Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Imperial College London

Smart Robotics Lab
Dr. Stefan Leutenegger

Vision for Robotics Lab
Prof. Margarita Chli

Master Thesis

# Multimodal Feedback Control for Aerial Manipulation

**Spring Term 2020**

**Supervised by:**
Prof. Dr. Margarita Chli
Dr. Stefan Leutenegger

**Author:**
Felix Graule

# Declaration of Originality

I hereby declare that the written work I have submitted entitled

**Multimodal Feedback Control for Aerial Manipulation**

is original work which I alone have authored and which is written in my own words.[1]

**Author**

| | |
|---|---|
| Felix | Graule |

**Student supervisors**

| | |
|---|---|
| Margarita | Chli |
| Stefan | Leutenegger |

**Supervising lecturer**

| | |
|---|---|
| Margarita | Chli |

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (`https://www.ethz.ch/content/dam/ethz/main/education/rechtliches-abschluesse/leistungskontrollen/plagiarism-citationetiquette.pdf`). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

London, 12.05.2020 [illegible]

Place and date Signature

[1]Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

# Contents

# Abstract

Aerial Manipulation aims at combining the maneuverability of aerial vehicles with the interaction capabilities of robotic arms. This comes at the cost of additional control complexity due to the dynamic coupling between the two systems. The basis of this Master thesis is a custom built MAV-delta arm system using a Nonlinear Model Predictive Control (NMPC) method specifically designed for Micro Aerial Vehicles (MAVs) equipped with a robotic arm. As part of this thesis, the aforementioned MAV-delta arm system and NMPC method are assessed in extensive experiments performing Aerial Writing tasks on a whiteboard. The writing precision is evaluated through an appearance-based visual error which is proposed as a new standardised precision metric for Aerial Writing tasks. During the experiments, the NMPC is provided with accurate MAV and whiteboard localisation from an external motion capture system. However, in a real-world Aerial Manipulation task outside of a flight arena, such pose information would not be accessible. Instead, a Visual-Inertial SLAM system would give both noisy and drifting MAV pose estimates. In addition, the target surface pose would not be known exactly and parameter mismatches between the NMPC model and the physical system would be expected. The main contribution of this Master thesis is an extension to the NMPC method to allow operation in real-world setups by introducing a multimodal feedback pipeline. Both visual and tactile measurements are fused into three separate estimators to determine the drone's body, end effector and target surface pose. The proposed algorithms are evaluated extensively in a highly realistic, novel simulator allowing interactive Aerial Writing in the loop. The multimodal feedback is shown to accurately remove drift from the MAV position estimate, accounts for orientation misalignments in the whiteboard pose and corrects for errors in the end effector position caused by model mismatches. The resulting drawing precision is shown to be improved significantly and repeatably for different configurations of the error sources.

# Preface

This Master thesis was written at the Smart Robotics Lab (SRL) at Imperial College London as part of my Master's in Robotics, Systems and Control at ETH Zurich.
Throughout my six months at Imperial College, I was supervised and mentored by Dr. Stefan Leutenegger who is a Senior Lecturer (US equivalent Associate Professor). My biggest 'Thank you' goes to him for giving me this amazing opportunity, his trust in my abilities to let me work independently when possible and his outstanding support and guidance when facing conceptual and practical issues. Stefan combines an incredibly fast and sharp mind with a genuine and kind temperament. I am very glad to have worked under his supervision and profited a lot from him.
Secondly, I want to thank my supervisor Prof. Dr. Margarita Chli from the Vision for Robotics Lab (V4RL) at ETH Zurich. I am grateful for her support in organising my Master thesis at a top university abroad. I further want to thank her for the helpful and motivating feedback on the progress of my thesis. She always struck the right balance between highlighting positive aspects of the work presented and spotting pain points which needed improvement.
My third 'Thank you' goes to Dimos Tzoumanikas who is a PhD student at SRL. He provided me with a very interesting Aerial Manipulator platform used in this thesis and gave me the opportunity to cooperate on some of his research efforts. Doing experiments and paper writing with him, I learned a great deal about practical aerial robotics and scientific work in general.
I am also extremely grateful for the financial support received from the Zeno Karl Schindler Foundation in the form of a Master Thesis grant. Their belief in the proposed research allowed me to cover all living costs and hence to focus fully on doing research. Finally, I want to thank the rest of the SRL team who made my stay at Imperial truly memorable and fun! I also want to thank my partner, family and friends for their support.

# Acronyms

| | |
|---|---|
| CoM | Centre of Mass |
| DoF | Degree of Freedom |
| ETH | Eidgenössische Technische Hochschule |
| EKF | Extended Kalman Filter |
| FoV | Field of View |
| ICL | Imperial College London |
| ICP | Iterative Closest Point |
| IMU | Inertial Measurement Unit |
| MAV | Micro Aerial Vehicle |
| MPC | Model Predictive Control |
| NMPC | Nonlinear Model Predictive Control |
| QP | Quadratic Program |
| SLAM | Simultaneous Localisation and Mapping |
| VI-SLAM | Visual Inertial SLAM |

# Chapter 1

# Introduction

## 1.1 Motivation

Over the past decades, aerial manipulation has received great attention in the robotics research community, with many different systems in use [1, 2]. The tasks solved by aerial manipulators range from grasping, fetching and transporting arbitrary objects, to pushing against fixed surfaces. Potential use cases are numerous: inspection of infrastructure like bridges or manufacturing plants [3, 4, 5], physical interaction through tools like grinding, welding, drilling and other maintenance work in hard-to-reach places [6, 5], and the pick-up and transport of objects [7, 8]. All of these tasks require the MAV to be equipped with an additional mechanism referred to as the end effector. The coupling of the MAV dynamics with the moving end effector poses an interesting challenge from a control perspective given the inherent instability of MAVs. The requirement for high precision in real-world aerial manipulation applications further increases the difficulty of the control problem. The basis for this Master thesis is the aerial manipulator platform proposed in [9] which attempts to solve the mentioned control challenge. The system uses a Nonlinear Model Predictive Control (NMPC) method which is based on a hybrid model capturing the effect of contact and coupled MAV-arm dynamics. This method will be referred as the 'pre-existing NMPC' throughout the thesis. The MAV is designed to have low mechanical complexity and consists of an under-actuated hexacopter and a 3-DoF parallel delta arm. As part of this thesis, this system was evaluated in extensive real-world experiments performing Aerial Writing tasks, i.e. drawing on a whiteboard using a pen mounted at the end effector as visualised in Figure 1.1, which serves as an example for any aerial task requiring the MAV to carry a tool.

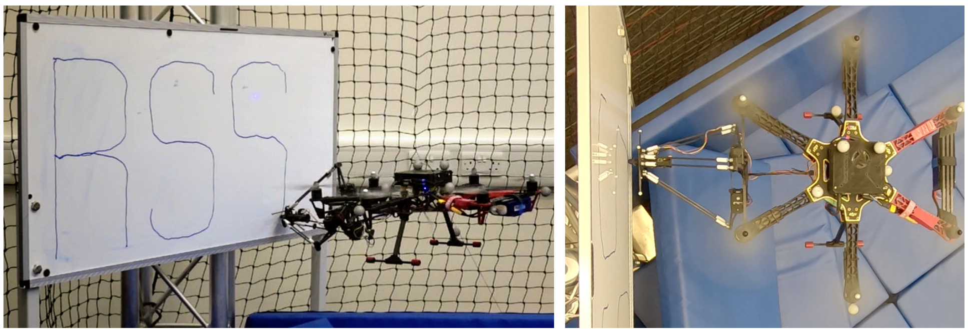


Figure 1.1: The MAV-arm system performing a real-world Aerial Writing task using the pre-existing NMPC method and external pose tracking.

Due to the lack of a widely used error metric for Aerial Writing, this thesis proposes an appearance-based visual error to be used as a new standardised evaluation method. To the author's best knowledge, the platform achieves unprecedented accuracy for contact-based aerial manipulation while using an under-actuated MAV-arm system.
However, it should be noted that during these experiments, the NMPC was provided with external, high accuracy pose estimates for both the MAV and manipulation target coming from an external motion capture system. In a real-world aerial manipulation setup, such accurate measurements would not be available. Instead, the MAV would run some implementation of SLAM onboard, e.g. a system like OKVIS [10]. Hence, MAV pose estimates would be noisy and drift over time. Furthermore, in a real-world aerial manipulation setup, the pose of the manipulation target would only be known approximately and should be updated by the robot sensing as the mission progresses. Even during the experimental evaluation using external localisation tracking, the limited accuracy of the whiteboard pose estimate proved to degrade the writing quality. Finally, both the use of a simplified, zero-order dynamics model of the delta arm as well as uncertainty in its physical parameters lead to a mismatch between the expected system behavior in the NMPC and the actual delta arm motion. In summary, three sources of error are identified:

1. MAV Position Drift caused by VI-SLAM running onboard.
2. Whiteboard Orientation Misalignment caused by imperfect calibration.
3. End Effector Position Errors caused by mismatches between the delta arm model and the physical system.

The main goal of this thesis is to introduce three individual feedback components to address these error sources and allow the aerial manipulator to be used outside laboratory conditions. According to the error sources, the three feedback components are:

1. MAV Position Drift Estimator introduced in Section 5.2.2.
2. Whiteboard Orientation Estimator introduced in Section 5.3.2.
3. Relative End Effector Position Control introduced in Section 5.4.3.

All three components are tested in depth both individually as well as in combination through simulation of the Aerial Writing task in a highly realistic drone simulator as illustrated in Figure 1.2. The feedback is based on multimodal measurements from both a camera and a force sensor mounted on the MAV.

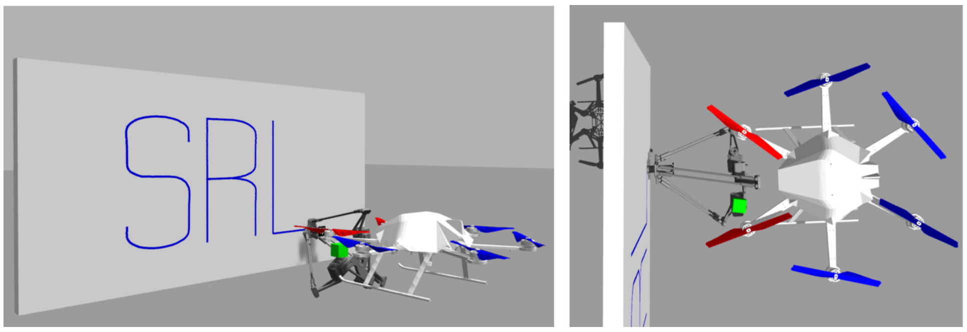


Figure 1.2: The MAV-arm system performing a simulated Aerial Writing task using the NMPC method with additional multimodal feedback.

## 1.2 Claimed Contributions

In this thesis, the author claims to make the following contributions:

1. The pre-existing NMPC method is extended to include multimodal feedback from visual and tactile measurements to allow Aerial Writing in a real-world setup with onboard MAV localisation, faulty Whiteboard orientation calibration and delta arm model mismatches.
2. A highly realistic simulator is developed to evaluate the proposed feedback while providing full simulation of the Aerial Writing process, the error sources as well as visual and tactile sensor outputs.
3. The NMPC method proposed by [9] is evaluated experimentally in Aerial Writing tasks using an appearance-based visual error metric. This metric is proposed as a new standardised way to measure the precision of platforms performing such tasks.

It should be highlighted that both the pre-existing NMPC as well as the novel multimodal feedback method are generic enough to be applied to different types of vehicles such as omni-directional ones and/or other types of manipulators.

## 1.3 Thesis Structure

This thesis is organised as follows: In Section 2 the reader will find an overview of the related work on aerial manipulation in general and the use of visual and tactile feedback in particular. In Section 3 the NMPC method introduced by Dimos Tzoumanikas and the associated notation and coordinate frames are explained. Section 4 gives an overview of the software architecture and the simulation setup. The multimodal feedback is presented in detail in Section 5. Both the experimental results for the pre-existing NMPC method and the simulations for the multimodal feedback are shown in Section 6. Finally, Section 7 discusses the findings before drawing conclusions and directions for future work in Section 8.

## 1.4 Notation

Vectors are denoted as bold lower case symbols, e.g. $\mathbf{v}$. Left-hand subscripts, e.g. ${}_A\mathbf{v}$, are used to indicate the coordinate representation in the $\underrightarrow{\mathcal{F}}_A$ frame of reference. The rotation matrix $\mathbf{C}_{AB}$ changes the representation of the vector ${}_B\mathbf{v}$ from $\underrightarrow{\mathcal{F}}_B$ to $\underrightarrow{\mathcal{F}}_A$ as ${}_A\mathbf{v} = \mathbf{C}_{AB}\,{}_B\mathbf{v}$. Analogously to the rotation matrix $\mathbf{C}_{AB}$ , the quaternion $\mathbf{q}_{AB}$ with $\otimes$ denoting the quaternion multiplication is used. The skew symmetric matrix of the vector $\mathbf{v}$ is denoted as $[\mathbf{v}]^{\times}$. The motion of the MAV Body frame $\underrightarrow{\mathcal{F}}_B$ ($x$: forward, $y$: left, $z$: upward) is expressed with respect to the World frame $\underrightarrow{\mathcal{F}}_W$ ($z$: upward). Measured values are denoted with tildes, e.g. $\widetilde{x}$, while estimated values are denoted with hats, e.g. $\widehat{z}$.

# Chapter 2

# Related Work

This chapter introduces relevant prior work and connects it to the thesis at hand where possible. The first section considers aerial manipulation and contact-based tasks such as Aerial Writing. The second section focuses on the error sources tackled in this thesis while the third section gives an overview on visual and tactile feedback.

## 2.1 Aerial Manipulation

Aerial manipulation systems can be distinguished based on the MAV type (as being omni-directional or under-actuated) and the end effector (as being fixed or moving).

### 2.1.1 Omni-directional Platforms

Broadly speaking, omni-directional MAVs do not require a moving end effector to fulfill complex aerial tasks as the necessary 6-DoF are provided by the MAV itself. Examples include the works presented by [11, 12, 13]. Brescianini and D'Andrea [11] show an omni-directional MAV called OmniCopter that achieves 6-DoF motion by using eight fixed rotors in a non co-planar configuration. In a subsequent study [12], this platform is used with a fixed end effector to fetch moving objects. Using a similar approach with a fixed configuration of tilted rotors, Ryll et al. [13] propose a novel paradigm to control all 6-DoF of the MAV while using a rigid end effector to exert forces and torques independently. The system is demonstrated in numerous experimental tasks, e.g. surface sliding. However, tilt-rotor configurations introduce significant turbulence, decreasing energy efficiency and thus flight time.
Following a different approach, Kamel et al. [14] develop a setup of six rotors which tilt individually to control the direction of their thrust vector. [5, 15] leverage this system, named Voliro, to solve a variety of aerial manipulation tasks with a rigidly mounted, low complexity end effector. The authors further show precise force control when in contact with unstructured environments while running online visual-inertial state estimation. While this platform allows for accurate 6-DoF flight and longitudinal force exertion with a relatively simplistic control method, it is mechanically more complex and thus more costly compared to classical multi-copter platforms. They also point out that due to aging infrastructure the demand for contact-based inspection will continue to grow in the future. At the same time, sensor miniaturisation will enable more accurate force tracking using aerial manipulators. Recently, a similar approach was followed by Ángel Trujillo et al. [4], who introduced AeroX, an omni-directional octocopter for contact-based inspection. Their clever end effector design minimises the torque caused by contact, and features wheels on its base to move along a surface while remaining in contact. In

[6, 16] the authors show a less complex but highly capable tri-tilt-rotor MAV for surface grinding and obstacle manipulation. The control model consists of two disjoint modes: one for free-flight and another for physical interaction. The authors further discuss different force exertion principles for under- and fully-actuated MAVs.

### 2.1.2 Under-actuated Platforms

Employing an under-actuated MAV to perform aerial manipulation typically increases the complexity of the end effector since it has to provide additional DoF. Many different end effector designs have been proposed over the last years. We can categorise these works by the increasing complexity of the end effector: Darivianakis et al. [17] use a fixed end effector on an under-actuated MAV to perform contact-based tasks. In [18], the authors use different light-weight, low complexity grippers to perch, pick up, and transport payload. Meanwhile, Kessens et al. [7] use a self-sealing suction mechanism to pick up and carry objects. Moving up in terms of complexity, Kim et al. [19] suggest mounting a 2-DoF robotic arm on an MAV to allow grasping and transporting of objects. The authors propose an adaptive sliding mode controller for the combined system. In [20] the authors present an aerial manipulator with two robotic 2-DoF arms to open a valve. The MAV and arms are controlled as a coupled system which is modeled as a switched nonlinear system during valve turning. The approach presented in [21] controls an MAV with a servo robot arm by considering the arm's motion and using a moving battery as well as the MAV thrust to counteract it. Furthermore, external forces and moments are estimated and fed back into the controller. In a more recent work, Suarez et al. [22] propose a light-weight, human-sized dual arm system designed to minimise the inertia transferred to the MAV. Each of the two arms add 5-DoF to the system and the applied arm control law takes into account that low-cost servo motors do not allow torque control but require position commands. Further, a torque estimator is used to predict the torques produced by the servos and inform the MAV control algorithm accordingly. In order to minimise such disturbances coming from the end effector, Nayak et al. [23] propose a light-weight design which can produce longitudinal forces for contact-based inspection using a switched system MPC method incorporating the contact dynamics. While attaching a serial robotic arm on an MAV increases the number of tasks it can perform, they only provide limited precision when using low-cost and light-weight actuators. Some previous efforts try to mitigate this by mounting a parallel delta arm on an MAV instead, which allows higher precision at lower mechanical complexity. In [24, 25] the authors demonstrate a multi-objective dynamic controller which considers dynamic effects between the MAV and its 3-DoF delta arm. A linear model predictive control approach including external disturbance estimation is used for MAV tracking while the end effector is controlled using a PID controller. The same system is used in [26] to inspect tree cavities with a camera mounted at the end effector.

As mentioned, some methods specifically consider the coupling between MAV and end effector. A similar approach is taken in [27], where a shifting CoM is assumed. However, instead of correcting for end effector motion, the CoM displacement is used to adapt to any offsets between the true CoM of the MAV and its geometric centre to remove remaining tracking errors in the system.

In summary, when comparing under-actuated to fully-actuated approaches, the former result in mechanically simpler, cheaper and thus more widespread platforms which in turn have less control authority over lateral DoFs. Hence, they require complex control methods to provide reasonably precise force exertion. The latter can provide more accurate force control with simpler methods. However, the higher mechanical complexity not only increases cost but also the number of model parameters and hence model uncertainty [15].

### 2.1.3 Aerial Writing

In this work, Aerial Writing is performed for evaluation of the presented system and control method. It is used to show the ability of the aerial manipulator to track a reference trajectory while being in contact and exerting a given force. The same evaluation task was already used in efforts by [17, 28, 5]. In Darivianakis et al. [17], the authors use a fixed end effector on an under-actuated MAV controlled by a linear MPC method and mode switching between a free-flight and in-contact model. Hence, there are no coupled dynamics to be taken into account. A similar task is performed by a paint spraying drone presented in [29]. This however does not require contact and therefore poses a somewhat different control challenge.

When comparing the approach presented in this thesis to the first approach [17], it achieves higher accuracy and flexibility in terms of potential use cases due to the moving end effector. In contrast to the second approach [5], the NMPC achieves on par precision while relying on a simpler, under-actuated platform. However, the fully-actuated platform in [5] can be adapted to a much wider range of use cases and is able to achieve higher lateral force exertion precisely.

## 2.2 Considered Error Sources

### 2.2.1 Drift in VI-SLAM

Any kind of SLAM system running onboard a robot to provide localisation based on relative sensors like wheel encoders, IMU or visual landmarks is bound to drift away from the true position of the robot. This phenomenon is referred to as *dead reckoning* and happens gradually over time as small errors accumulate. For certain outdoor applications, this can be solved by using GPS as an absolute position measurement. However, for most aerial manipulation tasks, relying on GPS is not possible due to lack of satellite visibility (indoor operation) and the requirement for millimetre-level accuracy as even the best Differential GPS systems can only deliver centimetre-level precision [30]. Hence, the ability to remove drift using onboard sensing is crucial. A popular approach taken in Visual Inertial SLAM systems is *Loop Closure* [31], where the robot tries to recognise previously visited areas to reduce the uncertainty accumulated in the map. Another method to reduce drift is to perform tight fusion of IMU data into a key-frame based nonlinear optimisation as shown by the authors of [10]. Further, tracking visual changes induced by camera motion in a dense manner as done in [32, 33] also reduces the amount of drift.

In summary, the amount of drift which a VI-SLAM system accumulates over time depends on factors like its underlying method and the environment in which it performs. Therefore, there is no single true assumption to be made when simulating or modelling drift.

### 2.2.2 External Motion Capture

The external pose measurements used in this thesis are provided by a Vicon system. As described in [34], such a setup allows reliable measurements with sub-millimetre accuracy for both static and moving objects when calibrated perfectly. However, during the experiments performed in Section 6.2, the observed accuracy was significantly lower and not as reliable. Possible reasons for this are discussed later on.

### 2.2.3 Parallel Delta Robot

A delta robot consists of three arms connected to a base through rotary joints. Each arm includes another universal joint connecting the rigid upper part to a parallelogram shaped lower part which connects to the end effector. The end effector platform always remains the same orientation as the base, hence the name parallel robot. All parts together form a closed kinematic chain. In [35], a complete modelling for such a delta parallel robot is introduced. The parameters in such a fully feed-forward model can only be measured up to a certain precision. Hence, there will be a mismatch between the setpoint positions sent to the delta arm and the actually achieved position. This is addressed in [36] where a kinematic calibration scheme is used to update the model parameters according to observations of its actual motion. In [37], a structured approach to finding an optimal set of delta arm parameters is given. Finally, [38] lists the Jacobians for the delta arm setup and uses them to analyse singularities. However, for the sake of simplicity, the Jacobians used in Section 5.4.3 are computed numerically based on the inverse kinematics.

## 2.3 Multimodal Feedback

### 2.3.1 Visual Feedback

When looking into visual feedback in connection with robotic manipulation, one major field of study is Visual Servoing. In [39, 40] the authors give a comprehensive introduction to this collection of methods. The distinction between Image-based Visual Servoing (IBVS) and Position-based Visual Servoing (PBVS) is explained. The former, IBVS, formulates the control law based on an error in the image plane between the current and intended features. No pose estimation for the target is performed in these approaches. In [41] such an IBVS method was extended to include state constraints and predictions along a planning horizon. The authors of [42, 43] then present a similar predictive IBVS approach with additional robustness to control a fixed serial robotic arm with the camera attached on the end effector (eye-in-hand). Finally, in [44] a IBVS method is extended to a stochastic MPC to perform aerial grasping of a cylindrical object using a robotic grasping arm mounted on an MAV. One major drawback of IBVS is its rigid approach and the target pose is not estimated. The method used in this thesis would rather fall into the latter set of approaches, PBVS. The pose of the target is estimated with respect to the camera, based on which commands are sent to the robot.
To compute the visual measurement used in the proposed approach, template matching as explained in [45] is applied. To allow fast matching, a metric based on the normalised sum of squared differences as presented in [46] is used. To measure how reliable the current visual measurement is, the matching uncertainty is estimated as proposed by the authors of [47].

### 2.3.2 Tactile Feedback

To directly control the force acting from the end effector onto the target, one can either estimate the exerted force based on the MAV state or even incorporate actual tactile measurements. An external force and wrench estimator is used by [20, 28]. Similar to the tactile feedback used in the proposed method at hand, the system shown in [15] employs a force sensor to provide direct haptic feedback when in contact.

# Chapter 3

# Background

In this chapter, the pre-existing NMPC pipeline proposed in [9] is explained. After defining the coordinate frames, the subsequent sections introduce the hybrid modelling approach as well as the model-based control and delta arm kinematics.

## 3.1 Coordinate Frames

The different coordinate frames used in the background work are shown in Figure 3.1. For the pre-existing NMPC, the World and Touch frame are known exactly.

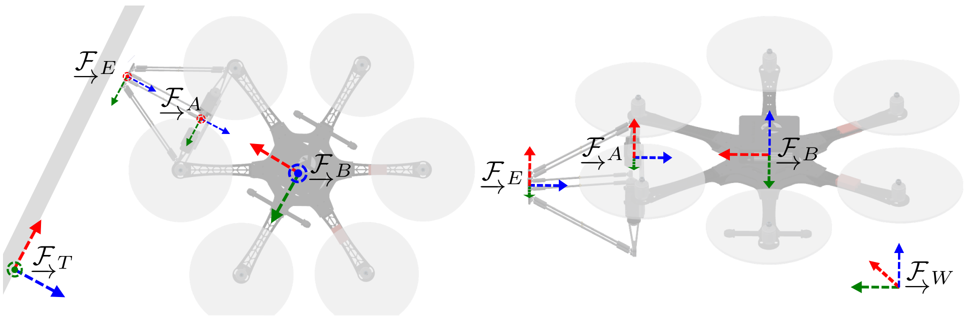

$\underrightarrow{\mathcal{F}}_E$
$\underrightarrow{\mathcal{F}}_A$
$\underrightarrow{\mathcal{F}}_B$
$\underrightarrow{\mathcal{F}}_T$
$\underrightarrow{\mathcal{F}}_E$
$\underrightarrow{\mathcal{F}}_A$
$\underrightarrow{\mathcal{F}}_B$
$\underrightarrow{\mathcal{F}}_W$

Figure 3.1: The coordinate frames $\underrightarrow{\mathcal{F}}_W$, $\underrightarrow{\mathcal{F}}_B$, $\underrightarrow{\mathcal{F}}_A$, $\underrightarrow{\mathcal{F}}_E$, and $\underrightarrow{\mathcal{F}}_T$ stand for the World, MAV Body, Arm, End Effector, and Touch frame, respectively.

## 3.2 Hybrid Modelling

The standard Newton-Euler equations are used to model the combined MAV-arm dynamics. The MAV is modeled as a single rigid body object while only quasi-static forces introduced by the arm dynamics and its interaction with the environment are considered. The effect of end effector motion and interaction with the environment on the MAV dynamics is approximated by introducing additional, external forces and moments. Overall, the combined dynamics take the following form:

$$
\begin{aligned}
{}_{W}\dot{\mathbf{r}}_{B} &= {}_{W}\mathbf{v}_{B}\,, && (3.1a)\\
\dot{\mathbf{q}}_{WB} &= \frac{1}{2}\boldsymbol{\Omega}({}_{B}\boldsymbol{\omega})\mathbf{q}_{WB}\,, && (3.1b)\\
{}_{W}\dot{\mathbf{v}}_{B} &= \frac{1}{m_c}\mathbf{C}_{WB}\,({}_{B}\mathbf{F}_r + {}_{B}\mathbf{F}_e) + {}_{W}\mathbf{g}, && (3.1c)\\
{}_{B}\dot{\boldsymbol{\omega}} &= \mathbf{J_c}^{-1}({}_{B}\mathbf{M}_r + {}_{B}\mathbf{M}_e - {}_{B}\boldsymbol{\omega} \times \mathbf{J_c}{}_{B}\boldsymbol{\omega}), && (3.1d)\\
\boldsymbol{\Omega}({}_{B}\boldsymbol{\omega}) &= \begin{bmatrix} {}_{B}\boldsymbol{\omega}^{\times} & -{}_{B}\boldsymbol{\omega} \\ {}_{B}\boldsymbol{\omega}^{\top} & 0 \end{bmatrix}, && (3.1e)
\end{aligned}
$$

where $m_c$, $\mathbf{J_c}$ are the combined MAV-arm mass and inertia tensors, respectively. Regarding the forces and moments ${}_{B}\mathbf{F}_i$, ${}_{B}\mathbf{M}_i$, the subscript $i \in \{r, e\}$ is used to distinguish the ones generated by the MAV motors $r$ from the ones caused by the end effector movement $e$ and its potential contact with the environment. The MAV motor-generated forces and moments are given by:

$$
\begin{aligned}
{}_{B}\mathbf{F}_r &:= \left[0, 0, T\right]^{\top},\ T = \sum_{i=1}^{6} f_i, && (3.2a)\\
{}_{B}\mathbf{M}_r &:= \sum_{i=1}^{6}\Big(f_i\,{}_{B}\mathbf{r}_i \times {}_{B}\mathbf{e}_z + (-1)^{i+1} k_m f_i\,{}_{B}\mathbf{e}_z\Big), && (3.2b)
\end{aligned}
$$

with $f_i \in \mathbb{R}$ the thrust produced by the $i^{\text{th}}$ motor, ${}_{B}\mathbf{r}_i$ its position with respect to the MAV Body frame, $k_m$ the known thrust to moment constant and ${}_{B}\mathbf{e}_z = [0, 0, 1]^{\top}$. Equation 3.2 can be summarised as $\left[{}_{B}\mathbf{M}_r,\ \ T\right]^{\top} = \mathbf{A}\left[f_1\ \ f_2\ \ \ldots\ \ f_6\right]^{\top}$ with $\mathbf{A} \in \mathbb{R}^{4\times 6}$ the allocation matrix similar to the one described by Achtelik et al. [48]. ${}_{B}\mathbf{F}_e$ and ${}_{B}\mathbf{M}_e$ are given by:

$$
\begin{aligned}
{}_{B}\mathbf{F}_e &:= \mathbf{C}_{BE}\,{}_{E}\mathbf{F}_c, && (3.3a)\\
{}_{B}\mathbf{M}_e &:= {}_{B}\mathbf{r}_E \times {}_{B}\mathbf{F}_e + ({}_{B}\mathbf{r}_E - {}_{B}\mathbf{r}_{E_0}) \times (\mathbf{C}_{BW}\, m_e\, {}_{W}\mathbf{g}), && (3.3b)
\end{aligned}
$$

where ${}_{E}\mathbf{F}_c$ is the contact force acting on the end effector expressed in its frame $\underrightarrow{\mathcal{F}}_E$ and ${}_{B}\mathbf{r}_{E_0} \in \mathbb{R}^3$ the nominal end effector position which results in no CoM displacement. The two terms in the Equation 3.3b represent the moments due to contact and due to the displacement of the CoM respectively. The combined mass $m_c = m + m_e$ is the sum of the MAV and end effector, respectively, while the combined rotational inertia can be computed as $\mathbf{J_c} = \mathbf{J} + m_e \text{diag}({}_{B}\mathbf{r}_E - {}_{B}\mathbf{r}_{E_0})^2$ with $m_e$ being the mass of the end effector, $\mathbf{J} = \text{diag}(J_x, J_y, J_z)$ the inertia tensor of the MAV (including the arm mass in nominal position) and $\text{diag}(\cdot)$ the corresponding diagonal matrix.

Regarding the contact force, an approximation is made by applying a linear spring model as:

$$
{}_{E}\mathbf{F}_c = \mathbf{C}_{ET}\,(k_s\,{}_{T}\mathbf{r}_{E_z}), \qquad (3.4)
$$

where $k_s$ is a known spring coefficient and ${}_{C}\mathbf{r}_{E_z}$ is the normal component of the contact surface penetration. This way, the controller can anticipate contact before it even happens and there is no need for a switching mode controller (one for free flight and another one for contact dynamics).

## 3.3 Model Based Control

For the control formulation the following control state and input are defined:

$$\mathbf{x} := \left[{}_W\mathbf{r}_B, {}_W\mathbf{v}_B\, , \mathbf{q}_{WB}\, , {}_B\boldsymbol{\omega}\,\right]^\top \in \mathbb{R}^6 \times \mathbb{S}^3 \times \mathbb{R}^3, \tag{3.5a}$$

$$\mathbf{u} := \left[{}_B\mathbf{M}_r, T, {}_A\mathbf{r}_E\right]^\top \in \mathbb{R}^7. \tag{3.5b}$$

Note that ${}_B\mathbf{r}_E$ is used for the formulation of the control model, while ${}_A\mathbf{r}_E$ is used in the control input. Using the constant and known homogeneous transformation $\boldsymbol{T}_{BA}$ one can switch between the coordinate representation of these position vectors.
The following error functions are used for the position of the MAV, the position of the end effector, the MAV linear and angular velocity, the orientation, the contact force and the control input, respectively:

$$\mathbf{e}_{rB} = {}_W\mathbf{r}_B - {}_W\mathbf{r}_B^r, \tag{3.6a}$$

$$\mathbf{e}_{rE} = {}_W\mathbf{r}_E - {}_W\mathbf{r}_E^r, \tag{3.6b}$$

$$\mathbf{e}_v = {}_W\mathbf{v}_B\; - {}_W\mathbf{v}_B^r\, , \tag{3.6c}$$

$$\mathbf{e}_\omega = {}_B\boldsymbol{\omega}\; - \mathbf{C}_{BB^r}\, {}_{B^r}\boldsymbol{\omega}^r, \tag{3.6d}$$

$$\mathbf{e}_q = [\mathbf{q}_{WB}^{-1} \otimes \mathbf{q}_{WB}^r\,]_{1:3}, \tag{3.6e}$$

$$e_f = f_c - f_c^r, \tag{3.6f}$$

$$\mathbf{e}_u = \mathbf{u} - \mathbf{u}^r, \tag{3.6g}$$

with $f_c := {}_E\mathbf{F}_{c_z}$ and the superscript $r$ used to denote the time-varying reference quantities. The optimal input sequence $\mathbf{u}^*$ is obtained by the online solution of the following constrained optimisation problem:

$$\mathbf{u}^* = \underset{\mathbf{u}_0,\ldots,\mathbf{u}_{N_f}}{\operatorname{argmin}} \Big\{ \varPhi(\mathbf{x}_{N_f}, \mathbf{x}_{N_f}^r) + \sum_{n=0}^{N_f-1} L(\mathbf{x}_n, \mathbf{x}_n^r, \mathbf{u}_n) \Big\}, \tag{3.7a}$$

$$\text{s.t. : } \mathbf{x}_{n+1} = \mathbf{f}_d(\mathbf{x}_n, \mathbf{u}_n), \tag{3.7b}$$

$$\mathbf{x}_0 = \hat{\mathbf{x}}, \tag{3.7c}$$

$$u_{\text{lb}} \le u_i \le u_{\text{ub}}, \quad i = 1, \ldots, 7, \tag{3.7d}$$

where $N_f$ is the discrete horizon length, $\mathbf{f}_d$ is the discrete version of the dynamics given in Equations 3.1-3.3, $\hat{\mathbf{x}}$ is the current state estimate and $u_{\text{lb}}, u_{\text{ub}}$ are the appropriate lower and upper bounds of the control input defined in Equation 3.5. For the intermediate $L$ and final terms $\varPhi$ quadratic costs of the form $\mathbf{e}_i^\top \mathbf{Q}_i \mathbf{e}_i$ $\forall \mathbf{e}_i \in \{\mathbf{e}_{rB}, \mathbf{e}_{rE}, \mathbf{e}_v, \mathbf{e}_\omega, \mathbf{e}_q, e_f, \mathbf{e}_u\}$ are used as defined in Equation 3.6 where the gain matrices $\mathbf{Q}_i \succcurlyeq 0$ were experimentally tuned.
The optimal control problem is implemented using a modified version of the CT toolbox [49] with a 10 ms discretisation step and a 2 s constant prediction horizon. A Runge-Kutta 4 integration scheme is applied, followed by a re-normalisation for the quaternion. As common in receding horizon control, the first input $\mathbf{u}_0^*$ is applied to the system and the whole process is repeated once a new state estimate $\hat{\mathbf{x}}$ becomes available. The motor commands $\mathbf{f} = \begin{bmatrix} f_1 & f_2 & \ldots & f_6 \end{bmatrix}^\top$ for the MAV are obtained by solving the following QP:

$$\mathbf{f}^* = \underset{\mathbf{f}}{\operatorname{argmin}} \left( \left\lVert \mathbf{A}\mathbf{f} - \mathbf{u}_{0_{1:4}}^* \right\rVert_{\mathbf{W}}^2 + \lambda \left\lVert \mathbf{f} \right\rVert_2^2 \right), \tag{3.8a}$$

$$\text{s.t. : } f_{\min} \le f_i \le f_{\max}, \quad i = 1, \ldots, 6, \tag{3.8b}$$

where $f_{\min}, f_{\max} \in \mathbb{R}$ are the minimum and the maximum attainable thrusts, $\mathbf{W} \in \mathbb{R}^{4\times4}$ is a gain matrix and $\lambda = 10^{-7}$ is a regularisation parameter. The end effector

position commands $\mathbf{u}^*_{0_{5:7}}$ are mapped into servo angle commands $\theta_1, \theta_2, \theta_3$ by solving the inverse kinematics problem for the delta arm explained in Section 3.4. In the case of an infeasible (e.g. outside the arm's workspace) or unsafe end effector position command (e.g. one that results in a collision between the MAV propellers and the arm's links), the position command is reprojected onto the boundary of the feasible and safe to operate workspace. In practice, this was rarely the case as the MAV and end effector reference trajectories are designed so that the end effector operates close to its nominal position. In this way the usable workspace is maximised while the effect of the CoM displacement (which is captured by the control model) is minimised.
The C++ implementation of the above, requires 6.7 ms with a standard deviation of 0.57 ms per iteration. On average, 98% of the computation time is spent on the optimisation problems defined in Equations 3.7 and 3.8.
It should be highlighted that the proposed method is generic enough to be applied to different types of vehicles such as omni-directional ones and/or other types of manipulators. In that case the control input and the appropriate control allocation method has to be changed based on the vehicle and manipulator DoF and their actuator type. Similarly, the model can easily be extended to capture aerodynamic friction, gyroscopic moments, handle multiple contact points or use more sophisticated contact models (e.g. ones that include combination of linear springs and dampers). Similarly, the Aerial Writing task which are used for the experimental evaluation of the framework, is just an example application that requires precision. The algorithms are believed to be adaptable to other tasks such as inspection through contact.

## 3.4 Delta Arm Kinematics

The MAV is equipped with a custom built 3-DoF delta arm [35] as shown in Figure 4.2. Its main advantages are speed, as its few moving parts are made of lightweight materials, precision and the easy to solve forward and inverse kinematics. The forward kinematics problem (i.e. determining the position of the end effector ${}_A\mathbf{r}_E$ given the joint angles $\theta_1, \theta_2, \theta_3$) can be solved by computing the intersection points of three spheres (shown in Figure 3.2) of radius $l$ with the following centres:

$$\begin{aligned}
{}_A\mathbf{r}_{J_1} &= \big(R - r + L\cos(\theta_1)\big)\,{}_A\mathbf{e}_x - \sin(\theta_1)\,{}_A\mathbf{e}_z,\\
{}_A\mathbf{r}_{J_2} &= \mathbf{C}_z(120^o)\Big(\big(R - r + L\cos(\theta_2)\big)\,{}_A\mathbf{e}_x - \sin(\theta_2)\,{}_A\mathbf{e}_z\Big),\\
{}_A\mathbf{r}_{J_3} &= \mathbf{C}_z(240^o)\Big(\big(R - r + L\cos(\theta_3)\big)\,{}_A\mathbf{e}_x - \sin(\theta_3)\,{}_A\mathbf{e}_z\Big),
\end{aligned}$$

where $R, r, L$ correspond to the arm physical parameters shown in 3.2, ${}_A\mathbf{e}_x = [1,0,0]^\top$, ${}_A\mathbf{e}_z = [0,0,1]^\top$ and $\mathbf{C}_z(120^o), \mathbf{C}_z(240^o)$ rotation matrices of $120^o$ and $240^o$ degrees around ${}_A\mathbf{e}_z$. The maximum number of intersection points is two and this corresponds to an end effector position above (${}_A\mathbf{r}_{E_z} > 0$) and below (${}_A\mathbf{r}_{E_z} < 0$) the arm base. Solutions above the arm base are mechanically impossible for the delta arm and thus the forward kinematics solver returns the solution that satisfies ${}_A\mathbf{r}_{E_z} < 0$. For the inverse kinematics the intersection between a sphere with radius $l$ and a circular disk with radius $L$ has to be computed for every joint angle. For the first joint as shown in Figure 3.2 the centre of the sphere is ${}_A\mathbf{r}_{P_1} = {}_A\mathbf{r}_E + r\,{}_A\mathbf{e}_x$ with ${}_A\mathbf{e}_x = [1,0,0]^\top$ while the centre of the circular disk is ${}_A\mathbf{r}_{S_1} = R\,{}_A\mathbf{e}_x$. Given the intersection point ${}_A\mathbf{r}_{I_1}$, the joint angle can be recovered as $\theta_1 = \arcsin({}_A\mathbf{r}_{I_{1z}}/L)$. The joint angles $\theta_2$ and $\theta_3$ can be computed by performing the same procedure for the spheres with centres ${}_A\mathbf{r}_{P_2}, {}_A\mathbf{r}_{P_3}$, same radius $l$ and the unit disks centred at ${}_A\mathbf{r}_{S_2}$

and ${}_A\mathbf{r}_{S_3}$ with radius $L$. The points ${}_A\mathbf{r}_{P_i}, {}_A\mathbf{r}_{S_i}$ $\forall i = 2, 3$ can be easily computed as follows:

$$
\begin{aligned}
{}_A\mathbf{r}_{P_2} &= {}_A\mathbf{r}_E + r\ \mathbf{C}_z(120^o)\,{}_A\mathbf{e}_x, && (3.10a)\\
{}_A\mathbf{r}_{P_3} &= {}_A\mathbf{r}_E + r\ \mathbf{C}_z(240^o)\,{}_A\mathbf{e}_x, && (3.10b)\\
{}_A\mathbf{r}_{S_2} &= \mathbf{C}_z(120^o)\,{}_A\mathbf{r}_{S_1}, && (3.10c)\\
{}_A\mathbf{r}_{S_3} &= \mathbf{C}_z(240^o)\,{}_A\mathbf{r}_{S_1}. && (3.10d)
\end{aligned}
$$

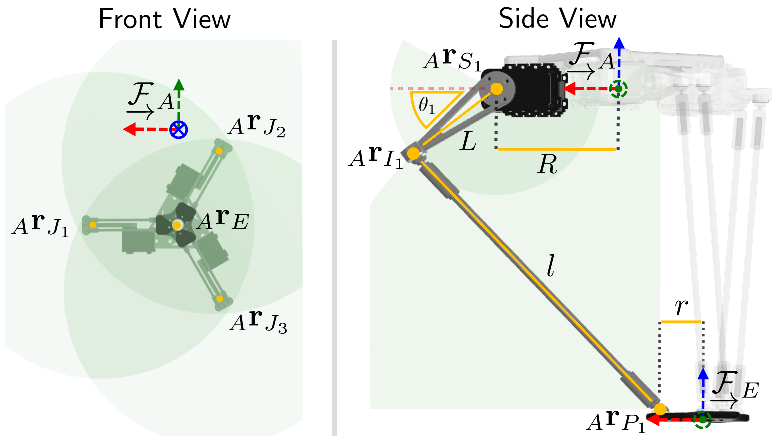


Figure 3.2: Two different views of a 3D model of the delta arm used. The green areas show the virtual spheres and disks used for the solution of the forward and inverse kinematics.

# Chapter 4

# System Overview

## 4.1 Pre-existing NMPC Method

### 4.1.1 Software Components

An overview of the different software components of the proposed system is outlined in Figure 4.1. The NMPC is given full state trajectory commands for the MAV and the end effector corresponding to a given aerial manipulation task. Based on those references and the estimated system state, it produces the desired MAV body moments, collective thrust, and end effector position. The control allocation block is responsible for converting the moments and thrust into individual motor commands while the inverse kinematics block computes the desired link angles for the given end effector position. All algorithms run onboard at a rate of 100 Hz.

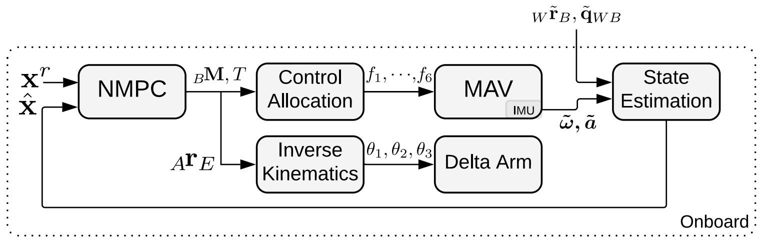


Figure 4.1: An overview of the software running onboard the MAV in an aerial manipulation task using the pre-existing NMPC. ROS is used to interface with the MAV and the motion capture system while all the other software blocks are implemented as a single executable.

### 4.1.2 Real-world Drone Platform

The real-world experiments presented in section 6.2 were performed using a custom built hexacopter equipped with a sideways mounted delta arm manipulator. The MAV features a frame with a 550mm diameter, a Pixracer flight controller running a modified version of the PX4 firmware, and an Intel NUC-7567U onboard computer running Ubuntu 16.04. It uses 960KV motors and DJI 9450 propellers. The delta arm uses magnetic universal joints for the connection of the servos with the end effector which maximises the workspace, minimises backlash and allows the arm to disassemble during possible crashes preventing it from breaking. The system is powered by a 4S 4500mAh battery and has total weight of 2.6 kg. A photo showing

Table 4.1: Numeric values of real-world MAV and Arm parameters

| | | | |
|---|---|---|---|
| $J_x$ | $0.042\ \mathrm{kg\,m^2}$ | $k_s$ | $42.95\ \mathrm{N\,m^{-1}}$ |
| $J_y$ | $0.054\ \mathrm{kg\,m^2}$ | $R$ | 7.2 cm |
| $J_z$ | $0.110\ \mathrm{kg\,m^2}$ | $r$ | 2.5 cm |
| $k_m$ | $1.58 \times 10^{-2}$ Nm/N | $L$ | 6.5 cm |
| $m_e$ | 0.058 kg | $l$ | 20.2 cm |

the used platform and its different components is shown in Figure 4.2. The arm uses three Dynamixel AX-18A servo motors which are comparably fast and accurate but have limited maximum torque. The end effector holds the pen which is mounted on a spring to provide additional compliance. We set the coefficient of the contact model in Equation 3.4 to match the used spring. The applied force is measured by a SingleTact force sensor mounted at the end of the spring. We estimate the spring coefficient $k_s$ by measuring the applied force for known tip displacements. The dimensions of the delta arm are based on a highly detailed CAD file and were verified manually. We measured the inertia of the MAV $\mathbf{J}$ by checking its angular response to constant input torque while it is hanging to freely rotate. The thrust to moment coefficient $k_m$ is measured using a trust stand. A table with the numeric values of the system parameters is given in Table 4.1.

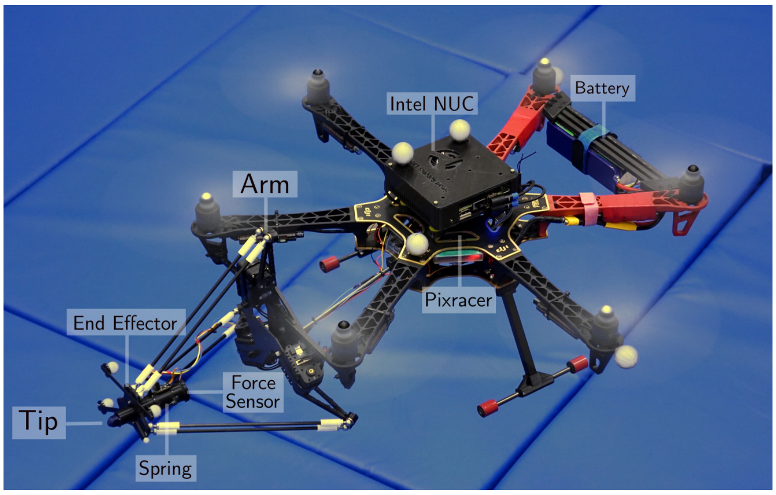


Figure 4.2: The aerial manipulation platform used in the Aerial Writing experiments in described Section 6.2 with labels for its individual components.

## 4.2 Multimodal Feedback Pipeline

### 4.2.1 Software Components

In addition to the NMPC pipeline illustrated in Figure 4.1, the multimodal feedback requires processing of the visual and tactile measurements to inform the multimodal feedback components. An overview of the combined software stack is given in Figure 4.3. This illustration is very exhaustive and refers to many details only presented later on in Section 5. The reader is advised to come back after finishing that section to get a in-depth understanding of the software stack.

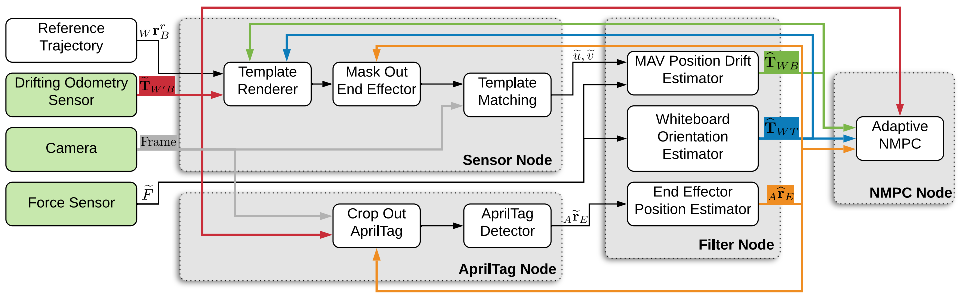


Figure 4.3: An overview of the software running onboard the MAV in an aerial manipulation task using the extended multimodal feedback pipeline. ROS is used to interface with the MAV and its sensors as well as to parallelise the computationally demanding template rendering and the time-critical state updates.

The three sensors (camera, force sensor and drifting SLAM) are shown in light green on the left. The Sensor Node takes care of the computationally demanding visual processing. Based on the reference trajectory and the drifting odometry estimate (red), it renders the template image and masks out the end effector, both as described in Section 5.2.2. This further requires the latest estimates of the non-drifting MAV position (green), the Whiteboard orientation (blue) and the end effector position (orange). Combining this with the current camera image, the Sensor Node performs the template matching and passes the resulting pixel measurements $\tilde{u}, \tilde{v}$ to the Filter Node. The detection of the AprilTag mounted on the end effector is done in a separate node. The off-the-shelf detection algorithm runs on the camera image after cropping out only the expected AprilTag position plus some margin around it. This is done to speed up the detection and subsequent pose estimation. The pose measurements based on the detections are passed on. All computations in the Sensor Node are callback-driven and occur at 30 Hz according to the frame rate of the camera. The Filter Node takes care of time critical filter updates for the MAV Position Drift Estimator, the Whiteboard Orientation Estimator and the End Effector Position Estimator. These are explained in Section 5. The Filter Node then informs the NMPC Node by updating the estimates of the MAV Position with respect to the non-drifting World frame and corrected Touch frame Orientation as well as providing the end effector position estimate used in the Relative End Effector Controller. All computations in the Filter and AprilTag Node are callback-driven and happen either at 100 Hz for tactile updates or at 30 Hz for visual updates.

### 4.2.2 Aerial Writing Simulation

To simplify the development of the feedback pipeline the whole process should be mapped to a simulation environment. To this end, the popular Gazebo [50] framework is used as the basis for the simulator. To bootstrap the simulation of MAVs, Gazebo is extended using RotorS [51] giving easy access to various MAV models such as the Firefly used in this work and shown in Figure 4.5 as well as various virtual sensors such as IMU or odometry.

The actual Aerial Writing is simulated through small line segments within the 3D world of the simulator. To improve computational performance, the segments are generated in the beginning of the simulation and only shifted around afterwards. However, the maximal number of segments must still be limited to a couple hundred in order for the simulation to run at least at half the speed of real-time. Depending on the font thickness, the amount of line segments is not enough for the entire text to be drawn. If this is the case, the segment that was shifted the longest ago is moved. In other words, the line segments are shifted in a circular FIFO manner. The segments are blue-coloured square tiles with the dimensions matching the font thickness. To optimally use the number of segments feasible to simulate, they are put next to each other such that they form a continuous line. Their orientation is adjusted to match the velocity of the pen when the respective segment is placed. Figure 4.4 shows a few line segments up close. The approach of simulating Aerial Writing through full interaction with the environment in the form of actual drawing while further running vision in the loop is claimed to be novel according to the author's best belief. It is stated as contribution two in Section 1.2.

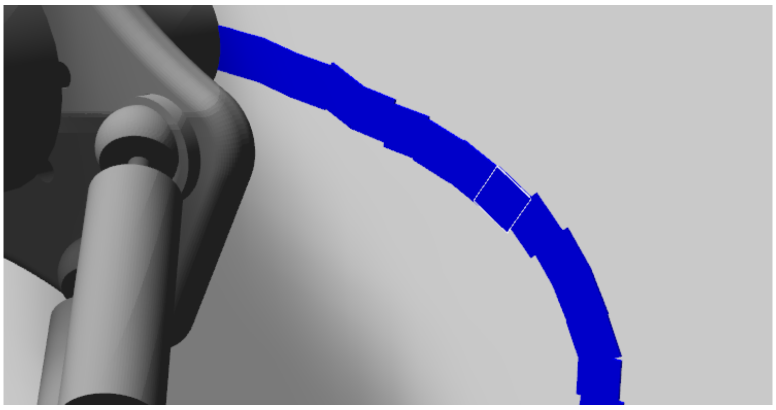


Figure 4.4: A close up of the line segments shifted in the 3D space of the simulator to imitate the actual writing process.

### 4.2.3 Simulator Drone Platform

A virtual model of a Firefly is used as the simulated MAV as depicted in Figure 4.5. A visually and physically accurate replica of the delta arm is attached to its front. It uses three simulated servo motors to move the end effector which always remains parallel to the base plate. Similarly to the real system, each servo uses an internal PID controller to reach angle setpoints sent to it by the NMPC pipeline.

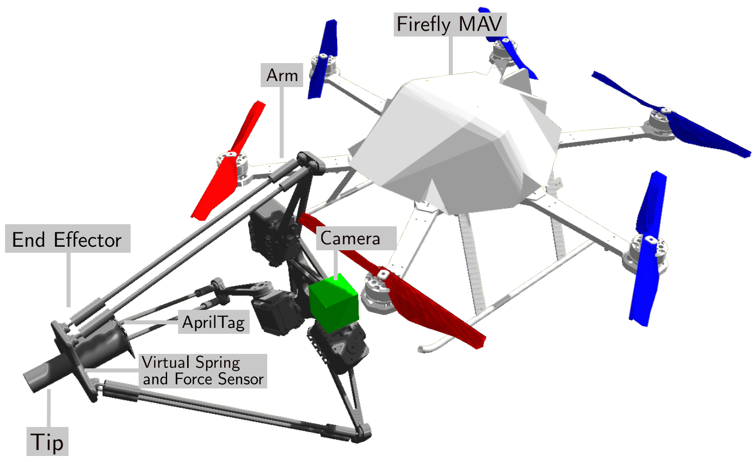


Figure 4.5: The aerial manipulation platform used in the Aerial Writing simulations presented in Section 6.3 with labels for its individual components.

Table 4.2: Numeric values of simulated MAV and Arm parameters

| | | | |
|---|---|---|---|
| $J_x$ | $0.035\ \mathrm{kg\,m^2}$ | $k_s$ | $5.0\ \mathrm{N\,m^{-1}}$ |
| $J_y$ | $0.046\ \mathrm{kg\,m^2}$ | $R$ | 7.2 cm |
| $J_z$ | $0.097\ \mathrm{kg\,m^2}$ | $r$ | 2.5 cm |
| $k_m$ | $1.6 \times 10^{-2}$ Nm/N | $L$ | 6.6 cm |
| $m_e$ | 0.020 kg | $l$ | 21.8 cm |

At the tip of the end effector, the pen is mounted to the pen case using a simulated spring. This simulated spring also features a virtual force sensor which measures the pressure acting from the pen onto the end effector. On the back of the pen case, an AprilTag is attached which is used for visual estimation of the true end effector position. To provide the required visual information for this and other visual feedback components, a virtual camera is mounted near the delta arm base plate. The camera has a frame rate of 30 Hz, a rather low resolution of $640 \times 480$ pixels and no camera distortion is assumed. It is free-floating for simplicity but could easily be attached in such a configuration using a 3D printed mount. Hidden within the MAV, an odometry sensor provides localisation data as a SLAM system running onboard would do. As explained in section 5.2 this signal is assumed to be drifting over time. Table 4.2 shows the numeric values of the simulated system parameters. In comparison with the real-world system, the spring is chosen to be less stiff to allow more accurate tracking of force references due to increased compliance. All other parameters are equal or similar enough to not cause meaningful differences in simulation.

## 4.3 Trajectory Generation

As part of this thesis, a trajectory generator was developed to map arbitrary sets of characters to end effector trajectories. This component is used for both the real and simulated MAV-arm system.
To this end, the input sequence of letters is separated into smooth segments by detecting sharp corners and splitting the trajectory there. Corners are detected based on the angle spanned by the two lines defined through the current candidate trajectory point and its predecessor as well as successor, respectively. The contact segments are then connected by segment flown in detached mode. A constant acceleration motion model is applied to generate trajectories with a smooth velocity profile. This is of special importance to ensure dynamic feasibility when approaching sharp corners at the end of a segment, i.e. to make sure the pen reaches zero velocity before doing drastic direction changes or detaching. An example of a generated trajectory is shown in Figure 4.6 with the line coloured according to the end effector velocity. The trajectory generator allows for adjustment to the velocity and acceleration profile by changing the maximum $\|{}_W\mathbf{v}^r_E\|$ and $\|{}_W\mathbf{a}^r_E\|$. It should be noted here that the trajectory generator is based on a previous implementation by Qingyue Yan but has been changed significantly over the course of this thesis.
Once the trajectory for the end effector has been computed, the computation of the reference position ${}_W\mathbf{r}^r_B$ and velocity ${}_W\mathbf{v}^r_B$ for the MAV is performed as follows:

$$ {}_W\mathbf{r}^r_B = {}_W\mathbf{r}^r_E - \mathbf{C}_{WB}\,{}_B\mathbf{r}_{E_0}, \tag{4.1a}$$

$$ {}_W\mathbf{v}^r_B = {}_W\mathbf{v}^r_E - \mathbf{C}_{WB}\,{}_B\boldsymbol{\omega}^r \times {}_B\mathbf{r}_{E_0}. \tag{4.1b}$$

Here, $\mathbf{C}_{WB}$ is the rotation between the MAV Body and World frame, ${}_B\mathbf{r}_{E_0}$ denotes the nominal end effector position which results in no CoM displacement and ${}_B\boldsymbol{\omega}^r$ is the angular velocity of the MAV. The reference MAV orientation $\mathbf{q}_{WB}^r$ is chosen such that the end effector position is always perpendicular to the contact surface, assuming perfect position tracking. Each trajectory is accompanied by an appropriate flag which disables or enables the position tracking for the end effector. This is achieved by setting the appropriate gains to zero. In that case the NMPC may decide to move the arm to assist the reference tracking of the MAV due to the CoM displacement. This potentially unwanted behaviour can be avoided by further penalising (i.e. by increasing the input gains) the arm displacement from its nominal position. However, it is an interesting capability enabled by the hybrid modelling explained in Section 3.2.

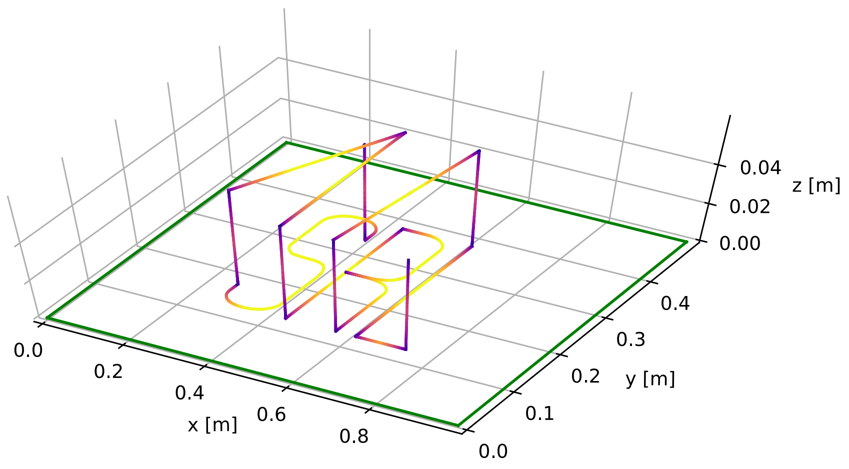


Figure 4.6: The generated dynamically feasible trajectory used in the simulations presented in Section 6.3 with the end effector velocity reference coloured in by going from dark violet, meaning slow, to bright yellow, meaning fast.

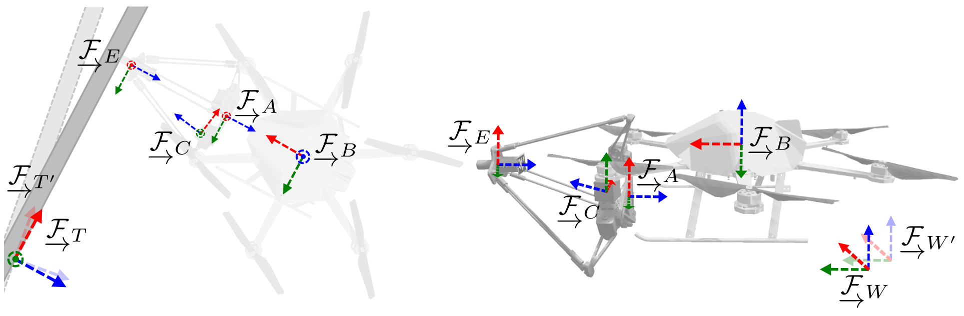
# Chapter 5

# Multimodal Feedback

As mentioned previously, the pre-existing NMPC method described in Section 3 and evaluated in Section 6.2 relies on external MAV and whiteboard pose measurements while also assuming zero-order delta arm dynamics and perfect knowledge of its model parameters. In this chapter, the three individual feedback components to address these potential short-comings are explained. They are informed using multimodal measurements from both a camera and a force sensor and should allow the current aerial manipulator to be used for real-world tasks outside of laboratory conditions. Further, it is explained how each error source is simulated and how this relates to the expected error seen in a real-world environment. The proposed pipeline is novel in the sense that it combines task specific visual and tactile feedback for Aerial Writing and is hence listed as the main contribution in Section 1.2.

## 5.1 Coordinate Frames

Before introducing the actual method, Figure 5.1 shows the relevant coordinate frames used in the multimodal feedback pipeline. When compared with the version for the pre-existing NMPC in Figure 3.1, the Camera frame $\underrightarrow{\mathcal{F}}_{C}$ as well as the drifting World frame $\underrightarrow{\mathcal{F}}_{W'}$ and the rotated Touch frame $\underrightarrow{\mathcal{F}}_{T'}$ are added. The latter two are defined implicitly by assuming both a drifting MAV position as well as an offset whiteboard orientation. Following this terminology, the true frames $\underrightarrow{\mathcal{F}}_{W}$ and $\underrightarrow{\mathcal{F}}_{T}$ shown in the illustration are not known to the MAV and estimated by the multimodal feedback.



$\underrightarrow{\mathcal{F}}_{E}$
$\underrightarrow{\mathcal{F}}_{A}$
$\underrightarrow{\mathcal{F}}_{C}$
$\underrightarrow{\mathcal{F}}_{B}$
$\underrightarrow{\mathcal{F}}_{T'}$
$\underrightarrow{\mathcal{F}}_{T}$
$\underrightarrow{\mathcal{F}}_{E}$
$\underrightarrow{\mathcal{F}}_{B}$
$\underrightarrow{\mathcal{F}}_{A}$
$\underrightarrow{\mathcal{F}}_{C}$
$\underrightarrow{\mathcal{F}}_{W'}$
$\underrightarrow{\mathcal{F}}_{W}$

Figure 5.1: The different coordinate frames used for the multimodal feedback. Specifically $\underrightarrow{\mathcal{F}}_{W}$, $\underrightarrow{\mathcal{F}}_{B}$, $\underrightarrow{\mathcal{F}}_{C}$, $\underrightarrow{\mathcal{F}}_{A}$, $\underrightarrow{\mathcal{F}}_{E}$, and $\underrightarrow{\mathcal{F}}_{T}$ stand for the World, MAV Body, Camera, Arm, End Effector, and Touch (contact) frame, respectively. Further, $\underrightarrow{\mathcal{F}}_{W'}$ and $\underrightarrow{\mathcal{F}}_{T'}$ denote the drifting World frame and the rotated Touch frame, respectively.

## 5.2 MAV Position Drift

In the following, the drift simulation is explained before introducing the MAV Position Drift Estimator and describing its process and measurement model as well as the respective EKF updates.

### 5.2.1 Drift Simulation

The MAV pose measurement is simulated to be drifting based on the assumption of running an onboard Visual-Inertial odometry system. As explained in Section 2.2.1, in a general setup without loop closure, one could expect the drift error to grow indefinitely as the MAV moves through space. However, for most aerial manipulation tasks, the MAV would actually remain within a more local region and hence loop closure could happen regularly. In other words, the VI-SLAM system keeps relating to the same visual landmarks. Therefore, one would expect the drift error to remain bounded. To mimic this behavior, the drift simulation is based on a reverting zero-mean random walk modeled by an *Ornstein–Uhlenbeck process.* More specifically, the pose estimate would be expected to oscillate around a slowly shifting 'mean' MAV position which in itself drifts due to biases in the IMU. Such IMU biases are ignored in the drift simulation and hence the 'mean' MAV position remains at zero. The drift is assumed to affect the position only, orientation as well as linear and angular velocities are assumed to be correct. Further, all three axes are assumed to be affected independently. The simulated drift is expressed in the World frame $\underrightarrow{\mathcal{F}}_W$ and termed ${}_W\Delta\mathbf{r}_k$. At every time step $k$ of the simulation, we compute the drift as follows:

$$
{}_W\Delta\mathbf{r}_k = \begin{bmatrix} \Delta x_k \\ \Delta y_k \\ \Delta z_k \end{bmatrix} = \left(1 - \frac{\Delta t}{\tau}\right) {}_W\Delta\mathbf{r}_{k-1} + \begin{bmatrix} w_{p,x} \\ w_{p,y} \\ w_{p,z} \end{bmatrix}, \tag{5.1}
$$

$$
w_{p,i} \sim \mathcal{N}(0, \Delta t \sigma_p^2) \quad \text{with} \quad i \in \{x, y, z\}. \tag{5.2}
$$

Here, $\Delta t$ is the time since the simulation step and the time constant $\tau$ captures how quickly the drift reverts back to zero. The variables $w_{p,i}$ denote additive white noise which, for each time step and axis, is drawn from a zero-mean Gaussian distribution which spreads wider as the time step $\Delta t$ increases.
The drift is added onto the ground-truth MAV position ${}_W\mathbf{r}_B$ to form the position part of the drifting odometry measurement ${}_{W'}\mathbf{r}_B$ provided by the SLAM system. As mentioned, a simplification is made by neither adjusting the orientation $\mathbf{q}_{WB}$ nor the linear and angular velocities of the true MAV pose. Omitting drift on the MAV yaw is based on the observation that in practice, this DoF will only drift slowly. Based on this, the relationship between the true World frame $\underrightarrow{\mathcal{F}}_W$ and the drifting equivalent $\underrightarrow{\mathcal{F}}_{W'}$ can be written through a homogeneous coordinate transform as

$$
\boldsymbol{T}_{W'B} = \left[\begin{array}{ccc:c} & \mathbf{C}_{WB} & & {}_W\mathbf{r}_B + {}_W\Delta\mathbf{r} \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right]. \tag{5.3}
$$

Note that since we assume only translational drift, the odometry orientation $\mathbf{q}_{WB}$ is correct. To illustrate the nature of the simulated drift, Figure 5.2 shows the odometry position signal provided to the MAV along the $x$ axis, ${}_{W'}\mathbf{r}_B^x$, and the associated ground-truth value, ${}_W\mathbf{r}_B^x$, which is fixed to zero. By adjusting the numeric values of $\sigma_p$ and $\tau$ the per step noise and the magnitude of oscillations around the mean can be changed.

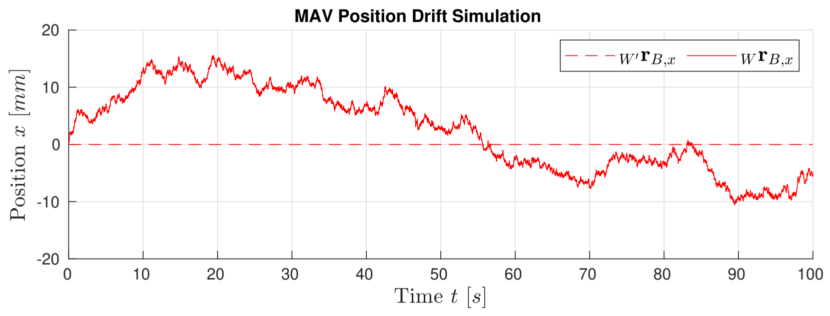


Figure 5.2: An example of simulated drift using the reverting-mean random walk model. The true value of the MAV position is kept at the origin, i.e. is zero.

### 5.2.2 Drift Estimation

An Extended Kalman Filter is used to estimate the current drift to then compute the MAV position with respect to the true World frame. In other words, the goal is to estimate the difference along all axes between the MAV position in the true World frame $\underrightarrow{\mathcal{F}}_W$ and the drifting World frame $\underrightarrow{\mathcal{F}}_{W'}$. We define the EKF state vector $\widehat{\mathbf{x}}_k$ as follows:

$$\widehat{\mathbf{x}}_k = \begin{bmatrix} \Delta\widehat{x}_k \\ \Delta\widehat{y}_k \\ \Delta\widehat{z}_k \end{bmatrix} . \tag{5.4}$$

As for the drift simulation, $\Delta\widehat{x}, \Delta\widehat{y}, \Delta\widehat{z}$ represent the estimated position drift along all axes expressed in the World frame. The raw odometry sent by the SLAM system carries the MAV pose with respect to the drifting frame as shown in Equation 5.5a. By subtraction of the EKF state, this can be transformed into the MAV pose with respect to the non-drifting World frame following Equation 5.5b:

$$\boldsymbol{T}_{W'B} = \left[\begin{array}{ccc:c} \multicolumn{3}{c:}{\mathbf{C}_{WB}} & {}_{W'}\mathbf{r}_B \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] , \tag{5.5a}$$

$$\widehat{\boldsymbol{T}}_{WB} = \left[\begin{array}{ccc:c} \multicolumn{3}{c:}{\mathbf{C}_{WB}} & {}_{W'}\mathbf{r}_B + -\widehat{\mathbf{x}}_k \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] . \tag{5.5b}$$

#### Prediction

As mentioned in Section 5.2.2, the drift is modeled as a random walk with reverting mean. Hence, the EKF state prediction simplifies to adding noise to a damped version of last iteration's value. The resulting process model is given by

$$\widehat{\mathbf{x}}_{k|k-1} = \left(1 - \frac{\Delta t}{\tau}\right) \widehat{\mathbf{x}}_{k-1|k-1} + \begin{bmatrix} w_{p,x} \\ w_{p,y} \\ w_{p,z} \end{bmatrix} , \tag{5.6}$$

$$w_{p,i} \sim \mathcal{N}(0, \Delta t \sigma_{p,i}^2) \quad \text{with} \quad i \in \{x, y, z\}. \tag{5.7}$$

Since the drift is simulated, $\tau$ and $\sigma_p$ are known exactly. In a real-world setting where drift is caused by a SLAM system, these parameters would have to be identified to model the drift as closely as possible. From the process model, the prediction for the filter state $\widehat{\mathbf{x}}_k$ and its associated covariance matrix $\mathbf{P}_k$ are derived as

$$\widehat{\mathbf{x}}_{k|k-1} = \left(1 - \frac{\Delta t}{\tau}\right) \widehat{\mathbf{x}}_{k-1|k-1}, \tag{5.8}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_p, \tag{5.9}$$

$$\mathbf{Q}_p = \operatorname{diag}(\Delta t \sigma_p^2). \tag{5.10}$$

The matrix $\mathbf{Q}_p$ captures the process noise and $\mathbf{F}_k$ is the trivial Jacobian of the process model and equals $\mathbf{F}_k = \operatorname{diag}(1 - {}^{\Delta t}/_{\tau})$.

### Measurements

The measurement vector $\widetilde{\mathbf{z}}_k$ consists of a visual template matching pixel location, $\widetilde{u}, \widetilde{v}$, as well as a tactile measurement $\widetilde{d}$ which is denoted as follows:

$$\widetilde{\mathbf{z}}_k^v = \begin{bmatrix} \widetilde{u} \\ \widetilde{v} \end{bmatrix}, \tag{5.11}$$

$$\widetilde{z}_k^t = \widetilde{d}. \tag{5.12}$$

The measurements $\widetilde{u}, \widetilde{v}$ and the associated visual measurement function $\mathbf{h}^v$ are explained in subsection 'Visual Measurements' whereas $\widetilde{d}$ and the associated tactile measurement function $h^t$ are described in subsection 'Tactile Measurements'. The measurement functions are defined separately from one another as follows:

$$\begin{bmatrix} \widehat{u} \\ \widehat{v} \end{bmatrix} = \mathbf{h}^v(\widehat{\mathbf{x}}_{k|k-1}), \tag{5.13}$$

$$\widehat{d} = h^t(\widehat{\mathbf{x}}_{k|k-1}). \tag{5.14}$$

To complete the formulation of the measurement models, noisy measurement processes are assumed:

$$\widehat{\mathbf{z}}_k^v = \mathbf{h}^v(\widehat{\mathbf{x}}_{k|k-1}) + \begin{bmatrix} v_u \\ v_v \end{bmatrix}, \tag{5.15}$$

$$\widehat{z}_k^t = h^t(\widehat{\mathbf{x}}_{k|k-1}) + v_d. \tag{5.16}$$

Here, the components of the measurement noise $v_i, i \in \{u, v, d\}$ are explained in more detail in the following. Since the measurement update depends on whether a visual or tactile measurement is being fused into the filter, each equation is given separately after explaining the respective measurement model.
For all measurement updates presented in this thesis, outlier rejection schemes based on $\chi^2$-tests are used. In the interest of space, the general idea is given here and the specific implementations are not described. Following the approach in [52], the measurement residual $\mathbf{y}$ and residual covariance $\mathbf{S}$ are used to define the validation gates in Equation 5.17c:

$$\mathbf{S} = \mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R}, \tag{5.17a}$$

$$\mathbf{y} = \widetilde{\mathbf{z}} - \mathbf{h}(\widehat{\mathbf{x}}), \tag{5.17b}$$

$$e^2 = \mathbf{y}\mathbf{S}^{-1}\mathbf{y}^T \leq g^2. \tag{5.17c}$$

Here, $\mathbf{H}$ is the Jacobian of the measurement model and $\mathbf{R}$ the measurement covariance. If a measurement leads to a value of $e^2$ higher than $g^2$ allows, it is considered an outlier and rejected. Here, $g^2$ depends on the desired confidence level and number of DoFs, i.e. the number of dimensions in the measurement vector.

### Visual Measurement

The visual measurement is based on performing template matching between the current camera image and a rendering of what the camera mounted on the MAV should be seeing according to the planned text to be drawn. In the following, this approach is explained step-by-step.

***Reference Drawing*** The first concept to be introduced is the reference drawing shown in Figure 5.3. While the MAV is fulfilling its drawing task, a global reference drawing is updated continuously. At every iteration, the current reference for the pen tip, ${}_T\mathbf{r}_P$, is coloured in the reference drawing which represents the entire whiteboard surface in an image. Hence, the reference drawing carries the answer to the question 'What should the MAV have been drawing up until this point in time?'

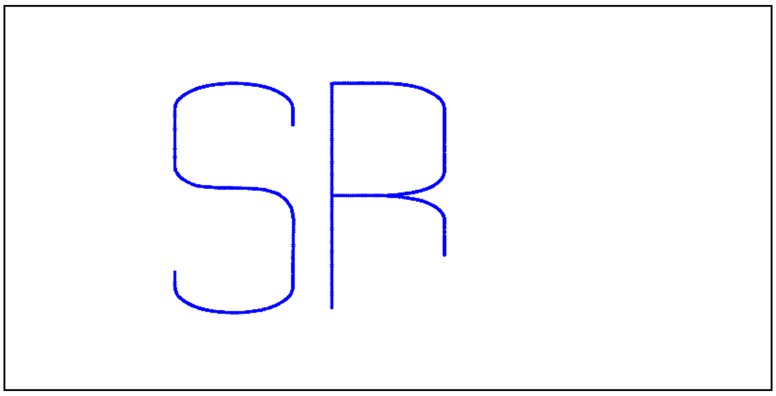


Figure 5.3: An example of the reference drawing showing what lines the MAV should have been writing onto the whiteboard until the current time step.

***Template Image*** The reference drawing is then projected into a virtual camera positioned at $\boldsymbol{T}_{WC}$ . This rendering will be referred to as the template image. On the left of Figure 5.4, the template image corresponding to the reference drawing in Figure 5.3 is shown. To make the rendered template image correspond to what the actual camera sees, areas that are covered by the end effector get masked out in the template image. In other words, occlusions caused by the end effector are replicated in the template image. Occlusions caused by the push rods are not considered as they are much smaller. The mask is generated by adding constant offsets to the current end effector position estimate ${}_B\hat{\mathbf{r}}_E$ to form its corner points which are then projected into the virtual camera giving a 2D mask. The end effector position is estimated based on the AprilTag attached on its back which is explained in more detail in Section 5.4.2.

***Real Image*** At the same time, the camera perceives the image shown on the right in Figure 5.4. To ensure temporal consistency, the aforementioned mask is chosen to be slightly bigger than the actual occlusion and then also applied to the real image (not shown in Figure 5.4). However, in the current implementation no tight time-synchronisation is performed as seen in various VI-SLAM systems, e.g. by using the same trigger signal for the virtual camera and the rendering of the template image.

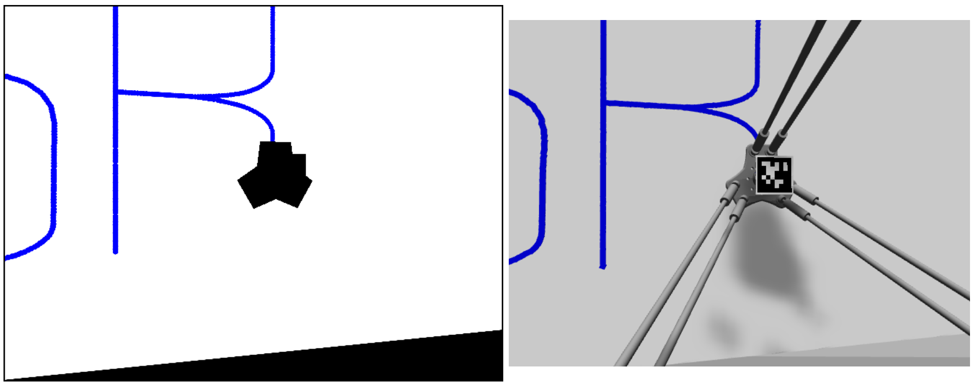


Figure 5.4: The template image (left) showing what parts of the drawing the camera is expected to see given its current position; and the end effector position and the real image (right) received at roughly the same time as template image was rendered.

***Colour Filtering*** To increase robustness and performance of the subsequent matching, both the template and camera image are filtered to only include the drawing. As the pen colour is blue, the filtering can be done effectively through colour segmentation in HSV space.

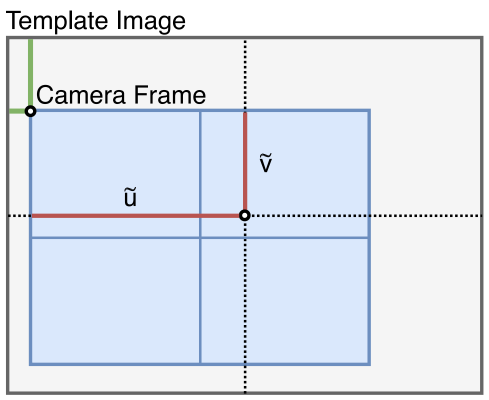
***Template Matching*** Based on the camera image and the template image, one can determine how far the camera drifted away from its expected position. This is done by registering the camera image to the template through template matching.

The actual measurements $\widetilde{u}, \widetilde{v}$ are the pixel coordinates of the template image centre after being matched to the camera image as previously described by Equation 5.11. This is illustrated on the left in Figure 5.5 where the template image is shown in grey and the camera image in blue. To arrive at $\widetilde{u}, \widetilde{v}$, the match location (shown in green) is subtracted from the size centre point of the template image (shown in red). From this illustration, it can further be seen that in order to measure deviations between the expected and actual camera position, the FoV of the virtual camera has to be bigger than the real one. By scaling the FoV ratio, a limit on how far the MAV position is expected to drift away from its measured value can be set.

The template matching is performed using the standard OpenCV [53] implementation. To allow real-time performance, a fast matching metric based on the Normalised Squared Sum of Differences is chosen. When matching the camera image **C** with the template image **T**, the former is shifted across the latter computing the template matching metric shown in Equation 5.18:

$$R(x,y) = \frac{\sum_{x',y'} \left(C(x',y') - T(x+x',y+y')\right)^2}{\sum_{x',y'} C(x',y')^2 \ \sum_{x',y'} T(x+x',y+y')^2}. \tag{5.18}$$

The best match is located at the minimum location of the 2D matching metric matrix. An example of such a result matrix is shown on the right in Figure 5.5 which further encodes the measurement uncertainty shown as the green ellipse and explained in more detail later on. To perform all these steps in real-time, the camera resolution is chosen to be rather low at $640 \times 480$ pixels.



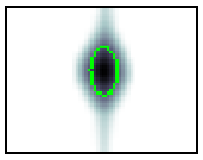


Figure 5.5: Left: Illustration of how the visual error measurement is computed. The blue frame corresponds to the camera image while the template image is grey. The template match location is shown in green and the actual measurement in red. Right: The corresponding matching metric matrix (inverted for visualisation). The ellipse shown in green is used to estimate the visual measurement uncertainty.

### Visual Measurement Model

The remaining question is how the visual measurement and the EKF state are associated, i.e. the measurement model $\mathbf{h}^v$. To better understand the following equations, please refer to Figure 5.6 which visualises all the following steps.

***Camera Ray*** By projecting the camera image centre point out of the camera based on its expected position ${}_W\mathbf{r}_C$ one arrives at the ray vector ${}_W\mathbf{k}$ shown in blue in Figure 5.6. This ray can be computed by rotating a unit-vector along the z axis of the Camera frame, ${}_C\mathbf{e}_z$, by the orientation of the camera with respect to the World frame, $\mathbf{C}_{WC}$ :

$$ {}_W\mathbf{k} = \mathbf{C}_{WC}\, {}_C\mathbf{e}_z. \tag{5.19}$$

***Intersection*** Intersecting the camera ray with the whiteboard plane yields the nominal intersection point ${}_T\mathbf{r}_I$ shown as the left dot. Here, ${}_W\mathbf{n}_T$ denotes the surface normal of the whiteboard and ${}_W\mathbf{r}_T$ is the Touch frame origin. The intersection can be formulated as

$$ {}_T\mathbf{r}_I = {}_W\mathbf{r}_C - t\, {}_W\mathbf{k}, \tag{5.20}$$

$$ t = \frac{({}_W\mathbf{r}_C - {}_W\mathbf{r}_T) \cdot {}_W\mathbf{n}_T}{{}_W\mathbf{k} \cdot {}_W\mathbf{n}_T}. \tag{5.21}$$

***Transformation*** The nominal intersection point ${}_T\mathbf{r}_I$ is then transformed from Touch to Camera frame resulting in ${}_C\mathbf{r}_I$ shown as the right dot. Here, $\boldsymbol{T}_{CB}$ and $\boldsymbol{T}_{WT}$ are fixed and assumed to be known. $\widehat{\boldsymbol{T}}_{WB}$ depends on the EKF state as it is the estimate of the MAV pose in the non-drifting World frame. In summary, one can formulate

$$ {}_C\mathbf{r}_I = \boldsymbol{T}_{CB} \left(\widehat{\boldsymbol{T}}_{WB}\right)^{-1} \boldsymbol{T}_{WT}\, {}_T\mathbf{r}_I. \tag{5.22}$$

***Projection*** As shown in red, projection of this point back into the camera leads to the expected values for the visual measurements, i.e. $\widehat{u}$ and $\widehat{v}$. Here, $\mathbf{K}$ is the camera's intrinsic matrix with the per image axis focal lengths $f_u, f_v$ and principal points $c_u, c_v$. The z component of the nominal point, ${}_C r_{I,z}$, is used for normalisation. The resulting visual measurement model first introduced in Equation 5.13 can now be written as

$$\mathbf{h}^v(\mathbf{x}) = \frac{1}{{}_C r_{I,z}} \mathbf{K} \, {}_C\mathbf{r}_I, \tag{5.23}$$

$$\mathbf{K} = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \end{bmatrix}. \tag{5.24}$$

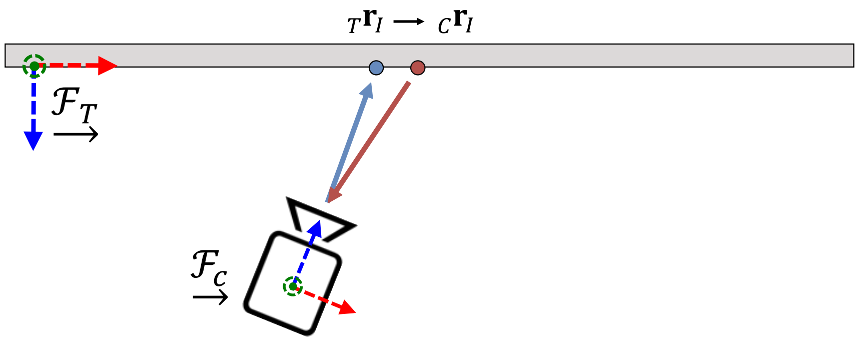

${}_T\mathbf{r}_I \longrightarrow {}_C\mathbf{r}_I$
$\underrightarrow{\mathcal{F}_T}$
$\underrightarrow{\mathcal{F}_C}$

Figure 5.6: Illustration of the visual measurement with the whiteboard and the camera mounted on the MAV.

In essence, the visual error presented here corresponds to a reprojection error: if the MAV position estimate $\widehat{\boldsymbol{T}}_{WB}$ is accurate, the nominal intersection point ${}_C\mathbf{r}_I$ (red) will be projected into the camera's centre and the visual EKF innovation would be zero. However, if the estimate is off, there will be a non-zero error between the two points which will effect the estimate. This allows the method to track the drift in a frame-to-frame manner. Based on this visual measurement model, the measurement step for an incoming camera image is defined as:

$$\mathbf{K}_k^v = \mathbf{P}_{k|k-1}\mathbf{H}_k^v(\widehat{\mathbf{x}}_{k|k-1})^T \left(\mathbf{H}_k^v(\widehat{\mathbf{x}}_{k|k-1})\mathbf{P}_{k|k-1}\mathbf{H}_k^v(\widehat{\mathbf{x}}_{k|k-1})^T + \mathbf{R}_k^v\right)^{-1}, \tag{5.25}$$

$$\mathbf{y}_k^v = \widetilde{\mathbf{z}}_k^v - \mathbf{h}^v(\widehat{\mathbf{x}}_{k|k-1}), \tag{5.26}$$

$$\widehat{\mathbf{x}}_{k|k} = \widehat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k^v\mathbf{y}_k^v, \tag{5.27}$$

$$\mathbf{P}_{k|k} = \left(\mathbf{I} - \mathbf{K}_k^v\mathbf{H}_k^v(\widehat{\mathbf{x}}_{k|k-1})\right)\mathbf{P}_{k|k-1}. \tag{5.28}$$

Here, $\mathbf{K}_k^v$ is the visual Kalman gain, $\mathbf{H}_k^v$ is the Jacobian of the visual measurement model and $\mathbf{R}_k^v$ the measurement noise. $\mathbf{y}_k^v$ denotes the visual EKF innovation. The visual Jacobian is defined by

$$\mathbf{H}_k^v(\mathbf{x}) = \begin{bmatrix} \partial u/\partial x_1 & \partial u/\partial x_2 & \partial u/\partial x_3 \\ \partial v/\partial x_1 & \partial v/\partial x_2 & \partial v/\partial x_3 \end{bmatrix}. \tag{5.29}$$

It captures the non-linear measurement model as a first-order approximation and allows the process uncertainty to be propagated correctly. It is derived symbolically using MATLAB and evaluated at run-time. In the interested of space, the full expression is omitted here. The measurement uncertainty $\mathbf{R}^v$ is contained implicitly in the matching metric matrix shown on the right in Figure 5.5. The green ellipse shown corresponds to a level-set of the underlying 2D Gaussian distribution describing the uncertainty in $\widetilde{u}$ and $\widetilde{v}$ as well as the correlation between them. As shown in [47], one can retrieve the noise characteristics from the matching metric matrix by appropriately scaling the error ellipse to contain 68% of the matching residual. The ellipse dimensions then represent a rotated version of the measurement noise covariance matrix as shown in Equation 5.30 where $a, b$ denote the semi-major and semi-minor axes of the ellipse:

$$\mathbf{R}^v_{xy} = \begin{bmatrix} a^2 & 0 \\ 0 & b^2 \end{bmatrix}. \tag{5.30}$$

By rotating $\mathbf{R}^v_{xy}$ by the ellipse angle $\theta$, one arrives at the respective covariance matrix in the $(u, v)$ pixel space which is used as the measurement uncertainty:

$$\mathbf{R}^v = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \mathbf{R}^v_{xy} \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}. \tag{5.31}$$

### Tactile Measurement

The second modality in use is tactile feedback from the force sensor mounted to measure the pressure on the pen. To get a better understanding of the variables used in the following, please refer to Figure 5.7. The measured force is denoted as $\widetilde{F}$ based on which one can estimate the displacement of the spring $\widetilde{s}$ as follows:

$$\widetilde{s} = \frac{\widetilde{F}}{k_s}. \tag{5.32}$$

The spring constant $k_s$ is required to be known in advance. Using the zero-displacement pen length $D$, the length of the part of the pen which sticks out of the end effector can then be expressed as done in Equation 5.33 as follows:

$$\widetilde{d} = D - \widetilde{s}. \tag{5.33}$$

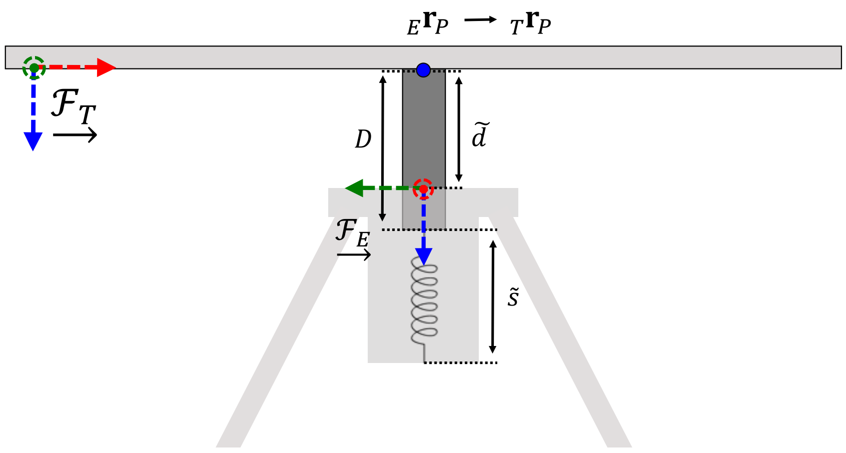
This is used as the actual measurement in the EKF updates which leads the tactile measurement defined previously in Equation 5.12.

### Tactile Measurement Model

As for the visual measurement, the remaining question is how this measurement relates to the current filter state, i.e. how the tactile measurement model is defined. Based on $d$, one can define the point where the pen touches the whiteboard as a simple $z$ offset in the End Effector frame as shown in Equation 5.34a. As expressed in Equation 5.34b contact point can then be transformed from the End Effector frame to the Touch frame using constant coordinate transforms and the MAV position estimate which depends on the EKF state. In combination, this yields

$$_E\mathbf{r}_P = \begin{bmatrix} 0 & 0 & -d \end{bmatrix}^T, \tag{5.34a}$$

$$_T\mathbf{r}_P = \boldsymbol{T}_{TW}\, \widehat{\boldsymbol{T}}_{WB}\, \boldsymbol{T}_{BA}\, \boldsymbol{T}_{AE}\, {_E\mathbf{r}_P}. \tag{5.34b}$$



${}_E\mathbf{r}_P \longrightarrow {}_T\mathbf{r}_P$
$\underrightarrow{\mathcal{F}_T}$
$D$
$\tilde{d}$
$\underrightarrow{\mathcal{F}_E}$
$\tilde{s}$

Figure 5.7: Illustration of the tactile measurement with the whiteboard and the end effector with the pen attached on its tip.

In the Touch frame, one can now define the **Contact Condition** which states that as long as the pen in touch with the whiteboard, the $z$ component of the contact point must be zero:

$$
{}_T\mathbf{r}_{P,z} \overset{!}{=} 0. \tag{5.35}
$$

This condition can now be solved for the expected tactile measurement $\widehat{d}$ which is done symbolically in MATLAB. Since the resulting expression for the tactile measurement model first introduced in Equation 5.14 is long and not insightful, it is not explicitly stated here. Based on this, the measurement step for an incoming tactile measurement is

$$
\mathbf{K}^t_k = \mathbf{P}_{k|k-1}\mathbf{H}^t_k(\widehat{\mathbf{x}}_{k|k-1})^T\left(\mathbf{H}^t_k(\widehat{\mathbf{x}}_{k|k-1})\mathbf{P}_{k|k-1}\mathbf{H}^t_k(\widehat{\mathbf{x}}_{k|k-1})^T + R^t_k\right)^{-1}, \tag{5.36}
$$

$$
y^t_k = \tilde{z}^t_k - h^t(\widehat{\mathbf{x}}_{k|k-1}), \tag{5.37}
$$

$$
\widehat{\mathbf{x}}_{k|k} = \widehat{\mathbf{x}}_{k|k-1} + \mathbf{K}^t_k\mathbf{y}^t_k, \tag{5.38}
$$

$$
\mathbf{P}_{k|k} = \left(\mathbf{I} - \mathbf{K}^t_k\mathbf{H}^t_k(\widehat{\mathbf{x}}_{k|k-1})\right)\mathbf{P}_{k|k-1}. \tag{5.39}
$$

Here, $\mathbf{K}^t_k$ is the tactile Kalman gain, $\mathbf{H}^t_k$ is the Jacobian of the tactile measurement model and $R^t_k$ the measurement noise. $y^t_k$ denotes the tactile EKF innovation. The tactile Jacobian is defined as:

$$
\mathbf{H}^t_k(\mathbf{x}) = \left[{}^{\partial d}/_{\partial x_1} \quad {}^{\partial d}/_{\partial x_2} \quad {}^{\partial d}/_{\partial x_3}.\right] \tag{5.40}
$$

Equivalent to the visual update, this captures the measurement model as a first-order approximation and allows the process uncertainty to be propagated. It is also derived symbolically using MATLAB and evaluated at run-time. Again, the full expression is omitted in the interested of space. As shown in Equation 5.16, the tactile measurement is assumed to be noisy. The tactile measurement noise covariance is defined as $R^t = \sigma^2_d$. The virtual force sensor is simulated accordingly, hence it provides measurements with noise covariance $\sigma_f$. The connection between the tactile measurement $d$ and the actual force measurement is defined by assuming linear error propagation as follows:

$$\sigma_d = \frac{\sigma_f}{k_s}. \tag{5.41}$$

As for the drift process noise characteristics, this parameter would have to be identified to match the force sensor used in a real-world experiment.

## 5.3 Whiteboard Orientation Misalignment

In the following, the whiteboard misalignment simulation is explained before introducing the Whiteboard Orientation Estimator and describing its process and measurement model as well as the respective EKF updates.

### 5.3.1 Misalignment Simulation

The whiteboard orientation is simulated to be misaligned based on the assumption of having a wrong calibration of the Touch frame $\underrightarrow{\mathcal{F}}_T$. This is based on experience gathered in the real-world experiments done with the pre-existing platform which are presented in Section 6.2. Even when using optimisation-based calibration operating on thousands of external pose measurements of the whiteboard surface, the found whiteboard orientation was usually off by up to 2 degrees while the position calibration was more accurate. Such small misalignments in the orientation calibration then lead to the MAV flying too close to the whiteboard which causes it to apply too much pressure and eventually to disassembly of the magnetic links in the delta arm. If rotated in the other direction, it leads to the MAV flying too far away and causes the pen to lose contact which interrupts or fully stops the writing.
The orientation misalignment is simulated and modelled as subsequent rotations in the yaw $\Delta\psi$ (around the $z$ axis) and the pitch $\Delta\theta$ (around the $y$ axis) of the Touch frame with respect to the World frame. In the nominal case, the roll rotation $\Delta\phi$ is perpendicular to the pen and does therefore not affect the drawing. To simplify the approach, it is therefore ignored entirely. The values of yaw and pitch are fixed throughout a mission and chosen randomly based on the following distributions:

$$\Delta\theta \sim \mathcal{N}(0, \sigma_r^2), \tag{5.42}$$

$$\Delta\psi \sim \mathcal{N}(0, \sigma_r^2). \tag{5.43}$$

The relation between the true Touch frame $\underrightarrow{\mathcal{F}}_T$ and the rotated equivalent $\underrightarrow{\mathcal{F}}_{T'}$ can be written through a homogeneous coordinate transform as follows:

$$\boldsymbol{T}_{WT'} = \left[\begin{array}{c|c} \mathbf{C}_y(\Delta\theta)\ \mathbf{C}_z(\Delta\psi)\ \mathbf{C}_{WT} & {}_W\mathbf{r}_T \\ \hline 0 \quad 0 \quad 0 & 1 \end{array}\right]. \tag{5.44}$$

Here, $\mathbf{C}_i(\cdot)$ denotes a 3D rotation matrix around the $i$ axis. Note that since we assume no translational misalignment, the whiteboard position ${}_W\mathbf{r}_T$ is correct. To illustrate the nature of the simulated orientation misalignment, Figure 5.8 shows the actual physical whiteboard in grey and the misaligned calibration in green.

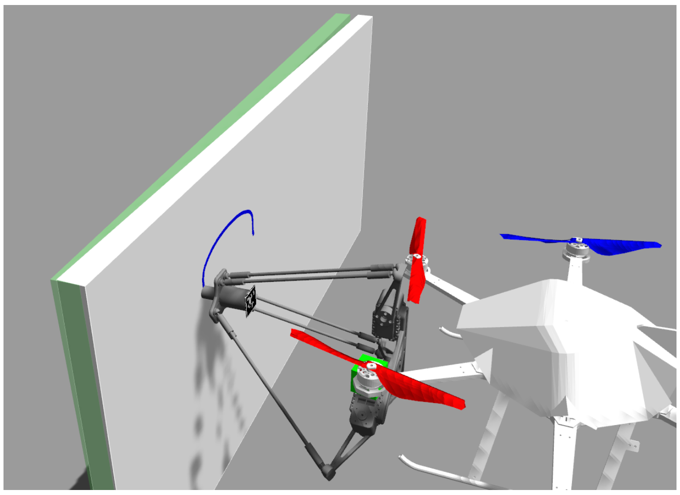


Figure 5.8: Visualisation of the whiteboard orientation misalignment with the true physical whiteboard in grey and wrong calibration in green.

## 5.3.2 Misalignment Estimation

As for the drift estimation, an Extended Kalman Filter is used to estimate the orientation misalignment of the whiteboard to arrive at the correct Touch frame. The goal is hence to estimate the pitch and yaw offsets $\Delta\theta, \Delta\psi$. Therefore the EKF state vector $\widehat{\mathbf{x}}_k$ is defined as:

$$\widehat{\mathbf{x}}_k = \begin{bmatrix} \Delta\widehat{\theta}_k \\ \Delta\widehat{\psi}_k \end{bmatrix} \tag{5.45}$$

Here, $\Delta\widehat{\theta}, \Delta\widehat{\psi}$ represent the estimated pitch and yaw of the Touch frame misalignment with respect to the World frame. By performing the inverse rotations as done during simulation, one can define the estimated true Touch frame as:

$$\widehat{\boldsymbol{T}}_{WT} = \left[\begin{array}{c|c} \mathbf{C}_z(-\Delta\widehat{\psi})\ \mathbf{C}_y(-\Delta\widehat{\theta})\ \mathbf{C}_{WT'} & {}_W\mathbf{r}_T \\ \hline 0 \quad 0 \quad 0 & 1 \end{array}\right]. \tag{5.46}$$

### Prediction

As mentioned in Section 5.3.2, the whiteboard misalignment is modeled as a fixed rotation around pitch and yaw. Hence, the state prediction is trivial yielding the following process model:

$$\widehat{\mathbf{x}}_{k|k-1} = \widehat{\mathbf{x}}_{k-1|k-1} + \begin{bmatrix} w_{n,\theta} \\ w_{n,\psi} \end{bmatrix}, \tag{5.47}$$

$$w_{n,i} \sim \mathcal{N}(0, \sigma_n^2) \quad \text{with} \quad i \in \{\theta, \psi\}. \tag{5.48}$$

Here, the noise is added to avoid numerical instabilities arising from ever decreasing state uncertainty as the mission goes on. Hence, $\sigma_n$ is chosen very small. From the process model, the following predictions for the filter state $\widehat{\mathbf{x}}_k$ and its associated covariance matrix $\mathbf{P_k}$ are derived:

$$\widehat{\mathbf{x}}_{k|k-1} = \widehat{\mathbf{x}}_{k-1|k-1}, \tag{5.49}$$

$$\mathbf{P}_{k|k-1} = \mathbf{P}_{k-1|k-1} + \mathbf{Q}_n, \tag{5.50}$$

$$\mathbf{Q}_n = \operatorname{diag}(\sigma_n^2). \tag{5.51}$$

The matrix $\mathbf{Q}_n$ captures the noise added for numerical stability. The Jacobian of the process model is identity and can thus be dropped from the equations.

**Measurements**

In contrast to the drift estimation, the whiteboard orientation estimation is just informed by the tactile measurements. This is for two reasons: first, both the pitch and yaw are fully observable only given the tactile feedback. Second, the whiteboard misalignment is a constant offset which does not change gradually like the drift. The visual feedback is more suited for slow changes as it relies on the camera and template images being similar to each other such that the template matching does not fail. The whiteboard misalignment is an abrupt and fixed error potentially leading to strongly distorted drawings which could not be matched to the undistorted template image. Hence, the measurement $\widetilde{z}_k^t$ just includes the tactile measurement $\widetilde{d}$ defined before in Equation 5.12. The associated tactile measurement model is based on the same contact condition (see Equation 5.35) but now assumes the MAV position to be fixed and the Touch frame estimate $\widehat{\boldsymbol{T}}_{WT}$ to be the varying parameter. More specifically, the pen contact point is transformed to the Touch frame as follows:

$$_T\mathbf{r}_P = \widehat{\boldsymbol{T}}_{TW}\, \boldsymbol{T}_{WB}\, \boldsymbol{T}_{BA}\, \boldsymbol{T}_{AE}\; _E\mathbf{r}_P. \tag{5.52}$$

As before, solving this condition for the expected tactile measurement $\widehat{d}$ yields the tactile measurement model. This is done symbolically in MATLAB and the actual expression is again omitted in the interest of space and since the term does not offer any more insights. In contrast to the drift estimation, the expected tactile measurement now depends on the EKF states being the pitch and yaw estimates $\widehat{\Delta\theta}, \widehat{\Delta\psi}$. The measurement step for an incoming tactile measurement can now be written in the standard EKF form as done before in Equations 5.36 to 5.39. Again, the tactile Jacobian $\mathbf{H}_k^t$ is derived symbolically using MATLAB but is now defined as follows:

$$\mathbf{H}_k^t(\mathbf{x}) = \begin{bmatrix} \partial d/\partial x_1 & \partial d/\partial x_2 \end{bmatrix}. \tag{5.53}$$

## 5.4 Relative End Effector Position Control

In the following, the simulation of the mismatches in the delta arm model is explained which cause an offset in the end effector positioning. Then, the End Effector Position Estimator and the closely connected Relative End Effector Controller are introduced.

### 5.4.1 End Effector Offset Simulation

The key assumption for this third error source is a mismatch between the delta arm model used by the NMPC method and the actual physical delta arm. This could be wrongly measured components dimensions or unknown offsets in the achieved servo angles. Such mismatches then lead to position offsets in the end effector control. Furthermore, since the pre-existing NMPC method assumes a zero-order dynamics model for the end effector, the true end effector position always lags behind its control input as the true delta arm does not have an instantaneous dynamic response. Two different approaches to simulating the root-cause of the end effector offsets are taken:

***Servo Offset*** This approach assumes that the servos do not achieve the exact angles that they are commanded to due to wrong calibration of the zero angle and other inaccuracies. This is modelled as additive and multiplicative offsets on the servo angles:

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + \Delta\boldsymbol{\theta}, \tag{5.54}$$

$$\Delta\theta_i = \alpha_i\theta_i + \beta_i, \tag{5.55}$$

$$\alpha_i, \beta_i \sim \mathcal{N}(0, \sigma_s^2) \quad \text{with} \quad i \in \{1, 2, 3\}. \tag{5.56}$$

Here, $\boldsymbol{\theta}'$ are the three servo angles which are achieved by the servos and $\boldsymbol{\theta}$ the angle commands sent by the NMPC. The angle offsets are captured in $\Delta\boldsymbol{\theta}$ which depend linearly on the current angle commands $\boldsymbol{\theta}$ as shown in Equation 5.55. Here, $\boldsymbol{\alpha}$ denotes the multiplicative per-servo factor and $\boldsymbol{\beta}$ an additive per-servo offset.

***Length Offset*** Here, the NMPC model of the push rod length $l$ (please refer to Figure 3.2) is assumed to be off by a fixed but random length error:

$$l' = l + \Delta l, \tag{5.57}$$

$$\Delta l \sim \mathcal{N}(0, \sigma_l^2). \tag{5.58}$$

Here, $l'$ is the true value of the physical delta arm and $l$ is the wrong length used by the NMPC. The length mismatch is captured in $\Delta l$. The same offset is added to all three push rods according to the parallel manipulator condition which ensures that the base plate and the end effector remain parallel.

Both these errors in the NMPC model have the same effect on the end effector position control of the pre-existing NMPC which is fully model-based: there will be a non-zero distance between the model-based end effector command, ${}_A\mathbf{r}_E^c$, and its actually achieved position, ${}_A\mathbf{r}_E$.

### 5.4.2 End Effector Offset Estimation

In order to remove the offset between the commanded and achieved end effector position, the latter has to be known to allow any reasoning about the current offset. Given that the delta arm model used by the NMPC is wrong, the achieved end effector position can no longer be recovered based on the forward kinematics model since it includes these errors. Instead, it is estimated based on visual observations of an AprilTag attached to the back of the pen case. Figure 5.9 shows the AprilTag in relation to the delta arm and the camera which was already used for the visual feedback to estimate MAV position drift.

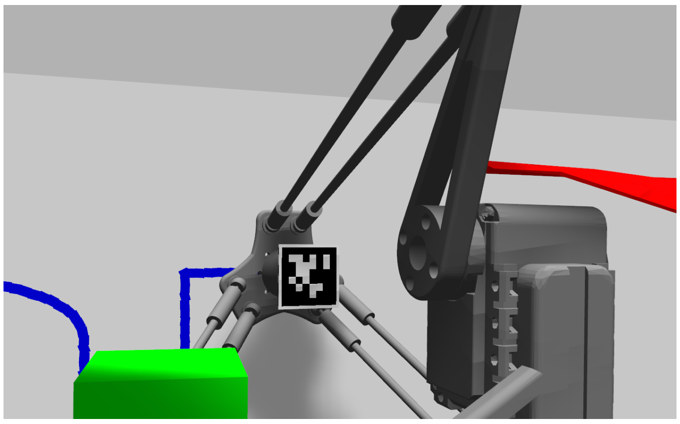


Figure 5.9: Visualisation of the AprilTag attached to the back of the end effector, the delta arm and the camera. The AprilTag is used to estimate the true end effector position in case of delta arm model mismatches.

The AprilTag is detected using an off-the-shelf ROS-wrapper [54] of the popular AprilTag3 framework [55]. It also provides estimated 3D pose measurements of the AprilTag, which we denote as ${}_A\widetilde{\mathbf{r}}_E$. In order to allow robust and yet aggressive tracking of the end effector position, these pose measurements are fused in a Kalman filter resulting in the end effector position estimate ${}_A\widehat{\mathbf{r}}_E$. The state vector $\widehat{\mathbf{x}}_k$ of this Kalman filter is defined as follows:

$$\widehat{\mathbf{x}}_k = \begin{bmatrix} \widehat{x}_k \\ \widehat{y}_k \\ \widehat{z}_k \end{bmatrix}. \tag{5.59}$$

Here, $\widehat{x}, \widehat{y}, \widehat{z}$ are the estimated end effector positions along all axes in the End Effector frame. In the following, the process and measurement models are described as well as the respective KF updates.

#### Prediction

Based on the the end effector commands sent by the NMPC and assuming somewhat continuous motion of the end effector, a guess about the true end effector motion can be made yielding the process model in Equation 5.60 stating

$$\widehat{\mathbf{x}}_{k|k-1} = \widehat{\mathbf{x}}_{k-1|k-1} + \Delta\mathbf{r}_{E,k} + \begin{bmatrix} w_{e,x} \\ w_{e,y} \\ w_{e,z} \end{bmatrix}. \tag{5.60}$$

Here, $\varDelta \mathbf{r}_E$ denotes the difference between the last model-based end effector command and the current one. This command change is computed as described Equation 5.61 while $w_{e,i}$ denotes additive white noise which is added on each axis:

$$\varDelta \mathbf{r}_{E,k} = {}_A\mathbf{r}^c_{E,k} - {}_A\mathbf{r}^c_{E,k-1}, \tag{5.61}$$

$$w_{e,i} \sim \mathcal{N}(0, \sigma^2_{e,i}) \quad \text{with} \quad i \in \{x, y, z\}. \tag{5.62}$$

**Measurements**

The raw AprilTag measurements, ${}_C\widetilde{\mathbf{r}}_T$, are with respect to the Camera frame and measure the tag's position, not the end effector. Hence, the measurements are transformed to the Arm frame and translated from the back of the pen case to the end effector in Equation 5.63 where $c$ denotes the pen case length:

$${}_A\widetilde{\mathbf{r}}_E = \begin{bmatrix} 0 & 0 & c \end{bmatrix}^T + \boldsymbol{T}_{AB}\, \boldsymbol{T}_{BC}\, {}_C\widetilde{\mathbf{r}}_T. \tag{5.63}$$

This defines the actual measurement used for the Kalman filter as seen in Equation 5.64 stating

$$\widetilde{\mathbf{z}}_k = {}_A\widetilde{\mathbf{r}}_E = \begin{bmatrix} \widetilde{x}_k \\ \widetilde{y}_k \\ \widetilde{z}_k \end{bmatrix}. \tag{5.64}$$

In order to speed up the detection and pose estimation of the AprilTag, the current end effector estimate is used to crop out a region-of-interest in the camera image at the expected AprilTag position. The patch has the expected size of the AprilTag perceived by the camera plus a margin to ensure robustness. The rest of the image is blacked out. In contrast to the other two Kalman filters presented in this thesis, the measurement model for the end effector position estimation is trivial as it directly measures what it estimates, apart from some constant transformations. Therefore, the measurement is fused into the filter as follows:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\Big(\mathbf{P}_{k|k-1} + \mathbf{R}_k\Big)^{-1}, \tag{5.65}$$

$$\mathbf{y}_k = \widetilde{\mathbf{z}}_k - \widehat{\mathbf{x}}_{k|k-1}, \tag{5.66}$$

$$\widehat{\mathbf{x}}_{k|k} = \widehat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \mathbf{y}_k, \tag{5.67}$$

$$\mathbf{P}_{k|k} = \Big(\mathbf{I} - \mathbf{K}_k\Big)\mathbf{P}_{k|k-1}. \tag{5.68}$$

Here, $\mathbf{K}_k$ is the Kalman gain and $\mathbf{y}_k$ denotes the KF innovation. The measurement noise $\mathbf{R}_k = \text{diag}(\sigma_m, \sigma_m, 2\sigma_m)$ is set based on experiments in the simulator comparing the ground-truth tag position with the measurement. As to be expected, the measurements parallel to the camera image plane are observed to be quite accurate while the depth estimate is more noisy. Hence, the measurement noise along the depth is set to be higher.

### 5.4.3 End Effector Position Control

The pre-existing NMPC controls the end effector position through the inverse kinematics model, hence uses angle setpoints as inputs for the servos. When looking at the control diagram in Figure 5.10, it can be observed that the model-based inverse kinematics act as a feed-forward term providing a slightly offset set of angle commands $\boldsymbol{\theta}^{ff}$. The connected delta arm can be viewed as the process plant which takes the three angles $\boldsymbol{\theta}$ as input and positions the the servos accordingly, thus positioning the end effector at ${}_A\mathbf{r}_E$. The KF described in the previous section then estimates its position based on the AprilTag, yielding ${}_A\widehat{\mathbf{r}}_E$. Using the difference between the model-based end effector position command and the estimate of its actual position, one can define a position error for the end effector given in Equation 5.69 which states

$$ {}_A\mathbf{e}_E = {}_A\mathbf{r}^c_E - {}_A\widehat{\mathbf{r}}_E. \tag{5.69} $$

However, to include this position error into the servo signal it must first be transformed from position to angle space by applying the delta arm Jacobian $\mathbf{J}^a$ as shown in Equation 5.70:

$$ \mathbf{e}_\theta = \mathbf{J}^a {}_A\mathbf{e}_E. \tag{5.70} $$

The Jacobian $\mathbf{J}^a$ is computed numerically using central finite differences on the inverse kinematics which map end effector positions to servo angles based on the NMPC model of the delta arm.

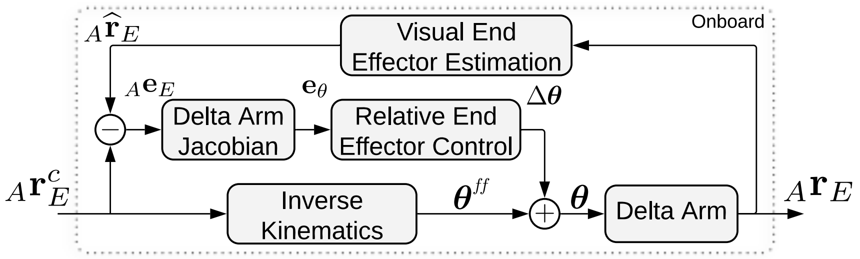


Figure 5.10: Control diagram showing the Relative End Effector Position Control as a cascaded controller around the pre-existing NMPC delta arm control.

The component which still has to be defined, is the Relative End Effector Control block. It should take the previously defined error term in angle space, $\mathbf{e}_\theta$ as input and output an appropriate set of angle corrections $\Delta\boldsymbol{\theta}$ to be added onto the imperfect model-based angle commands $\boldsymbol{\theta}$. In the following, an appropriate control scheme is derived by oversimplifying the control loop. The following three assumptions are made here:

1. The delta arm reacts perfectly to any changes in its input $\boldsymbol{\theta}$, meaning that it adjusts the servo angles accordingly. In other words, as soon as its input changes, the output follows.

2. The Visual End Effector Estimation is perfect, i.e. it follows the true end effector position exactly.

3. The closed-loop behavior around a fixed setpoint given by the feed-forward term is equivalent to the case when the feed-forward term changes.

Under these assumptions, the control diagram can be simplified in a sense that the ‘Delta Arm’ block is an open-loop integrator (Assumption 1) while the ‘Visual End Effector Estimation’ and ‘Inverse Kinematics’ block can be ignored entirely

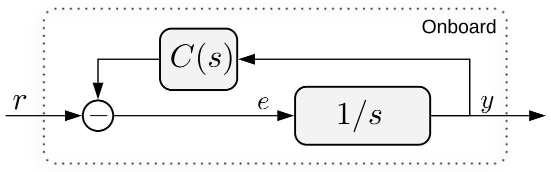


Figure 5.11: Simplified control diagram illustrating the implications by the assumptions made to derive an appropriate control scheme.

(Assumption 2 and 3, respectively). The resulting control diagram is shown in Figure 5.11. This diagram is only meant to illustrate the reasoning taken here and does not represent the real system.
From Figure 5.11, the closed-loop transfer function is worked out and is given in Equation 5.71 which reads as

$$H(s) = \frac{1}{s + C(s)}. \tag{5.71}$$

It can be concluded that given the simplifications made, the closed-loop system is stable for a simple P-controller, i.e. setting $C(s) = K_p$. Importantly, no additional integrator is required to track the reference in closed-loop operation.
However, the assumptions made oversimplify the real system. When running the feedback loop with a pure proportional controller, the end effector oscillates due to delays in the delta arm, the visual estimation and other effects. In order to get rid of these oscillations, a derivative control term is added which provides damping. The final control law is given in Equation 5.72 with $\Delta\mathbf{e}_\theta$ denoting the error change in angle space as defined in Equation 5.73:

$$\Delta\boldsymbol{\theta} = K_p\mathbf{e}_\theta + K_d\Delta\mathbf{e}_\theta, \tag{5.72}$$

$$\Delta\mathbf{e}_\theta = \frac{1}{\Delta t}\left(\mathbf{e}_\theta^k - \mathbf{e}_\theta^{k-1}\right). \tag{5.73}$$

To prevent the derivative component from amplifying high frequency noise, the error change signal $\Delta\mathbf{e}_\theta$ is low-pass filtered before being fed into the PD controller. This is treated as an implementation detail and therefore omitted in the formulas above.
It should be noted that the Relative End Effector Position Control scheme described here is a purely local feedback component for the delta arm. In contrast to the correction mechanism for the MAV Position Drift and the Whiteboard Orientation, this feedback loop only includes estimates directly connected to the MAV Body frame but not its position with respect to the World and Touch frame. It can therefore be implemented as a cascaded controller around the existing forward kinematics delta arm control of the NMPC.

# Chapter 6

# Results

## 6.1 Appearance-based Precision Evaluation

When looking at existing literature that deals with similar tasks like Aerial Writing (please refer to Section 2 for more detail on these related works), there seems to be a lack of a standardised evaluation metric. To cope with this, an appearance-based precision evaluation is proposed here to be used as the new standard metric for such tasks. Together with the extensive experimental evaluation (see Section 6.2) of the pre-existing NMPC platform, this appearance-based precision metric marks the third contribution presented in this Master thesis according to Section 1.2.

The metric consists of a per-trajectory-point distance error between the completed actual drawing and a rendering of the full reference trajectory similar to reference drawing used in the visual feedback and shown in Figure 5.3. The key advantage of this visual error is that it does not depend on any noise or biases in the pose measurements stemming from the external motion capture used in real-world experiments presented in Section 6.2. During these, inaccuracies in the pose measurements were observed either caused by bad calibration, poor object visibility, or marker reflections on the whiteboard surface.

In the following, the computation of the error is explained in more detail: after the Aerial Writing task is completed, a photo of the final drawing is taken, undistorted and the whiteboard is cropped out. The image is then filtered in HSV space (similar as done for the camera image used in the visual drift measurement). Next, the full reference trajectory is rendered on a black background with the same whiteboard dimensions and line thickness like in the experiment. Then, both images are then 'thinned', meaning that the thick lines get collapsed to one pixel wide lines along the middle of the stripes. This is done using the `thin`[1] method in the `morphology` package of `skimage` and should avoid an underestimation of the error due to the non-zero thickness of the drawn lines. Finally, the two point clouds representing the actual and intended drawings are registered by running a two-dimensional Iterative Closest Point (ICP) algorithm [56, 57]. A Python implementation[2] of ICP is used which iteratively minimises the following point-to-point error:

$$E_{ICP} = \sum_{i \in \mathbf{P}^a, \mathbf{P}^r} \left( \mathbf{R}\,\mathbf{p}_i^a + \mathbf{t} - \mathbf{p}_i^r \right)^2. \tag{6.1}$$

Here, $\mathbf{P}^a, \mathbf{P}^r$ denote the point clouds of the actual and reference drawings, respectively, and $\mathbf{p}_i^a, \mathbf{p}_i^r$ the corresponding nearest neighbour points in each point cloud.

[1]Source: https://scikit-image.org/docs/dev/api/skimage.morphology.html

[2]https://github.com/ClayFlannigan/icp

Minimisation of this error term results in an set rotation matrix **R** and translation vector **t** to register the two point clouds as well as the nearest neighbour distances for all points. The final nearest neighbour distance is used to evaluate the accuracy on a per-trajectory-point level. It should be noted here that this error ignores constant offsets that are globally consistent for the whole image since the ICP algorithm shifts and rotates the point clouds such that the total nearest neighbour distance between all points is minimised. Hence, the appearance-based error metric focuses on inconsistencies within the written trajectory, but not its global position with respect to the template. It should further be noted that using a point-to-plane error metric instead would lead to faster convergence. However, this is not crucial for the offline error analysis and was hence omitted in the interest of a simpler implementation. Finally, the visual error is plotted by showing the point distance as colour-coded dots on top of the actual drawing in black. Further, the mean error and its 3-sigma bound are displayed. An example is given in Figure 6.1.

## 6.2 Real-world Experiments

The real-world experiments shown in the following were conducted for a submission to RSS 2020 which was accepted and should appear as [9]. The evaluation as well as the RSS paper writing were a joint effort of Dimos Tzoumanikas and the author of this thesis. The flights were done in a flight arena and do not include the feedback components presented in Chapter 5. Hence, the method being evaluated in the following section is the full work of Dimos Tzoumanikas. In fact, issues during these experiments inspired the introduction of multimodal feedback to cope with drifting odometry, misaligned whiteboard orientation and delta arm model mismatches.

### 6.2.1 Experimental Setup

Throughout the mission, external pose estimates are provided by a Vicon motion capture system. The contact surface is a $1 \times 0.5$ m whiteboard for which its pose $\boldsymbol{T}_{WT}$ is estimated based on Vicon measurements. Each experiment consists of the following different trajectory stages: (i) approach, (ii) write, and (iii) return home. The end effector is enabled for the writing trajectory and disabled for the rest, using the appropriate flags as mentioned in Section 4.3. The results analysis focuses mainly on the trajectory writing which includes contact, whereas for the other two parts (approach/return) the MAV performs simple position tracking. The system's accuracy is evaluated by comparing the reference trajectories to those estimated by the Vicon motion capture system in four experiments. In Section 6.2.2 detailed and repeatable results for two different trajectories, namely RSS and $\mathrm{E} = \mathrm{mc}^2$, are presented. Sections 6.2.3 and 6.2.4 then show consistent tracking performance across varying MAV velocities and text sizes, respectively.

### 6.2.2 Trajectory Tracking

Figure 6.1 shows the tracking of the RSS trajectory visualised in the Touch frame $\underrightarrow{\mathcal{F}}_T$ for the end effector and the MAV. In Figure 6.2, both the pen position *tracking* error $\boldsymbol{e}^p$ and the spring force tracking, i.e. $F$ and its reference $F^r$ are shown. The maximum reference velocity was set to 7.5 $\mathrm{cm\,s^{-1}}$ and the maximum acceleration to 2.5 $\mathrm{cm\,s^{-2}}$. The trajectory consists of four contact segments with a combined duration of 65 s. Based on the Vicon estimates, the tracking error of the end effector remains in the $[-10, 10]$ mm range during the contact segments while the MAV position error is within the $[-40, 40]$ mm range. This highlights the efficacy of using a manipulator with faster dynamics than the MAV's for precision tasks such as Aerial Writing. Similarly to the above, Figures 6.3 and 6.4 show the trajectory tracking for the more challenging $E = mc^2$ experiment which contains ten contact segments with a combined duration of 63 s. Tracking accuracy is similar as before with the end effector and MAV tracking error in the $[-10, 10]$ and $[-50, 50]$ mm range. The accuracy can be visually verified since the overlapping segments of the 'R' and 'm' coincide almost perfectly. Additionally, the consistent approaching and retracting from the contact surface leads to identical starting points of individual letter segments, e.g. the three horizontal lines of the letter 'E'. In both cases, the maximum error based on the visual error analysis is 10 mm mostly originating from temporary loss of contact. Possible reasons for this are bad estimation of the orientation part of the Touch frame transformation $\boldsymbol{T}_{WT}$, the assumption of a perfectly flat contact surface being wrong and most importantly the finite accuracy of the delta arm. The imperfect tracking along the Touch frame normal direction (shown in blue in Figures 6.2, 6.4) is also reflected in the reference force tracking.

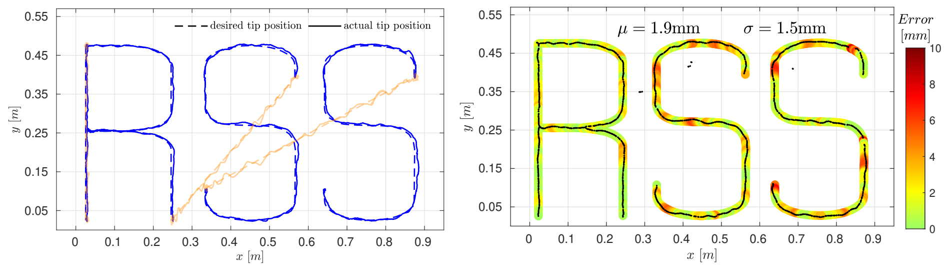


Figure 6.1: Reference and actual tip position (left) as estimated by Vicon. Blue corresponds to contact segments while orange refers to free flight. Visual error (right) between reference and actual tip position. The maximum estimated error is lower than 10 mm and is located at discontinuous segments as expected.

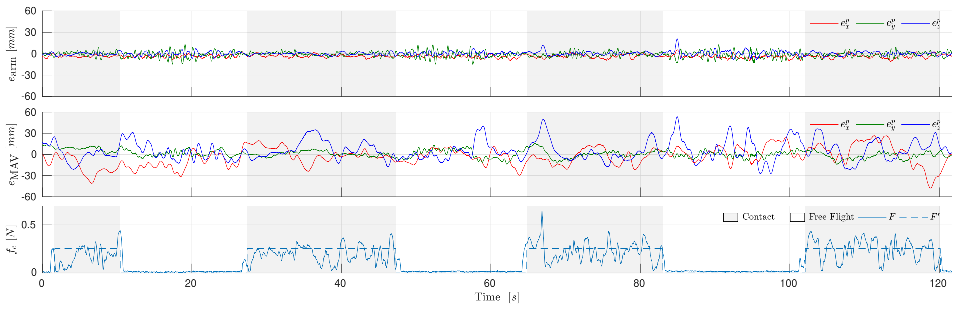


Figure 6.2: Reference tracking error of the tip position (top), MAV (middle), and measured contact force (bottom). The tracking error is plotted in the Touch frame $\underrightarrow{\mathcal{F}}_T$. The end effector accuracy is significantly greater than that of the MAV, given that they remain in the $[-10, 10]$ mm and $[-40, 40]$ mm ranges, respectively.

## Repeatability

In order to prove the repeatability of the approach, each experiment was conducted thrice. Relevant tracking statistics for the MAV and arm are given separately in Figure 6.5, in which the textured box plots correspond to MAV data and the plain ones to that of the end effector. The median and upper values for the end effector are significantly lower than the ones for the MAV, further showing the need of an aerial manipulator for precise tasks including contact. The MAV tracking accuracy

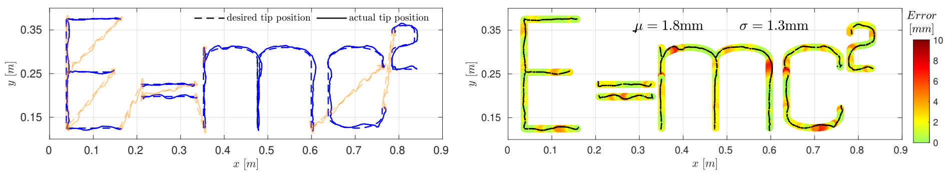


Figure 6.3: Reference and actual tip position (left) as estimated by Vicon. Blue corresponds to contact segments while orange refers to free flight. Visual error (right) between reference and actual tip position. Similarly as in the RSS experiment shown in Figure 6.2, maximum error does not exceed 1 cm.

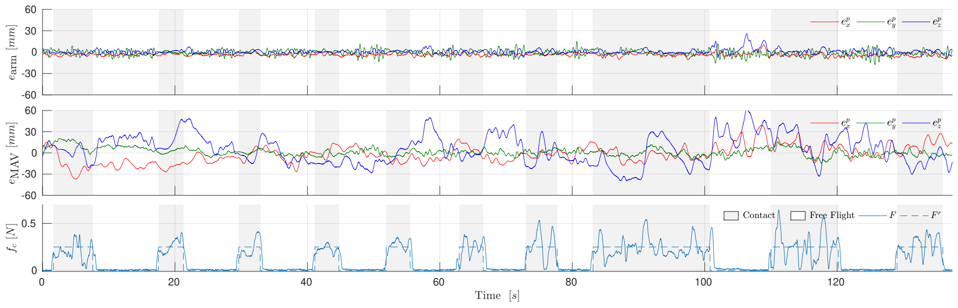


Figure 6.4: Reference tracking error of the tip position (top), MAV (middle) and measured contact force (bottom). The tracking accuracy of the end effector is significantly greater than that of the MAV, given that they remain in the $[-10, 10]$ mm and $[-50, 50]$ mm ranges, respectively

along the $z$ axis was the lowest amongst all axes, as this was most affected by the interaction forces and unmodelled torque disturbances due to servo motion.

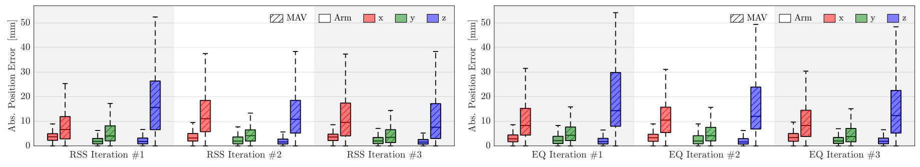


Figure 6.5: MAV and end effector box plots of the contact segments for 3 iterations of the RSS trajectory experiment (left) and the more challenging E = mc$^2$ trajectory experiment (right).

### 6.2.3 Velocity Sweep

The aim of the next experiment is to demonstrate the effects of the input velocity and acceleration on the writing accuracy. Five iterations of the same `Hello` trajectory experiment were performed while using different velocity and acceleration profiles with the following maximum velocities and accelerations:

$$v_{max} \in \{7.5, 12.5, 17.5, 22.5, 27.5\} \text{ cm s}^{-1},$$
$$a_{max} \in \{3.75, 6.25, 8.75, 11.25, 13.75\} \text{ cm s}^{-2}.$$

Figure 6.6 shows the box plots for the MAV and end effector tracking accuracy based on the Vicon measurements. The plots show that consistent tracking results

are obtained in all the different velocity and acceleration settings tested. The numeric values of the tracking error are similar to the ones previously presented with the end effector achieving sub centimetre accuracy (per axis) while the MAV error is consistently less than 50 mm. However, by observing the visual error, one can verify that as the reference velocity increases, the system struggles more with the trajectory segments containing curvature e.g. ‘e’ and ‘o’. In contrast, the performance on the straight line segments remains similar.
The tracking error of the MAV could be further reduced if the NMPC was given dynamically feasible trajectories not only for position and velocity but also acceleration, jerk, and snap. Regarding the end effector tracking error, it is generally expected to increase for reference velocities beyond the ones tested here. This is because the pre-existing control model assumes that the position of the end effector can be controlled with immediate response which is not the case for a real system.

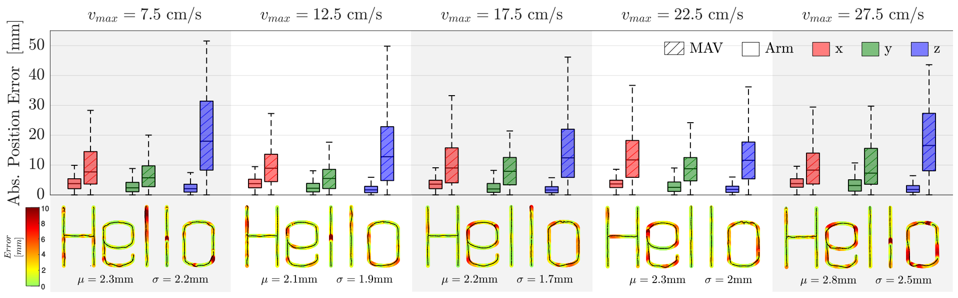


Figure 6.6: MAV and end effector box plots (top) and visual errors (bottom) for 5 iterations of the `Hello` trajectory. Different iterations correspond to different velocity and acceleration profiles.

### 6.2.4 Text Size Sweep

In Figure 6.7 shows the visual error for the same trajectory in four different text sizes ranging from 10 to 40 cm. The consistent accuracy observed shows that the system can handle the fast direction changes imposed by the small scale.

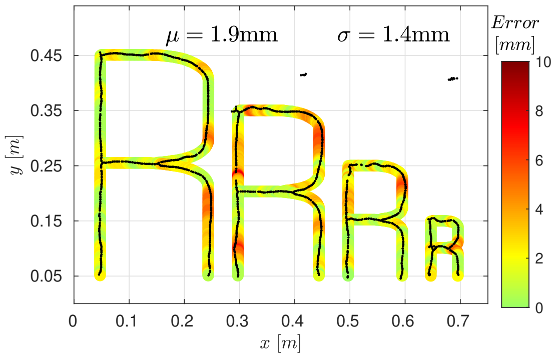


Figure 6.7: Visual error plot showing consistent results for varying text sizes

## 6.3 Simulations

In the following, the multimodal feedback presented in Chapter 5 is evaluated extensively using the Aerial Writing simulation presented in Section 4.2.2. After explaining the experimental setup, a baseline is set during which no errors are simulated or estimated. Then, each of the three error sources and feedback components is evaluated individually. Finally, all error sources are activated and estimated, which corresponds to a real-world Aerial Writing task outside the laboratory conditions of a flight arena as is used in the experiments shown in Section 6.2. For an overview of the scenarios described here, please refer to Table 6.1.

| **Scenario Name** | **Code** | **Simulated** | **Estimated** |
|---|---|---|---|
| Baseline | `B` | None | None |
| Drift | `D` | Drift | None |
| Corrected Drift | `CD` | Drift | Drift |
| Whiteboard Orientation | `W` | Whiteboard | None |
| Corrected Whiteboard Orien. | `CW` | Whiteboard | Whiteboard |
| Length Offset | `L` | Length Offset | None |
| Corrected Length Offset | `CL` | Length Offset | End Effector Offset |
| Servo Offset | `S` | Servo Offset | None |
| Corrected Servo Offset | `CS` | Servo Offset | End Effector Offset |
| Combined | `A` | All | None |
| Corrected Combined | `CA` | All | All |

Table 6.1: Overview of simulation scenarios. The first two columns state the scenario name and code, followed by which error source(s) are simulated and which are estimated, i.e. should be corrected for.

For each scenario, 5 iterations with randomly set drift profiles, whiteboard misalignment and delta arm model mismatches are performed. Full results are shown for one iteration only, i.e. the pen tracking error, the spring tracking error, the estimated errors and corrections as well as the actual and template drawing and the visual error between them. The repeatability of the approaches is shown in box plots of the spring and pen tracking error for all 5 iterations.

### 6.3.1 Experimental Setup

The experiments are performed in the simulation environment presented in Section 4.2.2 using the virtual MAV-arm system introduced in Section 4.2.3. The simulator runs along-side all other components on a powerful Desktop PC with Ubuntu 16.04 installed. Due to the high number of line segments used for the Aerial Writing, the simulator runs at about half the speed of real-time. However, thanks to the integration of ROS and Gazebo, the pre-existing NMPC and the multimodal feedback is still running in real-time compared to the simulated time. It was verified experimentally that the feedback pipeline alone would allow true real-time performance. The overall simulation setup with the true Touch frame $\underrightarrow{\mathcal{F}}_T$ and non-drifting World frame $\underrightarrow{\mathcal{F}}_W$ is shown in Figure 6.8. The noise parameters used in the multimodal feedback are set to the values shown in Table 6.2. They are kept constant to limit the amount of scenarios to be compared with each other. For all trajectories, the maximum reference velocity was set to 5.0 $\mathrm{cm\,s^{-1}}$ and the maximum acceleration to 5.0 $\mathrm{cm\,s^{-2}}$.

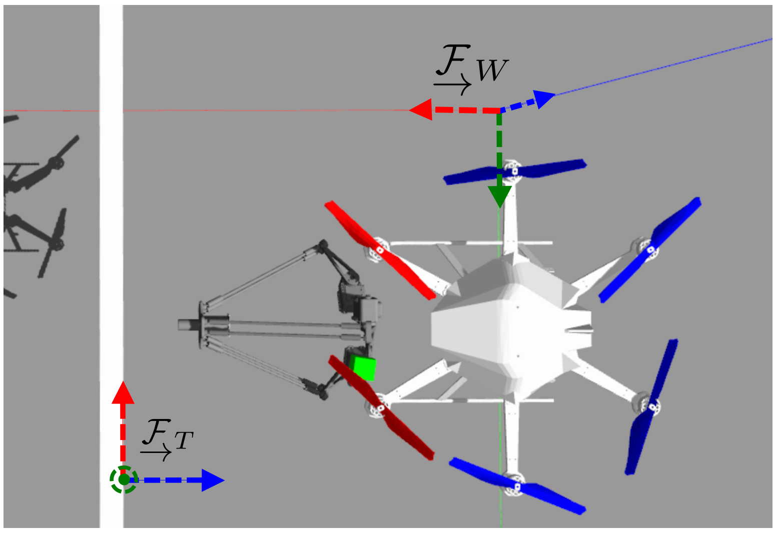

$\underset{\rightarrow}{\mathcal{F}}_W$
$\underset{\rightarrow}{\mathcal{F}}_T$

Figure 6.8: Overview of simulation setup with true Touch and World frame orientation given for reference.

| Parameter | Meaning | Value |
|---|---|---|
| $\sigma_d$ | Force Sensor noise | 1 mN |
| $\sigma_p$ | Drift Process noise | 1.5 mm/$\sqrt{\text{s}}$ |
| $\tau$ | Drift Time constant | 100 s$^{-1}$ |
| $\sigma_r$ | Whiteboard Orientation offset | 0.02 rad |
| $\sigma_n$ | Whiteboard Process noise | $3 * 10^{-5}$ rad |
| $\sigma_s$ | Servo offset | 0.1 rad |
| $\sigma_l$ | Length offset | 3 mm |
| $\sigma_e$ | End Effector Process noise | 0.1 mm |
| $\sigma_m$ | End Effector Measurement noise | 2 mm |
| $K_p$ | End Effector Control P gain | 1.0 |
| $K_d$ | End Effector Control D gain | 0.05 |

Table 6.2: Overview of numerical values assigned to noise parameters. The first two columns state the parameter name and meaning, followed by the actual value.

### 6.3.2 Baseline

Scenario `B` shows the performance of the underlying, pre-existing NMPC without any errors being simulated or feedback in place. This shows how well the algorithm works in simulation, setting a baseline to calibrate the results shown in the subsequent sections.
Figure 6.9 plots both the pen position *tracking* error $\boldsymbol{e}^p$ along all axes in the World frame and the spring displacement tracking, i.e. $s$ and its reference $s^r$. The pen tracking error remains in the one millimetre range while the spring tracking is very accurate as well. The approach phase of the MAV towards the whiteboard is cropped out in all figures, hence the plot starts at $t \approx 20$ s. The segments with grey background mark phases during which the pen was supposed to be in contact with the whiteboard. The first contact phase corresponds to the 'S', the second one to the vertical line in the 'R' after which the MAV detaches and flies back to the top again. Then the third contact phase starts corresponding to the rest of the 'R' while the fourth phase is the 'L'.
The baseline *visual* error is shown in Figure 6.10. As to be expected based on the

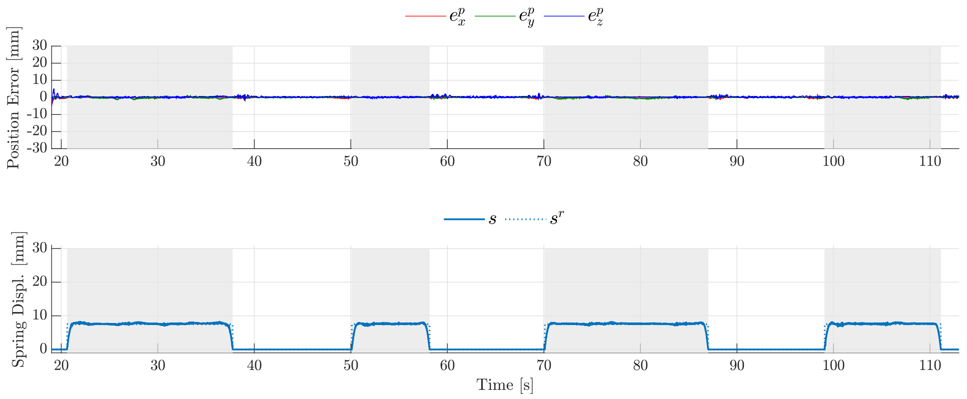


Figure 6.9: Pen tracking error (top) and Spring tracking (bottom) for scenario B.

accurate pen tracking, the visual error is very small as well. The sub-millimetre 3-sigma bound underlines the close-to-perfect writing precision. Overall, it is shown that the pre-existing NMPC method leads to extremely accurate tracking of the pen trajectory when used in the simulator. This validates the Aerial Writing simulation setup as well as the virtual MAV and delta arm replicas.

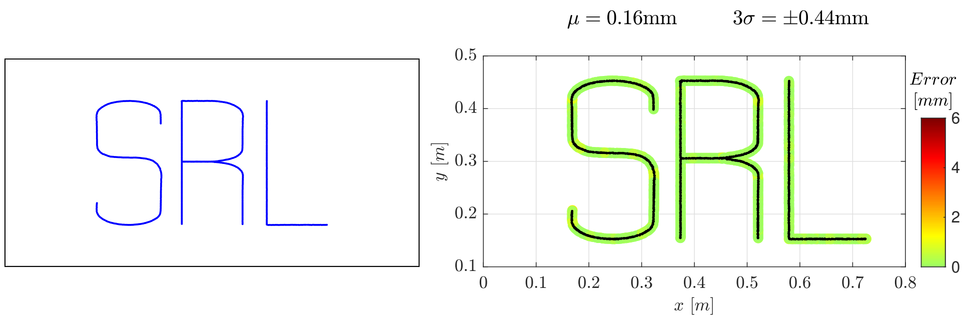


Figure 6.10: The actual drawing on the whiteboard (left) and the associated visual error for scenario B.

### 6.3.3 MAV Position Drift

Two different scenarios are evaluated here: in scenario D the MAV position drift described in Section 5.2.1 is simulated but not corrected. In scenario CD, the drift is simulated and the MAV Position Drift Estimator is turned on to correct for it.

#### Drift

As plotted in Figure 6.11, the pen tracking error in scenario D drifts along all three axes. According to the reverting-mean assumption, the MAV position drifts around the true position by up to 2 cm. This translates directly into a pen tracking error. It should be noted here that the error along $x$ is zero as long as the pen is in contact as its position is limited along this axis by the whiteboard as illustrated by Figure 6.8. Instead, the error along this axis translates to bad tracking of the spring displacement as seen in the lower plot.
The drifting MAV position has a strong effect on the visual writing quality as seen in the appearance-based error plotted in Figure 6.12. The most obvious mistake is

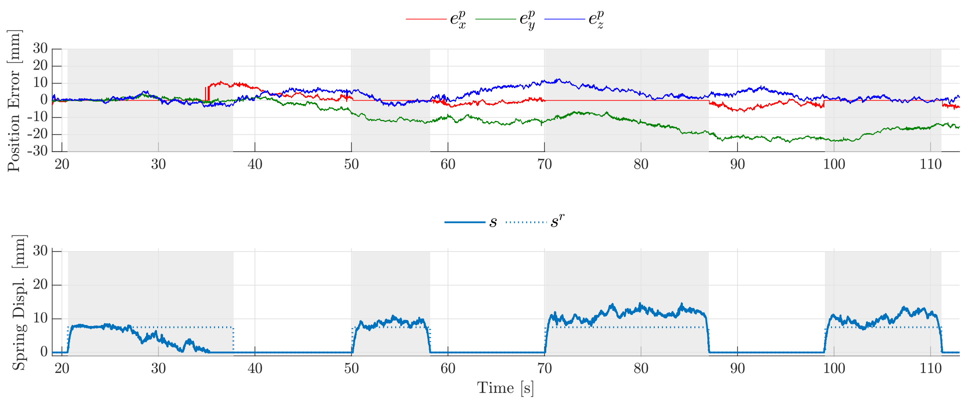


Figure 6.11: Pen tracking error (top) and Spring tracking (bottom) for scenario `D`.

the detachment during the final part of the 'S'. Due to drift along the $x$ axis, the pen loses contact with the whiteboard and stops writing. Further, the final drawing misses size and position consistency between the letters and some lines appear strongly slanted, e.g. the vertical and horizontal lines of the 'L'. The poor writing performance is also reflected in the 3-sigma bound which now indicates errors in the centimetre range. It should be noted here that the drift simulation only starts after the approach phase is finished and the pen attaches to the whiteboard. This is done to avoid constant offsets that could not be rectified by the vision-based drift estimation which is relative to the initial stroke drawn onto the whiteboard.

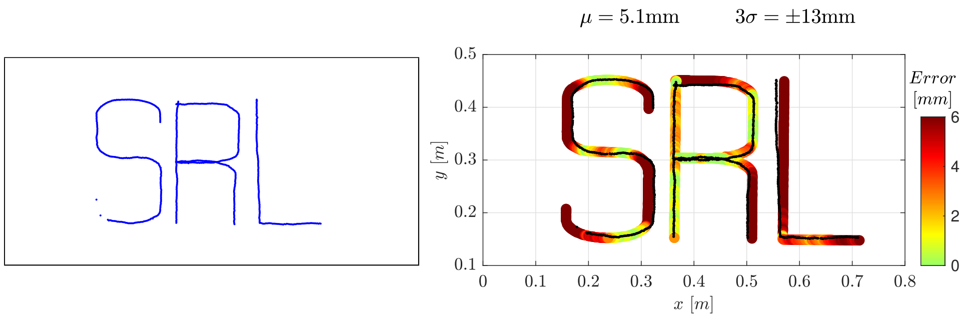


Figure 6.12: The actual drawing on the whiteboard (left) and the associated visual error for scenario `D`.

### Corrected Drift

For the scenario `CD`, the MAV Position Drift Estimator explained in Section 5.2.2 is turned on and the NMPC gets informed accordingly. In Figure 6.13, the estimated EKF states are plotted as continuous lines and the true value as dotted lines.
As long as the pen is in contact, the drift is estimated accurately. Especially the estimate along the $x$ axis, $\Delta\widehat{x}$, is tracked very closely. It coincides with the DoF along the spring and thus profits from direct and highly sensitive feedback. In contrast, the visual feedback cannot provide such accurate tracking due to discretisation errors in the camera's pixel space as well as time delays in the measurements i.e. missing image synchronisation. Therefore, the estimates $\Delta\widehat{y}$ and $\Delta\widehat{z}$ are not quite as exact but still follow the true value.
Once the pen detaches from the whiteboard, there is no more tactile measurements from the force sensor. Hence, the estimate $\Delta\widehat{x}$ no longer follows the true drift. The

other two DoFs are still tracked loosely by the visual feedback. However, as the drift along $x$ progresses, the template image, e.g. the one shown in Figure 5.4, will be rendered at the wrong scale. As the scale discrepancy between the template and actual image grows, the template matching performance deteriorates or even fails completely causing the visual feedback to stop. This effect can be seen in Figure 6.13 at $t \approx 37$ s where $\Delta\widehat{x}$ is not updated anymore and the estimate along $\Delta\widehat{y}$ starts to drift away from the true value. Another issue with the visual feedback can be seen at $t \approx 105$ s where $\Delta\widehat{y}$ and $\Delta\widehat{z}$ are no longer updated. This is caused by the small FoV of the camera which causes it to only see straight lines in one direction, the horizontal line of the letter 'L' in this case. This leads to high invariance of the template match in one direction. The EKF hence rejects the measurements as not reliable enough. In summary, the camera requires salient structures to perform the template matching based registration reliably.

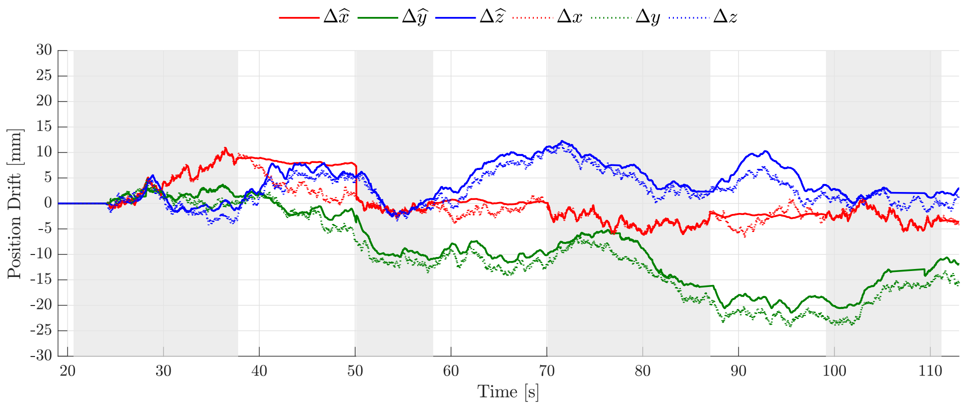


Figure 6.13: The estimated EKF states of the MAV Position Drift Estimator (continuous) and the true drift (dotted) for scenario `CD`.

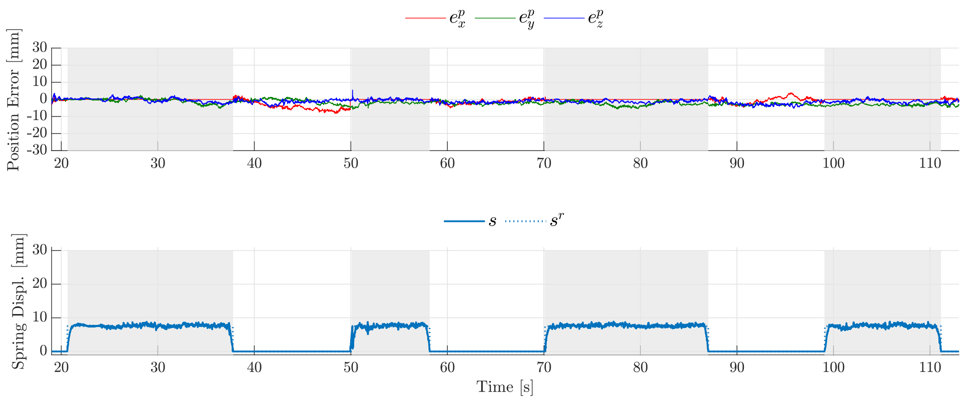


Figure 6.14: Pen tracking error (top) and Spring tracking (bottom) for scenario `CD`.

Overall, the accurate estimation of the MAV position drift allows to correct for errors as shown in Figure 6.14. For contact periods, the pen tracking error remains in the low millimetre range while the spring tracking is close to perfect. The massive improvement is further reflected in the visual outcome of the Aerial Writing shown in Figure 6.15. The visual error is consistently very low, aside some minor inaccuracies caused by delayed drift estimation, e.g. the last bit of the 'S'. The 3-sigma bound is brought down to the low millimetre range.

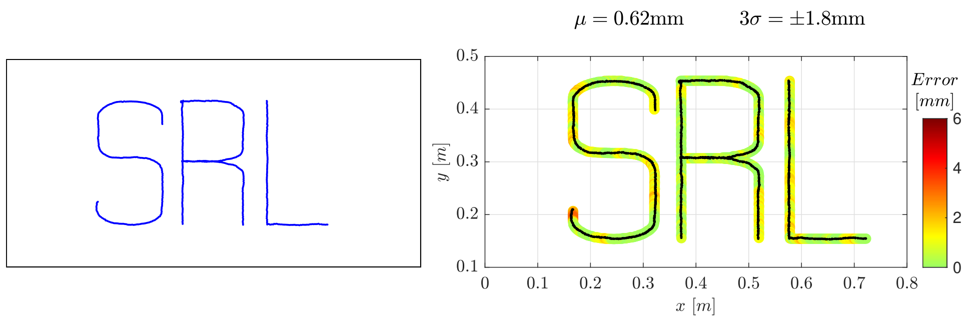


Figure 6.15: The actual drawing on the whiteboard (left) and the associated visual error for scenario `CD`.

## Repeatability

To show that the drift estimation works for random drift profiles, scenario `D` and `CD` are repeated 5 times. For each iteration, the two scenarios use the same simulated drift but it is varied across iterations. Figure 6.16 depicts the resulting box plot of the spring and pen tracking error along all axes with no feedback on the left and the drift estimation enabled on the right. It can be seen that the drift along $x$ is tracked particularly tightly, since the affected error $e^s$ is very close to zero. The other two axes are still tracked well, but not quite as tightly due to the less immediate visual feedback. Nevertheless, the error terms $e^p_y$ and $e^p_z$ are reduced from centimetre down to low millimetre range. One can see that $e^p_y$ is not reduced quite as much. This is caused by significant occlusions stemming from the push rods when first starting to write the letter 'S'. This causes a small offset along the direction of motion, $y$ in this case, which cannot be rectified afterwards. This can also be seen in Figure 6.13 where $\varDelta\widehat{y}$ is always slightly above the true value.

It should be noted here again that the error $e^x$ is zero as long as the pen touches the whiteboard surface as it limits the pen position along the $x$ axis as illustrated in Figure 6.8. Instead, the error along this axis translates to bad tracking of the spring displacement.

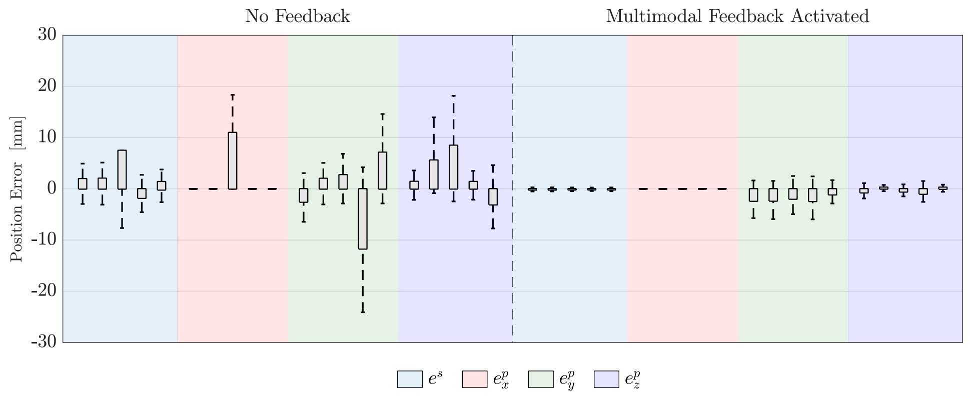


Figure 6.16: Box plot of spring and pen tracking errors for 5 iterations for scenario `D` (left) and `CD` (right).

### 6.3.4 Whiteboard Orientation Misalignment

In scenario `W` the Whiteboard orientation misalignment described in Section 5.3.1 is simulated but not corrected. In scenario `CW` the Whiteboard Orientation Estimator is turned on and informs the NMPC accordingly.

#### Whiteboard Orientation

Locally, the whiteboard found in the physical world is either further away or closer to the pen than expected, thus pushing back less or more. In contrast to the drift which affects all axes equally, the whiteboard misalignment therefore mostly affects the $x$ axis, i.e. the spring displacement. As shown on top of Figure 6.17, the pen tracking errors $e_y^p$ and $e_z^p$ are close to zero. Since the whiteboard limits the pen position along the $x$ axis by pushing back on it, the error $e_x^p$ appears to be zero as well. However, when looking at the bottom of Figure 6.17 it becomes clear that this error is simply absorbed by the spring getting displaced. Since the whiteboard is not translated but rotated, the effect of pushing back more or less should vary as the pen moves across the whiteboard. This can be seen in Figure 6.17 where the displacement error starts high as the whiteboard is rotated to the back in the upper left corner. While moving across the surface, more and more pressure is applied.

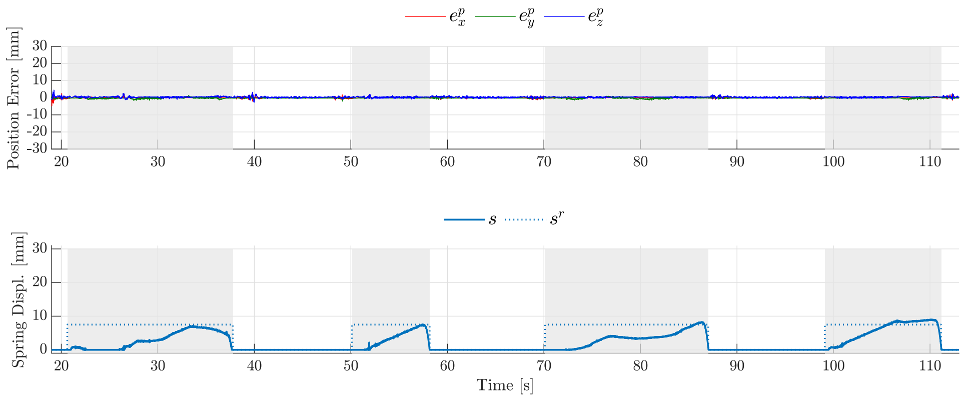


Figure 6.17: Pen tracking error (top) and Spring tracking (bottom) for scenario `W`.

The visual error of the misaligned whiteboard is depicted in Figure 6.18. The 3-sigma bound indicates sub-centimetre accuracy in terms of visual appearance. Since the whiteboard is rotated to the back in the upper left corner, the pen briefly loses contact when writing the letter 'S', translating to a large error in the visual metric.

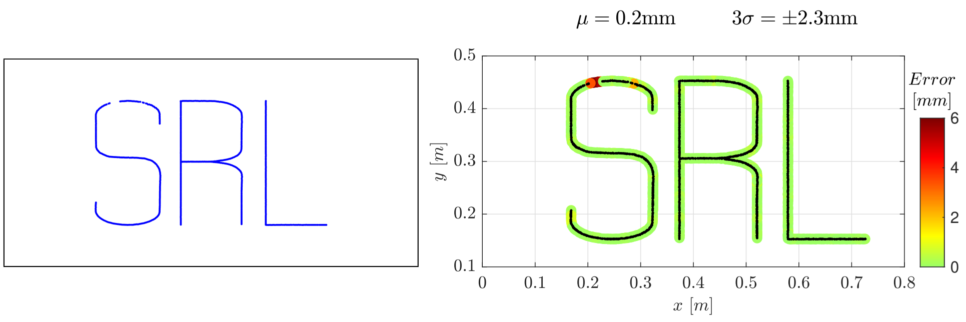


Figure 6.18: The actual drawing on the whiteboard (left) and the associated visual error for scenario `W`.

### Corrected Whiteboard Orientation

For scenario `CW`, the Whiteboard Orientation Estimator explained in Section 5.3.2 is turned on and informs the NMPC. In Figure 6.19, the estimated EKF states are plotted as continuous lines and the true values as dotted lines. As soon as the pen touches the surface, the estimates jump close to the correct value. As the pen keeps writing the first letter 'S', the estimate keeps converging towards the true value. As the letter 'S' contains both downward and sideways motion, both the pitch estimate $\Delta\widehat{\theta}$ and yaw estimate $\Delta\widehat{\psi}$ converge at the same speed. Since the whiteboard orientation is tracked well, the spring and pen tracking is precise. As expected, the effect on the errors $e_y^p$ and $e_z^p$ remains close to zero. When looking at the bottom of Figure 6.20, the reference spring displacement is now followed closely. There is no more loss of contact and also no strong pushing of the pen against the whiteboard. On the real platform, both an interrupted drawing or disassembly of the delta arm would be avoided. In line with these results, the visual error shown in Figure 6.21 is low. Inaccuracies are removed, e.g. seen in the upper left-hand corner of the letter 'S' and the 3-sigma bound is reduced significantly.

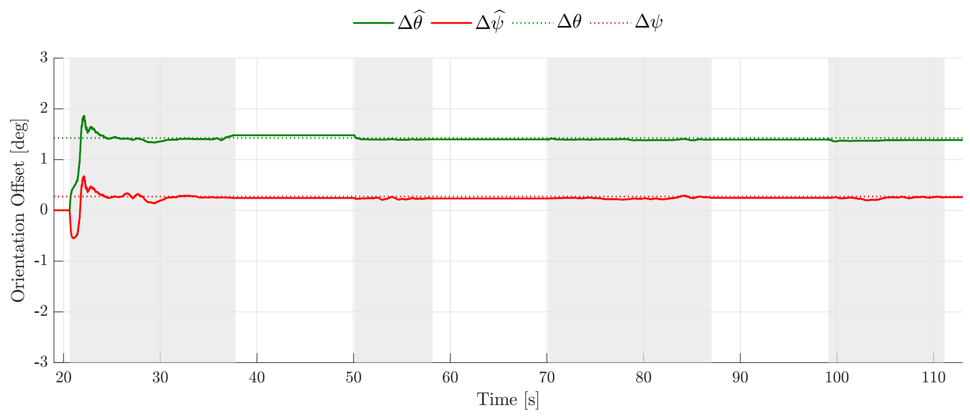


Figure 6.19: The estimated EKF states of the Whiteboard Orientation Estimator (continuous) and the true orientation misalignment (dotted) for scenario `CW`.

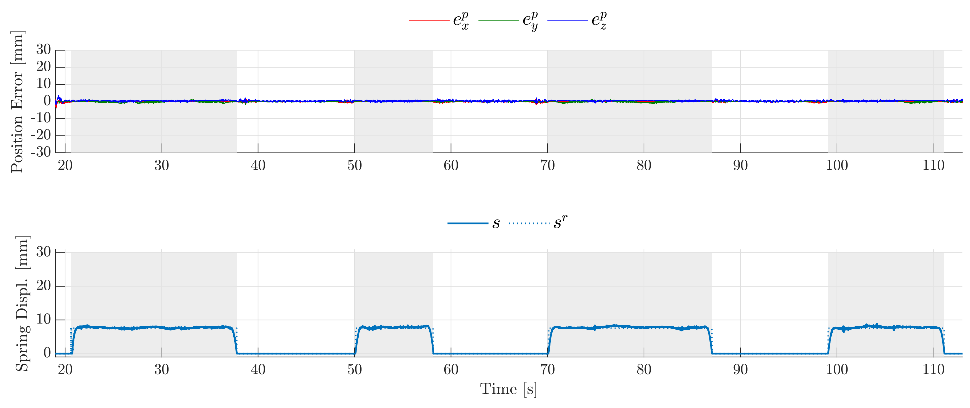


Figure 6.20: Pen tracking error (top) and Spring tracking (bottom) for scenario `CW`.

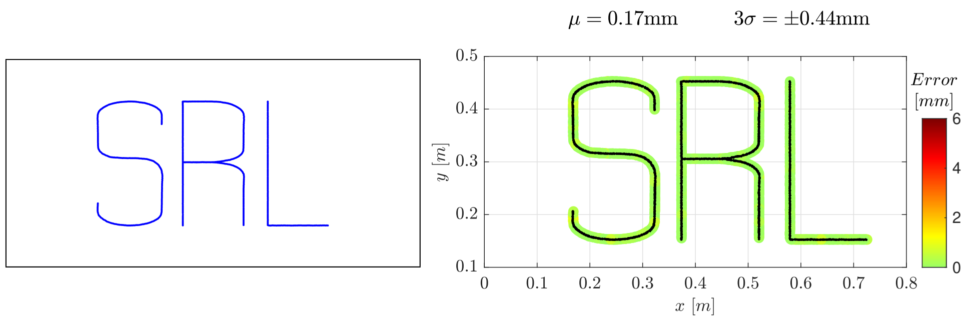


Figure 6.21: The actual drawing on the whiteboard (left) and the associated visual error for scenario CW.

## Repeatability

As for the drift estimation, the repeatability of the Whiteboard Orientation Estimator is shown by conducting the same experiment 5 times with varying pitch and yaw offsets. The resulting spring and pen tracking errors plotted in Figure 6.22 show the same as mentioned before: the misalignment of the whiteboard translates almost entirely into errors in the spring displacement. Using the tactile feedback the estimator is able to effectively remove these errors and guarantees the pen to apply the intended amount of pressure onto the whiteboard. Minor errors in the pen tracking along the $y$ and $z$ axis are reduced as well. It should be noted here that depending on the orientation of the whiteboard, very high spring displacement becomes the main issue as seen on the far left in Figure 6.22. In a real-world experiment, such high spring displacements or even a fully maxed out spring, quickly lead to the delta arm disassembling due to the high pressure on the magnetic links as there is no damping between the pen and the whiteboard anymore. When conducting the experiments shown in Section 6.2, this was the most frequent cause for system failure and decreased the reliability of the whole platform. However, this behavior is not replicated in the simulator.

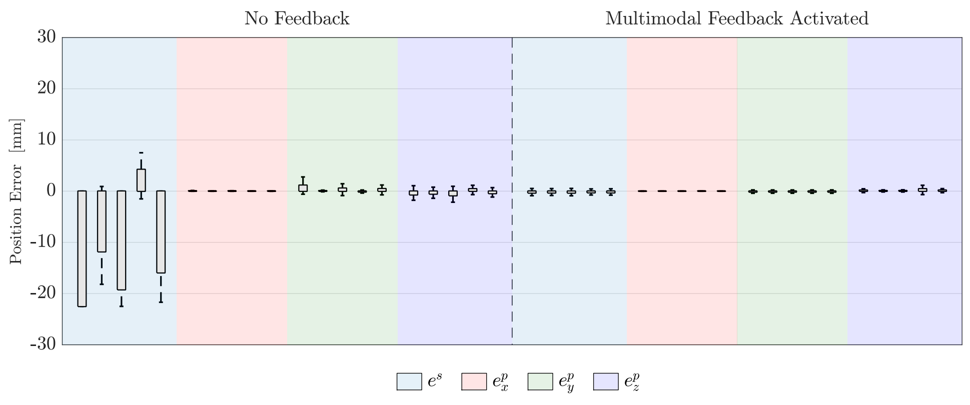


Figure 6.22: Box plot of spring and pen tracking errors for 5 iterations for scenario W (left) and CW (right).

### 6.3.5 Delta Arm Model Mismatch

In this section, the following scenarios are evaluated: in scenario `S` the servo offsets described in Section 5.4.1 are simulated but not corrected. In scenario `CS`, the offsets are simulated and the Relative End Effector Position Controller is turned on to correct for it. In addition, the mismatch between the NMPC model and the real delta arm is simulated based on length offsets for the push rods introduced in Section 5.4.1. In the interest of space, the results for these scenarios `L` and `CL` are only reported as the box plot in Figure 6.28.

#### Servo Offset

During this iteration of scenario `S`, the given servo offsets cause the end effector to be off along the $y$ and $z$ axis, but have little effect on the $x$ axis. This can be seen both in the pen and spring tracking shown in Figure 6.23. The error $e_y^p$ fluctuates around 7 mm while $e_z^p$ is in the range of 5 mm. Other servo offset configurations would have different effects on the three axes. For example, if all servos have a strong negative angle offset, i.e. the end effector is positioned too far back, the pen will never attach and thus fail to write at all. As reported before, the visual error ignores global offsets between the actual drawing and the template. This combined with the fact that the pen tracking error consists in large parts of constant offsets, causes the visual error shown in Figure 6.24 to be lower than one would expect given the poor pen tracking. One notices that horizontal lines as well as detaching and attaching cause the biggest problems for this specific set of servo offsets.

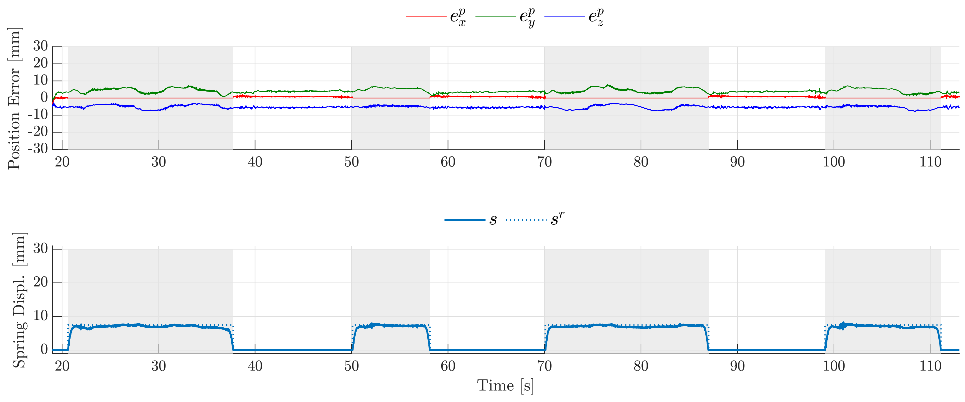


Figure 6.23: Pen tracking error (top) and Spring tracking (bottom) for scenario `S`.

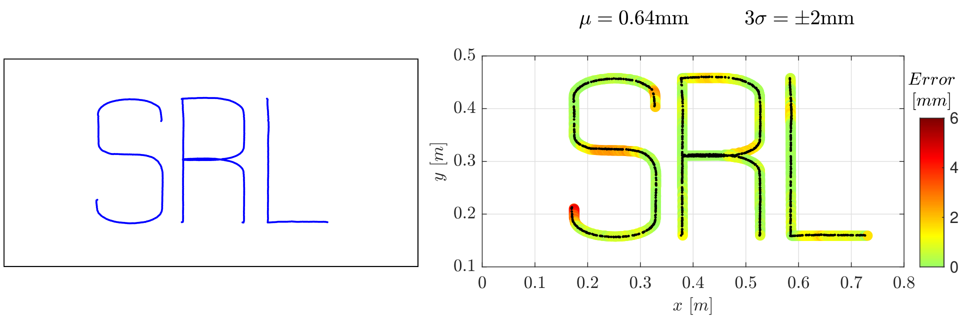


Figure 6.24: The actual drawing on the whiteboard (left) and the associated visual error for scenario `S`.

### Corrected Servo Offset

For scenario CS, the end effector position is estimated based on the AprilTag as explained in Section 5.4.2 which is used to inform the Relative End Effector Position Control explained in Section 5.4.3.

Figure 6.25 shows the estimated and hence applied angle corrections $\Delta\widehat{\theta_1}, \Delta\widehat{\theta_2}, \Delta\widehat{\theta_3}$ as continuous lines and the true value as dotted lines. All three servo angles are tracked well based on the observation of the end effector and the subsequent transformation to angle space. In some parts, small oscillations can be seen. To reduce these oscillations to an acceptable level as seen in the plot, the controller introduced in Section 5.4.3 includes derivative feedback to provide damping.

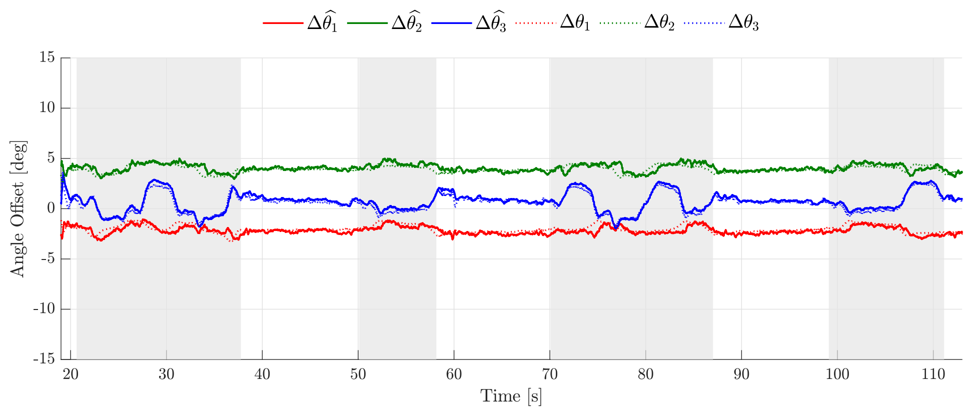


Figure 6.25: The applied angle offsets given by the Relative End Effector Controller (continuous) and the true values (dotted) for scenario CS.

Since the servo angle correction is being estimated correctly, the resulting pen and spring tracking errors are reduced significantly. As shown in Figure 6.26, the adaptive delta arm control allows the pen trajectory to be tracked with millimetre level accuracy. As to be expected, the improved tracking also results in a lower visual error. As shown in Figure 6.27, the poor writing along horizontal lines and when changing contact are rectified. The 3-sigma bound is brought down to sub-millimetre level accuracy.

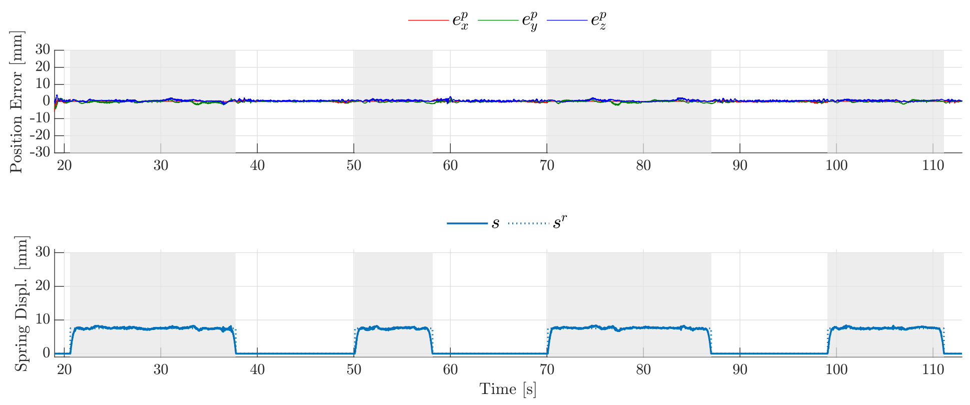


Figure 6.26: Pen tracking error (top) and Spring tracking (bottom) for scenario CS.

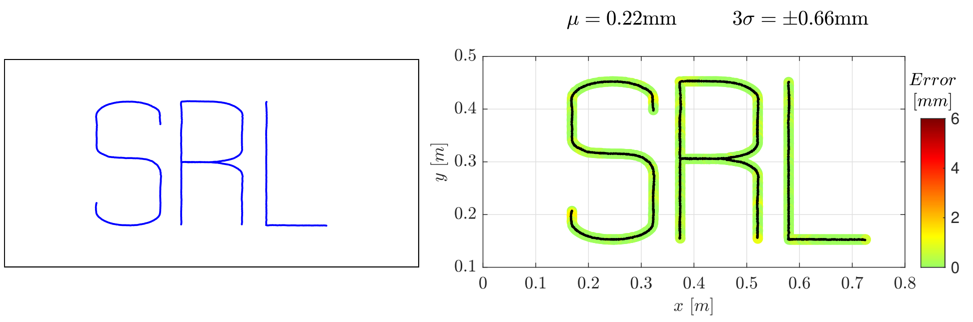


Figure 6.27: The actual drawing on the whiteboard (left) and the associated visual error for scenario `CS`.

## Repeatability

To show that the proposed adaptive control can cope with varying offset configurations, the experiments are repeated 5 times with randomly set additive and multiplicative servo offsets. The resulting tracking errors are shown in the top box plot in Figure 6.28 and are consistently reduced to the low millimetre range. Hence, the method works well for any servo angle offsets. To show that the method can also cope with other delta arm model mismatches, different length offsets on the push rod are simulated. The results are shown in the bottom box plot. As expected, perturbing the rod length affects the $x$ axis the most. Using the feedback, both spring and pen tracking errors are removed reliably.

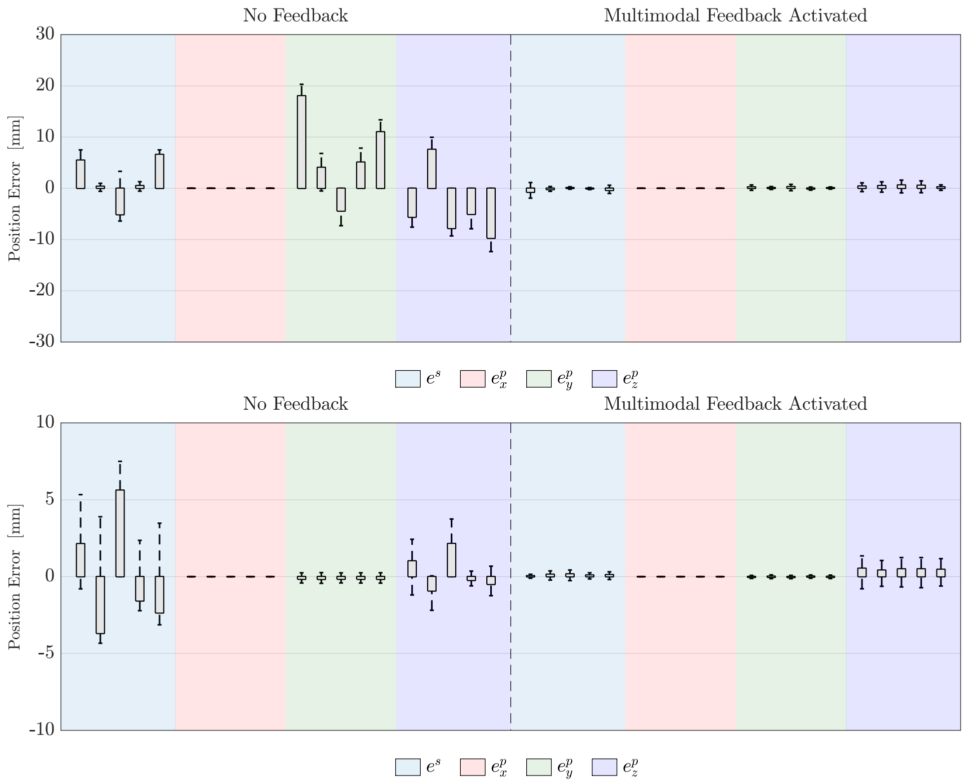


Figure 6.28: Box plot of spring and pen tracking errors for 5 iterations for scenario `S` (top-left), `CS` (top-right), `L` (bottom-left) and `CL` (bottom-right).

### 6.3.6 Combined Error Sources

After testing each of the three proposed feedback components individually, the natural next step is to combine them. This corresponds to the expected setting when performing real-world Aerial Writing tasks outside of a flight arena where no external motion capture is available, the whiteboard orientation would be hard to calibrate and the delta arm suffers from model mismatches.

#### Combined

As for the other simulations, the feedback is turned off during scenario `C` while the error sources are simulated. The error sources are kept identical as for the single error source experiments, meaning that the same drift, whiteboard misalignment and servo offsets are assumed for the corresponding iterations.
Figure 6.29 indicates that the spring and pen tracking is poor. The pen tracking error is in the centimetre range now while the spring goes from being not displaced at all to being pushed too hard. It should be noted here that the different error sources blend together in these plots and cannot be distinguished from each other. Given the poor pen tracking, the Aerial Writing performance shown in Figure 6.30 is poor as well. The final outcome misses consistency between the letters and the drawing is even interrupted at the end of the letter ‘S’ due to loss of contact. Further, the two ‘overlapping’ lines in the ‘R’ are not on top of each other. In addition, the vertical line of the ‘L’ appears to be extremely slanted. The 3-sigma bound indicates errors spanning across almost 5 cm around the intended drawing.

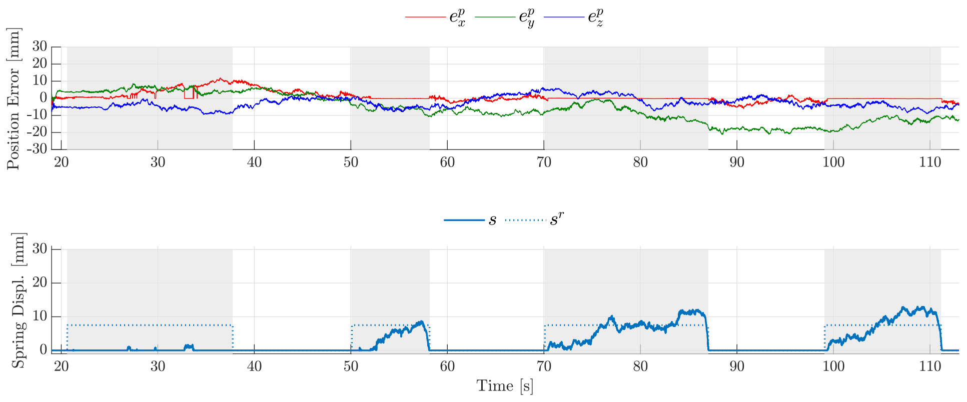


Figure 6.29: Pen tracking error (top) and Spring tracking (bottom) for scenario `A`.

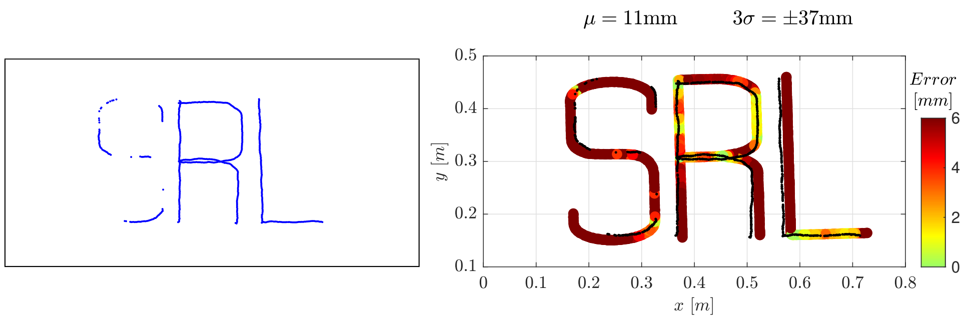


Figure 6.30: The actual drawing on the whiteboard (left) and the associated visual error for scenario `A`.

### Corrected Combined

For the last scenario `CA`, the three feedback components are activated in order to rectify the three error sources. In the following, the performance of each feedback loop as well as the resulting overall Aerial Writing precision are shown. First, the performance of the MAV Position Drift Estimator is plotted in Figure 6.31. As before, the estimated EKF states are plotted as continuous lines and the true values as dotted lines. The drift tracking is similar to the case where its done individually. As long as the pen is in contact, the drift is estimated accurately. The estimate along the $x$ axis, $\varDelta\widehat{x}$, is still tracked well. However, a small offset can be observed as the error is consistently underestimated. The estimates $\varDelta\widehat{y}$ and $\varDelta\widehat{z}$ show a similar tracking performance as in the associated scenario `CD`. As to be expected, once the pen detaches from the whiteboard, the estimate $\varDelta\widehat{x}$ no longer follows the true drift. The other two DoFs are still tracked loosely as long as the template matching does not fail due to the scale difference in the real and template image.

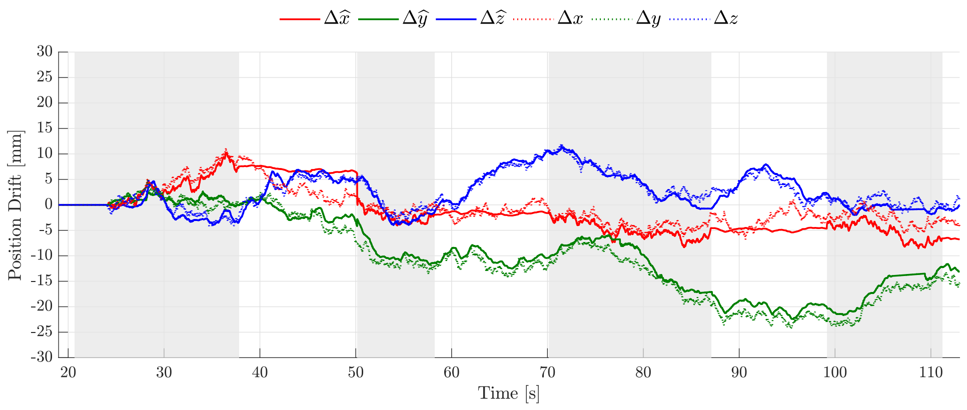


Figure 6.31: The estimated EKF states of the MAV Position Drift Estimator for scenario `CA`.

The Whiteboard Orientation Estimator is evaluated in Figure 6.32. The estimated EKF states are plotted as continuous lines and the true values as dotted lines. As soon as the pen touches the whiteboard, the estimates jump far past the correct value but then bounces back to be close to the true value. This can be explained as an effect of interference between the different error sources and feedback components. Here, the servo offsets cause the pen to be positioned further back if no feedback was used. Since the Relative End Effector Position Controller cannot correct this infinitely fast and exact, the pen is still too far back when it should first be touching the whiteboard. By adjusting the rotation estimates, the Whiteboard Orientation Estimator attempts to account for this offset, leading to the spikes in pitch and yaw. After the spike, the estimate moves towards the true value. However, as the first letter 'S' is written, the estimate does not converge perfectly. Instead, the orientation estimates $\varDelta\widehat{\theta}$ and $\varDelta\widehat{\psi}$ remain slightly off. This is can be linked to the small offset in the drift estimate along the $x$ axis mentioned before. Again, we see undesired interference between the different feedback components. Overall, the orientation estimation does not work as perfectly as in scenario `CW` but still delivers reasonable estimates.
The performance of the Relative End Effector Position Control method is plotted in Figure 6.33. The angle corrections $\varDelta\widehat{\theta_1}, \varDelta\widehat{\theta_2}, \varDelta\widehat{\theta_3}$ are plotted as continuous lines and the true values as dotted lines. All three servo angles are tracked accurately. As for scenario `CS`, small oscillations can be seen. Overall, this feedback loop shows comparable performance to when used on its own.

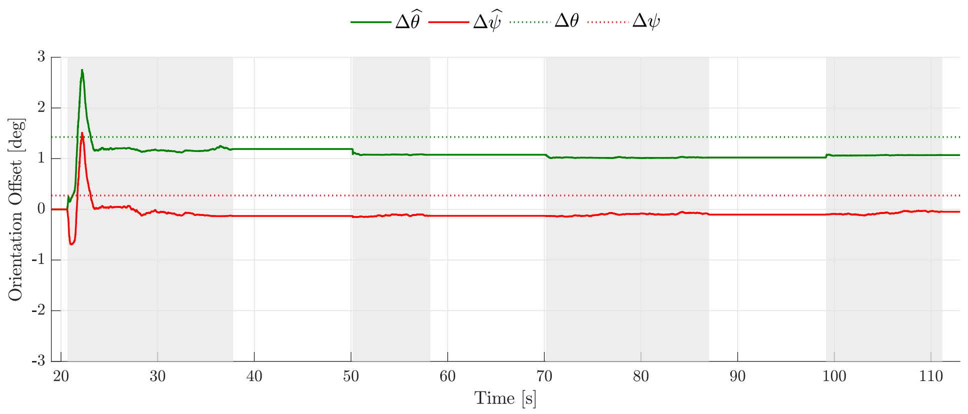


Figure 6.32: The estimated EKF states of the Whiteboard Orientation Estimator for scenario `CA`.

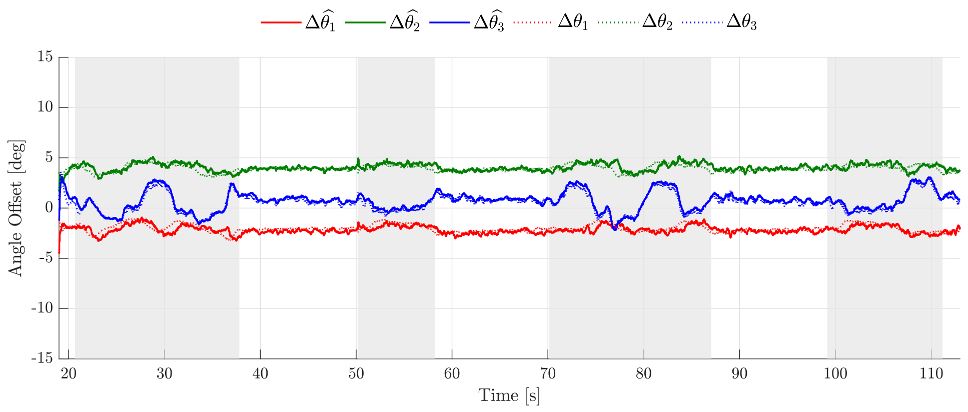


Figure 6.33: The applied angle offsets given by the Relative End Effector Controller for scenario `CA`.

The resulting overall tracking errors are small as shown in Figure 6.34. While in contact, they remain in the range of 5 mm, being significantly lower for most of the contact phases. This is in strong contrast to Figure 6.29 where the error is around 20 mm for extended periods. The achieved visual improvement is shown in Figure 6.35. Apart from minor inaccuracies, the drawing precision is very high ensuring consistent spacing between the letters and straight lines being truly straight. Further, the overlapping lines on the letter ‘R’ are now actually on top of each other. The 3-sigma bound being in the low millimetre range further shows the high accuracy of the Aerial Writing performed under the influence of all three errors.

### Repeatability

The combined scenario is also performed on 5 times while varying all error sources. As shown on the left of Figure 6.36, the tracking errors are in the range of centimetres, usually along multiple axes. Using the multimodal feedback, the error can be decreased significantly to being in the low millimetre range. Close to zero errors are achieved for the spring and hence the pen tracking along the $x$ axis. The errors along the $z$ axis are very low for all but one iteration while the error along $y$ is very low for all but two iterations. For these iterations, the specific drift profile causes the method to accumulate small offsets due to the discussed limited accuracy.

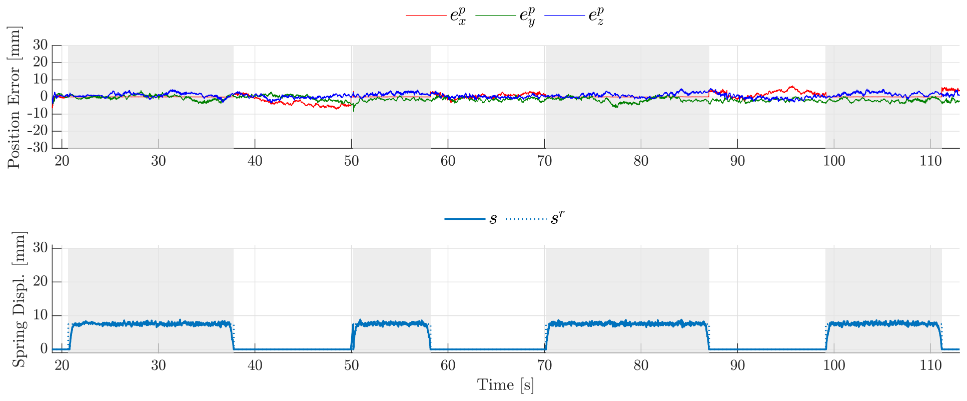


Figure 6.34: Pen tracking error (top) and Spring tracking (bottom) for scenario `CA`.

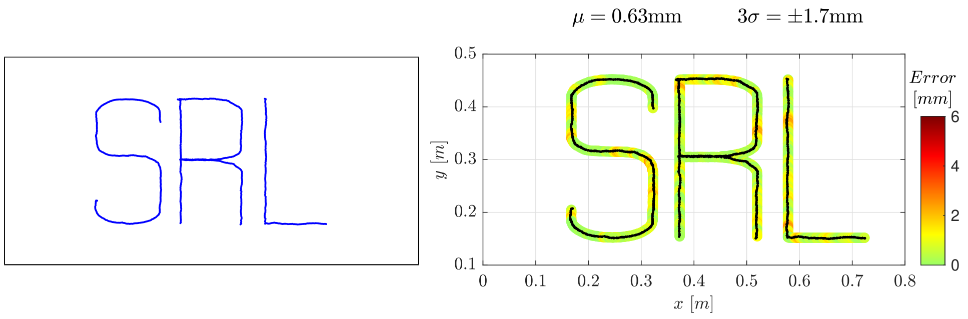


Figure 6.35: The actual drawing on the whiteboard (left) and the associated visual error for scenario `CA`.

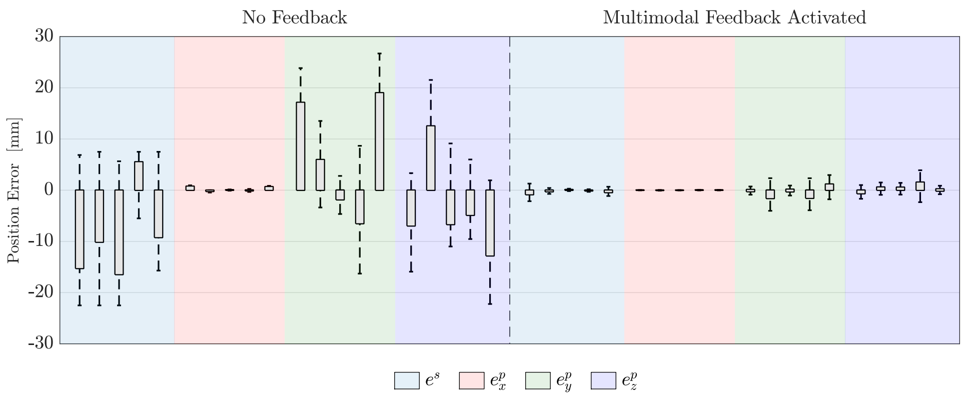


Figure 6.36: Box plot of spring and pen tracking errors for 5 iterations for scenario `A` (left) and `CA` (right).

# Chapter 7

# Discussion

## 7.1 Real-world Experiments

The system controlled by the pre-existing NMPC pipeline achieves accurate and consistent results over a series of different trajectories. The tracking error of the end effector is significantly lower than the one for the MAV, highlighting the accuracy boost due to the utilisation of the arm. It should be mentioned that the system was built using relatively cheap, off-the-shelf components and 3D printed parts. This leads to errors in the manufacturing of the aerial system with respect to the reference model e.g. errors in the true inverse kinematics of the arm due to non-identical dimensions of its links. As explained, this then lead to the introduction of a more adaptive delta arm controller presented in Section 5.4.

Another important issue that occurred during these experiments was the reliance on the motion capture system for localisation. Apart from issues related to WiFi delays which resulted in temporary loss of tracking, poor object visibility sometimes resulted in unreliable estimates during the missions of both the static objects, such as the whiteboard, and moving ones such as the MAV. In fact, during data analysis it was found that there are segments where Vicon returned mechanically impossible configurations for the system e.g. end effector positions below the surface of the Touch frame. Despite these problems which further propagate into tracking errors, the system was able to handle multiple transitions to contact during the same experiment. In addition, the motion capture system showed high orientation errors for the calibration of the Touch frame, even when using hundreds of pose samples in an optimisation-based calibration sequence.

It was further experimentally verified that for contact tasks, where the main objective is accuracy instead of speed, using a planner respecting full state dynamic feasibility is not an absolute necessity. Despite the use of a simplified motion planner, the system achieves sub-centimetre accuracy. However, for more aggressive maneuvers, a full state dynamically feasible plan would be required.

## 7.2 Simulations

The key finding from the simulated experiments of the proposed multimodal feedback is that all three error sources are estimated accurately and can hence be rectified effectively, both when appearing individually and in combination. The method is shown to work for random sets of errors with consistent performance. It can therefore be concluded that the proposed approach is repeatable. It is shown that both the spring and pen tracking are improved significantly to reach a high accuracy in the low millimetre range. Further, drastic improvements in terms of

the appearance-based visual precision metric are shown. The Aerial Writing performance goes from very poor, e.g. non-continuous, slanted lines, missing consistency between and within letters, to a similar performance as seen in the baseline scenario. Unintended detaching, crooked lines and wrong positioning and scaling of letters and parts thereof are all prevented through the use of feedback.
The issues with the real-world experiments, i.e. poor manufacturing of the delta arm and hence a model mismatch and inaccurate calibration of the Touch frame, are shown to be addressed by the respective feedback components. Further, issues with external motion capture as described in Section 7.1 would not be present in the intended mission environment as onboard SLAM would be in use. The SLAM system's drifting position estimates would get rectified effectively as shown in the drift estimation simulations.
It should be noted here that the simulation environment in use is very realistic: Gazebo combined with the RotorS framework allows highly accurate simulation of the MAV dynamics. By simulating the Aerial Writing carried out by an exact virtual replica of the real delta arm, as well as the noisy visual and tactile measurements all in the same loop, the simulation setup comes very close to what one would expect in real-world experiments. This should simplify the transition to real-world experiments using the same multimodal feedback. The achieved accuracy is within the maximum achievable precision of the real MAV-arm system (low millimetre range) and should thus allow similar precision as seen in the real-world experiments while removing the addressed errors.
During the simulations, the approach was shown to have three key limitations:

***Limited Precision of Visual Feedback*** The visual feedback based on template matching does not provide perfect tracking due to discretisation errors caused by the camera's low resolution and the missing tight time synchronisation between the camera and template image. The drift along the DoFs parallel to the camera image plane are therefore not tracked as accurately as the DoF informed by tactile feedback.

***Loss of Observability in Free-Flight*** When not in contact, errors in the DoF along the spring are no longer estimated correctly. This is due to the tactile sensor being disconnected and the visual feedback only providing a 2D error in the camera image plane. In other words, the method loses observability over the DoF perpendicular to the camera image plane. As the estimate along $x$ gets more inaccurate while not in contact, the template image will be rendered from the wrong position leading to a scale discrepancy between the template and real image. Over time, this causes the template matching performance to decrease or even fail entirely. When reattaching, the accumulated errors can only be corrected once the pen touches the whiteboard again and the tactile feedback is back. Once the corrections are done, some strokes were already drawn and cannot be corrected anymore.

***Interference between Feedback Components*** The third limitation is that when running all feedback components together, a limited interference between them is observed. There is ambiguity in the errors that each of them is trying to minimise, leading to inaccuracies in the individual state estimates. For example, a tactile measurement indicating contact further back than expected can either be accounted for as drift along the $x$ axis, a wrongly oriented whiteboard or a mismatch in the delta arm model. This slightly deteriorates the overall drawing accuracy under error influence. However, given the local nature of the adaptive delta arm control and the different model assumptions for drift and whiteboard orientation, i.e. random walk vs. fixed offset, the ambiguities are cleared up to a satisfactory degree as the mission progresses.

# Chapter 8

# Conclusion

To conclude this Master thesis, the proposed contributions are stated again and some direction for future work is given.

## 8.1 Validated Contributions

In this thesis, the following contributions are proposed:

1. The pre-existing NMPC method is extended to include multimodal feedback from visual and tactile measurements to allow Aerial Writing in a real-world setup with onboard MAV localisation, a faulty Whiteboard orientation calibration and a erroneous delta arm model. The approach is validated in extensive and highly realistic simulations.
2. A highly realistic simulator is developed to evaluate the proposed feedback while providing full simulation of the Aerial Writing process, the error sources as well as visual and tactile sensor outputs. The simulator is shown to accurately reflect the real-world system behavior.
3. The pre-existing NMPC method proposed in [9] is evaluated experimentally in Aerial Writing tasks using an appearance-based visual error metric. Due to the lack of a widely used error metric for Aerial Writing, this metric is proposed as a new standardised way to measure the precision of platforms performing such tasks. The error metric is shown to be suitable to capture the systems performance in a natural and intuitive manner.

## 8.2 Future Work

In terms of future work, the following should be distinguished: additional steps that are planned be done after this thesis is completed, other minor changes which could be added to the existing approach and major changes that alter the conceptual idea of the proposed method. The planned steps are discussed first:

***Real-world Evaluation*** The multimodal feedback pipeline proposed in this thesis is targeted towards enabling Aerial Writing under real-world conditions with various kinds of errors affecting the system. However, it is only evaluated in simulations in this thesis. This is caused by lock-down measures implemented to fight the outbreak of SARS-CoV-2 in London in the spring of 2020. During the period of the thesis during which real-world experiments were planned, Imperial's campus was closed down. Hence, neither the aerial manipulator nor other Aerial Writing equipment, e.g. the whiteboard, were accessible.

As the lock-down restrictions will be eased, the multimodal feedback will be evaluated on a drone performing real Aerial Writing tasks.

***Submission of Findings*** The findings presented in this thesis along with the results from the just described real-world evaluation will then be submitted as a conference paper. As of the time of writing this thesis, the planned goal is to submit to ICRA 2021.

Minor improvements which would improve the existing approach are listed here:

***Scale Sensitive Template Matching*** By adding scale sensitivity to the template matching, the visual feedback could provide a depth error thus going from a 2D to a 3D signal. This could be done by rendering the template image in two more versions, one at a slightly lower and one at a higher scale. The real image could then be matched to all three templates and the best match would be chosen and the according shift in scale would serve as the depth error. Ideally, this would prevent the loss of observability along one axis when the pen is not in contact.

***High Resolution and Wide FoV Camera*** Especially when drawing large letters, the camera can end up observing only one straight line due to its small field-of-view. When performing template matching, this leads to low invariance of the match along one direction and thus a less reliable visual update signal. Further, the current resolution of the camera is limited to allow real-time computation. This leads to a lower overall accuracy of the visual feedback. By using a camera with a bigger FoV, e.g. by adding a fish-eye lens, and higher resolution these issues could be fixed.

***Mask Push Rods*** In the current implementation, only the end effector with the attached pen case is masked out in the template and real image. However, occlusions caused by the push rods are ignored. Including these into the mask would improve the template matching reliability.

***Pressure Driven Drawing*** Visual effects based on changing font thickness depending of how much pressure is applied are ignored in the simulator as the pen in the real-world experiments is very stiff. Including this by assuming a softer pen tip could open up new interesting ways of Aerial Drawing, as the range of achievable strokes and patterns would increase dramatically.

Finally, major improvements requiring significant changes or alterations to the underlying approach include the following:

***Time Synchronisation*** Instead of processing the camera images and other sensor measurements in a callback-driven approach, a tightly timed pipeline should be used. This would allow much more exact temporal synchronisation between the measurements, which would benefit the visual drift tracking the most. In the current implementation, this was not done due to the significant additional engineering efforts required.

***Dense Visual Tracking*** Following a similar approach as [58], the matching between camera and template image could be done in a dense fashion instead of doing simple template matching. By leveraging dense photometric residuals and additional ICP errors if depth is available, it should be possible to achieve more precise tracking.

# Bibliography

[1] F. Ruggiero, V. Lippiello, and A. Ollero, “Aerial Manipulation: A Literature Review,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1957–1964, 2018.

[2] H. B. Khamseh, F. Janabi-Sharifi, and A. Abdessameud, “Aerial manipulation—A literature survey,” *Robotics and Autonomous Systems*, vol. 107, pp. 221 – 235, 2018.

[3] T. Ikeda, S. Yasui, M. Fujihara, K. Ohara, S. Ashizawa, A. Ichikawa, A. Okino, T. Oomichi, and T. Fukuda, “Wall contact by octo-rotor uav with one dof manipulator for bridge inspection,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 5122–5127.

[4] M. Ángel Trujillo, J. R. M. de Dios, C. Martín, A. Viguria, and A. Ollero, “Novel aerial manipulator for accurate and robust industrial ndt contact inspection: A new tool for the oil and gas inspection industry,” *Sensors*, vol. 19, no. 6, 2019.

[5] K. Bodie, M. Brunner, M. Pantic, S. Walser, P. Pfändler, U. Angst, R. Siegwart, and J. Nieto, “An Omnidirectional Aerial Manipulation Platform for Contact-Based Inspection,” in *Robotics: Science and Systems*, June 2019.

[6] C. Papachristos, K. Alexis, and A. Tzes, “Technical activities execution with a tiltrotor uas employing explicit model predictive control,” *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 11 036 – 11 042, 2014, 19th IFAC World Congress.

[7] C. C. Kessens, J. Thomas, J. P. Desai, and V. Kumar, “Versatile aerial grasping using self-sealing suction,” in *IEEE International Conference on Robotics and Automation*, 2016, pp. 3249–3254.

[8] F. Augugliaro, S. Lupashin, M. Hamer, C. Male, M. Hehn, M. W. Mueller, J. S. Willmann, F. Gramazio, M. Kohler, and R. D’Andrea, “The flight assembled architecture installation: Cooperative construction with flying machines,” *IEEE Control Systems Magazine*, vol. 34, no. 4, pp. 46–64, Aug 2014.

[9] D. Tzoumaniks, F. Graule, Q. Yan, D. Shah, M. Popovic, and S. Leutenegger, “Aerial manipulation using hybrid force and position nmpc applied to aerial writing,” in *Robotics: Science and Systems*, 2020.

[10] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, “Keyframe-based visual–inertial odometry using nonlinear optimization,” *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[11] D. Brescianini and R. D’Andrea, “Design, modeling and control of an omni-directional aerial vehicle,” in *IEEE International Conference on Robotics and Automation*, May 2016, pp. 3261–3266.

[12] D. Brescianini and R. D'Andrea, "Computationally efficient trajectory generation for fully actuated multirotor vehicles," *IEEE Transactions on Robotics*, vol. 34, pp. 555–571, 2018.

[13] M. Ryll, G. Muscio, F. Pierri, E. Cataldi, G. Antonelli, F. Caccavale, D. Bicego, and A. Franchi, "6D interaction control with aerial robots: The flying end-effector paradigm," *The International Journal of Robotics Research*, vol. 38, no. 9, pp. 1045–1062, 2019.

[14] M. Kamel, S. Verling, O. Elkhatib, C. Sprecher, P. Wulkop, Z. Taylor, R. Siegwart, and I. Gilitschenski, "Voliro: An Omnidirectional Hexacopter With Tiltable Rotors," *arXiv*, 2018.

[15] K. Bodie, M. Brunner, M. Pantic, S. Walser, P. Pfändler, U. M. Angst, R. Siegwart, and J. I. Nieto, "Active interaction force control for omnidirectional aerial contact-based inspection," *ArXiv*, vol. abs/2003.09516, 2020.

[16] C. Papachristos, K. Alexis, and A. Tzes, "Efficient force exertion for aerial robotic manipulation: Exploiting the thrust-vectoring authority of a tri-tiltrotor uav," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 4500–4505.

[17] G. Darivianakis, K. Alexis, M. Burri, and R. Siegwart, "Hybrid predictive control for aerial robotic physical interaction towards inspection operations," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 53–58.

[18] D. Mellinger, Q. Lindsey, M. Shomin, and V. Kumar, "Design, modeling, estimation and control for aerial grasping and manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 2668–2673.

[19] S. Kim, S. Choi, and H. J. Kim, "Aerial manipulation using a quadrotor with a two DOF robotic arm," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 4990–4995.

[20] M. Orsag, C. Korpela, S. Bogdan, and P. Oh, "Valve turning using a dual-arm aerial manipulator," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, May 2014, pp. 836–841.

[21] F. Ruggiero, M. A. Trujillo, R. Cano, H. Ascorbe, A. Viguria, C. Peréz, V. Lippiello, A. Ollero, and B. Siciliano, "A multilayer control for multirotor uavs equipped with a servo robot arm," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4014–4020.

[22] A. Suarez, A. E. Jimenez-Cano, V. M. Vega, G. Heredia, A. Rodriguez-Castaño, and A. Ollero, "Design of a lightweight dual arm system for aerial manipulation," *Mechatronics*, vol. 50, pp. 30 – 44, 2018.

[23] V. Nayak, C. Papachristos, and K. Alexis, "Design and control of an aerial manipulator for contact-based inspection," *ArXiv*, vol. abs/1804.03756, 2018.

[24] 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.

[25] M. Kamel, S. Comari, and R. Siegwart, "Full-body multi-objective controller for aerial manipulation," in *2016 24th Mediterranean Conference on Control and Automation (MED)*, June 2016, pp. 659–664.

[26] K. Steich, M. Kamel, P. Beardsley, M. K. Obrist, R. Siegwart, and T. Lachat, "Tree cavity inspection using aerial robots," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 4856–4862.

[27] M. Sharma and I. Kar, "Adaptive geometric control of quadrotors with dynamic offset between center of gravity and geometric center," *Asian Journal of Control*, vol. n/a, no. n/a, 2020.

[28] S. Park, J. Lee, J. Ahn, M. Kim, J. Her, G.-H. Yang, and D. Lee, "Odar: Aerial manipulation platform enabling omni-directional wrench generation," *IEEE/ASME Transactions on Mechatronics*, vol. PP, pp. 1–1, 07 2018.

[29] A. S. Vempati, M. Kamel, N. Stilinovic, Q. Zhang, D. Reusser, I. Sa, J. Nieto, R. Siegwart, and P. Beardsley, "Paintcopter: An autonomous uav for spray painting on three-dimensional surfaces," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2862–2869, 2018.

[30] L. S. Monteiro, T. Moore, and C. Hill, "What is the accuracy of dgps?" *Journal of Navigation*, vol. 58, no. 2, p. 207–225, 2005.

[31] P. Newman and Kin Ho, "Slam-loop closing with visually salient features," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 635–642.

[32] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for rgb-d cameras," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 2100–2106.

[33] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 834–849.

[34] P. Merriaux, Y. Dupuis, R. Boutteau, P. Vasseur, and X. Savatier, "A study of vicon system positioning performance," *Sensors*, vol. 17, no. 7, p. 1591, Jul 2017.

[35] F. Pierrot, C. Reynaud, and A. Fournier, "Delta: a simple and efficient parallel robot," *Robotica*, vol. 8, no. 2, p. 105–109, 1990.

[36] P. Vischer and R. Clavel, "Kinematic calibration of the parallel delta robot," *Robotica*, vol. 16, no. 2, pp. 207–218, 1998.

[37] M. Stock and K. Miller, "Optimal Kinematic Design of Spatial Parallel Manipulators: Application to Linear Delta Robot ," *Journal of Mechanical Design*, vol. 125, no. 2, pp. 292–301, 06 2003.

[38] E. Castillo Castaneda, G. a, G. García, and A. Bashir, "Delta robot: Inverse, direct, and intermediate jacobians," *Proceedings of The Institution of Mechanical Engineers Part C-journal of Mechanical Engineering Science - PROC INST MECH ENG C-J MECH E*, vol. 220, pp. 103–109, 01 2006.

[39] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *IEEE Robotics Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[40] F. Chaumette and S. Hutchinson, "Visual servo control, part ii: Advanced approaches," *IEEE Robot. Autom. Mag.*, vol. 14, 01 2007.

[41] G. Allibert, E. Courtial, and F. Chaumette, “Predictive control for constrained image-based visual servoing,” *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 933–939, 2010.

[42] A. Assa and F. Janabi-Sharifi, “Robust model predictive control for visual servoing,” in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 2715–2720.

[43] A. Hajiloo, M. Keshmiri, W. Xie, and T. Wang, “Robust online model predictive control for a constrained image-based visual servoing,” *IEEE Transactions on Industrial Electronics*, vol. 63, no. 4, pp. 2242–2250, 2016.

[44] H. Seo, S. Kim, and H. J. Kim, “Aerial grasping of cylindrical object using visual servoing based on stochastic model predictive control,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 6362–6368.

[45] R. Brunelli, “Template matching techniques in computer vision,” 09 2008.

[46] M. B. Hisham, S. N. Yaakob, R. A. A. Raof, A. B. A. Nazren, and N. M. W. Embedded, “Template matching using sum of squared difference and normalized cross correlation,” in *2015 IEEE Student Conference on Research and Development (SCOReD)*, 2015, pp. 100–104.

[47] K. Nickels and S. Hutchinson, “Estimating uncertainty in ssd-based feature tracking,” *Image and Vision Computing*, vol. 20, no. 1, pp. 47 – 58, 2002.

[48] M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, “Inversion based direct position control and trajectory following for micro aerial vehicles,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 2933–2939.

[49] M. Giftthaler, M. Neunert, M. Stauble, and J. Buchli, “The control toolbox — an open-source c++ library for robotics, optimal and model predictive control,” *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, May 2018.

[50] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154 vol.3.

[51] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *Robot Operating System (ROS): The Complete Reference (Volume 1)*. Cham: Springer International Publishing, 2016, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625.

[52] L. Kleeman, “Understanding and applying kalman filtering,” Online, 2006.

[53] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.

[54] D. Malyuta, “Guidance, Navigation, Control and Mission Logic for Quadrotor Full-cycle Autonomy,” Master thesis, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91109, USA, Dec. 2017.

[55] J. Wang and E. Olson, “AprilTag 2: Efficient and robust fiducial detection,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, oct 2016, pp. 4193–4198.

[56] B. K. P. Horn, “Closed-form solution of absolute orientation using unit quaternions,” *J. Opt. Soc. Am. A*, vol. 4, no. 4, pp. 629–642, Apr 1987.

[57] P. J. Besl and N. D. McKay, “A method for registration of 3-d shapes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb 1992.

[58] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. J. Davison, and S. Leutenegger, “Mid-fusion: Octree-based object-level multi-instance dynamic slam,” *2019 International Conference on Robotics and Automation (ICRA)*, pp. 5231–5237, 2019.