Flying with Payloads

Other Publication

Author(s):
Fekry Kamel, Mina S.

Publication date:
2013-05

Permanent link:
https://doi.org/10.3929/ethz-a-010186593

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute for Dynamic Systems and Control
Institut für Dynamische Systeme und Regelungstechnik

Mina Samir Fekry Kamel

Flying with Payloads

Semester Thesis

Institute for Dynamic Systems and Control
Swiss Federal Institute of Technology (ETH) Zurich

Supervision

Federico Augugliaro
Prof. Raffaello D'Andrea

May 2013

IDSC-RD-FA-05

Abstract

This semester thesis shows how carrying a payload affects the dynamics of the quadrocopter and trajectory feasibility. Moreover it shows that compensation for the payload's inertial parameters can improve the tracking performance. First, recursive least square estimation algorithm with fading memory is presented to estimate the payload's inertial parameters, and modification in the on board controller is introduced to compensate for the payload. The results are implemented and tested in the Flying Machine Arena in ETH Zurich. Finally we show how the feasible set of time-optimal trajectories can be influenced by the presence of a grasped object.

Contents

Introduction

Recently aerial construction has been one of the research areas carried out in the Flying Machine Arena at ETH Zurich [1, 14]. Autonomous flying robots (quadrocopters) are used to assemble small building elements autonomously. This is a challenging task that requires multidisciplinary effort.

Aerial construction offers great advantages over conventional construction approach: aerial robots have fewer workspace constraints allowing the architecture to be more creative in the design, it does not require scaffolding or special purpose tower cranes. However, this novel construction technique has some limitations such as the limited payload that aerial robot can carry and the limited battery life. The first limitation is partially solved using cooperative techniques as in [11, 12]. The second limitation is solved using charging stations, allowing the robots to recharge their batteries whenever needed.

An important aspect in the aerial construction is the ability of the vehicle to transport objects with a good reference tracking. In this work we investigate an adaptive strategy to improve the tracking performance in the presence of unknown payload. Assembling a structure shouldn't be only precise, but it should be also as quick as possible. So, time-optimal trajectory generation is considered [7] and we investigated how the feasible set of time-optimal trajectories can be influenced under the presence of a payload.

In Chapter 1 we use rigid body equations to derive the quadrocopter model with payload. In Chapter 2 recursive least square estimation with fading memory algorithm is presented to estimate the payload inertial parameters and experimentally validated. In Chapter 3 we derive the control law from the rigid body equations and we show how adapting the controller can improve the reference tracking of the vehicle. Finally in Chapter 4 we show the influence of the payload on the trajectories feasible set.

Nomenclature

Symbols

R_O^V	Rotation matrix from reference O to reference V	
x_v, y_v, z_v	Axes rigidly attached to the vehicle	
U_1	Total thrust	[N]
U_2	Moment around x_v axis	[Nm]
U_3	Moment around y_v axis	[Nm]
U_4	Moment around z_v axis	[Nm]
m_T	Total mass (vehicle + payload)	[kg]
m_v	Mass of the vehicle	[kg]
m_L	Mass of the payload	[kg]
g	Gravity acceleration, 9.81	[m/s^2]
ω	Rotational rate in the vehicle frame	[rad/s]
r_{off}	Vector indicating the center of mass offset w.r.t. the geometric center of mass	[m]
x_{off}	Center of mass offset on the x_v axis	[m]
y_{off}	Center of mass offset on the y_v axis	[m]
$\mathbf{J}$	Tensor of inertia	[kg m^2]
F_i	Thrust of the $i-th$ propeller	[N]
p, q, r	Rotational rates around x_v, y_v and z_v axes	[rad/s]
K_k	Kalman gain matrix	
λ	Forgetting factor	
P_k	Estimation covariance matrix	
R_k	Measurement noise covariance matrix	
f_x, f_y	Mass normalized disturbance on x and y axes	[m/s^2]
$\hat{x}_{off}$	Estimation of center of mass offset along x axis	[m]
$\hat{y}_{off}$	Estimation of center of mass offset along y axis	[m]
$\hat{m}_T$	Estimation of total mass	[kg]
$\check{p}, \check{q}, \check{r}$	Desired rotational rates in the vehicle frame	[rad/sec]
$\check{F}_i$	Desired $i-th$ motor thrust	[N]
τ_{pq}	Time constant for p, q	[s]
τ_r	Time constant for r	[s]
$\check{\tau}$	Mass normalized desired thrust	[m/s^2]
F_{min}	Minimum motor thrust	[N]
F_{max}	Maximum motor thrust	[N]
ω_{max}	Maximum rotational rate	[rad/s]
acc_{max}	Maximum acceleration	[m/s^2]
u_x	Maximum allowable jerk along x axis	[m/s^3]
u_y	Maximum allowable jerk along y axis	[m/s^3]

Acronyms and Abbreviations

GCoM	Geometric center of mass.
CoM	Center of mass.
RLS	Recursive least square.
MSE	Mean squared error.

Chapter 1

Quadrocopter Model With Payload

In this chapter, we derive an analytical model of the quadrocopter with an attached payload using rigid body dynamics. This model is crucial to understand the effects of an additional payload on the quadrocopter dynamics. Moreover we will use it to estimate the inertial parameters in Chapter 2 and to derive a controller that can compensate for the additional payload in Chapter 3.

First, we make the following assumptions about the quadrocopter and the attached payload to simplify the model derivation:

- The inertia matrix is time-invariant.
- The quadrocopter is symmetric with respect to x and y axes.
- The payload does not significantly affect the inertia of the quadrocopter. This assumption is valid when the dimension of the payload is not so large and the payload's mass distribution is uniform.

Two frames are defined to derive the model, an inertial frame attached to the ground O and another non inertial frame attached to the vehicle V in the geometric center of mass (GCoM) . The matrix R_O^V that represent the rotation matrix from the frame V to the frame O is given by

$$R_O^V = \begin{pmatrix} c_\psi c_\theta & -s_\psi c_\phi + c_\psi s_\theta s_\phi & s_\psi s_\phi + c_\psi s_\theta c_\phi \\ s_\psi c_\theta & c_\psi c_\phi + s_\psi s_\theta s_\phi & -c_\psi s_\phi + s_\psi s_\theta c_\phi \\ -s_\theta & c_\theta s_\theta & c_\theta c_\phi \end{pmatrix} \tag{1.1}$$

where ϕ, θ, ψ are the roll, pitch and yaw angles respectively, c and s denote cosine and sine.
We also define the vector $\mathbf{r_{off}}$ to indicate the center of mass (CoM) with respect to the GCoM as shown in Figure 1.1.

Moreover we assume that the payload is rigidly attached to the quadrocopter, and generally it will introduce an offset in the center of mass (CoM). The CoM offset is measured with respect to the GCoM in the vehicle frame.

1.1 Translational Dynamics

To derive the translational dynamics we have to identify the external forces acting on the quadrocopter, these forces are given by:

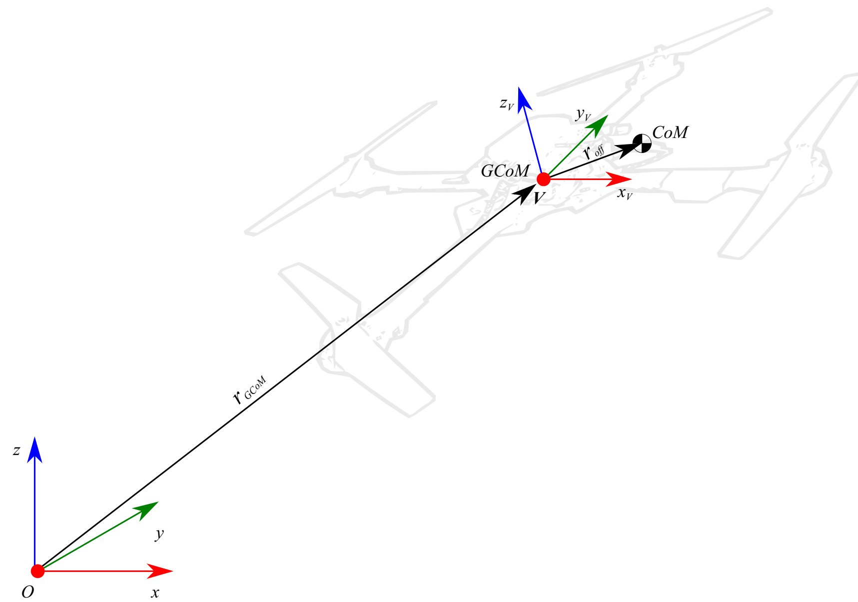


Figure 1.1: GCoM and CoM of quadrocopter with payload

- The gravity force, defined in the inertial frame by the vector $\begin{pmatrix} 0 & 0 & -m_T g \end{pmatrix}^T$.
- The total thrust generated by the propellers, defined in the vehicle frame by the vector $\begin{pmatrix} 0 & 0 & U_1 \end{pmatrix}^T$ where $U_1 = \sum_{i=1}^{4} F_i$, F_i is the $i-th$ motor thrust.

Using Newton's equations of motion with respect to the GCoM it is possible to derive the GCoM acceleration as follows

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = R_O^V \begin{pmatrix} 0 \\ 0 \\ U_1/m_T \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} - \dot{\omega} \times r_{off} - \omega \times (\omega \times r_{off}) \tag{1.2}$$

Note that the terms $\dot{\omega} \times r_{off}$ and $\omega \times (\omega \times r_{off})$ are the centripetal acceleration and coriolis acceleration respectively. These terms are generally small in comparison with other terms, so we assume that they will be rejected by the controller and in the following analysis we will neglect them.

1.2 Rotational Dynamics

The rotational dynamics are derived from Euler equation

$$\frac{dL}{dt} + \omega \times L = M, \tag{1.3}$$

where L is the angular momentum with respect to the inertial frame, and M is the vector of torques acting on the vehicle. The angular momentum is given by $L = \mathbf{J}\omega$. where $\mathbf{J}$ is the inertia tensor with respect to the vehicle principle axes

The torques acting on the vehicle M are given by

$$M = \begin{pmatrix} U_2 \\ U_3 \\ U_4 \end{pmatrix} - r_{off} \times \begin{pmatrix} 0 \\ 0 \\ U_1 \end{pmatrix}, \tag{1.4}$$

where $U_1 = \sum_{i=1}^{4} F_i$, F_i is the $i-th$ motor thrust, $U_2 = L\,(F_2 - F4)$, $U_3 = L\,(F_3 - F_1)$, $U_4 = \bar{k}\,(F_1 - F_2 + F_3 - F_4)$ are the moments around x_V, y_V and z_V respectively, $\bar{k}$ is an appropriate constant.

Substituting in Equation (1.3) we have

$$\mathbf{J}\dot{\omega} = \mathbf{J} \begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} U_2 \\ U_3 \\ U_4 \end{pmatrix} - \omega \times \mathbf{J}\omega - r_{off} \times \begin{pmatrix} 0 \\ 0 \\ U_1 \end{pmatrix}. \tag{1.5}$$

The total thrust U_1 is applied in the center of mass, hence it introduces torque given by the last term in Equation (1.5).

Equations (1.2) and (1.5) represent a full model of the quadrocopter with a payload.

Chapter 2

Inertial Parameter Estimation

> The most probable value of the unknown quantities will be that in which the sum of the squares of the differences between the actually observed and the computed values multiplied by numbers that measure the degree of precision is minimum.
>
> (Carl Friedrich Gauss)

In this chapter, the algorithm used in the inertial parameters estimation is presented. By inertial parameters we mean the total mass and the center of mass offset. A recursive least square (RLS) algorithm with fading memory is presented and the experimental results of the estimation in hover condition as well as in forward flying condition are shown.
In Section 2.1 we introduce the recursive least square estimation algorithm, and we will show how to apply it to estimate the vehicle's total mass in Section 2.2. In Section 2.3 we will show how to estimate the CoM offset. Finally, Section 2.4 presents the experimental results of the algorithm.

2.1 Least Squares Estimation

We start by introducing the batch least squares problem, then we will introduce the recursive least squares with fading memory.

2.1.1 Batch Least Squares

Formally, suppose that θ is a vector of unknown n parameters and Z is a vector of k noisy measurements. Now we consider a simple case where every measurement z_i is a linear combination of the parameters θ_i, with some additional measurement noise v_i. Thus we can write the following linear system :

$$Z = H\theta + V \tag{2.1}$$

where V is the measurement noise vector, and H is a $k \times n$ matrix.
The goal of least squares estimation is to find the best estimate of θ, $\hat{\theta}$ that minimizes the following cost function

$$J\left(\hat{\theta}\right) = \left(Z - H\hat{\theta}\right)^T S \left(Z - H\hat{\theta}\right) \tag{2.2}$$

where S is a semidefinite positive weighting matrix. Clearly, $J\left(\hat{\theta}\right)$ is a convex function, and has a global minimum given by

$$\hat{\theta} = \left(H^T S H\right)^{-1} H^T S Z \tag{2.3}$$

Equation (2.3) is good when all the measurements are available ahead of time, however when the measurements are obtained sequentially, the dimension of the matrix H becomes very large over time, and the computational effort will be prohibitive. To overcome this problem, we use the recursive least squares algorithm presented in the Subsection 2.1.2;

2.1.2 Recursive Least Squares With Fading Memory

Let's suppose that we receive a new measurement every time step, and we want to choose the weighting matrix S such that older data is gradually discarded in favor of more recent information. We choose S as follows

$$S = \begin{pmatrix} \lambda^{(k-1)} & & & \\ & \lambda^{(k-2)} & & \\ & & \ddots & \\ & & & 1 \end{pmatrix} \tag{2.4}$$

where λ is called *the forgetting factor*, and $\lambda \in (0, 1]$.
In order to update recursively our estimation every time a new measurement is available, we use a linear recursive estimator given by

$$\begin{aligned} z_k &= H_k \theta + v_k \\ \hat{\theta}_k &= \hat{\theta}_{k-1} + \underbrace{K_k}_{\text{Kalman gain}} \underbrace{\left(z_k - H_k \hat{\theta}_{k-1}\right)}_{\text{correction term}} \end{aligned} \tag{2.5}$$

the matrix K_k is called *Kalman gain* or *gain matrix*, and $\left(z_k - H_k \hat{\theta}_k\right)$ is called the correction term because it compares the difference between the new measurement and the prediction using the last estimate.Before we present the expression of the optimal gain matrix, we have to do some assumptions regarding the measurement noise v_k. We assume that v_k is zero mean noise with covariance R_k . Moreover, we assume that v_k is independent at each time step, i.e. the noise is white.

The expression of the optimal gain matrix K_k, and the estimation covariance P_k is given by

$$\begin{aligned} K_k &= \lambda^{-1} P_{k-1} H_k^T \left(R_k + \lambda^{-1} H_k^T P_{k-1} H_k\right)^{-1} \\ P_k &= \left(\lambda P_{k-1}^{-1} + H_k^T R_k^{-1} H_k\right)^{-1} \end{aligned} \tag{2.6}$$

the complete derivation of (2.6) can be found in [6].

Summary of the algorithm

- Initialize the estimator by setting $\hat{\theta}_0$ to our initial guess of the parameters, and by setting P_0 to the covariance of the initial guess. If we know perfectly the parameters before any measurements, we set $P_0 = 0$, if no any prior knowledge is available, we set $P_0 = \infty I$.

- for $k = 1, 2, \ldots$
  - obtain a new measurement z_k.
  - compute the optimal filter gain

    $$K_k = \lambda^{-1} P_{k-1} H_k^T \left(R_k + \lambda^{-1} H_k^T P_{k-1} H_k\right)^{-1}$$

  - update the estimation

    $$\hat{\theta}_k = \hat{\theta}_{k-1} + K_k \left(z_k - H_k \hat{\theta}_{k-1}\right)$$

  - update the estimation covariance

    $$P_k = \left(\lambda P_{k-1}^{-1} + H_k^T R_k^{-1} H_k\right)^{-1}$$

2.2 Total Mass Estimation

In this section we show how to apply the recursive least squares algorithm to estimate the vehicle's total mass.

2.2.1 Vehicle Calibration

To have a good estimation of the total mass, it is necessary to have information about the total thrust produced by the four propellers. To take into account possible degradation and non ideality of the propellers, we use the on-board calibrator to find the propeller factors defined as the ratio between the true thrust and the commanded thrust. More details about the calibration can be found in [9].

It is necessary to calibrate the vehicle without payload and with on-board mass parameter as close as possible to the true mass value. An offset in the on board mass parameter during calibration will lead to an offset in the estimated total mass when flying with additional payload.

2.2.2 Mass Estimation

From (1.2), neglecting the centrifugal and Coriolis terms, we can write the vehicle acceleration as follows

$$\underbrace{\begin{pmatrix} x_{acc} \\ y_{acc} \\ z_{acc} + g \end{pmatrix}}_{z_k} = \underbrace{\begin{pmatrix} U_1\left(s_\psi s_\phi + c_\psi s_\theta c_\phi\right) \\ U_1\left(-c_\psi s_\phi + s_\psi s_\theta c_\phi\right) \\ U_1 c_\theta c_\phi \end{pmatrix}}_{H_k} \underbrace{1/m_T}_{\theta} \tag{2.7}$$

In Wquation (2.7) we omitted the obvious dependency on time k, from now on we will omit the dependency on time. The measurement vector $z_k = \begin{pmatrix} x_{acc} & y_{acc} & z_{acc} + g \end{pmatrix}^T$ can be obtained from the VICON through numerical derivation or from the on-board accelerometers. Note that the on-board accelerometers provide measurements in the vehicle frame, so it is necessary to transform it to the inertial frame as follows

$$z_k^O = R_O^V z_k^V \tag{2.8}$$

The collective thrust U_1 is calculated as a function of the motors command as follows

$$\begin{aligned} F_i &= \alpha_2 \left(\frac{\mathsf{MtrCmd}_i}{200}\right)^2 + \alpha_1 \left(\frac{\mathsf{MtrCmd}_i}{200}\right) + \alpha_0 \\ U_1 &= \sum_{i=1}^{4} F_i \end{aligned} \tag{2.9}$$

where MtrCmd_i is the i-th motor command, F_i is the i-th motor thrust and $\alpha_2, \alpha_1, \alpha_0$ are constants.

The vehicle attitude used to calculate H_k is obtained from a previously existing estimator. Equation (2.7) doesn't assume any external disturbance and it shows good performance in hovering, but it has poor performance in forward flying. In [10], an augmented model with additional disturbances is presented as follows

$$\underbrace{\begin{pmatrix} x_{acc} \\ y_{acc} \\ z_{acc} + g \end{pmatrix}}_{z_k} = \underbrace{\begin{pmatrix} U_1\left(s_\psi s_\phi + c_\psi s_\theta c_\phi\right) & 1 & 0 \\ U_1\left(-c_\psi s_\phi + s_\psi s_\theta c_\phi\right) & 0 & 1 \\ U_1 c_\theta c_\phi & 0 & 0 \end{pmatrix}}_{H_k} \underbrace{\begin{pmatrix} 1/m_T \\ f_x \\ f_y \end{pmatrix}}_{\theta} \tag{2.10}$$

where f_x and f_y are the mass normalized disturbance forces on x and y axis respectively.
We update our estimator every time a new feedback packet is available from the vehicle, more details about the feedback and the communication standard used in the Flying Machine Arena can be found in [9].

Summary

- Initialize the estimator by setting the initial guess about the total mass and the normalized external disturbances on x and y axes. Moreover, we set the initial covariance P_0. In our case, we choose to fix the measurement noise covariance R and we choose an appropriate forgetting factor λ.

- For every feedback packet available, do the following:
  - obtain a new measurement $\begin{pmatrix} x_{acc} & y_{acc} & z_{acc}+g \end{pmatrix}^T$. If the measurement are obtained from the on-board accelerometers, we must transform it according to equation (2.8).
  - calculate U_1 according to Equation (2.9).
  - calculate H_k as shown in Equation (2.10).
  - compute the optimal filter gain

$$K_k = \lambda^{-1} P_{k-1} H_k^T \left(R_k + \lambda^{-1} H_k^T P_{k-1} H_k\right)^{-1}$$

  - update the previous estimation

$$\hat{\theta}_k = \hat{\theta}_{k-1} + K_k \left(z_k - H_k \hat{\theta}_{k-1}\right)$$

  - update the estimation covariance

$$P_k = \left(\lambda P_{k-1}^{-1} + H_k^T R_k^{-1} H_k\right)^{-1}$$

  - set $P_{k-1} = P_k$ and $\hat{\theta}_{k-1} = \hat{\theta}_k$

2.3 Center of Mass Offset Estimation

Beside the total mass, we want to estimate the center of mass (CoM) offset. When the quadrocopter grasps an object, due to non-ideality in the grasping and non uniformity in the mass distribution of the payload, we may have an offset in the center of mass. By estimating that offset, and providing this information to the controller, we can achieve better tracking performance as will be shown in Chapter 3.
Under hovering condition, we can rewrite Equation (1.5) as

$$\begin{pmatrix} U_2 \\ U_3 \\ U_4 \end{pmatrix} = \begin{pmatrix} y_{off} U_1 \\ -x_{off} U_1 \\ 0 \end{pmatrix} \tag{2.11}$$

considering U_i as constant over time, we can write the center of mass offset $\hat{x}_{off}$ and $\hat{y}_{off}$ as follows

$$\begin{aligned} \hat{x}_{off} &= -\frac{\sum_{\tau=1}^{N} U_3(\tau)}{\sum_{\tau=1}^{N} U_1(\tau)} \\ \hat{y}_{off} &= \frac{\sum_{\tau=1}^{N} U_2(\tau)}{\sum_{\tau=1}^{N} U_1(\tau)} \end{aligned} \tag{2.12}$$

where N is the averaging interval.

	EXP1	EXP2	EXP3
Theoretical CoM offset	$(0, 0.66)$	$(0, 1.41)$	$(0, 2.13)$
Estimated CoM offset	$(-0.1, 0.54)$	$(0.06, 1.36)$	$(-0.1, 2.17)$

Table 2.1: CoM offset experiments summary - all measurements are in cm

2.4 Experimental Results

The estimation algorithm has been implemented in C++ and tested in the Flying Machine Arena in ETH Zurich. Here we show the relevant experiments results.

2.4.1 Mass Estimation

Figure 2.1(a) shows how the forgetting factor may influence the estimation error, high forgetting factor will lead to small oscillation around the real mass, but it may take longer time to converge. On the other hand, small forgetting factor will not filter out the measurement noise, so it is necessary to tune this parameter based on the specific estimation problem we are dealing with, for instance, time varying mass estimation problem will require small forgetting factor to track correctly the real mass, while in forward flying as in Figure 2.1(b) it is necessary to have a high forgetting factor to reject the disturbance effects on the estimation. Figure 2.1(c) shows the percentual error for 5 different payloads, we repeated each experiment 10 times to verify repeatability of the method. Figure 2.1(d) shows the convergence rate of the algorithm for 2 different payloads, we can see that as the averaging time increases, the error converges to zero. A reasonable averaging time is $2.5 \sim 3$ seconds. Note that the convergence rate depends on the forgetting factor and the initialization of the estimator.

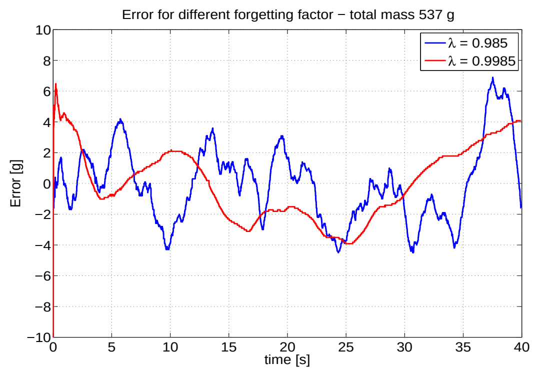
2.4.2 Center of Mass Offset Estimation

Here we present the experimental results of the CoM estimation. Attaching a payload on one side of the vehicle, as shown in Figure 2.2 , we calculate the theoretical CoM offset as follow:

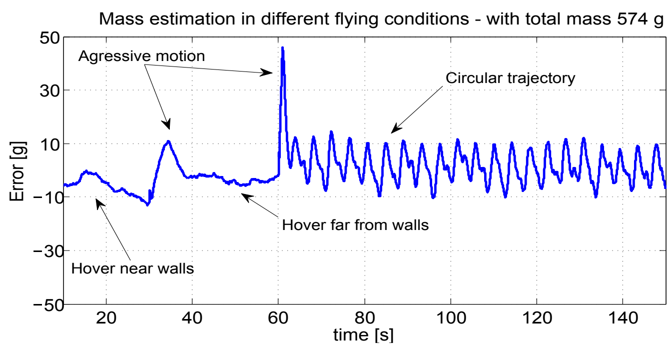
$$\begin{aligned} x_{off} &= \frac{m_L}{m_T} d_x \\ y_{off} &= \frac{m_L}{m_T} d_y \end{aligned} \tag{2.13}$$

where m_L is the mass of the payload, m_T is the total mass, d_x and d_y are the distance between the GCoM and the CoM of the payload on the x axis and y axis respectively.

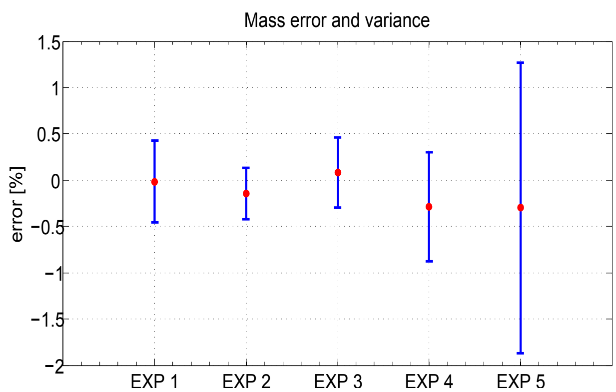
In Table 2.1 we show the theoretical CoM offset calculated by Equation (2.13) and the estimated CoM offset for 3 experiments. In Figure 2.3 we show the percentual error in the CoM offset estimation.



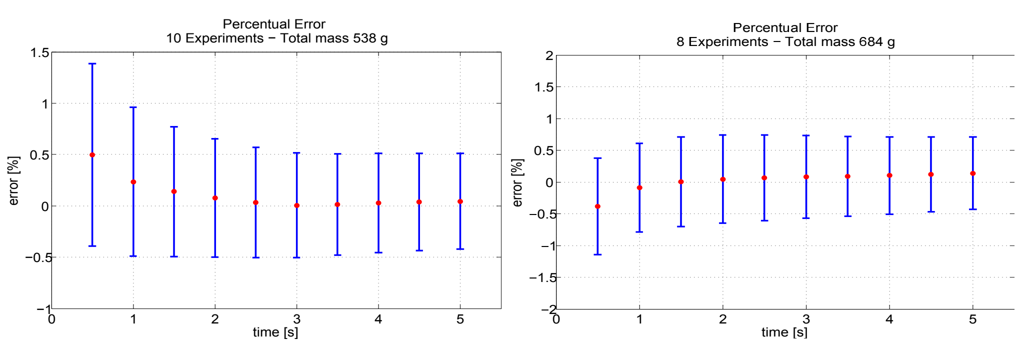
(a) Mass estimation error for different forgetting factors



(b) Mass estimation error in different flying conditions



(c) Mass percentual error and variance for 5 experiments. Each experiment is repeated 10 times to verify repeatability of the method. The total mass in the 5 experiments is $527g, 538g, 574g, 627g$ and $680g$ respectively



(d) Estimation percentual error and variance over time for 10 experiments.$\lambda = 0.995$

Figure 2.1: Experimental results

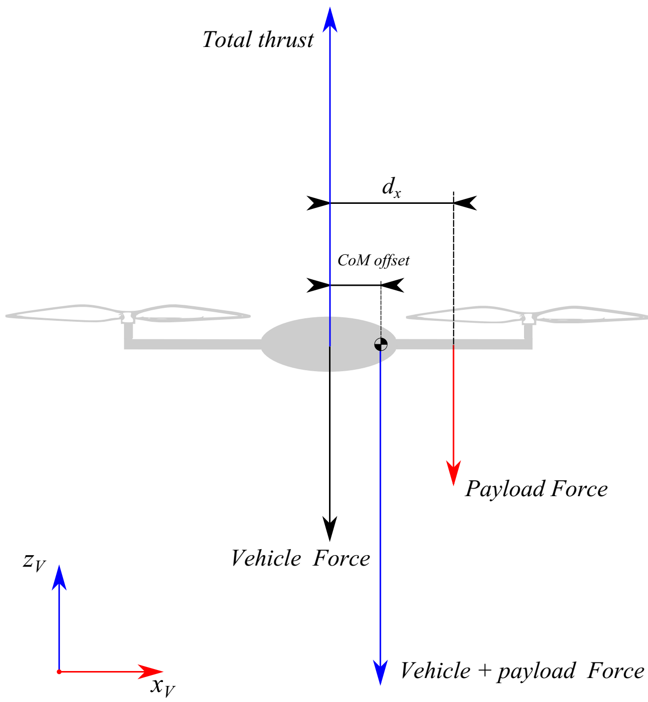


Figure 2.2: Ideal center of mass offset calculation

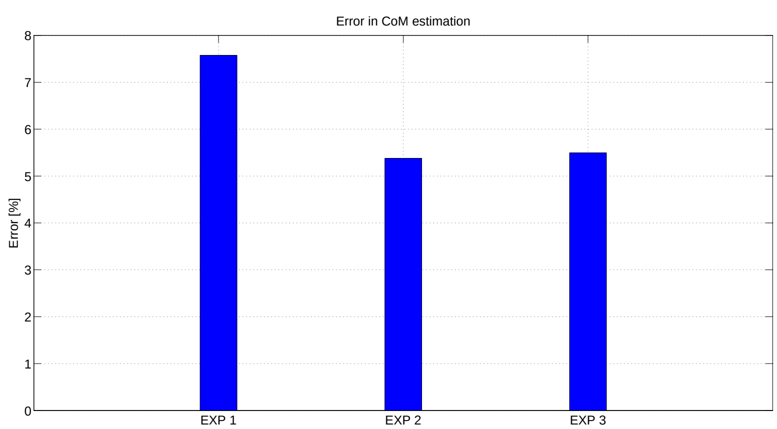


Figure 2.3: CoM estimation error

Chapter 3

Controller Extension

In this chapter we will present a controller structure that can compensate for the additional payload. The performance of the new controller structure is compared with the existing controller [9] using a random trajectory generated that uses the algorithm presented in [5].

3.1 Controller Structure

By elaborating and inverting Equation (1.5) we obtain the following controller structure

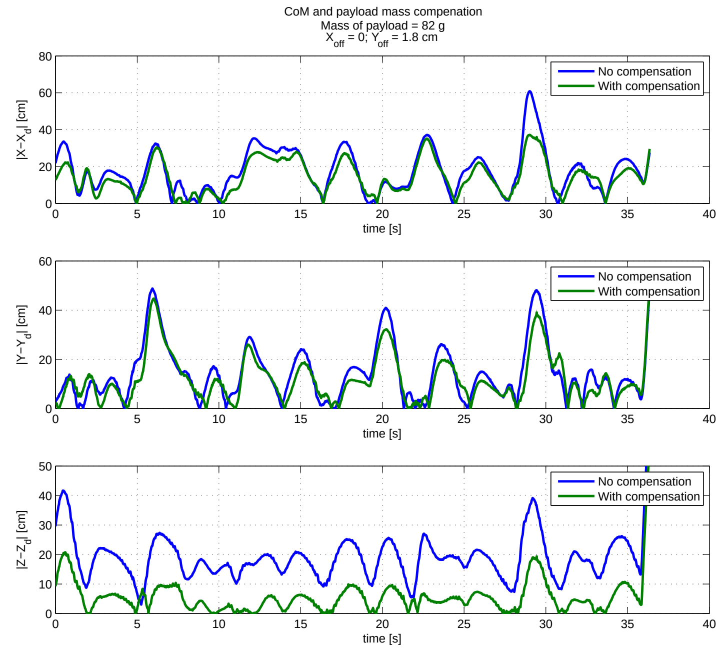
$$\begin{pmatrix} L\left(\check{F}_2-\check{F}_4\right) \\ L\left(\check{F}_3-\check{F}_1\right) \\ \bar{k}\left(\check{F}_1-\check{F}_2+\check{F}_3-\check{F}_4\right) \end{pmatrix} = \mathbf{J}\begin{pmatrix} \frac{1}{\tau_{pq}}\left(\check{p}-p\right) \\ \frac{1}{\tau_{pq}}\left(\check{q}-q\right) \\ \frac{1}{\tau_{r}}\left(\check{r}-r\right) \end{pmatrix} + \omega\times\mathbf{J}\omega + \begin{pmatrix} \hat{m}_T\check{\tau}\hat{y}_{off} \\ -\hat{m}_T\check{\tau}\hat{x}_{off} \\ 0 \end{pmatrix} \tag{3.1}$$

$$\check{F}_1+\check{F}_2+\check{F}_3+\check{F}_4=\hat{m}_T\check{\tau} \tag{3.2}$$

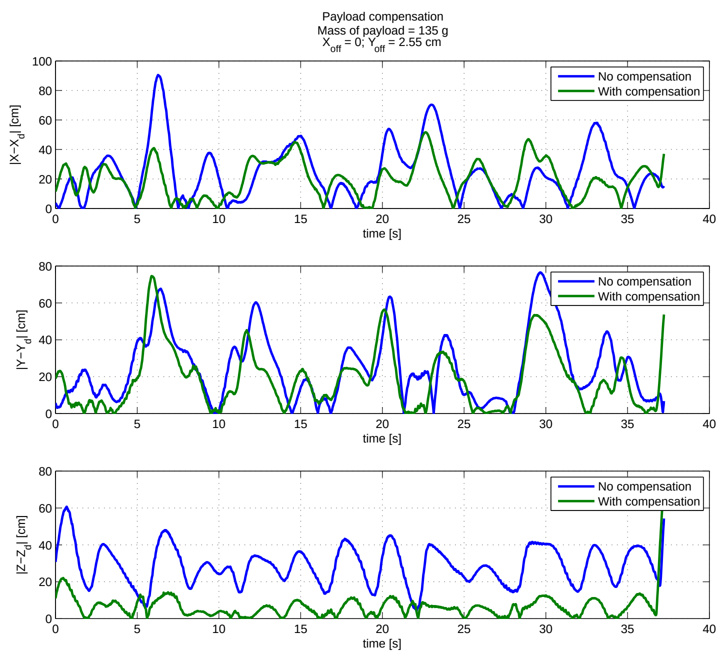
where $\check{p}, \check{q}$ and $\check{r}$ are the desired rotational rate around x_V, y_V and z_V respectively, $\check{\tau}$ is the desired mass normalized thrust, $\check{F}_1 \dots \check{F}_4$ are the desired motor thrust, τ_{pq}, τ_r are the closed loop time constant. $\hat{m}_T, \hat{x}_{off}$ and $\hat{y}_{off}$ are the estimated inertial parameters using the algorithm presented in Chapter 2.
It is straight forward to solve Equations (3.1) and (3.2) for the desired motors thrust $\check{F}_1 \dots \check{F}_4$.

3.2 Experimental Results

In this section we will show a comparison between the performance of the old controller presented in [9] and the extended controller expressed by Equations (3.1) and (3.2). Figure 3.1 shows the tracking error of both controllers on x, y and z axes for 2 different payloads. Table 3.1 shows the estimated center of mass and the tracking RMS error of both controllers. In these experiments, the trajectory is generated using collision-free trajectory generator presented in [5].



(a) Center of mass offset compensation and payload mass compensation - $m_T = 573g$



(b) Center of mass offset compensation and payload mass compensation - $m_T = 628g$

Figure 3.1: Controller comparison

	EXP1	**EXP2**
Payload mass	82 g	135 g
Estimated CoM offset	$(-0.1, 1.8)$	$(0.06, 2.55)$
Tracking RMS error with standard controller	19.6 cm	31.8 cm
Tracking RMS error with extended controller	14.2 cm	20.1 cm

Table 3.1: Controllers comparison

Chapter 4

Feasibility of Trajectories with Payload

In this chapter we show how a payload attached to the quadrocopter may affect the feasibility of a given trajectory. We try to answer the following question: Given a trajectory characterized by $X(t) \in \mathbb{R}^3$ and yaw profile $\psi(t)$, a payload rigidly attached to the quadrocopter characterized by mass and center of mass offset, is the trajectory feasible?

4.1 Trajectory feasibility

In this section, we extend the work presented in [3], our goal is to test trajectory feasibility with respect to vehicle physical constraints. First, we will show how to construct rotational rate and rotational matrix from the trajectory. Then we construct the single motor thrust. Finally we present the vehicle physical constraints.

4.1.1 Derivation of rotational rates from trajectory

Construction of rotational rates

Let's define the thrust vector as

$$f = R_O^V \begin{pmatrix} 0 \\ 0 \\ U_1/mT \end{pmatrix} = \begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} \tag{4.1}$$

and the normalized thrust vector as

$$\bar{f} = \frac{f}{||f||} = \frac{f}{U_1/m_T} = R_O^V \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{4.2}$$

differentiating (4.2) with respect to time and pre-multiplying by R_V^O we obtain

$$R_V^O \dot{R}_O^V \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = R_V^O \dot{\bar{f}} \tag{4.3}$$

moreover we have that (see [13])

$$R_V^O \dot{R}_O^V = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix} \tag{4.4}$$

From Equations (4.3) and (4.4) we obtain

$$\begin{pmatrix} \omega_y \\ -\omega_x \\ 0 \end{pmatrix} = \left(R_O^V\right)^T \dot{\bar{f}} \tag{4.5}$$

$\dot{\bar{f}}$ can be obtained by differentiating Equation (4.2) giving the following expression

$$\dot{\bar{f}} = \frac{\dddot{X}}{||f||} - \frac{f^T \dddot{X} f}{||f||^3} \tag{4.6}$$

where $X = \begin{pmatrix} x & y & z \end{pmatrix}^T$
To construct ω_z we can write the rotational rates in the inertial frame as follows

$$\begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \begin{pmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & c_\theta s_\phi \\ 0 & -s_\phi & c_\theta c_\phi \end{pmatrix} \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} \tag{4.7}$$

from Equation (4.7) we can write ω_z as

$$\omega_z = -\dot{\theta} \sin\phi + \dot{\psi} \cos\theta \cos\phi \tag{4.8}$$

and $\dot{\theta}$ as

$$\dot{\theta} = \frac{\omega_y - \dot{\psi} \cos\theta \sin\phi}{\cos\phi} \tag{4.9}$$

substituting Equation (4.9) into (4.8) we get

$$\omega_z = \frac{\cos\theta \dot{\psi} - \sin\phi \omega_y}{\cos\phi} \tag{4.10}$$

Construction of rotational matrix

The rotational matrix R_O^V is given by

$$R_O^V = \begin{pmatrix} c_\psi c_\theta & -s_\psi c_\phi + c_\psi s_\theta s_\phi & s_\psi s_\phi + c_\psi s_\theta c_\phi \\ s_\psi c_\theta & c_\psi c_\phi + s_\psi s_\theta s_\phi & -c_\psi s_\phi + s_\psi s_\theta c_\phi \\ -s_\theta & c_\theta s_\theta & c_\theta c_\phi \end{pmatrix} \tag{4.11}$$

from (4.2) we observe that the normalized thrust vector is equal to the last column of the matrix R_O^V, hence

$$\bar{f} = \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} = \begin{pmatrix} s_\psi s_\phi + c_\psi s_\theta c_\phi \\ -c_\psi s_\phi + s_\psi s_\theta c_\phi \\ c_\theta c_\phi \end{pmatrix} \tag{4.12}$$

from Equation (4.12) we can find θ as

$$\theta = \text{atan2}\left(f_x \cos\psi + f_y \sin\psi, f_z\right) \tag{4.13}$$

knowing θ and ψ we can construct the first column of R_O^V, and from orthogonality of R_O^V we can construct the second column doing the cross product of the third column by the first column.
From Equations (4.5) and (4.10) we can derive the rotational rates given a trajectory $X(t)$ and yaw profile $\psi(t)$.

Derivation of single motor thrust

To find the single motor thrust, we have to solve the following equations for $F_1, \ldots, F_4$

$$\begin{pmatrix} -y_{off}F_1 + (L - y_{off})\,F_2 - y_{off}F_3 - (L + y_{off})\,F_4 \\ (x_{off} - L)\,F_1 + x_{off}F_2 + (L + x_{off})\,F_3 + x_{off}F_4 \\ \bar{k}\,(F_1 - F_2 + F_3 - F_4) \end{pmatrix} = \mathbf{J}\dot{\omega} + \omega \times \mathbf{J}\omega \tag{4.14}$$

$$F_1 + F_2 + F_3 + F_4 = m_T||f|| \tag{4.15}$$

Equation (4.14) is obtained by elaborating the rotational dynamics presented in Equation (1.5), while Equation (4.15) is obtained from the fact that $U_1/m_T = ||f||$.
Note that $\dot{\omega}$ is obtained by numerical derivation.

4.1.2 Vehicle Constraints

The physical constraints on the vehicle can be summarized as:

- Constraint on single motor force
  $F_{min} \leq F_i \leq F_{max}$
  with $F_{min} > 0$ because the propeller cannot be switched off during flight.
- Constraint on the maximum rotational rate
  $||\omega||_\infty \leq \omega_{max}$

Given a trajectory and a payload, if any of the physical constraint shown above is violated, the trajectory is not feasible. When the trajectory is not feasible, we have to generate another one with smaller maximum allowable jerk. In the next section we will consider time optimal trajectory, and we will show how the presence of a payload may affect the feasible set of trajectories.

4.2 Feasible set under the presence of payload

In this section, we consider time optimal trajectory (bang-bang) presented in details in [8], the trajectory can be characterized by the magnitude of the maximum allowable jerk $|u|$, the magnitude of the maximum acceleration $|a_{max}|$ and the switching times. See Figure 4.1 for an example of time optimal trajectory. The closed form solution for the switching time can be found on [2].

Let's consider the time optimal trajectory between two points, for simplicity, we fix the z axis and we assume that the magnitude of the maximum allowable jerk and the magnitude of the minimum allowable jerk are equal. We check the feasibility for a discrete grid of trajectories covering the set $\left\{(u_x, u_y) \in \mathbb{R}^2 | -40 \leq u_x \leq 40, -40 \leq u_y \leq 40\right\}$ to generate the feasible set. Note that the sign of u fixes in which direction we are going to move, for instance, if $u_x > 0$ we will have $x(t_f) > x(t_0)$, on the other hand, if $u_x < 0$ we will have $x(t_f) < x(t_0)$.
In the following analysis we consider the constraints shown in table 4.1. In particular, we fix $\omega_z = 0$ to minimize power consumption.

Results

Here we present the analysis results. In figure 4.2 we show the influence of different payloads attached to the vehicle, without any center of mass offset. The additional payload will reduce the feasible set as expected intuitively.

In Figure 4.3 we show the influence of the center of mass offset on the feasible set. The payload mass is fixed ($m_L = 100\,\mathrm{g}$) while the center of mass is changed. We can see that the center of mass offset shrinks even more the feasible set, and it is not perfectly symmetric anymore. We can observe that moving in the direction where there is no center of mass offset is easier (higher jerk is allowable).

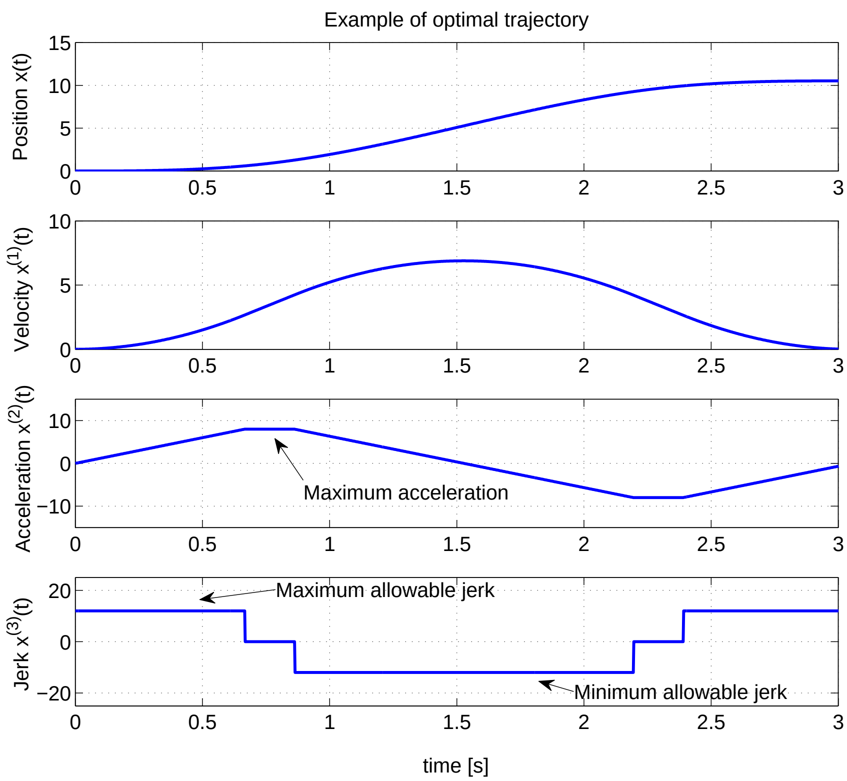


Figure 4.1: example of time optimal trajectory

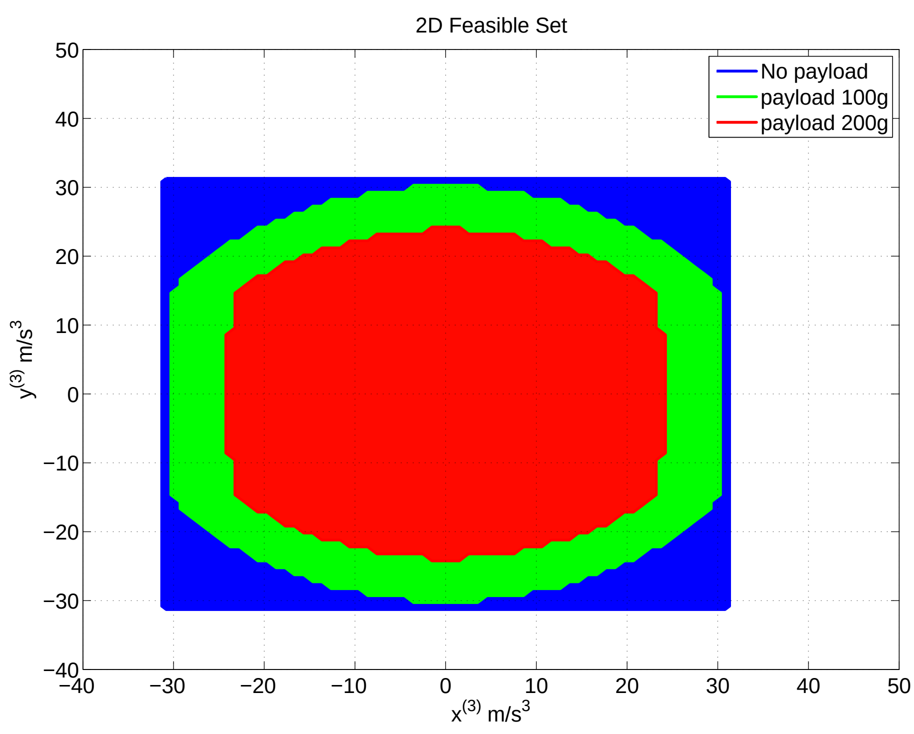


Figure 4.2: Feasible set for different payloads

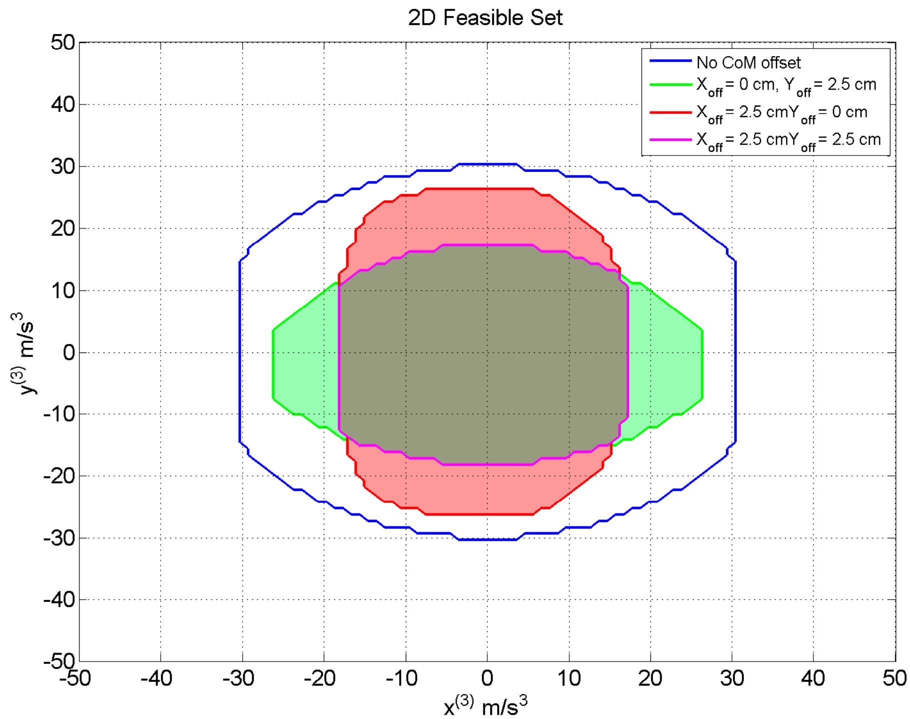


Figure 4.3: Feasible set for different center of mass offset

Constraint	Value
ω_{max}	$15 rad/s$
ω_z	0
acc_{max}	$12\,\mathrm{m/s^2}$
F_{max}	$3.9\,\mathrm{N}$
F_{min}	$0.15\,\mathrm{N}$

Table 4.1: Constraints

Conclusion

In this semester thesis, we showed how a payload attached to the quadrocopter may influence the tracking performance and the feasibility of trajectories. Moreover this estimator can be used for other purposes such as

- checking if we successfully picked or dropped a payload.
- update on-board parameters online for time varying payload, for instance in rope construction tasks. See [4].

The feasibility analysis gives an indication on the selection of maximum allowable jerk, and shows how the feasible set is deformed under the presence of additional payload.

Future work

The estimation can be improved in forward flying if we include an accurate noise model based on flight dynamics.
It might be interesting if we move the whole estimator on-board, and autonomously estimate inertial parameters and update them online.

Acknowledgments

I would like to thank my girlfriend Ni Ya and my parents for their constant support. I also would like to thank Federico Augugliaro for his valuable advice and suggestions during this project.
Finally I would like to express my special thanks to all the FMA team, and prof. Raffaello D'Andrea who supported this project.

Appendix A

Code

The code is implemented in a class called InertialParamEstimator, here we explain the main functions and we present an example showing how to use this class.

A.1 InertialParamEstimator Class

Public functions

```
InertialParamEstimator()
```

The constructor does not take any argument.

```
void Initialize(VehicleID ID, FMA::SettingsManager Settings)
```

This function initialize the quadrocopter ID, temporary variables, counters, and accumulators. Moreover it reads the necessary parameters from InertialParamEstimator.xml, the parameters will be explained in Section A.2

```
void UpdateCopilotStatus(const FMA::CP2::Copilot2StatusSet& statusSet)
```

This function is used in the copilot status callback to update the copilot status.

```
void Calibrate(float height)
```

This function will send calibration command to the copilot to a specified height. The calibration should be done without any payload to find the propellers factor.

```
void UpdateEstimation(const X3DFeedbackSet& fbs, shared_ptr<Estimator> est)
```

This function is the core of the estimation algorithm, it has to be called every time a new feedback packet is available. Moreover it takes a pointer to the vehicle estimator.

```
void StartEstimation(double EstimationInterval)
```

This function will run the estimator for EstimationInterval seconds.

```
double getEstimatedMass()
double getEstimatedMassAverage()
double GetCovariance()
```

The function getEstimatedMass will return the current mass estimation, while getEstimatedMassAverage will return the average of the estimated mass over the estimation period (EstimationInterval seconds). The function GetCovariance() will return the current estimation covariance.

```
BNUV <double,2> getCoMOffset()
BNUV <double,2> getCoMOffsetAverage()
double GetCoMOffset(int i)
```

The function getCoMOffset will return the current estimation of the center of mass offset, while getCoMOffsetAverage will return the average over the estimation interval. The function GetCoMOffset(i) will return x_{off} if $i = 1$ and y_{off} if $i = 2$.

```
void ResetEstimation()
```

This function will rest the estimator.

```
void StopEstimation()
```

This function will stop the estimator without resetting it.

```
void SendEstimatedMassToVehicle()
```

This function will send the estimated inertial parameters to the on-board controller.

```
bool FinishedEstimation()
```

This function will return True if the estimation is finished, otherwise it returns False.

A.2 Parameters

InertialParamEstimator.xml

Table A.1: Algorithm parameters in InertialParamEstimator.xml

Parameter	Value	Unit	Description
Theta0	$\begin{pmatrix} 0.5 & 0.1 & 0.1 \end{pmatrix}$	$\begin{pmatrix} \text{Kg} & \text{N} & \text{N} \end{pmatrix}$	The initial guess of total mass $m_T(0)$, x and y disturbances $F_x(0)$ and $F_y(0)$.
P0	$\begin{pmatrix} 10 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{pmatrix}$	[-]	The initial covariance matrix.
ForgettingFactor	0.985	[-]	The forgetting factor of the estimation algorithm.
R	$\begin{pmatrix} 50 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 50 \end{pmatrix}$	[-]	The measurement noise covariance.
Delayms	150	[ms]	Delay used in the code to increase reliability.
Filter_tau	0.02	[sec]	Time constant used in the numerical derivation to derive acceleration from ViCon.
minAccDt	0.005	[sec]	Minimum time step in numerical derivation, used to avoid numerical problems (avoid dividing by zero).

A.3 Example

In this section we show through an example how to use the InertialParamEstimator class.

```
...

InertialParamEstimator PayloadEst;          //Create InertialParamEstimator object
PayloadEst.Initialize(ID_QUAD1,settings);   //Initialization

...

//Define callbacks (copilot and feedback)

void ParamEstUpdateCopilotStatus(const FMA::CP2::Copilot2StatusSet& statusSet){
// This function is called every time a copilot status set is available
    PayloadEst.UpdateCopilotStatus(statusSet);
}

void ParamEstUpdateEstimation(const X3DFeedbackSet& fbs){
// This function is called every time a feedback packet is available
    PayloadEst.UpdateEstimation(fbs,VehicleEstimator);
}

...

//Here we add the callback
statusclient.AddCallback(ParamEstUpdateCopilotStatus);  //Copilot status update
fbClient.AddCallback(ParamEstUpdateEstimation);         //Feedback
```

```
...

//We use NoPayloadCalibration to switch between calibration and estimation
switch(NoPayloadCalibration){
        case 1:
            //Calibrate
            printf("Start Calibration\n");
            PayloadEst.Calibrate(2.5);
            break;
            //The vehicle will take off and calibrate the propellers factor at 2.5 m
        case 0:
            //Estimation
            printf("Estimation\n");
            ReqSet.timestamp = cmdTimer->ElapsedTicks();
            ReqSet.requests[ID_QUAD1] = FMA::CP2::Copilot2EntityRequest::Takeoff(2.5);
            crServer->Send(ReqSet); //Send take off request to 2.5 m
            printf("Sent Takeoff Request\n");

            //Wait 6 seconds - can be improved by monitoring the copilot status and waiting till the take off
            Sleep((DWORD)6000.0);


            PayloadEst.StartEstimation(10); //Start estimation for 10 seconds

            //If the estimation is not finished, monitor the mass, CoM offset and covariance
            double m, P;
            BNUV <double,2> CoMOffset;
            while(!PayloadEst.FinishedEstimation()){
                m = PayloadEst.getEstimatedMass();
                P = PayloadEst.GetCovariance();
                CoMOffset = LSE.getCoMOffset();

                ...

                //Monitor m,P, CoMOffset, for instance through debug server.

                ...

                Sleep((DWORD)1.0);  //Used to slow down the loop
                }

            //Finished estimation
            printf("Finished Estimation\n");

            /*Some checks can be done by the user to guarantee that the estimation didn't fail.
            (for instante check that the mass, CoMOffset and covariance are within certain range) */

            ...

            //If the estimation is ok, we can send it to the on-board controller

            PayloadEst.SendEstimatedMassToVehicle(); //Will send mass and CoMOffset.

            ...

            //Send landing request, or do something else!

            }
```

Bibliography

[1] http://www.idsc.ethz.ch/Research_DAndrea/Aerial_Construction/.

[2] http://www.idsc.ethz.ch/people/staff/hehn-m/.

[3] F. Augugliaro. Dancing quadrocopters: Trajectory generation, feasibility and user interface. Master's thesis, ETH Zurich, September 2011.

[4] F. Augugliaro, A. Mirjan, F. Gramazio, M. Kohler, and R. D'Andrea. Building tensile structures with flying machines. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[5] F. Augugliaro, A. P. Schoellig, and R. D'Andrea. Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages pp.1917–1922., 2012.

[6] M. Dahleh, A. Munther, A. Dahleh, and G. Verghese. *Lectures on Dynamic Systems and Control*, chapter Chapter 2 - Least Squares Estimation. Massachuasetts Institute of Technology, 2011.

[7] M. Hehn and R. D'Andrea. Quadrocopter trajectory generation and control. *IFAC World Congress*, 2011.

[8] M. Hehn and R. D'Andrea. Real-time trajectory generation for interception maneuvers with quadrocopters. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[9] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. D'Andrea. A platform for aerial robotics research and demonstration: the flying machine arena. *Mechatronics*, 2013.

[10] D. Mellinger, Q. Lindsey, M. Shomin, and V. Kumar. Design, modeling, estimation and control for aerial grasping and manipulation. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, Sept 2011.

[11] D. Mellinger, M. Shomin, N. Michael, and V. Kumar. Cooperative grasping and transport using multiple quadrotors. *Distributed Autonomous Robotic Systems, Lausanne, Switzerland.*, November. 2010.

[12] N. Michael, J. Fink, and V. Kumar. Cooperative manipulation and transportation with aerial robots. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.

[13] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Springer Verlag, 2009.

[14] J. Willmann, F. Augugliaro, T. Cadalbert, R. D'Andrea, F. Gramazio, and M. Kohler. Aerial robotic construction: Towards a new field of architectural research. *International journal of architectural computing*, VOL 10-3:pp. 439–460, 2012.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute for Dynamic Systems and Control
Prof. Dr. R. D'Andrea, Prof. Dr. L. Guzzella

Title of work:

Flying with Payloads

Thesis type and date:

Semester Thesis, May 2013

Supervision:

Federico Augugliaro
Prof. Raffaello D'Andrea

Student:

Name:	Mina Samir Fekry Kamel
E-mail:	fmina@ethz.ch
Legi-Nr.:	112-946-117
Semester:	2

Statement regarding plagiarism:

By signing this statement, I affirm that I have read and signed the Declaration of Originality, independently produced this paper, and adhered to the general practice of source citation in this subject-area.

Declaration of Originality:

`http://www.ethz.ch/faculty/exams/plagiarism/confirmation_en.pdf`

Zurich, 16. 7. 2014: ________________________________