TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Elektrotechnik und Informationstechnik
Self-Organizing Systems Lab

Visually Localizing, Approaching and Carrying Objects using Magnetic MAVs

Bachelor-Thesis
Elektro- und Informationstechnik

Eingereicht von

Caio Victor Gouveia Freitas

am
18.11.2022

1. Gutachten: Prof. Dr. techn. Heinz Koeppl
2. Gutachten: M.Sc. Kai Cui

Erklärung zur Abschlussarbeit gemäß §22 Abs. 7 APB TU Darmstadt

Hiermit versichere ich, Caio Victor Gouveia Freitas, die vorliegende Bachelor-Thesisgemäß §22 Abs. 7 APB der TU Darmstadt ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Mir ist bekannt, dass im Falle eines Plagiats (§38 Abs.2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

English translation for information purposes only:

Thesis statement pursuant to §22 paragraph 7 of APB TU Darmstadt:
I herewith formally declare that I, Caio Victor Gouveia Freitas, have written the submitted thesis independently pursuant to §22 paragraph 7 of APB TU Darmstadt. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form. I am aware, that in case of an attempt at deception based on plagiarism (§38 Abs. 2 APB), the thesis would be graded with 5,0 and counted as one failed examination attempt. The thesis may only be repeated once.

Darmstadt, den 18.11.2022

_____

(Caio Victor Gouveia Freitas)

Abstract

Unmanned aerial vehicles (UAVs) have become increasingly popular in the last few decades. Among the applications, the use of UAVs for carrying objects by air stands out as an important industry trend, especially useful for critical time deliveries and deliveries in little accessible or hard-to-reach places, it presents itself as a more agile and flexible alternative to the traditional delivery systems. This thesis explores the idea of aerial manipulation in the context of object transport and proposes a drone-based delivery system. This means a final system in which MAVs are able to, given an approximate location of a predetermined container, visually search for the desired object, approach it visually as precisely as possible to enable magnetic coupling with the object and takeoff with it in order to bring it to a desired destination. The proposed system is tested in a simulated environment.

Contents

1 Introduction

Unmanned aerial vehicles (UAVs) have become increasingly popular in the last few decades. At first within the research community and at military applications, and more recently in the industry. Due to the possibility to transmit sensor data in real-time, sensing and inspection tasks have applications in the agribusiness, construction, preventive maintenance and mining industries that have become increasingly popular [1]. The freedom to position the vehicle in hard access or dangerous places makes it ideal for this type of application, offering the possibility to complete such risky tasks without exposing workers.

Another important branch of automation with UAVs is object manipulation applications, combining the versatility and agility of drones with the ability to interact with the environment and manipulate objects. Here, UAV-based delivery systems play an important role. This sort of system has been extensively explored and has the tendency of becoming increasingly more popular in the near future [2], thanks to the advantages of enabling a significantly faster delivery time, and the capability of reaching hard access locations. Thus, it is ideal for time-critical and/or urgent delivery applications. This has been exploited extensively by the industry, e.g. with a large investment by the world's largest delivery company at last-mile delivery with MAV's [3].

Such manipulation tasks present several challenges in their implementation. Starting with the precise positioning of the drone in relation to the object in order to interact with it. Another challenge lies in the coupling between the drone and the object, where a robotic manipulator usually comes into focus. Finally, there is also the challenge of controlled take-off and flight in possession of the object.

The focus of this thesis is on developing a vision-based system that controls the drone until it is coupled with an object (a visual marker) accurately. For this, the concept of visual servoing is explored. In addition, other relevant tasks are also considered. Such as the overall architecture of the final system (described by a state machine, which safely handles a range of situations that may occur) and the controlled flight of the drone coupled to an object is also simulated with the same configuration, such that an end-to-end pick-up and delivery system is possible.

This thesis is organized as follows: Chapter 2 formally introduces the foundational concepts that are used in the development of the solution. Chapter 3 presents the frameworks used to develop the solution and run the simulations and the hardware that is simulated. Chapter 4, 5 and 6 present the methods used to localize the target objects, approach the object and fly with it to the new desired location, respectively. Finally, the work is concluded in Chapter 7.

1.1 Related Work

This thesis is based on a lot of previous research on fiducial marker detection, visual servoing and aerial gripping. This section introduces some of the previous work in those fields that were used as a reference for this thesis.

Visual servoing as a robotics control scheme has a significant background story [4]. Since it was first proposed in 1979 by Hill [5], multiple approaches to the problem were developed and real-time and reliable implementations were significantly improved [4, 6, 7], but still are considered a major issue. The problem can be categorized according to the given system, considering the number and positioning of the cameras i.e. if they are part of the robot or external. Given the system, another option can be made on which type of visual features will be used and whether they will represent some 3D position or features in the image plane.

The design of a control scheme for a dynamic system based on computer vision data has been successfully applied to UAVs since the early 2000s [8, 9], with autonomous helicopters and in the work from [10], an external camera is used to estimate the orientation of the quadrotor and **stabilize** it. Later in 2009 works like [11] presented position-based visual servoing in quadrotors, using a pinhole camera model to estimate position. More recently, [12] achieved a successful aerial picking of steady and moving magnetic objects with quadrotors in indoor environments, making use of a position-based visual servo controller, calculating the relative localization of the MAV's frame to the object, and controlling the x, y coordinates to the desired point. Here the z-direction is treated separately with the estimated height value for the drone, after descending to a desired height, if the object is still close enough to the vehicle, an approach maneuver is initiated. A magnetic gripper is developed so that the magnetic object can be released by the MAV.

The work of [13] is an example of how image-based visual servoing (IBVS) can be applied to control the MAV's attitude using a dynamic model and simplifying the classic point IBVS interaction matrix [14, 6] according to the available degrees of freedom. Applied to a system of an interceptor drone, the central coordinates of the interceptor in the image (given by a properly trained CNN) are used so the drone is aligned with it, while an uncoupled controller takes care of the approaching speed towards an "intruder drone".

An autonomous indoor gripping using a quadrotor was implemented by [15], using a mechanical gripper and an infrared marker (which required an infrared camera). With the indoor MAV localization navigation available through a visual monocular SLAM algorithm and the object approaching strategy was similar to the one implemented in this project - using the offsets in the image coordinates and the area of infrared blob as control variables.

While in the work of [15, 16, 17] single-DOF mechanical grippers are used, in [12] an electropermanent magnetic gripper is introduced, with the ability to pick up and release magnetic objects of varying shapes and weights. In [18] the use of ingressive grippers is explored, which have a very similar practical effect to the simpler magnetic coupling considered for this project. Both require contact between the drone and the gripper and are equivalent to a fixed joint between the drone and the object.

(a) Container (box) with visual marker on top, simulated in the Gazebo world from Figure 3.2

(b) Iris 3DR MAV on gazebo, on top of the container after execution of the precision landing

Figure 1.1: Pictures from the simulation environment on Gazebo

1.2 Proposed System and Applications

This thesis explores the idea of visual-based guidance of micro-air vehicles for object approach, as well as aerial manipulation in the context of object carrying, and proposes an end-to-end drone delivery system. That means a final system in which MAVs equipped with magnets are able to, given an approximate location of a predetermined container, visually search for the desired object, approach it visually as precisely as possible to enable magnetic coupling with the object and takeoff with it in order to bring it to the desired destination. The approximate location of the object can be given to the system through a GPS system integrated into the container's hardware or given by the user - e.g. through another device or picked roughly on a map. The container is a box with a visual marker on top, and magnets that enable coupling with the drone's base, as illustrated in Figure 1.1a.

In this work, the uncoupling of the object after landing at its destination is not considered. That means there is the supposition that the object is removed by an operator (or user), which could be solved with the use of a magnetic gripper like the one from [12].

In general, such a system could be implemented in the industry as an end-to-end delivery system, with autonomous pick-up and drop-off of lightweight high-value items (such as medicines, vaccines, etc.) in predetermined packages; and has the advantage of being capable of reaching hard-access places via air, while also enabling a faster pick-up and delivery. The proposed system is tested in a simulated environment with and without wind conditions.

This work assimilates the closest with the one from [13], which also uses IBVS to MAV, but handles a vertical approach toward the object instead of a horizontal one; also, in this thesis both the approach and the centering problems are considered in the same controller design, by introducing more visual features that fully describe the desired parameters to be controlled in the interaction matrix. Also, it compares to [12] by the system architecture as a state machine and vertical approach, but a different control strategy is used (position-based instead of image-

based) and an electropermanent magnetic gripper attached to the bottom of the MAV, which is not covered on this project.

2 Foundations & Task Definition

As stated in Section 1.2, the goal of this project is to achieve a final system in which objects are visually located and manipulated (essentially transported) by small UAVs with magnets. This goal can be broken down into 3 main problems. Searching and visually localizing the objects, approaching the object and attaching it to the MAV, and at last flying the drones in a controlled manner with the load (object) to its goal destination.

This chapter provides an overview of the fundamental concepts and building blocks used throughout the thesis for the solution.

2.1 Visual Fiducial Markers and Detection

Fiducial markers are artificial visual features, designed to be easily detectable and distinguishable [19] in the middle of an environment. Mostly famous for their applications to augmented reality, this kind of marker is also extensively used in the field of robotics for being able to provide the position and orientation of a tag relative to the camera. Among the different types of fiducial markers, AprilTag was created to be robustly detectable despite occlusion, warping, and other disturbances, and hence is the visual feature on the container used in the simulations [19, 20]. An example of AprilTags and its detection is presented in Figure 2.1.

The detection algorithm used is the one from [20]. The second implementation of AprilTag that presented dramatic improvements in detection speed and fewer false positives.

2.2 Object Search

As described in Section 1.2, for the scope of this project, the approximate position of the desired object in local or global coordinates are given. However, this does not necessarily mean that the object (the visual marker) is within the field of view (FOV) of the drone's camera from that given location. Therefore, a search algorithm might be necessary to get the first visual detection of the visual marker to be able to initiate the visual approach.

To avoid the need to treat collision avoidance, once at the desired location, the first step in the proposed search routine is to increase the height of the MAV until slightly below the limited distance of identification of the visual markers, which also allows the camera to cover a bigger

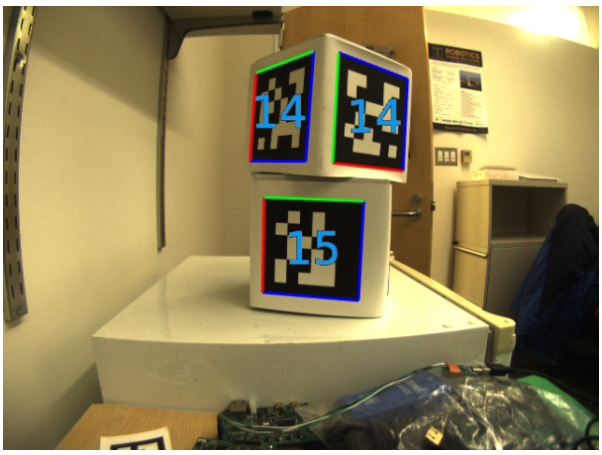

Figure 2.1: Apriltag detection example: annotation of multiple visual markers being detected by the detector from [20].

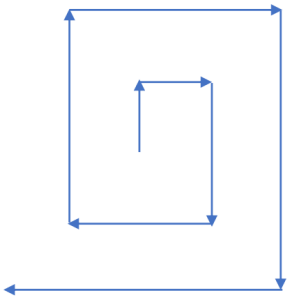

Figure 2.2: Coverage trajectory based on a square flight pattern [24]

area, making the search more effective and safer.

Supposing the existence of a "safe altitude" where no obstacles are expected is a common practice in the drone industry and research, usually applied as a failsafe policy [21, 22, 23].

After that, a coverage trajectory is performed to explore the surroundings of the given point. Here, a square of dimension 5 m is taken as the area of interest, and a square flight pattern (growing square spiral, as shown in Figure 2.2) is performed, which is appropriate for this case, since uniform coverage of the area is desired, without information about the likely target meeting point [24].

2.3 Object Approach

Once the object is first detected, that means that it is in the field of view (FOV) of the drone's downward-facing camera. This leads to the task of visually guiding it until it is coupled with the object. Essentially a visual servoing problem: using the data provided by one or more cameras

to control the motion of a robotic system [25]. There are a couple of strategies to solve this according to [4], i.e. position-based, image-based, and 2.5D vision servoing. The first relies on position estimation to perform the control, having the advantage of enabling the independence of position estimation and control problems. The second realizes an online comparison between the measured image signal and target image coordinates, being more robust to deviations on the camera model and noise, but requiring an estimation of the image Jacobian that contains depth information (usually unknown). The latter enables a combination of the image signal and the position signal to generate a synthetic error for feedback.

2.3.1 Visual Servoing

According to [14], the visual servoing task can be described as regulation to error of the error

$$\mathbf{e}(t) = \mathbf{s}(\mathbf{m}(\mathbf{r}(t)), \mathbf{a}) - \mathbf{s}^*(t) \tag{2.1}$$

with $\mathbf{m}(\mathbf{r}(t))$ being a set of image measurements (that depend on the camera position $\mathbf{r}(t)$), $\mathbf{a}$ external knowledge about the system (usually camera parameters, object models, etc.) and finally $\mathbf{s}$ is the vector of features to be regulated upon. The types of visual servoing differ on the design of the vector of visual features $\mathbf{s}$. When those are expressing 2D parameters directly on the image, then we have image-based visual servoing (IBVS). If the elements of $\mathbf{s}$ describe some 3D parameter related to the position between camera and target, then position-based visual servoing (PBVS) takes place.

The general dynamic equation for such a set-up is

$$\dot{\mathbf{s}} = J_s\dot{\mathbf{q}} + \frac{\partial \mathbf{s}}{\partial t} \implies \dot{\mathbf{s}} = J_s\dot{\mathbf{q}}. \tag{2.2}$$

The term $\frac{\partial \mathbf{s}}{\partial t}$ describes the variation of $\mathbf{s}$ in the case where the object is moving and can be dismissed for the scope of this project since the package is assumed to be steady.

The Jacobian J_s is derived from the robot's Jacobian $J(\mathbf{q})$, i.e. the matrix that relates variations in the configuration $\dot{\mathbf{q}}(t)$ with the absolute linear and angular velocity of the end effector ${}^0\mathbf{v}_n(t) = {}^0J_n\dot{\mathbf{q}}(t)$, alongside with the transformation between camera and end effector ${}^cT_b$ (in this case the base link of the drone) and the so-called interaction matrix L_s through the equation

$$J_s = L_s\ {}^cT_b\ J(\mathbf{q}). \tag{2.3}$$

While the interaction matrix L_s is defined by

$$\dot{\mathbf{s}} = L_s\ {}^c\mathbf{v}. \tag{2.4}$$

From these relations, a classical control scheme is to use the form

$$\dot{\mathbf{q}} = -\lambda J_s^+(\mathbf{s} - \mathbf{s}^*) + J_s^+\frac{\partial \mathbf{s}^*}{\partial t}, \tag{2.5}$$

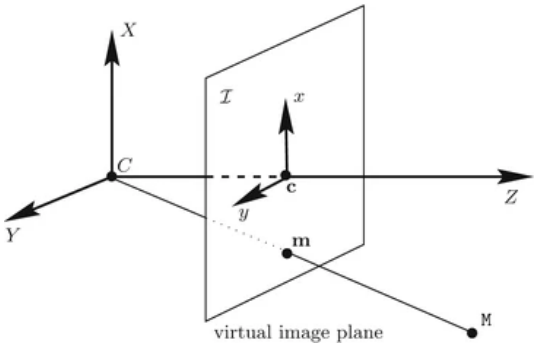

Figure 2.3: Perspective projection model geometry, relating a point M in the 3D camera frame with it's projection onto the virtual image plane m [27]

where J_s^+ is the Moore-Penrose pseudo-inverse of the jacobian J_s and the term $\frac{\partial \mathbf{s}^*}{\partial t}$ represents the variation of the setpoint $\mathbf{s}^*$. Such visual servoing control schemes can be demonstrated to be locally asymptotically stable, if the errors in the estimated J_s are little significant [26].

Image Based Visual Servoing (IBVS)

As stated by [4], this method provides both greater robustness of the system to camera model deviations and noise, as well a much smaller computational cost than the position-based one, but requires the estimation of the image Jacobian matrix, which contains depth information usually solved by depth estimation. IBVS method can only be used when the robot's initial position and attitude are near the target, which is already ensured by the object search phase.

The interaction matrix for IBVS can be derived by reducing the camera to a perspective projection model (also known as pinhole camera model), illustrated in Figure 2.3. This model is derived by considering the focal aperture as punctual and disregarding the use of lenses to focus light. As a first-order approximation, some effects are not considered by this model, such as camera distortions [27] and are described by

$$\begin{pmatrix} u_x - o_x \\ u_y - o_y \end{pmatrix} = \begin{pmatrix} x^i \\ y^i \end{pmatrix} = \frac{f}{Z} \begin{pmatrix} {}^c x \\ {}^c y \end{pmatrix}, \tag{2.6}$$

where f is the focal length of the camera, Z is the distance between the point and the image plane, x^i, y^i are the coordinates in the camera's image frame relative to the image center o_x, o_y. u_x, u_y are the pixel coordinates of the point.

With this model, considering a punctual visual feature, [6] shows that the interaction matrix that relates the derivative of the visual features with the velocity in the camera frame (as in Equation 2.4) can be derived through

$$\begin{cases} \dot{u}_x = \frac{\dot{u_x}}{Z} - x\frac{\dot{Z}}{Z^2} \\ \dot{u}_y = \frac{\dot{u_y}}{Z} - y\frac{\dot{Z}}{Z^2} \end{cases}, \tag{2.7}$$

where the first terms correspond to the diagonal values in the interaction matrix, while the second terms represent interactions between other velocities or rotations on the point's velocity. After relating the velocities of the 3D point to the camera spatial velocity [26, 25], the final form for the interaction matrix of a point in the image is

$$\dot{\mathbf{s}} = \begin{pmatrix} \dot{u}_x \\ \dot{u}_y \end{pmatrix} = L_s^c \mathbf{v} = \begin{bmatrix} -f/Z & 0 & x^i/Z & x^i y^i/f & -(f^2+(x^i)^2)/f & y^i \\ 0 & -f/Z & y^i/Z & (f^2+(y^i)^2)/f & -x^i y^i/f & -x^i \end{bmatrix} \begin{pmatrix} {}^c v_x \\ {}^c v_y \\ {}^c v_z \\ {}^c w_x \\ {}^c w_y \\ {}^c w_z \end{pmatrix}. \tag{2.8}$$

Here, the parameter Z is usually estimated, but in the scope of this project, Z corresponds to the vertical distance from the drone to the object below. Since the relative height is essential data for MAVs, it is usually already available through a distance sensor or other estimation methods to provide better flight stability. That means the estimation of the vertical distance to the object can be made through the height data from the drone as $Z = {}^c z$. (An alternative would be to use the desired Z^* as an estimate).

Position Based Visual Servoing (PBVS)

In this strategy, the 3D position of the camera frame is estimated and used as an input to a position controller, that can be set to guide the vehicle until the goal. The advantage of this method is that it enables the use of existing mature control methods directly by separating the visual processing from the robot control. This is an old problem in the literature, known as the perspective n-point problem (PnP). It consists of finding the position and orientation of a camera with respect to a scene object from n correspondence points (known points of a reference object) [28, 29].

However, it presents limitations, such as the closed-loop performance dependency on the accuracy and overhead of the position estimation algorithm. The accuracy of the position information might depend on the accuracy of the parameters (camera and object), and the image signal is independent of the control. This means that it cannot be guaranteed that the reference object will be contained in the camera's field of view (FOV). In order to mitigate some of those limitations, some techniques like the Smith predictor can be used to cancel the delay from the position estimation in the control loop, and very precise calibration of the camera parameters and a good number of reference points helps reduce the position estimation error.

Relative Camera Position Estimation

Between the available solutions to the relative position estimation through the perspective n point (PnP) problem, the solutions could be divided between those with $3 \leq n \leq 5$ and $n \geq 6$, the former usually involves nonlinear problems and the latter linear [30]. In addition, a

division can be observed between iterative and non-iterative methods, as in [31]. The iterative try to optimize an objective function (usually based on geometric errors), having the advantage of being more precise to the cost of computational load and the possibility of local minima solutions.

Visp autotracker [32] is an ROS package that implements a position estimation based on QR codes, flash codes or AprilTags. Using the corners of the visual marker to compute the 3D position of the object relative to the camera using a PnP algorithm, the tracker can also detect loss of tracking and recover from it.

2.4 Object Carrying

In order to manipulate the object with the use of magnets, which according to [33] constitutes a Flying Hand (FH) class Aerial Manipulation problem, a number of problems that might arise have to be taken into account. At first, it is intuitive that the coupling between the object and the UAV during flight can represent several problems in vehicle flight stability due to unstable vehicle dynamics and coupled object-aircraft motion. Those problems were addressed in multiple works, like the one from [16], where is demonstrated that dynamic load disturbances that are inside a certain bound of mass-inertia parameter changes are rejected by a helicopter with PID position controller without adjustments to the parameters to maintain stability [17].

In the work from [18], studies about robustness and estimation of inertial parameters in flight are introduced in a quadcopter with a grasping mechanism similar to the one supposed in this project (a mechanical gripper with 1 DOF). This implementation allows the MAV to carry an object with unknown mass and moments of inertia by estimating the parameters of the system as a whole (drone and payload) by applying least-squares methods to the quadrotor modeling equations and updating the controller accordingly.

3 Frameworks and Hardware

This chapter briefly presents the hardware used in the simulations, as well as some considerations regarding a feasible physical implementation of the system.

3.1 Simulation, Environment and Tools

This section presents the frameworks used for the development of the solution. That includes the flight control software, simulation tool, development environment, communication tools among others.

3.1.1 Robot Operating System (ROS)

ROS is an open-source development framework and set of software libraries for robotics applications. It is widely used in the global community and provides several high-quality tools, libraries, and capabilities for robot development, allowing the integration of standard modules with custom software in order to provide a more reliable and time-efficient solution [34].

In this project, the development was modularized into 3 main nodes: the state machine, the highest level of abstraction that handles state transition and their implementation, the Apriltag detector node, and the vision-based controller. In addition to that, the MAVROS package also instantiates a node that is responsible for interfacing the flight controller with the other ROS nodes. The communication between the nodes is mostly given through ROS topics, but also ROS services are used e.g. for the flight mode switching and arm/disarm commands.

The ROS nodes used for the control run in a Linux environment and can be either embedded to the MAV (for onboard operation), e.g. with the use of a single-board computer, or executed offboard in a companion computer, communicating via telemetry. The first option provides bigger ratios (no telemetry delays etc., more similar to the simulated results) and would be the ideal case for a physical implementation.

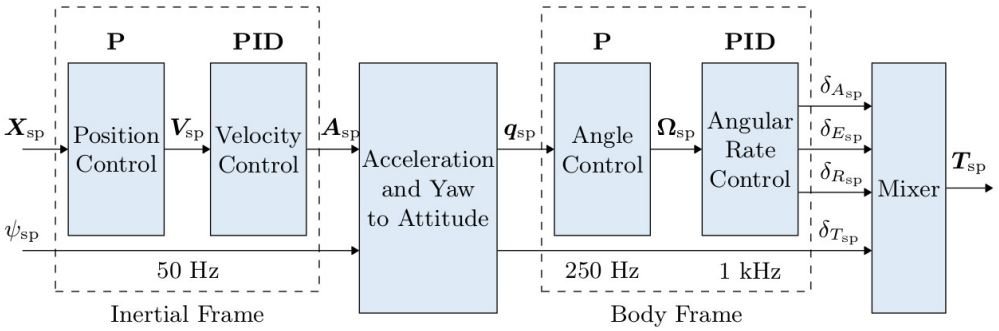

Figure 3.1: Multicopter control architecture implemented in the PX4 firmware [37]

3.1.2 PX4 Flightstack

When looking into the drone industry standards, the PX4 autopilot software stack stands out as an open standard [35]. Since the original Pixhawk project [36] by Lorenz Meier at the ETH Zurich, it has grown to be today's most used standards for flight control hardware and autopilot software in the drone industry and has made available so-called open standards for the industry, such as telemetry communication protocols (MAVLink and UAVCAN), and hardware standards (Pixhawk), providing open guidelines for drone systems development.

The PX4 flight stack provides a robust cascade PID controller (Figure 3.1), that allows to send setpoints at various levels of the control stack, such as positions, velocities, accelerations, actuator controls or mixed setpoints like trajectories (setting position and velocities for a point). That gives the flexibility to choose the level of controller design for each phase - e.g. the MAV is controlled via position setpoints until the target location and via velocity setpoints once the visual control is initiated.

This MAV used for the simulations is the 3DR Iris quadcopter [38] (introduced in Chapter 3), which has a PX4 compatible flight controller (based on the Pixhawk FCU), using the MAVROS package to interface the simulated drone with the ROS nodes developed (according to the PX4 guidelines). The development environment used was Ubuntu 20.04 with ROS Noetic, the simulation environment was built on Gazebo 11.

3.1.3 Gazebo

Gazebo is an open-source robotics simulator, capable of simulating an extensive set of actuators and sensors with noise models [39]; it is also integrated with the ODE Physics Engine and OpenGL rendering. It is widely used in state-of-the-art robotics applications, such as NASA's Space Robotics Challenge [40].

In the scope of this project, Gazebo provides both accurate dynamics as also supports inputs from the drone's PX4 firmware running on simulation time, as well as provides the data from

Figure 3.2: World used for running simulations on Gazebo

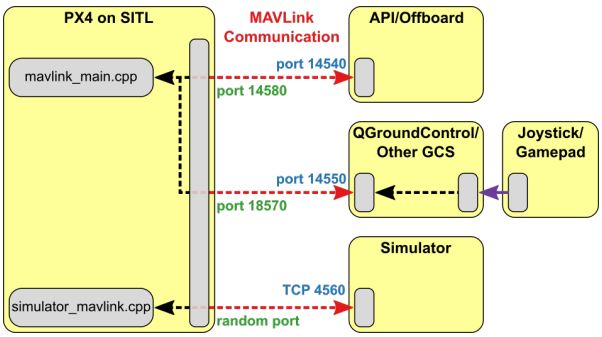

Figure 3.3: Diagram of the communication interfaces between the simulated PX4 Firmware with the API (ROS) and simulator (Gazebo), reprinted from [41]

the simulated world to it.

The created simulation environment attempts to illustrate one possible use case of such a system, for a residential pick-up and delivery between end-costumers. More specifically, the package with the visual marker is placed outside a house, on a grassy ground, to be picked up by the MAV - as shown in Figure 3.2.

The PX4 firmware receives Gazebo sensor data and sends actuator values, and ROS (the Offboard API) receives telemetry data from the simulated environment (MAVLink messages through the MAVROS package) and sends commands to the control architecture as illustrated in Figure 3.3.

3.1.4 System Architecture

The overall system is implemented as a state machine using the SMACH ROS library, which enables the modular development of the complete mission. The structure of the overall state machine is presented in Figure 3.4.

In the state ”GoToGoalPose”, the vehicle takes off to a determinate height and follows a linear

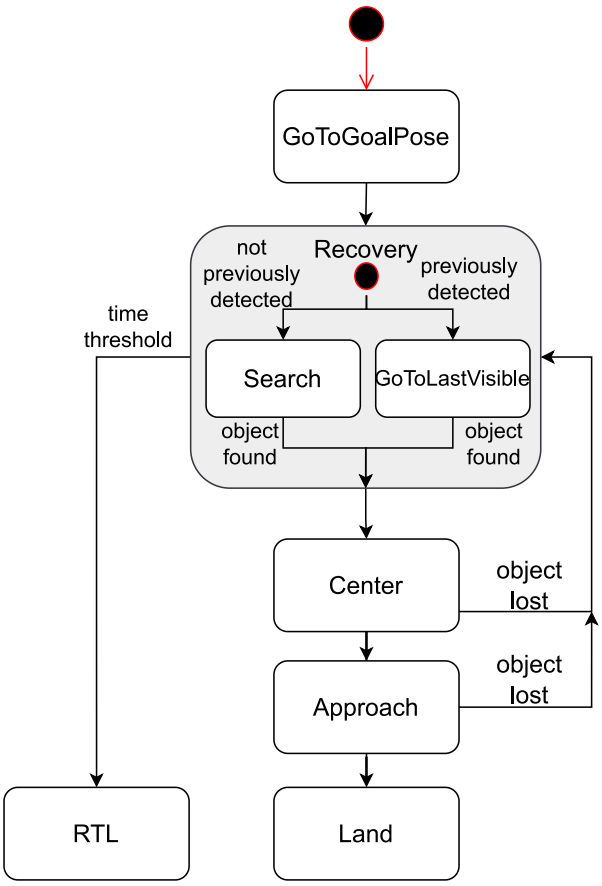

Figure 3.4: State machine for operation of the vehicle from takeoff until object picking

trajectory until a given location. Once the desired position is reached if the visual marker was not yet detected, the ”Search” state is triggered, and the procedure described in Chapter 4 is executed - here if no object is detected during the whole search trajectory or the attempt of approach is taking a time longer than a certain threshold, the MAV returns to its origin with the state ”RTL” (return to launch). As soon as the object is found, the state ”Center” comes in place in order to maintain the MAV's altitude and center the object in the desired configuration in the image. When the object's pixel coordinates and orientation are sufficiently close to the desired ones, the ”Approach” state is initiated, in order to approximate the MAV to the object until a certain threshold. When the object is close enough (and sufficiently centered), the state Land of the MAV is executed. This maintains its horizontal position and applies a vertical approach velocity towards the object.

During the execution of both the ”Center” and ”Approach” states, if visual contact with the object is lost, the MAV enters recovery mode and goes to the position where the object was last visible (State ”GoToLastVisible”).

The resulting state machine for the object approach has a similar outcome to the one used in the work from [12], but without the assumption that the object is in the MAV's FOV.

Figure 3.5: 3DR Iris quadcopter drone

3.2 Hardware - 3DR Iris

This MAV used for the simulations is the 3DR Iris quadcopter [38] (in Figure 3.5), which has a PX4 compatible flight controller (based on the Pixhawk FCU), that performs the MAV's state estimation and control according to the scheme in Figure 3.1. The FCU also allows serial communication with an embedded "companion" computer through the MAVLink protocol. The vehicle is also equipped with an inertial measurement unit (IMU), GPS, magnetometer, barometer, and a telemetry radio that allows communication with a ground computer. The motor-to-motor dimension is 550 m, the height is 100 m, and the MAV weighs in total 1.282 kg (with the battery).

It has a payload capacity of 400 g and an estimated flight time of 10-15 minutes, given the default 3-cell 11.1 V 3.5 A h battery. The relative height of this MAV in the default configuration is calculated through an Extended Kalman Filter (EKF) in the PX4 firmware, fusing the data from the IMU and barometer, and can be extended with the use of a distance sensor such as a lidar or ultrasound facing down.

The approach taken can be significantly generalized to other vehicles. The software implementation for the Iris drone is compatible with any MAV with a down-facing camera that accepts velocity commands as input via ROS or MAVLink messages - including but not limited to, all PX4-compatible vehicles.

4 Object Localization

In order to search for the object whose approximate position was given, the first step taken is to achieve the height equivalent to the maximal detection distance, i.e. the maximal distance in which the visual markers can be safely identified, that should both providing the maximal coverage area and a higher altitude operation, which in common outdoor contexts implies in the absence of obstacles. After that, the MAV initiates the execution of a square flight pattern for covering the area around the given location.

The simulated camera has a resolution of 800x800 and parameters and the specifications in Table 4.1.

Where f is the focal length and c_x, c_y are the coordinates of the principal point of projection. The camera image, converted to greyscale, can be used as an input to the Python implementation [42] of the Wang apiltag detection algorithm [20], which provides a robust and efficient detection. One example of the camera simulated image in Gazebo with the identification outputs is presented in Figure 4.1.

In Figure 4.2, the maximal distance of coverage for the simulated camera is estimated. That was experimented by placing the MAV above the visual marker on the ground and raising it while recording the detections. The height at which the detections start to break down (given the rate of correctly detected markers) is the desired distance. This experiment is presented in Figure 4.2.

The visual marker of size 30 cm x 30 cm can be identified up to a height of 15 m. The distance used in practice is 10% smaller (13.5 m), in order to provide greater safety in detection.

With this information, which depends on the camera's internal parameters and the size and type of marker, as well as the detection algorithm, the search algorithm can be executed as follows. While the object is not found, the drone rises until the maximal identification height, then the execution of a square flight pattern is initiated until the object is detected or the trajectory scope is ended. If no marker is detected during the scope of the trajectory, the state becomes return to launch (RTL) and the MAV returns to its origin position.

Camera Parameters	Value
Pixel array	800 x 800
f	692.978391
c_x	400.5
c_y	400.5
Frame rate	30 FPS
Field of view (FOV)	59,98º

Table 4.1: Simulated Camera Parameters

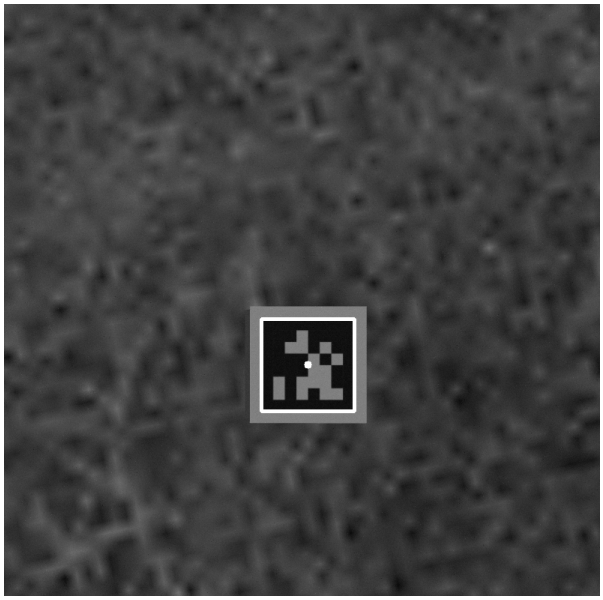

Figure 4.1: Camera image from the simulator with annotations of marker's center and corners from the AprilTag detector.

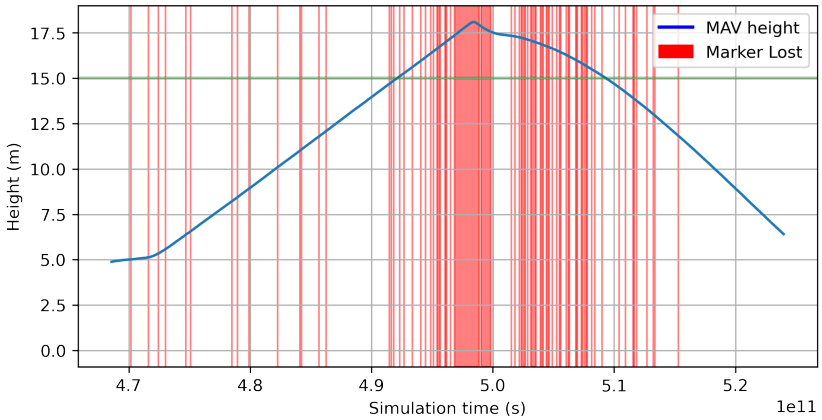

Figure 4.2: Range of detection experiment - MAV's height and regions in which tracker of the marker is lost. In green, an approximate maximal detection distance.

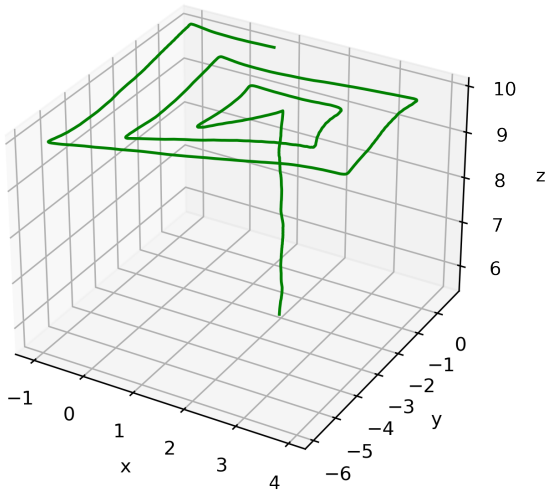

Figure 4.3: MAV's 3D absolute position while performing the object square search trajectory around the given point.

5 Object Approach

Here, the goal is to control the drone based on its velocities (using the rest of the cascade PIDs provided by the PX4 firmware in Figure 3.1). As the angular velocities in the x and y angles cannot be set without affecting the linear velocities (given the underactuated nature of the vehicle), the input for the designed controller is

$$\dot{q} = \begin{pmatrix} \mathbf{V}_s p \\ \phi_s p \end{pmatrix} = \begin{pmatrix} v_x^b \\ v_y^b \\ v_z^b \\ w_z^b \end{pmatrix} \tag{5.1}$$

The camera is facing downward on the MAV as in Figure 5.1. That means that the rotation and transformation matrix between the camera link in relation to the drone's base link is given by

$${}^bR_c = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{5.2}$$

$${}^bT_c = \begin{bmatrix} 0 & -1 & 0 & d_x \\ -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

The distance d_x is 15 cm, the offset between the camera and the drone base link. That means the resulting Jacobian matrix and control scheme with a constant setpoint s^* is

$$J_s = L_s \; {}^cT_b \; J(q) \tag{5.4}$$

And the control scheme according to Equation (2.5) is

$$\dot{q} = -\lambda J_s^+ (\mathbf{s} - \mathbf{s}^*) \tag{5.5}$$

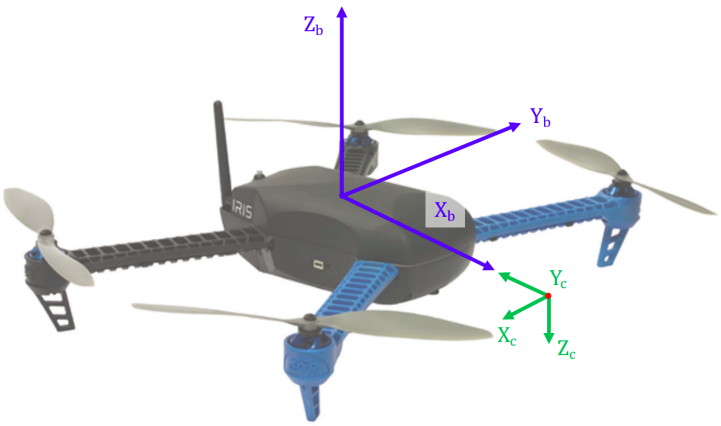

Figure 5.1: MAV's base reference frame and camera reference frame

5.1 Image-based Visual Servoing

For the scope of this project, the configuration $\mathbf{q}$ is the MAV's position and yaw orientation, and the input variable will be set to the velocity of the base frame itself, which implies that $J(q)$ is the identity matrix and

$$\dot{q}(t) = {}^{b}\mathbf{v}(t) = \begin{pmatrix} {}^{b}v_x \\ {}^{b}v_y \\ {}^{b}v_z \\ w_z \end{pmatrix}. \tag{5.6}$$

These velocity inputs are saturated at the output of the controllers to have a smaller or equal value than the predefined limits. Those are for the horizontal velocity 1.41 m/s 0.2 m/s for the vertical velocity and 1.41 rad/s for the angular velocity of the yaw.

In order to control the MAV by IBVS the desired 2D visual features need to be chosen. The ones used for this project are

$$\mathbf{s} = \begin{bmatrix} \hat{u}_x \\ \hat{u}_y \\ u_A \\ u_\phi \end{bmatrix}, \tag{5.7}$$

where u_x and u_y represent the x and y positions (in image coordinates) of the center of the visual marker. u_A represents the area of the visual marker on the image over the total resolution of the camera, and u_ϕ is the angle of the lower corner of the visual marker (in radians). The last two features are introduced to control the camera's distance to the visual marker and its orientation, respectively.

5.1.1 Extended IBVS

In addition to the classical point IBVS matrix, the components for the other features u_A and u_ϕ are derived below. With the perspective projection model for the camera (as introduced in the foundations), the projected area u_A decreases quadratically with the distance z^c (when the orientation is maintained) and is a feature that can be used to control the MAV's height. The derivative of this visual feature is

$$u_A = \frac{A_0}{({}^c z)^2} \implies \dot{u}_A = -2\frac{A_0}{({}^c z)^3}\dot{{}^c z} = -2\frac{A_0}{({}^c z)^3}\begin{bmatrix}0 & 0 & 1 & {}^i y & {}^i x & 0\end{bmatrix}\begin{bmatrix}{}^c v_x \\ {}^c v_y \\ {}^c v_z \\ {}^c w_x \\ {}^c w_y \\ {}^c w_z\end{bmatrix}, \tag{5.8}$$

and is used for computing the interaction matrix with this feature. The parameter A_0 can be calculated heuristically as the relative area of the visual marker at a distance of 1 m.

For the feature u_ϕ, this derivation is rather trivial, since the orientation relative to the object's bottom line is already directly related to the MAV's absolute yaw position to less than a constant.

$$u_\phi = \phi_z + \phi_{z_0} \implies \dot{u}_\phi = \dot{\phi}_z = {}^c w_z = \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}{}^c v_x \\ {}^c v_y \\ {}^c v_z \\ {}^c w_x \\ {}^c w_y \\ {}^c w_z\end{bmatrix}, \tag{5.9}$$

With this, the interaction matrix L_s becomes

$$\dot{\mathbf{s}} = \begin{pmatrix}\hat{u}_x \\ \hat{u}_y \\ \hat{u}_A \\ \hat{u}_\phi\end{pmatrix} = L_s\ {}^c\mathbf{v} = \begin{bmatrix}-1/Z & 0 & {}^i x/Z & {}^i x * {}^i y & -(1+({}^i x)^2) & {}^i y \\ 0 & -1/Z & {}^i y/Z & (1+({}^i y)^2) & -{}^i x * {}^i y & -{}^i x \\ 0 & 0 & -2A_0\frac{1}{Z^3} & 2A_0\frac{{}^i y}{Z^3} & -2A_0\frac{{}^i x}{Z^3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}\begin{pmatrix}{}^c v_x \\ {}^c v_y \\ {}^c v_z \\ {}^c w_x \\ {}^c w_y \\ {}^c w_z\end{pmatrix} \tag{5.10}$$

As stated above, the degrees of freedom to be controlled are only those in Equation (5.1), which means that for this scope, rows 4 and 5 of ${}^c\mathbf{v}$ are not of interest as input of the control. The MAV with a fixed roll or pitch position is unable to maintain its absolute x and y positions,

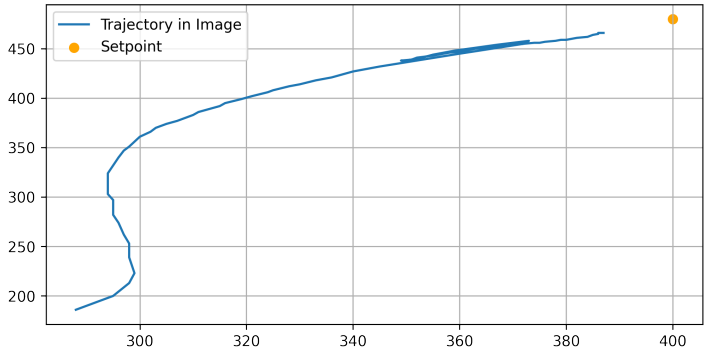

Figure 5.2: Trajectory of object's center (pixel) coordinates in the image during execution of the state "center" with the controller from Section 5.1.1

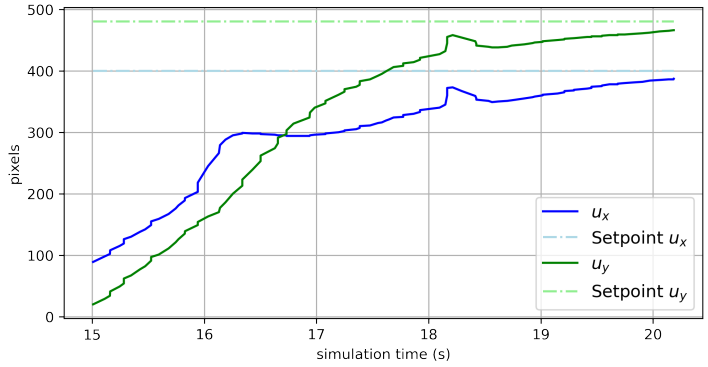

Figure 5.3: Time domain plot of object's center (pixel) coordinates in the image during execution of state "center" with the controller from Section 5.1.1

so they are not used as input. For this reason, columns 4 and 5 of L_s are dismissed, resulting in the matrix

$$L_s = \begin{bmatrix} -1/Z & 0 & {}^{i}x/Z & {}^{i}y \\ 0 & -1/Z & {}^{i}y/Z & -{}^{i}x \\ 0 & 0 & -2A_0\frac{1}{Z^3} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5.11}$$

from which the control of Equation (2.5) is implemented at a rate of 50 Hz, to provide velocity controls as in the control architecture in Figure 3.1. This is done with the use of a ROS Rate object, that enables control of the loops frequencies in the ROS Time (time used for the simulation, not in real execution time).

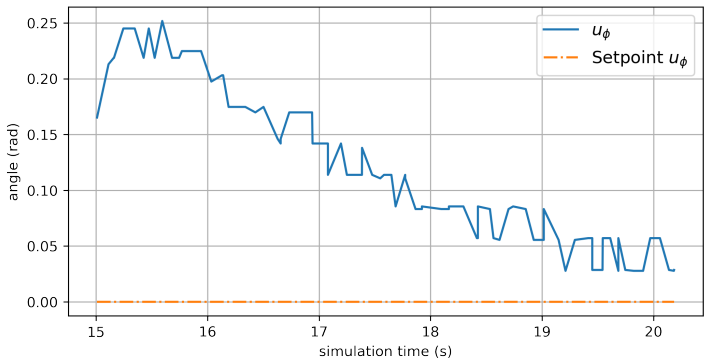

Figure 5.4: Time domain plot of angular feature (in radians) during execution of state "center" with the controller from Section 5.1.1

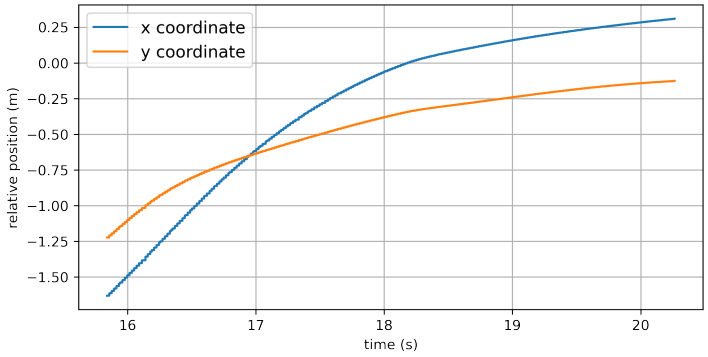

Figure 5.5: Time domain plot of drone's groundtruth relative position to the visual marker during execution of state "center" with the controller from Section 5.1.1

Experiments and results

This controller is used for both the center and approach states of the state machine presented in Figure 3.4. With the difference that in the "center" state the setpoint is constant and equals $\mathbf{s}^* = (400, 480, u_{A_0}, 0)^T$, with u_{A_0} being the area of the first recognition of the marker - essentially intending to maintain the initial area/height. While in the "approach" state, the desired area ratio is increased quadratically until it reaches around 30% of the image, that is, $u_A = 0.3$.

The marker's center position in the image at $(400, 480)$ is a position that centers the MAV's base frame with the center of the object, given that the camera is slightly shifted in the ${}^bX$ axis.

With this model plugged into the control scheme from Equation 2.5, the controller stabilizes the simulated vehicle and correctly follows the setpoints of the visual features, as can be observed in Figures 5.2, 5.3, 5.4 and 5.5 during the performance of the "centering" state, and in Figure 5.6 on the execution of the states "approach" and "land".

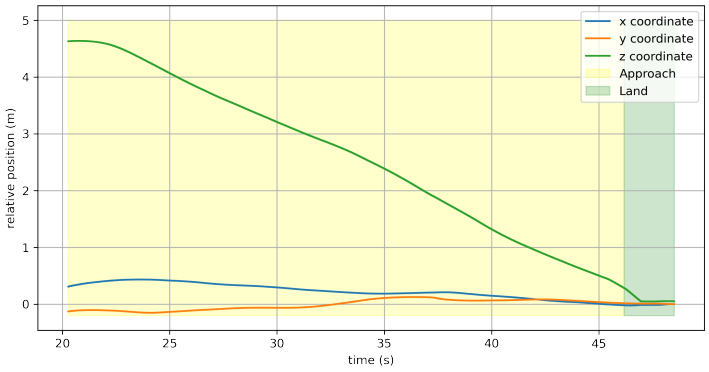

Figure 5.6: Ground truth position relative to the object during states ”approach” and ”land” with the controller from Section 5.1.1

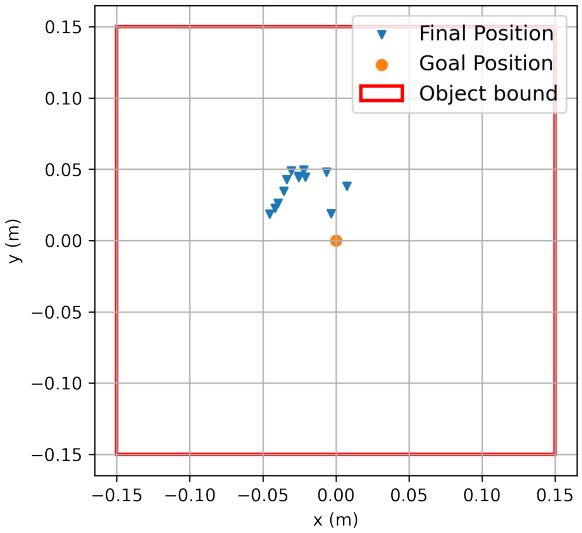

Figure 5.7: Ground truth position relative to the object after landing with the controller from Section 5.1.1

After running the simulation multiple times, the result was an average accuracy - that means an error - of 4.5 cm to the desired position and a standard deviation of 0.9 cm. The distribution of final positions is presented in Figure 5.7. Such a low variance is expected in an experiment like this since the effective variability between the simulations was only due to sensor and camera noises and different initial orientations of the object. A similar study with the wind as an external perturbation is executed at the end of this chapter.

Such an error in the final landing position is considerably small compared to the object's 30 cm dimensions and is sufficient to enable magnetic contact of the magnets on the MAV with those on the object.

It is also worth mentioning that although the actual magnets are not considered in the simulation, those would not decrease the quality of those results but actually increase them, since

the last-centimeter performance could be slightly facilitated by the attractive magnetic forces.

5.1.2 Decoupled Controller

As a simpler approach to control the MAV based on the chosen image features, a control loop can be established considering only the direct first order effect of the drone's velocity in the visual features. For instance, consider that the x position in the image plane is only influenced by the velocity on the camera's x-axis. This ignores the influence of the vertical velocity and yaw twist in the value of this feature on grounds of having the ratio $\frac{\dot{Z}}{Z^2}$ small, in the equations 2.7, such as

$$\dot{\hat{u}}_x = \dot{X}/Z - \underbrace{X\frac{\dot{Z}}{Z^2}}_{\approx 0, for \dot{Z}/Z^2 \text{ small}} . \tag{5.12}$$

By adopting this, the controller achieved has uncoupled degrees of freedom, i.e. the interaction matrix becomes diagonal and easily invertible and is presented below.

$$\dot{s} = L_s \begin{pmatrix} \mathbf{V}_s p \\ \phi_s p \end{pmatrix} = \begin{bmatrix} -1/Z & 0 & 0 & 0 \\ 0 & -1/Z & 0 & 0 \\ 0 & 0 & -2A_0/Z^3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{q} \tag{5.13}$$

And can be controlled through the scheme below

$$\dot{\mathbf{q}}(t) = \begin{bmatrix} v_x^b \\ v_y^b \\ v_z^b \\ w_z^b \end{bmatrix} = -\lambda J_s^+ e(t) = -\lambda J_s^+ \underbrace{\left(\begin{bmatrix} \hat{u}_x \\ \hat{u}_y \\ u_A \\ u_\phi \end{bmatrix} - \begin{bmatrix} \hat{u}_x^* \\ \hat{u}_y^* \\ u_A^* \\ u_\phi^* \end{bmatrix} \right)}_{\mathbf{e}(t)} \tag{5.14}$$

Experiments and results

The performance of this approach also stabilizes the simulated MAV and correctly follows setpoints of the visual features, as can be observed when plotting the trajectory of the visual features during the performance of the state "center" in Figures 5.8, 5.9 and 5.10 and the state "approach" and "land" in Figure 5.11. Both the trajectory of the features and their time series converge to the desired values.

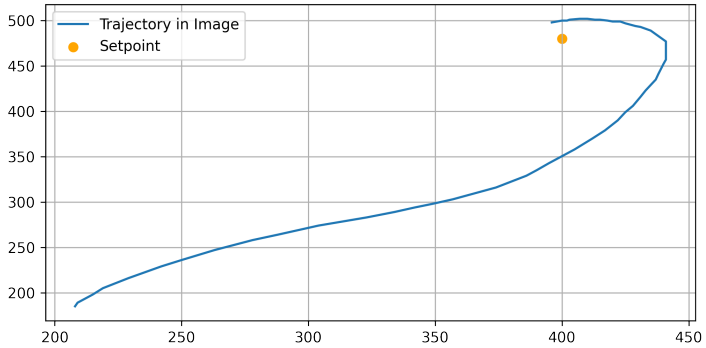

Figure 5.8: Trajectory of object's center (pixel) coordinates in the image during execution of state "center" with the controller from Section 5.1.2

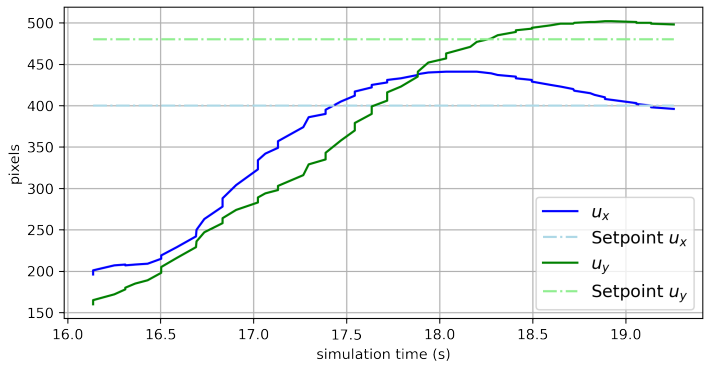

Figure 5.9: Time domain plot of object's center (pixel) coordinates in the image during execution of state "center" with the controller from Section 5.1.2

When executing the "approach" state, the MAV gets as close as possible to the object while the visual marker is still visible; when the area feature u_A assumes a value of approximately 0.3 - then the landing on the object while holding the position is commanded. The ground truth position of the MAV's base frame to the object is used as a metric of how close the object was approached, and is taken from the final values from the graph in Figure 5.11.

After running the simulation for multiple times, the result was an average accuracy - that means an error - of around , with the distribution of 7.7 cm and a standard deviation of 0.92 cm. The distribution of final positions is presented in Figure 5.12.

That means an error of around 8 cm is achieved even after visual contact is lost with the marker. As expected, for a simplified approach, it presents a very similar error compared to the one in Section 5.1.2, but it is still capable of leading the vehicle to a position sufficiently close to the desired location so that the MAV lands on it.

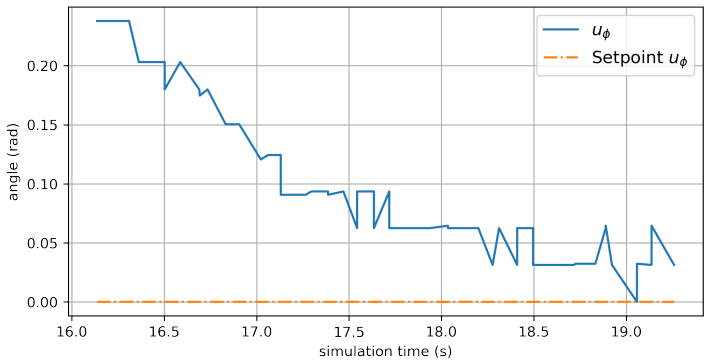

Figure 5.10: Time domain plot of angular feature (in radians radians) during execution of state "center" with the controller from Section 5.1.2

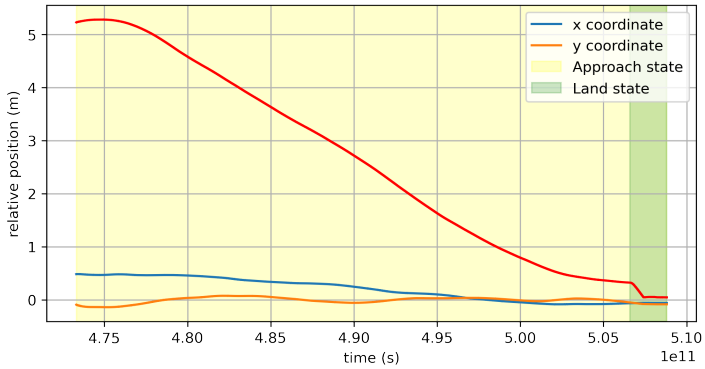

Figure 5.11: Ground truth position relative to the object during states "approach" and "land" with the controller from Section 5.1.2

5.1.3 Robustness against perturbations

To analyze the robustness of the control system (from Section 5.1.1) against external disturbance, wind and wind gusts were added to the simulated environment. The wind on gazebo is simulated using the gazebo wind plugin, that allows the creation of wind in a simple uniform worldwide model [43] - in this case, the wind is independent from the position in order to simplify the computations, and can be configured according to the parameters 5.1.

When such wind is added to the system, the behavior in Figure 5.14 is obtained. According to the tests, the wind is particularly problematic for the image-based approach when the MAV is significantly close to the visual marker. In this case, there is a significant movement of the visual marker in the image, sometimes occasioning visual loss of the target. However, this is treated by the system through the state machine; when this happens, the drone gets back to a position in which it had visual contact with the object and a safe centering height (1 m) and then, after centralizing the target in the image, attempts a new approach to the object.

In addition, during the very last moments of the landing when the visual marker is not vis-

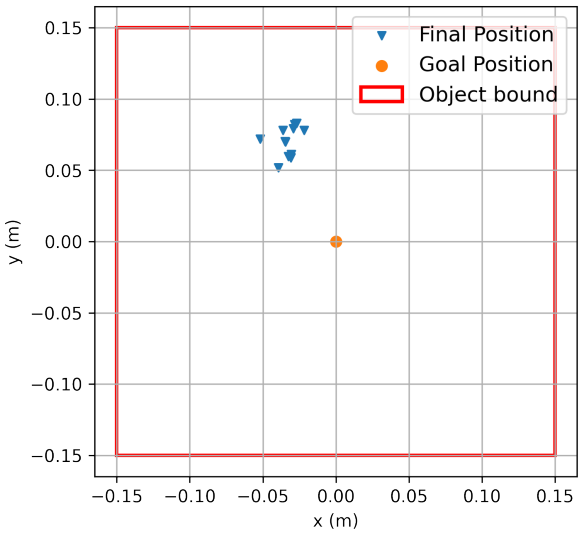

Figure 5.12: Ground truth position relative to the object after landing with the controller from Section 5.1.2

Mean Velocity	Velocity Variance	Mean Gust Velocity	Gust Variance Velocity
2.0 m/s	4.0 m^2/s^2	0.0 m/s	4.0 m^2/s^2

Table 5.1: Properties of the simulated wind

ible, holding the MAV's horizontal position while applying a vertical approach velocity has proven to be more robust to the wind than using velocity-based control. That's done using a *PositionTarget* setpoint, which allows controlling desired positions, velocities, and/or accelerations. When running multiple experiments with the perturbation, the mean final horizontal ground truth distance relative to the object was 13 cm, with 6 cm with the distribution of landing positions presented in Figure 5.13. From the attempts of landing in those conditions, a success rate with the wind of 66% was achieved. The achieved average error distance is considerably small given the perturbations, and the MAV was able to land on top of the object for the majority of attempts.

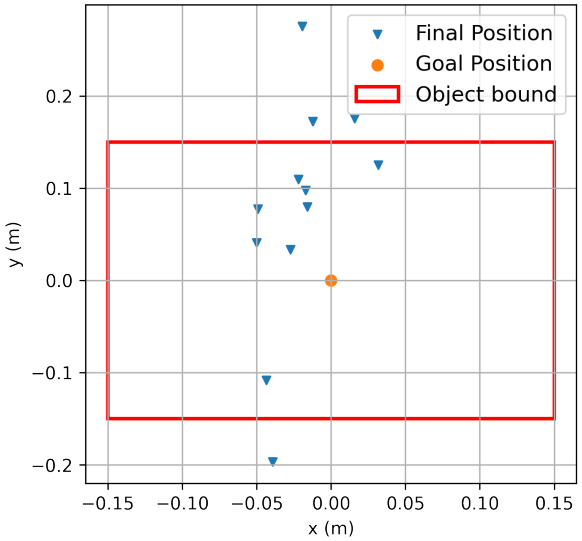

Figure 5.13: Ground truth position relative to the object after landing with the controller from Section 5.1.1 under the effect of wind

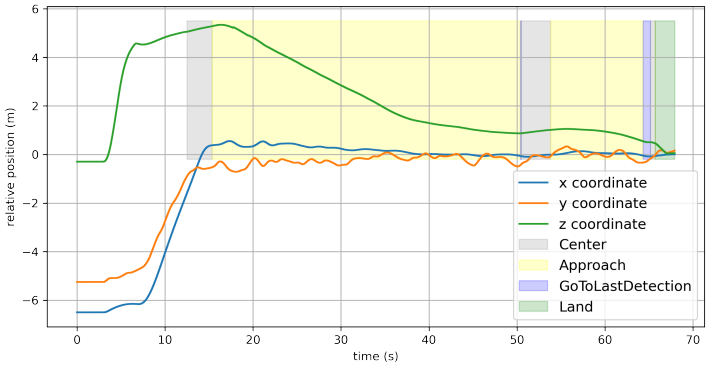

Figure 5.14: MAV's ground-truth position relative to the visual marker during the whole simulation. With annotations regarding the state transitions.

6 Object Carrying

This section briefly discusses the object manipulation i.e. object carrying by the quadrotor in the context of the project and presents how the simulations were executed.

6.1 Simulation

As described in Section 1.2, the MAV is to land on an object in order to enable magnetic coupling of the magnets on the object with those on the vehicle. The problems of approaching and landing on the object and taking off with it can be considered separate; therefore, a second simulation was prepared with the drone starting attached to the object by a fixed joint, which, assuming the magnets are sufficiently strong, illustrates well the real case.

In terms of controlled flight with the object, a regular absolute position control is used to guide the vehicle until the desired location. The setpoint for the position is given to the controller in Figure 2.5. In the PX4 Firmware, the gross weight of the vehicle can be updated to account for the extra payload relative to the object in the simulation. In case of small variation in the mass of the objects, the control cascade has proven to be fairly robust [44], otherwise those parameters can be identified in-flight as shown by [18]. Different payload values were tested until the 400 g limit specified in 3, for which the trajectory obtained followed by the vehicle during flight is presented in Figure 6.1.

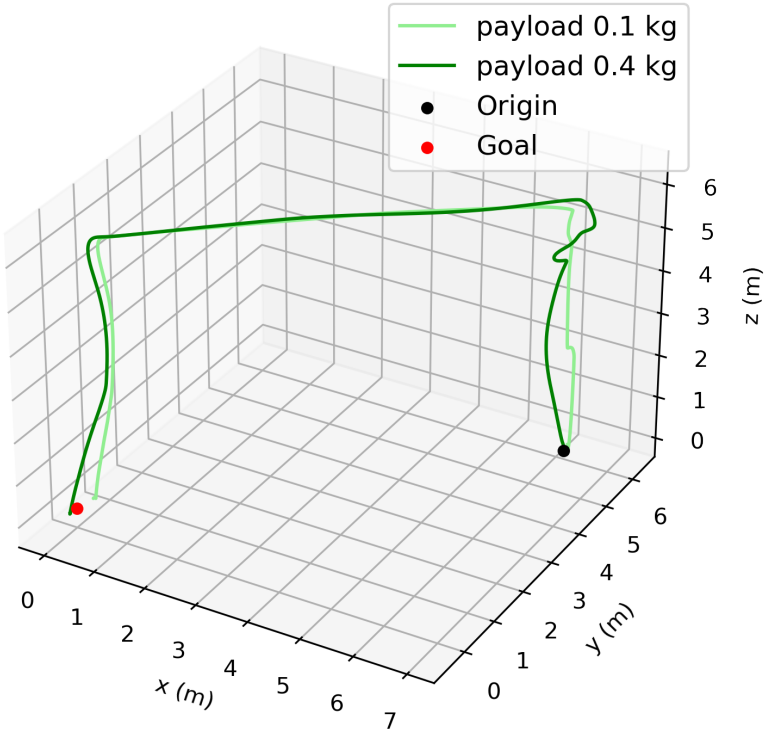

Figure 6.1: MAV's ground-truth trajectory relative to world's origin with object attached to it, from takeoff until landing in destination; trajectories are shown with both payloads of 100 g and 400 g

7 Conclusions and Outlook

In this work, the challenges faced in the implementation of a drone delivery system are explored and a solution is implemented and tested in a simulated environment. The simulation environment and the communication between the modules are made on top of ROS and the PX4 Autopilot in order to control and run the drone in Gazebo; the proposed state-machine-based system architecture for picking up and delivering objects proves to add up in safety and robustness of the system, both when it comes to dynamic perturbations and to operation cases such as the imprecise initial input of the object's position. In addition to that, a control scheme is implemented to stabilize and control the vehicle based on visual features from the camera's image, ultimately being able to visually guide it until landing in the object with considerable precision in a noisy environment. At last, the takeoff and the trajectory until the desired destination of the package are also implemented in the simulations.

The control strategy was based on image-based visual servoing: a low-cost (camera enabled) and computationally cheap strategy that allows drones to land accurately even under disturbance with considerable precision, when the boundary conditions (e.g. end of vision-based control or loss of visual contact) are well handled by the system as a whole.

When applying IBVS to drones in the proposed configuration, one of the main problems (estimating the distance of the points to calculate the interaction matrix) can be solved by using the estimated height of the drone, either by a distance sensor (ideal when objects are not necessarily on the ground), or estimates of the relative height of the aircraft.

Visual control in the proposed case can be simplified to have no coupling between the visual features, without much loss of performance throughout the flight; this computationally simpler scheme can be used for applications in which the desired accuracy is slightly lower and computational resources are scarce.

7.1 Future Work

This section briefly discusses some of the future work that would aggregate high value to this project's proposal.

7.1.1 Physical Implementation

Between the expected differences in the real system functioning compared to the simulated one, a factor that can be considered is the battery operation, which is not accounted for in the simulation and provides a limitation to the system's functionality range. Besides that, other image disturbances outer than Gaussian noise at the camera occur in the real world, such as low lighting settings and shadows.

One other aspect that needs to be considered in the real implementation of such a system is where the high-level processing will be done. In the simulations, both the performance of the ROS nodes and the drone's firmware are executed on the same computer as the simulation, but in a physical system, the firmware and low-level control would be at the iris's flight controller unit (FCU), while ROS nodes would need to be executed on an embedded computer (such as a single-board computer) and connected to the flight controller via a cable interface, through the MAVLink protocol.

Finally, the physical implementation can be made in such a way that some physical guidance from the vehicle's magnets to the ones on the object is added to the vehicle's structure, allowing perfect coupling even with small errors in the final positioning of the vehicle after landing.

7.1.2 Implementation with Multiple Agents

The first concern when implementing such a project with multiple agents is to ensure that collisions won't happen. This can be maintained in multiple ways. One very effective way used in the robotics industry to deal with obstacle avoidance is the use of potential fields. Essentially the addition of an attractive field (low potential) to the desired goal location while adding a strongly repulsive field (high potentials) to obstacles - in this case, other MAVs [45] and guiding the vehicle based on the gradient of the generated potential field. One of them, if a centralized processing with telemetry of all MAVs is available, is to use virtual potential fields to generate the desired trajectories. Those repulsive fields could also be implemented locally, when two or more MAVs are close enough to communicate, the location can be exchanged in order to create the repulsive fields. Here, this communication between the vehicles is another problem to be solved.

Another simpler but more limited approach is to define an operation height for each agent, which, however, might lead to less efficient trajectories, since the ascending of a MAV is a very energy-consuming action.

Besides that, the efficiency of the system can also be addressed in the multi-agent configuration. In this case, a problem arises when optimally assigning drones to their packages and considering the possible landing sites.

7.1.3 Object pick-up and dropping during simulation

In order to provide a more complete simulation, and enable the simulation of such a system with multiple agents, or a single agent with multiple objects to deliver, the coupling of the MAV with the object would need to be made during the simulation time. That means in this case, a fixed joint would need to be created dynamically in the simulation when contact occurs between the MAV and the object. That can be implemented through ”detachable joints” that are available in the Gazebo simulator [46], but was not implemented in this project for time restrictions.

Bibliography

[1] Diego Moreno-Jacobo, Gustavo Toledo-Nin, Alberto Ochoa-Zezzatti, Vianey Torres, and Fernando Estrada-Otero. Evaluation of Drones for Inspection and Control in Industry 4.0. In Alberto Ochoa-Zezzatti, Diego Oliva, and Angel Juan Perez, editors, *Technological and Industrial Applications Associated with Intelligent Logistics*, Lecture Notes in Intelligent Transportation and Infrastructure, pages 579–595. Springer International Publishing, Cham, 2021.

[2] Hanlin Zhang, Sixiao Wei, Wei Yu, Erik Blasch, Genshe Chen, Dan Shen, and Khanh Pham. Scheduling Methods for Unmanned Aerial Vehicle Based Delivery Systems. In *2014 IEEE/AIAA 33rd Digital Avionics Systems Conference (DASC)*, pages 6C1–1–6C1–9, 2014.

[3] Amazon Staff "Amazon Prime Air Prepares for Drone Deliveries". `https://www.aboutamazon.com/news/transportation/amazon-prime-air-prepares-for-drone-deliveries`. (accessed 2022-10-28).

[4] Xiaoying Sun, Xiaojun Zhu, Pengyuan Wang, and Hua Chen. A Review of Robot Control with Visual Servoing. In *2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 116–121, Tianjin, China, July 2018. IEEE.

[5] J. Hill and W. Park. Real Time Control of a Robot With a Mobile Camera. In *Proc. of the 9th International Symposium on Industrial Robots*, volume 1, pages 233–246 vol.1, 1979.

[6] B. Espiau, F. Chaumette, and P. Rives. A new Approach to Visual Servoing in Robotics. *IEEE Transactions on Robotics and Automation*, 8(3):313–326, June 1992.

[7] A. Sanderson and L. Weiss. Image-based Visual Servo Control using Relational Graph Error Signals. In *Proc. the IEEE International Conference on Robotics and Automation*, pages 1074–1077, 1980.

[8] T. Hamel and R. Mahony. Visual Servoing of an Under-actuated Dynamic Rigid-body System: an Image-based Approach. *IEEE Transactions on Robotics and Automation*, 18(2):187–198, April 2002.

[9] Omid Shakernia, Yi Ma, T. John Koo, and Shankar Sastry. Landing an Unmanned Air Vehicle: Vision Based Motion Estimation and Nonlinear Control. *Asian Journal of Con-*

trol, 1(3):128–145, 1999. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1934-6093.1999.tb00014.x.

[10] E. Altug, J.P. Ostrowski, and R. Mahony. Control of a Quadrotor Helicopter using Visual Feedback. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 1, pages 72–77 vol.1, 2002.

[11] Syaril Azrad, Farid Kendoul, Dwi Perbrianti, and Kenzo Nonami. Visual Servoing of an Autonomous Micro Air Vehicle for Ground Object Tracking. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5321–5326, 2009.

[12] Abel Gawel, Mina Kamel, Tonci Novkovic, Jakob Widauer, Dominik Schindler, Benjamin Pfyffer von Altishofen, Roland Siegwart, and Juan Nieto. Aerial Picking and Delivery of Magnetic Objects with MAVs. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5746–5752, Singapore, Singapore, May 2017. IEEE.

[13] Kun Yang and Quan Quan. An Autonomous Intercept Drone with Image-based Visual Servo. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2230–2236, Paris, France, May 2020. IEEE.

[14] Katsushi Ikeuchi. *Computer Vision: A Reference Guide*. Springer US, Boston, MA, 2014.

[15] Vaibhav Ghadiok, Jeremy Goldin, and Wei Ren. Autonomous Indoor Aerial Gripping using a Quadrotor. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4645–4651, San Francisco, CA, September 2011. IEEE.

[16] Paul E. I. Pounds, Daniel R. Bersak, and Aaron M. Dollar. Grasping from the Air: Hovering Capture and Load Stability. In *2011 IEEE International Conference on Robotics and Automation*, pages 2491–2498, Shanghai, China, May 2011. IEEE.

[17] Paul E. I. Pounds, Daniel R. Bersak, and Aaron M. Dollar. Stability of Small-scale UAV Helicopters and Quadrotors with Added Payload Mass Under PID Control. *Autonomous Robots*, 33(1):129–142, August 2012.

[18] Daniel Mellinger, Quentin Lindsey, Michael Shomin, and Vijay Kumar. Design, Modeling, Estimation and Control for Aerial Grasping and Manipulation. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2668–2673, San Francisco, CA, September 2011. IEEE.

[19] Edwin Olson. AprilTag: A Robust and Flexible Visual Fiducial System. In *2011 IEEE International Conference on Robotics and Automation*, pages 3400–3407, Shanghai, China, May 2011. IEEE.

[20] John Wang and Edwin Olson. AprilTag 2: Efficient and Robust Fiducial Detection. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4193–4198, Daejeon, South Korea, October 2016. IEEE.

[21] Return Mode | PX4 User Guide. `https://docs.px4.io/main/en/flight_modes/return.html`. (accessed 2022-10-17).

[22] Shuai Wang, Xunhua Dai, Chenxu Ke, and Quan Quan. RflySim: A Rapid Multicopter Development Platform for Education and Research Based on Pixhawk and MATLAB. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1587–1594, Athens, Greece, June 2021. IEEE.

[23] Doosan Cho. A Study on a Flight Safe System in Unmanned Aerial Vehicles. 13(9):3, 2018.

[24] Tauã M. Cabreira, Lisane B. Brisolara, and Paulo R. Ferreira Jr. Survey on Coverage Path Planning with Unmanned Aerial Vehicles. *Drones*, 3(1):4, March 2019. Number: 1 Publisher: Multidisciplinary Digital Publishing Institute.

[25] L. Pari, J. M. Sebastián, A. Traslosheros, and L. Angel. Image Based Visual Servoing: Estimated Image Jacobian by Using Fundamental Matrix VS Analytic Jacobian. In Aurélio Campilho and Mohamed Kamel, editors, *Image Analysis and Recognition*, Lecture Notes in Computer Science, pages 706–717, Berlin, Heidelberg, 2008. Springer.

[26] Francois Chaumette and Seth Hutchinson. Visual Servo Control. I. Basic Approaches. *IEEE Robotics & Automation Magazine*, 13(4):82–90, 2006.

[27] Zhengyou Zhang. *Perspective Camera*, pages 590–592. Springer US, Boston, MA, 2014.

[28] R. Horaud, B. Conio, O. Leboulleux, and B. Lacolle. An analytic solution for the perspective 4-point problem. In *Proceedings CVPR '89: IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 500–507, San Diego, CA, USA, 1989. IEEE Comput. Soc. Press.

[29] T. Shakunaga. An Object Pose Estimation System Using A Single Camera. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 1053–1060, Raleigh, NC, 1992. IEEE.

[30] Yihong Wu, Fulin Tang, and Heping Li. Image-based Camera Localization: an Overview. *Visual Computing for Industry, Biomedicine, and Art*, 1(1):8, December 2018.

[31] Luis Ferraz, Xavier Binefa, and Francesc Moreno-Noguer. Very Fast Solution to the PnP Problem with Algebraic Outlier Rejection. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 501–508, Columbus, OH, USA, June 2014. IEEE.

[32] visp_auto_tracker - ROS Wiki. `http://wiki.ros.org/visp_auto_tracker`. (accessed 2022-10-20).

[33] Fabio Ruggiero, Vincenzo Lippiello, and Anibal Ollero. Aerial Manipulation: A Literature Review. *IEEE Robotics and Automation Letters*, 3(3):1957–1964, July 2018.

[34] ROS: Home. https://www.ros.org/. (accessed 2022-10-28).

[35] Open Standards Unify the Drone Industry. https://auterion.com/open-standards-unify-the-drone-industry/. (accessed 2022-10-28).

[36] Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys. Pixhawk: A system for autonomous flight using onboard computer vision. In *2011 IEEE International Conference on Robotics and Automation*, pages 2992–2997, 2011.

[37] Controller Diagrams | PX4 User Guide. https://docs.px4.io/main/en/flight_stack/controller_diagrams.html. (accessed 2022-10-12).

[38] 3DR Iris - The Ready to Fly UAV Quadcopter. http://www.arducopter.co.uk/iris-quadcopter-uav.html. (accessed 2022-10-24).

[39] Gazebo. https://gazebosim.org/home. (accessed 2022-10-28).

[40] NASA Fact Sheet: Space Technology Game Changing Development; Human Robotics Systems: Space Robotics Challenge (SRC). https://www.nasa.gov/sites/default/files/atoms/files/hrs_src_fs_170306.pdf. (accessed 2022-10-28).

[41] ROS with Gazebo Simulation | PX4 User Guide. https://docs.px4.io/main/en/simulation/ros_interface.html. (accessed 2022-10-20).

[42] AprilTag. https://april.eecs.umich.edu/software/apriltag.html.

[43] WindEffects Class Reference. https://gazebosim.org/api/gazebo/3.9/classignition_1_1gazebo_1_1systems_1_1WindEffects.html. (accessed 2022-10-17).

[44] Dario Brescianini, Markus Hehn, and Raffaello D'Andrea. Nonlinear Quadrocopter Attitude Control: Technical Report. Report, ETH Zurich, 2013. Accepted: 2017-06-13T20:16:31Z.

[45] Jawad N. Yasin, Sherif A. S. Mohamed, Mohammad-Hashem Haghbayan, Jukka Heikkonen, Hannu Tenhunen, and Juha Plosila. Unmanned Aerial Vehicles (UAVs): Collision Avoidance Systems and Approaches. *IEEE Access*, 8:105139–105155, 2020.

[46] Ignition Gazebo: Detachable Joints. https://gazebosim.org/api/gazebo/3.2/detachablejoints.html. (accessed 2022-10-28).