TECHNISCHE UNIVERSITÄT DRESDEN

IFA LS-AT

Fakultät Elektrotechnik und Informationstechnik Institut für Automatisierungstechnik

Bachelor Thesis

zum Thema

Image Based Visual Servoing for Aerial Robot

vorgelegt von Pablo Rodríguez Robles
im Studiengang Luft- und Raumfahrt - Bachelor of Science (B.Sc.), Jg. 2014
geboren am 28.02.1996 in León, Spain

Betreuer: Dipl.-Ing. Chao Yao
Verantwortlicher Hochschullehrer: Prof. Dr. techn. Klaus Janschek
Tag der Einreichung: 27.03.2018

Aufgabenstellung

Test der PDF-Integration

Achtung

Auch wenn die Möglichkeit besteht, die eingescannte Aufgabenstellung als PDF zu integrieren, muss in **einem einzureichendem Exemplar** die Aufgabenstellung **im Original** eingebunden werden.

TECHNISCHE UNIVERSITÄT DRESDEN

IFA LS-AT

Fakultät Elektrotechnik und Informationstechnik Institut für Automatisierungstechnik

Image Based Visual Servoing for Aerial Robot

Hier muss der Text für die deutsche Kurzfassung inklusive eines aussagekräftigen Bildes eingefügt werden.

Betreuer:	Dipl.-Ing. Chao Yao
Hochschullehrer:	Prof. Dr. techn. Klaus Janschek
Tag der Einreichung:	27.03.2018

Bearbeiter: Pablo Rodríguez Robles

TECHNISCHE UNIVERSITÄT DRESDEN

IFA LS-AT

Fakultät Elektrotechnik und Informationstechnik Institut für Automatisierungstechnik

Image Based Visual Servoing for Aerial Robot

Here an English abstract including one significant image must be inserted.

Tutor:	Dipl.-Ing. Chao Yao
Supervisor:	Prof. Dr. techn. Klaus Janschek
Day of Submission:	27.03.2018

STUDENT RESEARCH THESIS Author: Pablo Rodríguez Robles

Contents

1 Introduction **1**
- 1.1 Motivation and Background . . . 1
- 1.2 Aims and Objectives . . . 2

2 Theoretical Background and State of the Art **3**
- 2.1 Visual Servoing Theoretical Basics . . . 3
  - 2.1.1 Pinhole Camera Model . . . 7
  - 2.1.2 Image Moments as Visual Features . . . 7
  - 2.1.3 Robot Operating System (ROS) . . . 9
- 2.2 State of the Art . . . 10
  - 2.2.1 Visual Servoing for Aerial Robots . . . 10
  - 2.2.2 Visual Servoing for Aerial Manipulators . . . 14

3 Software Requirements Specification and Structured Analysis **20**
- 3.1 Software Requirements Specification . . . 20
  - 3.1.1 Product Perspective . . . 20
  - 3.1.2 User Characteristics . . . 21
  - 3.1.3 Assumptions and Dependencies . . . 21
  - 3.1.4 Functional Requirements . . . 21
  - 3.1.5 Other Requirements . . . 22
  - 3.1.6 General Constraints . . . 22
- 3.2 Structured Analysis . . . 23
  - 3.2.1 Context Diagram . . . 23
  - 3.2.2 Level A: IBVS Controller . . . 24
  - 3.2.3 Level B1: Compute Desired Visual Features . . . 26
  - 3.2.4 Level B2: Compute Current Visual Features . . . 27
  - 3.2.5 Level B4: Compute Control Law . . . 28

4 Visual Servoing Algorithm Description **30**
- 4.1 IBVS Using Perspective Projections . . . 30

5 Implementation of the Visual Servoing Controller **32**
5.1 IBVS Using Perspective Projections 32

6 Final Results and Conclusions **33**

7 Future Work **34**

List of Figures

2.1 Pinhole camera model 6

3.1 Context Diagram 23
3.2 Data Flow Diagram - Level A 24
3.3 Data Flow Diagram - Level B1 26
3.4 Data Flow Diagram - Level B2 27

List of Tables

2.1 Overview of the different approaches for VS in flying manipulators 19

3.1 Data Dictionary for Level A 25
3.2 Data Dictionary for Level B 29

List of Listings

Nomenclature

Abkürzungen

DOF	Degree of Freedom
GAS	Global Asymptotic Stability
GPS	Global Positioning System
IBVS	Image Based Visual Servoing
IMU	Inertial Measurement Unit
LIDAR	Light Detection and Ranging
OS	Operative System
PBVS	Position Based Visual Servoing
PD	Proportional Derivative
ROS	Robot Operative System
ROS	Structure Analysis
SRS	Software Requirement Specification
SVS	Self Visual Servoing
TUD	Technische Universität Dresden
UAV	Unmanned Aerial Vehicle
VS	Visual Servoing

1 Introduction

1.1 Motivation and Background

During the last decade, the use of Unmanned Aerial Vehicles (UAVs) has spread among very different applications. Flying robots can be very helpful to improve the way some tasks are already achieved by terrestrial platforms. For example, object transportation, environment mapping or surveillance. At the Institute of Automation Engineering[1] of the Technical University of Dresden, a drone is being developed in cooperation with the Institute of Solid Mechanics[2] to investigate the use of flying robots in aerial manipulation.

When dealing with manipulation of objects, it is desired that the aerial robot adopts a certain pose with respect to the target before the manipulation process really starts. The present work deals with the development of a Visual Servoing (VS) control system that helps a quadrotor robot to acquire the desired pose by means of image data.

A monocular monochrome camera as well as an Inertial Measurement Unit (IMU) are planed to be the only available on board sensors. For the controller proposed the feedback is directly computed from image features rather than estimating the robot's pose and using the pose errors as control input.

Vision results to be a passive (in contrast to GPS) and cheap sensor (in contrast to LIDAR system). Visual odometry is very helpful to navigate in GPS denied environments like indoors, but its not appropriated to regulate the relative navigation of the vehicle with respect to a target.

In order to integrate the visual servoing algorithm into the future modular robot system, the algorithm has been designed and tested on a under-actuated conventional quadrotor. The aerial robot is implemented within the ROS[3] framework, where the visual servoing controller developed for this thesis is also integrated. Instead of using real hardware the complete system is simulated

[1] Technische Universität Dresden. Institut für Automatisierungstechnik. 01062 Dresden, Germany

[2] Technische Universität Dresden. Institut für Festkörpermechanik. 01062 Dresden, Germany

[3] www.ros.org

using Gazebo[4].

1.2 Aims and Objectives

The aim of this work is to implement and test a VS control algorithm for a quadrotor, which could be later used by the Flypulator (TODO: Add reference) project. This includes the review of the state of the art with regard to Visual Servoing, the design of a solution and a prototypical implementation with in the ROS framework and simulation with Gazebo of a test case.

The present thesis documents comprehensively the theoretical background, implementation details and results of the conducted work through the following structure. In Chapter 2 the theoretical background and state of the art of Visual Servoing is presented. Chapter 3 gives a description of the system requirements as well as the system decomposition by Structure Analysis (SA) (see [**SA_Braune**]). Chapter 4 describes the solution developed and the algorithms to be tested. Chapter 5 deals with the implementation, testing and validation. Finally, Chapter 6 contains the final results and conclusions and Chapter 7 suggests future improvement and research paths.

[4] www.gazebosim.org

2 Theoretical Background and State of the Art

2.1 Visual Servoing Theoretical Basics

In this section the theoretical background of visual servo controllers is briefly discussed. The different parts of a Visual Servoing scheme are presented and the basic strategy is illustrated. It is usual in the literature to take [**chaumette_visual_2006**] and [**chaumette_visual_2007**] as the main reference when it comes to the theoretical setup of the discipline. As a result, the following description is based on these popular sources[1].

Visual Servoing is defined in the literature as the use of computer vision data to control the motion of a robot. The image data comes from a camera, which can observe the robot fixed in the space or moving with the robot. The latter approach is know as eye-in-hand Visual Servoing and is the selected one for the case of this work.

Visual servo controllers accomplish their task of reaching a certain pose by trying to minimize the following error $\boldsymbol{e}(t)$

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

Here, $\boldsymbol{m}(t)$ is a set of image measurements (e.g. the image coordinates of the interest points or the image centroid of an object), that is, information computed from the image data. With the help of these measurements a vector of k visual features, $\boldsymbol{s}(\boldsymbol{m}(t), \boldsymbol{a})$ is obtained, in which $\boldsymbol{a}$ is a vector containing different camera parameters. In contrast, $\boldsymbol{s}^*$ defines a set of desired features.

For the present case, where the target is not moving, $\boldsymbol{s}^*$ and the changes in $\boldsymbol{s}$ depend only on the camera motion.

There exist two main variants of Visual Servoing depending on how the features vector $\boldsymbol{s}$ is defined. On the one hand, Image Based Visual Servoing (IBVS) takes as $\boldsymbol{s}$ a set of features already available within the image data. It

[1] The interested reader should visit the Lagadic research group home page (http://www.irisa.fr/lagadic), pioneers in the area.

can be seen as a control of the features in the image plan such that moving the features to a goal configuration implicitly results in the task being accomplished (see [**espiau_1992**]). On the other hand, Position Based Visual Servoing (PBVS) considers for $\boldsymbol{s}$ a set of 3D parameters that must be estimated from the image data. Once the parameters are available, pose estimation is conducted and the Visual Servoing task results in a cartesian motion planing problem.

Using the PBVS approach leads to the necessity of camera calibration and estimation of the flying robot pose, these are two big disadvantages for the application intended in this work. On the other side, IBVS needs no camera calibration and allows the robot to achieve the pose desired without any pose estimation process. Resulting in a convenient method for cheap systems.

A simple velocity controller can be arranged in the following way. Let $\boldsymbol{v}_c = (v_c, \boldsymbol{\omega}_c)$ be the spatial velocity of the camera, with v_c the instantaneous linear velocity of the origin of the camera frame and $\boldsymbol{\omega}_c$ the instantaneous angular velocity of the camera frame, as a result the temporal variation of the features can be expressed as

$$\dot{\boldsymbol{s}} = \boldsymbol{L}_{\boldsymbol{s}} \boldsymbol{v}_c \tag{2.2}$$

Where $\boldsymbol{L}_{\boldsymbol{s}} \in \mathbb{R}^{k \times 6}$, the feature Jacobian, acts as interaction matrix relating the camera velocity and the change in the visual features.

The time variation of the error to be minimized can be obtained by combining 2.1 and 2.2

$$\dot{\boldsymbol{e}} = \boldsymbol{L}_{\boldsymbol{e}} \boldsymbol{v}_c \tag{2.3}$$

with $\boldsymbol{L}_{\boldsymbol{e}} = \boldsymbol{L}_{\boldsymbol{s}}$. The input for such a controller is the camera velocity $\boldsymbol{v}_c$, which, using 2.3, can be set in such a way that an exponential decrease of the error is imposed (i.e. $\dot{\boldsymbol{e}} = -\lambda \boldsymbol{e}$)

$$\boldsymbol{v}_c = -\lambda \boldsymbol{L}_{\boldsymbol{e}}^{+} \boldsymbol{e} \tag{2.4}$$

Here, $\boldsymbol{L}_{\boldsymbol{e}}^{+} \in \mathbb{R}^{k \times 6}$ is the Moore-Penrose pseudoinverse of $\boldsymbol{L}_{\boldsymbol{e}}$. It is computed as $\boldsymbol{L}_{\boldsymbol{e}}^{+} = (\boldsymbol{L}_{\boldsymbol{e}}^{T} \boldsymbol{L}_{\boldsymbol{e}})^{-1} \boldsymbol{L}_{\boldsymbol{e}}^{T}$, provided that $\boldsymbol{L}_{\boldsymbol{e}}$ is of full rank 6. Imposing this condition leads to $\|\dot{\boldsymbol{e}} - \lambda \boldsymbol{L}_{\boldsymbol{e}}^{T} \boldsymbol{L}_{\boldsymbol{e}} \boldsymbol{e}\|$ and $\|\boldsymbol{v}_c\|$ being minimal. Note that for the special case

of $k = 6$, if $\boldsymbol{L_e}$ is nonsingular, it is possible to obtain a simpler expression using the matrix inversion $\boldsymbol{v}_c = -\lambda \boldsymbol{L}_{\boldsymbol{e}}^{-1} \boldsymbol{e}$.

When implementing real systems it is not possible to know perfectly either $\boldsymbol{L_e}$ or $\boldsymbol{L}_{\boldsymbol{e}}^{+}$. Thus, an approximation of these two matrices is introduced, noted with the symbol $\widehat{\boldsymbol{L_e}}$ for the approximation of the error interaction matrix and $\widehat{\boldsymbol{L}_{\boldsymbol{e}}^{+}}$ for the approximation of the pseudoinverse of the interaction matrix. Inserting this notation in the control law we obtain

$$\boldsymbol{v}_c = -\lambda \widehat{\boldsymbol{L}_{\boldsymbol{e}}^{+}} \boldsymbol{e} \tag{2.5}$$

Once the basic appearance of a visual servo controller has being presented, the goal is to ask the following questions: How should $\boldsymbol{s}$ be chosen? What is the form of $\boldsymbol{L_s}$? How should we estimate $\widehat{\boldsymbol{L}_{\boldsymbol{e}}^{+}}$?

In the simplest approach, the vector $\boldsymbol{s}$ is selected as a set of image-plane points, where $\boldsymbol{m}$ are the set of coordinates of these image points and $\boldsymbol{a}$ the camera intrinsic parameters. Later in this work, a more complex definition for the image features vector $\boldsymbol{s}$ will be used.

Concerning the interaction matrix $\boldsymbol{L_s}$

The Interaction Matrix

The interaction matrix, which relates the camera velocity to the change of the visual features, is strictly related to the the camera model. A camera model describes the correspondence between objects in 3D space and their appearance in a 2D image. Here, the pinhole camera model is used.

The camera image capture is a procedure which projects a 3D point from its coordinates in the camera frame, $\boldsymbol{X} = (X, Y, Z)$, to a 2D image point with coordinates $\boldsymbol{x} = (x, y)$. From this geometry we have

$$\begin{cases} x = X/Z = (u - c_u)/f\alpha \\ y = Y/Z = (v - c_v)/f \end{cases} \tag{2.6}$$

where $\boldsymbol{m} = (u, v)$ gives the coordinates of the image point in pixel units, and $\boldsymbol{a} = (c_u, c_v, f, \alpha)$ is the set of camera intrinsic parameters: c_u and c_v are the coordinates of the principal point, f is the focal length, and α is the ratio of

the pixel dimensions. For a feature point: $\boldsymbol{s} = \boldsymbol{x} = (x, y)$.

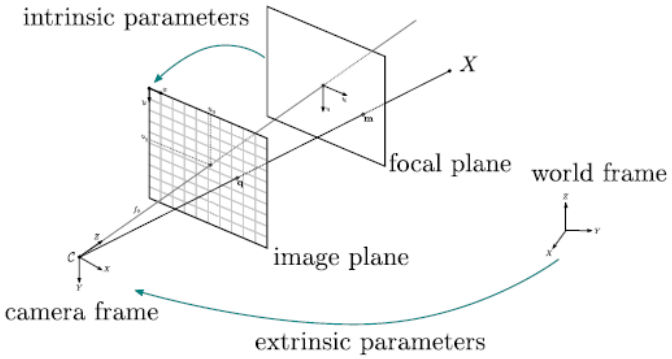

Figure 2.1: Pinhole camera model

Taking the time derivative of the projection Equation 2.6, we obtain

$$\begin{cases} \dot{x} = \dot{X}/Z - X\dot{Z}/Z^2 = (\dot{X} - x\dot{Z})/Z \\ \dot{y} = \dot{Y}/Z - Y\dot{Z}/Z^2 = (\dot{X} - y\dot{Z})/Z \end{cases} \tag{2.7}$$

The velocity of the 3D point can be related to the spatial velocity of the camera using the equation for the velocity in a non-inertial reference frame

$$\dot{\boldsymbol{X}} = -\boldsymbol{v}_c - \omega_c \times \boldsymbol{X} \Leftrightarrow \begin{cases} \dot{X} = -v_x - \omega_y Z + \omega_z Y \\ \dot{Y} = -v_y - \omega_z X + \omega_x Z \\ \dot{Z} = -v_z - \omega_x Y + \omega_y X \end{cases} \tag{2.8}$$

Introducing 2.8 in 2.7 and grouping terms, it can be written

$$\begin{cases} \dot{x} = -v_x/Z + xv_z/Z + xy\omega_z - (1 + x^2)\omega_y + y\omega_z \\ \dot{y} = -v_y/Z + yv_z/Z + xy\omega_z - (1 + y^2)\omega_x + x\omega_z \end{cases} \tag{2.9}$$

using matrix notation

$$\dot{\boldsymbol{x}} = \boldsymbol{L}_{\boldsymbol{x}} \boldsymbol{v}_c \tag{2.10}$$

where the interaction matrix that relates the camera velocity $\boldsymbol{v}_c$ to the velocity of the image point $\dot{\boldsymbol{x}}$ is

$$\boldsymbol{L}_{\boldsymbol{x}} = \begin{bmatrix} \frac{-1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & \frac{-1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{bmatrix} \tag{2.11}$$

In Equation 2.11, the value Z corresponds to the depth of the point relative to the camera frame. As a result, any Visual Servoing scheme using this form of the interaction matrix must provide an estimation of this value. Furthermore, the camera intrinsic parameters are necessary to compute x and y. Therefore, it is not possible to use directly $\boldsymbol{L}_{\boldsymbol{x}}$, but an approximation $\widehat{\boldsymbol{L}_{\boldsymbol{x}}}$ is to be used.

Approximation of the Interaction Matrix

When the current depth Z of each point is known, there is no need of approximation and $\widehat{\boldsymbol{L}_{\boldsymbol{e}}^+} = \boldsymbol{L}_{\boldsymbol{e}}^+$ for $\boldsymbol{L}_{\boldsymbol{e}} = \boldsymbol{L}_{\boldsymbol{x}}$ can be used. However, this approach requires the estimation of Z for all iterations of the scheme control (see [**hutchinson_1996**]), which may be conducted by means of pose estimation methods.

A second alternative is to use $\widehat{\boldsymbol{L}_{\boldsymbol{e}}^+} = \boldsymbol{L}_{\boldsymbol{e}^*}^+$, where $\boldsymbol{L}_{\boldsymbol{e}^*}$ is the value of $\boldsymbol{L}_{\boldsymbol{e}}$ for the desired position ($\boldsymbol{e} = \boldsymbol{e}^* = 0$) (see [**espiau_1992**]). Here, the depth parameter only needs to be estimated once for every point.

2.1.1 Pinhole Camera Model

2.1.2 Image Moments as Visual Features

As in other computer vision problems, the choice of visual features has a strong influence on the performance of an visual servo controller. An image feature is a piece of information extracted from an image and may be helpful to solve a certain task. Depending on the problem, different kind of features can be interesting for the user. These features are a specific structure in the image such as points, edges or closed contours, which are detected by means of operations on their pixel neighborhood.

Usually, the desired characteristics [**Szeliski2011**] for a visual feature are: locality (robust to occlusion and cluttering), invariance to transformations such as rotation or scaling, robustness against noise or compression and efficiency (so they can be computed in real-time). In addition to that, for a visual servoing task it is convenient that the set of features is decoupled, so it can be used to control independently all the degrees of freedom of the robot.

Image moments are the result of a particular weighted average of the pixels' intensities of an image or a combination of various of these moments. They are useful to describe an object after segmentation, and have been used in computer vision for patter-recognition tasks [**hu_1962**].

Before the introduction of image moments, other visual features such as points from corner detectors, Fourier descriptors or light intensity where used to carry visual servoing tasks. Image moments result of particular interest for planar objects when the desired configuration is such that the camera plane and the object are parallel. They are easy to compute, robust to noise and its interaction matrix can be computed for any case thanks to the analytical method presented in [**chaumette_image_2004**]. The inclusion of spherical image moments as visual features has the advantage that they are invariant under rotation of the camera frame [**bourquardez_stability_2006**].

Image moments have also the advantage of being intuitive for simple cases. The simplest example of image moments are the area and centroid coordinates of an object.

For a greyscale image with pixel intensities $I(x, y)$, the raw image moments m_{ij} are computed as

$$m_{ij} = \sum_{k=1}^{n} x_k^i y_k^j \tag{2.12}$$

While entered moments are given by

$$\mu_{ij} = \sum_{k=1}^{n} (x_k - x_g)^i (y_k - y_g)^j \tag{2.13}$$

Where for the area $a = m_{00}$ and for the centroid coordinates $\{\bar{x_g}, \bar{y_g}\} = \{m_{10}/m_{00}, M_{01}/M_{00}\}$. Any moment defined in this way, is known to be invariant

to 2D translational motion.

Several works [**bourquardez_2009**] [**tahri_2005**] introduced the area and centroid coordinates of a target as visual features. For this case, the interaction matrix has the following form [**tahri_2005**].

$$\boldsymbol{L}_{x_g} = \begin{bmatrix} -C & 0 & Cx_g & \varepsilon_1 & -(1+\varepsilon_2) & y_g \end{bmatrix} \tag{2.14}$$

In this work, the normalized area and normalized centroid coordinates of a planar target are used as visual features.

2.1.3 Robot Operating System (ROS)

The Robot Operating System (ROS)[2] is an open-source, operating system for robots. It provides all the capabilities expected from any operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionalities, message-passing between processes and package management [**ROS_Intro**].

ROS works as a peer-to-peer network of processes, called *nodes*, connected through the ROS communication infrastructure. ROS implements several communication interfaces such as *services*, *topics* and a *Parameter Server*.

As a result, ROS works as a framework where the user can implement the desired robotic system modularly as a graph connecting several independent processes. The main ROS concepts are [**ROS_Concepts**]:

Nodes Nodes are processes that perform a certain computation. The robotic system is decomposed in functionalities running in each of the in different nodes. For example, one node processes de camera information, other node controls the wheel motors and a third node is responsible of the path planning.

Master The ROS Master builds and maintains the name registration and lookup of the rest of components of the computation graph, so it allows the identification between nodes to establish the desired communication.

Messages: Messages are the data transfered between nodes. A message is a data structure comprising typed fields. The standard primitive types

[2] www.ros.org

(integer, float point, boolean, etc.) can be combined in nested structures or arrays.

- *Topics*: Topics are an asynchronous streaming of data between nodes following a publish/subscribe semantic. For example, data coming from a sensor. A node can send out a message through a topic publishing into it and an interested node can receive the message by subscribing the topic. A topic can use only a unique type of message. Topics work as an abstraction layer to separate the data production from the data manipulation using different nodes that communicate between each other without depending one on the other.

- *Services*: A constant stream of data is not always the desired communication case. It is also possible to desire a request/reply interaction between to nodes. For example to start the motors of a robot. This kind of communication is conducted using services. A providing node offers a service under a name and a client uses the service by sending the request message and awaiting the reply.

- *Parameter Server*: The Parameter Server allows data to be stored in a centralized data structure similar to a dictionary, where it can be accessed by any node.

In the present work, the system designed works as a set of nodes establishing communication between the *action interface* described in the Section TODO, which subscribe to the sensor topics of the robot and publish control command to the command topic of the robot.

2.2 State of the Art

2.2.1 Visual Servoing for Aerial Robots

In this section, the literature on the use of Visual Servoing to control de translation and yaw motion of aerial robots is analyzed. The focus of this review is on the IBVS approach, where image feature errors between the current and target image are mapped to actuator inputs through the inverse of the Jacobian matrix. The different approaches presented complete the basic IBVS formulation with alternatives on feature selection, Jacobian matrix construction

and different control strategies to make the error converge to zero, and with it, the relative pose between UAV and target object will converge to the desired one.

The task is to move the camera attached quadrotor to match the observed image features with the predefined desired image features obtained from a stationary object.

In [**ceren_vision-based_2009**], the visual servo controller presented is very simple, using feature points and a simple proportional law to control the kinematics of the robot. The VS technique is not explained in detail since it uses the MATLAB Visual Servoing Toolbox[3]. However, the paper is a good reference when it comes to the system description of the most simple approach to follow for the control of a quadrotor using visual servoing: The desired velocities on the image plane are computed with the visual information, transformed to the robot body frame and later to motor speeds that provide the robot with this motion. In this case, a PD controller is used to make the robot follow the chosen velocities.

The Visual Servoing research was originally developed for serial manipulators. In order to translate these techniques to aerial robots there are different aspects that may be considered, derived of the fact that a quadrotor is an under-actuated system:

- Feature depth estimation.
- Dynamics of the system leading to a high coupling between camera motion and target (visual feature) motion on the image plane.

To cope with these problems different solutions have been proposed and applied for the usual configuration of a quadrotor helicopter, with the camera mounted on its platform (eye-in-hand configuration).

In the past, several approaches have been taken to face the problem of depth estimation, such as: partial pose estimation (see [**malis_2_1999**]), adaptive control (see [**Papanikolopoulos**]) or estimation of the Jacobian using quasi-Newton methods (see [**Piepmeier**]). Nowadays, the most frequent approach is the use of a partitioned control that allows the uncoupling of translational and

[3] https://sourceforge.net/projects/vstoolbox/

rotational degrees of freedom[4].

[**hamel_2002**] is one of the pioneering works to cope with the visual servo control of under-actuated systems as a quadrotor. Introduces the idea of controlling the full dynamics of the vehicle. The approach requires separate measurements of linear and angular velocities which are not needed in classical IBVS, since for this case the kinematic velocities are used directly as control variables. In particular the method presented, needs only a single inertial direction. For the case of quadrotors, which usually require hovering applications, this is the vertical axis acceleration, provided by a filtered IMU signal. Due to the rotational ego-motion of the camera, the dynamics of the image point involve the angular velocity as well as the velocity linear velocity of the vehicle. This dependence destroys the explicit triangular cascade structure of rigid body motion expressed in the inertial frame. However, under certain conditions on the camera geometry, it is possible to recover a passivity-like property (from virtual input to the backstepping error) sufficient to apply a backstepping control design for the non-linear dynamical system. The paper the formulates that only image geometry that preserves the passivity-like properties of the body fixed frame dynamics of a rigid object in the image space are those of a spherical camera. Thus, introducing the use of spherical image moments[5].

For serial manipulators, a low-level actuator is usually employed to compensate the dynamic behavior of the system, making possible to control it with velocity commands as a first order system. For a full-dynamics quadrotor model accounting for aggressive maneuvers this is not at all the case, since it is a fourth order system. Thus, other strategies are usually considered to control such a robot (see [**Mellinger**] for a comprehensive description of the control of the dynamics of a quadrotor and its trajectory generation in space). Already in the basic literature on Visual Servoing (see [**chaumette_visual_2007**]) the authors suggest that kinematic control may not be enough for the case of aggressive maneuvers.

Image moments are invariant to some transformations like scale, 2D trans-

[4]However, it is important to bear in mind that due to the under-actuated dynamics of an usual 4 DOF quadrotor, it is only possible to control the three linear velocities and the yaw (rotational) speed. While the other two rotational speeds, roll and pith, are used to move the thrust vectors of the propellers and thus generate movement on the horizontal plane.

[5]To use spherical moments is not necessary to implement physically a spherical camera, just to compute numerically the spherical moments of the image.

lation or 2D rotation. This has been intensively used in pattern recognition. Introducing visual features based on image moments allows to design a decoupled control scheme when the object is parallel to the image plane (see [**tahri_2005**]). Later, it is possible to generalize this property for the case of a non-parallel position with respect to the image plane.

In the work [**guenard_2008**], a IBVS controller for the full dynamics of a quadrotor system around the hovering position is proposed. The biggest difficulty to develop dynamic controllers based on IBVS is the high coupling in the image Jacobian between the rotational and translational dynamics. Something that does not happen in the case of PBVS. The paper introduces a new visual error formulation to improve the conditioning of the Jacobian matrix, following the idea of [**hamel_2002**]. Image moment features are augmented with the information provided by the IMU and used in conjunction with a non-linear controller.

Common strategy is to use to different loops to separate the control problem. The inner attitude loop runs at high gain using IMU measurements, while the outer loop runs at low gain with the visual input of the camera taking care of translation. The visual servo controller (outer loop) provides the attitude control (inner loop) the desired targets and outer loop ensures the stability of the system. An advantage of this approach is the possibility of reuse the IBVS scheme in other platforms since the low-level control of the specific material equipment of the aerial robot is not involved in it.

The paper [**bourquardez_2009**] focuses in the design of kinematic controllers. The use of zero and first-order image moments as visual features is considered inappropriate for aggressive maneuvers due to its lack of robustness for this case, where Global Asymptotic Stability (GAS) cannot be guaranteed. For this reason, first-order spherical moments are introduced. Different control schemes are proposed in the paper. First, one controller using perspective projections is able of regulating the translation of the system thanks to the decoupled relationship between image and task space obtained. For the case of the camera image plane being parallel to the target plane, it is possible to control the translation of the robot independently of its rotation in a way equivalent to PBVS but without any pose estimation involved. The target depth is introduced as an initial data, so no depth estimation is performed. The strategy is easy to implement and provides a good result within the assumed geometrical limitation.

To overcome the limitations of the simple model presented, the work intro-

duces order control schemes that include spherical projections as visual features and non-linear control techniques that improve the behavior for the case of aggressive maneuvers, although still around the hovering position.

In [**ceren_image_2012**], present two kinematic controllers, one based on points as visual features for IBVS and other implementing a Hybrid Visual Servoing scheme that requires partial pose estimation.

[**jabbari_dynamic_2012**] designs a controller for the dynamics of a quadrotor within the IBVS approach. Although spherical moments are able to guarantee GAS, they generate trajectories that are not adequate when transforming to Cartesian coordinates. In this work, the method presented allows creating trajectories that are not only convenient in the image plane (i.e. where a usual IBVS controller is considered) but also in the Cartesian space. A projection of the perspective image moment features. augmented with inertial information to better control the dynamics, on a virtual plane dependent on the pitch and roll of the robot is proposed. Thus, the paper extends the approach followed in [**bourquardez_2009**], here perspective projection was considered useful for a flight perpendicular to the target, but replaced by spherical moments to provide the GAS that it was not able to provide. In this way, a decoupled linear link between image and Cartesian coordinates is obtained using perspective image moments for the case in which the robot does not navigate perpendicular to the target. Finally a non-linear controller is derived for the system.

The authors in [**asl_vision-based_2015**], propose two improvements to the work presented in the previous paragraph. For most of the controllers in the literature the camera velocity is required. This comes usually from IMU estimations, but due to the usual lack of GPS of the robots the noisy measurements of the IMU may not be enough. The paper proposed a visual flow and non-linear observer strategy to overcome this inconvenience. Secondly, in the standard formulation of the Jacobian matrix a estimation of the visual feature depth is necessary, something which is expensive to include in the controller. This is usually solved tanks to the use of image moments and a predefined target depth. The authors include here a controller to deal with this uncertainty.

2.2.2 Visual Servoing for Aerial Manipulators

In this section a perspective of Visual Servoing applied to aerial manipulators is presented. The main available literature is analyzed and the common approaches

followed in it highlighted. An the end of the section, Table 2.1 summarizes the strategies commented with regard to the main characteristics of each method.

Some publications related to the University of Pennsylvania GRASP Laboratory[6] (see [**thomas_toward_2014**] and [**thomas_visual_2016**]) have studied the vision-based localization and servoing of quadrotors in grasping and perching tasks. However, the emphasis of these publications lays on the generation of dynamically-feasible trajectories in the image space, thus second order system control is performed instead of the most common kinematic control strategy. These vehicles are not manipulators in the sense of the rest of the approaches presented in this section, since the main task here is hanging from structures and grasping targets by means of aggressive, thus dynamical, maneuvers. In addition to that, the actuator used is not a high-DOF serial manipulator, but a 1 DOF gripper.

In the last years, a concrete aerial manipulator architecture has been popularized, for example in the context of the European projects ARCAS[7] and AEROARMS[8]. These aerial manipulators have usually the task of collecting an structural element from its initial position and fly it to a final position, where the element is used to assemble a strut structure. The main configuration of such a robot is an under-actuated rotary-wing aircraft (usually a quadrotor) and a robotic serial manipulator arm. Different degrees of freedom (DOF) are used for the arm and different camera placements are considered.

The usual implementation considers the simultaneous control (at the velocity level) of the mobile platform (i.e. quadrotor) and the manipulator for such a grasping task. Since the sum of the 4 DOF of a quadrotor plus the multiple-DOF of a serial manipulator leads to a redundant system, the possibility of choosing degrees of freedom is used to realize different subtasks (e.g. joint limit reaching prevention). The Visual Servoing controller chosen generates velocity inputs both for the manipulator joints (i.e. $\dot{\boldsymbol{q}}$) and for the quadrotor (i.e. translational velocity $\boldsymbol{v}$ and rotational velocity ω_z). The use of a weighted pseudo-inverse allows to favor the control of the mobile platform when the distances to the target are bigger and increase the manipulator mobility when it is close to the target.

[**mebarki_image-based_2014**] propose a quadrotor equipped with a 5

[6] https://www.grasp.upenn.edu
[7] http://www.arcas-project.eu
[8] http://www.arcas-project.eu

DOF arm. Traditional Visual Servoing distinguishes between two different classes of camera configuration: eye-to-hand (fixed in the workspace) and eye-on-hand (mounted on the mobile platform). In this paper a new configuration for the camera is presented, called onboard-eye-to-hand, i.e. the camera is placed on-board of the robot while it observes the manipulator. In this way, the manipulator can accomplish large rotations while the target is not left out of the camera field of view, as happens in the eye-in-hand configuration. Furthermore, for the case of eye-in-hand configuration, during assembly tasks the manipulator end-effector can contact or impact with objects and damage or obstruct the camera. Thanks to the onboard-eye-to-hand camera configuration the paper is able to introduce a variation of the IBVS approach, called Self Visual Servoing (SVS). Where the error nullified comes directly form the image itself (hence the adjective self) an there is no need for a target image. The servo controller implemented has two different tasks. The main task is positioning the feature points at a target position on the target object and the second one the end-effector motion. The error formulation decouples both tasks and a weighted pseudo-inverse is used to provide a different gain for the arm joints rates $\dot{\boldsymbol{q}}$ than for the UAV velocities $\boldsymbol{v}$ and ω_z control.

Image moments as features for the Visual Servoing are proposed in [**mebarki_exploiting_2013**] for the previous system. Furthermore, aerial manipulators have to cope with the change of the center of mass during flight due to the effect of suspended loads (see [**palunko_2012**]). To achieve this behavior, low-level attitude controllers are usually designed to compensate this effects using Cartesian impedance control (see [**lippiello_impedance_2012**]) or an adaptive control approach is followed. In this case, the system includes a controller to reduce dynamic effects by vertically aligning the arm center of gravity to the multirotor gravitational vector, along with one that keeps the arm close to a desired configuration of high manipulability and avoiding arm joint limits. In [**mebarki_cross-coupled_2014**], the author completes the robot with a nonlinear low-level controller that thanks to a integral approach allows the inclusion of the dynamic coupling of the UAV and the robotic arm, while the control of the system through the velocities provided by the IBVS high-level is maintained.

In this case the source (see [**danko_evaluation_2014**]) includes as host a gantry used to emulate an UAV and a 6 DOF manipulator with an end-effector mounted camera (i.e. eye-in-hand). Coordination of redundant degrees of freedom by means of partitioned control. Visual servoing is used to drive the

end-effector pose relative to a target thanks to the use of feature points and their desired positions.

[**lippiello_hybrid_2016**] uses a hybrid-control framework to take advantage of the main benefits of both IBVS and PBVS schemes to control a octorotor with a 6 DOF arm. Kinematic redundancy of the end-effector is used to accomplish secondary tasks and lead by a hierarchical task-composition algorithm, in conjunction with a smooth activation mechanism for the tasks.

DLR's work within ARCAS project (see [**laiacker_high_2016**]) uses of an helicopter and a 7 DOF manipulator. The helicopter is a bigger robot when compared to the rest of the systems, with more than 1 m from manipulator to center of gravity. Influence of the arm movement is significant to the helicopter flight and is actively compensated by the robot controller by means of a coordinated control of both elements. The paper discusses performance and accuracy in aerial manipulation, where the time it takes between the measurement of a position difference and its compensation using the manipulator or the flying platform is the main factor. Additionally, the work presents a multi-marker approach to compensate the possible occlusion of the target marker by the manipulator derived of the onboard-eye-to-hand camera configuration.

A combination of kinematic and dynamic models to develop a passivity-based adaptive controller which can be applied on both position and velocity control is proposed in [**kim_vision-guided_2016**]. Position control is used for waypoint tracking and landing, while velocity control is triggered for target servoing. The robot is a quadrotor with a 3 DOF arm and an eye-in-hand camera. The work uses IBVS with image moments as visual features and tries to solve two problems of the method when applied to aerial manipulators (under-actuated system). Firstly, movements of the manipulator produce movement of the camera, thus making probable that the target object is taken out of its field of view. For this reason a fisheye camera is used, so it is possible to introduce a bigger field of view. Secondly, the under-actuation of the robot is corrected introducing a image modification method. Velocity weighting of the Jacobian matrix in accordance of the situation is used for the simultaneous control of UAV and manipulator.

In [**santamaria-navarro_uncalibrated_2017**], redundant manipulation and hierarchical control law are combined with a new variation of the IBVS that does not need the camera parameters. The system establishes as primary task the avoidance of obstacles, as well as several secondary tasks. The visual servo strategy is used to drive the arm end-effector to a desired position and

orientation by using a camera attached to it. The configuration used is a 4 DOF quadrotor, which is equipped with a 6 DOF robotic arm. In the common IBVS approaches, Jacobian or interaction matrix, which relates the camera velocity with the image feature velocities, depends on a priori knowledge of the intrinsic camera parameters. The paper presents a variation of IBVS called Uncalibrated IBVS, the approach uses the barycenter of the features as control points. The method recovers the coordinates of these control points and also the camera focal length, with this data a new formulation of the Jacobian is constructed. The system also compensates by means of a hierarchical algorithm the position of the manipulator. The implementation of this method withing the ARCAS project uses Gazebo and is available on the Internet under the name Kinton[9].

[9] https://devel.iri.upc.edu/pub/labrobotica/ros/iri-ros-pkg_hydro/metapackages/iri_visual_servo/

Reference	Vehicle	Arm's DOF	Camera Configuration	VS Type	Visual feature	Comment
[**thomas_toward_2014**] and [**thomas_visual_2016**]	quadrotor	1	eye-in-hand	IBVS	Cylinder parameters	agressive maneuvers
[**mebarki_image-based_2014**]	quadrotor	5	onboard-eye-to-hand	SVS	points (no target)	-
[**mebarki_exploiting_2013**]	quadrotor	5	onboard-eye-to-hand	SVS	perspective projection image moments (no target)	-
[**mebarki_cross-coupled_2014**]	quadrotor	5	onboard-eye-to-hand	SVS	points (no target)	low-lever controller for dynamic coupling robot-arm
[**danko_evaluation_2014**]	gantry	6	eye-in-hand	IBVS	points	gantry used to emulate an UAV
[**lippiello_hybrid_2016**]	octortor	6	onboard-eye-to-hand	Hybrid VS	points	hierarchical task-composition algorithm, smooth task activation
[**laiacker_high_2016**]	helicopter	7	onboard-eye-to-hand	IBVS	points	discusses performance and accuracy, multi-marker approach
[**kim_vision-guided_2016**]	quadrotor	3	eye-in-hand	IBVS	corrected perspective projection image moments	adaptive controller for both position and velocity, fisheye camera
[**santamaria-navarro_uncalibrated_2016**]	quadrotor	7	eye-in-hand	Uncalibrated IBVS	blobs' barycenters	hierarchical task-composition algorithm

Table 2.1: Overview of the different approaches for VS in flying manipulators

3 Software Requirements Specification and Structured Analysis

This chapter deals with the Software Requirement Specification (SRS) [**IEEE8301998**] and the Structured Analysis [**SA_Braune**] of the system developed in this work. Thanks to these two procedures, the objectives that the system must fulfill and a decomposition of it into different functions are stated. This leads to a complete definition of the system.

The purpose of this work is to design a Visual Servoing controller to provide an under-actuated aerial robot the commands necessary to reach a desired pose with respect to a target object.

The visual servo controller developed is to be integrated into the HORUS aerial manipulator, a fully-actuated aerial robot equipped with a monocular monochrome camera pointing downwards.

3.1 Software Requirements Specification

In this section, the Software Requirement Specification [**IEEE8301998**] for the visual servo controller developed in this thesis is presented. The use of SRS helps to define the system that is being designed, tracking continuously that the product developed satisfies the needs of the user. Only when every requirement stated therein is fulfilled the implementation would be completed.

3.1.1 Product Perspective

The VS controller is to be used with an aerial robotic system based on the ROS framework. From the perspective of the robotic system, the VS controller subsystem will appear as a ROS node which publishes control commands through a ROS topic to the rest of the system.

The aerial robotic system used is the HORUS aerial manipulator developed at the Institute of Automation Engineering of the TUD.

The subsystem developed here is to interact with the camera hardware of the robot, a monocular monochrome camera pointing downwards. The output

of the subsystem are the control inputs of the aerial robot system, a kinematic control is adopted, being this inputs the linear and angular velocities of the vehicle. These inputs interact with the already implemented inner control loop for the attitude in the robotic system and are the main input for the outer control loop, the one in charge of the translation.

With the help of the visual servo controller, the robot is able to achieve a desired pose with respect to a target using only the visual information provided by the camera and with no pose computation involved.

3.1.2 User Characteristics

The product developed in this thesis will be used as part of a ROS-based system, thus the expected user is a designer willing to implement an Image Based Visual Servoing control strategy for his robotic system. The user should be familiarized with the ROS framework and the system will need the structure and interfaces of any standard ROS product.

3.1.3 Assumptions and Dependencies

The software has been tested on the following platform. Forward or backward support is not guaranteed on a different set-up.

- ROS version: ROS Indigo[1]
- Operating System: Ubuntu 14.04[2] Trusty Tahr, 64 bit

3.1.4 Functional Requirements

The functional requirements describe what the system must do to complete the overall task:

- F1: *Compute desired visual features.* For the desired pose with respect to the target, compute a vector of image features from the target.
- F2: *Compute current visual features.* For the current pose, compute a vector of image features from the target.

[1] http://wiki.ros.org/indigo
[2] http://releases.ubuntu.com/14.04/

- F3: *Compute feature error.* Compute the difference between the desired current feature vector and the desired feature vector to be used as error for the control law.
- F3: *Give visual servoing control input.* Control input based on image data so that the aerial robot achieves the desired pose with respect to the target. Control based on the error between current and desired image features, no pose estimation.
- F5: *Tell user when the target pose is achieved.* The system must be able of telling the user whether the target pose has been already achieved or not.

3.1.5 Other Requirements

- A1: All components are working reliably.
- A2: The software is sufficiently fast, modular and modifiable.
- A3: The implementation is transparent and comprehensible.
- A4: Control inputs must provide stable and smooth flight maneuvers.
- A5: Robot must be able to start from different initial positions.
- A6: Algorithm must be fast enough to allow real time control of the aerial robot.
- A7: The implementation should follow the style guide of ROS [**ROS_Style**].

3.1.6 General Constraints

- The environment must be sufficiently illuminated for the camera to work.
- The observed object must be planar and continuous and provided to the program as a binary image obtained by segmentation algorithms.
- The target pose must be provided by a sufficient number of features.
- The target must be always in the filed of view of the camera, so features can be extracted an control input computed.

- Testing computer is a MacBook Pro[3] (Early 2015) with 2.7 GHz Intel Core i5 processor, 8 GB 1867 MHz DDR3 memory and Intel Iris Graphics 6100 1536 MB graphics. Linux OS is run using Oracle VM VirtualBox[4] (Version 5.1.14 r112924) with 5 GB base memory and two processors.

3.2 Structured Analysis

The Structured Analysis [**Janschek2011**] is a formal method to describe relationships of functional type in complex processes. In order to do that, it views the system from the perspective of data flowing through it. The Structured Analysis is a top-down model, meaning that it analyses the system data flows through successive decomposition. The result of the Structured Analysis of a system is a set graphical diagrams representing the different processes or functions applied to the data and the outputs stored by the system.

3.2.1 Context Diagram

The context diagram shows the interfaces between the system developed in this work and its environment.

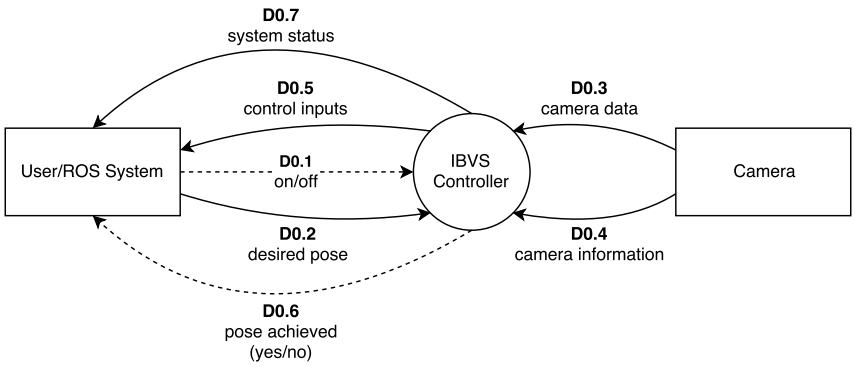

Figure 3.1: Context Diagram

[3] https://everymac.com/systems/apple/macbook_pro/specs/macbook-pro-core-i5-2.7-13-early-2015-retina-display-specs.html
[4] www.virtualbox.org

3.2.2 Level A: IBVS Controller

Level A is the first level below the context diagram and contains the main functions (defined in Section 3.1.4 as F1, F2, F3, F4 and F5) of the system developed, the *IBVS Controller*. In Figure 3.2 a Data Flow Diagram (DFD) is used to describe the interaction between data and processes. A Data Flow Diagram [**Janschek2011**] is a decomposition of the system developed resultant of the subjective perspective of the design engineer about it.

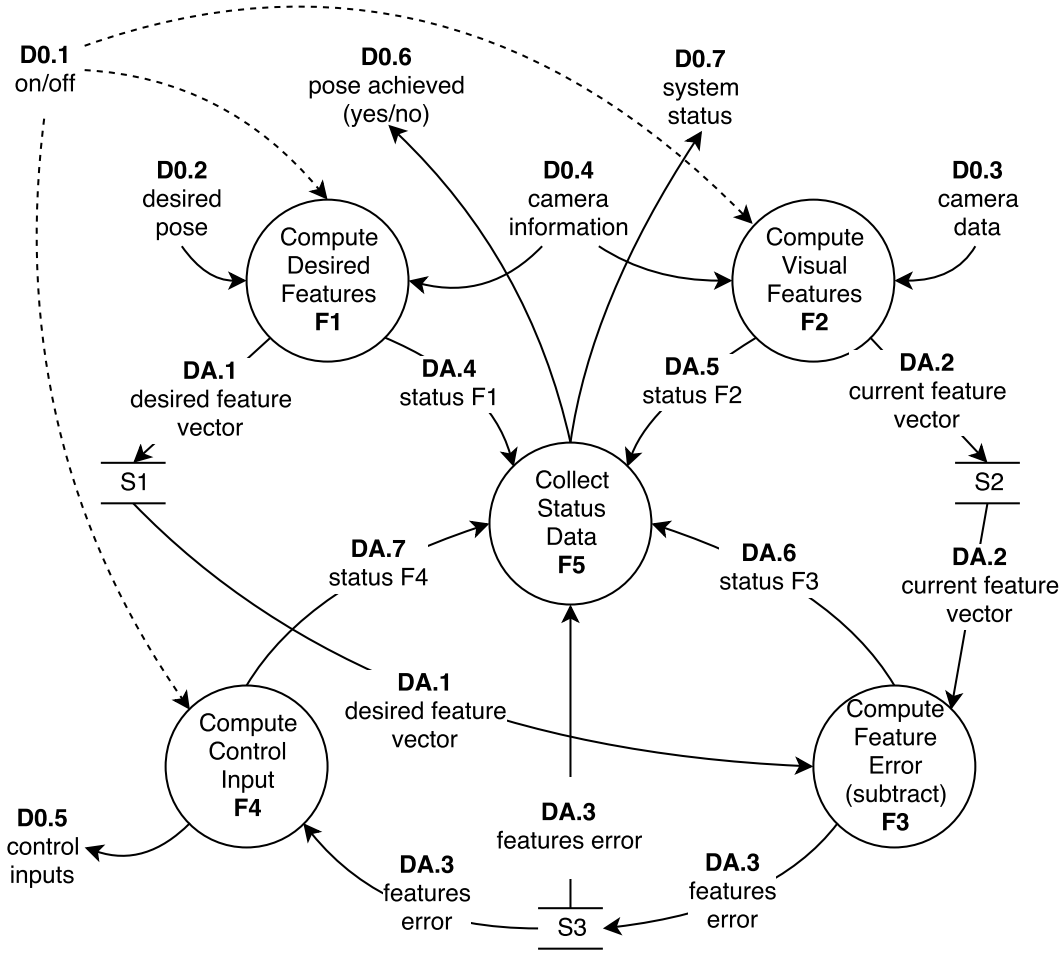

Figure 3.2: Data Flow Diagram - Level A

Table 3.1 contains the Data Dictionary for the DFD of Level A. The Data Dictionary [**Janschek2011**] is a detailed representation of all data flows involved in a Data Flow Diagram. Where D = data flow and C = control flow.

Flow	Type	Description
D0.1	C	User input to start or stop the system
D0.2	D	User defined desired pose w.r.t. the target
D0.3	D	Camera image data
D0.4	D	Camera information (sensor data, camera model, etc.)
D0.5	D	Velocity inputs for the robot
D0.6	C	Information about pose achieved or not
D0.7	D	System status presented to the user
DA.1	D	Feature vector for desired pose
DA.2	D	Current feature vector from camera image data
DA.3	D	Features error used for control law
DA.4	D	System status from process F1
DA.5	D	System status from process F2
DA.6	D	System status from process F3
S1	Datastore	Desired feature vector
S2	Datastore	Current feature vector
S3	Datastore	Features error vector

Table 3.1: Data Dictionary for Level A

3.2.3 Level B1: Compute Desired Visual Features

The following Data Flow Diagram describes the inner work of the process *F1: Compute desired visual features.*

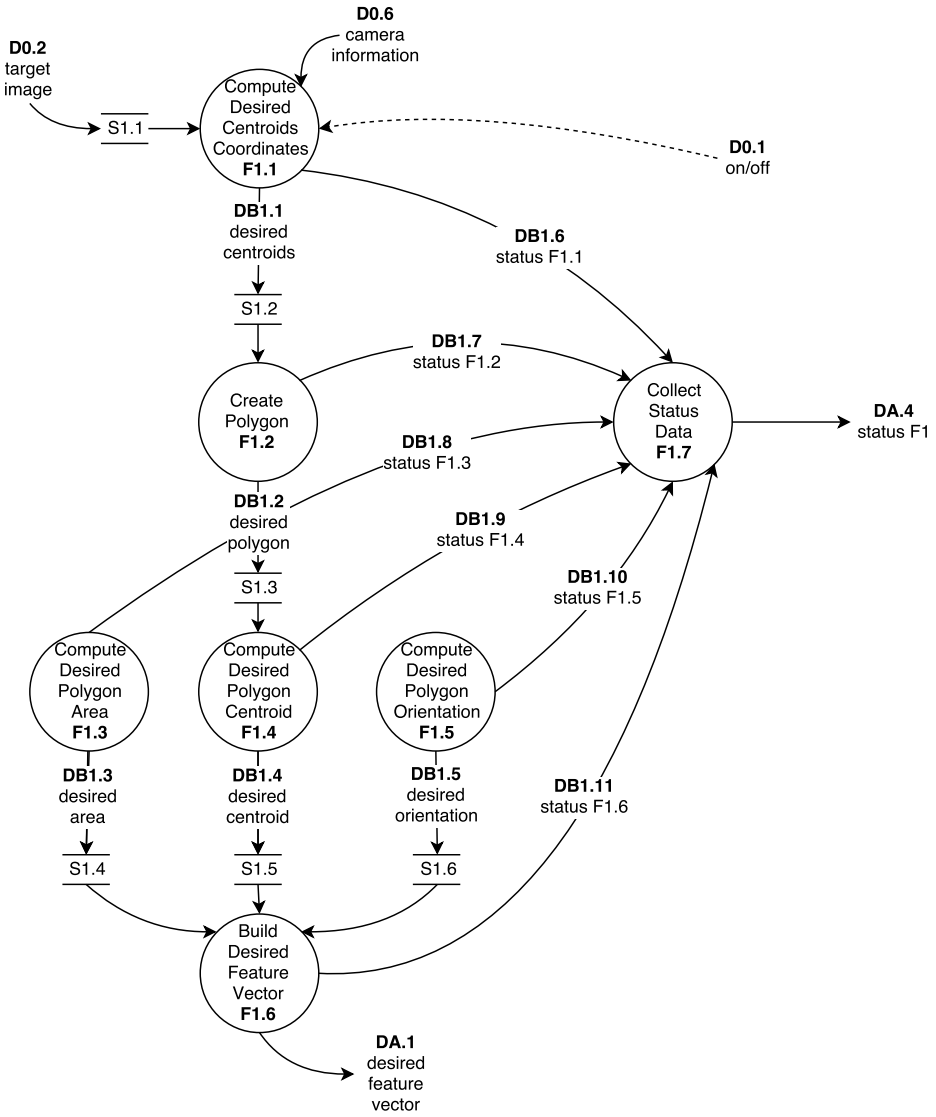

Figure 3.3: Data Flow Diagram - Level B1

3.2.4 Level B2: Compute Current Visual Features

The following Data Flow Diagram describes the inner work of the process *F2: Compute current visual features.*

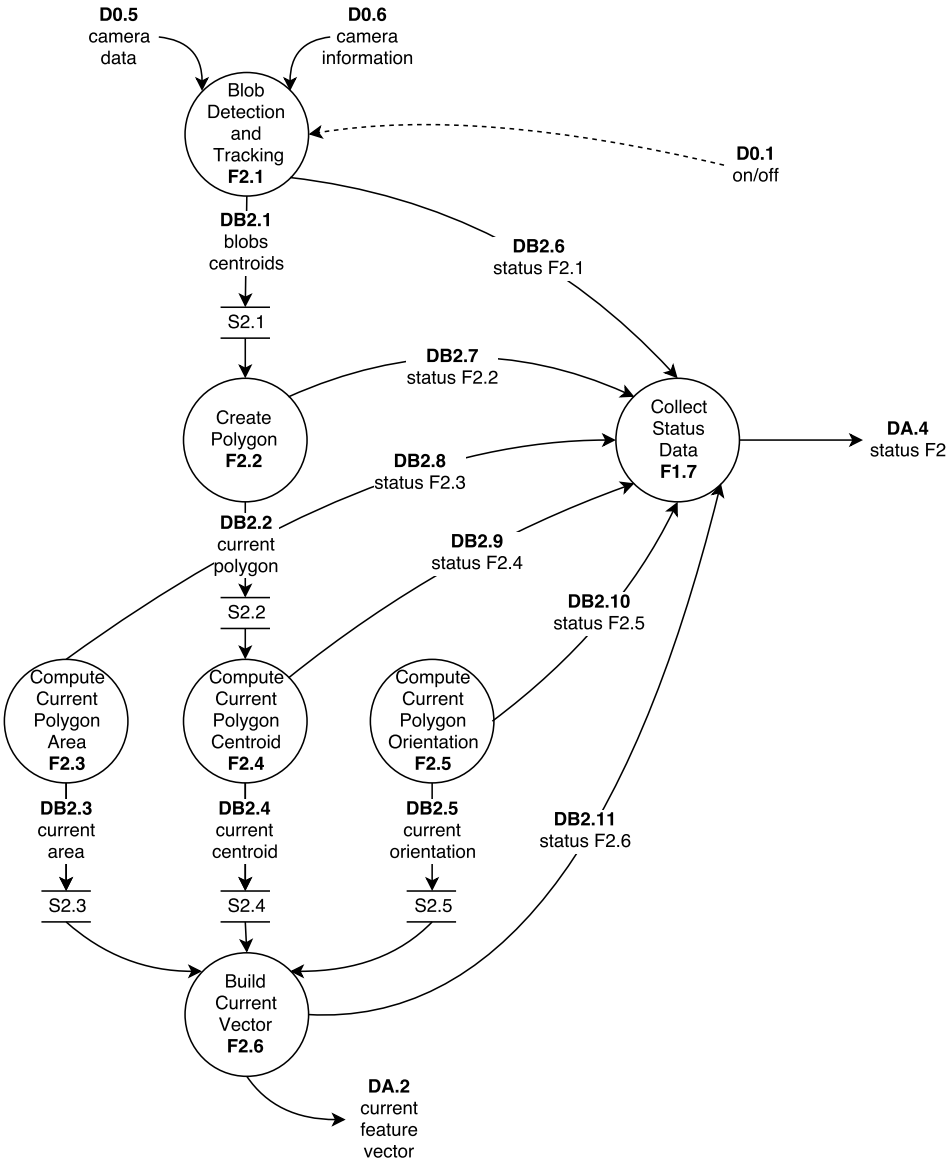

Figure 3.4: Data Flow Diagram - Level B2

3.2.5 Level B4: Compute Control Law

The following data flow describes the inner work of the process *F4: Compute control law*.

Flow	Type	Description
DB1.1	D	Vector containing the desired centroid coordinates for each blob
DB1.2	D	Desired polygon formed by the blobs
DB1.3	D	Desired polygon area
DB1.4	D	Desired polygon centroid coordinates
DB1.5	D	Desired polygon orientation angle
DB1.6	D	System status from process F1.1
DB1.7	D	System status from process F1.2
DB1.8	D	System status from process F1.3
DB1.9	D	System status from process F1.4
DB1.10	D	System status from process F1.5
DB1.11	D	System status from process F1.6
DB2.1	D	Vector containing the data from each of the detected blobs
DB2.2	D	Current polygon
DB2.3	D	Current area of the polygon
DB3.4	D	Current polygon centroid coordinates
DB2.5	D	Current polygon orientation angle
DB2.6	D	System status from process F1.1
DB2.7	D	System status from process F1.2
DB2.8	D	System status from process F1.3
DB2.9	D	System status from process F1.4
DB2.10	D	System status from process F1.5
DB2.11	D	System status from process F1.6
S1.1	Datastore	Fixed frame to desired pose homogeneous transformation
S1.2	Datastore	Desired centroid coordinates for each blob
S1.3	Datastore	Desired target polygon
S1.4	Datastore	Desired polygon area
S1.5	Datastore	Desired polygon centroid coordinates
S1.6	Datastore	Desired polygon orientation angle
S2.1	Datastore	Blob edges, gray level and centroid coordinates
S2.2	Datastore	Current target polygon
S2.3	Datastore	Current polygon area
S2.4	Datastore	Current polygon centroid coordinates
S2.5	Datastore	Current polygon orientation angle

Table 3.2: Data Dictionary for Level B

4 Visual Servoing Algorithm Description

4.1 IBVS Using Perspective Projections

In this section, an IBVS scheme for the control of the translation and kinematics of an aerial vehicle is presented (see [**bourquardez_2009**]). The implementation details of this algorithm are given in Section 5.1.

Given an object, the visual feature vector $\boldsymbol{s} = (x_n, y_n, a_n)$ is defined such that

$$a_n = Z^* \sqrt{\frac{a^*}{a}} \qquad x_n = a_n x_g \qquad y_n = a_n y_g$$

Where a is the area of the object in the image, x_g and y_g its centroid coordinates, a^* the desired area, and Z^* the desired depth between the camera and the target. The relationship between the relative motion of the camera and the object and the feature kinematics is given by

$$\dot{\boldsymbol{s}} = \boldsymbol{L}_v \boldsymbol{v} + \boldsymbol{L}_\omega \boldsymbol{\omega} \tag{4.1}$$

Here, the linear and angular velocities of the camera (expressed in the camera frame) are $\boldsymbol{v}$ and $\boldsymbol{\omega}$. The iteration matrix is separated in two matrices, $\boldsymbol{L}_v$ related to translation and $\boldsymbol{L}_\omega$ related to rotation. The desired image feature is denoted by $\boldsymbol{s}^*$, and the visual error is defined by $\boldsymbol{e} = \boldsymbol{s} - \boldsymbol{s}^*$.

As already mentioned in Section 2.1, linear exponential stability is imposed on the error kinematics to ensure an exponential decoupled decrease for $\boldsymbol{e}$ ($\dot{\boldsymbol{e}} = -\lambda \boldsymbol{e}$, with λ a positive gain). Now $\boldsymbol{e}$ can be used to control the translational [1] degrees of freedom using the following control input

[1] Remember that for the case of a quadrotor, it is an underactuated system. Thus, it is only possible to control its translation and yaw.

$$\boldsymbol{v} = -(\boldsymbol{L_v})^{-1}(\lambda \boldsymbol{e} + \boldsymbol{L_\omega \omega}) \text{ with } \lambda > 0 \tag{4.2}$$

Generally, the interaction terms $\boldsymbol{L_v}$ and $\boldsymbol{L_\omega}$ depend nonlinearly on the state of the system and cannot be reconstructed exactly from the observed visual data. To cope with this situation the system can be linearized around an equilibrium position.

For an aerial robot where the camera is pointing downwards towards the target object, it is possible to approximate $\boldsymbol{L_v} \approx -\boldsymbol{I}_3$, since the camera image plane is parallel to the target plane. Furthermore, the motion of the robot is smooth and slow, so the value of $\boldsymbol{L_\omega \omega}$ is small compared with the error $\lambda \boldsymbol{e}$. As a result, the following approximation is valid

$$\boldsymbol{v} = \lambda \boldsymbol{e} \text{ with } \lambda > 0 \tag{4.3}$$

In order to control the yaw motion of the system, it is possible to compute the rotation θ_z around z with respect to the target. Yaw control is achieved through the following law

$$\boldsymbol{\omega_z} = \lambda \theta_z \text{ with } \lambda > 0 \tag{4.4}$$

The method presented int his section has some limitations, since it depends on the geometry of the target and considers only smooth and slow trajectories. Any aggressive maneuver, or a case in which the parallel target assumption is invalidated, makes the approximations taken fail.

5 Implementation of the Visual Servoing Controller

5.1 IBVS Using Perspective Projections

6 Final Results and Conclusions

7 Future Work

Selbstständigkeitserklärung

Hiermit versichere ich, Pablo Rodríguez Robles, geboren am 28.02.1996 in León, Spain, dass ich die vorliegende Bachelor Thesis zum Thema

Image Based Visual Servoing for Aerial Robot

ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Bei der Auswahl und Auswertung des Materials sowie bei der Herstellung des Manuskripts habe ich Unterstützungsleistungen von folgenden Personen erhalten:

Dipl.-Ing. Chao Yao

Weitere Personen waren an der geistigen Herstellung der vorliegenden Bachelor Thesis nicht beteiligt. Mir ist bekannt, dass die Nichteinhaltung dieser Erklärung zum nachträglichen Entzug des Diplomabschlusses (Masterabschlusses) führen kann.

Dresden, den 27.03.2018 .....

Unterschrift