TECHNISCHE UNIVERSITÄT DRESDEN

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	Aerospace Engineering, 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

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

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

List of Figures

List of Tables

List of Listings

1 Introduction

1.1 Motivation and Background

During the last decade, the use of Unmanned Aerial Vehicles (UAVs) has spread among very different applications. The use of flying robots can be very helpful to improve the way some tasks are already achieved by terrestrial robots. 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.

In order to integrate the visual servoing algorithm into the future modular robot system, the algorithm has been designed and tested on a underactuated 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 using Gazebo[4].

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

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

[3] www.ros.org

[4] www.gazebosim.org

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) **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.

2 Theoretical Background and State of the Art

2.1 Visual Servoing Theoretical Basics

In this section the theoretical basic background of visual servo controllers is briefly discussed. 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 completely 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. On the other hand, Position Based Visual Servoing (PBVS) considers for $\boldsymbol{s}$ a set

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

of 3D parameters that must be estimated from the image data.

Using the PBVS approach leads to the necessity of camera calibration and estimation of the flying robot pose (TODO: Add reference or a bit more of information), 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.

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 we can express the temporal variation of the features vector as

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

Where $\boldsymbol{L_s} \in \mathbb{R}^{k \times 6}$, the feature Jacobian, acts as iteration 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

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

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

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

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

When implementing real systems it is not possible to know perfectly either $\boldsymbol{L_e}$ or $\boldsymbol{L_e^+}$. Thus, an approximation of these two matrices is introduced, noted with the symbol $\widehat{\boldsymbol{L}_{\boldsymbol{e}}}$ 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 $\boldsymbol{L}_{\boldsymbol{s}}$? 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 chosen.

The Interaction Matrix

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. In this case, we take as feature the image point, thus $\boldsymbol{s} = \boldsymbol{x} = (x, y)$.

Taking the time derivative of the projection equations 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 we can write

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

using matrix notation

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

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

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

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 $\boldsymbol{L_x}$, but an approximation $\widehat{\boldsymbol{L_x}}$ 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 $\widehat{\boldsymbol{L_e^+}} = \boldsymbol{L_e^+}$ for $\boldsymbol{L_e} = \boldsymbol{L_x}$ 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.

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 fulfil 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 underactuated aerial robot the commands necessary to reach a desired pose with respect to a target object.

The VS controller developed is to be integrated into the hector_quadrotor **2012simpar_meyer** an underactuated aerial robot equipped with a monocular monochrome camera pointing downwards.

3.1 Software Requiremet Specification

In this section the Software Requirement Specification **IEEE8301998** for the Visual Servoing controller developed in this thesis is presented. Using 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 used aerial robotic system is the hector_quadrotor[1] model **2012simpar_meyer** (TODO: Add diagram).

[1] wiki.ros.org/hector_quadrotor

The subsystem developed here is to interact with the camera hardware of the robot, a monocular monochrome camera pointing downwards (TODO: Add hardware). The output of the subsystem are the control inputs of the aerial robot dynamics (TODO: Add which are the quadrotor control inputs), this inputs interact with the inner control loop for the attitude and outer control loop for the position already implemented in the robotic system (TODO: Position loop is not related to vs system, but can be useful for benchmark).

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 a VS 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 platforms, forward or backward support is not guaranteed on a different set-up.

- ROS version: ROS Indigo[2]
- Operating System: Ubuntu 14.04[3] Trusty Tahr, 64 bit

3.1.4 Functional Requirements

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

- F1: *Give visual servoing control input.* Control input based on image data so that the aerial robot comes closer to the target pose. Control as a difference on the image features, no pose estimation.
- F2: *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.

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

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 manoeuvres.
- 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 target pose must be provided by a sufficient number of features (TODO: How many?) in form of a 2D code (TODO: At least in the first version).
- 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[4] (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[5] (Version 5.1.14 r112924) with 5 GB base memory and two processors.

3.2 Structured Analysis

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

4 Visual Servoing Algorithm Description

5 Implementation of the Visual Servoing Controller

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