Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Μηχανολογων Μηχανικων

Εργαστηριο Αυτοματου Ελεγχου και Ρυθμισεως Μηχανων

Διπλωματικη Εργασια

Αρπαγή Αντικειμένου από Εναέριο Ρομποτικό Όχημα

Αγγελόπουλος Αλέξανδρος

Επιβλέπων:

Κωνσταντίνος Κ. Κυριακόπουλος
Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2015

Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Μηχανολογων Μηχανικων

Εργαστηριο Αυτοματου Ελεγχου και Ρυθμισεως Μηχανων

Διπλωματικη Εργασια

Αρπαγή Αντικειμένου από Εναέριο Ρομποτικό Όχημα

Αγγελόπουλος Αλέξανδρος

Επιβλέπων:

Κωνσταντίνος Κ. Κυριακόπουλος
Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2015

Ευχαριστίες

Θα ήθελα να ευχαριστήσω όλα τα μέλη του εργαστηρίου για την συνεργασία που είχαμε και ιδιαίτερα τους Πάνο Μαράντο, Παναγιώτη Βλαντή, Γιώργο Ζωγόπουλο και Γιώργο Τζανέτο για την πάσης φύσεως συμβολή και υποστήριξή τους αλλά και τον κ. Κωνσταντίνο Κυριακόπουλο που μου έδωσε την ευκαιρία να εργαστώ υπό την επίβλεψή του σε ένα τόσο οργανωμένο και σύγχρονο εργαστήριο. Τέλος θα ήθελα να ευχαριστήσω την οικογένεια μου για την υποστήριξή της όλα αυτά τα χρόνια και τους ανθρώπους που μου έχουν σταθεί και έχουν συμβάλει θετικά στην διαμόρφωση της προσωπικότητάς μου.

Aggressive grasping by an aerial robotic multirotor

Angelopoulos Alexandros

Abstract

The objective of this thesis is to equip a commercial hexarotor vehicle with a robot hand and attempt to control its movement in an effort to grasp an object by tracking a $3D$ dynamic trajectory. Aiming for a bio-inspired approach, the platform will have to withstand the impact forces generated during grasping and deploy its strategy to minimize the outcome of the impact.

The impact analysis conducted, establishes the terminal conditions for the trajectory before grasping the object, while the findings of this analysis can help us predict the outcome. After grasping the object, the task is to hover in a location near the impact point. These tasks are performed by two individual non linear model predictive controllers which create and transmit attitude commands. The controller used for approaching the target has a variable time horizon because the final state of the vehicle is non-equilibrium.

AscTec Firefly, which is part of the Ascending Technologies research line, was set up and used for the experiments, receiving feedback only from onboard sensors. Being the first time that Firefly was used as an experimental platform in the Control Systems Laboratory of NTUA, extended analysis of the provided interfaces, quality of sensors, identification of model parameters and guidelines of proper usage have been determined and are included in the thesis.

The robot hand which was constructed is part of the OpenBionics project and has been redesigned to fit our experimental setup. The main advantage of the gripper's design is its ability to grasp safely a variety of different shapes while using a single motor keeping the extra payload low.

Contents

List of Figures

List of Tables

Chapter 1

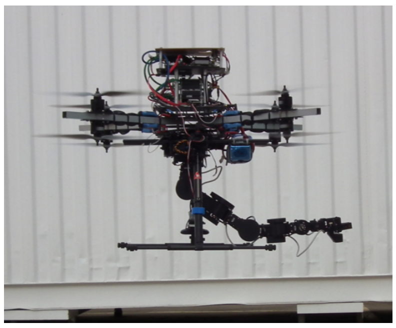
Introduction

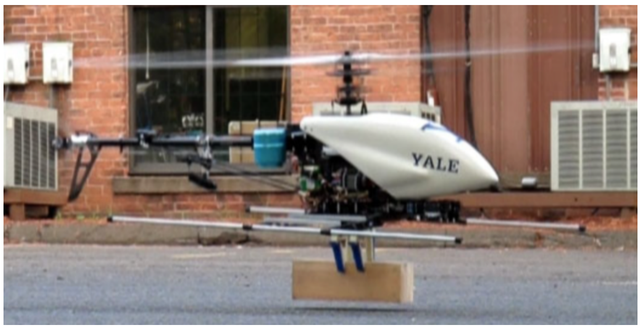
Aerial robotics is a fast-growing field of robotics and multirotor aircraft, are rapidly growing in popularity. In fact, quadrotor aerial robotic vehicles have become a standard platform for robotics research worldwide. They already have sufficient payload and flight endurance to support a number of indoor and outdoor applications, and the improvements of battery and other technology is rapidly increasing the scope for commercial opportunities. They are highly maneuverable and enable safe and low-cost experimentation in mapping, navigation, and control strategies for robots that move in three-dimensional space. This ability to move in $3D$ space brings new research challenges compared with the wheeled mobile robots that have driven mobile robotics research over the last decade. Small quadrotors have been demonstrated for exploring and mapping $3D$ environments; transporting, manipulating, and assembling objects; and acrobatic tricks such as juggling, balancing, and flips. Additional rotors can be added, leading to generalized N-rotor vehicles, to improve payload and reliability.

As a result, interaction with the environment has also been under research in recent years and can be applied to a wide range of applications like maintenance of power lines, construction in inaccessible sites or package delivery. Most of the already created prototypes use simple grasping mechanisms which are constructed and attached in a way that they do not change the dynamic behavior of the vehicle significantly (see [8], [13]). However there are some publications of more heavy duty experimental platforms which are equipped with multiple DOF robot manipulators and have investigated the tuning of the controllers depending on the different states of the aerial vehicle and manipulator (see [10]).

Aggressive maneuvers have also been accomplished aided by the usage of external cameras which are used for state feedback (see [9]) along with a combination of different controllers. The aforementioned systems (for example Vicon) are both expensive and not realistic because state estimation depends on external hardware which could not happen in a non structured environment.

Perhaps the most relevant published work on aggressive grasping is the one of Justin Thomas et al. ([22]) where an Ascending Technologies Hummingbird platform is used to grasp an object in high velocity. The trajectory generation is done offline while the analysis is two dimensional and there is a second controller to stabilize yaw and y direction. Moving in $z - x$ plane and utilizing a 2DOF gripper they were


(a) AMUSE [10]



(b) Yale [18]

Figure 1.1: Different aerial manipulator systems

able to grasp objects at a speed of $3[m/s]$ while using a VICON system for state feedback and object recognition.

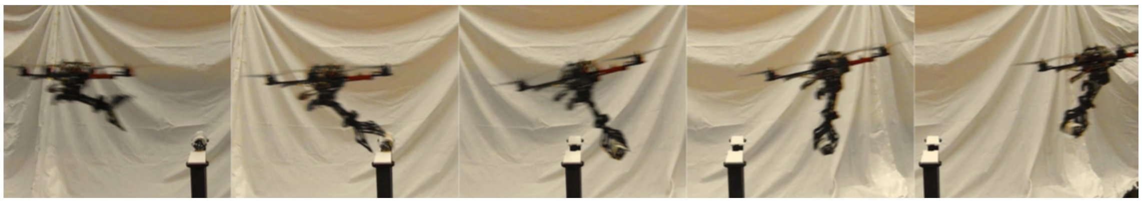

.

Figure 1.2: Aggressive grasping with a 2DOF gripper [22]

Chapter 2

Impact Mechanics Analysis

2.1 The problem

In order for the UAV to overcome the impact raised in the contact region of the gripper with the payload; it needs to be able to predict the magnitude of the resulting force. This prediction is crucial for the outcome of the impact because of the objective of this investigation. In other words due to our goal (grasping the target in high speed), one must consider that the resulting force can be so big that it can instantly result in system failure. That is mostly due to the fact that the impact interval is so small that we result in an open loop configuration. This means that the controllers will not have time to compensate the force outcome.

The first step in addressing the problem is to identify the parts which will be involved in the impact. This decision can be made after investigating the parameters that lead to a betterment of the force magnitude. These are:

- Relative Velocity of parts under impact
- Elasticity of the system

The problem suggests that one of the two bodies remains at rest while the UAV approaches in high speed. Given the gripper's structure we can see that our options sum up to a collision of the payload with either the finger structure or the finger's base structure. For the impact force's magnitude to be as low as possible we ought to choose the part of collision of the gripper keeping in mind that the final result is designated also by the relative velocity of the colliding bodies. For the aforementioned bodies relative velocities can be investigated right before the impact. One can clearly see that choosing the base of the gripper as the surface of collision is advantageous because apart from COM's velocity; the fingers velocity is designated also by their angular velocity resulting in a higher speed in the impact plane. Apart from that though, we are given the chance to choose a material that has properties (Elastic modulus) which help further the elongation of the time interval.

Clearly by selecting a material which prolongs the impact time interval and changes the geometry of the impact surface through time, grants us the ability to use smooth impact mechanics, rather than non-smooth (see [6]). Most researchers choose to use continuous-dynamics models of collision, such that the bodies deform during impact, and the collision dynamics are treated as continuous time dynamic phenomenons. The models used vary from case to case (properties of the materials being used) and can require FEA techniques which, although being more sophisticated and computationally heavier, yield similar results with linear models. The only drawback of linear models is that it is difficult to make a realistic estimate of the parameters that they have.

2.2 Maxwell model

To make a precise estimation of the exact impact magnitude we ought to consider the physical properties of our setup. The carefully chosen material covering the base of the gripper has viscoelastic properties. That means that the resulting impact force is not only derived from the displacement but also from the rate of deformation. The most comprehensive configuration that one can consider is that proposed by Maxwell. The Maxwell model uses a linear spring and a dashpot to describe the behavior of two colliding bodies. It is a simple model that help us take viscoelasticity into account which particularly in our case influences a lot the resulting force (see [27]).

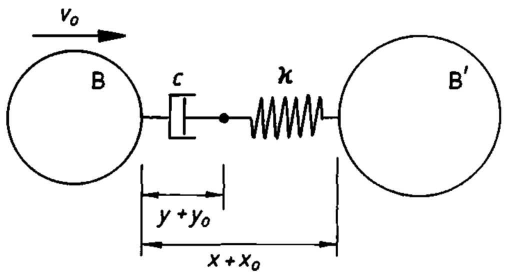


Figure 2.1: Collinear collision of bodies separated by a *Maxwell linear viscoelastic element*

Next the model is presented with its initial conditions.

$$F = -k \cdot x = -c \cdot \dot{y} \tag{2.2.0.1}$$
$$m^{-1} = M^{-1} + M'^{-1} \tag{2.2.0.2}$$
$$m \cdot \ddot{x} = -k \cdot (x - y) \tag{2.2.0.3}$$
$$z = x - y \tag{2.2.0.4}$$
$$\ddot{z} + 2\zeta\omega_0\dot{z} + \omega_0^2 z = 0 \tag{2.2.0.5}$$
$$x(0) = y(0) = z(0) = 0 \tag{2.2.0.6}$$
$$\dot{x}(0) = \dot{z}(0) = -v_0,\ \dot{y}(0) = 0 \tag{2.2.0.7}$$

Using the initial conditions we can find the solutions that suit us. The time interval of the phenomenon is equal to the compression time

$$z = -\omega_d^{-1} v_0 e^{-\zeta\omega_0 t} \sin(\omega_d t),\ \omega_d t \leq \pi \tag{2.2.0.8}$$
$$F = -kz \tag{2.2.0.9}$$

Using the above equations one can observe how the impact variables evolve with time. The phenomenon ends when the normal impulse is equal in magnitude with the initial momentum of relative motion mv_0 and separation occurs at time $t_f = \pi/\omega_d$.

2.3 Reality - Assumptions

To begin with, the Maxwell element uses a linear spring to model the behaviour of a solid part. This assumption is partially correct because the reality suggests that k changes through time. Nontheless an elastic solid can be viewed as a bundle of ideal springs. In other words the equivalent of Hooke's Law can be found in Young's Modulus.

$$\sigma = E\epsilon \tag{2.3.0.10}$$
$$\sigma dA = E\epsilon dA \tag{2.3.0.11}$$
$$F = EA\Delta x/L \tag{2.3.0.12}$$
$$k = EA/L \tag{2.3.0.13}$$

The above equation shows that k grows as the contact area A is becoming larger and becomes smaller as we increase the width of the material. Here we can see that with the real geometry (base being a flat surfuce, while object is a cylinder) the contact area is changing through time so that k is not constant. Apart from that different particles of the material have different displacements so that the resulting force F is different.

However the goal of this analysis is to produce easily a rough estimate of the force which will arise at the contact region and not its exact distribution. So by being able to know the geometry and mass of the object A_{max} can be calculated and used A_{max}/s for the calculation (where s is a constant derived from the object's geometry). As for the particles not having the same displacement for all different points of the surface we can again implement the same methodology to calculate z_{mean}.

This model assumes that the only material resulting in forces is the one on the robot hand's base. The aforementioned assumption is inconsistent because the robot hand itself and the object being grasped are supposed to be rigid. Nonetheless this is a worst case scenario and we can observe that the rigidity assumption is acceptable because $k_2 \ll k_1, k_3$. The only point that we need to pay attention to is that $z_{max} \leq L$ so that we don't reach unmodelled dynamics (k_1). This is easy to avoid by calculating first the maximum payload that our platform can carry and choose L accordingly. Another thing to keep in mind is that the model does not consider the retreat of the whole structure due to the torque created on the pitch plane.

Apart from that this is an 1D analysis resulting on a force parallel to the pitch plane. If the vehicle doesn't approach the target onto this plane then there will be a torque in the roll plane. To counteract this possibility the nonlinear predictive controller will have to keep the vehicle's trajectory onto the desired 2D plane right before the impact occurs.

As it has already been stated, the fact that we can select a specific material for the impact with the object can be advantageous. The choice of this material was made keeping in mind that we want relatively low elastic modulus. This would keep the calculated impact impulse close to the reality. Elastomers are polymeric materials which meet this criteria. Their mechanical behavior can be observed through the stress strain characteristic (see C at [3.1]). Silicone rubber was used at the base of the gripper. In order for our algorithm to work we need to have an estimate of its elastic modulus. Being an elastomer, silicone rubber has non-linear behavior but it can be considered linear during low compression. Researching for relative data sheets we have concluded that $E = 0.001[Gpa]$ (see [4]).

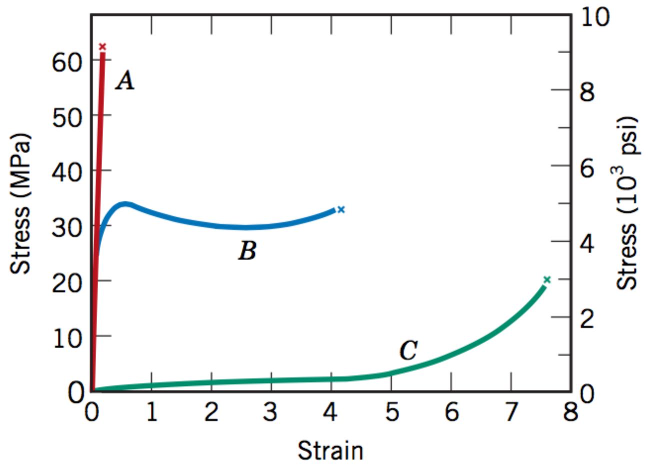


Figure 2.2: Stress-Strain behavior of elastomers (C) [see 26, ch. 15.9]

2.4 Strategy

The torque created in the pitch plane can almost instantly drive the MAV out of control if the object is heavy enough. That is mostly because, as mentioned before, during a part of that time interval we have an open loop configuration. So after identifying an estimate of that torque we can create a strategy of approach so that after the impact the system will remain intact. Specifically lets specify two discrete time instants: t_1 right before the impact and t_2 right after. We need to find the appropriate states θ, q, $\dot{q}$, $t = t_1$ to result at t_2 in acceptable states that can be further controlled.

Let T be the torque impulse resulting from the impact force F. In order to counteract this the thrusters should create an opposite torque impulse. This impulse can be generated through a long time interval before t_1. In other words apart from the torque that the thrusters generate during t_{12} which is constant due to open loop they can offer torque before t_1 when we have a closed loop. In conclusion, by roughly deciding the desired states after the impact we can calculate the exact states that we want at time t_1.

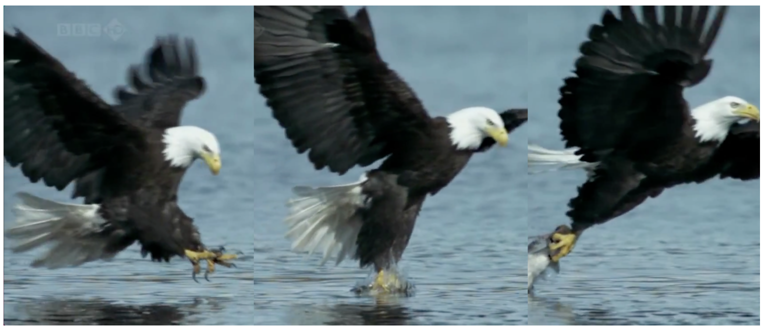

.

Figure 2.3: Bald eagle catching pray

This experimental setup is bio-inspired. Almost all birds catch their prey while flying and most of them seem to adjust their body in order to reduce their speed and to reject the torque that is being produced (see [3.1]). After they catch the prey and by feeling the payload that they need to carry, they plan their flight. As we can understand predators like the eagle depicted have a lot of experience and can thus plan how they will approach their prey.

Of course it is not possible for a robot using optical feedback to specify the mass of the object or to have a pre-evaluated plan of action. In order for our system to be robust we can use machine vision techniques to evaluate the volume V of the object. Then, we can choose densities ρ_{min}, ρ_{max} in which our strategy will work satisfactorily.

$$I_{yy}\ddot{\theta} = U_3 - T \tag{2.4.0.14}$$

The open loop system is governed by equation (4.2.2.1) during time interval t_{12}. By solving the double integrator using ρ_{min} and ρ_{max} we retrieve the different states that need to be implemented at time t_1. Then the states that will actually be implemented are calculated between the aforementioned cases. This methodology derives from the fact that in order for the system to be controllable after the impact the states ought to be in an acceptable range.

2.5 Robot hand

The robot hand used for the experiments (see [29]) is designed to provide the ability of stable grasping a vast range of objects while granting light weight and low cost of fabrication. Observing its design in Fig. 3.2 we can see its main attributes:

- *Bioinspired Design of Robot Fingers*: In spite the fact that it is a gripper, the design of the fingers is inspired from the human hand.

- *Compliant Flexure Joints and Soft Fingertips*: Deformation and adaptation of the geometry of the object being grasped can help to avoid fracture or failure. Although flexible, the materials being used are stiff enough to withstand considerable loads.

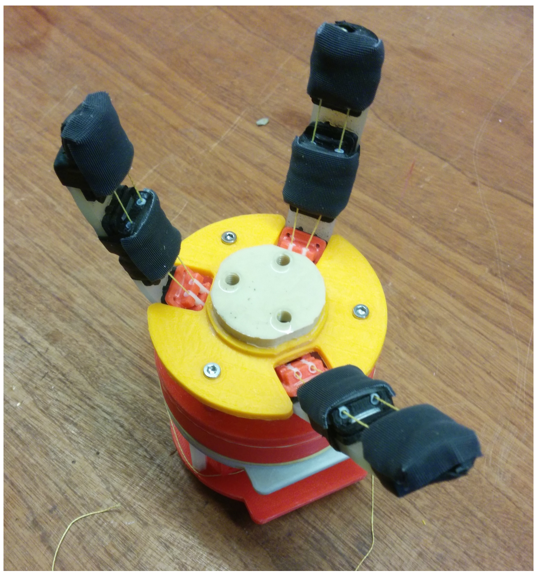

.

Figure 2.4: Gripper with silicone rubber base

- *Modular Fingers Basis with Multiple Slots*: The Gripper is very modular and can support many finger configurations. Keeping in mind that the application considered requires both lightweight and stability, a 3 finger configuration was used.

- *Cross-Servo Modular Actuator Basis*: All the fingers are actuated through a single servo motor keeping the weight low and the actuation simple. Our application demands very fast actuation and as a consequence a very fast motor was selected (see table 4.1)

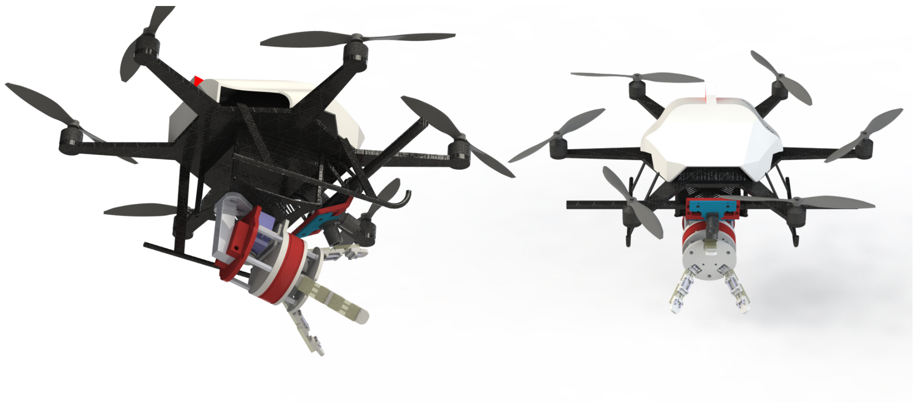
Weight (g)	29.5
Speed (6.0V sec/60)	0.07
Torque (6.0V oz-in)	34.7
Frequency (Hz)	$250 - 333$
Pulse Width Frequency	1520

Table 2.1: Savox SH-1257MG technical specifications

- *Disk-Shaped Differential Mechanism*: The differential mechanism allows for independent finger flexions, in case that one or multiple fingers have stopped moving, due to workspace constraints or in case that some fingers are already in contact with the object surface.

The gripper is mounted on Asctec Firefly through a minimalist base; the position of which is adjusted with three bolts directly attached under the vehicle (see Fig. 3.3). With the appropriate placement of the 178gr gripper we can achieve to keep the COM of the whole structure at the same point (in $x - y$ plane). This is necessary due to the fact that the low level controller is tuned for the original setup and it is not advised neither to overload the UAV nor to move it's body fixed frame which is suppose to coincide with the body principal axes of inertia.

Tip: Use the battery to bring the COM at the right place

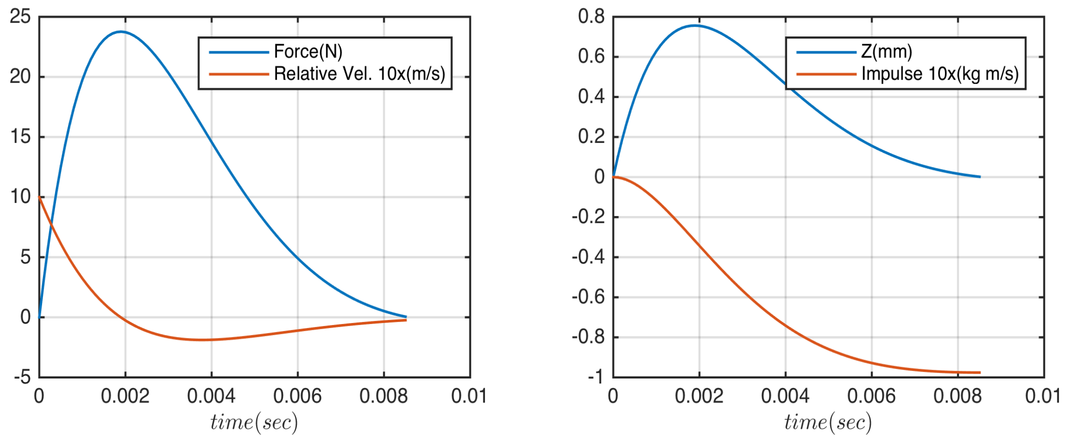


.

Figure 2.5: Firefly-Base-Robot hand assembly

2.6 Simulation Results

After the basic theory has been developed, some simulation results are being presented and discussed. There has been developed a program which takes all the aforementioned parameters into account and calculates the states which need to be achieved at time t_1. To do this first the force during the compression time is being calculated taking the impact mechanics into account.



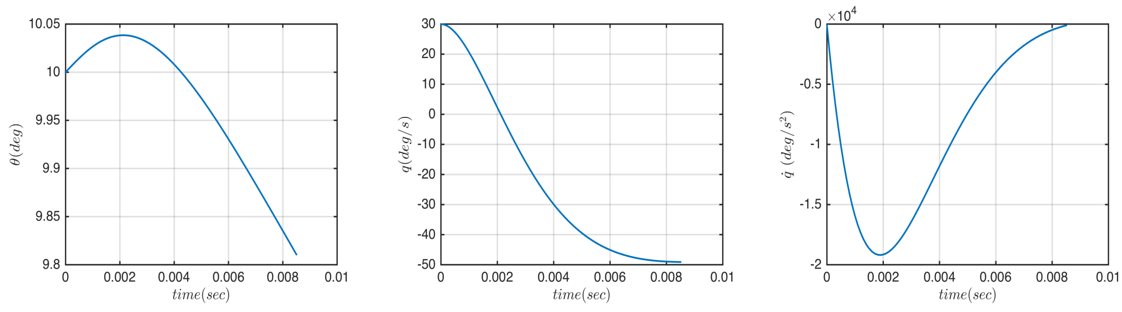
(a) Force and relative velocity (b) Compression and impulse created

Figure 2.6: Impact at 1.0 m/s with 100 g payload

Observing figure 2.6a we can see that the impact lasts only for $0.002sec$ before the two bodies begin to lose contact. As it will be shown in later sections the implemented non-linear controller works at $35Hz$ which confirms the open loop approach of the phenomenon. Apart from that through figure 2.6b it is shown that the compression of the elastomer on the base of the robot hand will be very small.

Through this result the thickness of the material mounted on the hand's base was chosen.

Now an estimation of the torque generated in the pitch plane can be produced. Then by using this torque as input to the system 4.2.2.1 solving numerically, the desired states can be calculated. In order to compute a viable solution we ought to create some attitude constraints for $\theta(t_2)$, $q(t_2)$, $\dot{q}(t_2)$ which would make some "easy" initial conditions for the second controller (driving the UAV away). The best case would be to have all initial conditions set to 0 so that would be our objective. Of course this terminal condition may not be viable for a heavy payload. The priority with which we would want the states to be zero is as follows $\dot{q}(t_2) > q(t_2) > \theta(t_2)$, $\dot{q}(t_2)$ being first because it is not directly controllable (not a state). Through experiments the maximum capabilities of the platform were tested to derive the $\dot{q}_{max}, q_{max}$ that could be achieved. A factor of safety was also used.



(a) θ (b) q (c) $\dot{q}$

Figure 2.7: Resulting states during impact simulation with 100gr payload

The results in Fig. 2.7 are an example of a possible solution for the given problem. So with $\theta_0 = 10°$, $q_0 = 0°/sec$ and $\dot{q} < 0°/sec^2$ we get fair results. To begin with, at the end of the phenomenon our platform has reached about 9.8° of pitch and with an angular velocity that cannot result in failure until the second controller starts to send commands. One can argue that with such light-weight load we can lower the initial condition for pitch. For observation reasons it will be kept at this value.

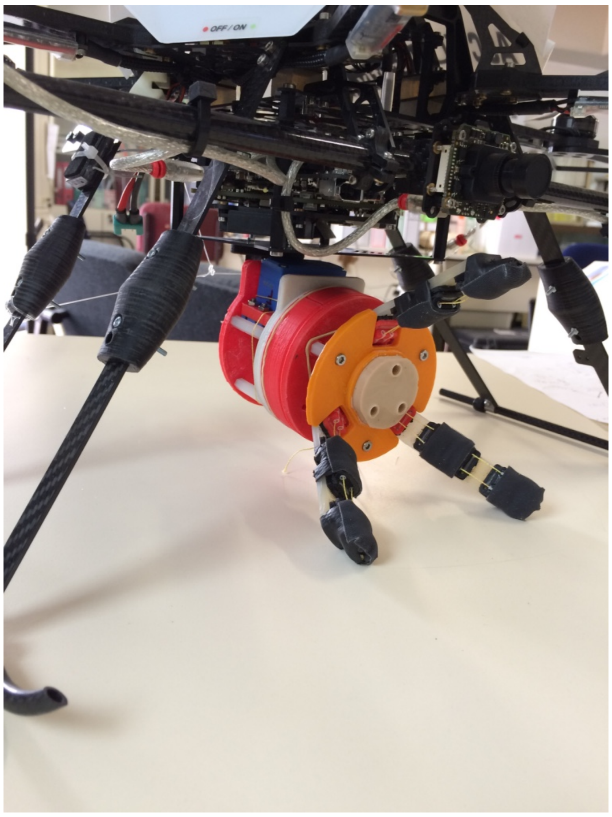


Figure 2.8: Experimental Platform assembly

Chapter 3

System Dynamics - Identification

3.1 System modelling

The nonlinear model predictive controller which was implemented in our setup needs a very detailed and accurate model to be able to work reliably. In most cases a linear model is used and can be sufficient if the platform is suppose to reach small angles (pitch, roll) (see [3]). In this case the system is suppose to have decoupled dynamics which is not the truth. In our experiment we want the platform to be as dexterous as possible. As a consequence a nonlinear model was used and identified which was derived from the generic 6 DOF rigid-body equation with the Newton-Euler formalism.

Let C_e be the inertial or earth frame which coincides with NWU (North-West-Up) convention and with C_{t1}. We also use the following frames most of which can be visualized in Fig.3.1:

- C_b: body fixed frame (front-left-up)
- C_{c1}: first camera frame
- C_{c2}: second camera frame
- C_{t1}: first target frame
- C_{t2}: second target frame
- C_o: frame of the object

In our case a hybrid system is used; composed of linear equations expressed in the inertial frame C_e and angular equations in body fixed coordinates C_b. The hexarotor generalized state space vector in this new frame is formed by the position of the body fixed frame relative to C_{t1} $P^e_{eb} = [x, y, z]^T$, the linear velocity vector $V_e = [\dot{x}, \dot{y}, \dot{z}]^T$, the Euler angles (ZYX convention) $\Phi = [\phi, \theta, \psi]^T$ and angular velocities $\omega = [p, q, r]$. Let ζ be the generalized velocity vector in the hybrid frame and ξ the generalized position vector:

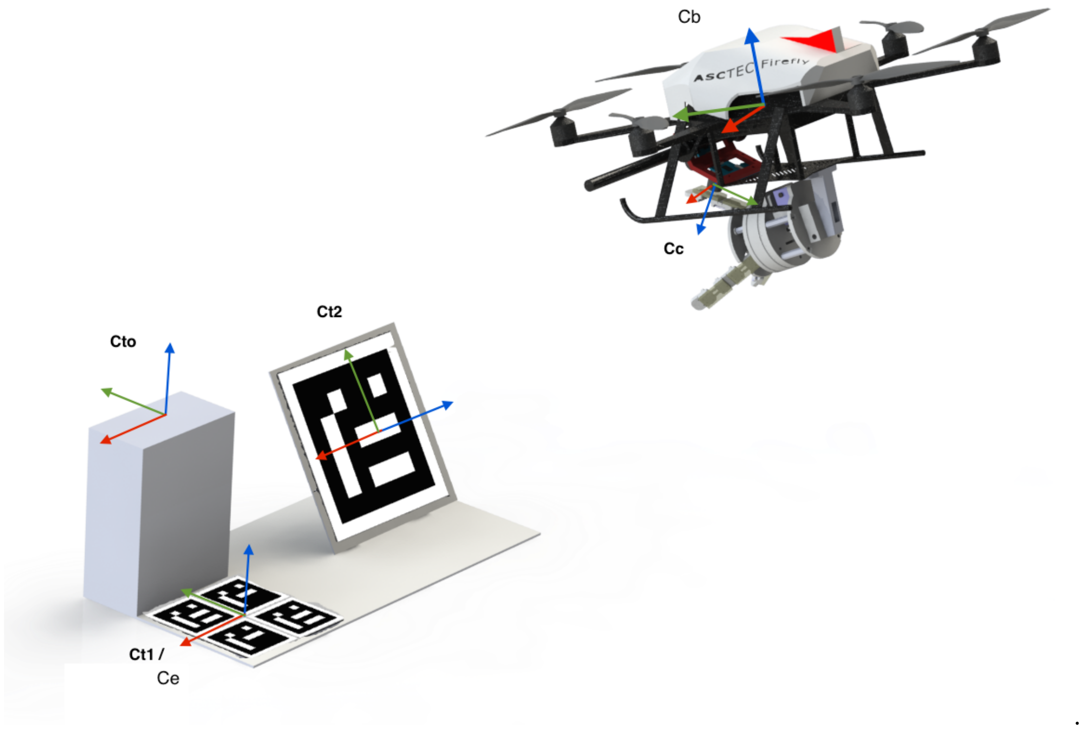


Figure 3.1: Coordinate frames used

$$\boldsymbol{\zeta} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & p & q & r \end{bmatrix}^T \tag{3.1.0.1}$$

$$\boldsymbol{\xi} = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^T \tag{3.1.0.2}$$

For our representation we also need the rotation matrix of the Euler angles ZYX $\Phi = [\phi, \theta, \psi]^T$ which can be derived after post-multiplying the three basic rotation matrices in the following order:

$$\begin{aligned} \boldsymbol{R(\Phi)} &= \boldsymbol{R}(\psi, z)\boldsymbol{R}(\theta, y)\boldsymbol{R}(\phi, x) \\ &= \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\phi & c\psi s\theta c\phi + s\psi s\phi \\ s\psi c\theta & s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \end{aligned} \tag{3.1.0.3}$$

Jacobian connecting $\dot{\Phi}$ and ω with the equation $\omega = E \cdot \dot{\Phi}$:

$$\boldsymbol{E(\Phi)} = \begin{bmatrix} 1 & 0 & -s\theta \\ 0 & c\phi & s\phi c\theta \\ 0 & -s\phi & c\phi c\theta \end{bmatrix} \tag{3.1.0.4}$$

where c stands for cosine and s for sine

For the dynamics of the system the Newton-Euler formulation was adopted. A more detailed explanation of the equations can be found in [5]. There are two basic assumptions made:

- *Origin of C_b*: The origin of the body-fixed frame is suppose to be coincident with the center of mass of the platform. If this was not taken into account, the equations would become more complicated.
- *Inertia matrix*: The body-fixed frame is also considered to have axes parallel to the body principal axes of inertia. This fact makes the inertial matrix diagonal which also simplifies the equations.

$$\boldsymbol{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{3.1.0.5}$$

In a matrix form the dynamic equations can be written as:

$$\boldsymbol{M}\dot{\boldsymbol{\zeta}} + \boldsymbol{C}(\boldsymbol{\zeta})\dot{\boldsymbol{\zeta}} = \boldsymbol{G} + \boldsymbol{E}(\boldsymbol{\xi})\Omega^2 \tag{3.1.0.6}$$

Where M is the system inertia matrix, C is the Coriolis-centripetal matrix, G defines the gravitational vector and E is the movement matrix where we can observe how thrust contributes to the movement of the platform in 3D space. The gyroscopic effects produced by the propeller rotation have been omitted due to their low contribution.

$$\boldsymbol{M} = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & Ixx & 0 & 0 \\ 0 & 0 & 0 & 0 & Iyy & 0 \\ 0 & 0 & 0 & 0 & 0 & Izz \end{bmatrix} \qquad \boldsymbol{E} = \begin{bmatrix} (c\psi s\theta c\phi + s\psi s\phi)U_1 \\ (s\psi s\theta c\phi - c\psi s\phi)U_1 \\ (c\theta c\phi)U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix}$$

$$\boldsymbol{C} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & Izzr & -Iyyq \\ 0 & 0 & 0 & -Izzr & 0 & Ixxp \\ 0 & 0 & 0 & Iyyq & -Ixxp & 0 \end{bmatrix} \qquad \boldsymbol{G} = \begin{bmatrix} 0 \\ 0 \\ -mg \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The inputs U_i derive from the speed of the motors and can be expressed according to the following equations for a hexarotor platform (see [11]):

$$\begin{aligned} U_1 &= T_1 + T_2 + T_3 + T_4 + T_5 + T_6 \\ U_2 &= U_\phi = 3l_{rg}(T_2 + T_3 - T_5 - T_6)/2 \\ U_3 &= U_\theta = l_{rg}(T_1 - T_4) + l_{rg}(T_2 - T_3 - T_5 + T_6)/2 \\ U_4 &= U_r = k_{fm}(T_1 - T_2 + T_3 - T_4 + T_5 - T_6) \end{aligned}$$

Where T_i are the thrust forces generated by the six propellers $T_i = k_\omega \cdot \Omega^2, i = 1, 2, ..., 6$, l_{rg} is the distance of each propeller to the COM in the $x - y$ plane and k_{fm} represents the force to moment scaling factor which is a positive constant.

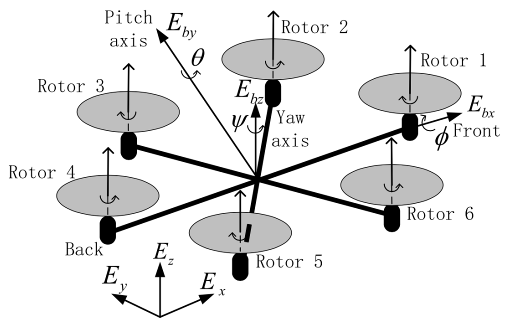


Figure 3.2: Schematic of a hexarotor

In our analysis we will send individual motor commands (this is done by the low level processor) but instead, use the attitude controller of AscTec Autopilot. By observing Fig.3.2 and the direction of rotation of each motor we can understand how the angular movements are being produced.

By rearranging eq.4.2.2.1 it is possible to isolate the derivative of the generalized velocity vector. Also by adding equations $\dot{\Phi} = E^{-1} \cdot \omega$ and $\frac{d(X)}{dt} = \dot{X}$ we can sum up the dynamic equations which govern the system. Note that air drag has also been added to the model both for linear and angular movement.

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \phi \\ \theta \\ \psi \\ p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ (c\psi\ s\theta\ c\phi + s\psi\ s\phi) \cdot \frac{U_1}{m} + \frac{F_{Dx}}{m} \\ (s\psi\ s\theta\ c\phi - c\psi\ s\phi) \cdot \frac{U_1}{m} + \frac{F_{Dy}}{m} \\ (c\theta\ c\phi) \cdot \frac{U_1}{m} + g + \frac{F_{Dz}}{m} \\ p + s\phi \frac{s\theta}{c\theta} q + c\phi \frac{s\theta}{c\theta} r \\ c\phi\ q - s\phi\ r \\ \frac{s\phi}{c\theta}\ q + \frac{c\phi}{c\theta}\ r \\ \frac{I_{yy}-I_{zz}}{I_{xx}}\ q\ r + \frac{U_2}{I_{xx}} + \frac{\tau_{Dx}}{I_{xx}} \\ \frac{I_{zz}-I_{xx}}{I_{yy}}\ p\ r + \frac{U_3}{I_{yy}} + \frac{\tau_{Dy}}{I_{yy}} \\ \frac{I_{xx}-I_{yy}}{I_{zz}}\ p\ q + \frac{U_4}{I_{zz}} + \frac{\tau_{Dz}}{I_{zz}} \end{bmatrix} \tag{3.1.0.7}$$

Where $F_D = -diag(d_u, d_v, d_w) \cdot R^T \cdot V_i$ and $\tau_D = -diag(d_p, d_q, d_r) \cdot \omega$ (both torques and forces are expressed in the body frame).

3.2 Low Level Controller

Until now we have only expressed the dynamics of the mechanical system. To be able to express the dynamics of the system as a whole, we have to know the structure of the low level controller. This controller runs on the Low Level Processor (LLP) and is in charge to transform our attitude commands (either from the RC or through serial connection) to individual motor speed commands. Since no exact description of the attitude controller exists and the control parameters are unknown, we ought to figure out a way to make an estimation of these gains.

Most of the times, simple PID controllers are used for individual control of each attitude state. So by receiving experimental data it is possible to check which structure best fits the data received by our platform. Below a diagram of how the LLP works is presented and in a later section the experimental results are demonstrated.

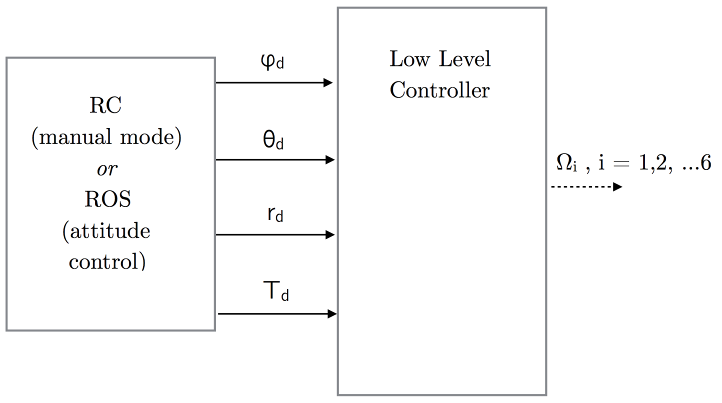


Figure 3.3: Low Level Controller I/O

In Fig.3.3 ϕ_d, θ_d are the desired roll and pitch angles, r_d is the desired yaw rate while T_d is the thrust. In manual RC mode full pitch and roll stick correspond to approximately 52° while a full yaw command will make the platform spin with $200°/s$. The thrust does not provide any height control and corresponds to values between 0 and 1; 0 being no thrust at all and 1 being max thrust ($36N$ for AscTec Firefly).

After testing many different control configurations the following control schemes where adopted to model the dynamics of the low level controller:

$$U_1 = K_T \; T_d \tag{3.2.0.8}$$
$$U_2 = I_{xx} \; (K_\phi \; (\phi_d - \phi) - K_p \; p) - (I_{yy} - I_{zz}) \; q \; r \tag{3.2.0.9}$$
$$U_3 = I_{yy} \; (K_\theta \; (\theta_d - \theta) - K_q \; q) - (I_{zz} - I_{xx}) \; p \; r \tag{3.2.0.10}$$
$$U_4 = I_{zz} \; K_r \; (r_d - r) - (I_{xx} - I_{yy}) \; p \; q \tag{3.2.0.11}$$

By substituting $U_i, i \in 1, 2, 3, 4$ in the Eq.4.2.2.1 we get the complete continuous nonlinear model of the system. We need to notice that the low level controller structure is helping to negate the excitation of terms like $(I_{zz} - I_{xx}) \; p \; r$ which would be unwanted. It is also worthwhile mentioning that the LLC negates almost all the inertial parameters of the equations.

The last step would be to derive the complete discrete time set of equations from the continuous model. To do that, the Forward Euler Method was used (see [28]) which is a first order technique. This means that if the frequency that the discrete model is being solved is not high enough then this could lead to big errors. To avoid this the method was tested with a simulator to determine the minimum frequency that the optimal control problem should be solved.

Given the initial state (t_n, y_n) and a step h the forward Euler method (FE) computes y_{n+1} as:

$$y_{n+1} = y_n + h\ f(y_n, t_n) \tag{3.2.0.12}$$

Using Eq.3.2.0.12 the final set of equations is created in the form $x_{k+1} = f(x_k, u_k)$:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ z_{k+1} \\ \dot{x}_{k+1} \\ \dot{y}_{k+1} \\ \dot{z}_{k+1} \\ \phi_{k+1} \\ \theta_{k+1} \\ \psi_{k+1} \\ p_{k+1} \\ q_{k+1} \\ r_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + h\ \dot{x}_k \\ y_k + h\ \dot{y}_k \\ z_k + h\ \dot{z}_k \\ \dot{x}_k + h\ ((c\psi_k\ s\theta_k\ c\phi_k + s\psi_k\ s\phi_k) \cdot \frac{K_T\ T_d}{m} + \frac{F_{Dx}}{m}) \\ \dot{y}_k + h\ ((s\psi_k\ s\theta_k\ c\phi_k - c\psi_k\ s\phi_k) \cdot \frac{K_T\ T_d}{m} + \frac{F_{Dy}}{m}) \\ \dot{z}_k + h\ ((c\theta_k\ c\phi_k) \cdot \frac{K_T\ T_d}{m} + g + \frac{F_{Dz}}{m}) \\ \phi_k + h\ (p_k + s\phi_k \frac{s\theta_k}{c\theta_k} q_k + c\phi_k \frac{s\theta_k}{c\theta_k} r) \\ \theta_k + h\ (c\phi_k\ q_k - s\phi_k\ r_k) \\ \psi_k + h\ (\frac{s\phi_k}{c\theta_k}\ q_k + \frac{c\phi_k}{c\theta_k}\ r_k) \\ p_k + h\ (K_\phi\ (\phi_d - \phi_k) - K_p\ p_k) \\ q_k + h\ (K_\theta\ (\theta_d - \theta_k) - K_q\ q_k) \\ r_k + h\ (K_r\ (r_d - r_k)) \end{bmatrix} \tag{3.2.0.13}$$

3.3 Parameters identification

Apart from the set of equations it is necessary to estimate all the parameters that are included in the model. These parameters consist of the ones concerning the Low Level Controller, aerodynamic and inertial constants. For the most of them a curve fitting technique on experimental data was used which minimizes the weighted norm of the prediction error. This method was conducted through Matlab and System Identification Toolbox, theory and results are presented later in this chapter. But apart from this powerful tool, common sense was of use. At the time when the experiments where conducted, no position or velocity sensors were available. As a result, other methods needed to be applied to calculate the remaining parameters.

To begin with, the thrust parameter K_T, being the simplest one, powers the vehicle with maximum thrust when the input $T_d = 1$. The maximum thrust that the vehicle can produce, according to Ascending Technologies is $36N$, which means that:

$$K_T = 36[N]$$

The aerodynamic constants were also an issue because of the linear parameters du, dv, dw. The angular aerodynamic parameters dp, dq, dr are unified with the gains of the LLC because they both scale with angular velocities p, q, r. By conducting an experiment we can see which are the maximum velocities of the platform by giving the appropriate RC commands. There is a time when the UAV cannot accelerate anymore due to the air drag. By using this data and the simulator which has been created we can test many du, dv, dw combinations (realistic values) and stop when the simulator has the exact same behavior. The results that we find from this trial and error technique can then be fed to the curve fitting algorithm to obtain the final values.

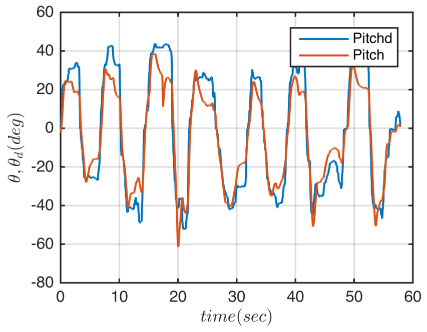
The Prediction Error Method (PEM) algorithm uses numerical optimization to minimize the cost function $V_N(G, H) = \sum_{t=1}^{N} e^2(t)$ where $e(t)$ is a vector and the difference between the measured output and the predicted output of the model. The subscript N indicates that the cost function is a function of the number of data samples and becomes more accurate for larger values of N (see[15]). As a consequence the data which are given as input to the algorithm ought to consist of many sampling instants and be representative (excitation of all platform dynamics). Identifying at once a lot of parameters would be difficult so a set of experiments were made for each individual movement in order to identify the parameters of each equation separately.

For example Fig.3.4 is the set of data which were received for the identification of K_θ, K_q while sampling at $50Hz$. From Fig.3.4a we can see that maximum attitude commands were given and reached. Also the platform reached velocities as high as $11m/s$ while in each experiment only one state was excited and at the end validation data was collected while all states were changing.

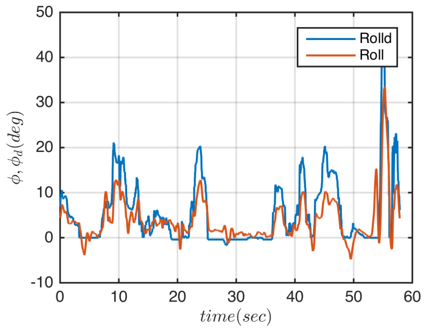
System Identification results		
Parameter	Value	Standard deviation
K_ϕ	237.6	14.1
K_θ	164.0	10.8
K_p	61.0	3.6
K_q	41.9	2.7
K_r	3.6	0.1
K_T	36.0	-
du	0.312	0.006
dv	0.269	0.007
dw	0.45	0.01

Table 3.1: Values of Eq.4.2.2.1 parameters

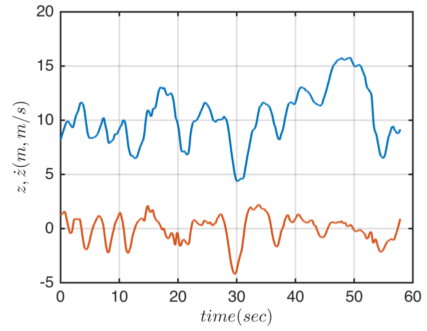
Observing figures 3.5, 3.6, 3.7 we can note that there was a difficulty to fit the angular velocities. Generally speaking experimental data can have a lot of noise and especially our inertial measurement unit (IMU), although automatically filtered, generates linear acceleration and angular velocity data with a lot of noise. GPS data, which are used to sense linear velocities, can also be problematic.



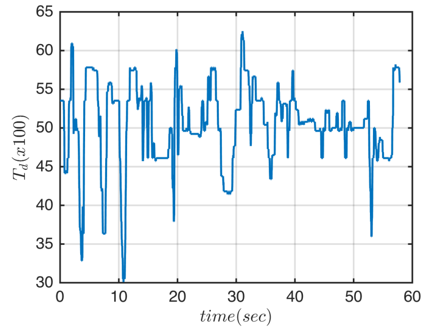
(a) Pitch



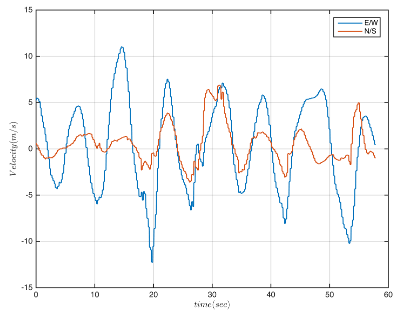
(b) Roll



(c) z and $\dot{z}$



(d) Thrust input (%)



(e) GPS velocities

Figure 3.4: Pitch experiment inputs and response

It is important to notice that these experiments were not conducted with the robot hand attached to the platform. Despite the fact that the hand does not change the position of the COM significantly it does create some changes. First of all, the LLC is tuned for the vehicle in its original configuration and as a consequence by adding the gripper the inertial matrix **M** is changing while equations 3.2.0.8 remain the same. As a result, the gains are absorbing some portion of the inertial objects of **M** and should be identified again. This also will occur when the object is grasped by the hexarotor. For this latter fact, it can be assumed that the object's inertia is a lot smaller and thus it can be ignored.

In ancient days two aviators procured to themselves wings. Daedalus flew safely through the middle air and was duly honoured on his landing. Icarus soared upwards to the sun till the wax melted which bound his wings and his flight ended in fiasco ... The classical authorities tell us, of course, that he was only 'doing a stunt'; but I prefer to think of him as the man who brought to light a serious constructional defect in the flying-machines of his day.

So, too, in science. Cautious Daedalus will apply his theories where he feels confident they will safely go; but by his excess of caution their hidden weaknesses remain undiscovered. Icarus will strain his theories to the breaking-point till the weak joints gape. For the mere adventure? Perhaps partly, that is human nature. But if he is destined not yet to reach the sun we may at least hope to learn from his journey some hints to build a better machine.

Sir Arthur Eddington, Stars and Atoms, 1927.

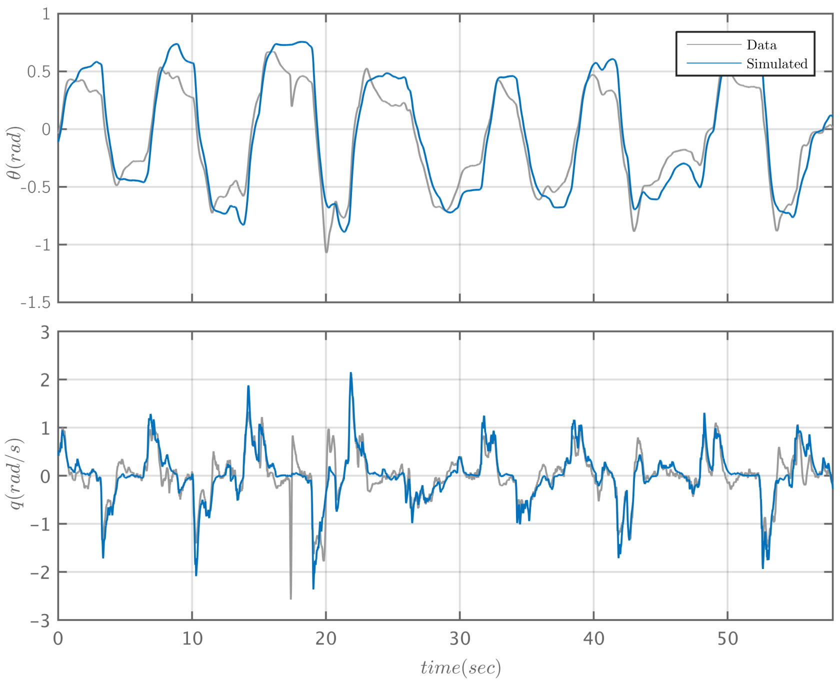


Figure 3.5: Pitch θ & q simulation comparison

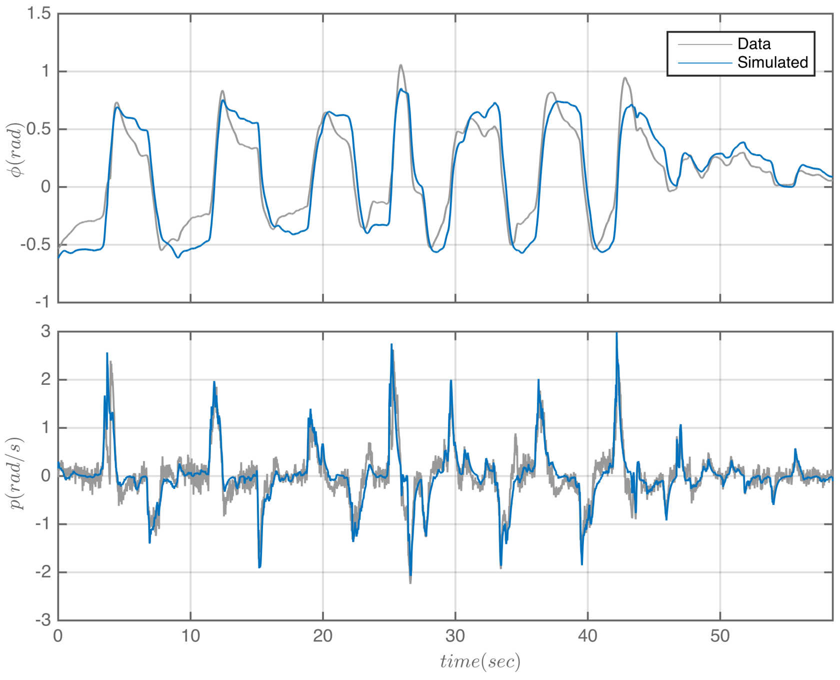
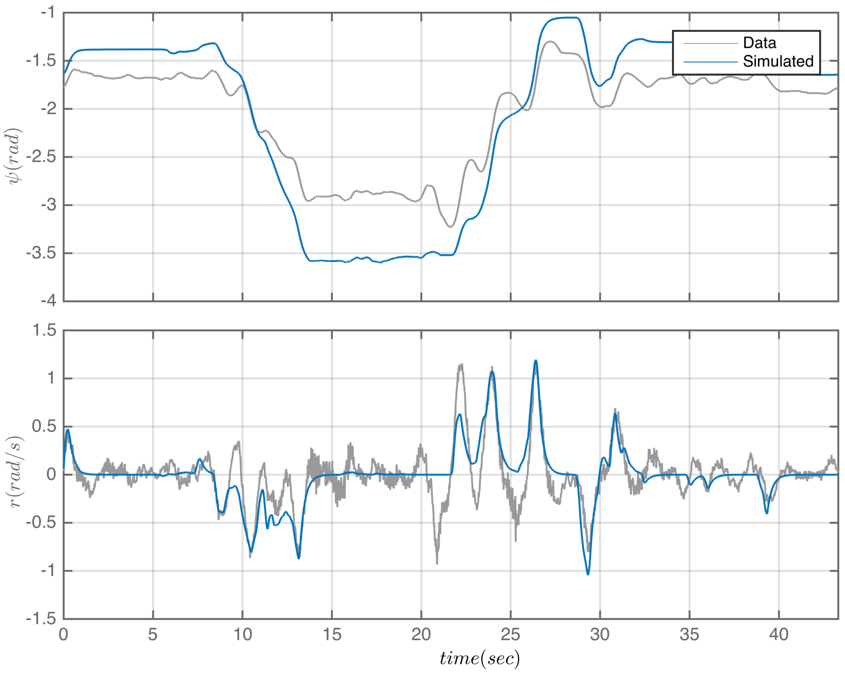
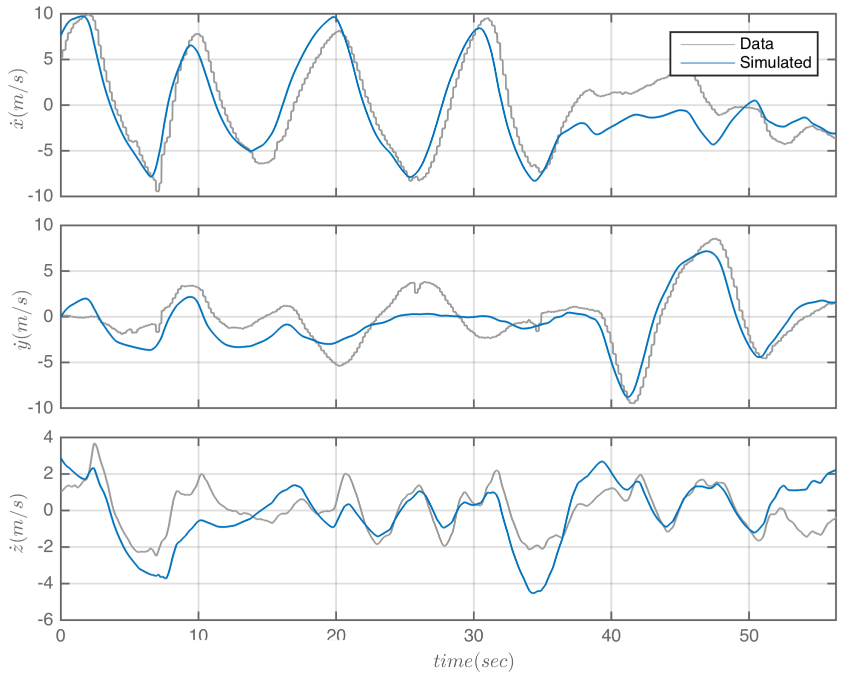


Figure 3.6: Roll ϕ & p simulation comparison



.

Figure 3.7: Yaw & r simulation comparison



.

Figure 3.8: $\dot{x}$, $\dot{y}$, $\dot{z}$ simulation comparison

Chapter 4

Platform

4.1 AscTec Firefly

The platform used for the experiments consists of a multirotor developed from Ascending Technologies. Firefly (see [20]) is a hexarotor suitable for Computer-Vision applications and powerful enough to facilitate the needs of our application. It's Flight Control Unit consists of a fast IMU (Inertial Measurements Unit) and two 32 bit 60 MHz microcontrollers for the flight control algorithms. Although there are two microcontrollers provided, only one is programmed and connected to all measurements units. The Low Level Processor (LLP), is connected to all the IMU sensors, which are three gyroscopes, three accelerometers, one air pressure sensor, magnetic field sensors and a GPS unit. The complete data fusion is performed on this processor at an update rate of 1kHz. It is worthwhile mentioning that all the flight control algorithms run on this processor (Attitude, Position, GPS). The High Level Processor (HLP) is free for custom code and connected to LLP via SPI. The LLP provides all measurements. To control the vehicle, the HLP can tap into the LLP control loop in three different levels by sending commands via SPI: Waypoint commands, attitude commands, or direct motor commands. Furthermore, the controllers are always running on the LLP so that a fallback in case of failures on the experimental code is possible in flight. The motor controllers are connected to the flight control unit via I2C and communicate at the overall update rate of 1 kHz.

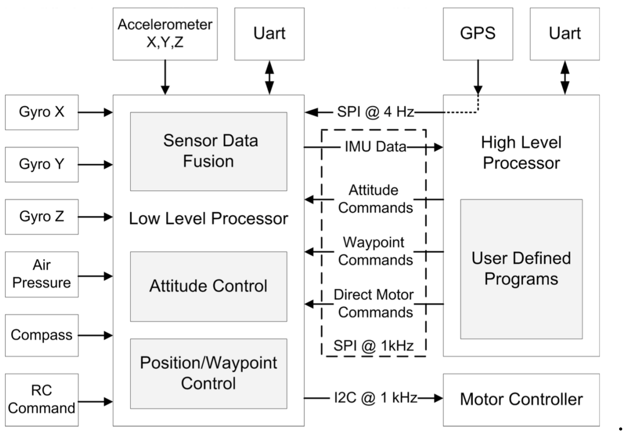


Figure 4.1: Electronic Architecture

Apart from the aforementioned microcontrollers, Firefly is equipped with an on-board computer further extending its computational capabilities by handling the heavy computational needs. Mastermind, as it is called, has an Intel® Core2Duo processor configured at $1,86GHz$ and offers many useful interfaces (serial, usb, GPIO, ethernet, WiFi etc. [21]) to connect external devices and equipment. The pre installed software is Ubuntu 12.04 and both ROS Hydro and Fuerte are installed. It is basically a ground computer designed for and mounted on the flight unit. There are some configured operating system downloads from the mastermind downloads section. However, the platform can be equipped with the latest ubuntu distribution if needed.

The computer is connected with the HLP and LLP via serial ports which makes the communication straight forward. The software used to receive all the available sensor data and send commands to the flight unit is asctec-mav-framework (see [1]), a ROS package which enables us to receive at fast update rates all data but also provides custom observers and a nonlinear position controller. For the needs of our experiment attitude commands were sent to the platform, and both observation and control were done externally.

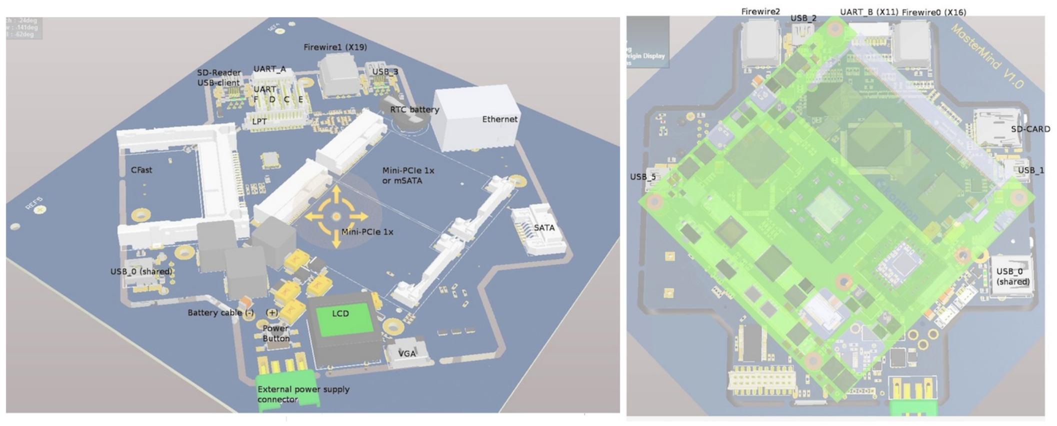


Figure 4.2: AscTec Mastermind

4.2 Available sensors

In order to get a fair understanding of the platform's feedback capabilities, experiments were conducted. As it is stated in the Ascending Technologies website on-board data fusion is being carried out so that the data produced are noise and bias free. The experimental setup is simple; the platform is kept stationary while the data produced are being recorded in a bagfile. The motivation behind providing this analysis is to help future researchers leverage the platform's capabilities and tackle issues that are critical for their needs.

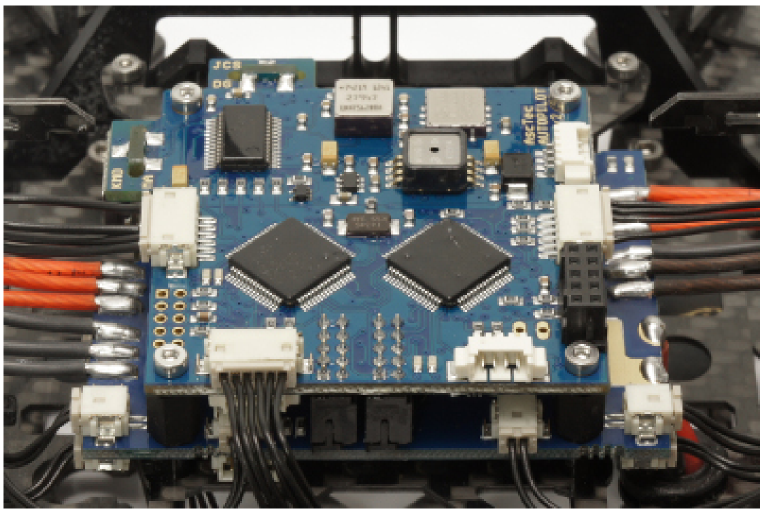

.

Figure 4.3: AscTec AutoPilot

4.2.1 Accelerometer

The accelerometers (in a strap down IMU configuration) measure the instantaneous linear acceleration of the body fixed frame of reference (B) due to exogeneous force. Accelerometers are highly susceptible to vibration and, mounted on a UAV, they require significant low-pass mechanical and/or electrical filtering to be usable. Most UAVs will incorporate an analogue anti-aliasing filter on a MEMS accelerometer before the signal is sampled. In the case of Firefly the data polled are said to be calibrated (according to the communication with LLP documentation) but we don't know if the firmware on the HLP polls the calibrated data structures or the raw. Nonetheless, the experiment conducted can grant us insight into this argument and to the quality of data that we can use.

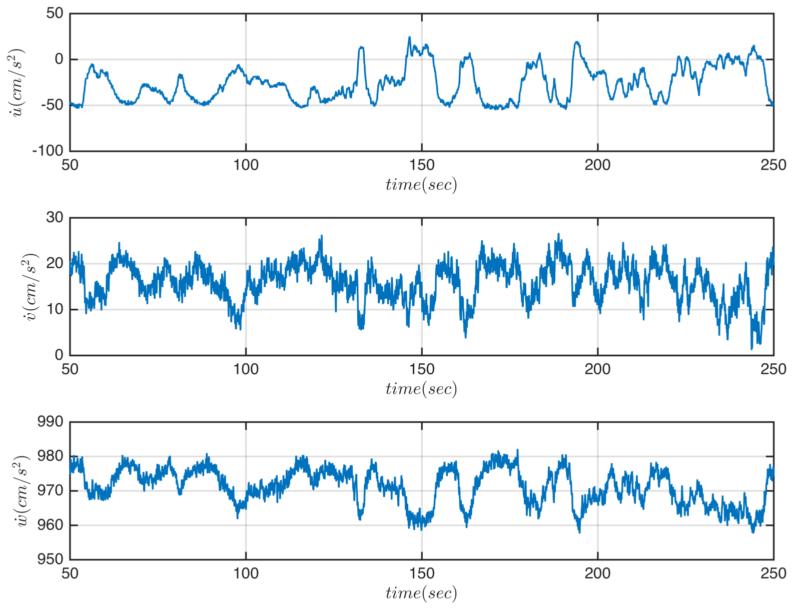
The response of an accelerometer can be modelled as

$$\alpha_{IMU} = R^T(\dot{v} - g\vec{z}) + b_\alpha + \eta_\alpha \in \{B\}, \tag{4.2.1.1}$$

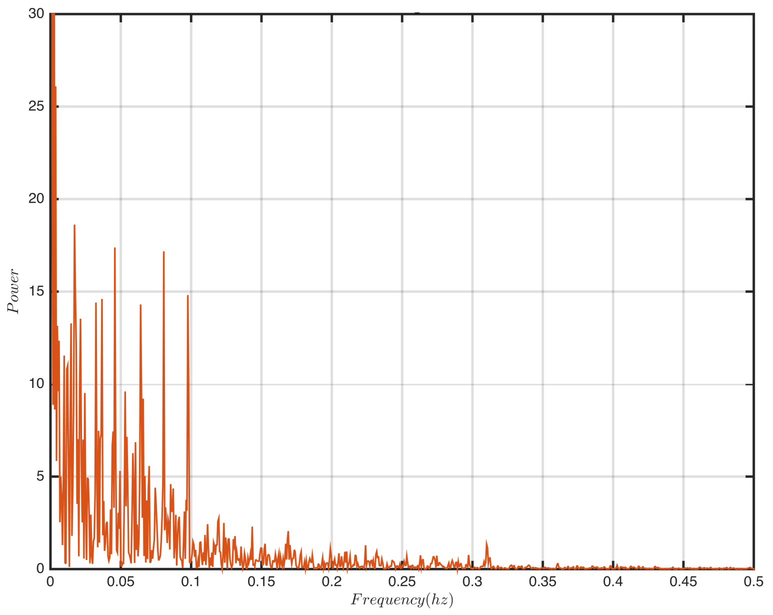
where b_α is a bias term, η_α denotes additive measurement noise, and $\dot{v} = [\ddot{x}\ \ddot{y}\ \ddot{z}]^T$ in the inertial frame.

A commonly used technique to estimate the bias b_α is to average the output of these sensors for a few seconds while the UAV is on the ground and the motors are not yet active. The bias is then assumed constant for the duration of the flight.

Through fig.4.4 we can see that, although filtered, both bias and measurement noise exist in our data. Using the aforementioned methodology in our twenty minute experiment we have some estimates of the bias which have been validated using real flight experimental data. It has to be pointed out that the bias term is not constant and can vary. Using the Fourier transform we can make a frequency analysis of the signal to find out if a lowpass filter can be used to counteract noise. As is can be seen later on, only low frequencies exist which cannot be dealt with.



(a) Stationary angular velocities



(b) $\ddot{x}$ Fourier transform

Figure 4.4: Three-axis accelerometer experimental data

Accelerometer bias		
Parameter	Value	Units
$b_{\alpha x}$	0.2852	m/s^2
$b_{\alpha y}$	-0.1480	m/s^2
$b_{\alpha z}$	0.0942	m/s^2

Table 4.1: Bias values of Eq.4.2.2.1

4.2.2 Gyroscope

The rate gyro measures the angular velocity of the body fixed frame relative to the inertial frame of reference expressed in $\{B\}$. These microelectromechanical devices are reliable and reasonably robust but can also be modelled in a similar way.

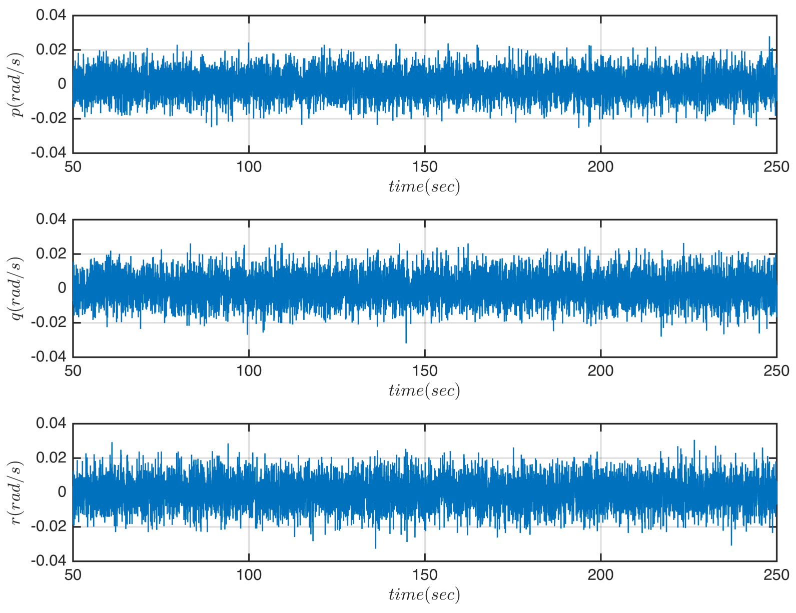
$$\Omega_{IMU} = \Omega + b_{\Omega} + \eta \in \{B\}, \tag{4.2.2.1}$$

As it can be observed through fig.4.5 the bias terms are close to zero. The noise consists of almost all frequencies (fig.4.5b) but has a very low amplitude and can be neglected.

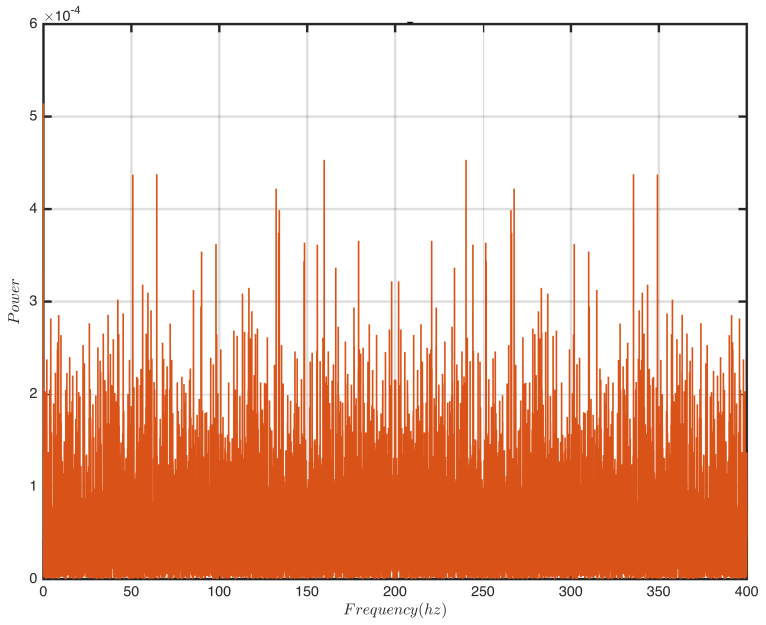
4.2.3 Global Positioning System

The fact that our platform has a GPS unit embedded can be very valuable. First of all, for computational efficiency GPS data can be used to sense both position and velocity and avoid using computationally heavy programs which rely on camera input. The problem is that GPS units in general have low accuracy and are very dependent to the number of fixed satellites which is influenced by the weather conditions, the surrounding buildings etc. There are ways with which better accuracy can be accomplished through data fusion and external hardware. For example Starmac, a custom built platform (see [12]) is fed with position and velocity at 10 hz using carrier-phase differential GPS relative to a stationary base-station, giving accuracy of 2 cm in the horizontal plane and then fuses this data with other sensors with an Extended Kalman Filter (EKF).

Our unit does not include a stationary base station and both position and velocity are sampled at 5 hz. As a result our measurements are of very low quality. The data seem to be filtered because both directions have only low frequencies in their signal. The bias terms are significantly low but there appears to be a low frequency vibration which cannot be eliminated with a low-pass filter. As a conclusion, our GPS data can grant an estimation of the x-y plane velocity with accuracy of about $\pm 40[cm/s]$ which is not good enough if our controller depends highly on velocity feedback. In our case, aiming to grasp the object at a predetermined speed, such oscillations cannot guarantee good performance if the control weights are the same for position and velocity. It is without saying, that we cannot expect a precise position estimation from the GPS unit if accuracy of some centimeters is desired.

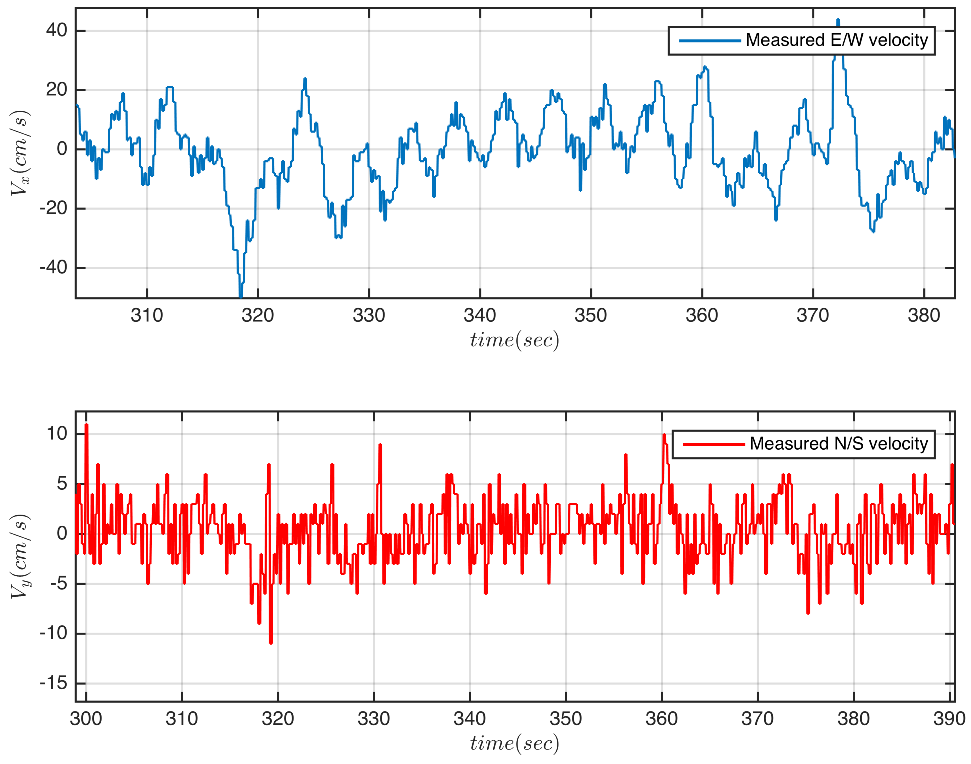


(a) Stationary angular velocities

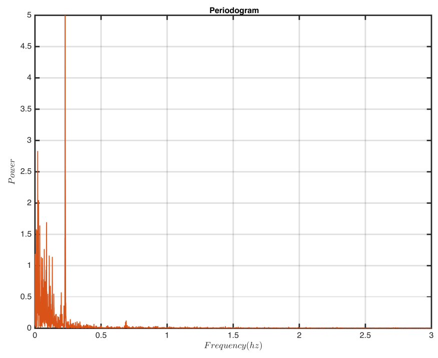


(b) q Fourier transform

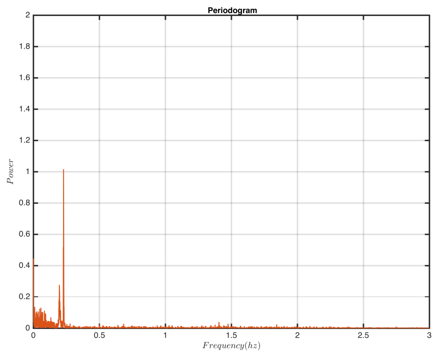
Figure 4.5: Three-axis rate gyro experimental data



(a) Stationary Velocity from GPS



(b) E/W Fourier transform



(c) N/S Fourier transform

Figure 4.6: GPS Velocity

4.2.4 Pressure Sensor

The pressure sensor can be used to sense both height z and $\dot{z}$ velocity. However, pressure sensors are very dependent to heat and need a reference height before their data is usable. Its competitive advantage against GPS measurements is that its measurements can be sampled a lot faster than 5 hz.

Observing fig.4.7 we notice that the height measurements are both inaccurate and also have an almost constant trend which does not seem to wear out. The oscillations are also of very low frequency for both z and $\dot{z}$ measurements and they can't be compensated. It is also very inefficient to wait for such a long time before the height measurements lose their trend and apart from that $100[cm]$ oscillations cannot contribute to our goal.

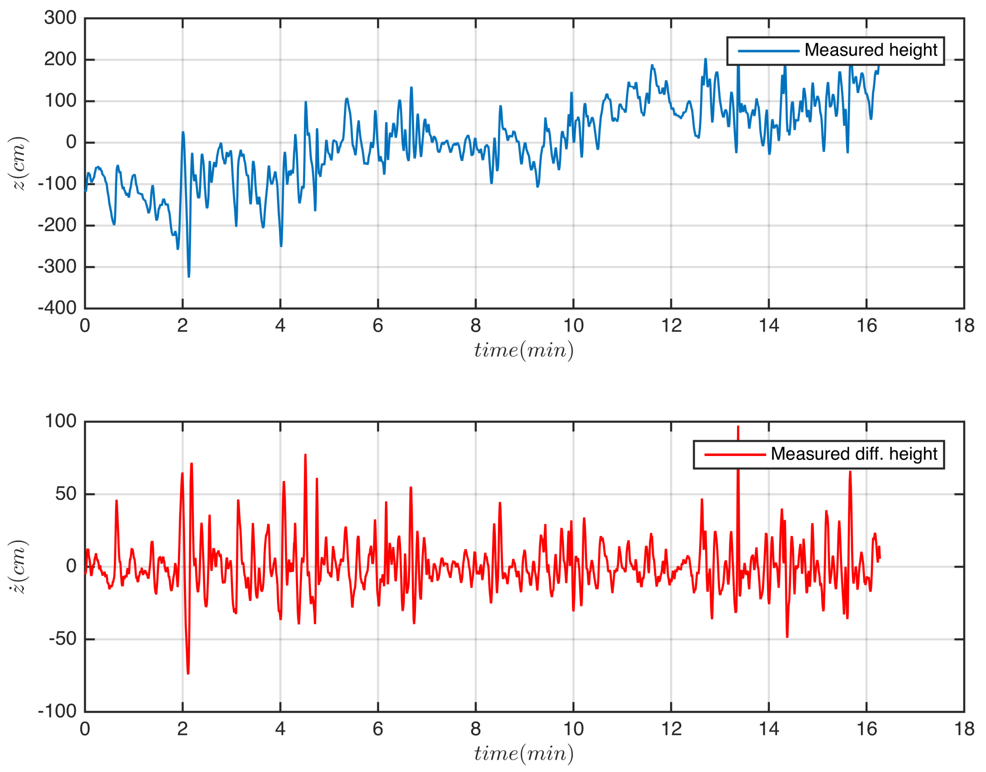
On the other side, differential height measurements seem to have more reasonable errors. To begin with, $\dot{z}$ measurements have no trend and oscillate at about $40[cm/s]$ which is reasonable in comparison with the GPS system.

4.2.5 Machine Vision

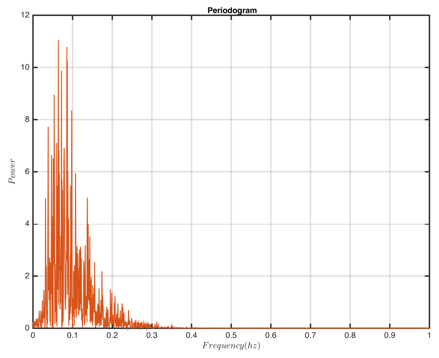
Given the precision that our goal requires we cannot rely on the sensors provided by the Asctec Autopilot. Mastermind provides several interfaces with which external sensors can be connected, like cameras. On-board cameras can be used for both position and velocity estimation which are indeed the most inaccurate measurements. After a lot of experimentation and given the capabilities of Mastermind Intel processor a double camera setup was adopted. In this chapter only some technical information and insights are provided; the actual math of how the image signal is being processed will be analyzed in another chapter.

The platform came with three external $USB2.0$ single-board cameras (see [23]) which can used three different lenses that provide a field of view of up to 140 degrees. However, using the latest official distribution of Ubuntu that Ascending Technologies provides, no ROS drivers exist so that we can use multiple cameras at once. That is the reason that an additional PS3 camera was used (see [25]) which operated in lower resolution and was responsible to feed the optical flow algorithm. Apart from that, given that our UAV will move in high velocities, our cameras need to have a very high frame-rate for the machine vision algorithms to work reliably. This means that if the images are distorted, no matter if the target is in the field of view of the camera, the object won't be recognized. Using multiple cameras with a big field of view would guarantee that the target would not be out of sight but would not guarantee that the quality of the images being processed would be satisfactory.

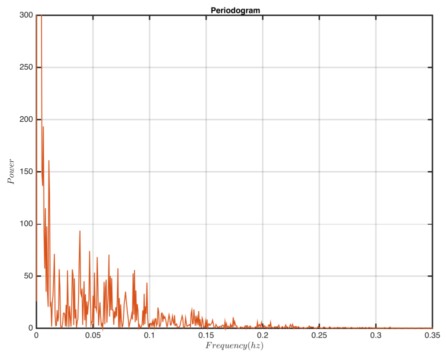
The only problem with using a large field of view lens is that the image can be distorted; especially in the edges. As a result, further image processing is needed before each image is fed to the machine vision algorithm. This is enabled through a ROS package called image proc which uses a yaml syntax calibration file created separately for each lens.



(a) Stationary data from pressure sensor



(b) Differential height Fourier transform



(c) Height Fourier transform

Figure 4.7: Pressure sensor

Cameras		
Parameter	mvBlueFOX-MLC200wc	PS3
Resolution	752x480	640x480/320x240
Mpixels	0.4	0.3/0.1
Max frame rate (hz)	90	70/120
Shutter type	Global	
Exposure time	$6\mu s - 460ms$	
Field of View	58°/100°/140°	56°/75°

Table 4.2: Machine Vision sensors

Table 5.1 can grant insight of the quality that our optical sensors have. In practice the maximum framerates that can be achieved are lower than the ones listed. This is mostly because of the computational load that the on-board computer can handle but also depends on the exposure time that is being used for each camera. For a setup with lots of light the exposure time can be very low and as a consequence we can maximize the frames that are being received each second.

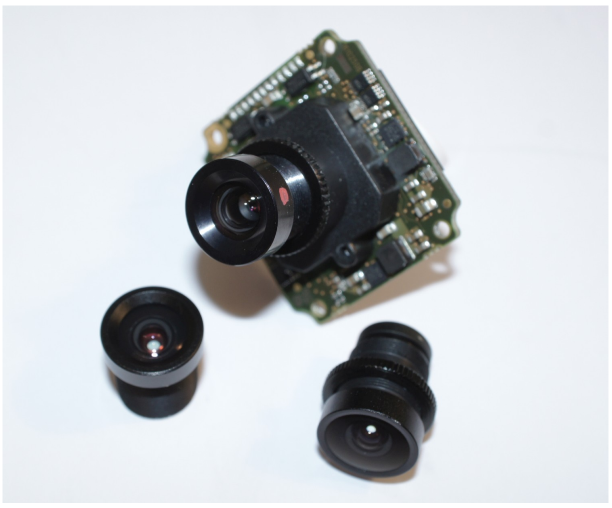

.

Figure 4.8: mvBluefox and different lenses

Chapter 5

Control Strategy

5.1 Model Predictive Control

Today's control software and technology offers the potential to implement more advanced control algorithms but often the preferred strategy of many industrial engineers is to design a robust and transparent process control structure that uses simple controllers. This is one reason why the PID controller remains industry's most widely implemented controller despite the extensive developments of control theory; however, this approach of structured control can create limitations on good process performance. One such limitation is the possible lack of a coordinator within the hierarchy that systematically achieves performance objectives. Another is the omission of a facility to accommodate and handle process operational constraints easily. The method of model predictive control (MPC) can be used in different levels of the process control structure and is also able to handle a wide variety of process control constraints systematically. These are two of the reasons why MPC is often cited as one of the most popular advanced techniques for industrial process applications.

Surprisingly, MPC and the associated receding horizon control principle have a history of development and applications going back to the late 1960s; Jacques Richalet developed his predictive functional control technique for industrial application from that time onward. Work on using the receding horizon control concept with state-space models can be identified in the literature of the 1970s, and the 1980s saw the emergence first of dynamic matrix control and then, towards the end of the decade, of the influential generalised predictive control technique.

Model Predictive Control has been widely used with great success in chemical and oil refinery industry, where the processes are slow and the systems order large, since the 1980s. In recent years model predictive controllers have been designed also for the navigation of UAVs but mostly for simple trajectories. That is mostly because the systems dynamics are very fast and if we want the set of inputs to be calculated online then the control problem ought to be relatively easy to solve.

The general design objective of model predictive control is to compute a trajectory of a future manipulated variable $\boldsymbol{u}$ to optimize the future behaviour of the plant output $\boldsymbol{y}$. The optimization is performed within a limited time window by giving plant information at the start of the time window.

The basic terms and variables that are being used in MPC design are listed below:

- Moving horizon window: the time-dependent window from an arbitrary time t_i to $t_i + T_p$. The length of the window T_p in general remains constant.
- Prediction horizon: is the length of the moving horizon window T_p and dictates how far in the future we want to predict the behavior of the plant being controlled.
- Control horizon: is the length of the moving horizon window T_c and shows how far in the future we will calculate the control inputs. Generally $T_c \leqslant T_p$.
- Receding horizon control: although the whole input trajectory withing the moving horizon window is being calculated, only the first sample is being fed to the system while the rest of the trajectory is neglected.
- In the planning process, we need information at time t_i in order to predict the future. This information is denoted as $x(t_i)$ and is the estimation of the plant state.
- A given model that will describe the dynamics of the system is paramount in predictive control. Model predictive control systems are designed based on a mathematical model of the plant. The model to be used in the control system design is taken to be a state-space model. By using a state-space model, the current information required for predicting ahead is represented by the state variable at the current time.
- In order to make the best decision, a criterion is needed to reflect the objective. The objective is related to an error function based on the difference between the desired and the actual responses. This objective function is often called the cost function J, and the optimal control action is found by minimizing this cost function within the optimization window.

There are three general approaches to predictive control design. Each approach uses a unique model structure. In the earlier formulation of model predictive control, finite impulse response (FIR) models and step response models were favoured. FIR model/step response model based design algorithms include dynamic matrix control and the quadratic DMC formulation. The FIR type of models are appealing to process engineers because the model structure gives a transparent description of process time delay, response time and gain. However, they are limited to stable plants and often require large model orders. This model structure typically requires 30 to 60 impulse response coefficients depending on the process dynamics and choice of sampling intervals. Transfer function models give a more parsimonious description of process dynamics and are applicable to both stable and unstable plants. The transfer function model-based predictive control is often considered to be less effective in handling multi-variable plants. Recent years have seen the growing popularity of predictive control design using state-space design methods.

For both approaching the target and flying away a discrete time non-linear model predictive controller was designed and used. In order for the optimization problem to grant a solution fast enough, we ought to consider the constraints which are vital to achieve our goal. To begin with, the approaching movement requires that the UAV has a non-equilibrium state vector right before the impact with the object as it has been proved in the impact mechanics chapter. This cannot be achieved with the use of regular MPC objective function (see [17]) where in each iteration the controller calculates a specific number of steps before the goal is reached. As a result, we ought to create a double minimization problem where the optimal number of remaining steps is also calculated and minimized.

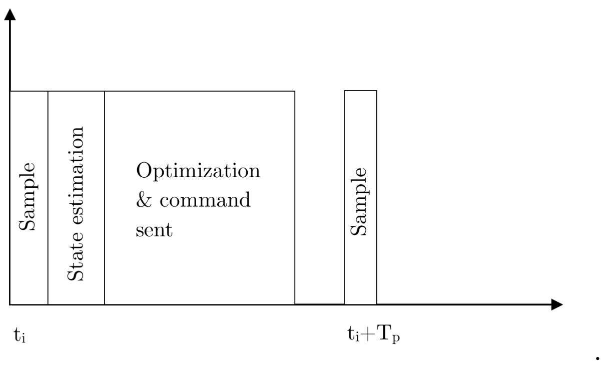


Figure 5.1: Typical MPC timing diagram

At each time step k the controller is fed with the hexarotor estimated state $\boldsymbol{x}_k$ and the last commands which were sent to the platform $\boldsymbol{u}_k$ and returns the calculated optimal trajectories of both states X_{opt} and inputs U_{opt} as well as K_{opt} which is the remaining number of steps until the target has been reached. The frequency that the optimal control problem is being solved is directly associated with the frequency that inputs are being fed to the platform. Solving the optimal control problem for the state trajectory at time k yields result at time $k+1$ when the platform has indeed moved. As a consequence, the input solution that is being sent to the platform is u_{k_1} and not u_{k_0}.

5.2 Minimization problem

The minimization problem that the predictive controller needs to solve at each time step is as follows:

$$\min_{K} \min_{\boldsymbol{X},\boldsymbol{U}} J(\boldsymbol{X}, \boldsymbol{U}, K) \tag{5.2.0.1}$$

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{5.2.0.2}$$

$$\boldsymbol{x}_{(0)} = \boldsymbol{x}_0 \qquad \boldsymbol{u}_{(0)} = \boldsymbol{u}_0 \tag{5.2.0.3}$$

$$\boldsymbol{u}_{\min} \leqslant \boldsymbol{u}_k \leqslant \boldsymbol{u}_{\max} \tag{5.2.0.4}$$

$$\boldsymbol{\Delta u}_{\min} \leqslant \boldsymbol{\Delta u}_k \leqslant \boldsymbol{\Delta u}_{\max} \tag{5.2.0.5}$$

$$\boldsymbol{y}_{\min} - s_v \leqslant \boldsymbol{y}_k \leqslant \boldsymbol{y}_{\max} + s_v \tag{5.2.0.6}$$

$$0 \leqslant K \leqslant K_{\max} \tag{5.2.0.7}$$

$$k = 0, 1, ..., K-1 \tag{5.2.0.8}$$

where $\boldsymbol{X} = [x^T_{(0)}\ x^T_{(1)}\ ...x^T_{(K-1)}]$, $\boldsymbol{U} = [u^T_{(0)}\ u^T_{(1)}\ ...u^T_{(K-1)}]$ and K is the number of steps needed to complete the task which in our case is also the outer minimization problem variable. The equality constraints model the need of the solution vector to be derived from the actual plant state space model as well as to begin from the last state (for simulation) and input vector. The hard constraints of the above minimization problem take the capabilities of the platform into account while the soft constraints on the outputs help us neglect some problems of the physical world that will be analyzed later on.

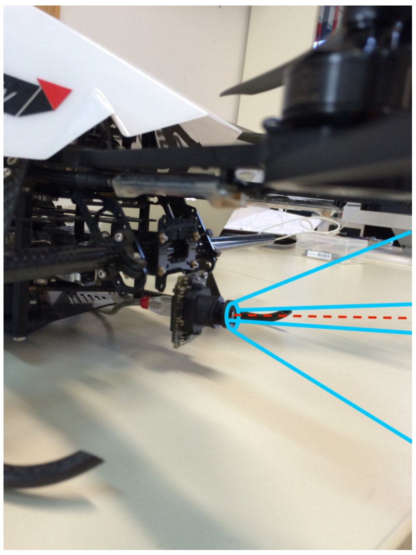
Traditionally the most valuable parameters of a MPC controller which can be tuned are the prediction and control horizon (K in our case represents both). To understand this concept lets assume that $K \to \infty$ or just sufficiently large. Then we would be able to predict and control the whole trajectory for some goal equilibrium state. The problem is that the computational load would be that high that we would not be able to calculate fast enough the next set of inputs that we want to send to the platform. In our case, control packages must arrive at rates higher than $10[hz]$ according to the asctec drivers but apart from that, the error of the Euler's forward approximation method which was used to create the discrete time model should be taken into account. Through experimentation with the model it was estimated that we need an algorithm capable to update the input trajectory in frequencies higher than $30[hz]$. If this scenario were not feasible, then another approximation method ought to be used. Keeping these aspects into mind we can select a K_{max} value for the outer minimization problem.

5.2.1 Hard constraints

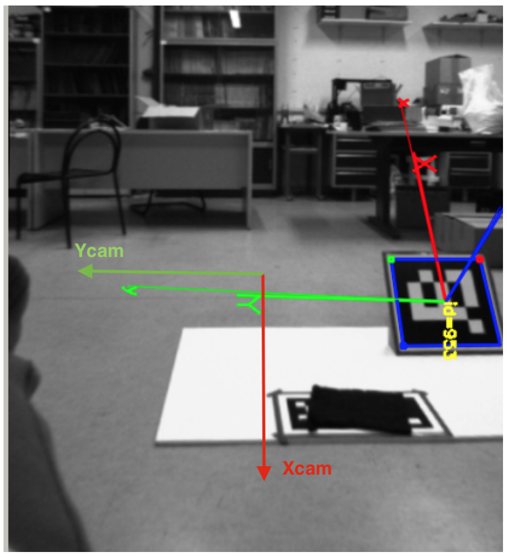
The hard constraints that have been applied to the controller have to do with the platform's input capabilities. Those capabilities have to do with not only the magnitude $\boldsymbol{u}_k$ of the input trajectory but also with the rate of change of the input variables $\boldsymbol{\Delta u}_k$. The actual upper and lower boundaries of these constraints can be valued with both some experimentation and research in the AscTec Autopilot manual. These values can be tuned according to safety measurements that we want to implement. For example we don't want for any reason the controller to send a thrust command of zero value. There are no actual constraints on the rate of change of the inputs but if we want to have a smooth input trajectory we need to embed them in the controller either as a hard constraint or as an additional cost. In our case the second approach was adopted.

Hard Constraints		
Input	Lower Bound	Upper Bound
Thrust T_d	0.0	1.0
Roll ϕ_d	$-52°$	$52°$
Pitch θ_d	$-52°$	$52°$
Yaw rate r_d	$-200°/s$	$200°/s$

Table 5.1: Input amplitude real hard constraints



(a) Camera pyramid field of view



(b) Target in C_c

Figure 5.2: Camera soft constraints

5.2.2 Soft constraints

An operating range for the plant output can also be specified. Output constraints are often implemented as soft constraints. This means that a slack variable s_v is added to relax the constraints. These constraints ought not be enforced because they can create a conflict with amplitude and incremental input constraints. These constraints can also be implemented to the estimated observer output. Their number can make the optimization time bigger and as a result the most trivial constraints should be considered.

In our case the trivial soft constraints that should be taken into account concern the movement of the UAV in 3D space.

$$x_{\min,\max} = \pm z^c_{cm} tan(a_v/2) \tag{5.2.2.1}$$
$$y_{\min,\max} = \pm z^c_{cm} tan(a_h/2) \tag{5.2.2.2}$$
$$x_{\min} - s_v \leqslant x^c_{cm} \leqslant x_{\max} + s_v \tag{5.2.2.3}$$
$$y_{\min} - s_v \leqslant y^c_{cm} \leqslant y_{\max} + s_v \tag{5.2.2.4}$$

The above set of equations keeps the marker, which is used to get the position output, within the camera frame. The notation p^c_{cm} is the position vector of the marker from the camera expressed in the camera frame while a_v and a_h are the angles of view in the vertical and horizontal plane which create a pyramid field of view. It is easier to formulate the set of equations within the camera frame C_c rather than in the inertial frame C_e but we first need to transform the estimated states in this specific frame which requires more computational power.

The last obvious constraint is the ground. This constraint was not relaxed with a slack variable because is is essential that the platform does not crash.

$$z_{\min} \leqslant z^e_{eb} \tag{5.2.2.5}$$

5.2.3 Objective function

The objective function penalizes state deviations from equilibrium (position equilibrium being the targets position) using a steady weighing matrix $Q \geqslant 0 \in \mathbb{R}^{nxn}$ where n is the number of states. It is also used to penalize incremental input change through another weighing matrix $R > 0 \in \mathbb{R}^{mxm}$ where m is the number of inputs. The incremental input term is neglected for the last $b > 0$ steps of the trajectory. This is done because we want to allow the platform to ask for inputs $u_i \in (u_{lb}, u_{ub})$ which will result in the final state vector that we want. Last but not least, terminal costs are added and tuned through matrix $Q_t \geqslant 0 \in \mathbb{R}^{nxn}$ which take the wanted final state into account and are applied for the whole trajectory generation. Of course the two controllers (approaching and retreat) have to be tuned separately because their objectives are different.

$$Ja(\boldsymbol{X},\boldsymbol{U},K) = \frac{1}{K+1}\left[\sum_{k=0}^{K} \boldsymbol{x}_{(k)}^T Q_a\, \boldsymbol{x}_{(k)} + \sum_{k=1}^{K-b} \boldsymbol{\Delta u}_{(k)}^T R_a\, \boldsymbol{\Delta u}_{(k)}\right] + \boldsymbol{x}_{\boldsymbol{t}(K)}^T Q_{ta}\, \boldsymbol{x}_{\boldsymbol{t}(K)} \quad (5.2.3.1)$$

$$Jr(\boldsymbol{X},\boldsymbol{U},K) = \sum_{k=0}^{K} \boldsymbol{x}_{(k)}^T Q_r\, \boldsymbol{x}_{(k)} + \sum_{k=1}^{K} \boldsymbol{\Delta u}_{(k)}^T R_r\, \boldsymbol{\Delta u}_{(k)} + \boldsymbol{x}_{\boldsymbol{t}(K)}^T Q_{tr}\, \boldsymbol{x}_{\boldsymbol{t}(K)} \quad (5.2.3.2)$$

The term $1/(K+1)$ is used to take the mean value of J_a with respect to K because otherwise the result of the outer minimization algorithm would constantly be $K = 0$ (see [17]). J_r, being the objective function of the second controller, does not need to have a second minimization problem because the goal is to hover some meters away from the object retrieval point. As a result all output constraints can be implemented because the optimization problem is a lot easier to solve.

5.3 Algorithm structure and optimization solution

In order to avoid a delay between the implementation of each controller, which could result in failure, the second controller problem is being solved before it is time for the commands to be sent. The initial state and input vectors that the second controller uses to do this are the predicted $\boldsymbol{x}_{(K-1)}^T$ and $\boldsymbol{u}_{(K-1)}^T$ which have been updated with the insight that the impact mechanics analysis has given. When the first controller finds that the goal has been reached K is set to zero and the second controller sends its first commands. This is the time when the object is suppose to be grabbed.

The optimization problems are being handled by a BFGS algorithm for the multidimensional optimization and Quadratic Interpolation for the single dimension problem without constraints (see [19]). This becomes possible with the use of Augmented Lagrange Multipliers. Lagrange Multipliers are additional independent

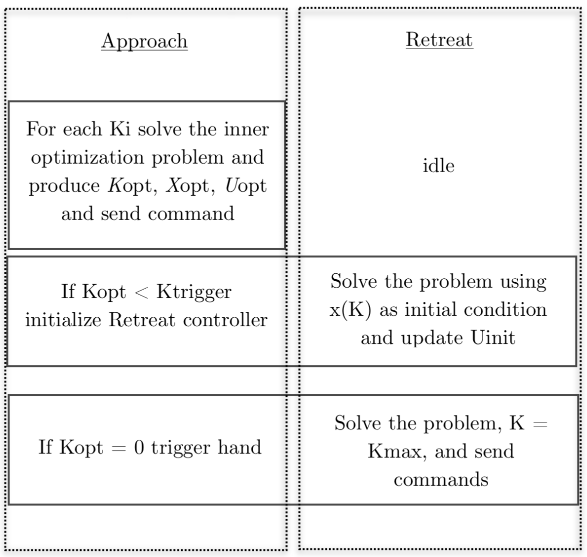

Approach
Retreat
For each Ki solve the inner optimization problem and produce Kopt, Xopt, Uopt and send command
idle
If Kopt < Ktrigger initialize Retreat controller
Solve the problem using x(K) as initial condition and update Uinit
If Kopt = 0 trigger hand
Solve the problem, K = Kmax, and send commands

Figure 5.3: Controllers cooperation

variables which are added to the optimization problem in order to take the equality and inequality constraints into account directly in the objective function. This helps significantly lower the time that the inner minimization problem needs (see [24]) . Below a simple example of this concept is provided where $\boldsymbol{\lambda}^T$ is the vector of Lagrange Multipliers.

Lets denote a simple objective function J with respect to the states (running and final) and a set of inequality (or equality) constraints.

$$J = \frac{1}{2}\boldsymbol{x}^T E\,\boldsymbol{x} + \boldsymbol{x}^T F$$
$$M\boldsymbol{x} \leqslant \gamma$$

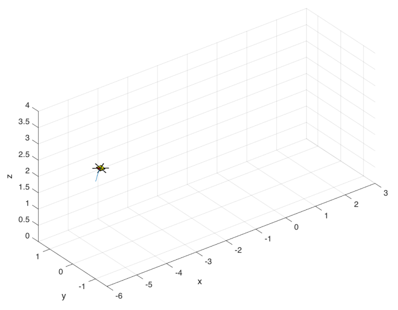
is equivalent to:

$$J = \frac{1}{2}\boldsymbol{x}^T E\,\boldsymbol{x} + \boldsymbol{x}^T F - \boldsymbol{\lambda}^T(M\boldsymbol{x} - \boldsymbol{\gamma})$$

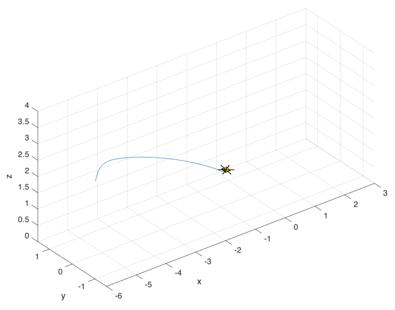
5.4 Simulation

The environment of the simulation matches the experimental setup which will be introduced later on. The platform starts from an equilibrium point with distances from the target in each direction $\vec{p}_0 = x_{tb}\hat{x} + y_{tb}\hat{y} + z_{tb}\hat{z}$. The main goal of the simulation is to show that the platform can indeed reach the desired final state with the proposed controller design with proper tuning of the weighing matrices. It is vital that the errors of the states converge at the same time to guarantee a bio-inspired trajectory.

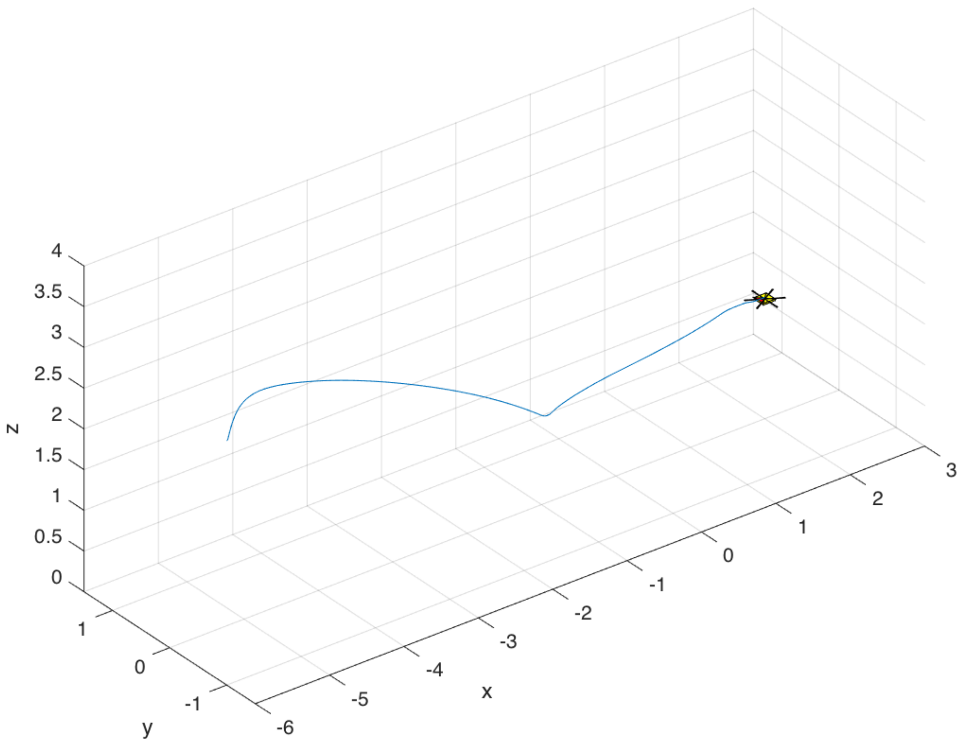
From fig. 5.4 it can be observed that the controller is designed to drive the platform in the first steps into a 2D plane with no oscillations in the x-y plane. This



(a) Simulation start $p_0 = [x_{tb}\ y_{tb}\ z_{tb}]$



(b) Non equilibrium retrieval point



(c) Simulation end - hover with grabbed object

Figure 5.4: Simulation

way, the impact mechanics analysis assumptions hold and as a result the impact force will create a torque mainly in the pitch plane. Right before the impact the UAV has a velocity $\dot{x} = 1.0[m/s]$ and a pitch angle $\theta = -10[deg]$ while all distances have converged to zero. In practice in order to be able to predict the impact force outcome but also to drive the platform with precision we need sensors which have substantially low errors. As is has been shown on-board sensors don't meet these criteria and as a result our objectives will be relaxed and an observer ought to be used to create more smooth state estimations and fuse IMU and Camera data.

Chapter 6

Implementation

6.1 Kalman Filter

The fact that we have so many problems in the sensing setup suggests that a sophisticated observer should be designed to fuse different sensors and get the best estimations for the UAV's position and velocity. As it has been shown, measurements have been taken to keep the markers within the camera frame but these constraints are soft and can be violated. Apart from that the vision algorithms running on-board don't operate necessarily in a constant frequency nor they are as fast as the IMU measurements. As a result our sensors have different operating frequencies and as a consequence model propagation is needed to achieve smooth estimated states which would result in a more stable controller.

To avoid adding non-linearities to the model of the Kalman filter (which would result in difficult tuning of the covariance matrices) acceleration propagation was used. The system model for an extended Kalman filter is in general (see [14]) as follows:

$$x_k = f(x_{k-1}, u_{k-1}) + w_{k-1} \tag{6.1.0.1}$$

$$z_k = h(x_x) + v_k \tag{6.1.0.2}$$

where w_{k-1} and v_k are the process and observation noises which are both assumed to be zero mean multivariate Gaussian noises with covariance matrices Q_k and R_k respectively. The EKF extends the scope of Kalman filter to nonlinear optimal filtering problem by forming a Gaussian approximation to the joint distribution of the state x and measurements z using a Taylor series based transformation.

System Update

$$\begin{aligned}
u &= [R_{eb}|\alpha_{IMU}|\Omega_{IMU}]^T \\
\alpha_e &= [\ddot{x}\,\ddot{y}\,\ddot{z}]^T = R_{eb}(\alpha_{IMU} + b_{ab}) - g\hat{z} \\
x_k &= x_{k-1} + dt \cdot \dot{x}_{k-1} \\
y_k &= y_{k-1} + dt \cdot \dot{y}_{k-1} \\
z_k &= z_{k-1} + dt \cdot \dot{z}_{k-1} \\
\dot{x}_k &= \dot{x}_{k-1} + dt \cdot \ddot{x}_{k-1} \\
\dot{y}_k &= \dot{y}_{k-1} + dt \cdot \ddot{y}_{k-1} \\
\dot{z}_k &= \dot{z}_{k-1} + dt \cdot \ddot{z}_{k-1} \\
\vec{b}_{\alpha b} &= \vec{b}_{\alpha b}
\end{aligned}$$

The acceleration bias is added to the model as a state because of the observer's sensitivity to acceleration measurements. Alternatively we could use the bias constants proposed in the platform sensors chapter. As it is designed the system model suggests that the bias values remain constant in the body frame (not in the inertial frame) but their values will be changed during the measurement update.

Measurement Update

The camera measurements are first transformed to the inertial frame and the they are passed to the filter as measurements. Below the procedure to obtain p^e_{eb} is provided, p^e_{eb} being the position of the UAV with respect to the inertial frame with respect to the inertial frame.

- Vision algorithm measurement: p^c_{mc} and R_{cm}
- Constant distance of camera and body in the camera frame: L^c_{cb}
- Rotation matrix from marker frame to earth: R_{em}
- $p_{em} = [0\ 0\ 0]^T$

$$\begin{aligned}
p^c_{mb} &= p^c_{mc} + L^c_{cb} \\
p^m_{mb} &= R^T_{cm} p^c_{mb} \\
p^e_{eb} &= p^e_{em} + R_{em} p^m_{mb}
\end{aligned}$$

The last equation gives us the camera measurement.

Optical flow has been used to get more accurate velocity feedback because during the experiments the platform will move in relatively low speeds. The optical flow measurement equation is as follows:

$$z_{of} = \frac{\hat{z}}{cos(\theta)cos(\phi)} \cdot (u_{of} - M\omega_c) = R_{cb} \cdot (R_{be}u_e + \omega_b \times L^b_{bc})$$

where u_{of} and M are retrieved from the optical flow algorithm and to avoid linearization both $R_{eb}(3,3)$ and $\hat{z}$ have been applied to create the optical flow measurement update. As a result the relation between the measurement and the actual states u_e are now known.

Last but not least, the measurement updates of GPS and differential height are highlighted:

$$
\begin{aligned}
z_{dh} &= \hat{\dot{z}} \\
z_{GPS}(1) &= -\hat{\dot{y}} \\
z_{GPS}(2) &= \hat{\dot{x}}
\end{aligned}
$$

Discrete-time predict and update equations

Predict

1. Predicted state estimate $\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_{k-1})$
2. Predicted covariance estimate $P_{k|k-1} = F_{k-1} Pk - 1|k - 1 F_{k-1}^T + Q_{k-1}$

Update

1. Innovation or measurement residual $\tilde{y}_k = z_k - h(\hat{x}_{k|k-1})$
2. Innovation (or residual) covariance $S_k = H_k P_{k|k-1} H_k^T + R_k$
3. Near-optimal Kalman gain $K_k = P_{k|k-1} H_k^T S_k^{-1}$
4. Updated covariance estimate $P_{k|k} = (I - K_k H_k) P_{k|k-1}$
5. Updated state estimate $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k$

where the state transition and observation matrices H and F are defined to be the Jacobians:

$$
F_{k-1} = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1|k-1}, u_{k-1}} \qquad H_k = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_{k|k-1}}
$$

6.1.1 Testing the filter

Offline testing of the filter was then implemented in order to tune matrices Q and R and to observe the quality of the state estimates. During the test flight the platform was navigated to lose the markers for some seconds in order to determine if it could still create a reliable estimation without optical feedback.

According to the results of the simulation, all states remain in reasonable limits which match those of the flight. This can be understood through fig.6.1 where the camera measurements are also shown. Of course the camera measurements are the most reliable sensor that we have to estimate the MAV's position and as a result the filter ought to follow the measurements and create a smooth trajectory while camera measurements are not available.

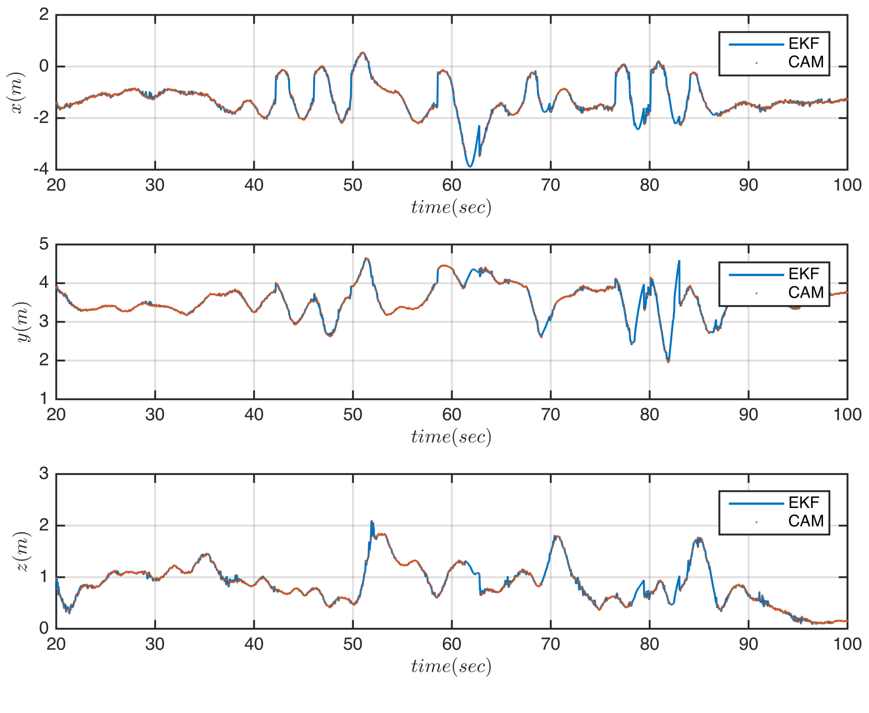


Figure 6.1: Position estimation & camera measurements

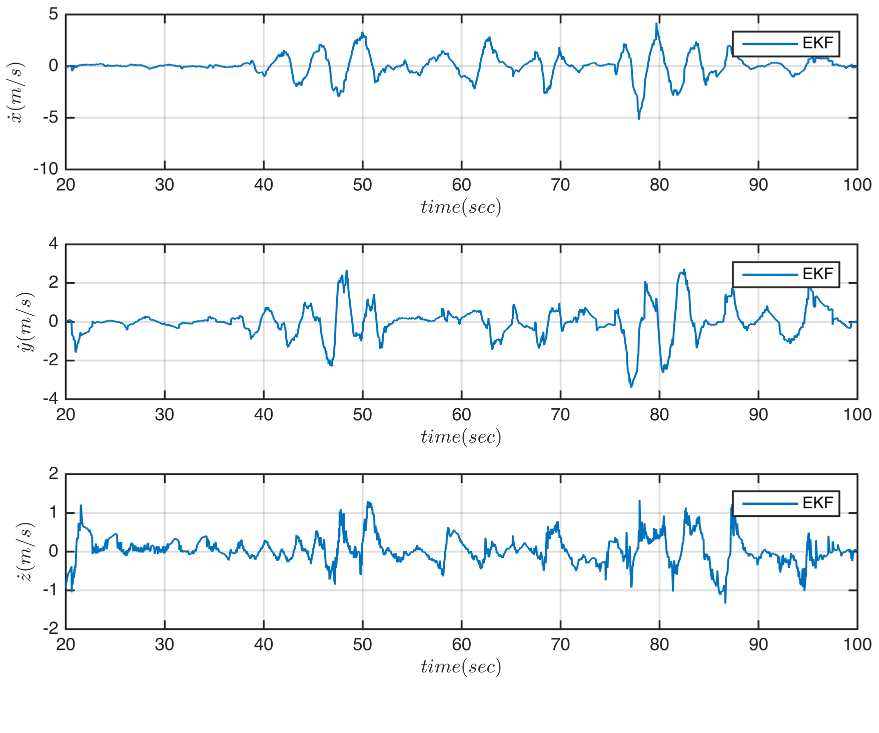


Figure 6.2: Velocity estimation

6.2 Robot Operating System & communication

The Robot Operating System (ROS) is an open source set of software libraries and tools that help build robot applications which can easily communicate with each other. Many platforms have adopted ROS and very reliable drivers exist in the web. This latter fact is very useful because we can directly move to creating high level control applications and forget about the necessary low level firmware that is needed.

Applications - Nodes

- *asctec mav framework*: This stack contains drivers, tools, a nonlinear position controller and imu data fusion for Ascending Technologies MAVs equipped with the Autopilot sensor board. Its substantial difference with other similar drivers is that it communicates with the user-programmable high level processor and in has configurable baudrate up to 921600 baud which can guarantee very fast rates of communication. For our needs this package was used to poll IMU and GPS measurements which are needed for state estimation.
- *mv bluefox driver*: A basic driver for the bluefox cameras. It can produce the basic image topics which will be then fed to the machine vision algorithm. Some parameters of the camera can be changes within the source code (fps, exposure time, etc).
- *uvc camera*: Driver used for the PS3 cameras. Several parameters can be set in the launch file.
- *ar sys*: The ar sys package contains the machine vision algorithm which is responsible to detect the markers in the camera frame.
- *optical flow*: Node which evaluates the velocity of the PS3 camera from optical feedback and publishes the results in a custom topic.
- *asctec ekf*: Implementation of the kalman filter. The node subscribes to the topics containing different sensing data and publishes the state estimation of the MAV and status at a constant rate.
- *nmpc approach*: The model predictive controller which receives state estimation data, evaluates the next sub optimal inputs and sends the commands to the platform. It also communicates with the second controller and the robotic gripper in order to trigger both the gripper and the second controller
- *nmpc escape*: Complimentary controller which listens to the trigger topic for a signal and then begins solving the optimal control problem and only sends attitude commands when the first controller has ended its execution.
- *openbionics*: A package which is used to create an interface between AscTec Mastermind and Arduino Micro microcontroller. In our case it acts like an on/off switch, either actuating or letting free the grip of the robotic hand.

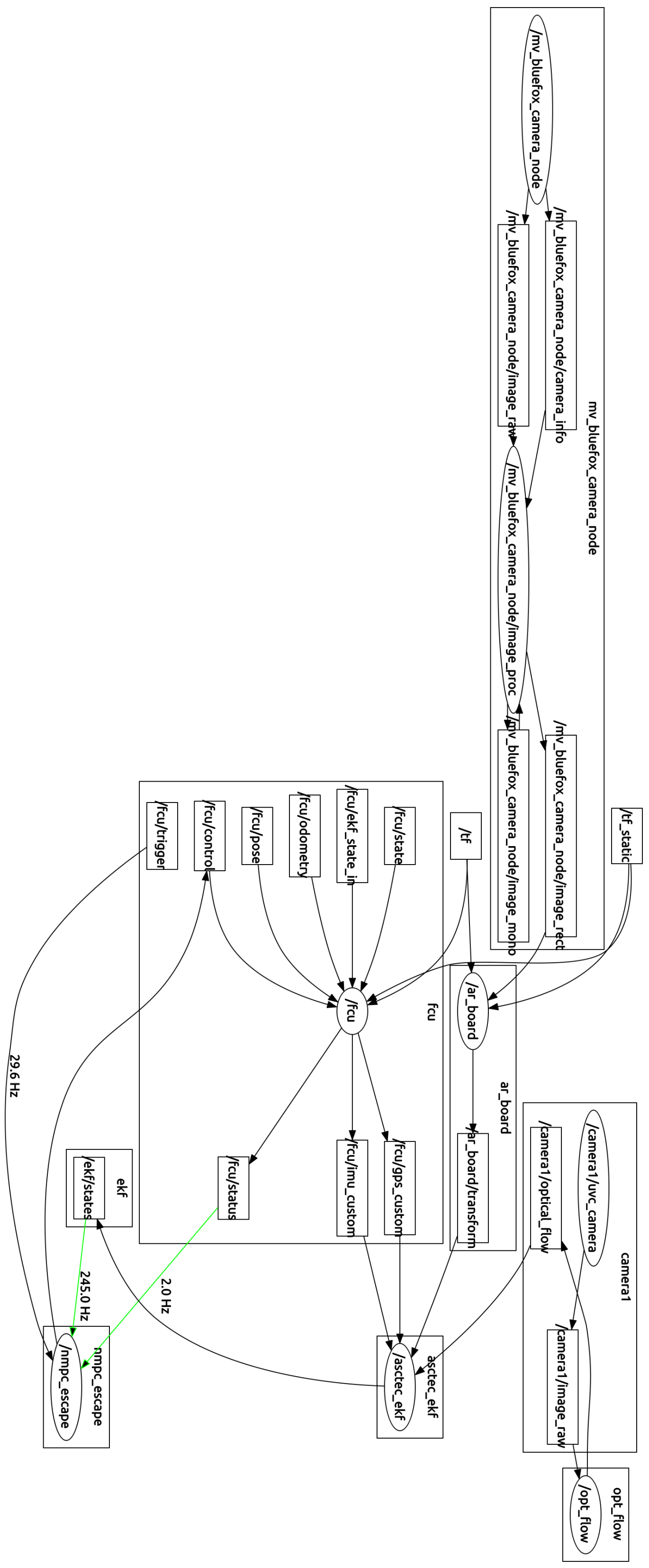


Figure 6.3: Nodes and topics communication while the 2^{nd} controller is active

Chapter 7

Results & discussion

The sensing scheme used in our platform was not considered good enough to support an aggressive trajectory tracking because of the large errors that were included in the position and velocity measurements. Nonetheless, several hover and target approach experiments were conducted to see with how much accuracy a target could be approached and if the platform could operate under such measurement noise conditions but no actual object retrieval was tested. The optical feedback was strengthened using larger markers in order to have a big area of operation. No hard constraints were implemented in order to solve the optimal control problem faster and due to the lack of an accurate position sensor.

7.1 Approaching the target

The approaching controller was tested first. The MAV is first driven to a specific distance and height from the target and then the controller is switched on. The serial interface is disabled when the outer optimization problem has a solution $K_{opt} = 1$ which means that the goal has been reached. As a result, control is passed to the safety pilot who then lands the platform.

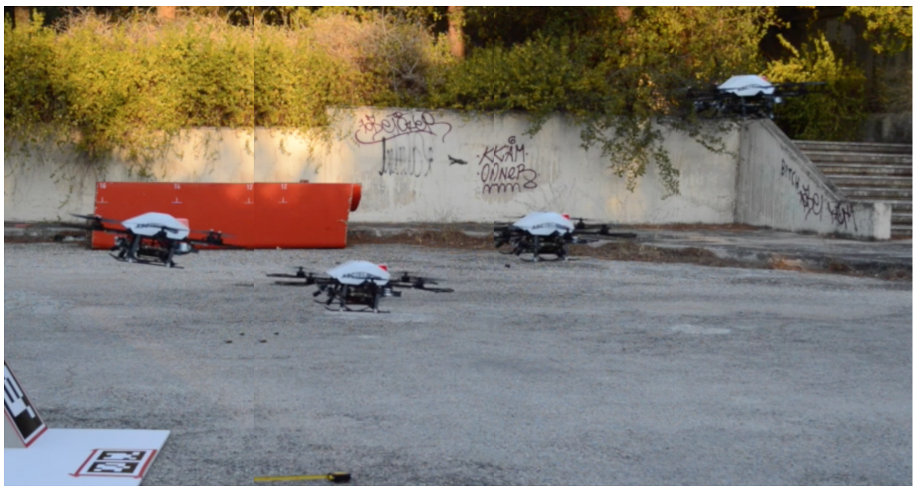
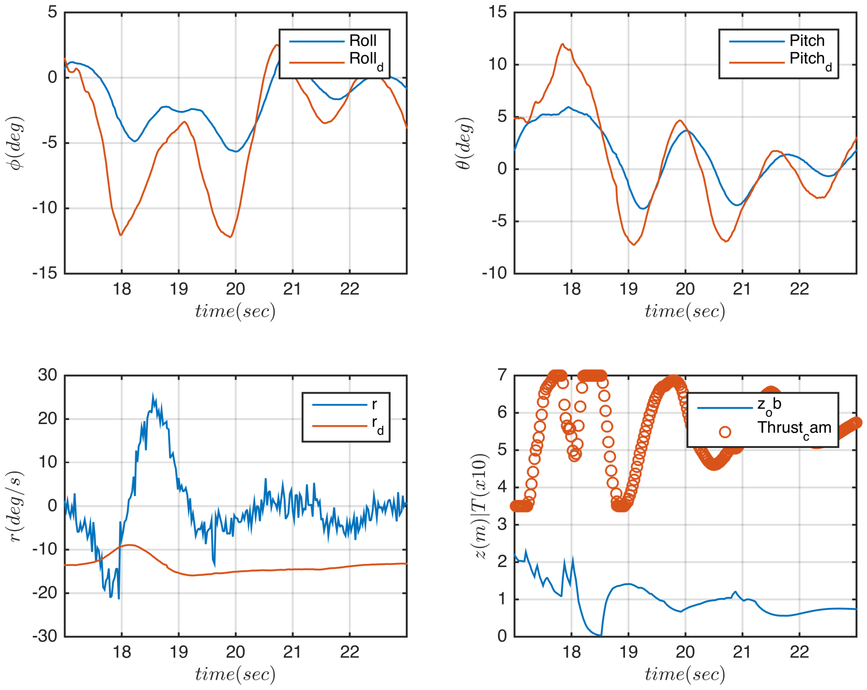
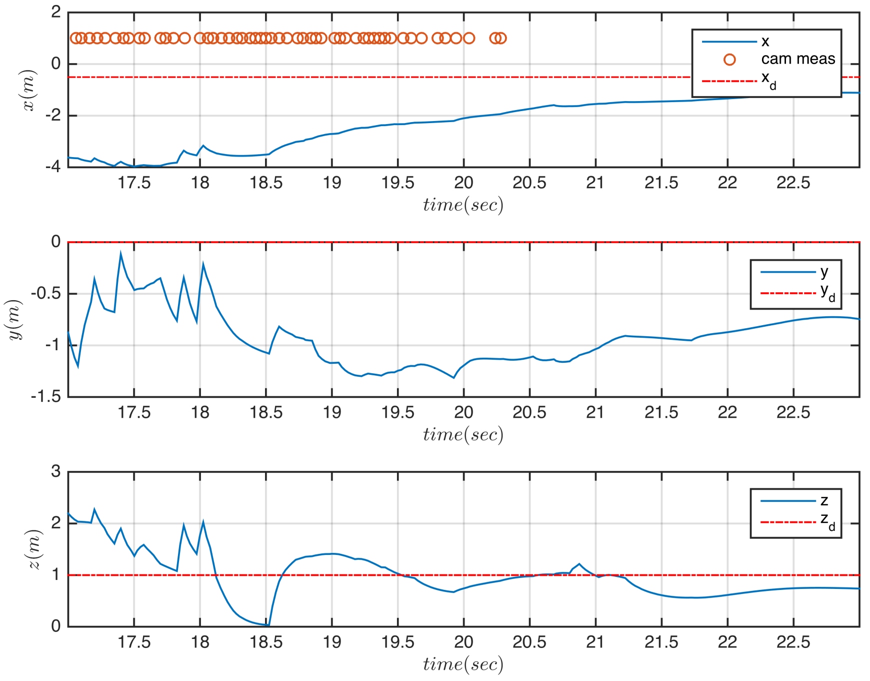

.

Figure 7.1: Approaching experiment



(a) Commands and response



(b) Position & Camera measurements

Figure 7.2: Approaching experiment

During this experiment a double camera configuration was used to get a wider angle of view and a static Kalman was implemented which propagated the accelerometer measurements to get an estimation of both position and velocity. With this setup not many fps could be achieved for each camera which resulted in poor position measurements both from accuracy and frequency standpoint. It can be seen from fig.7.2b that at some point the target was lost which as a result made the position estimation drift with the acceleration bias.

The most critical issue that can be observed is the error that the position estimation has, especially for z^e_{eb} and y^e_{eb} measurements. The displacements suggested by the measured position updates do not match the velocity GPS measurements which were lower than $2[m/s]$. This error occurs because the methodology used to get position measurements can only give reliable results for the depth z^c_{cm} while the other two measurements can have substantial errors depending on the distance from the marker. Apart from that, the position errors do not converge to zero when $K_{opt} = 1$ which is the main goal of the experiment (approaching the virtual object).

7.2 Approaching the target & hover

The second preliminary experiment's goal is to use both controllers to create and track a position trajectory similar to the simulation results (some additional tuning was done to compensate measurement noise). The virtual object was set at $p_o = [-1.0\ 0.0\ 0.5]$ and the hover location was set at $p_h = [0.0\ 0.0\ 1.0]$.

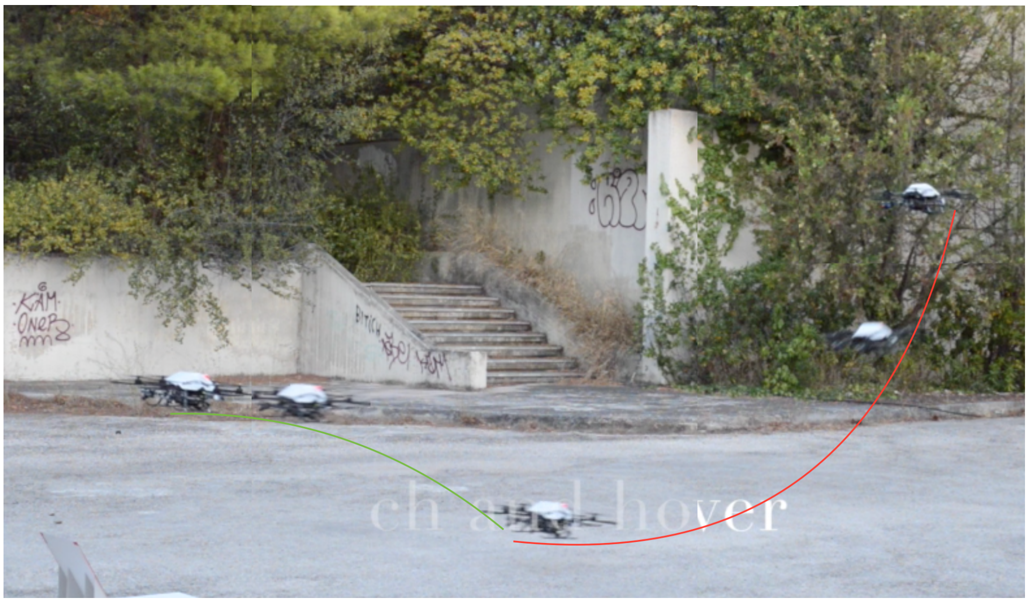
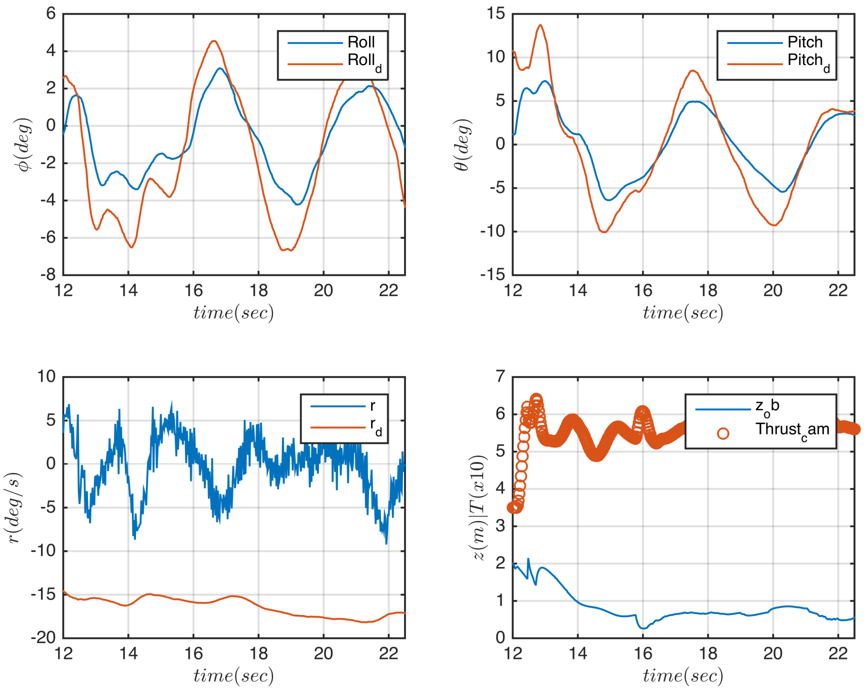
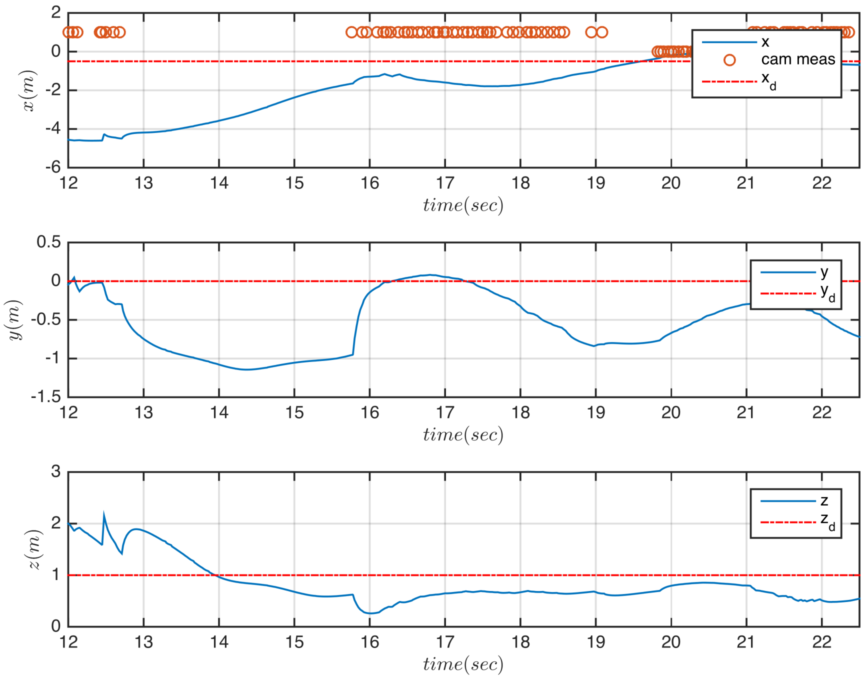


Figure 7.3: Approaching experiment

Similar observations can be made for this experiment as well. The first controller was unable to converge all errors at the same time, while hover was partially accomplished due to noise in GPS velocity measurements and external wind gusts. Of



(a) Commands and response



(b) Position & Camera measurements

Figure 7.4: Approaching experiment

big importance in the convergence of yaw error $\psi - \psi_d$ which, in spite the fact that there is substantial yaw rate input, does not become zero. This happens because the yaw measurement is the most inaccurate measurement of the IMU unit. This statement can be tested by rotating the platform for some degrees and checking the yaw measurement response which might not change at all.

Chapter 8

Issues for further research

8.1 Feedback

Until this point it should be clear that without a more sophisticated sensing scheme accurate position control (aggressive grasping nonetheless) cannot be accomplished. Below the different approaches in literature and their challenges are demonstrated while their potential contribution to our objective is outlined.

The most accurate and aggressive trajectories have been tracked using motion capture systems (see [16]) which can provide all pose measurements with extreme accuracy ($50\mu m$) at rates of $375[hz]$. The position measurements (having very little noise) can be numerically differantiated to compute the linear velocity of the robot. As a result, onboard sensors like IMU and GPS are not needed. However, motion-capture systems are expensive, and their sensor array has a limited spatial extent that is impractical to scale up for large indoor environments. Absolute position can also be valued with GPS measurements. Received onboard, GPS measurements create a more realistic approach for state estimation when fused properly with other measurements. Their accuracy can reach a few centimeters at up to $10[hz]$ when coupled with a stationary GPS unit. Stanford's STARMAC [12] quadrotors fuse GPS position measurements with IMU attitude rate and accelerometer measurements using an onboard Extended Kalman Filter (EKF) while local altitude sensing and control is achieved using an ultrasonic rangefinder.

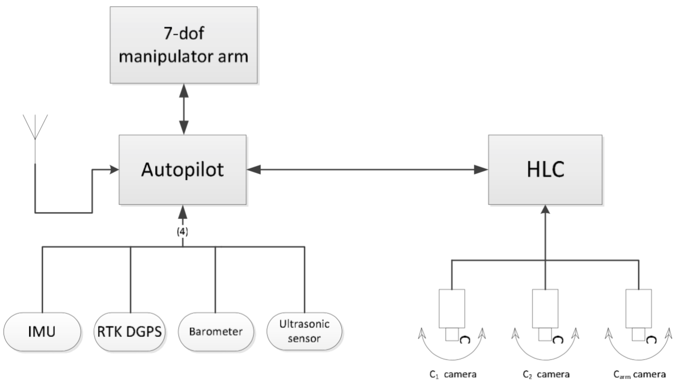


Figure 8.1: AMUSE control computers and sensors for outdoor operation [10]

Relative position can be estimated by measuring the distance to objects in the environment from onboard sensors, typically small onboard laser range finders (LRFs) or RGBD camera systems such as the Kinect. Well-known simultaneous localization and mapping (SLAM) techniques, borrowing LRF-based techniques similar to those developed for mobile ground robots over the last decade, have been applied to quadrotors. However, LRFs provide only a cross section of the 3-D environment and this scan plane tilts as the vehicle maneuvers, resulting in apparent changes to the distance of walls, and, in extreme cases, the scan plane can intersect the floor or ceiling. LRFs are heavy and power hungry, which could prevent their application to our project because they would significantly increase the payload and lower the autonomy of the platform. SLAM techniques generally require a lot of computational power.

Vision has the advantage that the sensor is small, lightweight, and low power, which will become increasingly important as the size of aerial vehicles decreases. Vision can provide essential navigational competencies such as odometry, attitude estimation, mapping, place and object recognition, and collision detection. Vision can also be used for object recognition based on color, texture, and shape, as well as collision avoidance. In our application we used vision to detect targets in the camera frames and thus detect the position of the platform in $3D$ space.

Vision is not without its challenges. First, vision is computationally intense and can result in a low sample rate which as a result can significantly lower the quality of sensed data. Second, there is an ambiguity between certain rotational and translational motions, particularly, when a narrow field of view perspective camera is used. Third, the underactuated quadrotor uses the roll and pitch DOF to point the thrust vector in the direction of the desired translational motion. For a camera that is rigidly attached to the quadrotor, this attitude control motion induces a large apparent motion in the image. It is therefore necessary to estimate vehicle attitude at the instant the image was captured by the sensor to eliminate this effect. Biological systems face similar problems, and interestingly, mammals and insects have developed similar solutions (gyroscopic sensors). Finally, there exists a problem with recovering motion scale when using a single camera.

Visual odometry can be a viable solution to leverage the low mass of onboard cameras and the computational power of Asctec Mastermind. Semi-direct Visual Odometry (SVO) and Parallel Tracking and Mapping (PTAM) techniques can grant an accuracy of some centimeters depending on the quality of the images being processed and the scenery. In this case the errors in the orientation of the object to be grasped do not pass in the estimation of the MAV's state and as a result this solution ought to be considered. These methods generally depend on feature extraction and mapping between consecutive pictures to determine how the vehicle has moved and as a result the framerate of the camera has to be high. Computational needs are also an issue and should be concerned. The results presented in [7] where received while using a Intel i7 8 core processor running at 2.8 GHz.

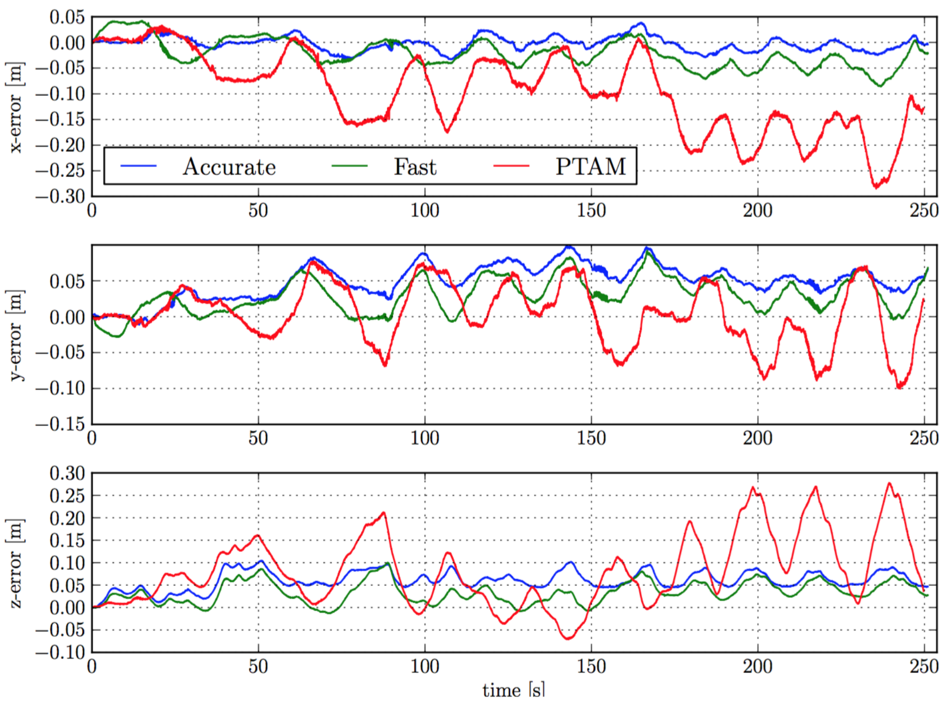


Figure 8.2: Position drift of SVO with fast and accurate parameter setting and comparison against PTAM [7]

8.2 System Dynamics

To achieve aggressive grasping one ought to consider the next three factors which can create implications in the system dynamics:

- Payload of Gripper
- Payload of the object to be grasped
- Aerodynamic effects near surfaces

These factors, for simplification of the problem and because the low level attitude control was used as is, were not taken into account. Having tested the platform in flight with the robot hand installed we ought to make some observations about the problems which occurred.

First of all, due to the robot hand and object (target) there is a displacement of the center of mass from the vertical axis at the geometrical center of the multirotor. Second, the variation of the mass distribution changes the moments of inertia. Lets keep in mind that the low level controller is tuned for the original platform. As a result, the system's dynamic behavior will vary and stability is not guaranteed if the payload overpasses some limits. This problem was neglected by keeping the COM in the same point in the x-y plane and by trying to create a low mass gripper. The mass of the object was also considered insignificant which in reality is not the case if we want our system to be robust. Although only one DOF was used to actuate the robot hand, reaction forces also exist which will have an impact in the overall balance of the MAV. For more insight on these problems and related publications towards possible solutions [10] is a good start.

Aggressive maneuvers require a sophisticated model and knowledge of the different aerodynamic effects which can change the behavior of the MAV. Those effects can be blade flapping and total thrust variation in translational flight (see [12]) as well as the ground effect. The first two should have been compensated in the original tuning of the commercial attitude controller. However if individual motor commands are to be sent in the future, these two phenomena should be taken into account.

The ground effect is a known phenomenon in aeronautics that happens when a flying body having a pressure difference above and below it, is near the ground. Thus, the airflow cannot freely homogenize pressures due to the presence of the ground, and it prevents the air from moving in all directions. As the system is subsonic the information is transmitted upstream altering the velocity field and also the pressure difference is accentuated by the presence of compressibilities. This phenomenon makes that the lift, in the case of fixed-wing aircraft, and rotor thrust, in the case of helicopters and multirotors, increase when the aircraft in question is near the ground. The ground effect is well known in helicopters, and has been studied for take-off, landing and hovering near the ground. However, for multirotors it has not received much attention. A similar effect can also occur while flying near a roof or walls of a building. These phenomena are significant if actuation in the natural environment is of interest.

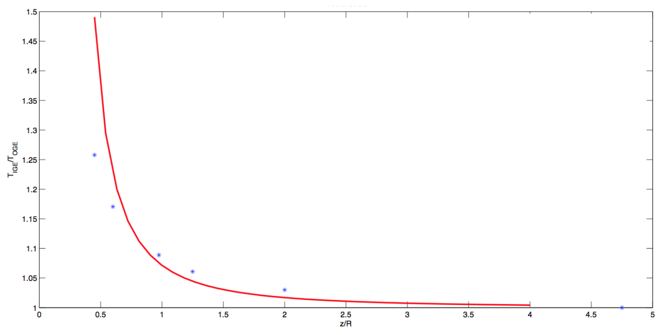


Figure 8.3: Ground effect results ARCAS Deliverable D4.2

Last but not least, the open bionics robot hand's elastomer material design, although being able to grasp a wide variety of objects, create vibrations during operation due to air flow which can significantly deteriorate the overall stability and make aggressive maneuvers not possible. These vibrations can be reduced if the hand is actuated during operation, released right before pickup and re-actuated, but as it can easily be understood this is not a very intuitive and power efficient approach.

Appendix A

Using Firefly

A.1 Onboard Computer

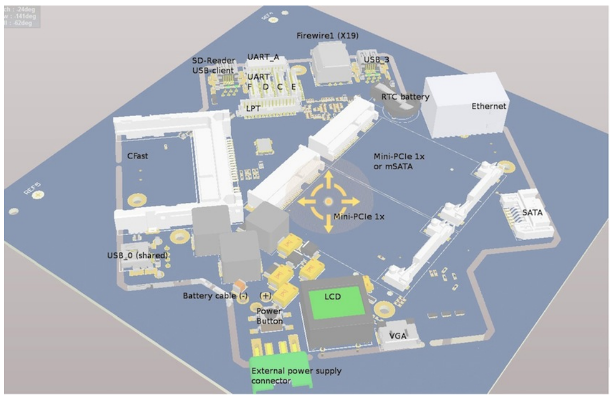


Figure A.1: Mastermind board - Top View

While working with AscTec Mastermind the user can connect a VGA monitor, USB-keyboard and mouse for interaction. Alternatively the computer is set up to create an ad-hoc connection so that anyone can log from a remote machine using an ad-hoc connection (pass 12345678). Every user is advised to first go through the Mastemind manual (see [21]) before using the onboard computer. It is vital that the computer is powered either via the 11.1V batteries from Ascending Technologies or via the Graupner Ultra Duo plus 60 using the **output 3**! If the Graupner solution is used then only Mastermind ought to be turned on. Do not try to use the cable to power the AscTec AutoPilot. There are two external power supply connectors, one that is fixed to the board of Mastermind and another which is connected to the Asctec AutoPilot. Connect the external power supply cable of Graupner only at the fixed connector (see A.1). In order to power the AutoPilot connect the battery to the connector which is attached to the AutoPilot board (one with the cable), this way both the onboard computer and the autopilot can be powered by the same power source.

If the Mastermind is powered through the Ascending Technologies batteries then they ought to be discharged and charged correctly. The computer by default will shut down if the voltage drops below $9.0[V]$. Although this safety feature exists, it is advised to change the batteries when their voltage drops below $11.0[V]$ to preserve their good condition. A simple bash script and a bashrc alias command have been created to aid the user with this task. While working in a terminal typing in the command "*power*" will return the current voltage of the battery. It is recommended that a seperate terminal window is opened for the "*power_alarm*" to run. The terminal window can be pinned in the front of other windows so that it is always visible. When the voltage drops below the aforementioned limit a constant warning will appear so that the user is notified.

Using AscTec Mastermind steps

1. Insert power source cable according to the above instructions and push on/off button on the breakout-board to boot up the Mastermind
2. Open a new terminal on your ground pc and connect to WiFi network with the name firefly (pass:12345678)
3. Use ssh protocol to connect to host (ssh asctec@10.10.0.1). You will be prompted to enter the superuser password (asctec). Alternatively you can copy in your machine the *sshconnection* script (from the media attached to the thesis).
4. If multiple terminals are needed; use the *terminals* script which opens many terminals and connects automatically to the platform (first the wifi connection should be established).
5. Source the ROS environment with ros_all command (an alias command which runs other multiple commands; for more info check the .bashrc file in the home directory).
6. If a ground PC is to be used (multiple machines in the ROS environment) remember to export the information of the master. Usually the onboard computer is the master and the ground computer is the slave. The commands that need to be run in each slave terminal are the following:
    - export ROS_MASTER_URI=http://10.10.0.1:11311
    - export ROS_IP=(*IP of slave in firefly network*)
7. In one of the opened terminals connected to firefly run the *power_alarm* script and set the window to *always on top* to keep track of the battery voltage.

A.1.1 Common problems

In order to use the *asctec mav framework* drivers the high level processor has been flashed with custom code and there have been several changes to the linux kernel causing some problems to the serial communication. The changes in the kernel were made to be able to boost the baudrate so that data can be polled in very high rates according to [2].

- Unable to identify external USB hardware: The newly compiled kernel has created problems when connecting external hardware like keyboards or a mouse.

- Poor serial port connection: Mastermind is equipped with several serial ports that the user can use. We were unable to use them correctly in order to actuate the robotic hand. Connection was established but data was not sent nor received correctly.

- RTC battery: This battery is used to keep the machine clock ticking. If the battery charge runs low and the computer is not connected to the Internet at startup then an inconsistency will occur and the computer won't start up normally. In this case change the battery using antistatic equipment.

A.2 Flight System and Experiments

Before attempting a flight with this platform the user is advised to get at least 5 hours of simulation practice using an appropriate remote control attached to the simulator and maybe flying first with Parrot AR.Drone platform which is easier to handle and less easy to destroy. Then it is advised to read the user manual in the AscTec Research webpage to familiarize with the different modes of operation, emergency mode and status feedback. Currently a Futaba remote control has been bound to the RC receiver on the platform (custom remote control). The mode of operation has been called FIREFLY and the commands have been set as normal. The left stick controls thrust and yaw while the right one pitch and roll. The Serial Interface switch and different modes of operation can be selected from switches E and C respectively (see the figure below). C is a 3 position switch, down is acceleration mode, middle is height mode and up is GPS mode. E switch is also a 3 position switch which when pointing away from the pilot will enable the serial interface. You can always check the actuation of different modes by using *rostopic echo /fcu/status* while *fcu* node is running and change the switches (while on the ground) to test if the results are normal.

If a controller is to be tested, a safety pilot should always be ready to take control of the vehicle. Currently due to the noise of the sensors only acceleration mode works reliably for takeoff and landing while being the hardest mode to handle. Be careful while in this mode with the thrust stick. Giving zero thrust means that the platform will fall like a stone and does not mean that it will be able to recover in flight. Usually (depending on payload) 53% thrust and above will make the platform gain height while below this limit will make the platform descend. These observations were made while hovering.

In order to pass inputs to the platform though a ROS node it is advised to first gain some height manually. A more stable hover control could be created because it is difficult for the safety pilot to keep the platform in hover and at the same time switch on the serial interface. While the controller is active always be ready to turn the serial switch to off and gain control of the MAV. Remember to have the sticks centered (especially thrust) because these commands will be immediately sent to the platform when the serial switch is turned off.

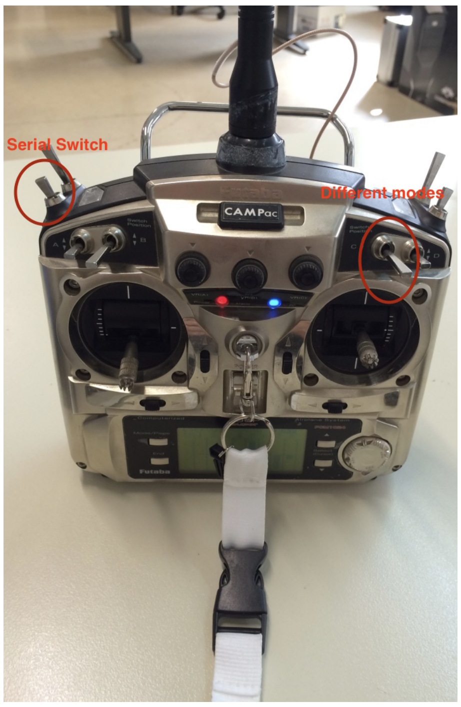


Figure A.2: Futaba RC - Serial Interface and modes of operation

Extensive demonstration of the different algorithms used is not provided, because it is easy to go through the ROS tutorials and use the ros wiki page to go through the different programs which were used. However, a small guide is provided with a typical experimental setup. All the experimental code and packages which were used for the experiments and simulation are included in the multimedia disk. If a controller is to be tested it is easier if two people help with the experiments (one using the ground PC and one as a safety pilot).

1. *Rosbag*: Decide which topics are needed to store for further analysis. If the camera image is needed then use the compressed topic to save space.

2. *Terminal windows*: Open the different terminal windows and connect to the platform before flight to ensure that all the different programs that you need are up and running. You can always search for command in the buffer with *Ctrl+R* keybind.

3. *Weather conditions*: Do not attempt a flight if the weather conditions are not appropriate so check before the probability of rain and the speed of the wind. Remember that the speed of the wind is bigger in higher altitudes.

4. *Optical feedback*: Ensure that the lighting is ok and the optical feedback nodes give reliable results with echoing the different topics and using *image_view* package.

5. *Positioning*: It is safer and easier if the safety pilot is always positioned behind the tail of the platform (easier to understand how the platform will behave while turning).

6. *Battery voltage*: The person controlling the ground PC is in charge of checking the battery voltage from the *power_alarm* script and notify the pilot accordingly.

7. *Start up all nodes*: Check if the programs run smoothly and all the sensors have initialized correctly.

8. *Gain height*: The safety pilot should gain some height and try to hover in the initial location.

9. *Initialize bagfile*: Start the bagfile and check that logging is done correctly. You can always check if and when the controller was active through */fcu/status* topic.

10. *Enable serial switch*: If the controller has already been switch on and sending viable commands to */fcu/control* topic then the safety pilot will immediatelly lose control of the platform. To regain control turn the serial switch to off.

11. *Disable serial switch*: The safety pilot should now manually fly the platform to the ground.

12. *Shutdown nodes*: It is recommended to try to preserve the battery voltage. After the experiment has ended; shutdown all the nodes and bagfile

13. *Shutdown the platform*: Use *sudo shutdown -h now* from the ground PC to turn off mastermind (while using an ssh connection). You need to separately switch off the AscTec Autopilot board.

14. *Retrieve bagfile*: The bagfile is stored at the directory where the *rosbag* command was used. Use *secure copy* to send the file to the ground PC.

15. *Analysis*: The bagfile can be converted to separate csv files each one containing a different topic using the *bag2csv* script. However there are ways to import bagfiles directly into Matlab using appropriate Matlab packages.

16. *Experimental Results*: All the results from the experiments can be accessed through the Matlab file structure which is included in the thesis multimedia disk.

Sending commands: While debugging the controller is was observed that there is an inconsistency in the way the attitude commands should be sent. The frame of all onboard sensors is alligned with front-left-up coordinate frame. However, the attitude commands sent to */fcu/control* topic should be aligned to the back-right-up frame. So lets say for example that the controller wants to pass an input $\theta_d = 0.1[rad]$ and $\phi_d = 0.2[rad]$; the commands which need to be passed for these two inputs are $\theta_d = -0.1[rad]$ and $\phi_d = -0.2[rad]$ due to this latter observation. Yaw and thrust inputs are regular because z points up.

Bibliography

[1] Markus Achtelik. *asctec-mav-framework*. 2014. URL: `http://wiki.ros.org/asctec_mav_framework`.

[2] Markus Achtelik. *Onboard Computer Setup*. 2015. URL: `http://wiki.ros.org/asctec_mav_framework/Tutorials/onboard_computer_setup`.

[3] Ruesch Andreas. "Dynamics Identification & Validation, and Position Control for a Quadrotor". Semester thesis. Swiss Federal Institute of Technology Zurich, 2010.

[4] AZoM.com. *Silicone Rubber*. 2001. URL: `http://www.azom.com/article.aspx?ArticleID=920`.

[5] Tommaso Bresciani. "Modelling, Identification and Control of a Quadrotor Helicopter". Master thesis. Department of Automatic Control Lund University, 2008.

[6] Bernard Brogliato. *Nonsmooth Impact Mechanics*. 1996.

[7] Davide Scaramuzza Christian Forster Matia Pizzoli. "SVO: Fast semi-direct monocular visual odometry". In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. 2014, pp. 15 –22. DOI: `10.1109/ICRA.2014.6906584`.

[8] Michael Shomin Daniel Mellinger Quentin Lindsey and Vijay Kumar. "Design, modeling, estimation and control for aerial grasping and manipulation". In: *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. 2011, pp. 2668 –2673. DOI: `10.1109/IROS.2011.6094871`.

[9] Nathan Michael Daniel Mellinger and Vijay Kumar. "Trajectory generation and control for precise aggressive maneuvers with quadrotors". In: *The International Journal of Robotics Research* 31.5 (2012), pp. 664–674. URL: `http://ijr.sagepub.com/content/31/5/664.abstract`.

[10] I. Sanchez D. Llorente V. Vega J. Braga J.A. Acosta G. Heredia A.E. Jimenez-Cano and A. Ollero. "Control of a multirotor outdoor aerial manipulator". In: *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. 2014, pp. 3417 –3422. DOI: `10.1109/IROS.2014.6943038`.

[11] Jonghyuk Kim Yisheng Zhong Hao Liu Dafizal Derawi. "Robust optimal attitude control of hexarotor robotic vehicles". In: *Springer Science & Business Media Dordrecht 2013*. 2013, pp. 1155–1168.

[12] Steven L. Waslander Claire J. Tomlin Haomiao Huang Gabriel M. Hoffmann. "Aerodynamics and Control of Autonomous Quadrotor Helicopters in Aggressive Maneuvering". In: *Robotics and Automation, 2009. ICRA '09. IEEE International Conference*. 2009, pp. 3277–3282. DOI: 10.1109/ROBOT.2009.5152561.

[13] Chun-Han Lin Joe Woong Yeol. "Development of Multi-Tentacle Micro Air Vehicle". In: *International Conference on Unmanned Aircraft Systems (ICUAS)*. 2014, pp. 815–820. DOI: 10.1109/ICUAS.2014.6842327.

[14] Arno Solin Jouni Hartikainen and Simo Särkkä. "Optimal Filtering with Kalman Filters and smoothers *a Manual for the Matlab toolbox EKF/UKF*". In: *Department of Biomedical Engineering and Computational Science, Aalto University School of Science*. 2011.

[15] Lennart Ljung. *System Identification: Theory for the User*. 1987.

[16] Vicon Motion Systems Ltd. *What is motion capture*. 2015. URL: http://www.vicon.com/what-is-motion-capture#.

[17] Charalampos P. Bechlioulis Panagiotis Vlantis Panos Marantos and Kostas J. Kyriakopoulos. "Quadrotor Landing on an Inclined Platform of a Moving Ground Vehicle". In: *Robotics and Automation, 2015. ICRA '15. IEEE International Conference*. 2015, pp. 2202–2207.

[18] Daniel R. Bersak Paul E. I. Pounds and Aaron M. Dollar. "Grasping From the Air: Hovering Capture and Load Stability". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 2491 –2498. DOI: 10.1109/ICRA.2011.5980314.

[19] Singiresu S. Rao. *Engineering Optimization Theory and Practice*. 2009.

[20] Ascending Technologies. *AscTec Firefly*. 2015. URL: http://wiki.asctec.de/display/AR/AscTec+Firefly.

[21] Ascending Technologies. *AscTec Mastermind*. 2015. URL: http://wiki.asctec.de/display/AR/Mastermind+Manual.

[22] Justin Thomas et al. "Toward autonomous avian-inspired grasping for micro aerial vehicles". In: *Bioinspiration & Biomimetics* 9.2 (2014), p. 025010. URL: http://stacks.iop.org/1748-3190/9/i=2/a=025010.

[23] Matrix Vistion. *USB 2.0 single-board camera - mvBlueFOX-MLC*. 2015. URL: http://www.matrix-vision.com/USB2.0-single-board-camera-mvbluefox-mlc.html.

[24] Liuping Wang. *Model Predictive Control System Design and Implementation Using MATLAB*. 2009.

[25] Wikipedia. *PlayStation Eye*. 2015. URL: https://en.wikipedia.org/wiki/PlayStation_Eye.

[26] Jr. David G. Rethwisch William D.Callister. *Materials Science and Engineering an Introduction 9th edition*. 2014.

[27] W.J.Stronge. *Impact Mechanics*. 2000.

[28] Michael Zeltkev. *Forward and Backward Euler Methods*. 1998. URL: http://web.mit.edu/10.001/Web/Course_Notes/Differential_Equations_Notes/node3.html.

[29] A.G. Zisimatos et al. “Open-source, affordable, modular, light-weight, under-actuated robot hands”. In: *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. 2014, pp. 3207–3212. DOI: `10.1109/IROS.2014.6943007`.