ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autonomous Systems Lab
Prof. Roland Siegwart

Semester Thesis

High Level Control of an MAV through Marker-Based Visual Commands

Autumn Term 2016

Supervised by:
Helen Oleynikova
Michael Burri

Author:
Dimitris Gryparis

Declaration of Originality

I hereby declare that the written work I have submitted entitled

High Level Control of an MAV through Marker-Based Visual Commands

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

Author(s)

Dimitris	Gryparis

Student supervisor(s)

Helen	Oleynikova
Michael	Burri

Supervising lecturer

Roland	Siegwart

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.

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

In this project is presented the high level control of a Micro Aerial Vehicle (MAV) through Marker-Based visual commands. In this project the main objective, is to control the pose of an AscTec's Firefly MAV, based only on the detections of specific visual markers from its camera sensor. At all cases, we want to ensure the user's safety, thus we emphasize at always leaving a safety distance between the MAV and the user. Furthermore, we also want to maximize the detection rate of the markers. In order to achieve our goals, all the system's parts were built and tested in simulation and then experiments were conducted on a real MAV platform to evaluate our results. Both in simulation and in the real system the used visual markers were Apriltags[2].

Symbols

Symbols

ϕ, θ, ψ	roll, pitch and yaw angle

Indices

x	x axis
y	y axis
z	z axis

Acronyms and Abbreviations

ETH	Eidgenössische Technische Hochschule
ASL	Autonomous Systems Lab
IMU	Inertial Measurement Unit
UAV	Unmanned Aerial Vehicle
MAV	Micro Aerial Vehicle
ROS	Robot Operating System
FPS	Frames Per Second

Chapter 1

Introduction

During the past decades, robotics has evolved from the heavy industrial manipulators, that worked only to a very specific structured environment doing specific repetitive tasks, to mobile robots that navigate themselves to unstructured environments and they are able to perform a variety of tasks. A particular field of robotics that has gained a lot of attention lately, is localization and motion of the mobile robot based only on information coming from its own sensors, and particularly its cameras.
The reason for choosing cameras as the main source of information in modern robotics, and as in our specific case to an MAV, is that they provide an abundance of information while their weight, power consumption and price is considerably less compared to other kinds of sensors, such as laser scanners. Furthermore, nowadays exist advanced computer vision algorithms that allow us to robustly extract the necessary information and of course the proper hardware that allows the on board and fast processing of the acquired data.
Furthermore, the scientific interest regarding autonomous mobile robots, is strongly attracted to low weight multi rotor MAVs. Some of their merits over conventional designs, such as fixed wing platforms, are their ability for vertical take off and landing, they can hover over specific areas as long as their energy source allows and their ability to perform tight maneuvers in confined spaces. Their wide commercial availability combined with their reduced prices makes them a very attractive platform for experimentation. Their uses, which are rapidly increasing, include parcel delivery, inspection of structures, search and rescue operations and of course use by hobbyists for various other tasks.
In this project, the AscTec's Firefly MAV is used, to present the realization of the MAV's high level control only with the use of visual commands. Thus, the MAV uses only the data available from its onboard sensors to deduce its desired pose in the three dimensional world based on the user's visual commands. These commands are given by the Apriltag markers. First the whole system was build in simulation, using the RotorS simulator created by the ASL [3] and then the developed algorithms were evaluated in the real system.

1.1 Summary

In chapter 2 is a thorough system description, providing information about the visual markers, the MAV used and the frame analysis of the system. In chapter 3, comes the setup of the system, explaining the various packets written in order to make the system run in simulation. In chapter 4, there will be presented the experimental results both from the simulated and the real system. Finally in chapter 5

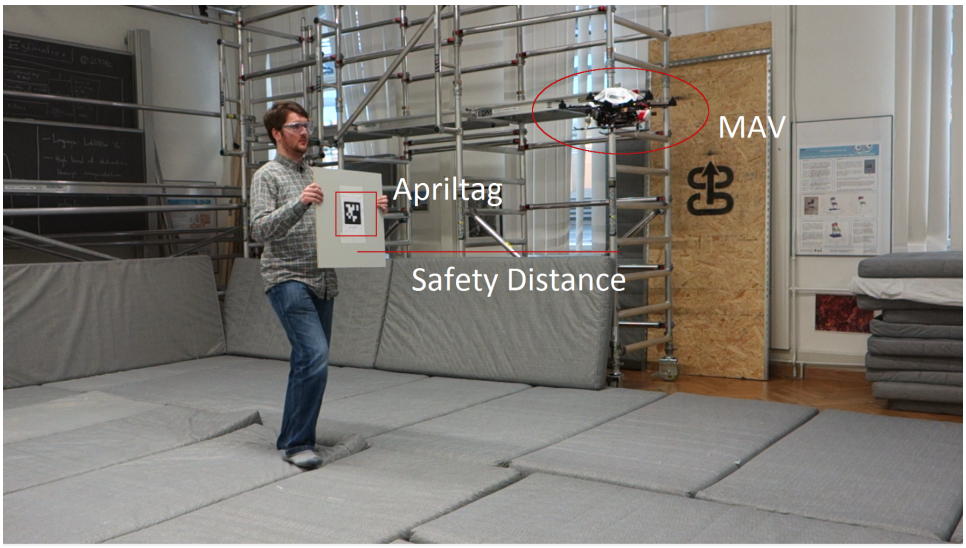

Figure 1.1: The MAV detects the marker and moves to the desired position, based on the marker detection

we present the conclusions along with some ideas about what could be done in the future.

1.2 Related Work

In the recent years, many teams have been able to perform various kinds of motions with the MAVs ([4],[5],[6], [7]), but all of them heavily relied on the use of external motion capture systems (VICON[1]), in order to obtain localization data and their maneuvers are generally predefined. Other teams, have used Apriltags[2] for providing localization data to their robots. Such examples are [8] and [9], they use them to provide localization data in ground vehicles with different configurations, in the first case the Apriltags are statically attached to the ceiling of the lab while in the latter are attached on the robots and on the walls of the lab. At the Carnegie Mellon team [9], their use is to provide both relative localization among the ground vehicles but also providing absolute localization information from the static located markers. Furthermore, in [10] and [11] they use the aforementioned markers in order to land the MAVs towards, static or moving landing sights. Other uses of the marker include [12] where they were used to compensate sensor loss in a different MAV by providing its 6 DOF pose estimate based on the marker detection and in [13] where it augments the localization data and compensates the IMU data drift.

In our case, we wanted a system that would be able to provide information that would allow us to follow the visual commands presented by the user and also ensure that there will be adequate distance between the user and the MAV, as presented in figure 1.1. Thus, we also used the Apriltag markers. Our approach differs form the aforementioned, since we don't use an external motion capture system in order to decide the MAV's desired position. The markers are not statically attached to the ceiling or to the lab's walls, the markers are held by the user and they can be moved without limitations, as long as they stay in the camera's field of view. Furthermore, our system is able to land anywhere just by presenting the proper marker to the MAV and finally we do not use the Apriltags in order to compensate or substitute sensor mistakes or loss. It can be easily seen, that the advantages

of our approach are the fact that we do not rely on any external system, for the marker detection and desired position estimation, our system is highly versatile, we don't use any static landmarks and we can land at any place the user wants. Of course, there are some disadvantages too. First of all, the precision and resolution of our system, is much lower compared to systems using VICON. Furthermore, the unconstrained motion of the marker leads sometimes to the loss of the marker's detection, since the Apriltag can get out of the camera's field of view. Last but not least, the ability of the MAV to land whenever the user commands it, may lead to problematic situations in cases where the user chooses poorly the landing position.

[1] http://www.vicon.com

Chapter 2

System Description

In this chapter, first will be a short analysis of the markers used in this project, the Apriltags. Then will be presented some information about the MAV, used to conduct experiments both in simulation and in real world. And finally there will be the system's coordinate frame analysis.

2.1 Apriltags

In this project, the markers used to provide visual commands to the MAV were the Apriltags. They were created by Edwin Olson at the university of Michigan. These are visual fiducials, that mean artificial landmarks easily recognizable and distinguishable from one another. In contrast with other similar systems, they provide low volume of information, usually between 4 till 12 bits, thus they don't need high resolution images which allows long detection ranges. They differ from other similar approaches because they provide robustly 6 DOF position estimation while they can be printed with a standard printer and they can still provide detection under variable lighting conditions or even deformation of the paper on which they are printed. They have been widely used to provide ground truth trajectories in structured environments, Simultaneous Localization And Mapping (SLAM) and they are very common in augmented reality applications [2].
There are various families of Apriltags, providing a large number of different markers. In figure 2.2 is presented the first aprliltag (id = 0) of the most common apriltag family, the "tag36h11". The aforementioned family, is the one suggested for use by the authors, and is also the default choice in the given detection algorithm. The difference between these families of markers is located at the amount of information carried by each tag.
As it can be easily seen by the aforementioned, the Apriltags were selected for use in this project for three main reasons. First, we wanted to use only detection data provided by the MAV's onboard camera. They provide robust position estimate and robust detection, by tolerating lighting variations and physical deformation of the marker.

2.2 AscTec Firefly

The AscTec's Firefly MAV was used both in the simulated and in the real experiments. This particular MAV, is a hexacopter, thus it has six rotors that provide a stable flight. It is equipped with a powerful Intel® Core^{TM} i7 processor, that allows us to run ROS along with all the visual tracking algorithms onboard, without significant time delays.

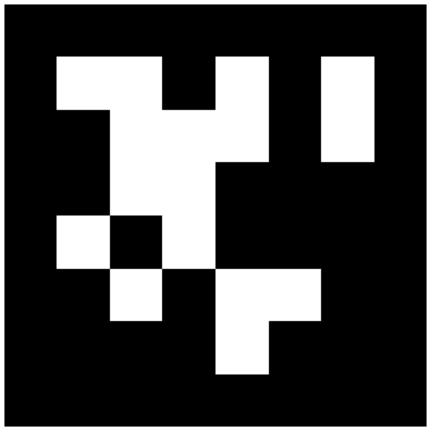

Figure 2.1: The first Apriltag (id=0) of the tag36h11 family

Figure 2.2: The Firefly MAV[1]

In our specific case the MAV was equipped with the Skybotix's VI (Visual Inertial) sensor [14]. The sensor is equipped with two cameras and an IMU unit. In our experiments only one camera was used in order to identify the Apriltags and localize against them. It should be mentioned, that the MAV with the aforementioned sensor are modeled into the RotorS simulator.

2.3 Coordinate Frames

A very crucial part of this project, was the analysis of the MAV's coordinate systems. This is very important because the MAV detects its desired position from its camera detections and then has to transform these detections, from the camera frame to the controller's reference frame with respect to the world frame. Furthermore, the Apriltag's position in the simulated world is defined by it's pose as detected from the usb camera. Thus, by correctly defining and analyzing the coordinate frames we will be able to correctly interpret the position data from the marker detection and also apply the safety distances, so as to achieve our goals to move the MAV based on the detections of the Apriltags and also keep a safety distance between itself and the marker. For the transformations the ASL's minkindr library was used. It should be noted, that the color coding in the axes frames follows the RVIZ color convention,

[1]The Firefly's image was taken from http://www.asctec.de/en

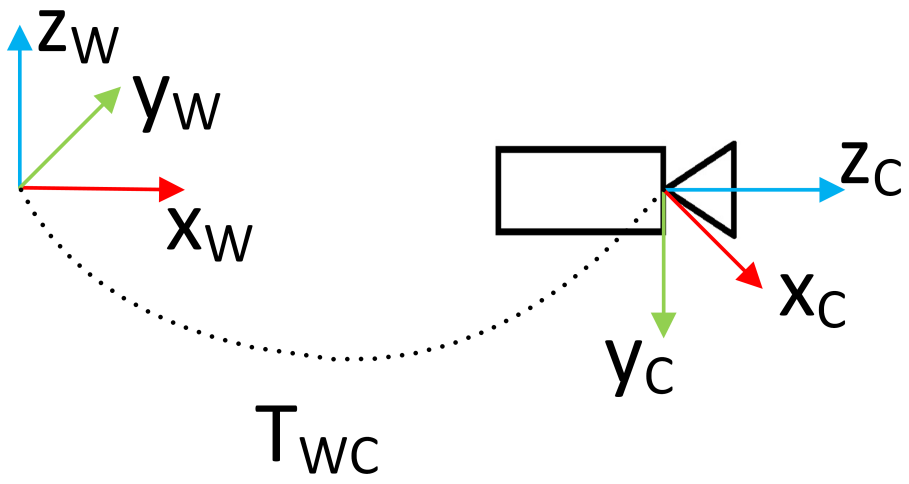

Figure 2.3: The image represents the world coordinate frame and the camera frame.

meaning red depicts the x-axis, green the y-axis and blue the z-axis respectively. Furthermore the ASL's frame naming convention [15] is followed throughout this project.

2.3.1 Coordinate Frames in the Simulation

The first thing that should be resolved in the simulation world, was the correct positioning of the apriltag with respect to the usb camera detection. Although the two frames share the same origin point, the have different orientations as seen by figure 2.3, the distance between the two frames is set to emphasize the rotation. As it can be seen the transformation form camera frame with respect to the world frame, R_{W_USBC}, consists of one rotation of 90° around the y-axis ($Rot_y(90°)$) and a successive rotation of $-90°$ around the z-axis ($Rot_z(-90°)$) as presented in the equation 2.1. With the aforementioned transformation, the coordinates as seen from the camera match the actual world coordinates.

$$\begin{bmatrix} x_{wc} \\ y_{wc} \\ z_{wc} \end{bmatrix} = Rot_y(90°)Rot_z(-90°) \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \tag{2.1}$$

After the transformation of the camera frame, now come the coordinate transformations regarding the MAV. The transformation between the MAV's body frame and the world frame is obtained by the firefly's odometry sensor, in the simulation it is named as odometry_sensor1, the obtained transformation is represented by T_{W_B}. This transformation reveals the real pose of the MAV in the world reference frame, as it is perceived from its IMU sensor. Then the static transform between the body frame and the onboard camera frame is provided as parameter by the configuration file and is represented as T_{B_C}. The aforementioned transform is passed as parameter in order to reduce the calculations needed to be perform and thus, speed up the procedure. The transformation between the MAV's camera and the Apriltag is obtained by the detection algorithm, by subscribing to the MAV's tag_detection_pose topic. This provides the pose of the detected Apriltag with respect to the camera coordinate frame. The last part was to obtain the transformation between the

$$T_{R\_A} = \left[\begin{array}{c|c} Rot_y(-90°) * Rot_z(90°) & \begin{array}{c} offset\_x \\ offset\_y \\ offset\_z \end{array} \\ \hline \overrightarrow{0} & 1 \end{array}\right]$$

Table 2.1: The transformation from Apriltag to Reference frame

Apriltag frame and the reference frame. It is easily understood, that this is the most crucial part since it defines the desired coordinates towards which, we command the MAV to move to. As it can be seen from the 2.4 the reference frame has the same orientation as the world system and is translated from it by the offsets specified by the user in the parameters' file. The transformation between the apriltag frame with respect to the reference frame can be seen from the image 2.4, while the rotation is composed by a rotation around the y-axis by $-90°$ and a rotation around the z-axis by $90°$.The offsets, and especially the offset in the x-axis since this defines the straight distance between the MAV and the Tag, are added so as to ensure that the MAV will keep a safety distance between itself, the Apriltag and the user holding it. If the x offset is set to zero, then the reference frame coincides with the Tag's frame, and they will collide, since the reference position of the MAV is set to the exact equal coordinates of the Tag. The aforementioned are presented in matrix 2.1, of course T_{R_A} is a 4 x 4 homogeneous transformation presented in matrix 2.1. The final coordinates that are sent to the controller are the result of the transformation presented in equation 2.2 and they express in the coordinates at the reference frame with respect to the world frame.

$$T_{W\_R} = T_{W\_B} * T_{B\_C} * T_{C\_A} * {T_{R\_A}}^{-1} \tag{2.2}$$

2.3.2 Coordinate Frames in the Real System

In the real system the coordinate frames are as expected the same, with minor numerical differences. These numerical differences were expected since it is not possible to have a "perfect" recreation of the real system in the simulation.

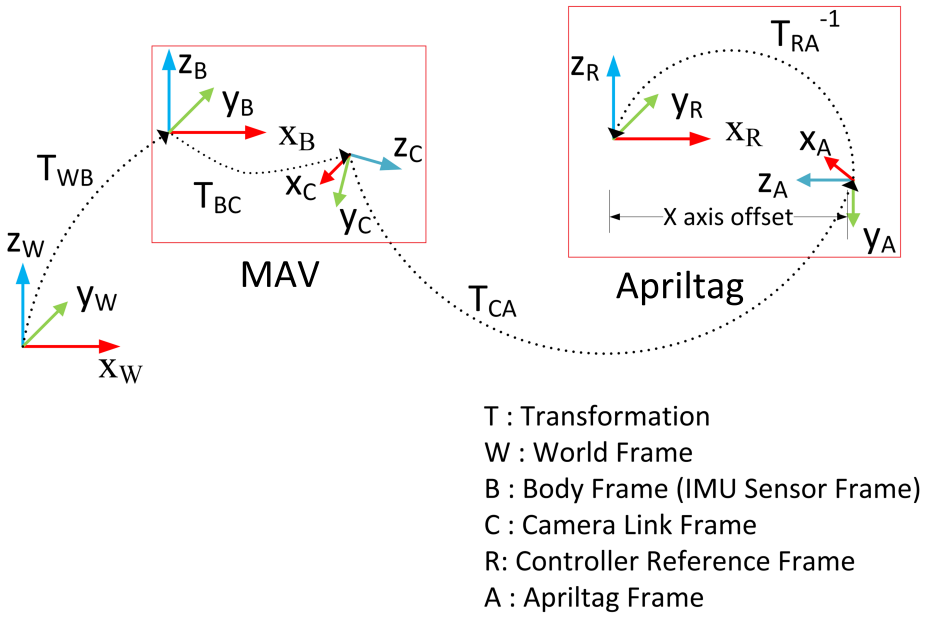

Figure 2.4: The MAV detects the Apriltags coordinates with respect to its camera frame. The appropriate transformations are made to transform the apriltag coordinates to the reference frame with respect to the world frame. In the controller reference frame the safety offsets are added so as to ensure that a collision between the MAV and the marker is avoided.

Chapter 3

System Setup

An important contribution of this thesis, was the incorporation of all system's parts in the simulation. By that, possibly dangerous or unpredicted situations in the real system were avoided by carefully studying the MAV's simulated responses to the experiments. At the beginning of the project, were created some ROS packages so as to obtain the necessary experience and get familiar with the system. All the following packages were written on a computer running on Ubuntu 14.04 LTS alongside with ROS Indigo Igloo[1]. The simulator used in this project is the RotorS simulator from ASL which is based on Gazebo[1].
Now follows the description of the ROS packages that were created for this project, some packets with very similar role and structure are omitted from the description. All the packages are written in the C++ programming language[16] and can be accessed at the ASL's GitHub page under the repository mav_demos[17].

3.1 Connecting the Camera with ROS

The very first task was to connect a camera to the ROS so as to obtain the images of the markers. The camera chosen for this project was the Logitech Tessar HD, for its calibration and image rectification the ROS camera_calibration package and the image _proc node were used respectively. After the camera setup, the detection of the Apriltags[2] came into focus. In order to detect the position and orientation of the aforementioned, the apriltag_ros package was used. This provides a ROS wrapper for the C++ library[18]. It should be mentioned that although this library detects position very accurately it faces some problems with the detection of the marker's orientation. The problem becomes worse as the distance between the camera and the target increases.

3.2 Moving the Simulated MAV

After the camera setup, the main focus moved towards connecting the Apriltag detection with the simulated MAV. This was conducted in the mav_demo_camera package. In there, two nodes are created, except of course those referring to camera, one that detects the Apriltag and publishes the detection data to a ROS topic and another one that conducts the motion. In this specific example the user moves the tag in front of the camera and the simulated MAV moves accordingly. It should be noted, that in this example the MAV does not detect the marker, its motion is based on the detection from the USB camera. As can be seen from figure 3.1 when

[1] http://wiki.ros.org/indigo

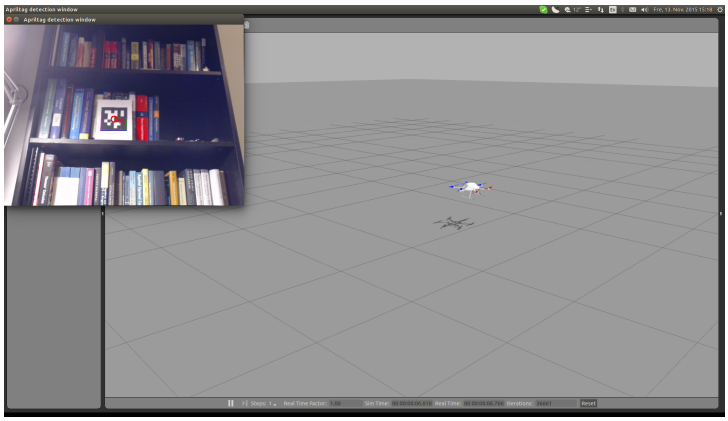

Figure 3.1: The Apriltag is detected in the real world, from the USB camera,as can be seen from the top right window, and its detected pose is set as the pose of the MAV in the simulated world. So the user can move the marker in the real world and control the pose of the MAV in the simulation.

the package is ran, two windows are created, one depicting the detections from the USB camera, for the user to know whether or not the marker is detected and the simulator's window with the moving MAV.
For safety reasons, the position of the MAV in the z axis is augmented by 50 cm. Thus, when the user holds the marker at the same height as the camera, the MAV does not collide with the ground.

3.3 Realistic Simulation Scenarios

Before moving to the real system, some more realistic simulation scenari were implemented so as to study the MAV's behavior under various circumstances. In this case the user inserts different markers inside the simulated world and the MAV performs a variety of motions with respect to the different detected markers.
The markers that are inserted into the simulation world were created so as to match the real world Apriltags. The created objects were simple cubiods which had as texture the Apriltag's image. It should be mentioned that the dimensions (width and height) of the cuboids should be identical as the parameters passed from the ROS parameter server to the detection node, so as to get the correct position estimation. Furthermore, the third dimension of the cuboid (length) should be kept minimal so as to avoid fallacious detections during the turning of the MAV. In order to distinguish between the various markers, we use the id of each tag. At each time only one Apriltag should be detected from the MAV's sensor so as to have a smooth operation.
The default motion, consists of the user moving a marker in front of the computer's camera which controls the pose of the Apriltag marker in the simulation. The MAV is programmed to follow the marker, by having adequate distance so as to guarantee the user's safety in the real world system. The predefined safety distance is set by the user at a configuration file that provides the system with various constant parameters. In the case that the program cannot find the configuration files for some reason, it sets the aforementioned values to predefined thresholds so as to guarantee the user's and system's safety.
Furthermore, the MAV is capable of performing take-offs and landings whenever the user wants. When the user wants to land the MAV, he just has to present

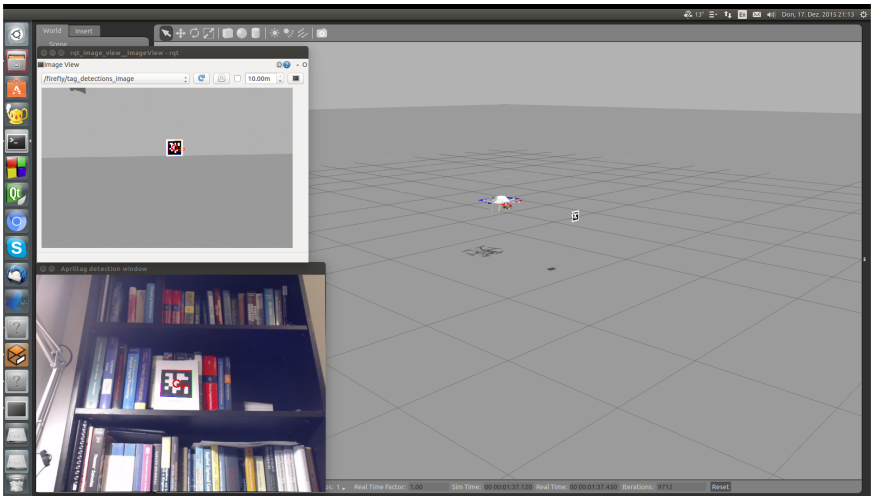

Figure 3.2: The algorithm detects the Apriltag in the real world from the USB camera, as is presented in the down right window, and projects its pose to the marker in the simulated world. The MAV detects, the Apriltag from its own camera into the simulation as can be seen in the upper right window, and moves based on this detection.

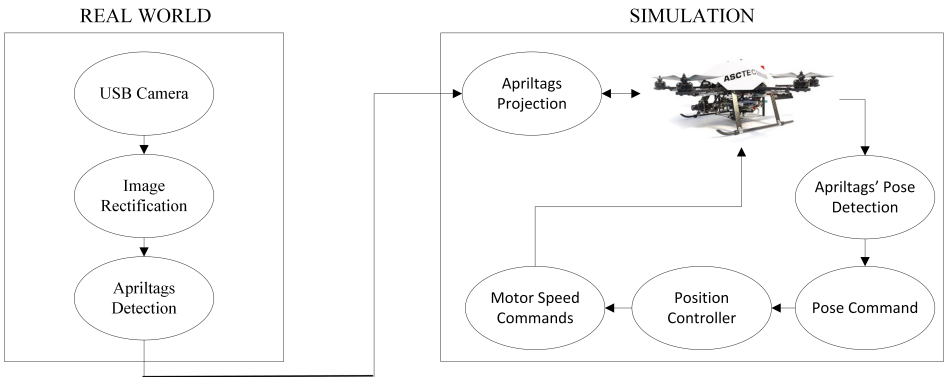

Figure 3.3: Approximation of nodes running in the simulation [3]. In the real world, the USB camera acquires images, after their rectification the Apriltag detection algorithm, detects the pose of the marker and based on this detection the pose of the simulated marker is controlled. The MAV runs the detection algorithm based on the images taken from its own camera and acquires the marker's pose. Based on this pose, it sends the desired position commands to the position controller.

a specific marker in front of the camera. The exact same procedure, but with a different marker, is followed to make the MAV hovering again. The program stores the last x,y and z coordinates along with the last orientation of The MAV before the landing command. It should be mentioned that in order to switch from landing state back to following the tag state the user must show to the MAV a specific marker that triggers a reset state. In this reset state, the MAV goes back to its starting position and height, just as before its landing. Furthermore, it is assumed that throughout the simulation the MAV's inertial measurement frame is assumed to be initialized relative to the ground.
A screenshot describing the operation of the MAV during the realistic scenari is provided at the figure 3.2. Here we can see, the detection of the Apriltag in the real world, at the lower left window, the marker as detected from the MAV's camera into the simulated world, in the upper left window, and finally the MAV and the Apriltag in the simulated world. Furthermore, in figure 3.3 there is a graphic approximation of the running nodes, which explains the underlying procedures. In other words, the marker is detected in the real world from the USB camera, and the detected pose commands the position and orientation of the simulated Apriltag. Then, the simulated MAV detects the marker from its own camera and sends the desired pose based on its detection to the position controller.

[3]The Firefly's image was taken from http://www.asctec.de/en

Chapter 4

Experimental Results

In order to evaluate the efficiency of the designed packages and also the Apriltag's detection algorithm, a series of experiments were conducted both in simulation and with the real system.

4.1 Experiments Conducted in Simulation

In order to evaluate the simulated system, the ability of the MAV to track and follow the Apriltag under various conditions was tested. As mentioned in 3.3 there are two offsets added in the robot's reference frame, these target to the security of the user, so as to avoid collision, and also to provide the MAV with a better field of view, thus improve the Apriltag detection rate.
We prove that the MAV always leaves an almost constant distance between itself and the detected marker. First the user presents the Apriltag that commands the MAV to hover. Then the marker that commands it to follow, after that the program automatically rotates this marker inside the simulation with a constant predefined turning rate. This is accomplished by changing the marker's yaw, while leaving the position unchanged. The tested rates are 0.5°, 2° and 5° per frame and since in simulation the USB camera records with 30 fps, the rates are of $15°/sec$, $60°/sec$ and $150°/sec$ respectively. In figure 4.1, only the x and y coordinates are presented. It can be seen that the MAV performs a circle while it follows the tag, thus it leaves a constant distance between itself and the marker. The lines that start from point (0,0) represent the Firefly's trajectory to its initial position relative to the tag. The spikes that appear in the $60°/sec$ and $150°/sec$ trajectories exist because by the time the MAV reached its initial position the tag had already rotated, thus the robot had to move greater distance. The fastest rotation the MAV can follow is about $240°/sec$, at greater rotation speeds the MAV is not able to follow the rotation. Conversely to what was expected, the Firefly has more oscillations in the z axis at the lowest turning rate, as can be seen from figure 4.2. That is normal since the MAV has to stay at the various positions before it receives the next measurable position value. Thus, the MAV has to stop and continue at all times in order to follow the Apriltag. On the other hand, when the tag is rotated faster the MAV always gets a new position command before it needs to stop and wait for a new one.

4.2 Experiments Conducted in the Real System

In order to validate our system, experiments with a real MAV were conducted. The name of the drone used is "AUK". First of all, during the experiments, it was observed that when the marker was moved at distances further than 2m away

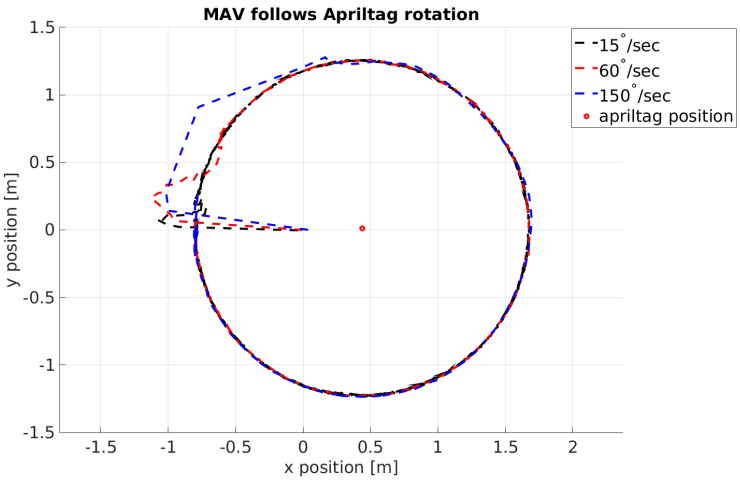

Figure 4.1: The user holds the Apriltag still and the algorithm automatically rotates it around z-axis. The MAV follows the rotating Apriltag and keeps a constant distance at all times. The spikes are caused because by the time the MAV reaches the position from the initial detection, the algorithm has already rotated the marker, thus the MAV "jumps" in order to cover that distance.

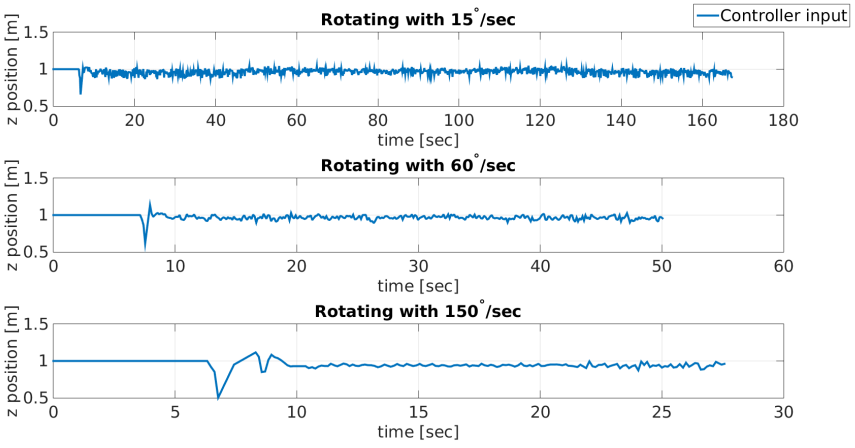

Figure 4.2: The controller input with respect to z-axis versus time during the three rotation experiments, that were conducted in simulation. The most oscillations are performed at the slowest rotation, while the least at the fastest.

Experiment	Detection Rate
1^{st} Experiment	90.86%
2^{nd} Experiment	96.73%
3^{rd} Experiment	96.3%
4^{th} Experiment	93.16%

Table 4.1: The Algorithms detection rates in the four conducted experiments

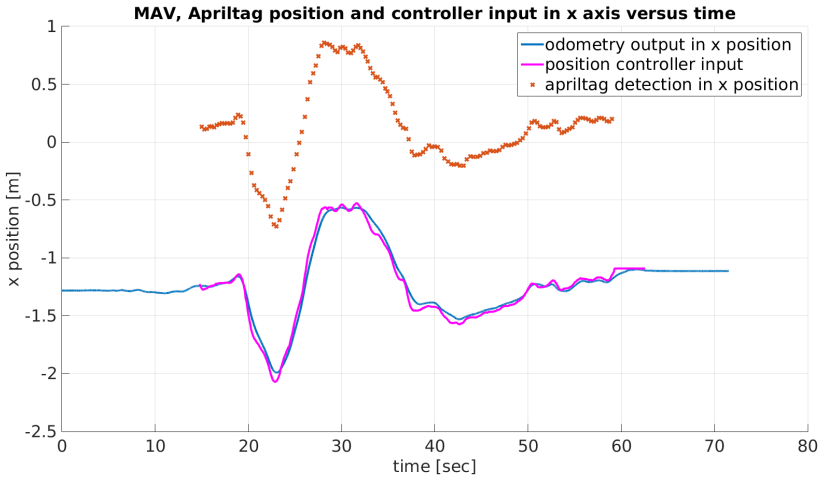

Figure 4.3: The real system's odometry output against the controller input and the apriltag detection position in x-axis versus time.

from the camera, then the orientation detection became unreliable and changed randomly. Although the algorithm could still measure distance accurately, it had some significant problems reading the correct orientation data from the Apriltag. That misdetection led to very abrupt random motions, since the coordinates sent to the controller use the detected pose of the marker in the transformation from the reference frame to the world frame. Thus, in order to avoid any dangerous situations, the orientation read from the Apriltag detection algorithm was substituted with a fixed rotation. Due to the aforementioned, the MAV could only follow the marker in straight lines since it did not track the Apriltag's yaw. Furthermore, during the experiments we experienced some difficulties with the onboard odometry system (ROVIO[1]), thus the VICON system was used to provide odometry data.
Four experiments were performed, in the first three the ability of the MAV to follow a "normally" moving target was tested, while in the fourth experiment the marker motions became faster and more abrupt. It should be mentioned, that during the first experiment we faced some difficulties due to problems with the MAV's trim. The overall Apriltags' detection rates based on the experimental results, are presented in table 4.1. In order to compute the detection rate, the number of frames during the manual phases (landing and take off) were excluded from the total number of frames. It can be seen that the algorithm, regardless the motion of the Apriltag achieved at least 90% detection at all cases. It should be mentioned, that due to the VICON system in the room the lighting conditions were not stable, since the camera sensor could detect the infrared flickering caused by its cameras and also, the onboard camera sensor had only a resolution of 480 x 752 pixels.

[1]https://github.com/ethz-asl/rovio

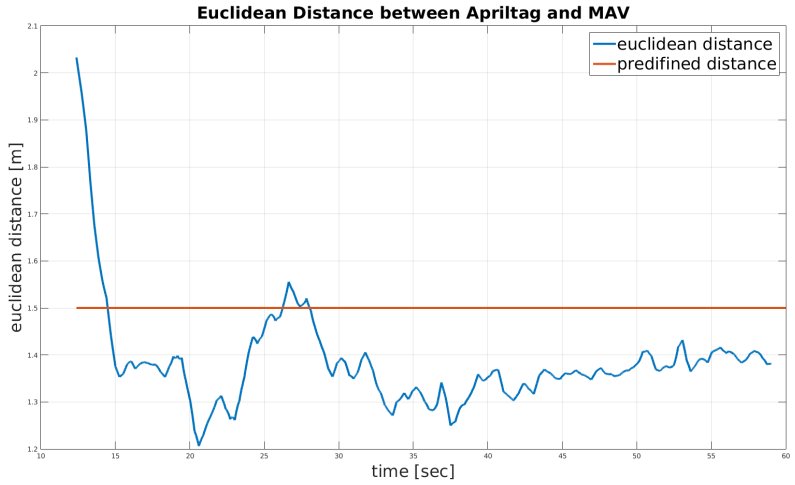

Figure 4.4: The euclidean distance between the MAV and the Apriltag at x axis, as resulted from the experiments in the real world.

Since the nature of the experiments is very much alike to each other, only one set of data will be presented. As can be seen from figure 4.3, the MAV manages to track the marker and follow it under all circumstances. The MAV, acts like a low pass filter on the controller inputs, thus it smoothens the aforementioned as it can be seen from the odometry output. Furthermore, we can observe from figure 4.4 the euclidean distance between the Auk and the marker with respect to x axis as it proceeds in time. Although the distance between the MAV and the user is, almost at all cases, less than the predefined safety distance, the MAV always leave adequate distance as it can be seen from the orevious figure. The reason for not always achieving the safety distance, is that the user moves the Apriltag before the MAV has reached that predefined distance, but even in this case, the average error from our desired safety distance is only 11.67 cm.
During the last experiment, the ability of the algorithm to track faster and more abrupt motions was tested. During this experiment, some discontinuities in the MAV's motion were observed, they were caused by the instantaneous loss of the Apriltag detection, since the user moved the marker fast from one position to another. During the time between the last observed marker position and until a new detected position was created from the detection algorithm, the MAV's controller was receiving as target position the last position where the marker was detected. Thus, this led to short sudden holds in the MAV's trajectory. During the analysis of the data, it was observed that the periods when the MAV could not detect the Apriltag varied between two to nine frames, or 66 to 300 msec. It should be mentioned, that the aforementioned cases refer only in situations where there is a loss of detection and not when the user moved the Apriltag out of the field of view of the MAV's camera. But even in the latter cases, the MAV was able to regain the marker detection since it had already started moving to its direction.

Chapter 5

Conclusion

In this thesis, we wanted to control a MAV through visual commands. In order to achieve this goal, we followed a step by step procedure. First every task was built and tested in simulation, by gradually building more and more realistic simulation scenari. We started from simply moving the MAV from the Apriltag pose as detected from the USB camera, then we created the marker models for the Gazebo simulator, these models were controlled by the pose of the real marker detections, and finally we controlled the pose of the MAV only by the Apriltag detections from its simulated camera sensor. After many experiments in simulation, we were ready to proceed to the real world, where we tested our algorithms on an AscTec's Firefly MAV.
From the experiments conducted, both in simulation and in the real world, we were able to validate both the ability of the algorithm to leave a sufficient distance between the marker and the MAV and also its ability to command the MAV to follow the marker under various motion scenari. Furthermore, high detection rates were achieved in all experiments.

5.1 Future Work

During this project, many ideas came up that could ot be implemented due to the pressing deadlines. First of all, instead of using Apriltags as visual markers, a gesture recognition algorithm could be implemented. With that the user would be able to control the MAV just by waving his or her hands, without having to hold any other markers. Furthermore, the algorithm could be modified so as to ignore any marker detections that are further than the specific distance that causes the orientation misdetections in the Apriltag detection algorithm. By doing the aforementioned and by also applying a low pass filtering in the desired position data that are derived from the detection algorithm, we could obtain much smoother paths and also make the MAV turn with respect to the detected yaw.

Bibliography

[1] Open Source Robotics Foundation, Gazebo Simulator http://gazebosim.org/.

[2] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2011, pp. 3400–3407.

[3] RotorS Simulator, http://wiki.ros.org/rotors _simulator.

[4] S. Lupashin, A. Schollig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 1642–1648.

[5] M. Hehn and R. D'Andrea, "Quadrocopter trajectory generation and control," in *Proceedings of the IFAC world congress*, 2011, pp. 1485–1491.

[6] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," in *Experimental Robotics*, ser. Springer Tracts in Advanced Robotics, O. Khatib, V. Kumar, and G. Sukhatme, Eds. Springer Berlin Heidelberg, 2014, vol. 79, pp. 361–373.

[7] Raffaello D'Andrea: The astounding athletic power of quadcopters, TEDGlobal.

[8] Lafaro Labs, http://wiki.lofarolabs.com/index.php/Localizing_with_AprilTags.

[9] RobornSwarm Carnegie Mellon, https://sites.google.com/site/mrsdproject201415teamc/.

[10] K. Ling, "Precision landing of a quadrotor uav on a moving target using low-cost sensors," University of Waterloo, Msc Thesis, Sep. 2014.

[11] S. M. Chaves, R. W. Wolcott, and R. M. Eustice, "NEEC research: Toward GPS-denied landing of unmanned aerial vehicles on ships at sea," *Naval Engineers Journal*, vol. 127, no. 1, pp. 23–35, 2015.

[12] R. W. P. Hoogervorst, "Collaborative position control of an uav using vision and imu data," University of Twente, BSc report 017RaM2015, Jul. 2015.

[13] J. Chudoba, M. Saska, T. Baca, and L. Preucil, "Localization and stabilization of micro aerial vehicles based on visual features tracking," in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, May 2014, pp. 611–616.

[14] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. Furgale, and R. Y. Siegwart, "A Synchronized Visual-Inertial Sensor System with FPGA Pre-Processing for Accurate Real-Time SLAM," in *IEEE International Conference on Robotics and Automation (ICRA), 2014 : May 31, 2014 - June 7, 2014, Hong Kong, China*. Piscataway, NJ: IEEE, 2014, pp. 431–437.

[15] Online repository, https://github.com/ethz-asl/minkindr/wiki/Common-Transformation-Conventions.

[16] B. Stourstrup, *The C++ Programming Language*, 4th ed. Boston, MA: Addison-Wesley, 2013.

[17] Online code repository, https://github.com/ethz-asl/mav_demos.

[18] Open Source Robotics Foundation, http://wiki.ros.org/apriltags_ros.