DESIGN AND CONTROL OF A COMPACT AERIAL MANIPULATION SYSTEM WITH A DELTA-TYPE PARALLEL ROBOT

BY

GABRIEL BARSI HABERFELD

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Mechanical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Adviser:

Professor Naira Hovakimyan

ABSTRACT

This thesis presents the design, modeling, and control of a quadcopter equipped with a Delta-type parallel manipulator. Such systems present demanding challenges in both control theory and task planning, which are addressed with novel mechanical features, modern flight controllers, and optimal trajectory generation. They are primarily designed for versatile indoor pick-and-place tasks where the characteristics of the proposed solution introduce useful kinematic properties. We explore these traits to address critical deficiencies found in previous approaches.

First, we introduce and discuss the mechanical design of the coupled system. Second, we derive the kinematics and dynamic relationships between all bodies. Third, we develop the flight controller, where baseline, feedforward, and adaptive components are combined and used in unison with an optimal trajectory generation algorithm. Finally, we present simulation results which reflect the feasibility of the concepts.

ACKNOWLEDGMENTS

I would first like to thank my mother Eva, without whom I'd never be where I am. Her unconditional support was paramount to all the events leading to this degree and many other milestones.

Second, I would like to thank my girlfriend Victoria for both emotional support and guidance in moments of need. She might as well be a second author of this thesis.

Next, I would like to acknowledge my adviser, Prof. Naira Hovakimyan, for her support and input in all aspects of this research.

Last but not least I thank my friends and colleagues: Thiago Marinho, Arun Lakshmanan, and Mitchell Jones, for their help, suggestions, discussions and, most importantly, the enjoyable times we all shared. Additionally, I thank all of my friends in Brazil, who always supported me in all my aspirations.

"The progressive development of man is vitally dependent on invention."

Nikola Tesla

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Aerial manipulation has undoubtedly become a major robotics research area in the past few years. The industry has driven this interest with investments in areas such as inspection, mapping, agriculture, age in place, and urban environments. The need for an autonomous aerial vehicle endowed with a robotic arm arises when a simple gripper attached to the bottom of the aircraft does not possess enough reach and/or DoFs to execute a given mission with sufficient robustness. This thesis' research arose from the age in place problem, where there exists a fundamental need for technologies to assist the ever-increasing elderly population [2, 3, 4]. The project "Automation Supporting Prolonged Independent Residence for the Elderly" (ASPIRE) [5] was funded by the NSF to investigate the use of indoor compact aerial and terrestrial co-robots. Following this framework, a compact quadcopter and robotic manipulator vehicle is designed for indoor pick-and-place tasks. Aerial vehicles typically excel in such environments given their ability to traverse common obstacles such as stairs and furniture and of reaching objects in a higher altitude, where ground robots would be unable to do so. The problem is, however, generalizable to a multitude of different frameworks.

Previous works tackled this problem in one of two ways. Either the system has a large quadcopter to manipulator mass ratio, giving it high flight actuation overhead, or it is developed for a constrained set of tasks, both of which make the stabilization problem easier to tackle. The former, although certainly feasible, is not representative of the current trend in quadcopter downsizing [6], and is hardly scalable. The latter is efficient and scalable but undermines the idea of a versatile platform reminiscent of having a flying multirole robotic arm.

Currently, aerial manipulation systems tend to be endowed with a serial manipulator, most notably [7, 8] recently attained strong results. The reasoning for this is understandable, seeing as serial manipulators are more accessible to model, control, design and implement. They also provide the aircraft with the much-needed reach, an area in which these types of manipulators excel. It does, however, come at a cost: the dynamics are *strong*, in the sense that the force/torque wrench generated at the base of the manipulator (i.e. the quadcopter) is high in magnitude.

In this thesis, the use of a parallel manipulator is explored. Due to its mass properties and actuator topology, Delta-type manipulator introduces fewer disturbances to the quadcopter as well as higher end-effector bandwidth at the cost of a smaller work volume and a complicated model. By designing the manipulator around the quadcopter these shortcomings are attenuated. In addition, the usefulness of the base to end-effector parallelism as well as the increased speed and precision are attained. In [9] the authors successfully implement a similar parallel manipulator in a quadcopter, achieving satisfactory performance and task execution. The work utilizes most of the components of this thesis approach. Their design mounts the manipulator in such a way that most of the disturbances are transmitted to the (non-critical) yaw axis and, although interesting, this topology severely constrains the set of possible tasks. Ultimately, this makes the vehicle fall into the task-specific category, as it is unable to grasp objects beneath the aircraft and, at the same time, imposing a constant mass asymmetry. In [10] the authors implemented a 6-DoF version of the Delta-type topology, however, their goals revolved around stabilization of the end-effector, instead of pick-and-place tasks.

The design and control of an aerial manipulation system with a low quadcopter to manipulator mass ratio and satisfactory reach and end-effector bandwidth is developed in this work to provide critical contributions to aerial manipulation research which are unattainable by previous approaches. We build upon previous works [11], which will be contrasted against the design presented here. The autonomous aircraft is designed to have little flight actuation overhead as well as wide task execution versatility. The autonomous aircraft displayed in Figure 1.1 is intended for small manipulation tasks such as pick-and-place of small objects (medicine, papers, glasses, etc.). This is achieved by the novel integration of a parallel 3-DoF manipulator with a compact, high-performance quadcopter. The manipulator is designed to attain a large workspace and fast servo motors are used to reach desired performance indexes. A torque compensating feedforward controller is added to the baseline autopilot to account for the fast dynamics of the manipulator.

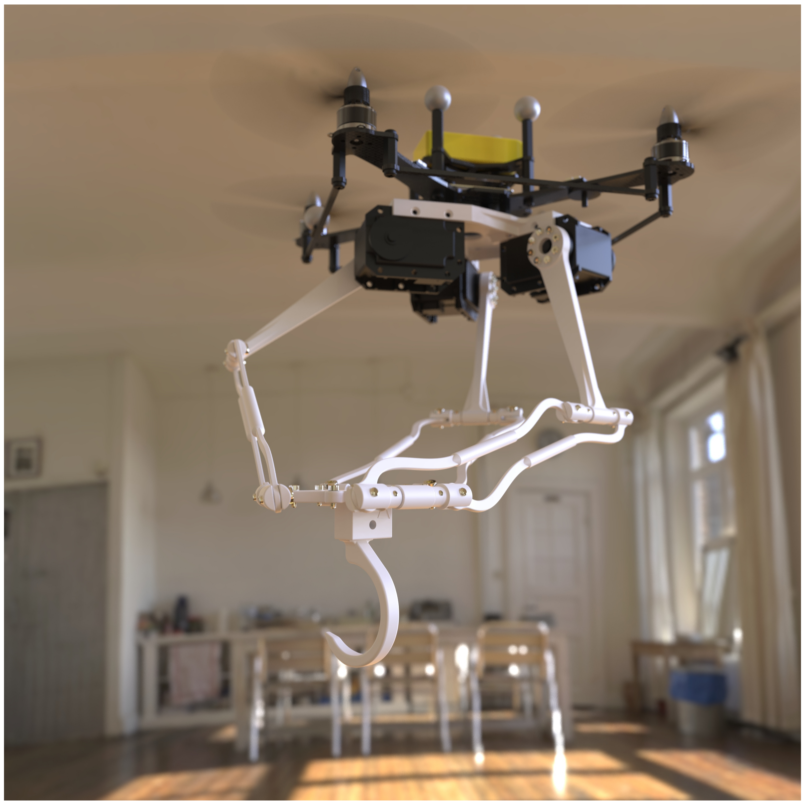


Figure 1.1: Autonomous quadcopter equipped with a 3-DoF parallel manipulator in and indoor environment.

In addition, an $\mathcal{L}_1$ adaptive controller is designed to account for unknown disturbances and improve tracking performance. A trajectory generation technique is developed where an optimal control algorithm minimizes energy while accounting for the kinematics of the coupled system. The design process of all components is discussed extensively.

This thesis is organized as follows: Chapter 2 discusses key design aspects for the coupled system; Chapter 3 develops the mathematical background to study the dynamics, and to design the controllers; Chapter 4 presents the control laws to attain stable flight; Chapter 5 develops an optimal end-effector trajectory generation algorithm. Lastly, Chapter 6 presents simulation results, concluding satisfactory performance, and Chapter 7 provides the conclusion and future directions of developments. The author would like to acknowledge the work done in [11] and its authors for the contributions to this thesis.

CHAPTER 2

MECHANICAL DESIGN

2.1 Quadcopter Design

Since it is intended for this framework to excel in indoor tasks, a compact and powerful quadcopter is necessary. To the knowledge of the author, no commercial counterpart provides the size/power ratio with enough flexibility, hence a new model needs to be developed for this purpose. In [11] a vehicle is designed which has a thrust to weight ratio greater than 7 before the addition of the manipulator, totaling over 1.6kg of maximum take-off load. In this iteration a slight increase in size yields a substantial increase in performance, allowing for the heavier parallel mechanism to fly with similar flight times. The resize allows the use of high-performance stiff two blade carbon fiber 7-inch propellers with slightly bigger motors, resulting in a total take-off load of around 2.4kg while having lower noise levels than the smaller prototype. The price paid is the increase in size and loss of agility. To accommodate for these changes a Lithium-Polymer battery with greater capacity is used (9.5Wh versus 5Wh). The controller of choice is the Crazyflie 2.0 nano quadcopter platform [12]. The platform provides all of the instrumentation and processing power that is necessary while offering a customizable serial channel, paramount for the complete integration of all the electronics. The control board is adapted for the larger frame and for use with brushless motors; it also includes a tunable cascaded PI/PID as the baseline flight controller. Factory gains stabilizes the larger vehicle, however retuning is necessary for satisfactory performance.

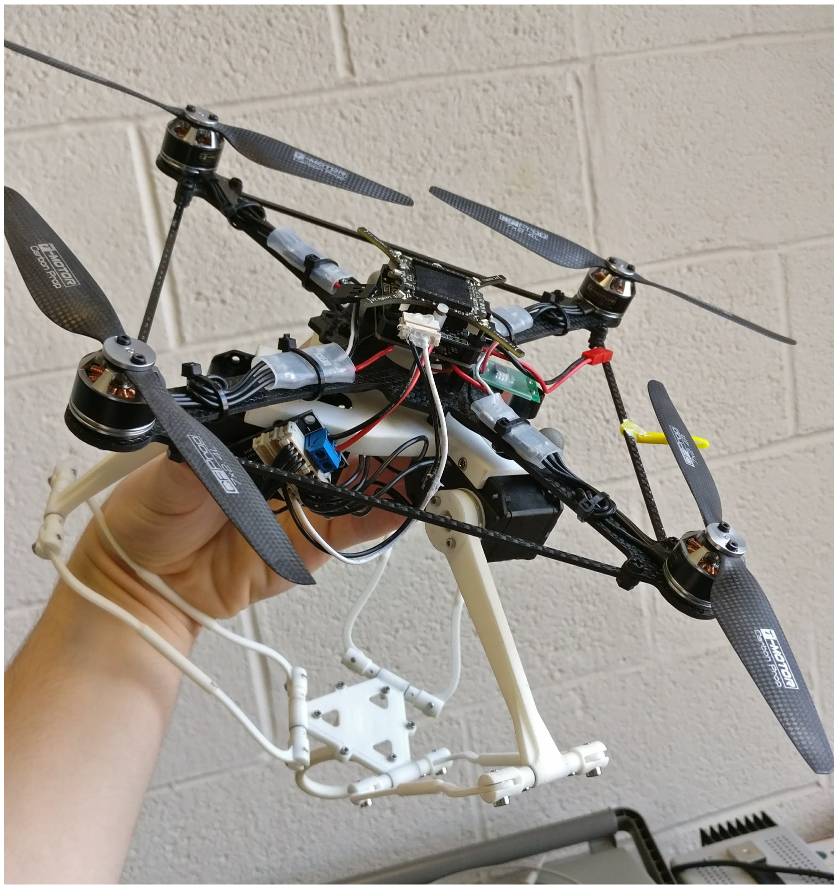


Figure 2.1: UAS prototype, the manipulator is printed using nylon SLS.

2.2 Manipulator Design

Manipulators, in general, are engineered on a task-specific basis, since each family of robotic arms provides the designer with a different set of advantages over the others. Traditional approaches utilizing serial manipulators aim to explore the better reach, ease of dynamics modeling, and use of readily available literature. In this parallel design, we consider the real needs an aerial manipulation system has along with the known ground-based results in order to elevate what is currently expected from this family of systems.

The first prototype in [11] is a serial manipulator with 2-DoF; however, having only 2-DoF requires the quadcopter to act as the third actuator for complete 3-DoF reachability and proper positioning of the end effector. Since it is logically desired for the end-effector to have a higher bandwidth than the vehicle in all of its axes of motion, a 3-DoF manipulator is needed. This immediately causes design conflicts for a serial type as at least one of the actuators would contribute negatively to the stability of the system by having to be placed in a moving part of the arm. The parallel-delta manipulator (often referred as Clavel's Manipulator [13]) however, manages no place all three actuators at its base, greatly reducing the dynamics induced on the aircraft. The move from serial to parallel changes several aspects of design, as shown in Table 2.1, where improvements of interest to this work over the serial type are highlighted in green, while declines in red.

This approach provides us several advantages such as improved stiffness, better accuracy, better manufacturing error rejection and others at the cost of a reduced workspace and complicated dynamics. The two major downsides are the small workspace and difficult dynamics since calibration can be done mid-flight utilizing modern positioning systems and a pseudo-inverse-Jacobian method in the outer loop (shown in Chapter 5). To increase the workspace, exceptional stiffness is explored. The stiffness of these devices is so abundant that any load heavy enough to cause deflection in the links would be far too heavy for the aircraft to lift. By curving the ends of the secondary links with Bézier curves to guarantee continuous lines (reducing mechanical stress), the design is able to achieve nearly 180 degrees of motion while maintaining enough stiffness to lift the maximum allowable payload. Figure 2.2 exemplifies the folding capabilities of these joints, and Figure 2.3 compares it to the industry-standard approach for ground-based delta-type parallel manipulators.

Feature	**Serial**	**Parallel**
Workspace	Large	Average and complex
Workspace/robot ratio	High	Low
Forward kinematics	Easy	Very difficult
Inverse kinematics	Difficult	Easy
Position error	Accumulates	Averages
Force error	Averages	Accumulates
Maximum force	Single actuator	Sum of actuators
Stiffness	Low	High
Dynamics characteristics	Poor	Excellent
Solving Dynamics	Relatively simple	Complex
Inertia	Large	Small
Payload/weight ratio	Low	High
Speed and acceleration	Low	High
Accuracy	Low	High
Component uniformity	Low	High
Calibration	Simple	Complex

Table 2.1: Serial and Parallel Topology Comparison

The actuators of choice are the Dynamixel RX-24F [14], which communicates via an RS-485 bus directly to the Crazyflie 2.0 board.

By utilizing this novel design the device nearly attains the reach of the serial manipulator variant while increasing accuracy, precision, stiffness, bandwidth and adding one extra degree of freedom. The only true downside is the increased weight, although future prototypes are expected to utilize compact, better-optimized actuators and links which will undoubtedly improve both load capacity and flight performance. An additional property to bear in mind is that the end-effector is now always parallel to the aircraft, which translates to a loss in the ability to orient the gripper as desired. This, however, is often unnecessary. Having a 4-DoF (three from the manipulator plus one from the quadcopter's yaw) platform eliminates the need for a more articulate end-effector, and in the rare occasions where it is required, the modular tool-base allows for interchangeable, mission-specific grippers to be installed.

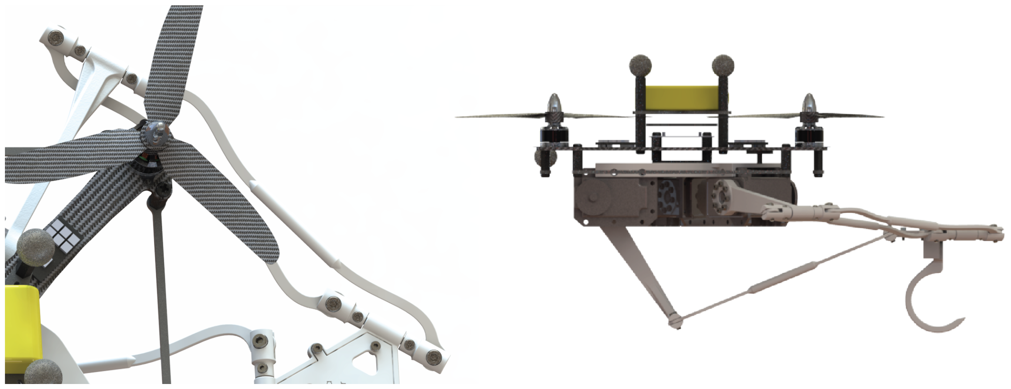
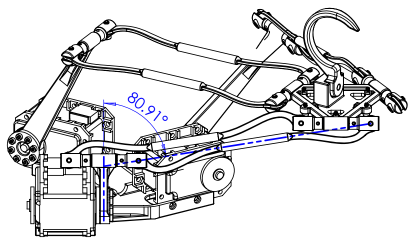
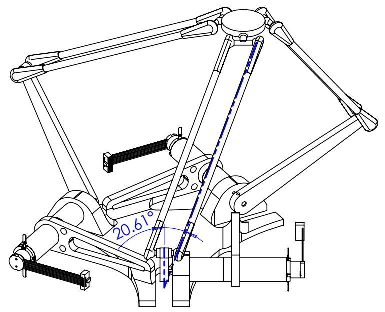


Figure 2.2: Folding property of the design, top and right views.



(a) Bézier joint design



(b) Spherical joint design.

Figure 2.3: Comparison of joint design angular travel: (a) Bézier design attains $\sim \pm 81^\circ$ and (b) spherical design attains $\sim \pm 21^\circ$.

CHAPTER 3

DYNAMICS

3.1 Vehicle Dynamics

3.1.1 Coordinate System

This thesis uses the equations of motion of [15]. The first step towards deriving the equations of motion is to define the rotation matrices. Using standard Euler rotations we write the rotation from world frame $\mathcal{W}$ to body frame $\mathcal{B}$ as

$$\begin{aligned} R_B^W &= R_{z,\psi} R_{y,\theta} R_{x,\psi} \\ &= \begin{bmatrix} c_\psi & s_\psi & 0 \\ -s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & -s_\theta \\ 0 & 1 & 0 \\ s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & s_\phi \\ 0 & -s_\phi & c_\phi \end{bmatrix} \\ &= \begin{bmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - c_\phi s_\psi & s_\psi s_\phi + c_\psi c_\phi s_\theta \\ c_\theta s_\psi & c_\psi c_\phi + s_\psi s_\theta s_\phi & c_\phi s_\psi s_\theta - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix}, \end{aligned} \tag{3.1}$$

where c_α and s_α are abbreviations of $\cos(\alpha)$ and $\sin(\alpha)$ for compactness. Then the Canonical mapping from $\mathcal{W}$ to $\mathcal{B}$ is

$$\begin{aligned} x_B &= R_B^W \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^\top, \\ y_B &= R_B^W \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^\top, \\ z_B &= R_B^W \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top. \end{aligned}$$

3.1.2 Translational Equations of Motion

Using Newton's laws we write the equations of motion of the aircraft in $\mathcal{W}$ as

$$\begin{aligned} m\ddot{x} &= -mgz_W + u_f z_B, \\ \ddot{x} &= -gz_W + \frac{u_f}{m} z_B, \end{aligned}$$

where $x \in \mathbb{R}^3$ is the position vector of the vehicle in $\mathcal{W}$, m is the mass of the vehicle, g is the gravitational constant, and $u_f \in \mathbb{R}$ is the thrust force exerted across all actuators.

3.1.3 Rotational Equations of Motion

The diagonal inertia tensor $\mathcal{I}$ is

$$\mathcal{I} = \begin{bmatrix} \mathcal{I}_{xx} & 0 & 0 \\ 0 & \mathcal{I}_{yy} & 0 \\ 0 & 0 & \mathcal{I}_{zz} \end{bmatrix},$$

where $\mathcal{I}_{xx}, \mathcal{I}_{yy}, \mathcal{I}_{zz} \in \mathbb{R}$ are the vehicle's scalar moments of inertia. We wish to find an expression for the angular acceleration dynamics of the quadrotor as a function of vehicle properties and the total moments acting on the body. The total moment acting on the vehicle can be written as

$$u_q = \begin{bmatrix} u_{c_1} \\ u_{c_2} \\ u_{c_3} \end{bmatrix} = u_m + u_a = \begin{bmatrix} u_\phi \\ u_\theta \\ u_\varphi \end{bmatrix} + \begin{bmatrix} \tau_{\text{roll}} \\ \tau_{\text{pitch}} \\ 0 \end{bmatrix},$$

where $[u_\phi, u_\theta, u_\varphi]^\top \in \mathbb{R}^3$ are the moments from the propellers in the roll, pitch, and yaw directions respectively, and τ is the torque induced by the manipulator in the pitch and roll directions, which will be derived in Section 3.3. We write the the relationship between input torque and rotational states as

$$u_q = \mathcal{I} \cdot \dot{\Omega}_B + \Omega_B \times (\mathcal{I} \cdot \Omega_B),$$

where $\Omega_B = [p, q, r]^\top \in \mathbb{R}^3$ is the angular velocity of the vehicle in $\mathcal{B}$. The angular accelerations in $\mathcal{B}$ are

$$\dot{\Omega}_B = \mathcal{I}^{-1}\left(-\Omega_B \times \mathcal{I} \cdot \Omega_B + u_c\right)$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{1}{\mathcal{I}_{xx}} & 0 & 0 \\ 0 & \frac{1}{\mathcal{I}_{yy}} & 0 \\ 0 & 0 & \frac{1}{\mathcal{I}_{zz}} \end{bmatrix} \left(\begin{bmatrix} (\mathcal{I}_{yy} - \mathcal{I}_{zz})\, qr + u_{c_1} \\ (\mathcal{I}_{zz} - \mathcal{I}_{xx})\, pr + u_{c_2} \\ (\mathcal{I}_{xx} - \mathcal{I}_{yy})\, pq + u_{c_3} \end{bmatrix} \right).$$

The angular acceleration about each of the body axes can be defined as

$$\begin{aligned} \dot{p} &= \frac{1}{\mathcal{I}_{xx}}\left((\mathcal{I}_{yy} - \mathcal{I}_{zz})\, qr + u_{c_1}\right), \\ \dot{q} &= \frac{1}{\mathcal{I}_{yy}}\left((\mathcal{I}_{zz} - \mathcal{I}_{xx})\, pr + u_{c_2}\right), \\ \dot{r} &= \frac{1}{\mathcal{I}_{zz}}\left((\mathcal{I}_{xx} - \mathcal{I}_{yy})\, pq + u_{c_3}\right). \end{aligned}$$

3.1.4 Motor Mapping

For a desired control signal $u = [u_f, u_\phi, u_\theta, u_\varphi]^\top$ the actuator commands $[u_1, u_2, u_3, u_4]$ are mapped as follows:

$$\begin{bmatrix} u_f \\ u_\phi \\ u_\theta \\ u_\varphi \end{bmatrix} = \underbrace{\begin{bmatrix} k_F & k_F & k_F & k_F \\ -\frac{k_F}{\sqrt{2}}L & -\frac{k_F}{\sqrt{2}}L & \frac{k_F}{\sqrt{2}}L & \frac{k_F}{\sqrt{2}}L \\ -\frac{k_F}{\sqrt{2}}L & \frac{k_F}{\sqrt{2}}L & \frac{k_F}{\sqrt{2}}L & -\frac{k_F}{\sqrt{2}}L \\ -k_Q & k_Q & -k_Q & k_Q \end{bmatrix}}_{A} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix},$$

where k_F is the thrust coefficient, k_Q is the torque coefficient, and L is length of the quadrotor arm. If the matrix A is invertible we have the solution

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} \frac{1}{k_F} & -\frac{\sqrt{2}}{k_F L} & -\frac{\sqrt{2}}{k_F L} & -\frac{1}{k_Q} \\ \frac{1}{k_F} & -\frac{\sqrt{2}}{k_F L} & \frac{\sqrt{2}}{k_F L} & \frac{1}{k_Q} \\ \frac{1}{k_F} & \frac{\sqrt{2}}{k_F L} & \frac{\sqrt{2}}{k_F L} & -\frac{1}{k_Q} \\ \frac{1}{k_F} & \frac{\sqrt{2}}{k_F L} & -\frac{\sqrt{2}}{k_F L} & \frac{1}{k_Q} \end{bmatrix} \begin{bmatrix} u_f \\ u_\phi \\ u_\theta \\ u_\varphi \end{bmatrix}.$$

3.2 Manipulator Kinematics

The kinematics of a manipulator describe the relationship between the angular position and rates of the actuators and the position and velocities of the end-effector. To develop these equations we first define the geometry. The angular position of the actuator are denoted by $\theta_i,\ i = 1,2,3$ while the coordinate of the moving platform (here moving platform is the alternate name for the end-effector, which is a standard substitution in manipulator kinematics literature) is described in standard Euclidean coordinates. Here we follow the loop-closure method, the geometric definitions are taken from [13]. First, convenient points shown in Figure 3.1 are defined; these are points of interest as they simplify the equations. Other necessary quantities are:

s_B – Side of the equilateral triangle at the base.

s_P – Side of the equilateral triangle ate the moving platform.

L – Length of the first links.

l – Length of the second links.

w_B – Distance from O_b to B_i.

u_B – Distance from O_b to the base triangle vertices.

w_P – Distance from O_p to P_i.

u_P – Distance from O_p to the moving platform triangle vertices.

Next we define the vectors:

${}^{\mathbf{B}}\mathbf{B_i}$ – Vector from O_B to each actuated axis.

${}^{\mathbf{P}}\mathbf{P_i}$ – Vector from O_P to each passive platform joint.

${}^{\mathbf{B}}\mathbf{L_i}$ – Vector described by the first link (l_1).

${}^{\mathbf{B}}\mathbf{l_i}$ – Vector described by the second link (l_2).

${}^{\mathbf{B}}\mathbf{P_P}$ – Cartesian coordinates of the platform.

The Loop-Closure vector equation has the form:

$${}^{\mathbf{B}}\mathbf{B_i} + {}^{\mathbf{B}}\mathbf{L_i} + {}^{\mathbf{B}}\mathbf{l_i} = {}^{\mathbf{B}}\mathbf{P_P} + {}^{\mathbf{P}}\mathbf{P_i}, \quad i = 1,2,3.$$

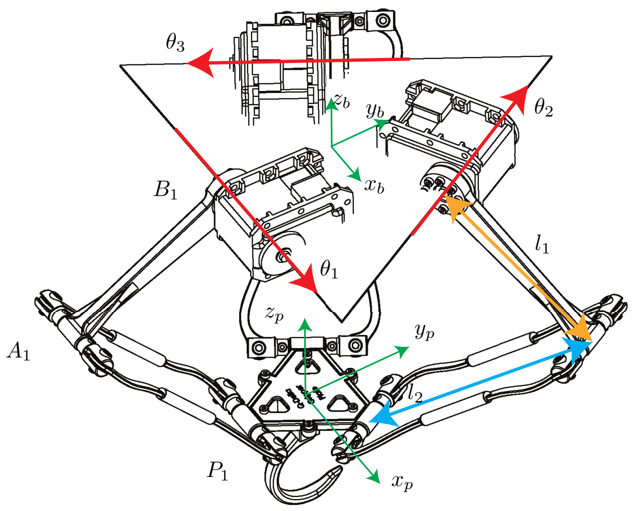


Figure 3.1: Kinematically significant points for the derivation of the equations of motion.

Fixing the second links to $l_i = l$, one gets

$$l_i = \left\| {}^{\mathbf{B}}\mathbf{l_i} \right\|_2 = \left\| {}^{\mathbf{B}}\mathbf{P_P} + {}^{\mathbf{P}}\mathbf{P_i} - {}^{\mathbf{B}}\mathbf{B_i} - {}^{\mathbf{B}}\mathbf{L_i} \right\|_2 \quad i = 1, 2, 3.$$

The constant vectors are:

$${}^{\mathbf{B}}\mathbf{B_1} = \begin{bmatrix} 0 \\ w_B \\ 0 \end{bmatrix}, \quad {}^{\mathbf{B}}\mathbf{B_2} = \begin{bmatrix} \frac{\sqrt{3}}{2} w_B \\ \frac{1}{2} w_B \\ 0 \end{bmatrix}, \quad {}^{\mathbf{B}}\mathbf{B_3} = \begin{bmatrix} -\frac{\sqrt{3}}{2} w_B \\ \frac{1}{2} w_B \\ 0 \end{bmatrix}$$

$${}^{\mathbf{P}}\mathbf{P_1} = \begin{bmatrix} 0 \\ -u_P \\ 0 \end{bmatrix}, \quad {}^{\mathbf{P}}\mathbf{P_2} = \begin{bmatrix} \frac{s_P}{2} \\ w_P \\ 0 \end{bmatrix}, \quad {}^{\mathbf{P}}\mathbf{P_3} = \begin{bmatrix} -\frac{s_P}{2} \\ w_P \\ 0 \end{bmatrix}.$$

Geometrically, we need to find the intersection between two parallel circular planes. From Figure 3.2 we can see that the intersection between these planes yields two solutions. Naturally we select the outermost solution as it provides a larger workspace. Solving the vector-loop equation yields the final relationship:

$$
\begin{aligned}
&2L(y+a)\cos\theta_1 + 2zL\sin\theta_1 \\
&\qquad + x^2 + y^2 + z^2 + a^2 + L^2 + 2ya - l^2 = 0, \\
&-L(\sqrt{3}(x+b) + y + c)\cos\theta_2 + 2zL\sin\theta_2 \\
&\qquad + x^2 + y^2 + b^2 + c^2 + L^2 + 2xb + 2yc - l^2 = 0, \\
&L(\sqrt{3}(x-b) - y - c)\cos\theta_3 + 2zL\sin\theta_3 \\
&\qquad + x^2 + y^2 + b^2 + c^2 + L^2 - 2xb + 2yc - l^2 = 0.
\end{aligned}
\tag{3.2}
$$

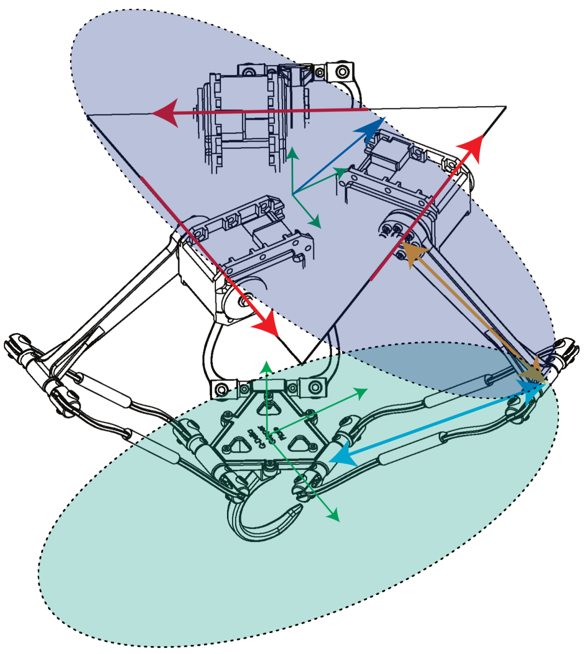


Figure 3.2: Kinematics solution: two solutions exist.

3.2.1 Inverse Kinematics

Utilizing (3.2) we can write the three independent inverse kinematics equations

$$E_i \cos\theta_i + F_i \sin\theta_i + G_i = 0, \quad i = 1, 2, 3,$$

where

$$
\begin{aligned}
E_1 &= 2L(y+a)\cos\theta_1, \\
F_1 &= 2zL\sin\theta_1, \\
G_1 &= x^2+y^2+z^2+a^2+L^2+2ya-l^2, \\
E_2 &= -L(\sqrt{3}(x+b)+y+c)\cos\theta_2, \\
F_2 &= 2zL\sin\theta_2, \\
G_2 &= x^2+y^2+b^2+c^2+L^2+2xb+2yc-l^2, \\
E_3 &= L(\sqrt{3}(x-b)-y-c)\cos\theta_3, \\
F_3 &= 2zL\sin\theta_3, \\
G_3 &= x^2+y^2+b^2+c^2+L^2-2xb+2yc-l^2, \\
a &= w_B-u_P, \\
b &= {}^{s_P}\!/_2, \\
c &= w_P-{}^{w_B}\!/_2.
\end{aligned}
$$

Solving for each θ_i yields the quadratic formula

$$\theta_i = 2\ \mathrm{atan}\left(\frac{-F_i \pm \sqrt{E_i^2+F_i^2-G_i^2}}{G_i-E_i}\right),$$

where we select the smallest root for the outermost position solution.

3.2.2 Forward Kinematics

The solution to the forward kinematics follows in a similar way. The algebraic equations are much longer (as expected from Table 2.1) and, hence, are solved using symbolic toolboxes. The algorithm is provided in Appendix C. Figure 3.3 exemplifies how the equations hold by going backward and forward between joint space and workspace.

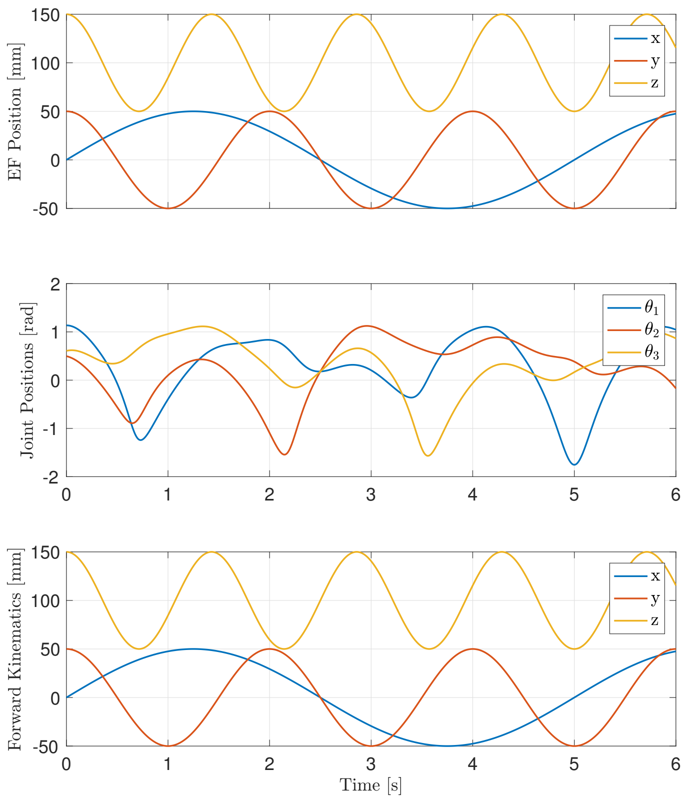


Figure 3.3: Kinematics mapping: from workspace to joint space and back.

3.2.3 Rate Analysis

The relation between the linear speed of the end-effector and the (often angular) speed of the actuators is given trough the system's Jacobian matrix:

$$\dot{p} = J(q)\dot{q}.$$

This matrix can be found by differentiating (3.2) and grouping the terms. The velocity equations are

$$\begin{aligned}
&x\dot{x} + (y+a)\dot{y} + L\dot{y}\cos\theta_1 + z\dot{z} + L\dot{z}\sin\theta_1 \\
&\qquad = L(y+a)\dot{\theta}_1\sin\theta_1 - Lz\dot{\theta}_1\cos\theta_1, && (3.3)\\
&2(x+b)\dot{x} + 2(y+c)\dot{y} - L(\sqrt{3}\dot{x}+\dot{y})\cos\theta_2 + 2z\dot{z} + 2L\dot{z}\sin\theta_2 \\
&\qquad = -L(\sqrt{3}(x+b)+y+c)\dot{\theta}_2\sin\theta_2 - 2Lz\dot{\theta}_2\cos\theta_2, && (3.4)\\
&2(x-b)\dot{x} + 2(y+c)\dot{y} + L(\sqrt{3}\dot{x}-\dot{y})\cos\theta_3 + 2z\dot{z} + 2L\dot{z}\sin\theta_3 \\
&\qquad = L(\sqrt{3}(x+b)-y-c)\dot{\theta}_3\sin\theta_2 - 2Lz\dot{\theta}_2\cos\theta_3, && (3.5)
\end{aligned}$$

which can be written in matrix form as

$$\underbrace{\begin{bmatrix} x & y+a+L\cos\theta_1 & z+L\sin\theta_1 \\ 2(x+b)-\sqrt{3}L\cos\theta_2 & 2(y+c)-L\cos\theta_2 & 2(z+L\sin\theta_2) \\ 2(x-b)+\sqrt{3}L\cos\theta_3 & 2(y+c)-L\cos\theta_3 & 2(z+L\sin\theta_3) \end{bmatrix}}_{J_p} \dot{p}$$

$$= \underbrace{\begin{bmatrix} J_{q_1} & 0 & 0 \\ 0 & J_{q_2} & 0 \\ 0 & 0 & J_{q_3} \end{bmatrix}}_{J_q} \dot{q},$$

where

$$\begin{aligned} J_{q_1} &= L((y+a)\sin\theta_1 - z\cos\theta_1), \\ J_{q_2} &= -L((\sqrt{3}(x+b)+y+c)\sin\theta_2 + 2z\cos\theta_2), \\ J_{q_3} &= L((\sqrt{3}(x-b)-y-c)\sin\theta_3 - 2z\cos\theta_3). \end{aligned}$$

We can write the Jacobian as

$$J(q) = J_p^{-1} J_q, \tag{3.6}$$

where $q = (\theta_1, \theta_2, \theta_3)$.

3.3 Manipulator Dynamics

3.3.1 Computed Torque

In robotic manipulator control literature, the Computed Torque Control method relies on calculating the expected torque generated by the dynamics of the manipulator, and feeding it forward to the actuators via a torque controller, augmenting the position PID controller. Since the Delta-type robot is widely used in the industry, these equations are well known. For this application, however, some changes have to be made so that the result reflects the torque generated at the actuators fixture to the base, instead of on its rotor. We mainly follow results of [16] which are modified as follows: For a given end-effector trajectory $\mathbf{p} = (p, \dot{p}, \ddot{p})$ and actuator topology

$\phi = (\phi_1, \phi_2, \phi_3) = (-\pi/2, \pi/6, 5\pi/6)$ the three torques are computed as

$$\begin{aligned}
\tau_1 &= (I_1 + m_2 l_1^2)\ddot{\theta}_1 - (m_1 l_{1c} + m_2 l_1) g_c \cos(\theta_1) - \\
&\quad 2 l_1 \lambda_1 [(p_1 \cos(\phi_1) + p_2 \sin(\phi_1) + b - a)\sin(\theta_1) - p_3 \cos(\theta_1)], \\
\tau_2 &= (I_1 + m_2 l_1^2)\ddot{\theta}_2 - (m_1 l_{1c} + m_2 l_1) g_c \cos(\theta_2) - \\
&\quad 2 l_1 \lambda_2 [(p_1 \cos(\phi_2) + p_2 \sin(\phi_2) + b - a)\sin(\theta_2) - p_3 \cos(\theta_2)], \\
\tau_3 &= (I_1 + m_2 l_1^2)\ddot{\theta}_3 - (m_1 l_{1c} + m_2 l_1) g_c \cos(\theta_3) - \\
&\quad 2 l_1 \lambda_3 [(p_1 \cos(\phi_3) + p_2 \sin(\phi_3) + b - a)\sin(\theta_3) - p_3 \cos(\theta_3)],
\end{aligned} \tag{3.7}$$

where the Lagrangian multipliers λ_i are obtained by solving the system

$$\begin{aligned}
&2\sum_{i=1}^{3} \lambda_i (p_1 + b\cos(\phi_i) - a\cos(\phi_i) - l_1 \cos(\phi_i)\cos(\theta_i)) = (m_p + 3m_2)\ddot{p}_1, \\
&2\sum_{i=1}^{3} \lambda_i (p_2 + b\sin(\phi_i) - a\sin(\phi_i) - l_1 \sin(\phi_i)\cos(\theta_i)) = (m_p + 3m_2)\ddot{p}_2, \\
&2\sum_{i=1}^{3} \lambda_i (p_3 - l_1 \sin(\theta_i) = (m_p + 3m_2)(\ddot{p}_3 - g_c).
\end{aligned} \tag{3.8}$$

The explicit symbolic solution to the system above exists, but it is extraordinarily long and, thus, omitted. The actuator topology ϕ represents the position of each actuator and the joint-space trajectory $(\theta_i, \ddot{\theta}_i)$ is obtained by applying inverse kinematics and passing it through the high fidelity actuator model. The remaining parameters are described in Table 3.1.

a	Radius of the fixed base (center of the base to actuator)
b	Radius of the moving platform (center of the platform to joint)
l_i	Length of link i
m_1	Mass of link 1
m_2	Half of mass of link 2
m_p	Mass of the moving platform
l_{1c}	Length of the center of mass of link 1
I_1	Moment of inertia of link 1
g_c	Gravity component orthogonal to the moving platform

Table 3.1: Kinematic parameters and mass properties of the Delta manipulator.

The transformation to the $x - y$ plane of the aircraft is then done by projecting each actuator torque into the $x - z$ and $y - z$ planes following the topology ϕ as seen in Figure 3.4. The final torque mapping is

$$\hat{\tau} = \begin{bmatrix} \tau_x \\ \tau_y \end{bmatrix} = \begin{bmatrix} \sum_1^3 \tau_{i_x} \\ \sum_1^3 \tau_{i_y} \end{bmatrix} \tag{3.9}$$

where the operator $\hat{}$ denotes estimation and $\hat{\tau}$ is the estimated torque vector

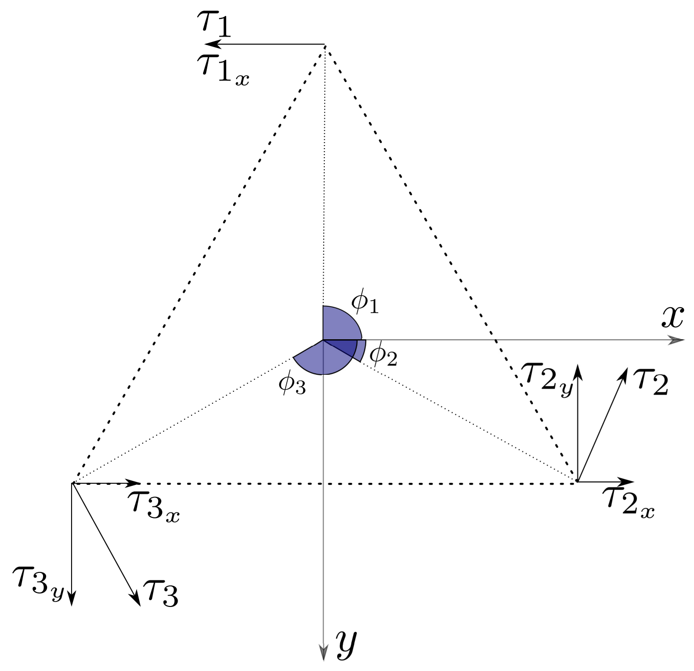


Figure 3.4: Topology of actuators and directions of torque vectors

in the $x - y$ plane. No torque exists on the z axis by construction.

Matrix Form

It is often convenient to have the equations in matrix form for computational purposes. First we rewrite (3.8) as

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = A^{-1} \begin{bmatrix} (m_p + 3m_2)\ddot{p}_1 \\ (m_p + 3m_2)\ddot{p}_2 \\ (m_p + 3m_2)(\ddot{p}_3 - g_c) \end{bmatrix},$$

where

$$A = \begin{bmatrix} (p_1+bc_{\phi_1}-ac_{\phi_1}-l_1c_{\phi_1}c_{\theta_1}) & (p_1+bc_{\phi_2}-ac_{\phi_2}-l_1c_{\phi_2}c_{\theta_2}) & (p_1+bc_{\phi_3}-ac_{\phi_3}-l_1c_{\phi_3}c_{\theta_3}) \\ (p_2+bs_{\phi_1}-as_{\phi_1}-l_1s_{\phi_1}c_{\theta_1}) & (p_2+bc_{\phi_2}-ac_{\phi_2}-l_1c_{\phi_2}c_{\theta_2}) & (p_2+bc_{\phi_3}-ac_{\phi_3}-l_1c_{\phi_3}c_{\theta_3}) \\ (p_3-l_1s_{\theta_1}) & (p_3-l_1s_{\theta_2}) & (p_3-l_1s_{\theta_3}) \end{bmatrix}.$$

Rewriting (3.7) yields

$$\tau = Q(\ddot{\theta}) - G(\theta) - M(\theta, \ddot{p}),$$

where

$$\begin{aligned} Q &= \begin{bmatrix} (I_1 + m_2 l_1^2)\ddot{\theta}_1 & 0 & 0 \\ 0 & (I_1 + m_2 l_1^2)\ddot{\theta}_2 & 0 \\ 0 & 0 & (I_1 + m_2 l_1^2)\ddot{\theta}_3 \end{bmatrix}, \\ G(\theta) &= \begin{bmatrix} g_{11} & 0 & 0 \\ 0 & g_{22} & 0 \\ 0 & 0 & g_{33} \end{bmatrix}, \\ M(\theta, \ddot{p}) &= L(\theta) \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix}, \\ L(\theta) &= \begin{bmatrix} l_{11} & 0 & 0 \\ 0 & l_{22} & 0 \\ 0 & 0 & l_{33} \end{bmatrix}, \\ g_{ii} &= (m_1 l_{1c} + m_2 l_1) g_c \cos(\theta_i), \\ l_{ii} &= 2l_1[(p_1 \cos(\phi_i) + p_2 \sin(\phi_i) + b - a)\sin(\theta_i) - p_3 \cos(\theta_i)]. \end{aligned}$$

Note that as long as the manipulator is not in a singularity position, A is invertible and, thus, a solution exists.

3.3.2 Inertia Estimation

The goal of inertia estimation is to cancel out the high disturbance-inducing masses of the system. Utilizing the Jacobian and the statics/dynamics duality we have

$$\tau_I = J(q)^T F_I, \tag{3.10}$$

where τ_I are the torques necessary to cancel the extra inertia caused by the platform, tool, and second links, $J(q)$ is the Jacobian matrix defined in (3.6), and F_I is the force the manipulator needs to exert at the end-effector to keep the platform in place. To apply the foregoing relation, we first estimate the vehicle rates and calculate the force necessary to keep the payload in place (i.e. not moving with respect to the quadcopter frame of reference), which follows from

$$\vec{F}_I = (m_p + m_l) \cdot \vec{a}_p \Rightarrow \tau_I = (m_p + m_l) J^T \mathbf{a_p},$$

where $\mathbf{a_p}$ is the platform acceleration with respect to the global frame. The last step is feedforwarding τ_I along with the previously defined ones.

3.4 End-effector Position Hold

This section concludes the chapter. The challenge is to account for the yaw axis and hold a global end-effector desired position. We start by writing the coordinate systems and position vectors as in Figure 3.5. In this solution, $\mathcal{W}$ is the world frame, $\mathcal{B}$ the quadcopter reference frame, and $\mathcal{E}$ is the end-effector reference frame. The vectors $\vec{r}_q$, $\vec{r}_p$ and $\vec{r}_e$ are the quadcopter position in $\mathcal{W}$, the end-effector position in $\mathcal{B}$, and the end-effector position in $\mathcal{W}$, respectively. Then we have vector-loop equation

$$\vec{r}_q + R_z \vec{r}_p = \vec{r}_e.$$

The rotation matrix about the z axis R_z represents the effect of the yaw angle φ. Solving for $\vec{r}_e$ and substituting the coordinates yields

$$\mathbf{x}_e = R_z^{-1} (\mathbf{x}_d - \mathbf{x}_q) \tag{3.11}$$

where $\mathbf{x}_e$ is the end-effector position command, $\mathbf{x}_d$ is the desired end-effector position in the global frame and $\mathbf{x}_q$ is the quadcopter position in the global frame. Additionally:

$$R_z(\varphi) = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The final step is to pass $\mathbf{x}_e$ trough the inverse kinematics mapping yielding the desired configuration θ_e and use it as the joints command signal. Proceeding the same way we can extend to full pitch-roll-yaw compensation by using the complete Euler rotation matrix:

$$\mathbf{x}_e = \left(R_B^W\right)^{-1}\left(\mathbf{x}_d - \mathbf{x}_q\right),$$

where R_B^W is defined in (3.1).

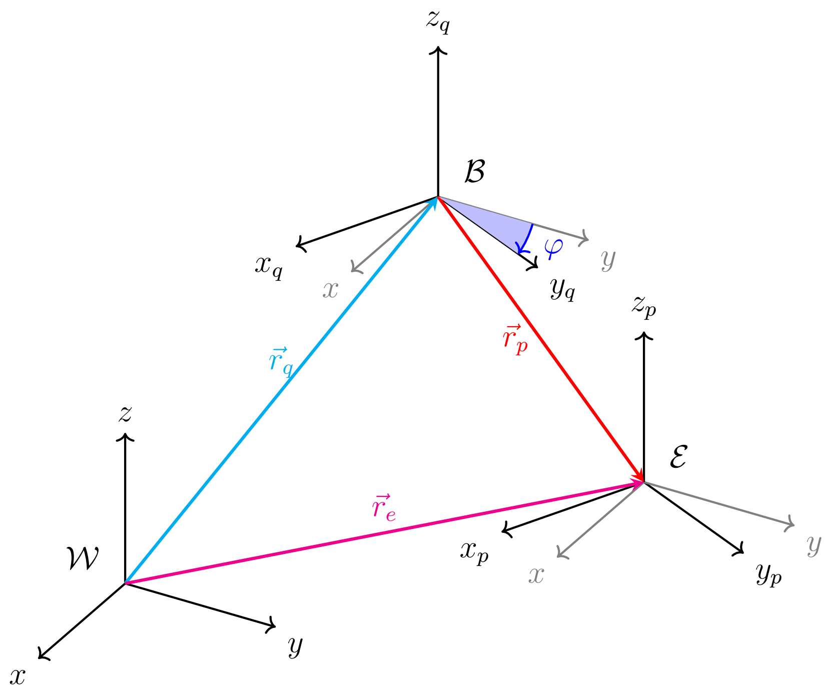


Figure 3.5: Coordinates systems.

CHAPTER 4

CONTROLLER DESIGN

In this section, the design of control laws is discussed. This process is using a realistic model of the system, where all signals flow as they would on the aircraft. Throughout the rest of this paper, the generalized coordinates of the system are $Q(t) = [\varphi_1(t), \varphi_2(t), \theta_1(t), \theta_2(t), \theta_3(t)]$, where $[\varphi_1(t), \varphi_2(t)] = \varphi(t)$ are the pitch and roll angles respectively and $[\theta_1(t), \theta_2(t), \theta_3(t)] = \theta(t)$ are the manipulator actuators angular position. The subscript c indicates a command (i.e. θ_{1_c} indicates the angular position command for the first manipulator actuator). We first state the problem formulation:

Problem Formulation 4.1 *(Stabilizing controller for a manipulation UAS) Given a quadcopter equipped with a Delta-type parallel manipulator and an end-effector trajectory* $\mathbf{p}$*, find a control signal* $\mathbf{u}$ *that stabilizes the vehicle.*

Assumptions:

1. *The trajectory* $\mathbf{p}$ *remains strictly inside the workspace.*
2. *Throughout the trajectory* $\mathbf{p}$ *the payload never exceeds the allowable maximum.*
3. *All manipulator actuators don't saturate torque, speed or acceleration.*

The assumptions here are mainly in place to not allow non-feasible trajectories. Assumption 1 stops the manipulator from reaching singularity positions, Assumption 2 guarantees that at every point in the trajectory stable flight is attainable, and Assumption 3 restricts the trajectories to the ones that the manipulator can perform.

4.1 General Control Scheme

Figure 4.1 shows the general control scheme of the system. The commanded joint-space trajectory θ_c and commanded pitch and roll attitudes φ_c are fed to the system. The manipulator dynamics generate the disturbance τ_m while the Computed Torque Feedforward (CTF) algorithm estimates this disturbance and generates a control signal consistent with the vehicle dynamics. The baseline controller is a cascaded PI/PID (rate PI and attitude PID) controller which is augmented with the $\mathcal{L}_1$ adaptive controller.

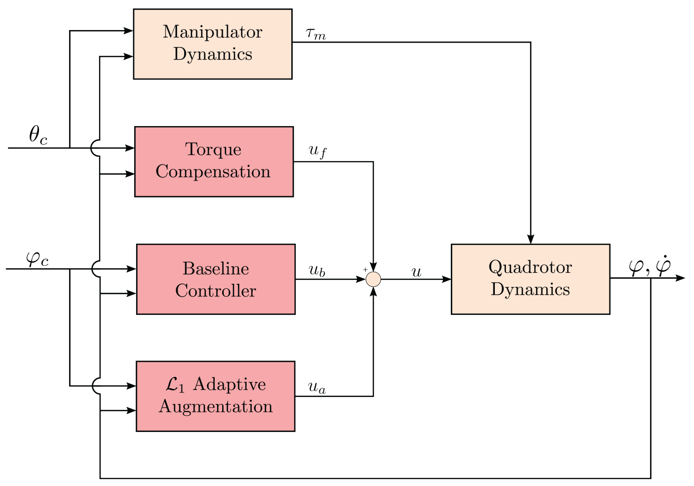


Figure 4.1: General control scheme

4.2 Baseline Controller

The baseline controller follows a cascaded PI/PID attitude structure as shown below:

$$u_{b_i}(t) = k_{P1}\left(\dot{\varphi}_{i_c}(t) - \dot{\varphi}_i(t)\right) + K_{I1}\int_0^t \left(\dot{\varphi}_{i_c}(t) - \dot{\varphi}_i(t)\right) d\tau$$

where

$$\dot{\varphi}_{i_c}(t) = k_{P2}\left(\varphi_{i_c}(t) - \varphi_i(t)\right) + K_{I2}\int_0^t \left(\varphi_{i_c}(t) - \varphi_i(t)\right) d\tau + k_D\varphi_i(t)$$

and k_{P1}, k_{I1}, k_{P2}, k_{I2}, $k_D \in \mathbb{R}$ are manually tuned control gains. Here, $u_b = [u_{b_1}\ u_{b_2}]$ is the vector of the baseline pitch and roll control signals respectively.

4.3 $\mathcal{L}_1$ Adaptive Control Augmentation

The aerial manipulator can pick up a large variety of objects, varying in mass, inertia, density, shape, etc. With a flight controller augmented only with the feedforward torque compensation of the manipulator, there is no way to account for the induced torques and forces from these uncertain payloads. To reject uncertainties introduced by unknown payloads, the $\mathcal{L}_1$ adaptive control structure is chosen as a robust augmentation.

An $\mathcal{L}_1$ adaptive control augmentation with piecewise constant adaptation law from [17] is considered. Here we implement this structure on the pitch and roll axis separately; this is possible due to the decoupling between the x and y torque axis of the manipulator. Let $x_{I1}(t)$ and $x_{I2}(t)$ denote the states of the integrator in the rate and attitude in each of the loops of the baseline controller respectively. Then, the rotational equation of motion of the quadrotor with the $\mathcal{L}_1$ augmentation can be expressed as

$$\begin{aligned}\dot{x}(t) &= A_m x(t) + B_r r(t) + B_m \left(u_a(t) + f_1(t,x)\right) + B_{um} f_2(t,x),\\ y(t) &= C_m x(t),\\ x(0) &= x_0,\end{aligned} \tag{4.1}$$

where $x(t) = [\varphi_1(t), \dot{\varphi}_1(t), x_{I1}(t), x_{I2}(t)]^\top$ is the vector of the system states, $r(t) = \varphi_{i_c}(t)$ is the reference attitude command, $f_1(t,x) : \mathbb{R} \times \mathbb{R}^4 \to \mathbb{R}$ is a nonlinear function containing information on the residual of the feedforward torque compensation for the disturbance torque from the manipulator and the matched uncertainty, $f_2(t,x) : \mathbb{R} \times \mathbb{R}^4 \to \mathbb{R}^3$ is a nonlinear function representing additional modeling uncertainty, $A_m \in \mathbb{R}^{4\times 4}$ is a known Hurwitz matrix defining the desired system dynamics, $B_r \in \mathbb{R}^{4\times 1}$ is the known command matrix, $B_m \in \mathbb{R}^{4\times 1}$ is the known control matrix, $C_m \in \mathbb{R}^{1\times 4}$ is a known full-rank constant matrix, and $B_{um} \in \mathbb{R}^{4\times 3}$ is a matrix such that $B_m^\top B_{um} = 0$ and $[B_m \; B_{um}]$ has full rank. The product $B_{um} f_2(t,x)$ represents the unmatched uncertainty. The matrices A_m, B_r, and B_m can be written as

$$A_m = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k_{P1}k_{P2}}{J_0} & \frac{k_{P1}k_D}{J_0} & \frac{k_{I1}}{J_0} & \frac{k_{P1}k_{I2}}{J_0} \\ -k_{P2} & k_D - 1 & 0 & k_{I2} \\ -1 & 0 & 0 & 0 \end{bmatrix},$$

$$B_r = \begin{bmatrix} 0 \\ \frac{k_{P1}k_{P2}}{J_0} \\ k_{P2} \\ 1 \end{bmatrix}, \quad B_m = \begin{bmatrix} 0 \\ \frac{1}{J_0} \\ 0 \\ 0 \end{bmatrix}.$$

For the system given in (4.1), the elements of the $\mathcal{L}_1$ adaptive controller are given below.

4.3.1 State Predictor

Taking the same structure as the system in 4.1, the state predictor is given by

$$\begin{aligned} \dot{\hat{x}}(t) &= A_m \hat{x}(t) + B_r r(t) + B_m \left(u_a(t) + \hat{\sigma}_1(t) \right) + B_{um} \hat{\sigma}_2(t), \\ \hat{x}(0) &= x_0, \end{aligned}$$

where $\hat{x}(t)$ is the predictor state, $\hat{\sigma}_1(t) \in \mathbb{R}$ and $\hat{\sigma}_2(t) \in \mathbb{R}^3$ are the estimates of the nonlinear functions $f_1(\cdot)$ and $f_2(\cdot)$ respectively.

4.3.2 Adaptation Law

Given an adaptation rate $T_s > 0$, the estimates $\hat{\sigma}_1(t)$ and $\hat{\sigma}_2(t)$ are updated according to the following piecewise constant adaptation law:

$$\begin{aligned} \begin{bmatrix} \hat{\sigma}_1(t) \\ \hat{\sigma}_2(t) \end{bmatrix} &= \begin{bmatrix} \hat{\sigma}_1(iT_s) \\ \hat{\sigma}_2(iT_s) \end{bmatrix}, \quad t \in [iT_s, (i+1)T_s) \\ \begin{bmatrix} \hat{\sigma}_1(iT_s) \\ \hat{\sigma}_2(iT_s) \end{bmatrix} &= - \begin{bmatrix} 1 & 0 \\ 0 & \mathbb{I}_3 \end{bmatrix} \left[B_m \; B_{um} \right]^{-1} \Phi^{-1}(T_s) e^{A_m T_s} \tilde{x}(iT_s), \\ & \qquad i = 0, 1, 2, 3, ..., \end{aligned} \tag{4.2}$$

where

$$\Phi^{-1}(T_s) = A_m^{-1}\left(e^{A_m T_s} - \mathbb{I}_4\right)$$

and $\tilde{x}(t) = \hat{x}(t) - x(t)$ is the state prediction error.

4.3.3 Control Law

The control law is generated as the output of the following system:

$$u_a(s) = -k_a D(s)\hat{\eta}(s), \tag{4.3}$$

where $\hat{\eta}(s)$ is the Laplace transform of the signal

$$\hat{\eta}(t) \triangleq u_a(t) + \hat{\eta}_1(t) + \hat{\eta}_2(t)$$

with $\hat{\eta}_1(t) = \hat{\sigma}_1(t)$ and $\hat{\eta}_2(s) = H_1^{-1}(s)H_2(s)\hat{\sigma}_2(s)$ and

$$\begin{aligned} H_1(s) &= C_m(s\mathbb{I} - A_m)^{-1}B_m, \\ H_2(s) &= C_m(s\mathbb{I} - A_m)^{-1}B_{um}. \end{aligned}$$

Here k_a is a feedback gain and $D(s)$ is a strictly proper transfer function, which lead to a strictly proper stable

$$C(s) \triangleq \frac{k_a D(s)}{1 + k_a D(s)},$$

where $C(s)$ is a low pass filter with DC gain $C(0) = 1$.

4.4 Solution Steps for the Stabilizing Flight Controller

We can summarize the steps to solve the stabilization problem as:

1. Find the joint-space trajectory (inverse kinematics, section 3.2.1).
2. Estimate $\ddot{\theta}_i$ with the actuator model.
3. Solve the Lagrangian Multipliers and the torques (section 3.3).
4. Feedforward the torques to the flight controller.

5. Augment with $\mathcal{L}_1$ to account for unknown payloads and model discrepancies.

Note that the quadcopter trajectory is implicitly included in the inertia cancellation algorithm and in the direction of the gravity vector. In addition, we assume that being a difficult task to perform with a perfect controller, the quadcopter trajectory is composed of relatively small angles. The next chapter discusses the generation of these trajectories.

CHAPTER 5

OPTIMAL TRAJECTORY GENERATION FOR REDUNDANT AERIAL MANIPULATION SYSTEMS

This chapter discusses a novel approach to task planning for these kinds of UAS based on real-world proven control techniques of redundant manipulators. The idea revolves around exploring the additional DoF's provided by the quadcopter and transform the system into a redundant manipulator, that is, the mapping from joint-space to workspace becomes onto. The implications are that now there are infinitely many solutions for every desired end-effector position and traditional workspace trajectory generation techniques fail. In this case, we propose a modification to optimal rate control methods so that these can be used on a UAS. First, we state the problem formulation:

Problem Formulation 5.1 *(Optimal Trajectory Generation for a Redundant Manipulation UAS) Find a minimum energy joint-space* ***trajectory*** *$\dot{q}$ that reaches a desired end-effector destination while satisfying* ***mission-specific*** *constraints. This is equivalent to the minimization problem*

$$\begin{array}{rl} \min\limits_{\dot{q}} & \|\dot{q}\|_2^2 \\ s.t. & \textit{Mission constraints.} \end{array} \tag{5.1}$$

Assumptions:

1. *A stabilizing controller exists that tracks the solution of* (5.1).
2. *The yaw angle always remains aligned with the global frame $\mathcal{W}$.*

The assumptions are realistic given the results of the preceding chapter. The most traditional constraints are spatial (desired end-effector location), mechanical (maximum joint travel) and energy-weighted (control cost of some joints over the others). The careful reader will note that this is a least-squares minimization problem. As another remark, we note that $\|\dot{q}\|_2^2$ is not exactly energy, but only proportional to it. Later in this chapter, there will be mathematical relations developed to better translate this problem into *real* minimum energy. Before continuing it is also important to note what is being optimized. The objective of the solution of 5.1 is to generate the *next* joint-space waypoint from the current location. The choice of this point takes into account the mission and, most notably, how expensive it is to move each joint. The optimality of the solution exists in the sense that from the infinite set of joint-space trajectories (from the redundancy property) to get to the next waypoint we're choosing the one which minimizes the norm of the joint-space velocity, based on user-defined weights. We can now show how to bridge the gap between redundant manipulator trajectory planning theory and *any* aerial redundant manipulator system.

5.1 Realization as a Unified Manipulation System

In order to proceed with the concept we, first, need to realize the UAS as single manipulation system. The idea is relatively natural: any quadcopter has four DoF which can be attained in steady-state: a position in $SE(3)$ and the yaw angle, which we denote as the generalized coordinates (x_q, y_q, z_q, ψ). Combining those with the degrees-of-freedom of a euclidean position-oriented manipulator we will have a redundant system. Since the yaw angle does not overlap with the manipulator coordinates it can be commanded externally and, thus, won't be a part of the unified system. Let $s = (x_q, y_q, z_q)$ be the quadcopter position and p be position of the manipulator's moving platform. We have

$$\dot{p} = J(q)\dot{q}$$

as the mapping from joint rates to position rates. Let $\mathbf{x} \in SE(3)$ be the global end-effector position and $\mathbf{q} = (s, q)$ be the joints of the unified system. Then

$$\dot{\mathbf{x}} = J_u(\mathbf{q})\dot{\mathbf{q}},$$

where $J_u(\mathbf{q})$ is the Jacobian of the unified system:

$$J_u(\mathbf{q}) = \begin{bmatrix} I_{n\times n} & J(q) \end{bmatrix} \tag{5.2}$$

and n is the dimension of the workspace (usually 2 or 3). This realization opens many doors for trajectory generation and tracking, which are explored throughout the rest of this chapter. Note that the non-squareness of J_u makes a kinematics-based approach to trajectory generation impossible.

5.2 Exact Unconstrained Solution

This solution (together with many others discussed in this chapter) is presented in [18]. For a given desired end-effector velocity we have that the exact solution is

$$\dot{\mathbf{q}} = J_u^\dagger \dot{\mathbf{x}}, \tag{5.3}$$

where $J_u^\dagger$ is the Moore-Penrose pseudoinverse and we omit its dependency on $\mathbf{q}$ for the benefit of compactness. In addition, (5.3) is a solution to the unconstrained version of (5.1). If J_u is non-singular, then $J_u^\dagger = J_u^T(J_u J_u^T)^{-1}$. This compact result reflects the efficiency one can attain by moving away from position-based methods. It is still required to account for singularity positions and mission constraints, as shown in the following sections.

5.3 Spatial Constraints

A basic constraint is a final end-effector position. This constraint is formulated as a running cost by setting $\dot{\mathbf{x}}$ to be proportional to the position error with a saturation function:

$$\dot{\mathbf{x}} = \mathrm{sat}(k_x(\mathbf{x_d} - \mathbf{x_e}), \delta), \tag{5.4}$$

where δ is a column vector containing the maximum speed of each joint, $\mathbf{x_d}$ and $\mathbf{x_e}$ are the desired and current end-effector positions, respectively, and k_x is manually tuned proportional gain. The saturation function convention throughout this thesis is

$$\text{sat}(\alpha, \beta) = \begin{cases} \beta, & \alpha > \beta \\ \alpha, & -\beta \geq \alpha \geq \beta \\ -\beta, & \alpha < -\beta \end{cases},$$

where in the case of the arguments being vectors the operations are evaluated element-wise. Substituting (5.4) into any of the solutions presented in this chapter will generate a trajectory which asymptotically converges to the desired position.

5.4 Singularity Avoidance

As the system approaches a singularity (which may happen for a variety of reasons), the norm of the joint velocities becomes arbitrarily large [1]. To avoid this undesirable behavior a manually tuned damping factor λ is introduced, which yields the result seen in Figure 5.1.

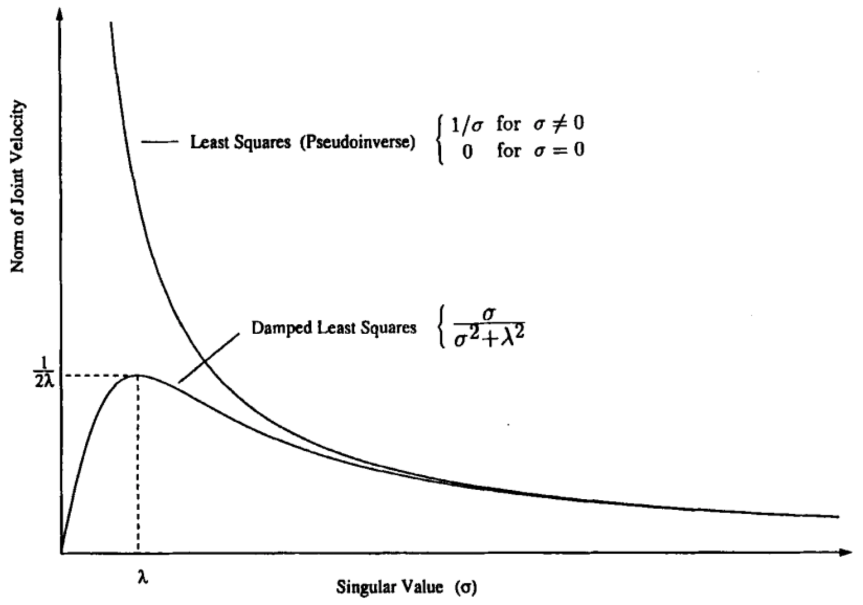


Figure 5.1: Effect of damping on the norm of the joint velocity where σ is a singular value of J_u [1].

The damped solution is

$$\dot{\mathbf{q}} = \left(J_u^T J_u + \lambda^2 I\right)^{-1} J_u^T \dot{\mathbf{x}}. \tag{5.5}$$

From (5.5) it becomes clear that the constant diagonal matrix will always make the bracketed term invertible, seeing as $J_u^T J_u \succ 0$ is symmetric.

5.5 Mechanical Constraints

In an aerial manipulation system it is often required to satisfy two forms of mechanical constraints: (i) the arm cannot attain a configuration which will impede proper function of the propellers, and (ii) the quadcopter cannot get too close to certain objects while the end-effector can. The formulation (5.2) allows us to satisfy these constraints in one step. First, we write the augmented form of (5.5):

$$\dot{\mathbf{q}} = \left(J_u^T J_u + W_{LA}(\mathbf{q}) + \lambda^2 I\right)^{-1} J_u^T \dot{\mathbf{x}}, \tag{5.6}$$

where an example weight matrix for this thesis' UAS is

$$W_{LA}(\mathbf{q}) = \begin{bmatrix} L_1(q_1) & 0 & 0 & 0 & 0 & 0 \\ 0 & L_2(q_2) & 0 & 0 & 0 & 0 \\ 0 & 0 & L_3(q_3) & 0 & 0 & 0 \\ 0 & 0 & 0 & L_4(q_4) & 0 & 0 \\ 0 & 0 & 0 & 0 & L_5(q_5) & 0 \\ 0 & 0 & 0 & 0 & 0 & L_6(q_6) \end{bmatrix} \tag{5.7}$$

and $L_i(q_i)$ represents a potential function for the i-th joint. The potential function is manually tuned according to the following guidelines:

- A zero value allows the joint to move freely.
- A high value is equivalent to introducing a high damping factor and, thus, tends to stop the joint.
- A small negative value repels the joint in the opposite direction to the unconstrained optimal.
- A large negative value is equivalent to a high value.

The reasoning behind these rules is obvious. The potential function should be smooth and usually it suffices to have an exponential form.

5.6 Energy-Weighted Solution

This section aims to allow the mission planner to increase the cost of moving one joint over another. Not only this is a way of better approximating the minimization problem to *real* energy but also enables a high-level controller to decide when it is beneficial or necessary to move the manipulator, as any such movement introduces undesirable dynamics.

A simple modification to (5.6) will achieve the desired results:

$$\dot{\mathbf{q}} = \left(J_u^T J_u + W_{LA}(\mathbf{q}) + \lambda^2 W_A\right)^{-1} J_u^T \dot{\mathbf{x}}, \tag{5.8}$$

where an example weight matrix for this thesis' UAS is

$$W_A = \begin{bmatrix} A_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & A_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & A_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & A_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & A_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & A_6 \end{bmatrix}. \tag{5.9}$$

Here, A_i represents the actuation cost for the i-th joint. If the mission planner wants an exact minimum energy solution the weights will be the moving mass or inertia (depending on type) of each link. Similarly, a high weight will be equivalent to aggressively damping the joint automatically reducing its speed. It is worth to state the remark that excessive damping applied to too many joints will cause the tracking error per step to increase, slowing the convergence rate to the desired end-effector position.

5.7 Additional Task Constraints

The goal here is to modify the cost function in order to to reach a desired manipulator configuration goal. This is equivalent to writing

$$\dot{\mathbf{q}} = \left(J_u^T J_u + W_{LA}(\mathbf{q}) + J_T^T W_T J_T + \lambda^2 W_A\right)^{-1} \left(J_u^T \dot{\mathbf{x}} + J_T^T W_T \dot{\mathbf{q}}_T\right), \quad (5.10)$$

where an example weight matrix for this thesis' UAS is

$$W_T = \begin{bmatrix} T_1 & 0 & 0 \\ 0 & T_2 & 0 \\ 0 & 0 & T_3 \end{bmatrix} \quad (5.11)$$

$$J_T = \begin{bmatrix} 0_{3\times 3} & I_{3\times 3} \end{bmatrix}, \quad (5.12)$$

and T_i represents the weight given for each joint's additional task. For example: if it is paramount that θ_1 reaches a desired goal, then $T_1 >> T_i,\ i = 2, 3$. Similar to (5.4) we define the additional task's desired velocity as

$$\dot{\mathbf{q}}_T = \text{sat}(k_q(\mathbf{q}_{a_d} - \mathbf{q}_a), \zeta), \quad (5.13)$$

where ζ is the column vector containing the manipulators' joints maximum speeds, $\mathbf{q_{a_d}}$ is the desired final configuration, $\mathbf{q_a}$ is the current configuration, and k_q is a manually tuned proportional gain.

CHAPTER 6

SIMULATION RESULTS

6.1 Stabilizing Controller

This section discusses simulation results for the stabilizing controller which solves Problem 4.1.

6.1.1 Simulation Setup

The simulation scenario is a simple movement which pulls the end-effector from 80mm to a position aligned with the z axis of the center of mass of the quadcopter. The choice of scenario is motivated to compare this prototype with the one developed in [11]. The simulation scenario is depicted in Figure 6.1. The simulation environment is a realistic simulator which includes the battery, motor, atmosphere, sensor, and radio models, among other details, which add realistic uncertainties and noise to the system. Additionally, Figure 6.2 depicts the workspace and joint space trajectories.

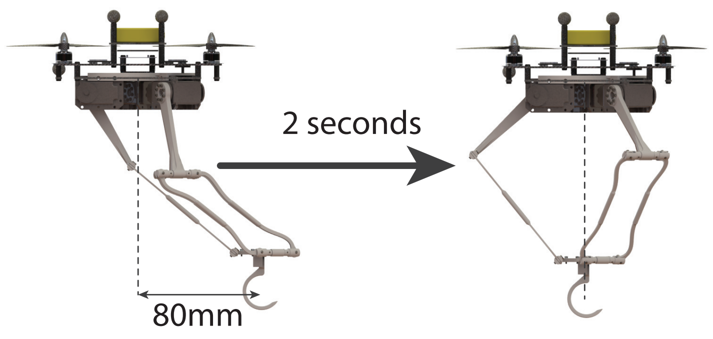


Figure 6.1: Simulation scenario.

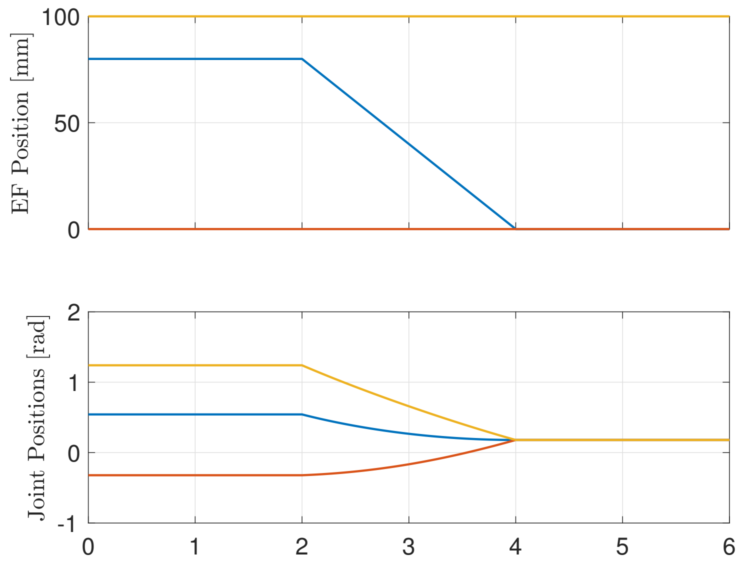


Figure 6.2: Simulation scenario kinematic trajectories.

6.1.2 Computed Torque Feedforward (CTF)

Here we first simulate the system without any payload. As we can see from Figure 6.3, in the absence of a payload the uncertainties are minimal and, thus, we attain satisfactory performance. This does not reflect reality, however. Figure 6.4 shows how the addition of a payload increases the model uncertainty dramatically and, thus, the performance degrades.

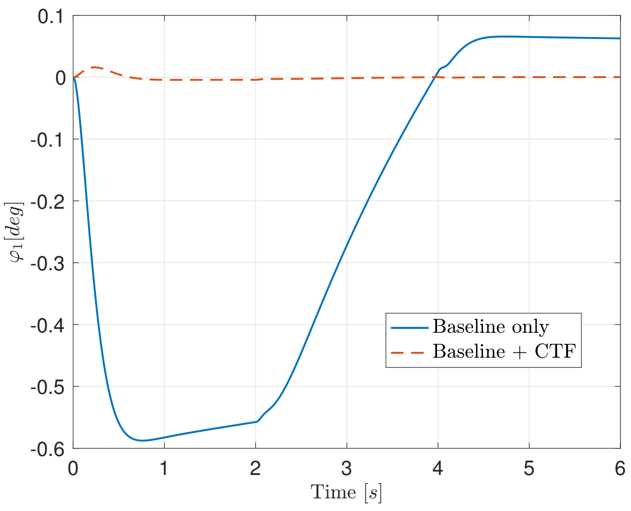


Figure 6.3: Simulation: baseline versus CTF, no payload.

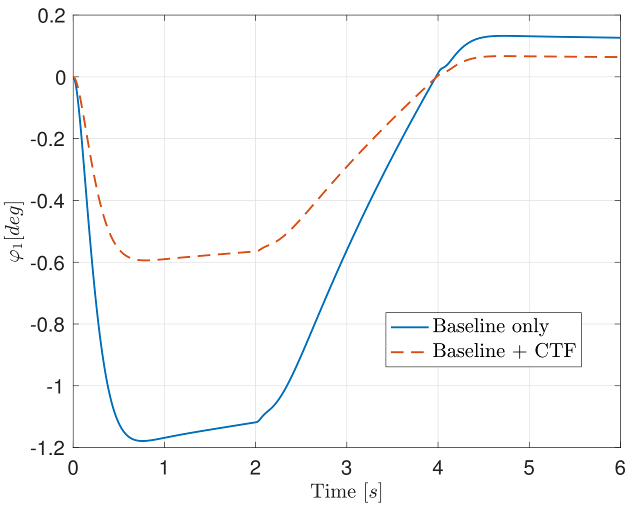


Figure 6.4: Simulation: baseline versus CTF, 25g payload.

6.1.3 $\mathcal{L}_1$ Adaptive Controller

The addition of the $\mathcal{L}_1$ adaptive controller proves to be necessary to account for model uncertainties. Figure 6.5 compares the different levels of compensation. Figure 6.6 includes sensor noise.

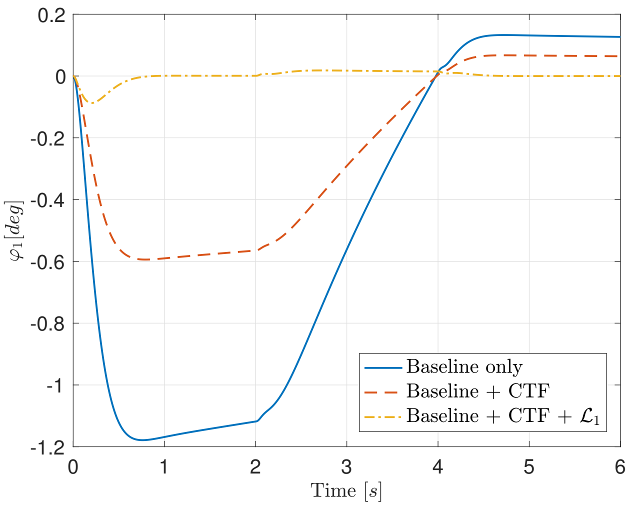


Figure 6.5: Simulation: baseline versus CTF versus $\mathcal{L}_1$-augmented, 25g payload.

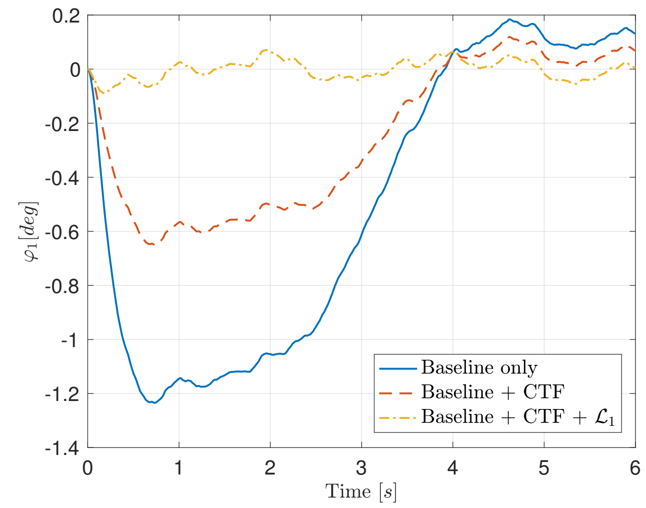


Figure 6.6: Simulation with sensor noise: baseline versus CTF versus $\mathcal{L}_1$-augmented, 25g payload.

6.1.4 Comparison against serial-type

One main result of this thesis is how the novel UAS design fares against the traditional serial-type approach. Figure 6.7 shows the time evolution of τ_m. The torques on the serial-type change much faster than the ones in the parallel case. At four seconds the parallel robot changes from 0 to -5mNm, while the seria-type chamges from 0 to -30mNm. This behaviour is detrimental to tracking performance as the quadcopter is unable to handle the fast dynamics in a satisfactory manner. Figure 6.8 depicts the pitch response for both systems. At two seconds, the parallel-type jumps to 0.005° and the serial-type jumps to 0.03°; a direct impact for the fast dynamics. For this comparison, the same actuator is used on both models to better isolate the contribution of the mechanical topology to the performance. Both systems are using the same actuators and attached to the same aircraft; we can conclude from the smaller tracking error that the parallel robot offers better performance. One might argue that the improvement is not significant, but it is important to note that the true benefit lies in the addition of the extra DoF as well as other advantages discussed in Section 2.2.

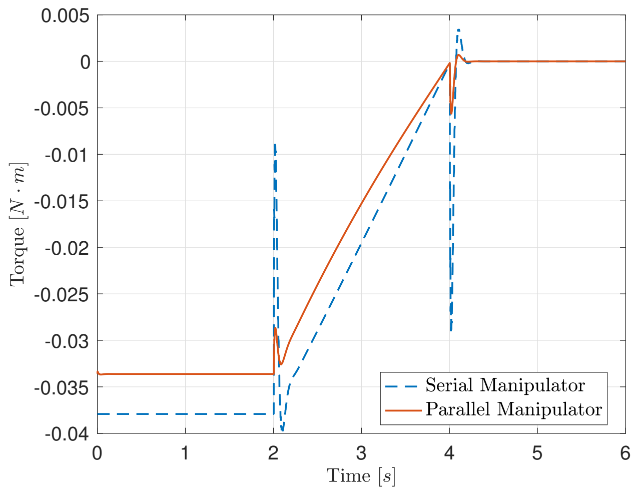


Figure 6.7: Torque profile: all controllers engaged, parallel versus serial-type, 25g payload.

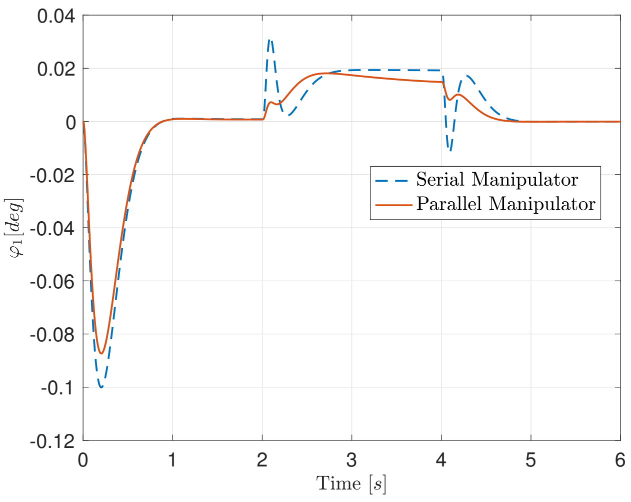


Figure 6.8: Simulation: all controllers engaged, parallel versus serial-type, 25g payload.

6.1.5 Multi-axis stabilization

So far we've only shown movements in the $y - z$ plane which only directly affect the pitch attitude. To showcase simultaneous pitch and roll stabilization we use the trajectory in figure 6.9, which utilizes all of the available DoF's.

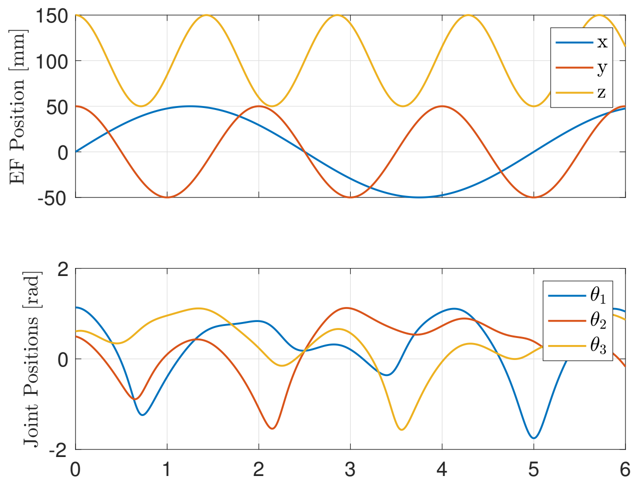


Figure 6.9: Simulation scenario kinematic trajectories for multi-axis manipulation.

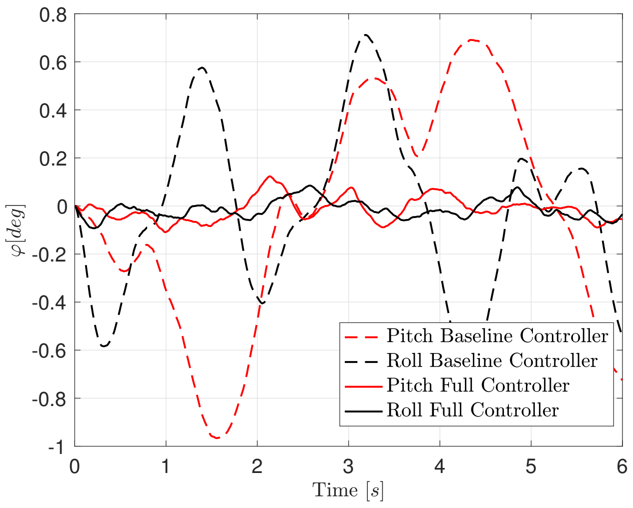


Figure 6.10: Simulation with sensor noise: all controllers engaged, multi-axis trajectory, 25g payload.

6.2 Optimal Trajectory Generation

For the benefit of simplicity, we will use a 2D example which illustrates the optimal trajectory generation algorithm functionality. The example is a quadcopter moving in the $y - z$ plane equipped with a 2-link rotational serial manipulator aligned with the same plane. Geometrically we have a $n = 2$ dimensional workspace and $m = 4$ DoF's, so the system is indeed redundant with $d = m - n = 2$ degrees of redundancy.

6.2.1 Spatially constrained scenario

The first task is a simple movement with no further constraints. Figure 6.11 shows the achieved behavior. The algorithm prefers to move the arm when no weights are given. In figure 6.12 weights are added to reflect the dynamic cost of moving the manipulator, and the end-effector and quadcopters trajectories become nearly identical, meaning there is minimal movement executed by the manipulator.

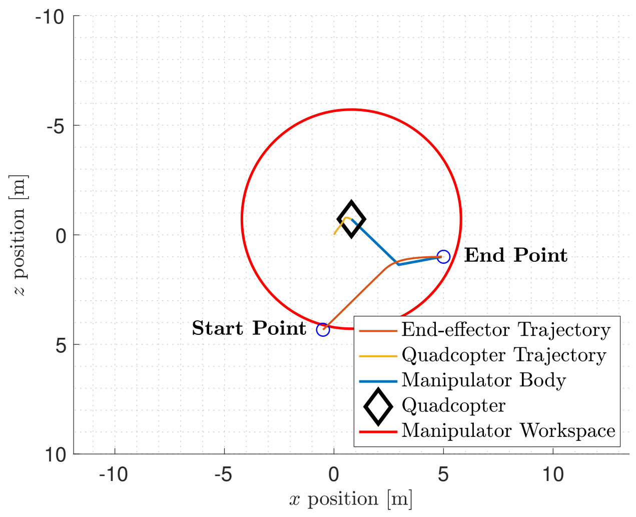


Figure 6.11: Mission: go from start to end.

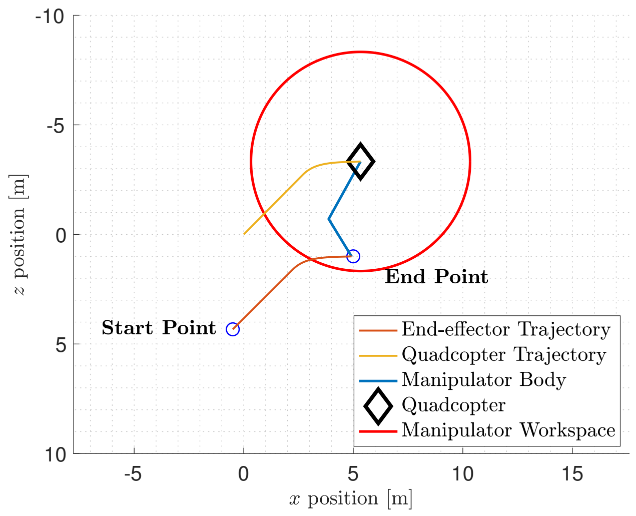


Figure 6.12: Mission: go from start to end, minimize kinetic energy.

6.2.2 Mechanically constrained scenario with additional task.

This scenario is similar to the previous one, but now there are two added constraints: collision avoidance and desired final configuration. In this scenario, we desire to reach the endpoint with the arm fully extended and with no collision between the quadcopter and the object (here depicted by a solid black sine wave). Figure 6.13 shows the achieved trajectory, the dashed line represents the region of influence, where the potential function is activated. We notice how the trajectory is adjusted as the aircraft flies into the region of influence.

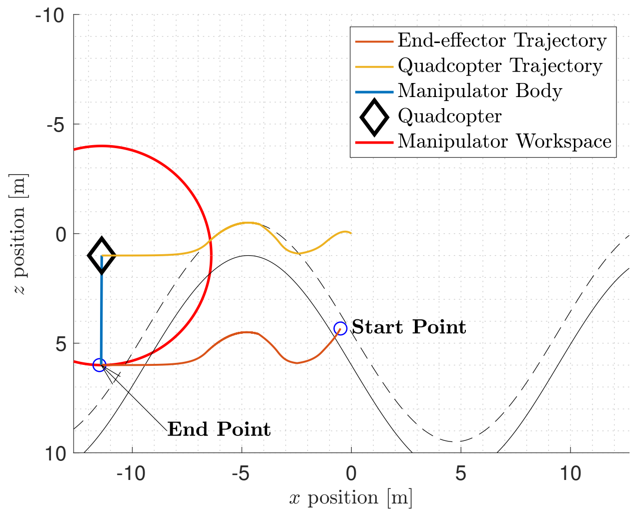


Figure 6.13: Mission: go from start to end with no collision arriving with a desired configuration.

Impossible task and relaxation

This is a modification of the previous scenario where the mission planner added more constraints than the system is able to resolve. The endpoint resides in a position where the system would either collide or not reach the desired configuration. Figures 6.14 and 6.15 show the impossible and relaxed mission respectively; in the former collision is avoided due to the higher weight given to that task, and in the latter the mission is relaxed to be attainable.

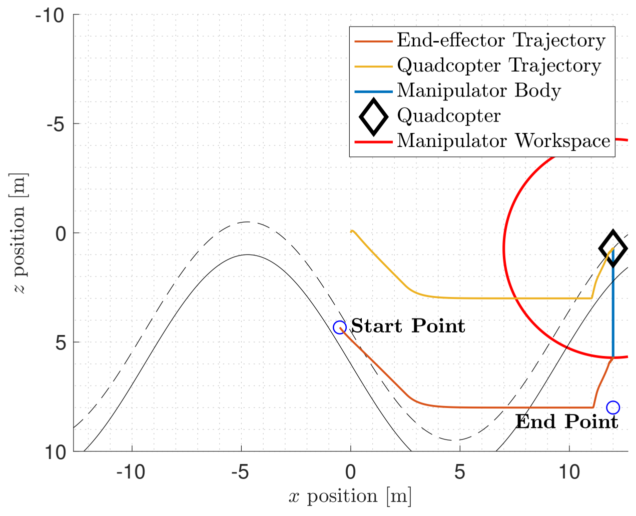


Figure 6.14: Impossible mission: go from start to end with no collision arriving with a desired configuration.

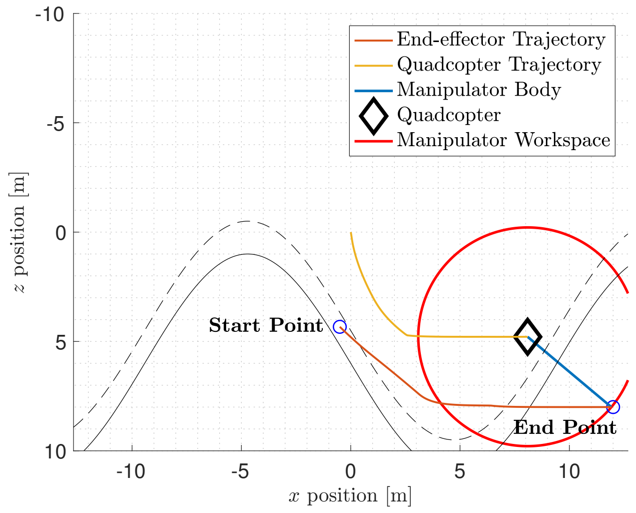


Figure 6.15: Relaxed mission: go from start to end with no collision.

CHAPTER 7

CONCLUSION

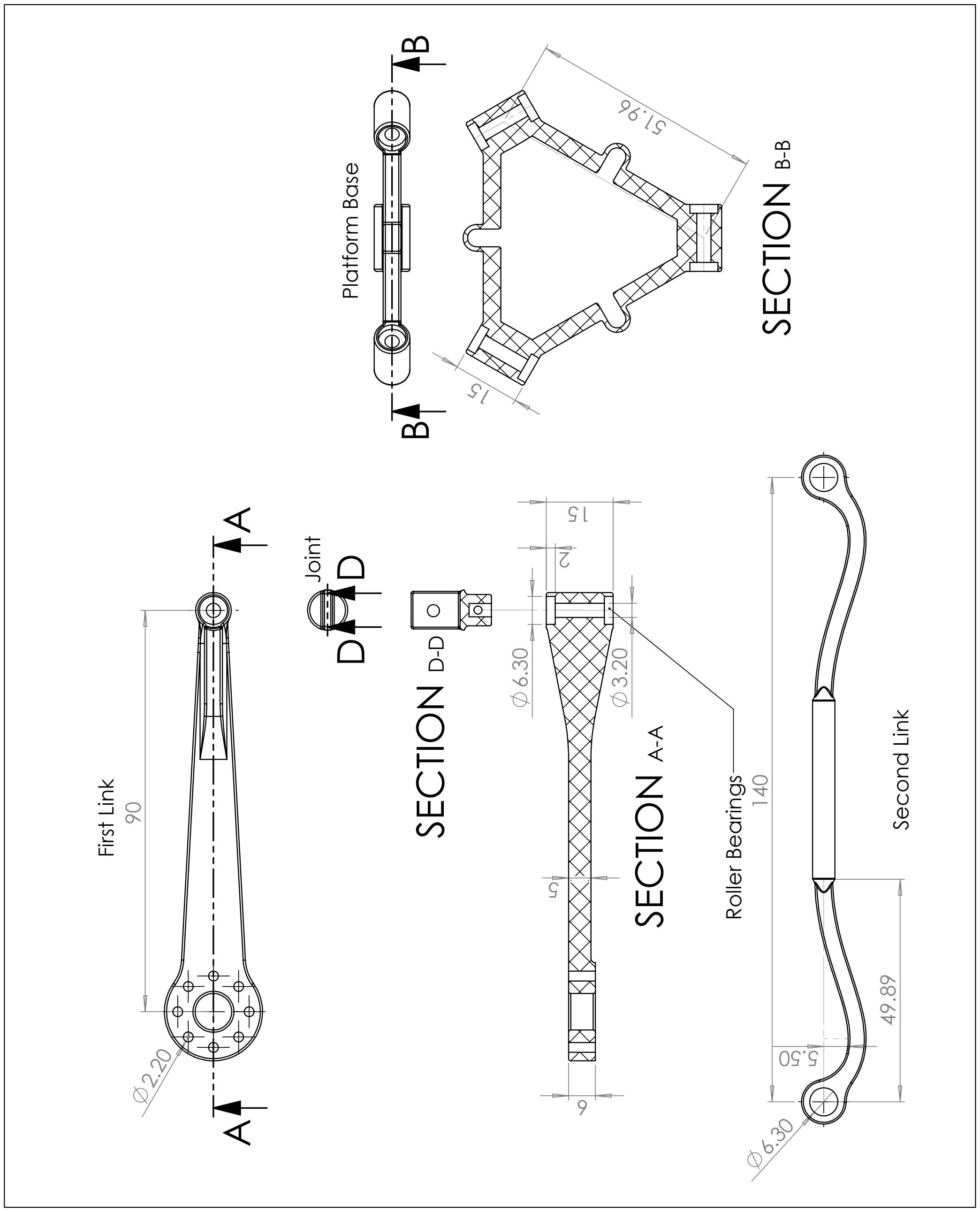
This thesis presented a new approach to aerial manipulation design. We developed the mechanism and it's equations to successfully stabilize an aircraft in the presence of uncertainties an then follow an optimal trajectory which satisfies flexible mission constraints. Realistic simulations provided results which support the theory behind the coupled system.

7.1 Future Directions

The next goal is flight testing. The major stepping stone towards it is the implementation of embedded algorithms for CTF and $\mathcal{L}_1$. The vehicle has flown with full control over both elements, however, the poor performance of the baseline controller reflected the critical need for the robust augmentation methods. Once these methods are validated, a high-level mission planner needs to be developed to complete the chain between user and machine.

APPENDIX A

DRAWINGS



APPENDIX B

INVERSE KINEMATICS ALGORITHM

```
function [ t , reach ] = FastInvKin(x,y,z,L,l,a,b,c)
sqrt3 = sqrt(3);
reach = 1;
t = zeros(3,1);

E(1) = 2*L*(y+a);
F(1) = 2*z*L;
G(1) = x*x+y*y+z*z+a*a+L*L+2*a*y-l*l;
E(2) = -L*(sqrt3*(x+b)+y+c);
F(2) = 2*z*L;
G(2) = x^2+y^2+z^2+b^2+c^2+L^2+2*(x*b+y*c)-l^2;
E(3) = L*(sqrt3*(x-b)-y-c);
F(3) = 2*z*L;
G(3) = x^2+y^2+z^2+b^2+c^2+L^2+2*(-x*b-y*c)-l^2;

if((G(1)-E(1))==0||(G(2)-E(2))==0||(G(3)-E(3))==0)
    t = 0;
    reach = 0;
    return
end

for i=1:3
    t(i) = 2*atan(-F(i)-sqrt(E(i)^2+F(i)^2-G(i)^2))
       /(G(i)-E(i));
end
end
```

APPENDIX C

FORWARD KINEMATICS ALGORITHM

```
function [p] = FastFwdKin(t,L,l,wb,wp,up,sp)

A1 = [0                                    -wb-L*cos(t
   (1))+up      -L*sin(t(1))];
A2 = [sqrt(3)/2*(wb+L*cos(t(2)))-sp/2   1/2*(wb+L*
   cos(t(2)))-wp -L*sin(t(2))];
A3 = [-sqrt(3)/2*(wb+L*cos(t(3)))+sp/2  1/2*(wb+L*
   cos(t(3)))-wp -L*sin(t(3))];

%solve the intersection of spheres (Ai,l)

l1 = l;
l2 = l;
l3 = l;

x1 = A1(1);
x2 = A2(1);
x3 = A3(1);
y1 = A1(2);
y2 = A2(2);
y3 = A3(2);
z1 = A1(3);
z2 = A2(3);
z3 = A3(3);

a11 = 2*(x3-x1);
a12 = 2*(y3-y1);
a13 = 2*(z3-z1);
```

```
a21 = 2*(x3-x2);
a22 = 2*(y3-y2);
a23 = 2*(z3-z2);

b1 = l1^2-l3^2-x1^2-y1^2-z1^2+x3^2+y3^2+z3^2;
b2 = l2^2-l3^2-x2^2-y2^2-z2^2+x3^2+y3^2+z3^2;

a1 = a11/a13-a21/a23;
a2 = a12/a13-a22/a23;
a3 = b2/a23-b1/a13;
a4 = -a2/a1;
a5 = -a3/a1;
a6 = (-a21*a4-a22)/a23;
a7 = (b2-a21*a5)/a23;

a = a4^2+1+a6^2;
b = 2*a4*(a5-x1)-2*y1+2*a6*(a7-z1);
c = a5*(a5-2*x1)+a7*(a7-2*z1)+x1^2+y1^2+z1^2-l1^2;

y = (-b-sqrt(b^2-4*a*c))/(2*a);
x = a4*y+a5;
z = a6*y+a7;

p = [x y z];
end
```

APPENDIX D

OPTIMAL TRAJECTORY GENERATION ALGORITHM

```
%% Q2 Arm Control by Inverse Jacobian

clc
close all

%% Simparam

l1 = 3; %parameters
l2 = 2;

theta1 = deg2rad(120); %initial conditions
theta2 = deg2rad(60);
d1 = 0;
d2 = 0;

t = 7; %end time
dt = 0.001; %control period
tol = 0.0001; %sim tolerance

cdes = [-11.5;6]; %desired position

kx = 5; %proportional gain on x axis
ky = 5; %proportional gain on y axis

sat = 3;%velocity control saturation
jsat = 10;%joint speeds saturation

Wt = 10*[5 0 ; 0 5]; %desired angles
```

```
W  = 2*[1 0 ; 0 1]; %task
Jt = [0 0 1 0 ; 0 0 0 1]; %desired angles
Jla = 1*eye(4); %angle limiter
Wla = zeros(4); %angle limiter
Wa = eye(4)+0*diag([0 0 500 500]); %actuation cost

lambda = 0.75;
qmin = deg2rad(30);
qmax = deg2rad(150);
buff = deg2rad(10);

pos0x = 0; %origin
pos0y = 0;

J = jacb(theta1,theta2,l1,l2);

syms xobj  %object
bar = matlabFunction(5*sin(xobj/3)+6);
xobj = -10*rg:0.1:10*rg;
yobj = bar(xobj);
ro = 1.5;

%% Start of simulation
d1dot = 0;
d2dot = 0;
theta1dot = 0;
theta2dot = 0;

x = [pos0x pos1x pos2x pos3x pos4x]; %manipulator "
   body"
y = [pos0y pos1y pos2y pos3y pos4y];

xe = x(3); %end effector position
ye = y(3);

f = 1;
```

```
for i = 0:dt:t

    error = cdes - [xe(end) ye(end)]';        %error
    pdes = min(sat,max(-sat,kx.*error));      %desired
        speed
    tdes  = [kx*(pi/2-theta1) kx*(pi/2-theta2)]';
       %desired joint speed

    if d1<(bar(d2)-ro)
        Wla(1,1) = 0;
    elseif bar(d2)>=d1&&d1>=(bar(d2)-ro)
        Wla(1,1) = -exp(-(1.15*d1-bar(d2)))-pi/2; %
           exponential potential
    elseif d1>bar(d2)
        Wla(1,1) = -1.5;
    end

    if theta1<qmin
        Wla(3,3) = 5000;
    elseif qmin<=theta1&&theta1<=qmin+buff
        Wla(3,3) = 5000/2*(cos(pi*(theta1-qmin)/buff
           )); %angular potential
    elseif qmin+buff<theta1&&theta1<qmax-buff
        Wla(3,3) = 0;
    elseif qmax-buff<=theta1&&theta1<=qmax
        Wla(3,3) = 5000/2*(cos(pi*(qmax-theta1)/buff
           ));
    else
        Wla(3,3) = 5000;
    end

    if theta2<qmin
        Wla(4,4) = 5000;
    elseif qmin<=theta2&&theta2<=qmin+buff
        Wla(4,4) = 5000/2*(cos(pi*(theta2-qmin)/buff
           )); %angular potential
    elseif qmin+buff<theta2&&theta2<qmax-buff
        Wla(4,4) = 0;
    elseif qmax-buff<=theta2&&theta2<=qmax
        Wla(4,4) = 5000/2*(cos(pi*(qmax-theta2)/buff
           ));
    else
        Wla(4,4) = 5000;
    end

    J = jacb(theta1,theta2,l1,l2);

    qotim = (J'*W*J+Jla'*Wla*Jla+Jt'*Wt*Jt+lambda^2*
       Wa)...
        \(J'*W*pdes+Jt'*Wt*tdes); %optimal law

    d1 = d1 + dt*qotim(1);
    d2 = d2 + dt*qotim(2);
    theta1 = theta1 + dt*qotim(3);
    theta2 = theta2 + dt*qotim(4);

    xe = [xe pos4x];
    ye = [ye pos4y];

    if(norm(error)+(max(max(Wt))>0)*1/1600*norm(tdes
       )<=800*tol)
        break %stop condition
    end
end
```

REFERENCES

[1] A. Maciejewski and C. A. Klein, "Numerical filtering for the operation of robotic manipulators through kinematically singular configuration," in *Journal of Robotic Systems*, 1988.

[2] G. R. B. E. Romer, H. J. A. Stuyt, and A. Peters, "Cost-savings and economic benefits due to the assistive robotic manipulator (arm)," in *9th International Conference on Rehabilitation Robotics, 2005. ICORR 2005.*, June 2005, pp. 201–204.

[3] F. Tobe, "Where are the elder care robots?" IEEE Spectrum, September 2017, http://spectrum.ieee.org/automaton/robotics/home-robots/where-are-the-eldercare-robots [cited September 2017].

[4] C.-A. Smarr, C. B. Fausset, and W. A. Rogers, "Understanding the potential for robot assistance for older adults in the home environment," Human Factors and Aging Laboratory, Georgia Institute of Technology, Technical Report HFA-TR-1102, 2011.

[5] "NRI: Collaborative research: ASPIRE: Automation supporting prolonged independent residence for the elderly," National Science Foundation, August 2015, http://www.nsf.gov/awardsearch/showAward?AWD_ID=1528036&HistoricalAwards=false [cited September 2017].

[6] M. Pomerleau, "Future of drones is small and cheap." 2017, https://www.c4isrnet.com/unmanned/uas/2017/05/12/future-of-drones-is-small-and-cheap/ [cited October 2017].

[7] G. Garimella and M. Kobilarov, "Towards model-predictive control for aerial pick-and-place," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, WA: Institute of Electrical & Electronics Engineers (IEEE), May 2015. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2015.7139850

[8] D. Lunni, A. Santamaria-Navarro, R. Rossi, P. Rocco, L. Bascetta, and J. Andrade-Cetto, "Nonlinear model predictive control for aerial manipulation," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2017, pp. 87–93.

[9] M. Kamel, K. Alexis, and R. Siegwart, "Design and modeling of dexterous aerial manipulator," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 4870–4876.

[10] T. W. Danko, K. P. Chaney, and P. Y. Oh, "A parallel manipulator for mobile manipulating UAVs," in *2015 IEEE International Conference on Technologies for Practical Robot Applications (TePRA).* Woburn, MA: Institute of Electrical & Electronics Engineers (IEEE), May 2015. [Online]. Available: http://dx.doi.org/10.1109/TePRA.2015.7219682

[11] R. M. Jones, D. Sun, G. B. Haberfeld, A. Lakshmanan, T. Marinho, and N. Hovakimyan, "Design and control of a small aerial manipulator for indoor environments," in *AIAA Information Systems - AIAA Infotech @ Aerospace, AIAA SciTech Forum.* Grapevine, TX: American Institute of Aeronautics and Astronautics, January 2017.

[12] Bitcraze, "Crazyflie 2.0," Bitcraze, September 2017, https://www.bitcraze.io/crazyflie-2/ [cited September 2017].

[13] R. Clavel, "Conception d'un robot parallèle rapide à 4 degrés de liberté," Ph.D. dissertation, EPFL, Lausanne, 1991.

[14] Robotis, "Dynamixel rx-24f," Robotis, September 2017, http://www.robotis.us/dynamixel-rx-24f-hn07-n101/ [cited September 2017].

[15] A. Lakshmanan, "Piecewise Bézier Curve Trajectory Generation and Control for Quadrotors," M.S. thesis, University of Illinois at Urbana-Champaign, Dec. 2016.

[16] S. B. Park, H. S. Kim, C. Song, and K. Kim, "Dynamics modeling of a delta-type parallel robot (isr 2013)," in *IEEE ISR 2013*, Oct 2013, pp. 1–5.

[17] N. Hovakimyan and C. Cao, *$\mathcal{L}_1$ Adaptive Control Theory: Guaranteed Robustness with Fast Adaptation.* SIAM, 2010.

[18] F. Fahimi, *Autonomous Robots.* Springer, 2009.