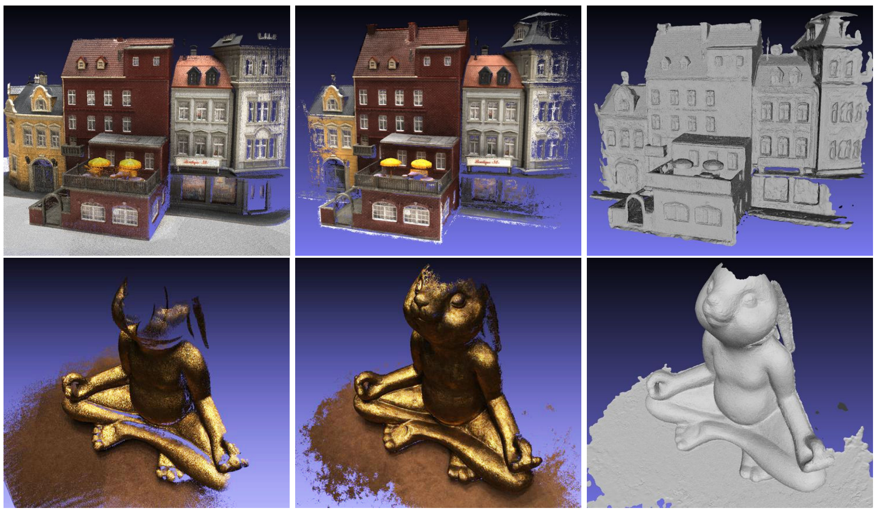


Figure 22: Reconstruction results of two DTU objects. From left to right: ground truth point cloud, textured point cloud and triangulated mesh surface.

ACCURACY VS. COMPLETENESS The fusion parameters f_ϵ , f_{ang} and f_{con} filter out 3D points that are deemed unreliable, and thus balance accuracy against completeness of the multiview reconstruction. Different applications require a different trade-off (*e.g.*, computer graphics applications often prefer complete models, whereas in industrial metrology sparser, but highly accurate reconstructions are needed). We explore different setting in our experiments, see Sec. 4.4. Note that the fusion step is very fast (≈ 5 seconds for 49 depthmaps of size 1600×1200) and does not change the depthmaps. One can thus easily switch from a more accurate to a more complete reconstruction, or even explore different levels interactively.

Consistency checking is an important step to remove outliers. For some applications, such as architectural surveying, the main concern is a correct, outliers-free result, whereas missing points and small holes are acceptable. In such cases it is recommended to stop at this point and triangulate only the consistent points into a 3D point cloud. In fact a main reason why in these application fields 3D point clouds are preferred over surface models is that the latter inherently rely on "guessing" the missing geometry. If, on the contrary a complete and hole-free model is preferred, such as in many computer graphics applications, the values that are missing due to the consistency check can be filled in the Poisson reconstruction.

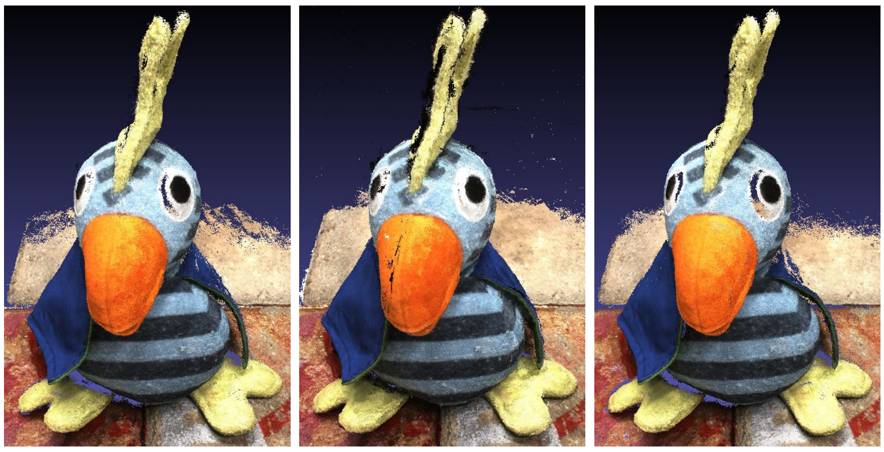


Figure 23: Reconstruction results for our three different settings. From left to right: *ours*, *ours comp*, *ours fast*. Note how the complete version is able to close the holes around the eye but suffers from boundary artifacts along the crest. On the other hand, the fast version, similar to the original, presents bigger holes around the eye and on the right side of the mantle.

4.4 RESULTS

We evaluate our multiview stereo GPU implementation on different datasets. We start with quantitative results on the recent DTU dataset for large scale multiview stereo [66]. To put our method in context we also evaluate on the Middlebury multiview benchmark [106], although the images in the dataset are very small by today's standards, and performance levels have saturated. When a triangulated mesh is required, we directly use our point cloud and normals with *Screened Poisson* reconstruction [76] with the program provided by the authors.

Additional qualitative results on aerial images and on KITTI dataset [45] are shown in Sec. 4.4.3 and 4.4.4 to demonstrate the broad applicability of our method.

4.4.1 *DTU Robot Image Dataset*

As our main testbed, we use the recent DTU large scale multiview dataset [66]. It contains 80 different objects, each covered by 49–64 images of resolution 1600×1200 pixels. The captured scenes have varying reflectance, texture and geometric properties and include fabric, print, groceries, fruit and metallic sculptures, see Fig. 22. The images have been captured with different lighting conditions, and with two different distances to the object, using a robot arm to accurately position the cameras. We use only the most diffuse lighting to select the same set of images as used by the other methods. The ground truth has been acquired with a structured light scanner.

We followed the protocol specified by the authors of the dataset, *i.e.* we compute the mean and median reconstruction errors, both for the estimated 3D point cloud and for a triangulated mesh derived from the points. Accuracy is defined as the distance from the surface to the ground truth, and completeness from the ground truth to the surface. In this way completeness is expressed in mm and not as a percentage.

Compared to other methods, we achieve the highest accuracy, marked as *ours* in Tab. 3, while at the same time delivering the second-highest completeness, behind [15] which has much lower accuracy. For this setting we employ $f_\epsilon = 0.1$ and $f_{con} = 3$ for fusion.

There is always a trade-off between accuracy and completeness, which depends on how strict one sets the thresholds for rejecting uncertain matches. We thus also run our method with different setting for the fusion, chosen to achieve high completeness (*ours comp* in Tab. 3) with $f_\epsilon = 0.3$ and $f_{con} = 2$. In that setting we surpass [15] in terms of completeness, while still achieving the second best accuracy, slightly below [115] which is a lot sparser. The runtime is ≈**14 seconds** per depthmap when the number of selected views is 10, growing linearly as more views are considered. The full reconstruction is obtained in 15 minutes.

SPEED SETTINGS To explore the behavior of our method when tuned for high speed, we also tried an extreme setting. To speed up the reconstruction we set the window size to 15, restrict the similarity computation in the window to every fourth row and column, and use at most 10 (randomly chosen) views within the view selection criteria for a depthmap. Furthermore, we stop the propagation after 6 iterations instead of 8 and use a reduced set of update candidates in the propagation step, as depicted in Fig. 17. For fusion, we used the parameters $f_\epsilon = 0.3$ and $f_{con} = 3$. With these settings, the method needs ≈**1.5 seconds** per depthmap on the GPU, respectively 5 minutes per complete object (including disk I/O). These extreme settings do lead to some loss in completeness, whereas the accuracy

		Accuracy		Completeness	
		Mean	Med.	Mean	Med.
Points	ours	**0.273**	**0.196**	0.687	0.260
	ours comp	0.379	0.234	**0.400**	0.188
	ours fast	0.289	0.207	0.841	0.285
	tola [115]	0.307	0.198	1.097	0.456
	furu [37]	0.605	0.321	0.842	0.431
	camp [15]	0.753	0.480	0.540	**0.179**
Surfaces	ours	**0.363**	**0.215**	0.766	0.329
	ours comp	0.631	0.262	**0.519**	**0.309**
	ours fast	0.358	0.221	0.939	0.350
	tola [115]	0.488	0.244	0.974	0.382
	furu [37]	1.299	0.534	0.702	0.405
	camp [15]	1.411	0.579	0.562	0.322

Table 3: Quantitative comparison with three different settings on the DTU dataset [66]. The quality metrics accuracy and completeness were computed in accordance to [66], stating the mean and median error in millimeters.

remains high. Even when tuned for speed the method is competitive with the state of the art, see row *ours fast* in Tab. 3.

Fig. 23 presents a qualitative comparison of the three different parameter settings presented.

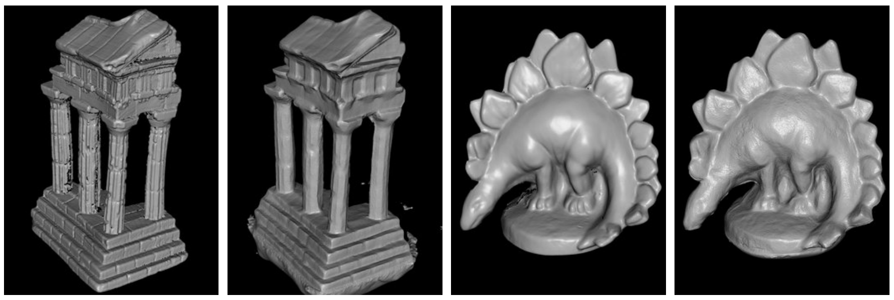


Figure 24: Ground truth surfaces and reconstructions for *Temple Full* and *Dino Full* of the Middlebury multiview stereo evaluation dataset [106].

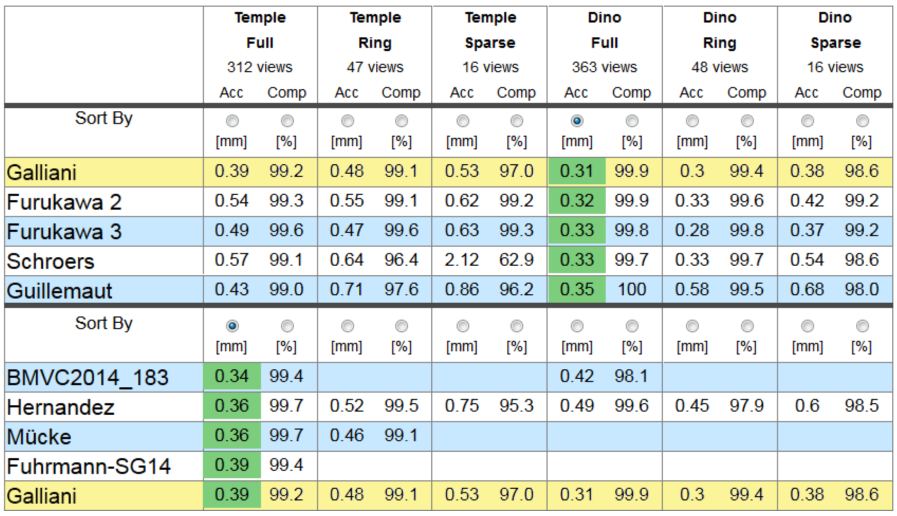
4.4.2 *Middlebury*

We evaluated our method also on the popular Middlebury multiview benchmark [106]. It was the first benchmark for multiview reconstruction, and by now is rather saturated. Moreover, the images are rather small at 640×480 pixels, and the dataset consists of only two objects, in three different settings with varying number of views. Fig. 24 shows our reconstructions of the two objects *Dino* and *Temple* ("full" setting with more than 300 views each).

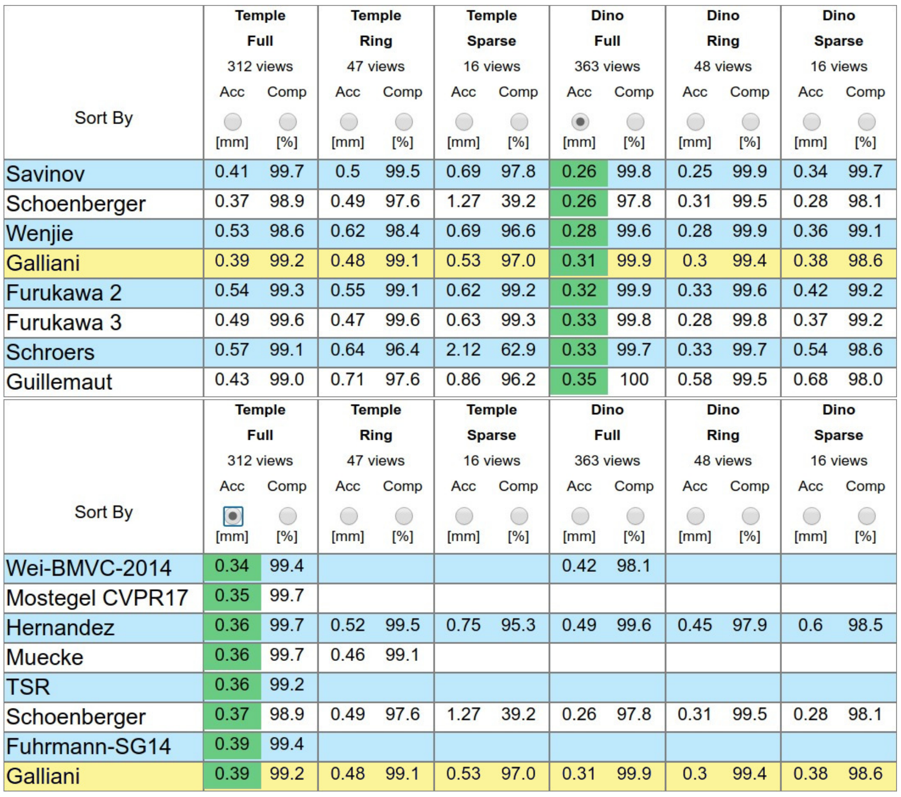
On *Dino* we rank 4th for the "full" version, 5th for the "ring" version, and 10th for the "sparse" version. On *Temple* we rank 8th on "full", 7th on "ring" and 5th on "sparse". For all six reconstructions the completeness lies between 97.0 and 99.9%, see Fig. 25. It is interesting to note that several methods perform well only on one of the two objects, or even provide reconstruction for only one of the 6 set of images provided, possibly tuning their method for that specific challenge. A fair comparison is then among methods providing a reconstruction for all the competitions. We achieve excellent performance on both datasets, in terms of both accuracy and completeness. To generate a depthmap from 10 views for (resolution of 640×480) our method needs 1.5 seconds, with window size 11. The complete reconstruction, including mesh triangulation, for the Dino and Temple "full" version is obtained in 35 minutes, one of the fastest available.

4.4.3 *Outdoor Images*

We have also tested our method on oblique aerial images from the city of Enschede. Oblique viewing angles are becoming popular in aerial mapping to cover vertical structures such as building facades, and are an ideal testbed for us. Aerial mapping images are routinely recorded in such a way that multiple overlapping views are available. They present a challenge for conventional matching methods, because the depth range is much larger compared to conventional nadir views. And they deviate strongly from the fronto-parallel setup assumed by many matching algorithms, but do tend to have piecewise constant normals, *e.g.* on the ground, building walls, roofs *etc*. The results in Fig. 29 highlight the properties of *Gipuma*: planes in general orientation are recovered without stair-casing artifacts (see the reconstruction with color coded normals); matching



	Temple		Temple		Temple		Dino		Dino		Dino	
	Full		Ring		Sparse		Full		Ring		Sparse	
	312 views		47 views		16 views		363 views		48 views		16 views	
	Acc	Comp	Acc	Comp	Acc	Comp	Acc	Comp	Acc	Comp	Acc	Comp
Sort By	[mm]	[%]	[mm]	[%]	[mm]	[%]	● [mm]	[%]	[mm]	[%]	[mm]	[%]
Galliani	0.39	99.2	0.48	99.1	0.53	97.0	0.31	99.9	0.3	99.4	0.38	98.6
Furukawa 2	0.54	99.3	0.55	99.1	0.62	99.2	0.32	99.9	0.33	99.6	0.42	99.2
Furukawa 3	0.49	99.6	0.47	99.6	0.63	99.3	0.33	99.8	0.28	99.8	0.37	99.2
Schroers	0.57	99.1	0.64	96.4	2.12	62.9	0.33	99.7	0.33	99.7	0.54	98.6
Guillemaut	0.43	99.0	0.71	97.6	0.86	96.2	0.35	100	0.58	99.5	0.68	98.0
Sort By	● [mm]	[%]	[mm]	[%]	[mm]	[%]	[mm]	[%]	[mm]	[%]	[mm]	[%]
BMVC2014_183	0.34	99.4					0.42	98.1				
Hernandez	0.36	99.7	0.52	99.5	0.75	95.3	0.49	99.6	0.45	97.9	0.6	98.5
Mücke	0.36	99.7	0.46	99.1								
Fuhrmann-SG14	0.39	99.4										
Galliani	0.39	99.2	0.48	99.1	0.53	97.0	0.31	99.9	0.3	99.4	0.38	98.6



	Temple		Temple		Temple		Dino		Dino		Dino	
	Full		Ring		Sparse		Full		Ring		Sparse	
	312 views		47 views		16 views		363 views		48 views		16 views	
	Acc	Comp	Acc	Comp	Acc	Comp	Acc	Comp	Acc	Comp	Acc	Comp
Sort By	[mm]	[%]	[mm]	[%]	[mm]	[%]	● [mm]	[%]	[mm]	[%]	[mm]	[%]
Savinov	0.41	99.7	0.5	99.5	0.69	97.8	0.26	99.8	0.25	99.9	0.34	99.7
Schoenberger	0.37	98.9	0.49	97.6	1.27	39.2	0.26	97.8	0.31	99.5	0.28	98.1
Wenjie	0.53	98.6	0.62	98.4	0.69	96.6	0.28	99.6	0.28	99.9	0.36	99.1
Galliani	0.39	99.2	0.48	99.1	0.53	97.0	0.31	99.9	0.3	99.4	0.38	98.6
Furukawa 2	0.54	99.3	0.55	99.1	0.62	99.2	0.32	99.9	0.33	99.6	0.42	99.2
Furukawa 3	0.49	99.6	0.47	99.6	0.63	99.3	0.33	99.8	0.28	99.8	0.37	99.2
Schroers	0.57	99.1	0.64	96.4	2.12	62.9	0.33	99.7	0.33	99.7	0.54	98.6
Guillemaut	0.43	99.0	0.71	97.6	0.86	96.2	0.35	100	0.58	99.5	0.68	98.0

	Temple		Temple		Temple		Dino		Dino		Dino	
	Full		Ring		Sparse		Full		Ring		Sparse	
	312 views		47 views		16 views		363 views		48 views		16 views	
	Acc	Comp	Acc	Comp	Acc	Comp	Acc	Comp	Acc	Comp	Acc	Comp
Sort By	● [mm]	[%]	[mm]	[%]	[mm]	[%]	[mm]	[%]	[mm]	[%]	[mm]	[%]
Wei-BMVC-2014	0.34	99.4					0.42	98.1				
Mostegel CVPR17	0.35	99.7										
Hernandez	0.36	99.7	0.52	99.5	0.75	95.3	0.49	99.6	0.45	97.9	0.6	98.5
Muecke	0.36	99.7	0.46	99.1								
TSR	0.36	99.2										
Schoenberger	0.37	98.9	0.49	97.6	1.27	39.2	0.26	97.8	0.31	99.5	0.28	98.1
Fuhrmann-SG14	0.39	99.4										
Galliani	0.39	99.2	0.48	99.1	0.53	97.0	0.31	99.9	0.3	99.4	0.38	98.6

Figure 25: Screenshots from the Middlebury evaluation for *Dino Full* and *Temple Full* sorted by accuracy at the standard threshold of 90%. The proposed multi view stereo method is highlighted in yellow. **Top:** Result at the time of submission to the website (November 2014). **Bottom:** Result as of October 2017.

is less reliable on structures without well-defined normals (*e.g.* trees in the near-field) but errors are detected and removed.

4.4.4 *KITTI*

We show additional results for the stereo dataset of the *KITTI Vision Benchmark Suite* [45]. The KITTI Vision Benchmark Suite [45] consists of 194 training and 195 test image pairs (rectified) with a resolution of 1240×376 pixels. It is well-known to provide a challenging, realistic test-bed due to outdoor lighting conditions. The experiment thus emphasizes that our method is not limited to (or even particularly tuned to) laboratory settings, but rather can deal with general stereo and multiview problems, also under uncontrolled lighting.

The dataset consists of outdoor images captured with a stereo rig from a moving car. Images include rural areas as well as streets around the city of Karlsruhe. Semi-dense ground truth disparity maps are provided for non-occluded areas as well as for the complete image (including regions that are occluded in the second view).

MULTI-VIEW KITTI In addition to computing results with the synchronous stereo pairs, we also process the data from three consecutive time steps together to obtain a 6-view reconstruction. This is possible, since the KITTI scenes are largely static. The intrinsic calibration of the two cameras as well as their relative pose are provided as part of the dataset. The relative rotation and translation of the stereo rig between consecutive frames were kindly provided by the authors of [118], who have in their work already computed the ego-motion of the stereo camera system. A similar ego-motion estimate was also described in [3].

QUALITATIVE RESULTS Exemplary reconstruction results for both 2 and 6 views are shown, to highlight the improvement obtained by collecting evidence from multiple images, see Fig. 26, 27, 28. As for two views, we show disparity maps from the left to the right image of the rig, at the second of the three adjacent time steps (reprojected from the estimated 3D points). We point out that the results were obtained without explicit smoothness assumptions, and without any post-processing. To reconstruct a single depthmap the computation takes $\approx$2.1 seconds for a stereo pair, respectively $\approx$6.5 seconds for six views. We are not able to show quantitative results on this dataset due to the lack of a smoothness assumption typical for stereo reconstruction algorithms which is necessary to obtain a dense stereo disparity.

4.4.5 *ETH3D*

Our method was run with default parameters on a recent multi-view benchmark, ETH3D [105], providing very high resolution images, multi camera videos with different field of views. To our surprise Gipuma was not performing well, ranking only 4th out of 6 submissions. This results could be explained by multiple factors. First of all probably the method was conceived for and tested only on a specific type of dataset, namely lab setting with a dense set of images covering the object. Little care was taken in the automatic selection of the best views to be used during fusion or matching, critical in Structure-From-Motion type of dataset, such as ETH3D. Also, the type of matching function, a simple combination of SAD and difference of gradients, is probably good enough for lab settings, but suffers on real world scenarios, where might be required a more robust similarity function.

We are nevertheless aware of the weakness of our method, and welcome the result of ETH3D as an incentive to make our method more robust for every type of scenario.

4.5 CONCLUSION

We have presented ***Gipuma***, a massively parallel multiview extension of Patchmatch stereo. The method features a new red-black propagation scheme tailored to modern GPU processing, and exploits multiview information during matching. Switching to a common 3D scene coordinate system, in which all camera poses reside, makes it possible to directly integrate information from multiple views in the matching procedure, via the planar homographies induced by a point and its associated normal vector. Like the original Patchmatch stereo, the method is based on slanted planes, and therefore allows slanted support windows without fronto-parallel bias. It is thus particularly well-suited for the frequent case of locally smooth scenes with large depth range. As a by-product, the resulting multiview matcher delivers not only dense depth maps, but also dense normal vec-

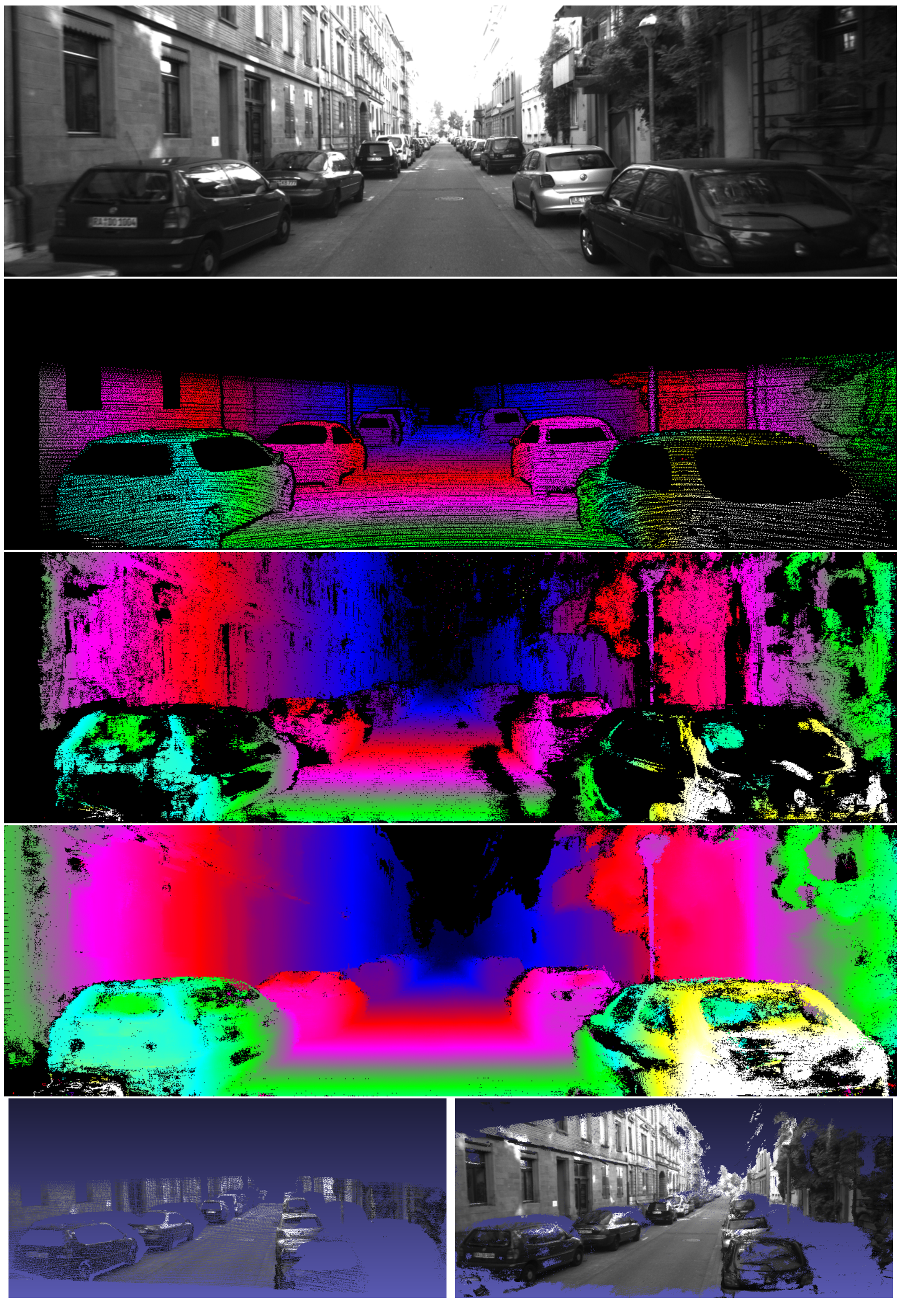


Figure 26: *From top to bottom:* Input image number 63 of KITTI training dataset. Ground truth disparity. Disparity with a stereo pair. Disparity with six input images. *Bottom left:* Ground truth point cloud. *Bottom right:* Reconstructed point cloud.

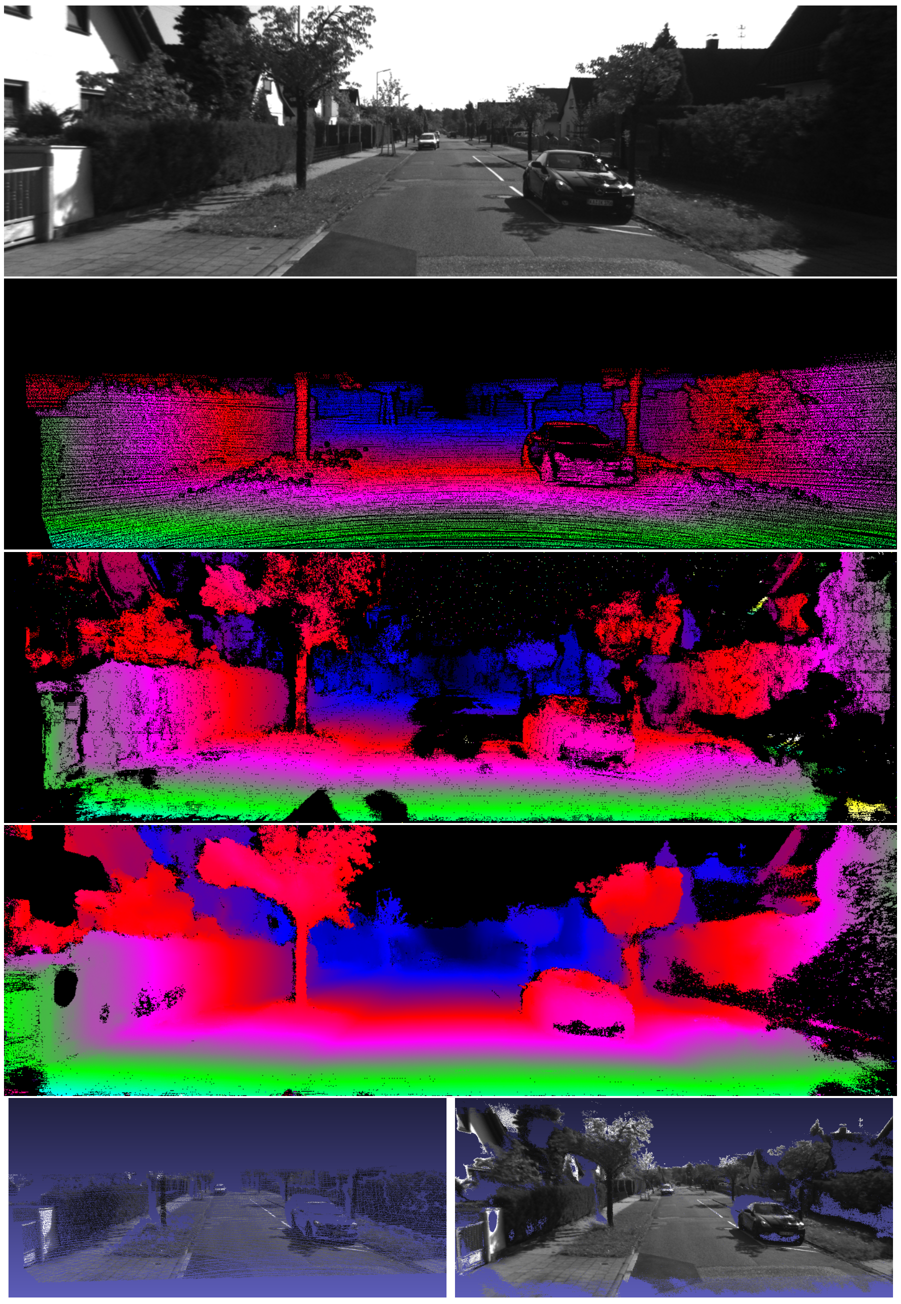


Figure 27: *From top to bottom:* Input image number 76 of KITTI training dataset. Ground truth disparity. Disparity with a stereo pair. Disparity with six input images. *Bottom left:* Ground truth point cloud. *Bottom right:* Reconstructed point cloud.

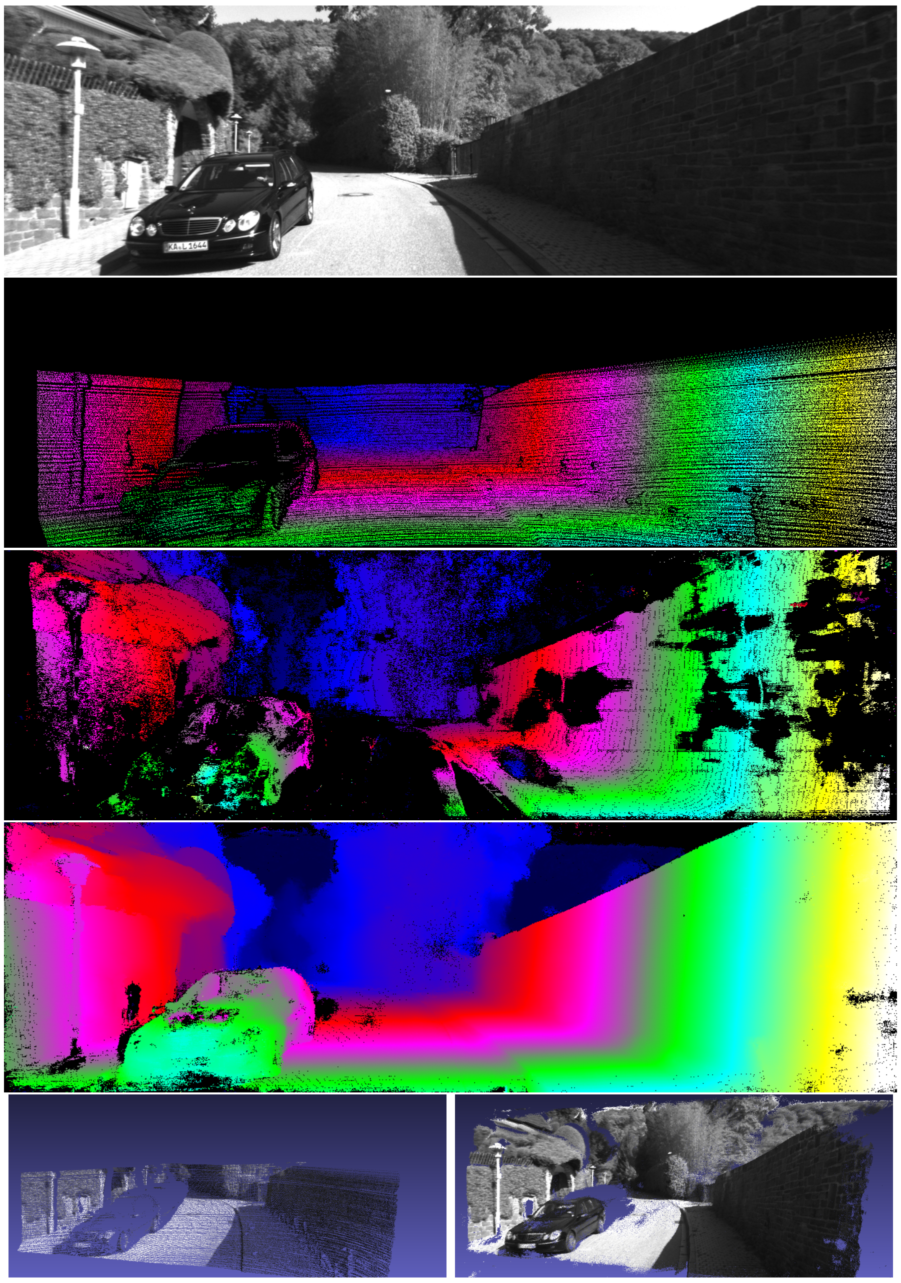


Figure 28: *From top to bottom:* Input image number 126 of KITTI training dataset. Ground truth disparity. Disparity with a stereo pair. Disparity with six input images. *Bottom left:* Ground truth point cloud. *Bottom right:* Reconstructed point cloud.

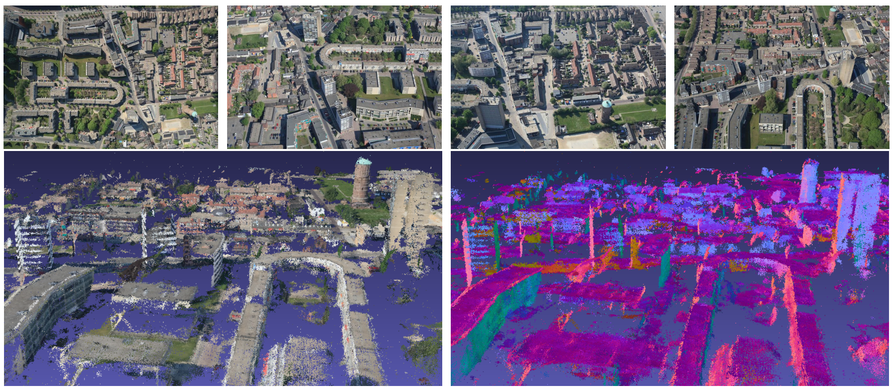


Figure 29: Point clouds generated from aerial images. *Top:* selection of input images. *Bottom:* textured and normal color coded point clouds.

tors in metric scene space. Our method achieves accurate and complete reconstruction with low runtime.

The computational cost of the method scales linearly with the number of pixels in an image. Its memory consumption is also linear in the number of pixels, and independent of the disparity range. These properties make it attractive for large images.

Gipuma is released to the community as open-source software[1].

1 http://github.com/kysucix/gipuma

5 LEARNED MULTI-PATCH SIMILARITY

This chapter is based on Hartmann et al. *[53].*

3D reconstruction from two or more images of the same scene is a central problem in computer vision. Assuming that the camera poses are already known, the problem reduces to (multi-view) stereo matching, *i.e.*, establishing dense point correspondences between the images, which can then be converted to 3D points by triangulating the corresponding rays. The core of stereo matching itself is a function to measure the similarity between points in different images, respectively between the points surrounding image patches. Once such a *similarity measure* is available, it can be computed for a list of different putative correspondences to find the one with the highest similarity.

For the classic two-view stereo case, the definition of a similarity function is comparatively straight-forward: transform the image intensities of the two patches such that more similar ones end up closer to each other, according to some pre-defined distance metric. Many methods have been proposed, including simple sum-of-squared differences, (inverse) normalized cross-correlation to afford invariance against linear brightness changes, and even more robust measures like the Hamming distance between Census descriptors. More recently it has also been proposed to learn the distance metric discriminatively from matching and non-matching training examples.

In practice, multi-view stereo is often the method of choice, since the higher redundancy and larger number of viewpoints allows for more robust 3D reconstruction. A host of rather successful multi-view stereo methods exist (see benchmark results such as [67, 104, 106, 113]). Surprisingly, these methods in fact have no mechanism to measure the similarity between >2 image patches that form a putative multi-view correspondence. Instead, they heuristically form a consensus over the pair-wise similarities, or a subset of them (most often the similarities from all other stereo partners to a "reference image" in which the depth map is parametrized) see Fig. 31. We note that the same is true for "multi-view stereo" methods that reconstruct

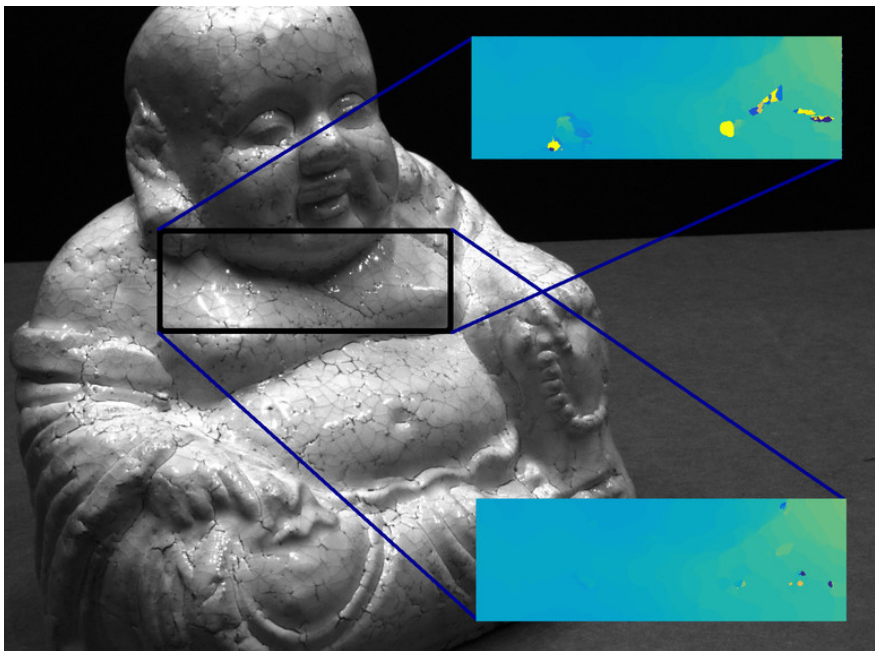


Figure 30: Multi-view depth estimation. A conventional, pairwise similarity like ZNCC *(top)* is unable to find the correct depth in corrupted regions, *e.g.* specular reflections; whereas the proposed multi-view similarity *(bottom)* can predict correct depth values.

implicit [129] or explicit [27] surfaces. These either reconstruct points from two images and delay the multi-view integration to the surface fitting stage; or they measure photo-consistency between pairs of images, or between some "mean" surface texture and the individual images.

Here, we pose the question *why not directly measure multi-view similarity?* Encouraged by the successes of learned similarity measures, we propose a multi-stream ("Siamese") convolutional neural network architecture that takes as input a number of $n > 2$ image patches, and outputs a scalar similarity score. The network is learned directly from matching and non-matching example patches, keeping decisions like the right weighting of individual images or image pairs (*e.g.*, to account for varying contrast) or the robustness of the consensus mechanism (*e.g.*, due to occlusions, specularities, and other disturbances in individual images) implicit. An alternative view of our work is as a multi-view extension of learning-based stereo correspondence [124, 130, 131] to more than two views. We posit that the learning-based stereo should profit from the multi-view setup, precisely because the

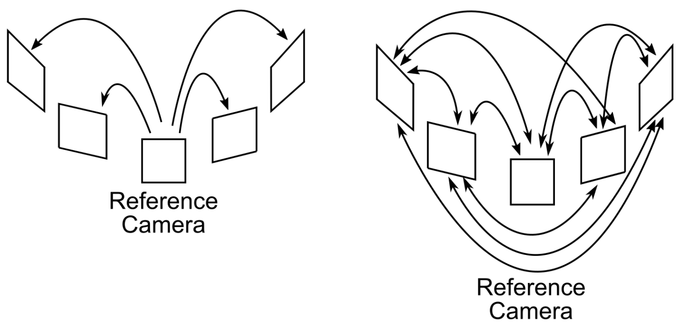


Figure 31: Left: standard similarity comparison of most multi-view stereo method. Right: how similarity computation should be done w.r.t. to the reference camera, and how is it done implicitly when our learned similarity is used.

additional images afford the necessary redundancy to detect and resolve situations where two-view stereo struggles. To test our similarity measure, we embed it into a standard depth estimation setup, namely multi-view plane-sweeping [23]: for each pixel in an image we compute similarity scores for a range of depths along the ray, and pick the depth with the highest similarity.

There are different strategies to cast stereo matching into a machine learning problem. One can either fix the metric (*e.g.*, Euclidean distance) and learn to map image patches to "descriptor vectors" that, according to that metric, have small distance for matching patches and large distance for non-matching patches [124]. For our case, that approach does not resolve the problem of defining an n-view distance. Alternatively, one can map raw patches to descriptors according to some conventional recipe, *e.g.* SIFT, and train a similarity/dissimilarity metric between them [127]. However, given the spectacular improvements that learned features from CNNs have brought to computer vision, we prefer to learn the mapping end-to-end from raw pixel intensities to a similarity score.

Conceptually, it is straight-forward to design a CNN for multi-view similarity. A standard architecture to jointly process two images with similar image statistics are "Siamese" networks: the two inputs are passed through identical streams with tied weights and then combined for the final decision layers by simple addition or concatenation. We do the same for $n > 2$ images and set up n parallel streams with tied weights, without introducing additional free parameters. The network is trained on a portion of the public DTU multi-view dataset [67], and evaluated on the remaining part of it, as well as on an unrelated public dataset. We will also show that it is possible to vary the number n of input image patches at test time without retraining the network. The comparison to other conventional and learning-based approaches demonstrates the benefit of evaluating direct multi-view similarity, especially in the case when the reference view is corrupted, *e.g.*, due to specular reflection – see Figure 37.

5.1 n-WAY PATCH SIMILARITY WITH A NEURAL NETWORK

We aim for a function that directly maps $n > 2$ image patches p_i to a scalar similarity score $S(p_1, p_2, \ldots, p_n)$. Our proposed solution is to learn that function from example data, in the spirit of what is sometimes called "metric learning".[1] As learning engine, we use a convolutional neural network. We point out that the strategy to learn a multi-patch similarity is generic, and not limited to stereo correspondence. In fact, a main argument for learning it is that a learned score can be tuned to different applications, just by choosing appropriate training data. In our target application, the n patches are the reprojections of a candidate 3D point into n different views of the scene.

For our purposes, we make the assumption that every set of patches in the training data is either "similar" or "dissimilar", and do not assign different degrees of similarity. *I.e.*, we cast the similarity as a binary classification problem. Conveniently, this means that the similarity score is bounded between 0 (maximally dissimilar) and 1 (maximally similar). Such a hard, discriminative approach reflects the situation of stereo matching (and correspondence estimation in general), where for any given pixel in a reference image there is only one correct answer, namely the set of patches that correspond to the ground truth depth. We note that for other applications, for example image retrieval, the definition may not be suitable and would have to be replaced with a more gradual, continuous

1 We refrain from using that name, since the learned similarity score is not guaranteed to be a metric in the mathematical sense.

one (for which it is however not as straight-forward to generate training labels).

5.1.1 *Network Architecture*

The network we use for learning a depth map is illustrated in Figure 32. Its inputs are n image patches (w.l.o.g. we set $n = 5$ for the remainder of the chapter) of size 32×32 pixels. The early layers process each patch separately with the same set of weights, corresponding to an n-way Siamese network architecture. Identical weights in the different branches are a natural choice, since the result should not depend on the order of the input patches. Note that the number of free weights to be learned is independent of the number n of views.

Each branch starts with a convolutional layer with 32 filter kernels of size 5×5. Then follow a $\mathtt{tanh}$ non-linearity and a $\mathtt{max}$-pooling layer with kernel size 2×2. That same sequence is then repeated, this time with 64 filter kernels of size 5×5, and $\mathtt{max}$-pooling over 2×2 neurons. From the resulting $5 \times 5 \times 64$ layers the mean is taken over all n branches and passed through two convolutional layers with 2048 filter kernels, each followed by a $\mathtt{ReLU}$ non-linearity, and a final convolutional layer with 2 filter kernels. The final network output is the similarity score.

The described, fully convolutional architecture has the advantage that it is much more efficient to compute for larger images at test time than a per-pixel similarity. Due to the two pooling layers, the network outputs a similarity score for every 4×4 pixel region.

When designing the network, the straightforward approach would be to concatenate the outputs of the individual branches, instead of averaging them. We evaluate both options and find their performance on par, see Tab. 5. Note that averaging the branch outputs makes it possible to input a varying number of views without retraining the network.

As usual, the exact network design is found empirically and is somewhat hard to interpret. We tested several architectures and found the described one to work best. Compared to other current architectures for image analysis, our network needs fewer convolutional layers, presumably because of the small input patch size; and, interestingly, old-school $\mathtt{tanh}$ non-linearities work better than $\mathtt{ReLU}$ in the early layers.

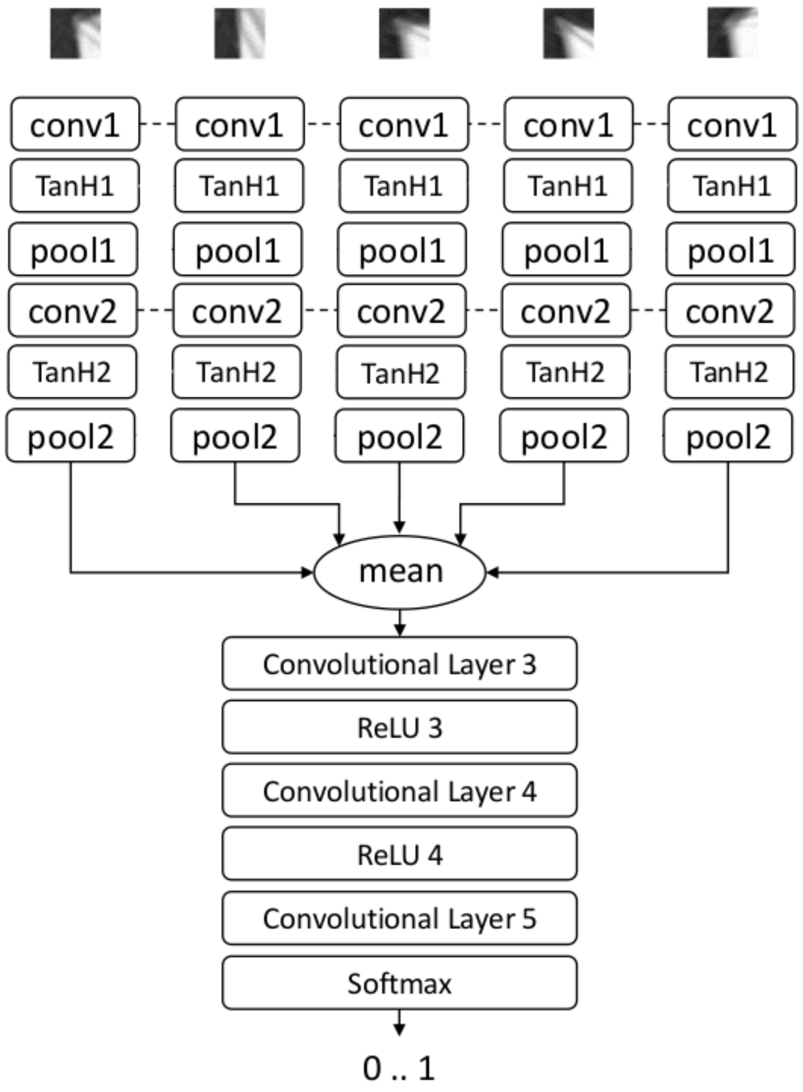


Figure 32: The proposed network architecture with five "Siamese" branches. The inputs are the five patches, the output is the similarity score that is used to select the correct depth value. Because the network is fully convolutional, *i.e.* contains only convolution and no fully connected layers, we are able to apply the network on a larger image patch and obtain directly the result for the corresponding shifted similarity value. This greatly reduce the runtime when applied on whole images by sharing the cost of convolution for nearby points.

5.2 APPLICATION TO MULTI-VIEW STEREO

Having introduced the multi-view similarity computation, we use it as a building block in a fairly standard multi-view stereo pipeline [50], assuming known camera poses (*e.g.*, from prior structure-from-motion computation). As usual, one camera is designated as the reference view for which the depth map is computed. We emphasize that, other than in most existing multi-view approaches, the reference view serves *only* to fix the parametrization of the depth map. In our method, its patches do not have a privileged role as the "source" templates to be combined to the "target" patches of the stereo partners in a pairwise fashion. Note, in a multi-view setup it can happen that points are occluded in the reference view, but visible in several other views. In that case the learned similarity score may assign the highest similarity to a point that is invisible in the reference image. If the final product of multi-view stereo is a 3D point cloud, this behavior does not hurt (except that the corresponding point on the occluder is missing).

To find the most likely depth for a given pixel, we discretize the depth along the viewing direction of the reference view, as in standard plane-sweep stereo. Matching then reduces to exhaustively testing all possible depth values and choosing the one with the highest similarity. For a given patch in the reference view the matching patches in the other images are extracted. This is repeated for all planes, where each plane corresponds to a discrete depth value. Each n-tuple of patches (including the reference patch, which is the same for every depth) is fed to the similarity network. Not surprisingly, rather larger patches give more reliable similarity estimates, but there is a trade-off against memory consumption and computational cost. We found 32×32 pixels to be a good compromise.

5.2.1 *Training the Network.*

Our network is implemented in Caffe [68], and learned from scratch. As training data, we sample image patches from 49 scenes of the DTU dataset [67] (Fig. 33).

Positive samples (similarity 1) are selected using the ground truth depth information. Negative samples are selected by picking patches from depth planes 15 steps before or after the ground truth. Note the power of the learning approach to optimally tune to the application: patches on correct epipolar

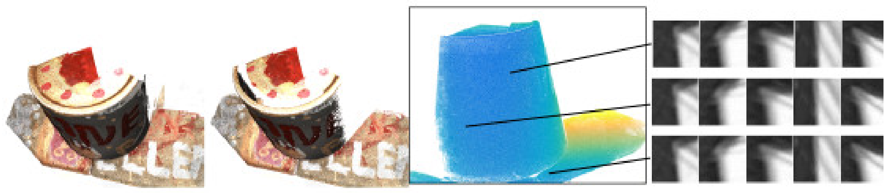


Figure 33: Generating training data. The ground truth 3D point cloud is processed using the visibility check of [74]. Points not visible in the reference view are removed. Next, points are randomly selected, projected into the image, and a patch centered at the projection is cropped out.

lines are the only ones the network will ever see at test time (assuming correct camera poses). Using only such patches for training ensures that the model is "complete", in the sense that it sees all sorts of patch combinations that can occur along epipolar lines; but at the same time it is also "tight" in the sense that it does not contain irrelevant negatives that do not correspond to any valid set of epipolar geometries and merely blur the discriminative power of the similarity.

It is a recurring question what proportion of positive and negative samples to use in situations with very unbalanced class distributions (for multi-view stereo, only one sample within the entire depth range is positive for each pixel). For our case it worked best to use a balanced training set in each training batch, which also speeds up convergence. In total, we sample 14.7 million positive and 14.7 million negative examples. Learning is done by minimizing the $\mathtt{softmax}$ loss w.r.t. the true labels with 500,000 iterations of standard Stochastic Gradient Descent (SGD), with batch size 1024; starting from a base learning rate of 0.001, that is reduced by a factor of 10 every 100,000 iterations.

DIFFERENCE WITH RESPECT TO LIFT Let's remark here a major difference w.r.t. pure learning based descriptors such as LIFT. Our method is trained directly on "real" points obtained from ground truth (captured with an accurate structure light scanner) and not on surface points results of a triangulation of matched featured descriptor such as LIFT (results of a Structure-From-Motion pipeline).

When using from data obtained from a Structure From Motion pipeline, we argue that the model is biased by the feature descriptor used, effectively training only on specific areas of the images rich of textures. The network then creates only a learned version of the descriptor, since it has no access to other part of the image, other then the ones detected by the descriptor. Instead, by randomly using data from the ground truth, the model makes use of every part of the image, even on parts without a unique pattern, and let the data emerge a similarity measure across images.

5.3 EXPERIMENTS AND RESULTS

To evaluate our learned similarity measure, we conduct multi-view depth estimation experiments on several scenes. In the first experiment, we demonstrate the performance of our approach using the DTU [67] evaluation framework and compare it to four existing baseline methods. Second, we test the similarity network's ability to generalize across different environments. Without retraining the network, we feed it with input images having different lighting conditions, depth range, and sensor characteristics than what was observed during training.

The way the similarity measure is employed to provide depth map predictions is the same for all the compared approaches. After picking a reference view, we find the four camera viewpoints closest to it. These five images and their camera poses are fed to a plane-sweeping routine – we use the publicly available code of [50]. The explored depth range is set according to the given dataset, and warped views are generated for 256 uniformly sampled disparity levels. The SAD and ZNCC similarity measures, already implemented in the plane-sweeping library, are used to select the best depth estimate for each pixel position in the reference view, based on the 4×256 pairwise similarity scores. To ensure a fair comparison with the proposed approach, the same patch size of 32×32 pixels is used.

For the other compared descriptors, SIFT [89] and LIFT [111, 124], we compute similarity via the pairwise (Euclidean) descriptor distances from the warped images, and feed them to the same decision mechanism to select the best per-pixel depth estimate. These descriptors were designed to work with patch size 64×64 pixels, so we use these dimensions, even though it gives them the advantage that they can base their prediction on $4\times$ more pixels. Note, the warping already corrects for scale and orientation, so we skip the corresponding parts and directly compute the descriptor entries from the warped images. We point out that this comparison is not completely fair, since the two descriptors were designed for sparse interest-point matching, where some translational invariance is desirable. Still, we wanted to test against LIFT, as the most recent learned, CNN-based descriptor. For completeness, we include its hand-crafted counterpart.

In order to have a computationally efficient depth map prediction with the proposed approach, we choose the input patch size to be 128×128 pixels. This allows the network to compute the similarity scores for 25×25 partially overlapping patches (stride 4) of size 32×32 in a single forward pass, filling up a region of 100×100 similarity scores (after upsampling). Doing this for every depth of the sweeping plane, we obtain a list of 256 multi-view similarity scores per pixel, and simply pick the depth with the highest score. To compute the $25 \times 25 \times 256$ similarity scores (covering an area of 100×100 original pixels) takes 70 ms, on an Intel i7 computer with Nvidia Titan X GPU using Caffe in Linux.

Practical matching software does not return raw depth estimates, but improves them with different simple steps. The plane-sweeping library [50] offers two such options: (i) interpolation of the discrete depth levels to sub-pixel accuracy, and (ii) box filtering to account for correlations between nearby depth values. We tested both options and found them to consistently improve the overall results, independent of the similarity measure. As we are primarily interested how well different similarity metrics perform under realistic conditions, we enable sub-pixel accuracy and box filtering in all following experiments.

5.3.1 *Evaluation on the DTU dataset*

The DTU Robot Image Data (DTU) contains more than 80 different indoor table-top scenes. Each DTU scene was recorded from predetermined, known camera poses with the help of a camera mounted on a robot arm, and a ground truth 3D point cloud was generated by mounting a high-accuracy active stereo sensor on the same robot arm. Images from 49 scenes are already used as our training data. Beyond those, also scenes that share one or more objects with any of the training scenes are excluded from testing, to avoid potential biases. For DTU, we set the depth range to $[0.45 \ldots 1]$ m.

QUALITATIVE RESULTS. The four example objects we use in the following are labeled BIRD, FLOWER, CAN, and BUDDHA. The BIRD has a uniform color so the intensity differences are largely due to the shading effects. The FLOWER is a difficult object due to the thin structure and the small leaves. Underneath the flowerpot, there is a newspaper with strong texture. The CAN has a metallic surface with homogeneous texture, while the background is rather strongly textured. The BUDDHA is made out of porcelain, with significant specular reflections.

The provided color images are converted to grayscale before processing, *c.f.* Fig. 34a-34d. The ground truth depth maps have been generated by back-projecting ground truth 3D point clouds and are sparse, *c.f.* Fig. 34e-34h, whereas the depth maps from multi-view stereo are dense. We thus evaluate only at the sparser ground truth depths. Also the difference plots in Fig. 34i-34t show only the pixels for which ground truth is available. Depth differences are color-coded on a logarithmic scale, where dark blue corresponds to zero difference and dark red denotes differences $> 20\,\text{mm}$.

For the BIRD the right wing, which is partly in shadow, is the most difficult part. For all three methods there are errors at the wing boundary, the largest errors are observed for LIFT. Note also errors on the left wing boundary present in ZNCC and LIFT but not in the proposed method. The most difficult object is the FLOWER. Here the ZNCC and our approach again outperform LIFT. All three methods struggle with the dark, homogeneous soil in the flowerpot. On the leaves, as far as present in the ground truth, our method has the smallest error, especially near the boundaries. For the CAN object, the homogeneous metal surface challenges the LIFT approach, whereas the two others can resolve it correctly. For the BUDDHA the most difficult parts for depth estimation are the small regions with specular reflection. These can be seen in Figs. 30, 34d. In those regions ZNCC and LIFT have large errors, while our direct multi-view approach copes a lot better (Fig. 34l). Overall, the examples suggest that our proposed similarity is more general: the classic ZNCC method often works well, but struggles near object boundaries. LIFT seems to have problems with certain surfaces materials like the metal of the CAN, and both competitors struggle with specularities (which, incidentally, are a main problem of stereo reconstruction, especially in industrial applications).

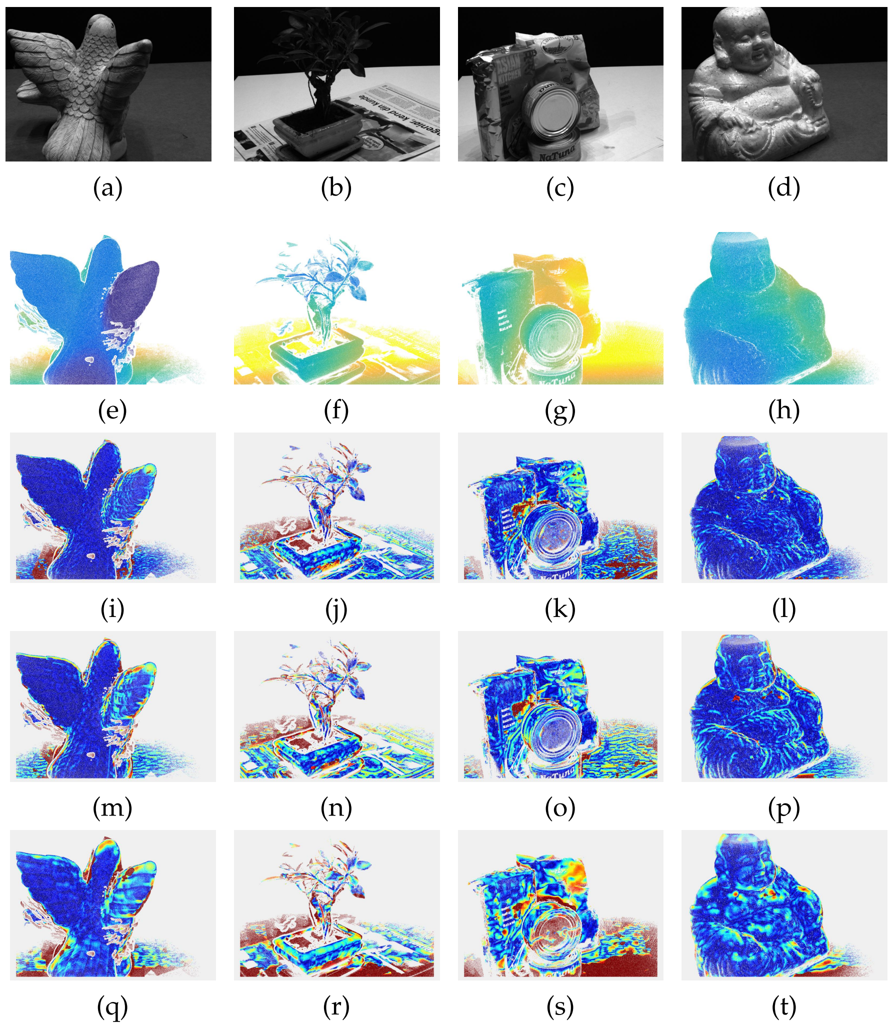


Figure 34: Plane sweeping using three different patch similarity measures. Proposed learned multi-view similarity *vs.* pairwise ZNCC and pairwise LIFT. Reference images (a-d). Ground truth (e-h). Absolute deviations from ground truth for the proposed method (i-k), ZNCC (m-p) and LIFT (q-t). Blue: no difference, red: high difference.

Similarity	#views	Accuracy		Completeness	
		Mean	Median	Mean	Median
SAD	4×2	1.868	0.617	2.534	1.148
ZNCC	4×2	**1.237**	0.577	2.260	1.025
OUR stereo	4×2	1.339	**0.453**	2.964	1.391
OUR 5-view	5	1.356	0.472	**2.126**	**0.868**

Table 4: Quantitative results for 20 objects from the DTU datasets. Four similarity measures are compared: sum of absolute differences, zero-mean normalized cross correlation, proposed similarity over 2 views, and proposed similarity over 5 views.

QUANTITATIVE RESULTS. The authors of the DTU dataset provide an evaluation framework in which, per object, the groundtruth 3D point cloud is compared to the one constructed from the depth estimates. For the evaluation we thus convert the estimated depth maps to 3D point clouds by lifting them with the known camera calibration. Note that we do not use depthmap integration across multiple viewpoints. *Accuracy* is defined as the average (truncated) distance from a reconstructed point to the nearest ground truth point. *Completeness* is defined inversely, as average distance from ground truth points to the nearest model points, *i.e.*, lower values are also better.

We start with a comparison to standard similarity measures for 20 scenes in Table 4. While the mean values describe the overall performance, median values are less sensitive to outliers. Note that, also for the mean, the per-point distances of outliers are truncated at 20 mm. To support the claim that direct multi-view similarity is preferable to consensus over 2-view scores, we also run our method in 2-view stereo mode and feed the scores to the same consensus mechanism used for SAD and ZNCC. Directly computing similarity over 5 input views delivers significantly better completeness at similar accuracy as the best competitors. We did not run the full experiment with LIFT, due to the excessive runtime of pixelwise CNN prediction without a fully convolutional architecture.

In Table 5 we compare the accuracy and completeness of all tested methods for the four example scenes. Differences are relatively small between SAD and ZNCC, probably due to the controlled lighting. The results for SIFT and LIFT are also quite

Similarity (5 views)	Accuracy Mean	Accuracy Median	Completeness Mean	Completeness Median
		BIRD		
SAD	2.452	0.380	**4.035**	1.105
ZNCC	**1.375**	0.365	4.253	1.332
SIFT	1.594	0.415	5.269	1.845
LIFT	1.844	0.562	4.387	1.410
OUR concat	1.605	0.305	4.358	1.133
OUR	1.881	**0.271**	4.167	**1.044**
		FLOWER		
SAD	2.537	1.143	2.768	1.407
ZNCC	2.018	1.106	2.920	1.467
SIFT	2.795	1.183	4.747	2.480
LIFT	3.049	1.420	4.224	2.358
OUR concat	2.033	0.843	**2.609**	1.267
OUR	**1.973**	**0.771**	**2.609**	**1.208**
		CAN		
SAD	1.824	0.664	2.283	1.156
ZNCC	1.187	0.628	2.092	1.098
SIFT	1.769	0.874	3.067	1.726
LIFT	2.411	1.207	3.003	1.823
OUR concat	**1.082**	**0.477**	**1.896**	**0.833**
OUR	1.123	0.478	1.982	0.874
		BUDDHA		
SAD	0.849	0.250	1.119	0.561
ZNCC	0.688	0.299	1.208	0.656
SIFT	0.696	0.263	1.347	0.618
LIFT	0.688	0.299	1.208	0.656
OUR concat	0.682	0.231	**1.017**	**0.473**
OUR	**0.637**	**0.206**	1.057	0.475

Table 5: Quantitative results for BIRD, FLOWER, CAN, and BUDDHA objects. Six similarity measures are compared: sum of absolute differences, zero mean normal cross correlation, SIFT, LIFT, proposed multi-view similarity using concatenation, and proposed multi-view similarity using averaging.

similar, except for the CAN object where SIFT clearly outperforms its learned counterpart.

The proposed method achieves the best median accuracy and median completeness in all the scenes, and the best mean accuracy and completeness for three of them. Note that there is virtually no difference between averaging and concatenation. There seems to be no performance penalty for averaging, while at the same time one gains the flexibility to use a variable number of input views. On the BIRD, our method ranks third in accuracy and second in completeness. There are relatively big differences between median and mean errors, apparently all measures show quite good performance on the rather diffuse surface, whereas outliers due to ambiguous texture inflate the mean values.

Overall, the proposed multi-view patch similarity exhibits the quantitatively and qualitatively best performance. In particular, the experiments support our claim that learning end-to-end multi-view similarity is preferable to comparing learned per-patch descriptors with conventional Euclidean distance, and to a consensus over learned 2-view similarities.

5.3.2 *Invariance to the number of input patches.*

We go on to demonstrate that the network architecture, in which branches are averaged and convolutional weights are shared, can be applied to different numbers of input views without retraining. We run experiments with either three or nine views, respectively patches. Both give reasonable depth estimates (Fig. 35). As expected, the results with only three views are a bit worse. However, using nine patches instead of five further improves performance – although the similarity network has only been trained with five. We speculate that information how to correctly weight pixels at discontinuities, occlusions etc. passes down the individual branches during training, so that the parallel Siamese streams can also use it for additional branches. On the other hand, averaging itself may have a stronger denoising effect with more branches. Further research is needed to clarify the underlying mechanisms. Whatever the reason, in our view flexibility with respect to the number of input views is an attractive and important feature for real-world applications of multi-view stereo.

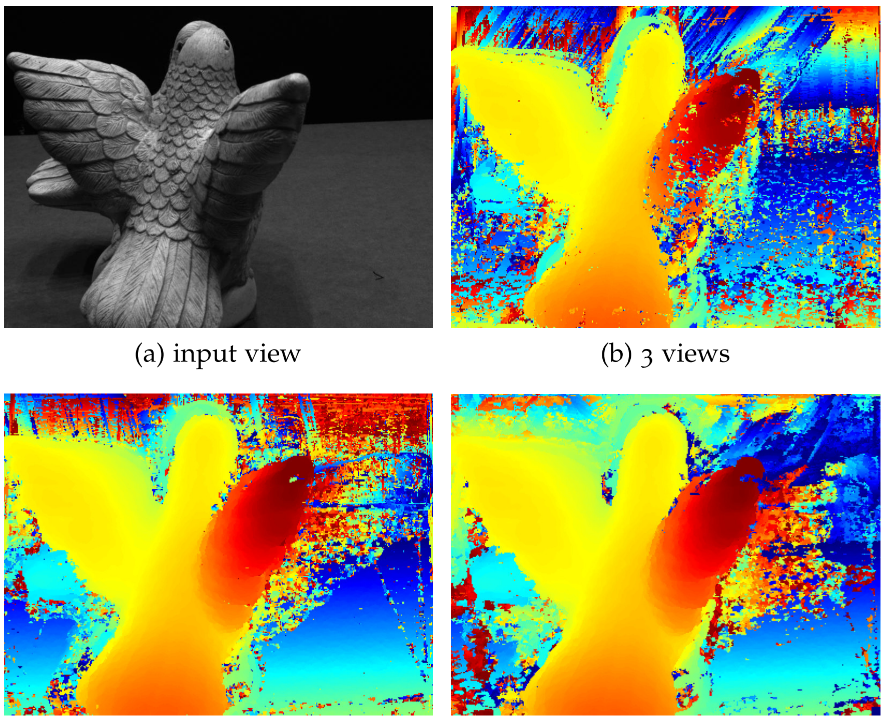

(a) input view
(b) 3 views
(c) 5 views
(d) 9 views

Figure 35: Matching different numbers of views with the similarity network can be done without retraining. Result displayed without sub-pixel refinement and box filtering to accentuate differences.

5.3.3 *Evaluation on the Fountain dataset*

The images recorded with the robot for the DTU dataset are all taken in an indoor laboratory environment, and there might be the risk that the specific lighting and camera characteristics of the dataset are captured in our trained network. Therefore, we apply the learned multi-view similarity measure also to the well-known Fountain dataset [113], without retraining it.

For the experiment we select five neighboring images and set the depth range to $[5 \ldots 10]$ m. The depth maps in Fig. 36 show the qualitative result for three different methods. Our method works at least as well as ZNCC and SIFT, which are generic and not learned from data while the network does not seem to overfit to the DTU setting. The small support of the input patch (32×32 on 2MPix images) makes it unlikely is to learn the scene specific image structure and overfit to that.

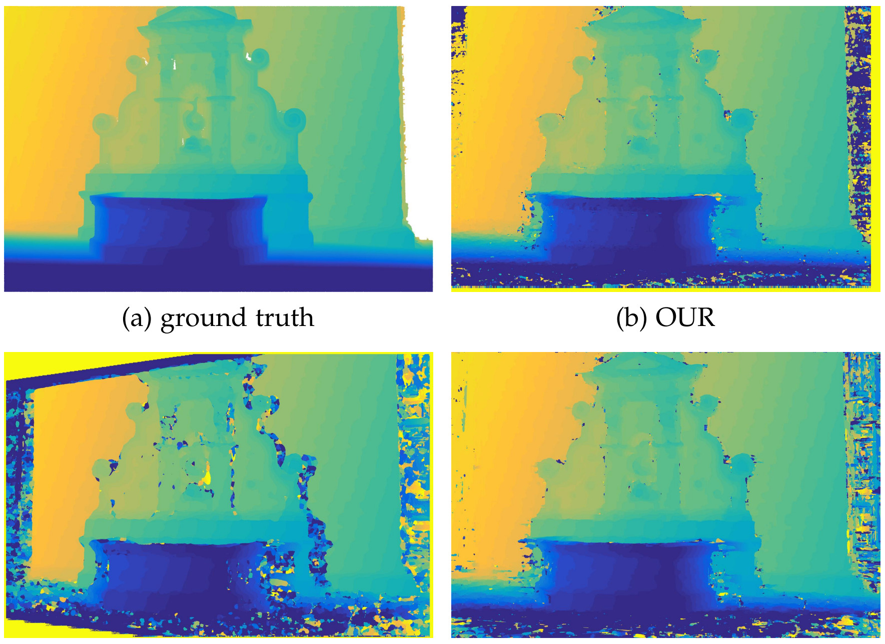

(a) ground truth (b) OUR (c) SIFT (d) ZNCC

Figure 36: The learned similarity generalizes to a different test environment, seemingly as well as the competing descriptors.

5.4 CONCLUSION

We have proposed to directly learn a similarity / matching score over a set of *multiple* patches with a discriminative learning engine, rather than heuristically assemble it from pairwise comparisons. An n-way Siamese convolutional network, which applies the same, learned transformation to all input patches and combines the results, was identified as a suitable state-of-the-art learning engine.

From a high-level perspective, machine learning for 3D vision started with very small steps like learning the distance metric between given descriptors, or learning descriptor extraction for pairwise matching. In our work, we have used the learned similarity score for multi-view stereo matching, but we believe that variants of it could also be beneficial for other applications where similarity, respectively distance, between more than two images must be assessed.

6 NORMAL PREDICTION FOR IMPROVED MULTI-VIEW RECONSTRUCTION

This chapter is based on Galliani and Schindler [42].

The reconstruction of 3D surfaces from images is a central problem of computer vision. The dominant approach is multi-view stereo (MVS): densely match image points in multiple views with known camera poses, then triangulate the corresponding rays to 3D points. MVS algorithms have greatly improved over the past decades and nowadays deliver high-quality point clouds, respectively surfaces derived from those point clouds [66, 106]. Yet MVS, being based on point correspondences between different images, only works in areas with sufficient texture. If no correspondence can be established, the methods fails. Most commonly this happens in surface regions with uniform albedo and on specular highlights, where matching is ambiguous due to a lack of high-frequency brightness/color variations. A further recurrent problem are occlusions, where many viewing rays are blocked and do not reach the surface point. In such regions the only options are to either not reconstruct 3D points, leaving holes in the surface; or to interpolate, which can lead to inaccurate or even totally wrong results.

We propose to fill in the missing regions with the help of shading information. As introduced in Chapter 1, complementary to MVS, shape reconstruction from shading requires only a single view, and works best for uniform albedo. Yet, recovering 3D surface normals from shading has proved remarkably difficult in practice, mainly because a number of important influence factors are hard to model. In real data the illumination can be quite complex and the illumination direction (s) are not exactly known. Most importantly, the reflectance properties (the bi-directional reflectance distribution function, or BRDF) of the surfaces in the scene are usually unknown.

The starting point for the present chapter is that if one wants to reconstruct surface shape from shading, it might not be necessary to model the global illumination and the complete reflectance distribution. Rather, one only needs to cover the

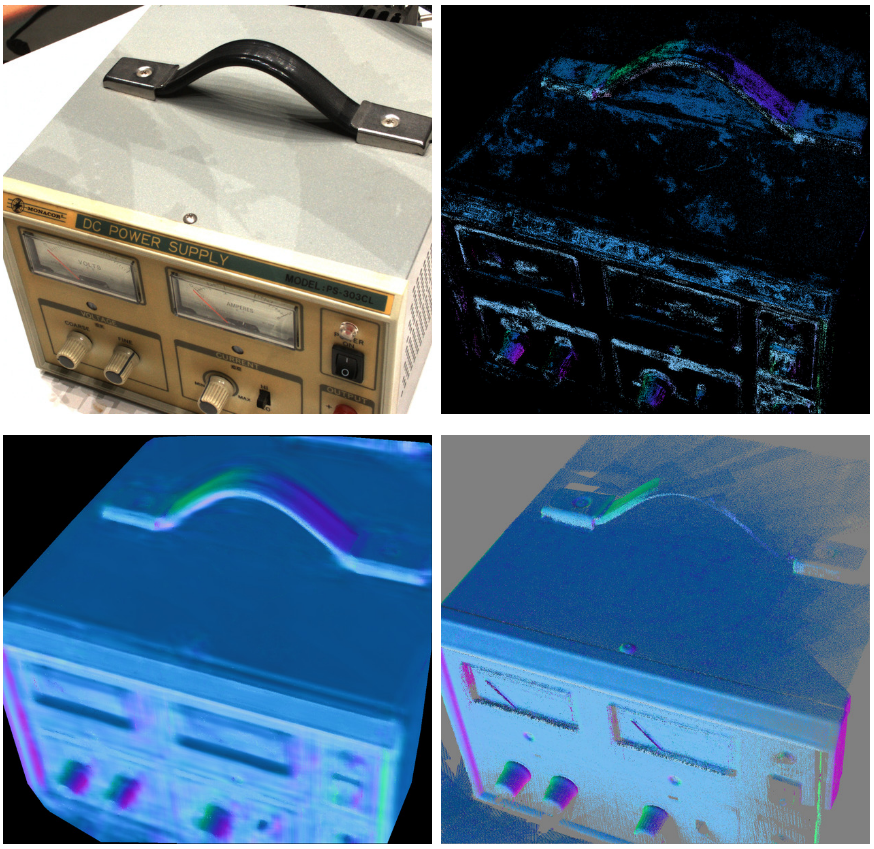


Figure 37: Illustration of our reconstruction method. Given an input image (**top left**) and an incomplete normal map from multi-view stereo (**top right**), we reconstruct the missing normals by CNN regression on the image (**bottom left**). The normal maps are then integrated to dense 3D models (not shown). **Bottom right**: Ground truth.

specific illumination, viewpoint and surface properties that are present in a given image. We exploit this by implicitly learning the view-specific shading patterns in a discriminative manner. Given that in most images there are pixels for which the surface normals are known (from 3D points reconstructed via multi-view stereo), we propose to learn a regression directly from raw RGB patches to surface normal directions, using a convolutional neural network (CNN).

In contrast to other recent work that predicts surface normals in a purely data-driven fashion [30, 84, 100] we do *not* aim for generality across different lighting and viewing conditions, and thus do not need a diverse training set that covers all possible conditions. Rather, we learn an individual, view-specific shading model per image, trained on reprojected 3D normals that we reconstruct from high-confidence MVS points. Such a model only needs to cover a subset of the BRDFs of (usually few) visible materials, under constant lighting, thus it can be expected to predict more accurately. *I.e.*, we argue that the image itself, together with an incomplete range/disparity image, contains sufficient information to predict surface orientation, without a globally valid shading model.

Our method is able to estimate surface normals with an accuracy similar to (sometimes even slightly better than) that of the training data. To complete the pipeline we integrate the dense normal field per image, together with the known 3D points from MVS, into a dense and hole-free depth map, and fuse the depthmaps from multiple views to obtain a more complete 3D model.

6.1 METHOD

We start with an overview of our complete surface reconstruction pipeline. As input data, we require multiple images of the same scene, with known camera poses. The first step is a MVS reconstruction with Gipuma, presented in Chapter 4. That method been shown to deliver state-of-the-art performance [41], and it returns point-wise normals in 3D scene space as a byproduct; but other algorithms that provide surface normals could be plugged in as well. The next step is to predict normals for pixels where multi-view stereo failed to compute a reliable depth. This is done separately for every viewpoint. From the points reconstructed successfully by MVS, we train a convolution neural network (CNN) to perform regression from raw image patches to surface normals. With the network, we densely predict all missing normals (Sec. 6.1.2). The dense vector fields are turned into a 3D surface model by first integrating them to depth maps with masked Poisson reconstruction (Sec. 6.1.5) and then fusing the depth and normal maps from multiple views.Optionally, the final 3D point cloud can also be turned into a triangulated surface model with screened Poisson reconstruction [76].

6.1.1 *Generation of normals for training*

The first stage of our method is a standard MVS reconstruction to obtain an initial (incomplete) cloud of reliable 3D object points. Among the many available algorithms we choose Gipuma, see Chapter 4. We pick this method for two reasons.

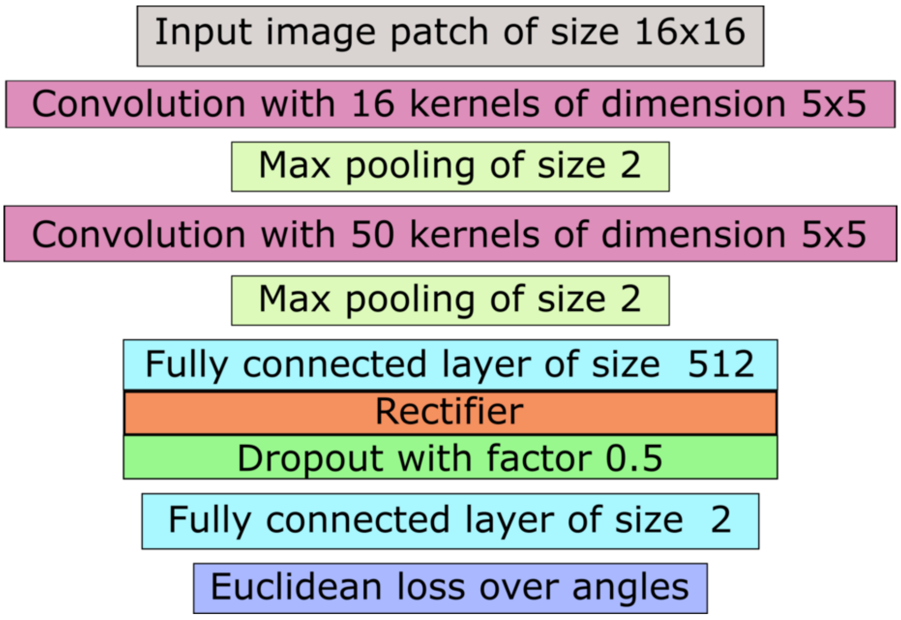

Input image patch of size 16x16
Convolution with 16 kernels of dimension 5x5
Max pooling of size 2
Convolution with 50 kernels of dimension 5x5
Max pooling of size 2
Fully connected layer of size 512
Rectifier
Dropout with factor 0.5
Fully connected layer of size 2
Euclidean loss over angles

Figure 38: CNN architecture for regression from image patches to surface normals.

On the one hand, it computes and outputs, by construction, not only 3D points but also explicit surface normals at those points. Since our further processing needs those normals, PatchMatch is a natural fit. On the other hand, the depthmap fusion relies on a consensus mechanism that checks both the consistency of the depth values *and* of the normal directions across several views. As a result, points with unreliable normals are discarded during fusion, which is important for our purposes, since those normals will later serve as training data.

6.1.2 *Normal prediction*

The philosophy of our second, shading-based stage is to learn the relation between surface normals and the appearance of the corresponding surface patches. That relation can then be used to predict surface normals at locations where no MVS points could be reconstructed. As explained, we prefer to initially do this on a per-image basis and again fuse the results afterwards. Estimating the normals individually in each image simplifies the learning problem, because in a single exposure the lighting conditions are constant; and it also simplifies the implementation, because one can work on the pixel grid rather than discretise the 3D scene surfaces.

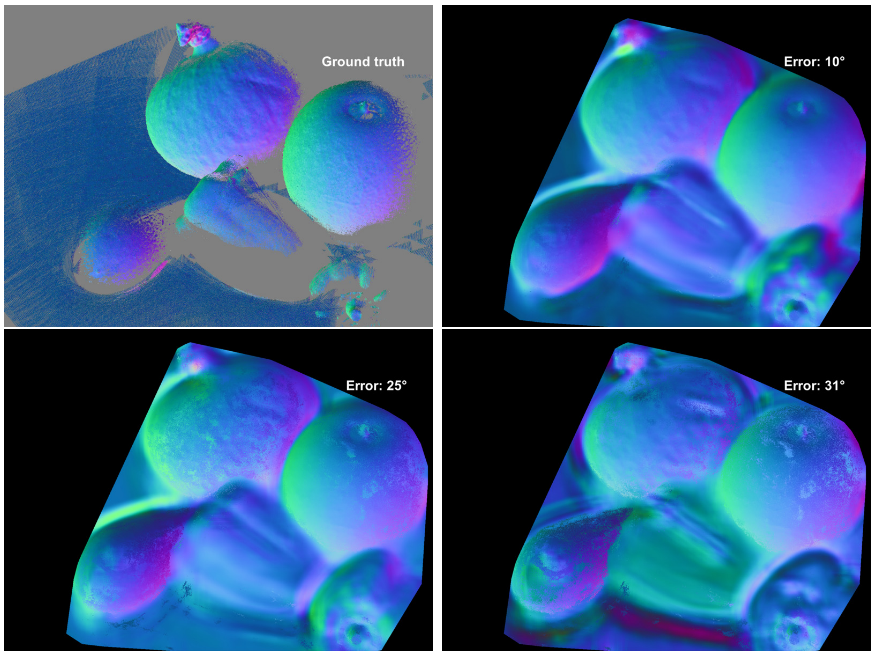


Figure 39: Comparison of different strategies for normal prediction. A model trained for a specific image (**top right**) works better than one trained on multiple views of the same scene (**bottom left**) or a generic model trained for a whole database (**bottom right**).

We also experimented with a single model for all views, effectively trying to learn the shading variation for a given object, under any viewpoint. This did not work well, see Fig. 39. We see two possible reasons. On the one hand, the learning problem obviously gets a lot more complicated and ambiguous if one has to cover two additional degrees of freedom (for the viewing direction) in the BRDF. On the other hand, it may well be that for certain materials the CNN also learns context and texture cues that are not independent of the viewpoint.

6.1.3 *Training data.*

As part of the MVS reconstruction, we have a surface normal map for each individual view, which holds, at every pixel, either a normal vector in camera-centric coordinates or a flag that no normal could be reconstructed. In order to ensure clean training data for CNN training, we filter those surface normal maps with the same fusion scheme used by Gipuma, see Section 4.3.3. Our goal at this point is high precision even at the cost of a bit lower recall, *i.e.* we try to ensure that only correct

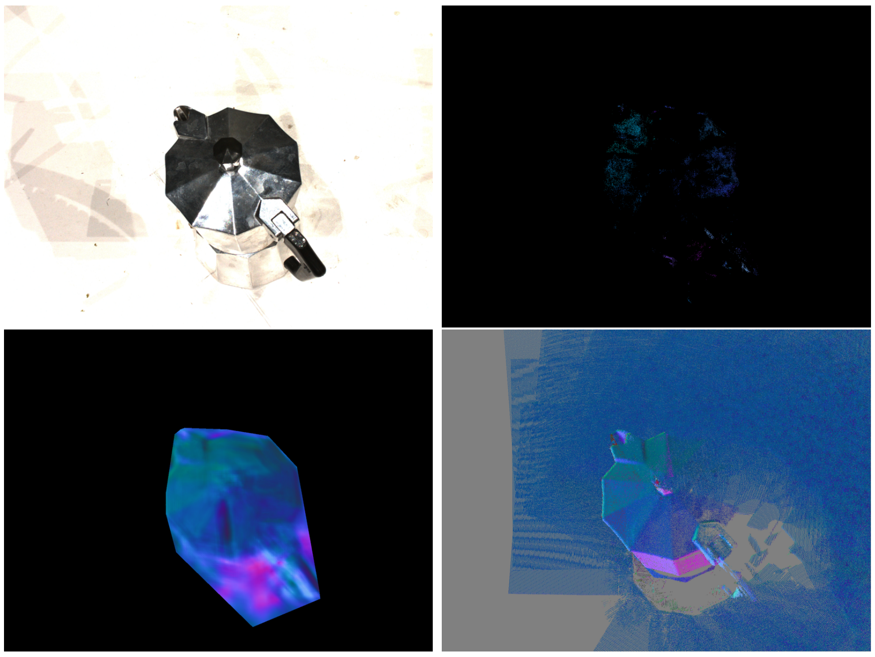


Figure 40: Normal prediction for a particularly difficult scene (DTU object 77). Even with few training points of a highly specular object the regressor is able to recover reasonable normals in many regions. **Top left**: input image. **Top right**: training normals from MVS. **Bottom left**: predicted normals. **Bottom right**: ground truth.

and accurate normals are retained. As a first filter, we remove all normals that did not survive the multi-view fusion (meaning that they did not fit the consensus). For those pixels which did contribute to the reconstruction of a 3D normal vector, we reproject the 3D vector and replace the original entry. This can be expected to improve the accuracy of the valid normals, because the inliers to the consensus voting are averaged during fusion to suppress noise. On very slanted surfaces it can, in rare cases, happen that the averaged normal points away from the camera; such normals are discarded. The final normal maps is sparse: have entries only where the original matcher found a depth, and thus also a normal, and that depth and normal were confirmed as correct and visible by a consensus over multiple viewpoints.

6.1.4 *Unsupervised Normal Regression with CNN.*

Having found a set of reliable normal vectors to serve as training data, we learn, separately for each view, a different convolutional neural network (CNN) to predict unknown surface normals. Note that no manually labeled training data is required, the regressor is trained only from automatically reconstructed MVS points. As input, the network takes 16×16 pixel RGB patches, downsampled from 64×64 pixel patches of the original image. As output, it returns the estimated normal vector at the center pixel of the patch, parameterized by two polar angles θ and ϕ (a.k.a. *azimuth* and *elevation*, or *yaw* and *tilt*). The patch size has been determined empirically: much smaller patches do not work as well, it seems that they do not capture sufficient shading information; larger patches slow down the computation without improving performance.

As loss function, we directly minimize the minimal planar angle $\alpha = \langle \mathbf{n}_{\text{true}}, \mathbf{n}_{\text{pred}} \rangle$ between the true normal and the predicted one. Our architecture follows the *LeNet* framework [87]: a convolution layer with 16 kernels of window size 5×5, followed by max-pooling over 2×2 blocks; a second convolutional layer with 50 kernels of size 5×5, again followed by 2×2 max-pooling; a fully connected layer of 512 neurons, with ReLU rectification and 50% drop-out; and a final fully connected layer with 2 output neurons for the angles θ and ϕ; See Fig. 38. The network is implemented in the Caffe framework [69], and trained with stochastic gradient descent, with a fixed momentum of 0.9 and a learning rate of 0.001. Training and prediction take $\approx$30 min per view, on a single PC.

The following reason motivated us to use a CNN: the perhaps biggest strength of CNNs and related deep learning methods, and the main reason for their phenomenal success in computer vision, has been the capability to learn good image representations from raw RGB data. We feel that this end-to-end learning, which relieves us from finding a suitable feature set, is particularly useful for our problem. Compared to well-researched vision tasks like pedestrian detection or semantic segmentation, little is known about the right choice of features for discriminative normal estimation, hence finding good features might end up being a lengthy trial-and-error process. We also point out that in the recent work of [133] CNNs were shown to perform well (and superior to regression forests) for a related regression task from visual appearance to a spatial direction, namely image-based gaze estimation.

After training the regressor, we apply it to the same image, and estimate normal vectors densely for all pixels except for the training data, which already possess normals from MVS. To

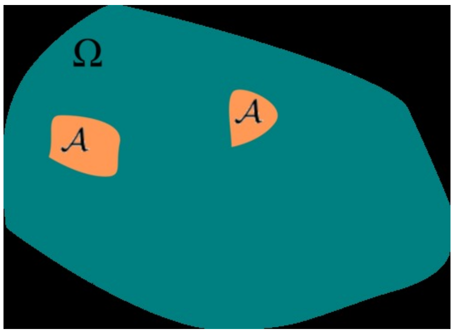


Figure 41: Visualization of the integration domain Ω of our surface normals. Points in $\mathcal{A}$ (corresponding to our self generated training data) are used as Dirichlet boundary conditions.

avoid excessive extrapolation, we only predict inside the convex hull of the training pixels.

6.1.5 *Surface normal integration*

The previous step yields a dense map of normals for every viewpoint. Since our goal is 3D surface reconstruction, we need to convert that normal map into a dense depth map, which however is constrained to pass through the known depth values from MVS. We do this with a masked version of the 2D Poisson equation. Formally, we face an interpolation problem: interpolate depth values at all points not reconstructed by MVS, such that they best agree with the predicted surface normals. To distinguish points with known MVS depth from those without one, we define two separate depth functions: f_{mvs} for MVS points is known, whereas f is the unknown to be recovered. The domain of f_{mvs} is only the discrete set $\mathcal{A}$ of MVS points, and f is defined everywhere in the image plane Ω excepts at the points $\mathcal{A}$. The vector field g consists of the gradients of both functions,

$$\forall x \in \Omega : g(x) = \begin{cases} \nabla f_{mvs}, & \text{if } x \in \mathcal{A} \\ \nabla f, & else \end{cases} \tag{27}$$

Our task is to find an interpolant f over $\Omega \backslash \mathcal{A}$ that minimizes the squared error

$$\min_f \iint_{\Omega \backslash \mathcal{A}} ||\nabla f - g||^2 \quad . \tag{28}$$

This leads to the Poisson equation

$$\Delta f = \operatorname{div} g \; , \tag{29}$$

with $\text{div}(\cdot)$ the divergence operator and $\Delta(\cdot)$ the Laplacian. The MVS points in $\mathcal{A}$ each contribute a Dirichlet boundary condition, ensuring that the depth map will pass through f_{mvs}. Together with standard von Neumann boundary conditions at the image border the equation has a unique solution. Since the domain is irregular, one must fall back to an iterative solver for (29), we use the Gauss-Seidel scheme with successive over-relaxation [95, 126].

6.1.6 *Depth map fusion*

Our setting is that we have multiple overlapping views of a scene — otherwise we could not perform MVS reconstruction. Having recovered depth maps in all these views, the last step is to fuse them into a consistent 3D model as Gipuma 4.3.3.

In many locations the appearance-based normal prediction (and subsequent integration) yields comparable accuracy to multi-view stereo, which uses similar fusion criteria. Obviously, the fusion parameters $f_{\varepsilon}, f_{ang}, f_{con}$ provide a simple interface to tune accuracy *vs.* completeness of the reconstruction. With strict values, fewer but more reliable points survive (*e.g.*, for applications in industrial metrology). With more generous settings the completeness of the reconstruction increases, at the cost of lower accuracy (*e.g.*, for graphics and visualization purposes).

6.2 RESULTS

To validate our method, we use a subset of 14 objects from the extensive DTU multi-view stereo dataset [66]. The reason we're using only such a subset, even if the dataset provides more than 80 objects, is mainly due to the fact that the current CNN implementation for normal prediction is not tuned for speed. Training the network from scratch on each image takes roughly 30 minutes, which means that the runtime to complete a full object with 49 views is one day. Which such timing, processing the whole dataset takes too much time. The dataset is, to our knowledge, the only large MVS testbed that is publicly available. It features a variety of objects and materials, and provides complete coverage with 49 images per scene. Ground truth of adequate density has been recorded with a structured light scanner. The large selection of shapes and materials, ranging from simple diffuse surfaces to specular plastic and metal objects, is well-

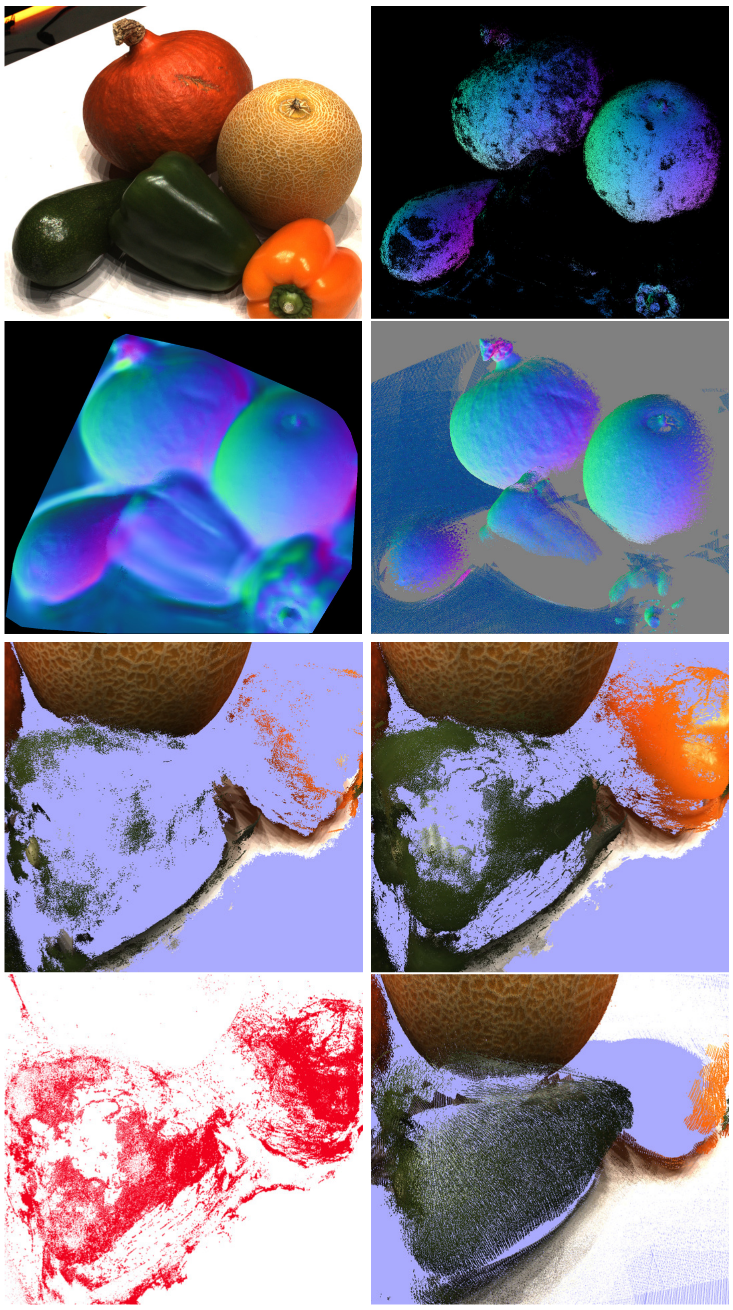


Figure 42: **Top**: Normal prediction: input image, training points, reconstruction, ground truth. **Bottom**: Reconstruction closeup of the peppers after normal integration and depth fusion. **From left to right**: Input. Our result. Difference. Ground truth. Our reconstruction with normal prediction is able to complete parts missed both by MVS and by the structured light scanner used for the ground truth.

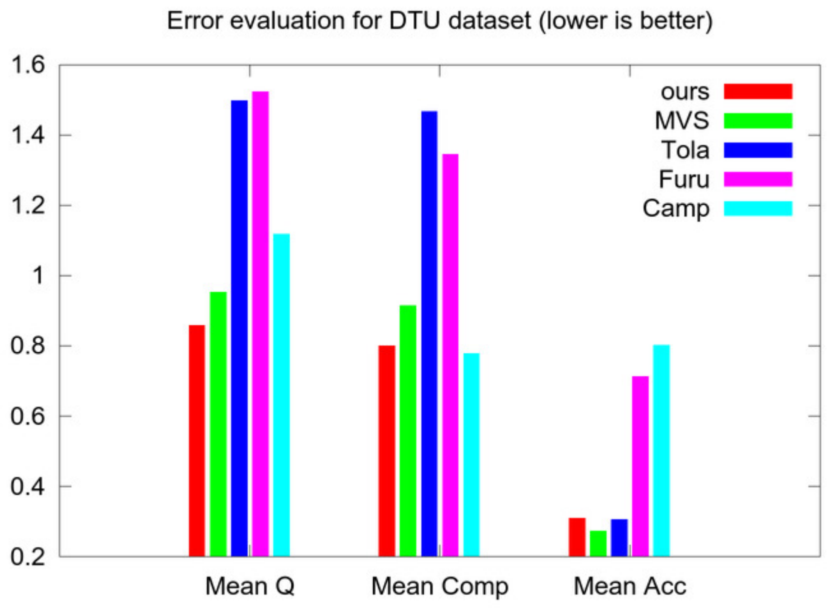


Figure 43: Quantitative comparison with our initialization and other pure MVS methods [15, 37, 116]. Lower values are better.

suited to test our normal prediction under realistic conditions. Importantly, the dataset is difficult enough to challenge multi-view stereo: even state-of-the-art methods, including the one that we use for MVS [41], do not manage to reconstruct large parts of some scenes. And it is also complex enough to defy shading methods based on simple Lambertian reflectance, with materials of different color, texture and specularity. We use the variant of the data recorded under standard (relatively diffuse) lighting conditions, because this is the only one for which multiple recent works have reported results. In principle it would be possible (and potentially beneficial) for our method to include images with various lighting conditions, in the hope that a certain illumination is better suited for certain parts of the scene than others.

We first evaluate the surface normal prediction separately, and then present an end-to-end comparison with the final 3D reconstructions.

6.2.1 *Normal prediction*

To quantify the accuracy of the normals predicted by the CNN, we measure the angular error w.r.t. to ground truth normal derived from the reference point cloud. As a first step, we compare the error on the "test" normals predicted by the regression to the one for the "training" normals estimated by MVS. Ideally, these errors should be similar, meaning that the appearance-based predictions would be as good as the multi-view estimates. We observe, not surprisingly, that the relation depends a lot on the difficulty of the scene. For simple, piecewise planar objects with little reflection, the predicted normals are even slightly more accurate than the training normals. Presumably, this is so because the learning problem is easy, and the "averaging" over training samples from the same surface reduces noise. *E.g.*, although the MVS result is rather sparse in Fig. 37, it is sufficient to obtain sensible predictions for most of the object. The corresponding mean and median errors are 13° and 9°, respectively, for the MVS points; and 12°, respectively 6° for the CNN prediction.

On the contrary, specular materials and complicated surface geometry, *e.g.* sharp creases, make the prediction more difficult. The most difficult object in the DTU database is the coffee-maker in Fig. 40. Even in that case, the appearance-based regression surprisingly gives reasonable predictions in many parts. However, the mean and median errors rise from 13° and 10° at the MVS points to 17° and 12° for the predicted ones.

In Fig. 42 the shiny surface of the peppers poses a serious problem for both MVS and for the structured light scanner that acquired the ground truth. Our method is able to predict normals in these areas. While the mean and median errors are significantly higher than at the MVS points (17° and 10°, compared to 8° and 6°), they are still good enough to reconstruct an important part of the missing surfaces to a depth accuracy of 0.3 pixels in disparity. Note that especially on the yellow pepper our reconstruction is also a lot more complete than the ground truth from the structured light scanner, which fails on very specular surfaces, too.

Over all 14 objects, the mean angular error is 11° for the training normals from MVS, and 18° for the predicted normals. The mean-of-median over all objects is 9° for the MVS normals and 16° for the predicted ones. The mean is consistently only a bit above the median, which indicates a relatively even error distribution not contaminated by many large outliers.

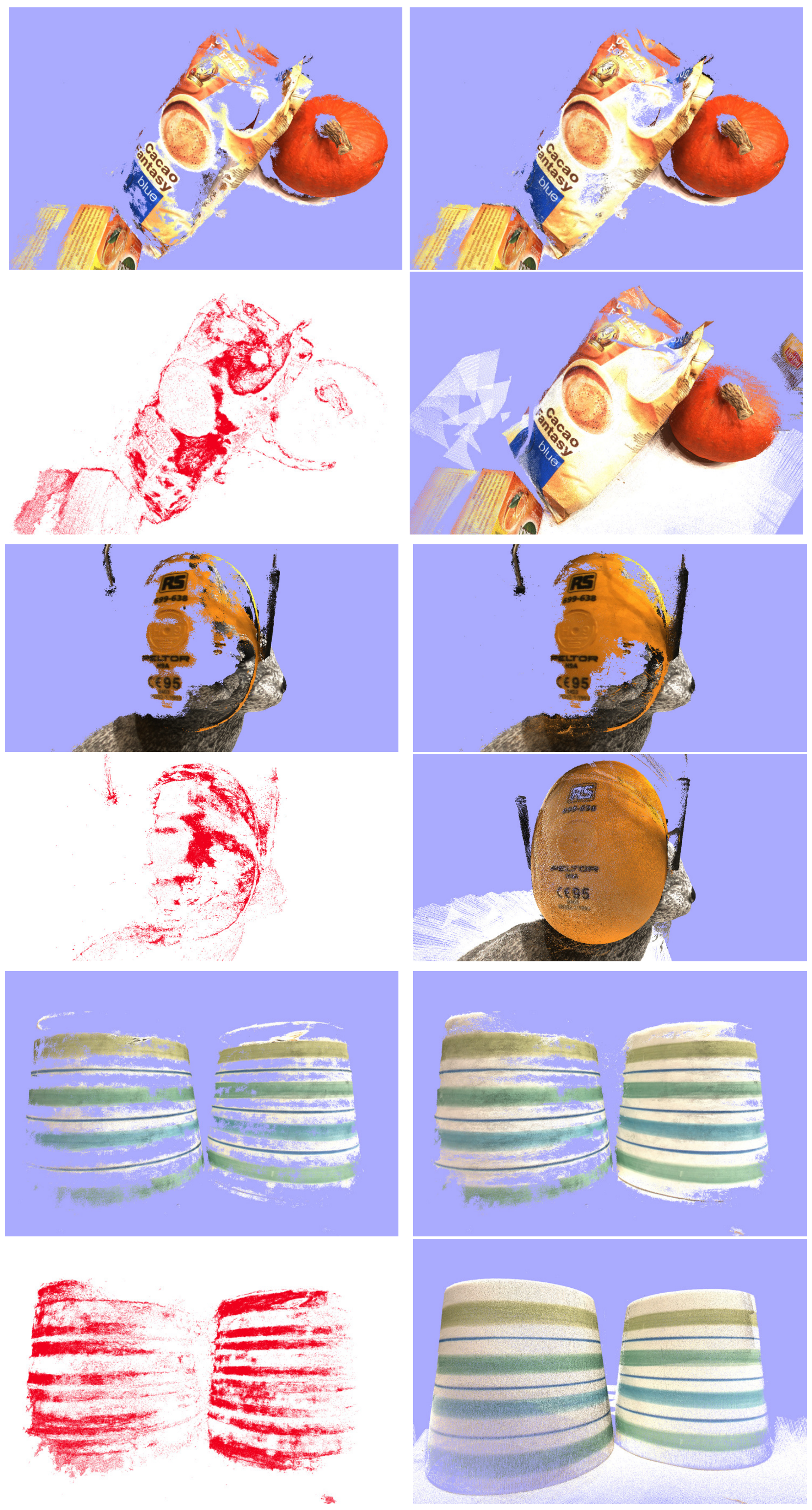


Figure 44: Reconstruction improvements of our method on three different objects. **Top**: Challenging object with multiple colors and with specularities. **Middle**: Object with homogeneous colour. **Bottom**: Vase with over-exposed and homogeneous white areas. For each object: **Top-Left**: MVS reconstruction. **Top-Right**: Improvement. **Bottom-Left**: Difference. **Bottom Right**: Ground Truth.

6.2.2 *Improved multi-view reconstruction*

Our overall goal is a better reconstruction of 3D point clouds, respectively surfaces. We thus go on to quantify the accuracy and completeness of the resulting 3D models. As baselines, we use the initial MVS reconstruction without normal prediction, as well as three further MVS methods for which results on DTU are available.

To ensure a fair comparison to pure MVS, we set the same fusion parameters (Sec. 6.1.6) both for fusing MVS depthmaps and for fusing depthmaps after normal integration. *I.e.*, points found with shading are added to the MVS reconstruction only if they fulfill the same strict reliability criteria.

Fig. 43 shows quantitative results averaged over all reconstructed objects. The proposed prediction and integration of the normals improves the mean completeness of the MVS initialization by 14 %, at the cost of a negligible increase in accuracy (accuracy is measured only at the reconstructed points, hence an improvement is virtually impossible when adding additional points to an existing, sparse reconstruction). Moreover, our results compare favorably w.r.t. other methods. In terms of accuracy, we are on par with the best result by [116], but with much higher completeness ($\approx$ 83% better). In terms of completeness, we are second best, narrowly behind [15], which however has a lot lower accuracy (61 % higher error).

Any multi-view reconstruction method can trade off accuracy against completeness. Tuning for high accuracy means strict consistency checks that reject many points and drive down completeness. Conversely, tuning for completeness means accepting more points, even if they have higher error. We thus also compute the overall *quality* of a reconstruction, defined as the geometric mean of accuracy and completeness $Q = \sqrt{acc^2 + prec^2}$, similar in spirit to the F1-score. On that measure our method clearly performs best, leading by 11% over the MVS initialization, and 30% over the next best method.

We end with some qualitative examples to illustrate where the proposed normal prediction can help. Overall, the experiments confirm the intuition that the prediction will fill in holes in homogeneous areas, where MVS struggles. A prime example is the bunny in Fig. 44. MVS does alright on the fur, but can only reconstruct the textured part of the earmuffs. Still, there are enough points on the earmuffs to learn the normal prediction, hence a good part of the untextured orange plastic gets filled in. The white stripes on the vases in Fig. 44, also challenge MVS. This is an example for a material with a non-lambertian shading component, nevertheless the prediction fills in a large part of the missing surface. The plastic packaging in Fig. 44 is even more challenging, with multiple colors as well as specularity. Note how the regression predicts adequate normals for different parts including the blue area at the bottom, the yellow/white area in the center, and even the shadow area on the red object behind the bag.

6.3 CONCLUSION

We have described a method to densify multi-view stereo reconstructions with the help of shading cues. Like some other recent methods, we sidestep analytic shading models. Instead, we view surface normal estimation as a discriminative unsupervised regression problem and train a CNN to predict normal vectors from raw image patches. The basic insight is that the regression problem can be greatly simplified if one sacrifices generality and learns an individual predictor for the fixed illumination, viewpoint and scene properties of each specific image. The prediction is embedded in a conventional multi-view reconstruction pipeline: point successfully reconstructed via Gipuma form the training set for normal estimation, and the resulting dense normal maps are integrated to depth maps to improve the 3D model.

A main message of our approach is that even a rather small number of training examples are enough to learn normal estimation from raw intensities, if the problem is tightly constrained. For a particular view of a particular scene, it is indeed possible to infer shape, with an accuracy similar to the one of MVS.

So far our method only fills in missing depth measurements. The original MVS points are not modified, and depth map fusion is done in a separate step. In future work we plan to investigate an early fusion, which directly reconstructs the 3D surface from multiple normal maps and sparse depth measurements. Additionally, in the future we plane also to make a more complete comparison of our view-specific normal prediction w.r.t. recent state-of-the-art generic models for normal prediction.

7 CONCLUSION AND OUTLOOK

The goal of this thesis has been set as to investigate the possible convergence of two complementary 3D reconstruction methods: Shape from Shading and Multi-View Stereo Reconstruction. In this chapter, we first encapsulate our major contributions. Then we'll examine the limitations and possible future direction of this work.

7.1 SUMMARY OF CONTRIBUTIONS

In this thesis, we have considered the problem of reconstructing the surface of a static object or scene by combining geometric based stereo reconstruction and shading based stereo reconstruction. We have investigated and contributed to the major steps that occur within popular 3D reconstruction pipelines and presented a shading based refinement method.

In Chapter 4 we propose Gipuma, a new massively parallel method for high-quality multiview matching. The starting point was Patchmatch stereo [12]: starting from randomly generated 3D planes in scene space, the best-fitting planes are iteratively propagated and refined to obtain a 3D depth and normal field per view, such that a robust photo-consistency measure over all images is maximized. We introduce two main novelties.

First of all a new formulation for Patchmatch Stereo in scene space. Working on the disparity space prohibits to directly compare similarity across multiple views, since image rectification involves by definition two cameras. Instead, we aggregate image similarity across multiple views to obtain a robust correspondence.

Moreover, we modify the inherently sequential algorithm of Patchmatch and propose a *red-black* scheme for the diffusion of putative depth values. Our method is massively parallel and delivers dense multiview correspondence over 10 1.9-Megapixel images in 1.5 seconds, on a consumer-grade GPU. We fit patches on the scene space and use a slanted support window to avoid fronto-parallel bias. Our method is local and parallel so that computation is linear w.r.t. image size and inversely proportional to the number of threads. Furthermore, it has low memory footprint (four values per pixel, independent of the depth range: 2 for unit normal, 1 for depth and 1 for cost). Therefore, it scales exceptionally well, can handle multiple large images at high depth resolution or can be used in mobile applications due to the low memory consumption. Experiments on the DTU and Middlebury multiview datasets as well as oblique aerial images show that our method achieves very competitive results with high accuracy and completeness, across a range of different scenarios.

The similarity measure described in Chapter 4 followed the general trend of rather successful multi-view stereo methods (see benchmark results such as [67, 104, 106, 113]). It means it heuristically forms a consensus over pair-wise similarities, or a subset of them, from all other stereo partners to a "reference image" in which the depthmap is computed. To directly measure multi-view similarity, in Chapter 5 we propose to learn a matching function which maps multiple image patches to a scalar similarity score. To that end, we introduce a multi-stream "Siamese" convolutional neural network architecture that takes as input more than two images and outputs a scalar similarity score. We train our network using image patches around projected 3D points measured from a structured light scanner [67]. It is in contrast with other (pairwise) learned based similarity networks such as LIFT [111, 124] which trains from 3D point extracted with a Structure from Motion pipeline, effectively learning the feature used to match the 3D points, usually SIFT [89].

The network has two important properties. First of all, thanks to shared convolution weights and to *mean pooling* of different network branches, our network becomes invariant to the number of input patches at test time. Thus, it allows training the network only once with a fixed number of input patches and predict with the number of patches at hand.

Secondly, when applied inside a multi-view stereo pipeline such as plane sweeping [23], it highlights the independence of the similarity measure w.r.t. the reference camera. That is observed in case of noise in front of the camera such as specularity, where traditional similarity measures fail, while we can obtain the correct reconstruction, see Fig. 30.

Starting from a 3D reconstruction obtained from Gipuma [41], presented in Chapter 4, we presented in Chapter 6 an unsupervised discriminative shading model used to complete the result. Reliable surface normals obtained directly from Gipuma, serve as training data for an unsupervised — or self-supervised — convolutional neural network, which predicts continuous normal vectors from raw image patches.

The estimated dense normal maps, together with the sparse known depth values from MVS, are integrated with a Poisson equation to a dense depth map. The resulting noisy depth maps are in turn fused into a 3D model (in the same way depth maps of Gipuma are fused together, see 4.3.3) to aggregate evidence and remove spurious depths.

In contrast to other recent work that predicts surface normals in a purely data-driven fashion [30, 84, 100] we do not aim for generality across different lighting and viewing conditions, and thus do not need a diverse training set that covers all possible conditions. Rather, we learn an individual, view-specific shading model per image, trained on reprojected 3D normals that we reconstruct from high-confidence MVS points. In other words, the prediction is specifically tailored to the materials and lighting conditions of the particular scene, as well as to the correct camera viewpoint.

Experiments on the DTU dataset show that our method delivers 3D reconstructions with the same accuracy as MVS, but with significantly higher completeness.

7.2 OUTLOOK AND FUTURE WORKS

Even if we provide multiples advances and insights within this thesis, capturing accurate 3D models of real-world objects or scenes remains an unsolved task. We will discuss here some possibilities for future work.

In Chapter 4 we present Gipuma, a massively multi-view stereo method which provides highly accurate reconstruction with low runtime. Even if we showed to be accurate, it could be improved in many ways. The selection of views used to compare the similarity is rather primitive (it takes into consideration the angle between the patch normal and the camera view) and it is fixed for all the cameras. A better per-pixel algorithm could be used instead, taking as inspiration the optimization framework of Zheng *et al.* [135] that could increase the accuracy of the reconstruction around occlusion boundaries or tilted surfaces, where the selection of the correct views during occlusion is crucial. The randomized optimization could include the size of the support windows as an additional variable, to obtain a sort of multi-scale result and potentially get finer results around small structures and overcome error inside homogeneous regions.

The original patch refinement step is a simple bisection search around the current solution, exponentially decreasing the search radius. Even if it proved to be reliable, it would be interesting to compare it with a proper optimization scheme such as gradient descent or a KLT refinement step as [56] with the potential to reduce runtime and improve reconstruction.

To take advantage of the parallel nature and low memory consumption of Gipuma, a possible continuation of the method might be to input a multi-camera video sequence of deforming surfaces. Starting from the initial reconstruction at the first time step, it could reason on subsequent frames and jointly compute 3D reconstruction and scene flow. In this case, the method would impose not only locally spatial smoothness (as it is the case when fitting patches in 3D space) but also temporal smoothness, effectively fitting patches in the space-time dimension.

Despite its power, Gipuma has various limitations. The main one is its generic approach: computing the depth map for each image limits the method for large scenes. A solution might be at least to try to cluster the scene in smaller parts, such that each one would not contain too many images. Another limitation lies in the implementation of the method. Since it's programmed completely on the GPU, it has strong memory limitations.

It still has to be tested how fast the method can be. It showed impressive numbers for runtime and accuracy on large images, but we don't know if it can be pushed to be used for real-time applications such as SLAM methods for robotics.

Both Gipuma 4 and the shading based surface refinement presented in Chapter 6 use the same fusion approach to merge depthmaps for the final reconstruction. The common pipeline includes conversion to a final triangulated mesh in a separate step with common surface fitting methods [14, 76].

Instead, depthmap fusion and mesh reconstruction could be merged in a single optimization step. We would directly fit a smooth surface on the 3D points, making use of the redundancy across views and the uncertainty estimation obtained directly from the image similarity measure. Otherwise, we could start with a rough initial triangulated mesh by common methods [14, 76] and refine it by minimizing a photometric error function over the images.

A similar surface fitting could be used for the dense but unreliable normals obtained with the shading based method of Chapter 6. In this case, rather than directly integrating normal maps in 2D, and fuse afterward the depth location in space, it would be better to keep the putative rays of the normal prediction in 3D and directly fit a smooth surface. The result would be a multi-view normal integration with a redundancy of noisy normal predictions and anchored on fixed 3D points.

We plan in the future to include a comparison of our method w.r.t. simpler interpolation schemes to inpaint the surface normal image.

Shading based refinement is still an open topic [85, 92], in this direction it would be interesting to continue exploring the potential of deep learning methods for the separation of shading and reflectance from the images [98, 109]. It is still not clear how well neural network are able to generalize in such a way that they could be used for every lighting configuration. Nevertheless, it would be worth exploring this direction as a tool to obtain additional object properties such as color and reflectance function from the surface.

It would be interesting to include the multi-patch similarity network of Chapter 5 inside the error computation of Gipuma. An interesting direction would be to stick to the idea of a view specific model but use it in similarity learning and incrementally refine the network in a multi-view stereo dataset, as soon as more images arrive, with application to video scenes. The similarity function would be effectively fine-tuned to match the images of the specific scene it is looking at.

To validate the effective generalization power of the similarity measure, in future work we plan to perform a quantitative comparison of a reconstruction of a dataset obtained with a model trained on another dataset, see Chapter 5. That's a crucial missing experiment for a proper scientific corroboration of our generality claim.

The architecture of the network could be modified to include occlusion and visibility of patches or even the underlining surface orientation. In this case, it could to be interesting to include it in inside the random sampling of Gipuma to guide the otherwise randomized selection of normals.

Our similarity network is just the starting point to include learning based approach inside multi-view stereo reconstruction. Recently, bolder ideas have emerged, all the way to learning an end-to-end mapping from images to (volumetric, low-resolution) 3D models [18, 73]. While we do not see a reason to replace the geometrically transparent and well-understood Structure from Motion pipeline with a learned black-box predictor, we do agree that certain steps of the pipeline lack a principled solution and might be best explored with deep learning.

BIBLIOGRAPHY

[1] Jens Ackermann, Martin Ritz, André Stork, and Michael Goesele. "Removing the example from example-based photometric stereo." In: *Trends and Topics in Computer Vision*. Springer, 2010, pp. 197–210.

[2] Amit Agrawal, Ramesh Raskar, and Rama Chellappa. "What is the range of surface reconstructions from a gradient field?" In: *ECCV 2006* ().

[3] Hernan Badino and Takeo Kanade. "A Head-Wearable Short-Baseline Stereo System for the Simultaneous Estimation of Structure and Motion." In: *IAPR MVA 2011* (). URL: `http://www.lelaps.de/papers/badino_mva12.pdf`.

[4] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation." In: *arXiv preprint arXiv:1511.00561* (2015).

[5] Christian Bailer, Manuel Finckh, and Hendrik PA Lensch. "Scale robust multi view stereo." In: *ECCV 2012* ().

[6] Linchao Bao, Qingxiong Yang, and Hailin Jin. "Fast Edge-Preserving PatchMatch for Large Displacement Optical Flow." In: *CVPR 2014* ().

[7] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. "PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing." In: *SIGGRAPH 2009* ().

[8] Thabo Beeler, Derek Bradley, Henning Zimmer, and Markus Gross. "Improved reconstruction of deforming surfaces by cancelling ambient occlusion." In: *ECCV 2012* ().

[9] Frederic Besse, Carsten Rother, Andrew Fitzgibbon, and Jan Kautz. "PMBP: PatchMatch Belief Propagation for Correspondence Field Estimation." In: *BMVC 2012* ().

[10] Andrew Blake, Andrew Zisserman, and Greg Knowles. "Surface descriptions from stereo and shading." In: *Image and Vision Computing* 3.4 (1985), pp. 183–191.

[11] Michael Bleyer, Christoph Rhemann, and Carsten Rother. “PatchMatch Stereo - Stereo Matching with Slanted Support Windows.” In: *BMVC 2011* (). URL: `http://publik.tuwien.ac.at/files/PubDat_201949.pdf`.

[12] Michael Bleyer, Christoph Rhemann, and Carsten Rother. “PatchMatch Stereo - Stereo Matching with Slanted Support Windows.” In: *BMVC 2011* (). URL: `http://publik.tuwien.ac.at/files/PubDat_201949.pdf`.

[13] Peter Burt, Lambert Wixson, and Garbis Salgian. “Electronically directed "focal" stereo.” In: *ICCV 1995* ().

[14] Fatih Calakli and Gabriel Taubin. “SSD: Smooth signed distance surface reconstruction.” In: *Computer Graphics Forum*. Vol. 30. 7. Wiley Online Library. 2011, pp. 1993–2002.

[15] Neill DF Campbell, George Vogiatzis, Carlos Hernández, and Roberto Cipolla. “Using multiple hypotheses to improve depth-maps for multi-view stereo.” In: *ECCV 2008* ().

[16] Augustin Cauchy. “Méthode générale pour la résolution des systemes d¿équations simultanées.” In: *Comp. Rend. Sci. Paris* 25.1847 (1847), pp. 536–538.

[17] Ju Yong Chang, Kyoung Mu Lee, and Sang Uk Lee. “Multiview normal field integration using level set methods.” In: *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE. 2007, pp. 1–8.

[18] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. “3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction.” In: *ECCV 2016* ().

[19] Dan Claudiu Cireşan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. “Deep, big, simple neural nets for handwritten digit recognition.” In: *Neural computation* 22.12 (2010), pp. 3207–3220.

[20] David Claus and Andrew W Fitzgibbon. “A rational function lens distortion model for general cameras.” In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE. 2005, pp. 213–219.

[21] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. "Fast and accurate deep network learning by exponential linear units (elus)." In: *arXiv preprint arXiv:1511.07289* (2015).

[22] Robert T Collins. "A space-sweep approach to true multi-image matching." In: *CVPR 1996* ().

[23] Robert T Collins. "A space-sweep approach to true multi-image matching." In: *CVPR*. IEEE. 1996, pp. 358–363.

[24] James M Coughlan and Alan L Yuille. "The Manhattan world assumption: Regularities in scene statistics which enable Bayesian inference." In: *NIPS 2000* ().

[25] James Edwin Cryer, Ping-Sing Tsai, and Mubarak Shah. "Integration of shape from shading and stereo." In: *Pattern Recognition* 28.7 (1995), pp. 1033–1043.

[26] George Cybenko. "Approximation by superpositions of a sigmoidal function." In: *Mathematics of Control, Signals, and Systems (MCSS)* 2.4 (1989), pp. 303–314.

[27] Amaël Delaunoy, Emmanuel Prados, Pau Gargallo I Piracés, Jean-Philippe Pons, and Peter Sturm. "Minimizing the multi-view stereo reprojection error for triangular surface meshes." In: *BMVC 2008* ().

[28] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 248–255.

[29] Frédéric Devernay and Olivier Faugeras. "Computing differential properties of 3-D shapes from stereoscopic images without 3-D models." In: *CVPR 1994* ().

[30] David Eigen and Rob Fergus. "Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture." In: *ICCV 2015* ().

[31] Nils Einecke and Julian Eggert. "Stereo image warping for improved depth estimation of road surfaces." In: *Intelligent Vehicle Symposium 2013* ().

[32] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. "Efficient Belief Propagation for Early Vision." In: *IJCV* 70.1 (2006).

[33] David F Fouhey, Arpan Gupta, and Martial Hebert. "Data-driven 3D primitives for single image understanding." In: *ICCV 2013* ().

[34] David Ford Fouhey, Abhinav Gupta, and Martial Hebert. "Unfolding an indoor origami world." In: *ECCV 2014* ().

[35] Robert T Frankot and Rama Chellappa. "A method for enforcing integrability in shape from shading algorithms." In: *IEEE TPAMI* 10.4 (1988), pp. 439–451.

[36] Pascal Fua and Yvan G Leclerc. "Object-centered surface reconstruction: Combining multi-image stereo and shading." In: *IJCV* 16.1 (1995), pp. 35–56.

[37] Yasutaka Furukawa and Jean Ponce. "Accurate, Dense, and Robust Multiview Stereopsis." In: *IEEE TPAMI* 32.8 (2010), pp. 1362–1376. ISSN: 0162-8828. DOI: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2009.161.

[38] Yasutaka Furukawa, Brian Curless, Steven M Seitz, and Richard Szeliski. "Manhattan-world stereo." In: *CVPR 2009* ().

[39] Andrea Fusiello, Vito Roberto, and Emanuele Trucco. "Efficient stereo with multiple windowing." In: *CVPR 1997* ().

[40] Silvano Galliani, Michael Breuß, and Yong Chul Ju. "Fast and Robust Surface Normal Integration by a Discrete Eikonal Equation." In: *BMVC 2012* ().

[41] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. "Massively Parallel Multiview Stereopsis by Surface Normal Diffusion." In: *ICCV 2015*.

[42] Silvano Galliani and Konrad Schindler. "Just look at the image: viewpoint-specific surface normal prediction for improved multi-view reconstruction." In: *CVPR 2016*.

[43] David Gallup, J-M Frahm, Philippos Mordohai, Qingxiong Yang, and Marc Pollefeys. "Real-time plane-sweeping stereo with multiple sweeping directions." In: *CVPR 2007* ().

[44] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite." In: *CVPR 2012* ().

[45] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite." In: *CVPR 2012* ().

[46] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[47] Alex Graves. "Generating sequences with recurrent neural networks." In: *arXiv preprint arXiv:1308.0850* (2013).

[48] T. S. F. Haines and R. C. Wilson. "Integrating Stereo with Shape-from-Shading derived Orientation Information." In: *BMVC 2007* ().

[49] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C. Berg. "MatchNet: Unifying Feature and Metric Learning for Patch-Based Matching." In: *CVPR 2015*.

[50] C. Häne, L. Heng, G. H. Lee, A. Sizov, and M. Pollefeys. "Real-Time Direct Dense Matching on Fisheye Images Using Plane-Sweeping Stereo." In: *3DV 2014*.

[51] Marsha J Hannah. "Computer Matching of Areas in Stereo Images." PhD thesis. Stanford University, 1974.

[52] Richard Hartley and Andrew Zisserman. *Multiple View Geometry*. second. Cambridge University Press, 2004.

[53] Wilfried Hartmann, Silvano Galliani, Michal Havlena, Konrad Schindler, and Luc Van Gool. "Learned multi-patch similarity." In: *ICCV 2017* ().

[54] Janne Heikkilä and Olli Silvén. "A four-step camera calibration procedure with implicit image correction." In: *CVPR*. Vol. 97. 1997, pp. 1106–1112.

[55] Philipp. Heise, Sebastian Klose, Brian Jensen, and Alois Knoll. "PM-Huber: PatchMatch with Huber Regularization for Stereo Matching." In: *ICCV 2013* (). DOI: `10.1109/ICCV.2013.293`.

[56] Philipp Heise, Brian Jensen, Sebastian Klose, and Alois Knoll. "Variational patchmatch multiview reconstruction and refinement." In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 882–890.

[57] Heiko Hirschmüller. "Stereo Processing by Semi-Global Matching and Mutual Information." In: *IEEE TPAMI* 30.2 (2008), pp. 328–341.

[58] Jeffrey Ho, Jongwoo Lim, Ming-Hsuan Yang, and David Kriegman. "Integrating surface normal vectors using fast marching method." In: *ECCV 2006* ().

[59] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. "Surface reconstruction from unorganized points." In: *Computer Graphics* 26.2 (1992), pp. 71–78.

[60] Berthold KP Horn and Michael J Brooks. "The variational approach to shape from shading." In: *CVGIP* 33.2 (1986), pp. 174–208.

[61] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks." In: *Neural networks* 4.2 (1991), pp. 251–257.

[62] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators." In: *Neural networks* 2.5 (1989), pp. 359–366.

[63] Xiaoyan Hu and Philippos Mordohai. "Least Commitment, Viewpoint-Based, Multi-view Stereo." In: *3DIMPVT 2012* ().

[64] Stephen S. Intille and Aaron F. Bobick. "Disparity-space images and large occlusion stereo." English. In: *ECCV 1994* (). DOI: `10.1007/BFb0028349`. URL: `http://dx.doi.org/10.1007/BFb0028349`.

[65] M. Jancosek and T. Pajdla. "Multi-view reconstruction preserving weakly-supported surfaces." In: *CVPR 2011* (). ISSN: 1063-6919.

[66] R. Jensen, A. Dahl, G. Vogiatzis, E. Tola, and Aanæ s H. "Large Scale Multi-view Stereopsis Evaluation." In: *CVPR 2014* ().

[67] R. Jensen, A. Dahl, G. Vogiatzis, E. Tola, and H. Aanæs. "Large Scale Multi-view Stereopsis Evaluation." In: *CVPR 2014*.

[68] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. "Caffe: Convolutional Architecture for Fast Feature Embedding." In: *arXiv preprint arXiv:1408.5093* (2014).

[69] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. "Caffe: Convolutional Architecture for Fast Feature Embedding." In: *arXiv preprint arXiv:1408.5093* (2014).

[70] Hailin Jin, Daniel Cremers, Dejun Wang, Emmanuel Prados, Anthony Yezzi, and Stefano Soatto. “3-d reconstruction of shaded objects from multiple images under unknown illumination.” In: *IJCV* 76.3 (2008), pp. 245–256.

[71] Takeo Kanade and Masatoshi Okutomi. “A stereo matching algorithm with an adaptive window: Theory and experiment.” In: *IEEE TPAMI* 16.9 (1994), pp. 920–932.

[72] Sing Bing Kang, R. Szeliski, and Jinxiang Chai. “Handling occlusions in dense multi-view stereo.” In: *CVPR 2001* (). ISSN: 1063-6919.

[73] Abhishek Kar, Christian Häne, and Jitendra Malik. “Learning a Multi-View Stereo Machine.” In: *arXiv preprint arXiv:1708.05375* (2017).

[74] Sagi Katz, Ayellet Tal, and Ronen Basri. “Direct Visibility of Point Sets.” In: *ACM SIGGRAPH 2007*.

[75] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. “Poisson surface reconstruction.” In: *Eurographics Symposium on Geometry Processing 2006* ().

[76] Michael Kazhdan and Hugues Hoppe. “Screened poisson surface reconstruction.” In: *ACM Transactions on Graphics* 32.3 (2013), p. 29.

[77] Ira Kemelmacher-Shlizerman and Steven M Seitz. “Face reconstruction in the wild.” In: *ICCV 2011* ().

[78] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. “Self-Normalizing Neural Networks.” In: *arXiv preprint arXiv:1706.02515* (2017).

[79] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. “Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction.” In: 36.4 (2017).

[80] Kalin Kolev, Maria Klodt, Thomas Brox, and Daniel Cremers. “Continuous global optimization in multiview 3d reconstruction.” In: *International Journal of Computer Vision* 84.1 (2009), pp. 80–96.

[81] Vladimir Kolmogorov and Ramin Zabih. “Computing Visual Correspondence with Occlusions using Graph Cuts.” In: *ICCV 2001* ().

[82] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks.” In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[83] Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. "Efficient Multi-View Reconstruction of Large-Scale Scenes using Interest Points, Delaunay Triangulation and Graph Cuts." In: *ICCV 2007* ().

[84] L'ubor Ladický, Bernhard Zeisl, and Marc Pollefeys. "Discriminatively Trained Dense Surface Normal Estimation." In: *ECCV 2014* ().

[85] Fabian Langguth, Kalyan Sunkavalli, Sunil Hadap, and Michael Goesele. "Shading-aware multi-view stereo." In: *European Conference on Computer Vision*. Springer. 2016, pp. 469–485.

[86] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. "Backpropagation applied to handwritten zip code recognition." In: *Neural computation* 1.4 (1989), pp. 541–551.

[87] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[88] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3431–3440.

[89] D. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints." In: *IJCV* 60.2 (2004), pp. 91–110.

[90] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. "Rectifier nonlinearities improve neural network acoustic models." In: *Proc. ICML*. Vol. 30. 1. 2013.

[91] George Marsaglia. "Choosing a Point from the Surface of a Sphere." In: *Annals of Mathematical Statistics* 43.2 (1972), pp. 645–646. DOI: 10.1214/aoms/1177692644. URL: http://dx.doi.org/10.1214/aoms/1177692644.

[92] Daniel Maurer, Yong Chul Ju, Michael Breuß, and Andrés Bruhn. "Combining Shape from Shading and Stereo: A Variational Approach for the Joint Estimation of Depth, Illumination and Albedo." In: *BMVC*. 2016.

[93] N. J. Mitra, A. Nguyen, and L. Guibas. "Estimating surface normals in noisy point cloud data." In: *Int'l J Computational Geometry & Applications* 14.4/5 (2004), pp. 261–276.

[94] M. Okutomi and T. Kanade. "A multiple-baseline stereo." In: *IEEE TPAMI* 15.4 (1993), pp. 353–363. ISSN: 0162-8828. DOI: `10.1109/34.206955`.

[95] James M Ortega and Werner C Rheinboldt. *Iterative solution of nonlinear equations in several variables*. Vol. 30. Siam, 1970.

[96] Bui Tuong Phong. "Illumination for computer generated pictures." In: *Communications of the ACM* 18.6 (1975), pp. 311–317.

[97] R. Ranftl, S. Gehrig, T. Pock, and H. Bischof. "Pushing the limits of stereo using variational stereo estimation." In: *Intelligent Vehicles Symposium 2012* (). ISSN: 1931-0587. DOI: `10.1109/IVS.2012.6232171`.

[98] Konstantinos Rematas, Tobias Ritschel, Mario Fritz, Efstratios Gavves, and Tinne Tuytelaars. "Deep reflectance maps." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4508–4516.

[99] Christoph Rhemann, Asmaa Hosni, Michael Bleyer, Carsten Rother, and Margrit Gelautz. "Fast Cost-Volume Filtering for Visual Correspondence and Beyond." In: *CVPR 2011* ().

[100] Stephan R Richter and Stefan Roth. "Discriminative Shape from Shading in Uncalibrated Illumination." In: *CVPR 2015* ().

[101] Dimitrios Samaras, Dimitris Metaxas, Pascal Fua, and Yvan G Leclerc. "Variable albedo surface reconstruction from stereo and shape from shading." In: *CVPR 2000* ().

[102] Daniel Scharstein and Richard Szeliski. "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms." In: *IJCV* 47.1-3 (2002), pp. 7–42. ISSN: 0920-5691. DOI: `10.1023/A:1014573219977`. URL: `http://dx.doi.org/10.1023/A:1014573219977`.

[103] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. "Pixelwise View Selection for Unstructured Multi-View Stereo." In: *European Conference on Computer Vision (ECCV)*. 2016.

[104] Thomas Schöps, Johannes SchÃ¶nberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. "A Multi-View Stereo Benchmark with High-Resolution Images and Multi-Camera Videos." In: *CVPR 2017*.

[105] Thomas Schöps, Johannes L Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. "A Multi-View Stereo Benchmark with High-Resolution Images and Multi-Camera Videos." In: ().

[106] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. "A comparison and evaluation of multi-view stereo reconstruction algorithms." In: *CVPR 2006* ().

[107] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. "A comparison and evaluation of multi-view stereo reconstruction algorithms." In: *CVPR*. 2006.

[108] Shuhan Shen. "Accurate Multiple View 3D Reconstruction Using Patch-Based Stereo for Large-Scale Scenes." In: *IEEE TIP* 22.5 (2013), pp. 1901–1914. ISSN: 1057-7149. DOI: `10.1109/TIP.2013.2237921`.

[109] Jian Shi, Yue Dong, Hao Su, and X Yu Stella. "Learning Non-Lambertian Object Intrinsics across ShapeNet Categories Supplementary Material." In: ().

[110] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. "Indoor Segmentation and Support Inference from RGBD Images." In: *ECCV 2012* ().

[111] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. "Discriminative Learning of Deep Convolutional Feature Point Descriptors." In: *ICCV 2015*.

[112] K. Simonyan, A. Vedaldi, and A. Zisserman. "Learning Local Feature Descriptors Using Convex Optimisation." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.8 (2014), pp. 1573–1585.

[113] C. Strecha, W. V. Hansen, L. J. V. Gool, P. Fua, and U. Thoennessen. "On benchmarking camera calibration and multi-view stereo for high resolution imagery." In: *CVPR 2008* ().

[114] Christoph Strecha, Wolfgang von Hansen, Luc J. Van Gool, Pascal Fua, and Ulrich Thoennessen. "On benchmarking camera calibration and multi-view stereo for high resolution imagery." In: *CVPR 2008* ().

[115] Engin Tola, V. Lepetit, and P. Fua. "DAISY: An Efficient Dense Descriptor Applied to Wide-Baseline Stereo." In: *IEEE TPAMI* 32.5 (2010), pp. 815–830. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2009.77.

[116] Engin Tola, Christoph Strecha, and Pascal Fua. "Efficient large-scale multi-view stereo for ultra high-resolution image sets." English. In: *MVA* 23.5 (2012), pp. 903–920. ISSN: 0932-8092. DOI: 10.1007/s00138-011-0346-8. URL: http://dx.doi.org/10.1007/s00138-011-0346-8.

[117] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. "Show and tell: A neural image caption generator." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3156–3164.

[118] Christoph Vogel, Konrad Schindler, and Stefan Roth. "Piecewise Rigid Scene Flow." In: *ICCV 2013* ().

[119] Christian Wallraven, Volker Blanz, and Thomas Vetter. "3D-Reconstruction of Faces: Combining Stereo with Class-Based Knowledge." In: *DAGM 1999* ().

[120] Michael Weinmann, Aljosa Osep, Roland Ruiters, and Reinhard Klein. "Multi-view normal field integration for 3d reconstruction of mirroring objects." In: *Proceedings of the IEEE international conference on computer vision*. 2013, pp. 2504–2511.

[121] Chenglei Wu, B. Wilburn, Y. Matsushita, and C. Theobalt. "High-quality shape from multi-view stereo and shading under general illumination." In: *CVPR 2011* ().

[122] Zhongquan Wu and Lingxiao Li. "A line integration based method for depth recovery from surface normals." In: *ICPR 1988* ().

[123] Songhua Xu, Athinodoros Georghiades, Holly Rushmeier, Julie Dorsey, and Leonard McMillan. "Image guided geometry inference." In: *3D Data Processing, Visualization, and Transmission, Third International Symposium on*. IEEE. 2006, pp. 310–317.

[124] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. "LIFT: Learned Invariant Feature Transform." In: *CoRR* abs/1603.09114 (2016).

[125] Kuk-Jin Yoon and In So Kweon. "Adaptive support-weight approach for correspondence search." In: *IEEE TPAMI* 28.4 (2006), pp. 650–656.

[126] David M Young. *Iterative solution of large linear systems*. Elsevier, 2014.

[127] Jie Yu, Qi Tian, Jaume Amores, and Nicu Sebe. "Toward robust distance metric analysis for similarity estimation." In: *CVPR 2006*.

[128] C. Zach. "Fast and high quality fusion of depth maps." In: *3DPVT 2008* ().

[129] Christopher Zach. "Fast and high quality fusion of depth maps." In: *3DPVT 2008*.

[130] S. Zagoruyko and N. Komodakis. "Learning to compare image patches via convolutional neural networks." In: *CVPR 2015*.

[131] J. Zbontar and Y. LeCun. "Computing the stereo matching cost with a convolutional neural network." In: *CVPR 2015*.

[132] Bernhard Zeisl, Christopher Zach, and Marc Pollefeys. "Stereo Reconstruction of Building Interiors with a Vertical Structure Prior." In: *3DIMPVT 2011* ().

[133] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. "Appearance-based Gaze Estimation in the Wild." In: *CVPR 2015* ().

[134] Enliang Zheng, Enrique Dunn, Vladimir Jojic, and Jan-Michael Frahm. "PatchMatch Based Joint View Selection and Depthmap Estimation." In: *CVPR 2014* ().

[135] Enliang Zheng, Enrique Dunn, Vladimir Jojic, and Jan-Michael Frahm. "Patchmatch based joint view selection and depthmap estimation." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1510–1517.

[136] Christian Zinner, Martin Humenberger, Kristian Ambrosch, and Wilfried Kubinger. "An optimized software-based implementation of a census-based stereo matching algorithm." In: *Advances in Visual Computing*. 2008, pp. 216–227.