DEGREE PROJECT

Vision Based Attitude Control

Maroš Hladký

Space Engineering, master's level
2018

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering

LULEÅ UNIVERSITY OF TECHNOLOGY

Disclaimer

This project has been funded with support from the European Commission. This publication (communication) reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the
Erasmus+ Programme
of the European Union

Abstract

The problematics of precise pointing and more specifically an attitude control is present since the first days of flight and Aerospace engineering. The precise attitude control is a matter of necessity for a great variety of applications. In the air, planes or unmanned aerial vehicles need to be able to orient precisely. In Space, a telescope or a satellite relies on the attitude control to reach the stars or survey the Earth. The attitude control can be based on various principles, precalculated variables, and measurements. It is common to use the gyroscope, Sun/Star/horizon sensors for attitude determination. While those technologies are well established in the industry, the rise in a computational power and efficiency in recent years enabled processing of an infinitely more rich source of information - the vision. In this Thesis, a visual system is used for the attitude determination and is blended together with a control algorithm to form a Vision Based Attitude Control system.

A demonstrator is designed, build and programmed for the purpose of Vision Based Attitude Control. It is based on the principle of Visual servoing, a method that links image measurements to the attitude control, in a form of a set of joint velocities. The intermittent steps are the image acquisition and processing, feature detection, feature tracking and the computation of joint velocities in a closed loop control scheme. The system is then evaluated in a barrage of partial experiments.

The results show, that the used detection algorithms, Shi&Tomasi and Harris, perform equally well in feature detection and are able to provide a high amount of features for tracking. The pyramidal implementation of the Lucas&Kanade tracking algorithm proves to be a capable method for a reliable feature tracking, invariant to rotation and scale change. To further evaluate the Visual servoing a complete demonstrator is tested. The demonstrator shows the capability of Visual Servoing for the purpose of Vision Based Attitude Control. An improvement in the hardware and implementation is recommended and planned to push the system beyond the demonstrator stage into an applicable system.

Zusammenfassung

Die Problematik des präzisen Ausrichtens und somit der Lagekontrolle ist seit den ersten Tagen der Luft- und Raumfahrttechnik präsent. Die präzise Lageregelung ist notwendig für eine große Vielfalt von Anwendungen. In der Luft müssen sich Flugzeuge oder unbemannte Luftfahrzeuge orientieren können, im Weltraum ist ein Teleskop oder ein Satellit auf die Steuerung der Lage angewiesen, um die Sterne zu erreichen oder die Erde zu beobachten. Die Lageregelung kann auf verschiedenen Prinzipien, vorberechneten Variablen und Messungen basieren. Es ist üblich, Gyroskope, Sonnen-, Stern- und/oder Horizontsensoren zur Bestimmung der Lage zu verwenden. Während diese Technologien in der Industrie bereits etabliert sind, ermöglicht die Zunahme an Rechenleistung und Effizienz in den letzten Jahren den Einsatz eines weitaus komplexeren und umfangreicheren Verfahren - dem optischen Tracking. In dieser Arbeit wird ein visuelles System zur Bestimmung der Fluglage verwendet und mit einem Steueralgorithmus zur Bildung eines Systems zur visionsbasierten Fluglagensteuerung kombiniert..

Ein Demonstrator für die visionsbasierte Fluglagensteuerung ist entworfen, gebaut und programmiert. Er basiert auf dem Prinzip des visuellen Servoing, einer Methode, die Bildmessungen mit Geschwindigkeiten für Manipulatorgelenke verknüpft. Einzelne Schritte sind die Bilderfassung und -verarbeitung, die Detektion und das Verfolgen von Merkmalen im Bild und die Berechnung der rotatorischen Geschwindigkeiten für die Antriebe. Das System wird anschließend mit Hilfe von verschiedenen Experimenten analysiert und bewertet.

Die Ergebnisse zeigen, dass die verwendeten Algorithmen zur Erkennung von Mermalen von Shi&Tomasi und Harris bei der gleich gut abschneiden und eine Vielzahl von Merkmalen für das anschließende Verfolgen liefern können. Die pyramidiale Implementierung von Lucas&Kanade Verfolgungsalgorithmus erweist sich als eine geeignete Methode für ein zuverlässiges Verfolgung der Merkmale, unabhängig von Drehungen und Maßstabsänderung. Um das visuelle Servoing weiter zu evaluieren, wird der komplette Demonstrator getestet. Der Demonstrator zeigt die Fähigkeit von visuellem Servoing für die visionsbasiertebasiert Fluglagensteuerung. Eine Verbesserung der Hardware und der Implementierung wird empfohlen und ist geplant, um das System weiterzuentwickeln und für eine reale Anwendung vorzubereiten.

Contents

Chapter 1

Introduction

Vision Based Attitude Control (VBAC) is an ongoing research topic within the realms of Aerospace engineering. In a figurative sense, the problem of VBAC was present from the first human controlled flight efforts. The eye as vision sensor, and brain and hands as a controller, were the only available system for VBAC for a long time. Thanks to the technological progress of the last few decades, computers are now able to support humans or even take the task completely.

In order for a robotic system to autonomously achieve its task, sensing and providing data in a reliable way to the control system is crucial. A combination of sensors are used, such as the Global positioning system (GPS), the more local sensors like range sensors, or relative ones like the inertial measurement unit. Those methods come with their own set of benefits and problems. Content of the received measurements is mostly straight forward, does not imply complicated extraction of the desired data, it is directly processable. On the other hand this content can be lost or not usable due to noise or malfunction. Inertial system's drift or bad reception of GPS data in high latitudes are only few of the problems. Nonetheless, for the most part of technological history, attitude is determined and controlled by measurements of these comparatively comprehensible means, mostly their combinations.

The environment in which most air/spacecrafts operates is infinitively more rich than mentioned technologies can ever register. Vision systems offer another possible, rich way to sense the environment. Using vision for attitude determination and control is promising, but challenging. Camera offers very precise, stable and reliable 2D measurements of the environment. Cameras contain none or only a few movable parts and therefore are less prone to failures compared to mechanical gyroscopes for example. Using cameras for the attitude control offers wide range of usage. Unmanned aerial vehicles (UAV), satellites, underwater robots or practically any other autonomously operated robot benefit strongly from vision based control. Keeping the attitude control in focus, more specific examples can be named. Small satellites might benefit even further from vision based control. Reliable cameras are often already implemented into design of small spacecrafts for observation. Using the same camera for attitude control means to be able to exclude other sensors and save space, expensive mass and potentially electrical power. More specific examples that will benefit from the VBAC in the future are Space debris recognition

and removal, or the idea of Space rendezvous, such was designed for *Automated Transfer Vehicle* (ATV) when providing service tasks to the *International Space Station* (ISS) [13]. Satellite communication is another field where VBAC research is of a great value. High data transfer in satellite communications in near visible spectrum is currently strongly researched and will undoubtedly rise in use. It relies on narrow beamwidth and therefore requires precise attitude control. Vision systems are ideal for such a purpose since VBAC poses the capability to reach sufficient pointing accuracy within a reliable and mechanically simple and robust way. To fully enable complex attitude control tasks like those mentioned, captured images needs to be analyzed and understood to provide sensible data for the control system. The complexity of images was preventing wide immersion of computer vision into practical tasks in the past. Consequently the optional decrease in mechanical intricacy thanks to substitution of sensors by immovable cameras and increase in capabilities, is balanced by increased algorithmic and software complexity.

This thesis will grasp the VBAC in a specific and practical manner. The focus will be shifted towards *Visual servoing* (VS), a technique encapsulating feature extraction used to control a robot in a feedback loop. VS techniques will be used to control the orientation of a camera attached to a robotic manipulator. Even though no attitude control in a strict sense of speaking is within the scope, the principles that will be shown in this thesis are transferable to it nonetheless.

1.1 Previous Work

Pure VBAC is a complicated method only seldom used in its full capability. Vision is comparably often used for manipulator or camera pose estimation, rather than for the full attitude control. No matter that, the attitude determination and its loose synonyms such the mentioned pose estimation or camera position, are essential step towards successful control, and so work done within those fields is briefly mentioned.

Certain other techniques [14], [20] rely on detection of vanishing points. Especially in an urban environment, since those are usually build with parallel lines that appear to converge in vanishing points. Those methods often rely on either clear visibility to the horizon or at least to the sky. Techniques based on the horizon detection allows for only partial attitude determination, since yaw angle is often impossible to estimate. In [33], authors propose an attitude estimation based on polarization. Within the method, the image is segmented by the degree of polarization and the angle of polarization in order to extract the horizon.

It comes as no surprise, that vision based control is intensively examined for Space applications. Yet again mostly in a theoretical manner, without full scale application, but with immense potential for the future. In [23], authors describe an automated system that derives satellite attitude by combining both observed image and known position by matching base map features to features from said images. The authors propose to use SURF descriptors to match feature pairs from the base known image and the satellite image. This method is claimed to achieve accuracy within 0.02°, roughly equal to a star sensor capability. Another research team [1] investigates

the option of spacecraft self localization and attitude control based on landmarks detected on unresponsive spacecraft. The method proposed is based on an active localization approach, data gathered by orbiting the target are used to map it through application of SLAM. The landmark estimation accuracy is captured in a cost function based on the Extended Kalman Filter. The optimization of the cost function is performed using cross entropy minimization. Actual efforts to control the attitude are proposed in [22]. The authors propose image based control law based on the SIFT image feature description and matching. Their goal is an automatic satellite steering to a reference point observed on the Earth's surface. Control of the attitude is performed based on feature pairs between the images from simulated satellites and reference images. The author's work shows particular similarity with the content of this thesis. Direct image based control of the end effector, based on feature detection is common to both projects, as it will be shown in later chapters of this Thesis. The differences are numerous however. While the authors use a completely simulated environment, this Thesis will focus on real hardware, although not for direct Space application. The authors further expects big differences between consecutive images compared to small displacement assumed in this Thesis. This comes from the images obtained at orbital velocities and operation frequency of $10hz$, what inevitably translates into significant differences between pictures. Therefore the authors use feature matching with reference image, unlike continuous frame to frame minimization of sum of square differences (SSD) which is used within this Thesis.

Another example of VBAC, that might soon result in an actual real application is proposed in [40], as a part of *RemoveDEBRIS* project. An algorithm build on a frame to frame tracking of a known object is proposed. Detected features are combined to align a 3D model projection to the observed target. From this alignment, a camera pose is estimated. The alignment and the source 3D model is shown on Fig. 1.1.

Perhaps the most crucial example of VBAC was designed for the docking procedure of *Automated Transfer Vehicle* (ATV) cargo carrier with the *International Space Station* (ISS). Hardware designed for the docking - *Videometer* [13], emitted a laser beam towards back part of Russian part of ISS where the docking was planned. Triangular and pyramidal set of retroreflectors mounted on the *Zvezda module* reflected the laser back to the ATV. The videometer analyzed the image and estimated the range and orientation. The onboard controller was able to compensate for sensor noise, cargo sloshing, ISS attitude motion or the flexibility of ATV solar panels.

1.2 Goals and Outline

VBAC is a very promising area of research that is still only seldom used in real applications. VBAC, just like every other new technology striving to become a norm, needs to show its feasibility during a barrage of tests in theoretical and later physical form. The goal of this thesis is to design such a demonstrator and implement the techniques of visual servoing for application in the attitude control. The demonstrator functioning as a proof of concept will be build into a simple handheld form. It will demonstrate the platform capability for reliable detection of

Figure 1.1: Example of the camera pose estimation from known 3D model. Left: 3D model, right: alignment of the model with content of the camera image [40].

trackable features, realtime tracking in the image stream and the attitude control with the aim to keep the tracked features within the centre of the image. Though there is already a panoply of research done on the topic, it is to our best knowledge that the demonstrator presented in this thesis is unique in functionality and approach.

The work done on the thesis is outlined for the reader's understanding in the six main chapters. Those are structured as follows

Chapter 2 introduces the theoretical knowledge that precedes the techniques and methods used in the Thesis. Theoretical background of the way how the 3D world is mapped to 2D through camera model, or how can a common stream of images control the movement of a robot manipulator. Chapter further continues by characterization how the features are detected in the image and what those features actually are, how are those features tracked from frame to frame and how the velocities in the camera frame are related to the robot joint velocities. At the end of chapter the software and hardware tools that made the whole Thesis possible are described briefly.

Chapter 3 displays the designed demonstrator structure put together, the requirements, the design choices done and explains the overall mechanical functionality.

Chapter 4 describes the software implementation of Visual Servoing. The complete program chain from the image, through features to joint velocities is explained on each functional program element - the node.

Chapter 5 describes the experiments that are performed in order to evaluate the capability of each individual section: Feature detection, Visual Servoing performance and at last the feasibility experiment of the Demonstrator.

Chapter 6 provides a short overview of the obtained results, concludes the Thesis and discuss the potential improvements and future work.

Chapter 2

Theoretical Background

This chapter describes the entire theoretical and technical background required throughout the Thesis. The chapter starts with camera model in Sec. 2.1. The following Sec. 2.2, provides an overview of the key mathematical apparatus applied in this Thesis - Visual Servoing (VS). Sections 2.3 - Feature detection, 2.4 - Feature tracking and Sec. 2.5 explain the pieces upon which the VS is build. Finally the last Sec. 2.6 portrays the choice of software and hardware tools used.

2.1 Camera Model

Every camera in its most elementary form is a device, that projects from 3D world coordinates onto 2D image plane. The simplest model of this projection is the basic pinhole camera model.

The Pinhole Camera. The pinhole camera model is the simplest model accounting for no lenses or obstacles in the path of light. It will be described as it was introduced in [18]. Fig. 2.1 of this model illustrates the basic concept of the model. Let the *camera centre* $\boldsymbol{C}$ be the origin of Euclidean coordinate system. A plane called the *image plane* is in a distance f the *focal length* from $\boldsymbol{C}$, perpendicular to a *principal axis* Z. A point at the intersection of the *principal axis*

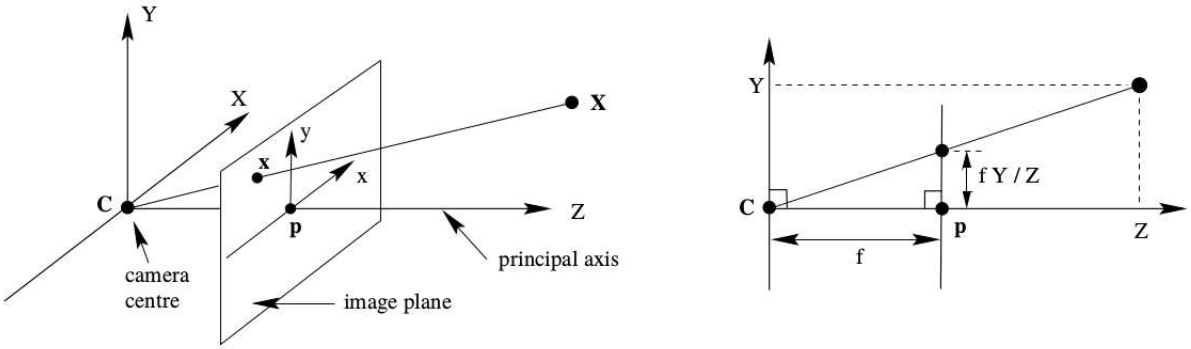

Figure 2.1: Pinhole camera model, illustration and parameters [18].

with the *image plane* is called *principal point*. Given this model, any point $\boldsymbol{X} = (X, Y, Z)^T$ is mapped from Euclidean $\mathbb{R}^3$ to Euclidean $\mathbb{R}^2$, i.e. world to image plane, as

$$(X, Y, Z)^T \mapsto \left(\frac{fX}{Z}, \frac{fY}{Z}, \right)^T. \tag{2.1}$$

Lets express the world and image point as homogeneous vectors, then Eq. (2.1) can be expressed as a matrix multiplication and becomes

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{2.2}$$

In reality the principal point and the origin of the image plane are not coincident. Consideration of the offset of the principal point with respect to the bottom left corner is needed. Then the general form of Eq. (2.2) accounting for the offset is

$$(X, Y, Z)^T \mapsto \left(\frac{fX}{Z} + p_x, \frac{fY}{Z} + p_y, \right)^T = (u, v)^T, \tag{2.3}$$

where (p_x, p_y) and (u, v) are the coordinates of principal point and projected point $\boldsymbol{x}$ in the image plane. This can be again expressed in homogeneous coordinates as

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX + p_x \\ fY + p_y \\ Z \end{pmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{2.4}$$

The matrix

$$K = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \tag{2.5}$$

is the *camera calibration matrix*, that describes the physical parameters of the camera.

2.1.1 Lens Distortion

The pinhole camera model does not include lens distortion of the camera. Nothing in the world is ideal, therefore the camera model needs to account for distortions. For example, if manufactured lenses are geometrically imperfect or are not perfectly aligned with the chip. A function describing the distortion is generally dominated by the radial components [41]. Moreover, more complicated distortion modeling would cause numerical instability [39]. A description of the radial distortion by [18] is briefly presented.

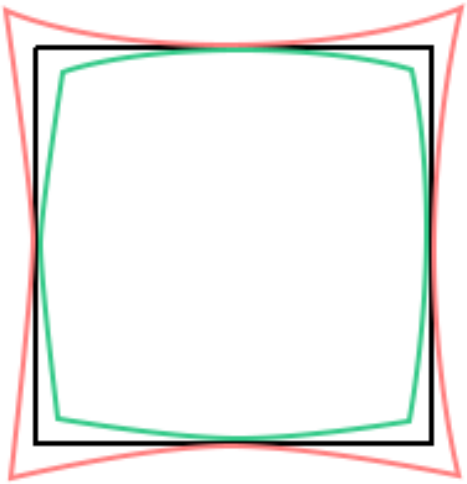

Figure 2.2: Geometrical camera distortion: Barrel (green) and Pincushion (red)

Let $(\tilde{x}, \tilde{y})$ be the ideal coordinates of a pixel in the image plane and (x_d, y_d) the corresponding distorted ones. Then the actual distorted point is related to the ideal one by

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = L(\tilde{r}) \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix},$$

where $L(\tilde{r})$ is the distortion factor, that is a function of radial distance $\tilde{r} = \sqrt{\tilde{x}^2 + \tilde{y}^2}$ from the centre of the image. Corrected coordinates $(\hat{x}, \hat{y})$ are then written as

$$\hat{x} = x_c + L(r)(x - x_c)$$
$$\hat{y} = y_c + L(r)(y - y_c),$$

where $r^2 = (x - x_c)^2 + (y - y_c)^2$ and (x_c, y_c) is the centre of the image. The actual distortion illustration is shown on Fig. 2.2.

2.2 Visual Servoing

Robotic systems have been living through an upswing in the recent years with a seemingly brighter future. Higher demands on the output side of this equation needs to be balanced with progress in the form of faster pneumatics, processing power or sensory capabilities and automation. While the two former ones far exceed human capabilities, understanding and a seamless fusion with the World is still lagging behind. VS provides a link between the visual representation of the reality and the robot's environment. It refers to the use of computer vision data as an input to control the motion of a robot in a closed loop control scheme [10]. In other words, it is a multi-topic field, spanning image processing, computer vision and control engineering.

Most, if not all visual servoing tasks enhances the iterative control loop. The following list exemplifies the steps taken by such a conjunction:

- Image acquisition;
- Extraction of useful data from the image → features;
- Computation of current features;
- Computation of error $\boldsymbol{e}$ between current and desired feature;
- Update of control law in order to minimize the error $\boldsymbol{e}$.

2.2.1 Classification

VS varies by the scheme of how $\boldsymbol{s}$, the set of visual features, is defined [9] and by the configuration of the camera with respect to the robot [19].
The first major classification splits VS into:

Image Based Visual Servoing (IBVS), proposed in [32], computes the control values on the basis of image features directly. This eliminates the delay related to image interpretation and errors caused by camera calibration [19]. Image features are usually defined by coordinates of the image plane in pixels. An example is shown in Fig. 2.3, an internal view of a simulation of orbiting satellite using features of Sardinia for the attitude control.

Position Based Visual Servoing (PBVS) is more complicated. Features are indirectly obtained from image measurements and are used in conjunction with known 3D model of the target to estimate the pose between the camera and the target. Error $\boldsymbol{e}$ is eliminated in estimated pose space. Fig. 2.4 shows control of camera position so that the actual detected (green) object aligns with the 3D model (blue).

Hybrid or **2 1/2D VS** [26] benefits from both IBVS and PBVS by eliminating the drawbacks of both. It does not require 3D model and allows depth estimation.

Second division is defined by the camera mounting configuration:

Eye in hand (EiH) is represented by a camera being mounted directly onto the end effector in a fixed or mobile manner. It may not provide the end point closed loop control due to the mounting and field of view (FoV).

Eye to hand (EtH) means a camera is mounted somewhere on the fixed frame, again either fixed or mobile (e.g. pan-tilt unit). It does provide closed loop control up to the end effector end point, but only in certain work space due to occlusion by robot itself or other structures.
Both configurations are illustrated in Fig. 2.5.

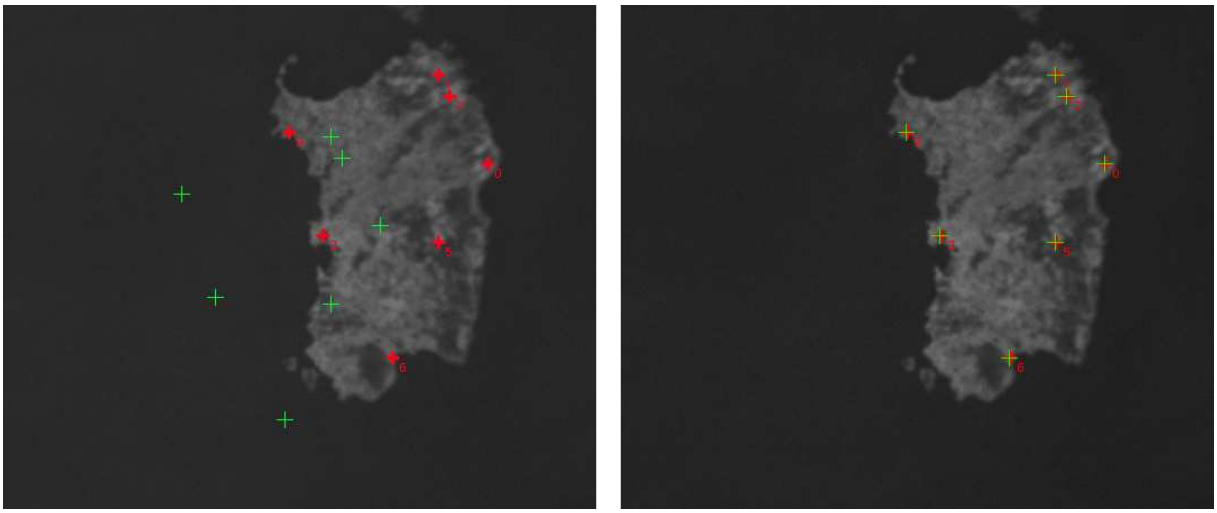

Figure 2.3: IBVS: A simulation of an attitude control on an orbiting satellite based on the actual tracked features (red) and the desired features (green).

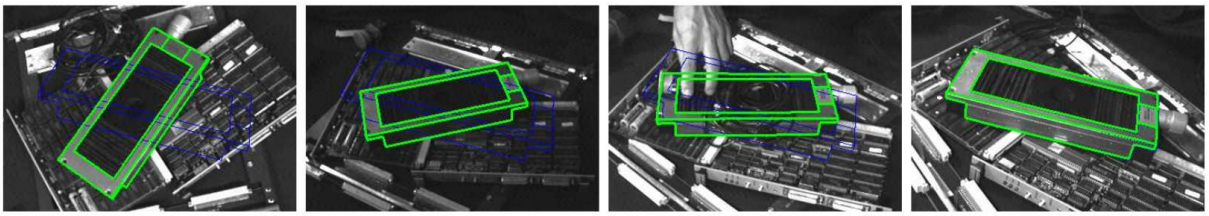

Figure 2.4: PBVS: Desired (blue) and detected (green) model pose in the camera view sequence [8].

2.2.2 Fundamentals

The basic mathematical scheme behind VS used directly in the closed loop scheme was first proposed in [15] during a time, when technological progress reached the level of sufficient processing power. The main component behind VS, as described in [15], is in the first steps independent of configuration or scheme. Most VS tasks are described by minimization of an error $\boldsymbol{e}$ between a current $\boldsymbol{s}$ and a desired $\boldsymbol{s}^*$ set of image features

$$\boldsymbol{e} = s(m(r(t)), a) - s^*(t), \tag{2.6}$$

$\boldsymbol{s}$ is build from image measurements $\boldsymbol{m}(\boldsymbol{r}(t))$, that contain meaningful information extracted from the image (e.g. 2D / 3D point coordinates, edges, area, or it's center of gravity (CoG)). Image measurements $\boldsymbol{m}$ depends on the pose $\boldsymbol{r}$ of the camera in the task space. The parameter $\boldsymbol{a}$ is the additional set of parameters, such as a 3D model of tracked object or camera intrinsic parameters.
To successfully control the robot, right side of Eq. (2.6) means that $\boldsymbol{s} - \boldsymbol{s}^*$ must be minimized. Since $\boldsymbol{s}^*$ is constant (or at least known in seldom cases), $\boldsymbol{s}$ is the controlled variable. Let $\dot{\boldsymbol{s}}$ be the time variation of the feature set and $\boldsymbol{v}_c$ the camera velocity screw. Their relation is then

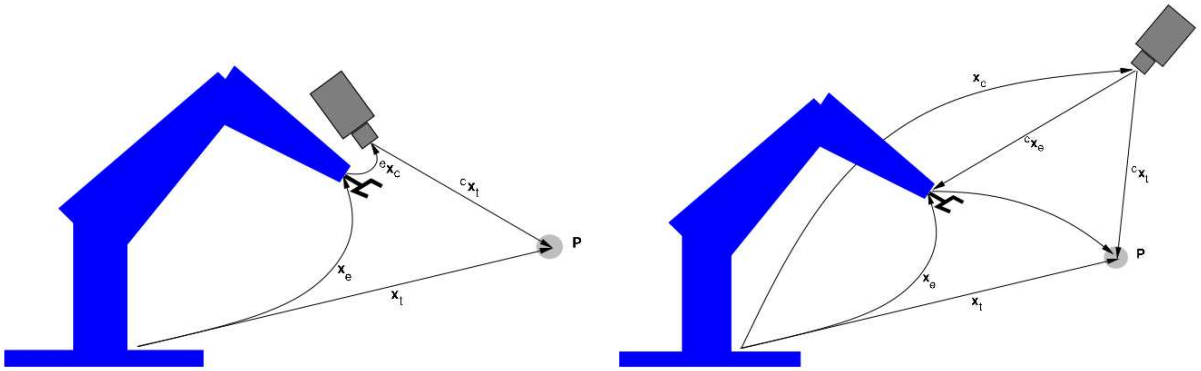

Figure 2.5: Camera - End effector manipulator configurations. Left: EiH, right: EtH [19]

defined by [11]

$$\dot{\boldsymbol{s}} = \boldsymbol{L_s} \boldsymbol{v}_c, \tag{2.7}$$

where $\boldsymbol{L_s}$ is the *interaction matrix*. $\boldsymbol{L_s}$ depends on the projection model, the number of features, and controlled variables (joint velocities $\dot{\boldsymbol{q}}$). If IBVS, control of classic 6 degrees of freedom (DoF) (instantaneous linear velocity $\boldsymbol{v} = (v_x, v_y, v_z)$, instantaneous angular velocity $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z,)$) and perspective projection is assumed, then for one feature $\boldsymbol{x} = (x, y)$, (x,y are the coordinates in image plane) the interaction matrix is defined as follows

$$\boldsymbol{L_x} = \begin{bmatrix} -1/Z & 0 & x/Z & xy & -(1+x^2) & y \\ 0 & -1/Z & y/Z & 1+y^2 & -xy & -x \end{bmatrix}, \tag{2.8}$$

where Z is the depth of feature point relative to the camera. $\boldsymbol{L_s}$ grows by stacking interaction matrices $\boldsymbol{L_{x_n}}$ of individual feature points $\boldsymbol{x_1}$,$\boldsymbol{x_2}$,...$\boldsymbol{x_n}$.

Eq. (2.7) is sufficient for idealistic Visual servoing tasks. In most other applications, the camera pose $\boldsymbol{r}$ is determined by joint velocities. Eq. (2.7) becomes

$$\dot{\boldsymbol{s}} = \boldsymbol{J_s}\dot{\boldsymbol{q}} + \frac{\partial \boldsymbol{s}}{\partial t}, \tag{2.9}$$

$\partial \boldsymbol{s}/\partial t$ is the variation due potential motion of object or camera and $\boldsymbol{J_s}$ is the *feature Jacobian*. $\boldsymbol{J_s}$ is a linear transformation from the joint space to feature set space. In mathematical notation of EiH scheme, it is written as [27]

$$\boldsymbol{J_s} = \boldsymbol{L_s}\,{}^{\boldsymbol{c}}\boldsymbol{V_e}\,{}^{\boldsymbol{e}}\boldsymbol{J_e}(\boldsymbol{q}), \tag{2.10}$$

where ${}^{\boldsymbol{e}}\boldsymbol{J_e}(\boldsymbol{q})$ is the *robot Jacobian*, or more often addressed as the *geometric Jacobian* (GJ), expressed in the end effector frame. GJ maps joint velocities into the end effector velocities in task space. The construction of GJ is further described at Sec. 2.5. Matrix ${}^{\boldsymbol{c}}\boldsymbol{V_e}$ maps velocity screw from end effector frame to camera frame. It is build from the rotation matrix ${}^{\boldsymbol{c}}\boldsymbol{R_e}$ and skew matrix of translation vector $[{}^{\boldsymbol{c}}\boldsymbol{t_e}]$

$${}^{\boldsymbol{c}}\boldsymbol{V_e} = \begin{bmatrix} {}^{\boldsymbol{c}}\boldsymbol{R_e} & [{}^{\boldsymbol{c}}\boldsymbol{t_e}] \times {}^{\boldsymbol{c}}\boldsymbol{R_e} \\ \boldsymbol{0}_3 & {}^{\boldsymbol{c}}\boldsymbol{R_e} \end{bmatrix}. \tag{2.11}$$

If a motionless target is considered, the derivation $\partial \boldsymbol{s}/\partial t$ becomes zero. Putting Eq. (2.10) into Eq. (2.9) yields

$$\dot{\boldsymbol{s}} = \boldsymbol{L_s}{}^c\boldsymbol{V_e}{}^e\boldsymbol{J_e}(\boldsymbol{q})\dot{\boldsymbol{q}}. \tag{2.12}$$

It is usually desired within the field of VS to compute joint velocities based on feature set, requiring to reverse the Eq. (2.12). By definition, $\dot{\boldsymbol{s}}$ can be written as error $\dot{\boldsymbol{e}}$, and reversing (2.12) renders general control law for EiH configuration

$$\dot{\boldsymbol{q}} = -\lambda \boldsymbol{J_s^+ e} = -\lambda \left(\widehat{\boldsymbol{L_e}}{}^c\boldsymbol{V_e}{}^e\boldsymbol{J_e}\right)^+ \boldsymbol{e} \tag{2.13}$$

where the change of error $\dot{\boldsymbol{e}}$ is expressed as $\dot{\boldsymbol{e}} = -\lambda \boldsymbol{e}$, what allows for exponential decrease of error [9]. Notice the J_s^+ is the general form of pseudoinverse matrix J_s. If the number of features $n = m$, the number of joint variables and J_s is nonsingular, then J_s^{-1} exists. If $n \neq m$, then J_s^{-1} does not exist and assuming J_s is full rank, Eq. (2.13) changes to

$$\dot{\boldsymbol{q}} = -\lambda \boldsymbol{J_s^+ e} + (\boldsymbol{I} - (\boldsymbol{J_s^+ J_s}))\boldsymbol{b}, \tag{2.14}$$

where $\boldsymbol{b}$ is an arbitrary vector and $\boldsymbol{I}$ is the identity matrix. In the case that $n > m$, then there are $n - m$ redundant features and the pseudo inverse is defined as

$$\boldsymbol{J_s^+} = (\boldsymbol{J_s}^T\boldsymbol{J_s})^{-1}\boldsymbol{J_s}^T, \tag{2.15}$$

and $(\boldsymbol{I} - (\boldsymbol{J_s^+ J_s})) = 0$ due to $m = rank(\boldsymbol{J_s})$. Eq. (2.14) is then simplified to

$$\dot{\boldsymbol{q}} = -\lambda \boldsymbol{J_s^+ e}. \tag{2.16}$$

For under-constrained system $n < m$, there is not enough features to determine $\dot{\boldsymbol{q}}$. The pseudoinverse matrix is defined as

$$\boldsymbol{J_s^+} = \boldsymbol{J_s}^T(\boldsymbol{J_s}\boldsymbol{J_s}^T)^{-1}. \tag{2.17}$$

Vectors $(\boldsymbol{I} - (\boldsymbol{J_s^+ J_s})) \neq 0$ lie in the null space of J_s. This problem of missing features and so the null space of J_s is elevated in hybrid methods, where part of the joint variables are computed through VS and part using other technique [6].

Another observable difference in Eq. (2.13) is the approximated form of interaction matrix $\widehat{\boldsymbol{L_s}}$. Depth Z of a feature point in the expression (2.11) cannot be estimated directly by one camera. Also, the intrinsic camera parameters influence the computation of feature image coordinates x,y due to perspective projection. Thus L_x (inherently L_s) cannot be computed directly and is only approximated. [7] specifies the options for approximation: $\widehat{\boldsymbol{L_s}}(\boldsymbol{s}, \boldsymbol{r})$ where the interaction matrix is computed for the current features $\boldsymbol{s}$, or $\widehat{\boldsymbol{L_s}}(\boldsymbol{s^*}, \boldsymbol{r^*})$ where the interaction matrix is computed only once for the desired features $\boldsymbol{s^*}$, or $\widehat{\boldsymbol{L_s}} = 1/2(\widehat{\boldsymbol{L_s}}(\boldsymbol{s}, \boldsymbol{r}) + \widehat{\boldsymbol{L_s}}(\boldsymbol{s^*}, \boldsymbol{r^*}))$ [25].

Eq. (2.13) provides a direct link to control the robot solemnly based on data from the camera. It is a complex and powerful technique. But it is only as good as it's partial components: **Feature Detection and Tracking** that provides $\boldsymbol{s}$, $\boldsymbol{s^*}$ the sets of current and desired features (Sec. 2.3,2.4), and the **Geometric Jacobian** ${}^e\boldsymbol{J_e}$ (Sec. 2.5) that links the end effector frame to the joint variables $\dot{\boldsymbol{q}}$.

2.3 Feature Detection and Description

The detection of stable features in an image stream is one of the key points in image processing and consequently, VS. To recognize whether a point in the image is part of a feature necessitates the capability of a certain level of decision making. The predominant question is how to decide what is or is't a feature. The way human intuition does this, does not mirror in a form applicable in the computer vision. Our mind has a billion years evolutionary advantage. It is hard to find a discrete definition of what a feature is, nor what it should consist of. At the same time, it can be stated that a reasonable requirement shared by all features is repeatability of detection. Single exceptionally bright or dark pixel, though recognizable and probably traceable, is not sufficient for non-basic application. Blobs, edges and keypoints are commonly used representations of features. Blobs and edges inevitably contain higher information content than corners. However, taking into account the infinitively complicated structure of the Earth's surface, one can hardly find structures to track other than keypoints. Ergo this Thesis focus on the detection and tracking of corners.

Computer vision problems are inherently depending on the multiplicity of related images or the image stream. In other words, to find correspondences between images. Any noise or small discrepancies should not influence this detectability. Therefore, features are described by properties linked to the patches of pixels in the area of interest. Examples of these patches are shown on Fig. 2.6. One can quickly distinguish correspondences between the pair from the mountain top. On the on the other hand it is not possible to locate the textureless sky patch. The patch from the shadow boundary is an example of the aperture problem. This patch can be placed with certainty only in direction of the highest gradient change, but cannot be localized along the edge. One's intuition would suggest, that patches with a rich texture are preferable. In fact, several pioneers in the field of computer vision settled upon the idea of patches with high spatial frequency. Moravec [29] propose to use patches where the spatial intensity profile has high standard deviation, by comparing SSD. Many other methods are based on similar basics. Harris & Stephens [17], base the detection on autocorrelation matrix (2.24) of image gradients and their product (Sec. 2.3.1).

When going more into details, a further clarification of potential methods is needed. Keypoint detection and finding their correspondences is done by two main approaches. One detects features in the image and tracks them in the continuous stream of image data that follows - detect then track. The other detects features in all available images, that are not necessarily acquired in continuous fashion from one stream, searches for the correspondences and matches them based on their local appearance [37]. This second approach is rendered not entirely applicable because the IBVS used for VS relies on steadily tracked features. Harris&Stephens [17], Shi&Tomasi [34] propose slightly different methods to robustly detect features for such a purpose. These methods, based on the SSD are presented in the following Sec. 2.3.1.

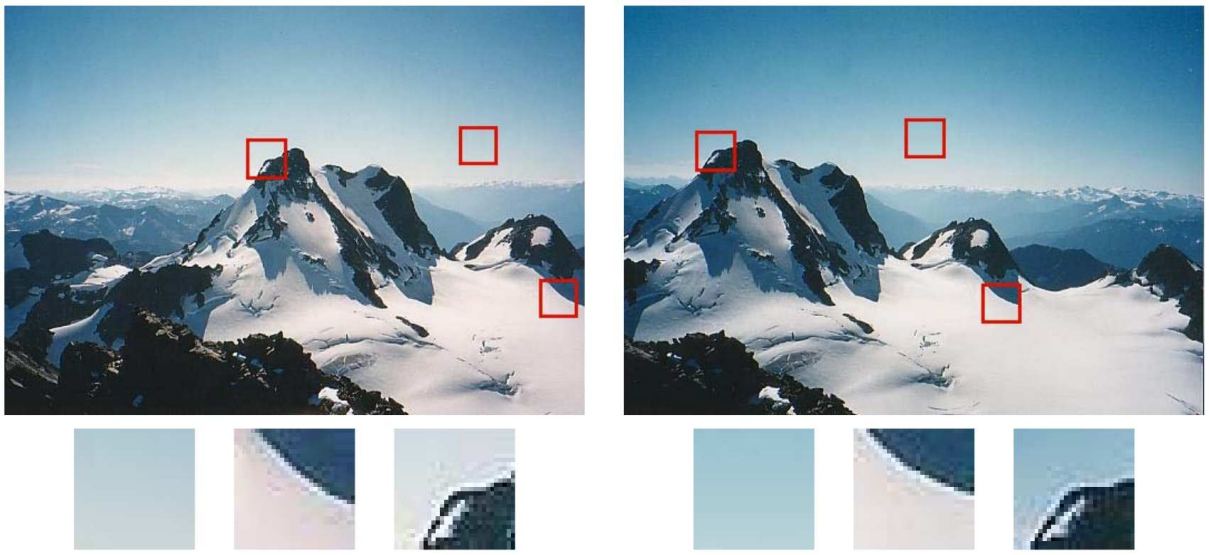

Figure 2.6 Example of feature detection in various image patches: a scenery and cutout image pairs. The sky cutout is not possible to localize, whereas the cliff edge is easily distinctive [37].

2.3.1 Harris / Shi&Tomasi corner detector

Let $I(x, y)$ be an intensity function for a certain point of an image, $\boldsymbol{u} = (u, v)$ is a displacement vector of point $\boldsymbol{x} = (x, y)$, $I(\boldsymbol{x_i} - \boldsymbol{u}) = I(x+u, y+v)$ denotes the shifted intensity in the original point and $w(x, y)$ is a weightening function over the window. Combining this together creates Eq. (2.18), a mathematical representation of Moravec's corner detector, the basic element in numerous feature detectors

$$E(u, v) = \sum_{x,y} w(x, y)[I(x + u, y + v) - I(x, y)]^2. \tag{2.18}$$

By summing all squared differences of the shifted and the original pixel in the window, one can obtain a measure of change $E(u, v)$. After applying Eq. (2.18) to the whole image, one can obtain the local maxima in $min(E)$ above a threshold and proclaim those maximas as corners. Notice, that for the purpose of feature detection, identical patches are compared against each other, and thus it is called *autocorrelation function.*

This basic method comes with list of problems as shown in [17]. E.g. the response to shifting is anisotropic due to the discrete displacement vector $\boldsymbol{u} = \{(1, 0), (1, 1), (0, 1), (-1, 1)\}$, and it is noisy due to binary window. The later problem is simply addressed by weighting by a Gaussian function:

$$w(x, y) = exp\left(-\frac{(x^2 - y^2)}{2\sigma^2}\right). \tag{2.19}$$

The first issue is mitigated by covering all small possible shifts of the $I(x + u, y + v)$. This is done by Taylor series expansion about the shifting origin, truncated to a linear term

$$I(x + u, y + v) \approx I(x, y) + I_x u + I_y v. \tag{2.20}$$

The terms I_x and I_y are the image gradients. Putting Eq. (2.20) into Eq. (2.18) yields

$$E(u,v) = \sum_{x,y} w(x,y)[I(x,y) + I_x u + I_y v - I(x,y)]^2. \tag{2.21}$$

Subtracting the twice occurring term $I(x,y)$, and writing the result in vector notation gives

$$E(u,v) = \sum_{x,y} w(x,y) \left[(u \quad v) \begin{pmatrix} I_x \\ I_y \end{pmatrix} \right]^2. \tag{2.22}$$

This equation can be reformulated further to

$$E(u,v) = (u \quad v)\boldsymbol{M}(u \quad v)^T, \tag{2.23}$$

where the matrix $\boldsymbol{M}$ is the *autocorrelation matrix* as

$$\boldsymbol{M} = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}. \tag{2.24}$$

The summation and weight was substituted by individual convolutions within the weighting kernel w [37]. Matrix $\boldsymbol{M}$, being build upon, describes a local estimate of shape of autocorrelation function (Eq. (2.18)). Moreover, the eigenvalues α and β of $\boldsymbol{M}$ provide the rotational invariant description of the feature. Eigenvalue analysis of the autocorrelation matrix $\boldsymbol{M}$ shows a strong relation between the character of the detected feature and the eigenvalues of the matrix at that image location. This relation is best summarized by three cases:

1. **Both eigenvalues are small**. Arbitrary shifts by $w(u,v)$ results in only small change to the autocorrelation function.

2. **One eigenvalue is high and other low**. Indicates an edge in the image. Shift in direction of the edge results in little change to the autocorrelation function, while a perpendicular shift in a high one.

3. **Both eigenvalues are high.** In this situation the autocorrelation function has sharp distinctive peak. It signalizes high change in case of any $\boldsymbol{u} = (u,v)$ and thus, a corner.

Fig. 2.7 illustrates this classification, where respective cases are clearly visible. The authors further propose a measure (response) of the quality of corners and edges as

$$R = det(M) - k\ trace(M)^2 = \alpha\beta - k(\alpha + \beta)^2. \tag{2.25}$$

The right side of the equation shows the analytical option how to calculate corner response R. If one considers 2×2 as a known matrix, R can be calculated without explicit eigenvalue decomposition of $\boldsymbol{M}$, while keeping the rotational invariance property. The corner response R has negative values for edges, positive for corners and is small in flat areas. The responses isocontours are also visible in Fig. 2.7.

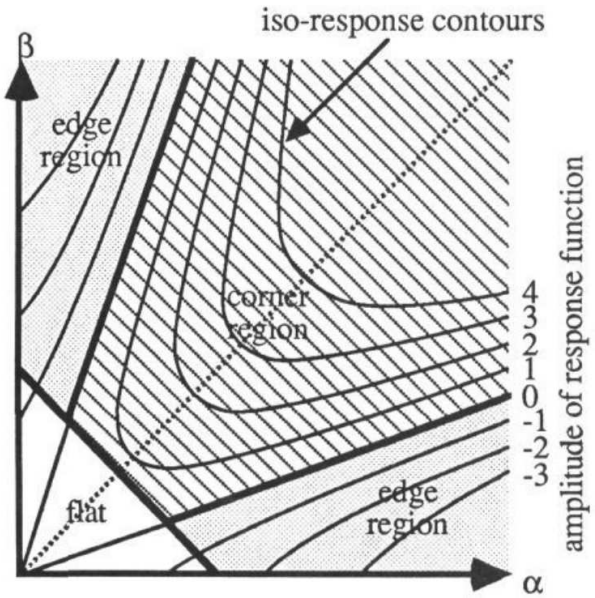

Figure 2.7: The eigenvalues space of autocorrelation matrix. The curves represent the corner response isocontours [17].

The question of corner selection based on autocorrelation matrix $\boldsymbol{M}$ is where Shi&Tomasi [34] perform differently. They consider a feature to be a good corner, if the smaller eigenvalue is higher than threshold

$$min(\alpha, \beta) > threshold. \tag{2.26}$$

Libraries used for the purposes of this Thesis offer both detection methods. The experiment will be performed to assess their difference and feasibility for VBAC (chapter 5).

One can see from the described method, that no effort to spatially distribute detection of features is used. This inevitably leads to their uneven distribution. For example Brown, Szeliski, and Winder [4] offers a method to mitigate this problem, by only detecting features within radius that are local maxima and whose response value is by a margin higher than neighbors. Implementation of similar threshold is available in the ViSP libraries (Sec. 2.6.1) used in this Thesis for VS.

2.4 Feature Tracking

After having acquired a list of features in one image, it is interesting to observe their apparent motion in the image stream. This process is called "*Feature tracking*". As with most techniques in computer vision, tracking was thoroughly investigated by various teams. Basics are image correlation [12], SSD methods [5] or optimization of matching criteria [24]. This last method was first introduced by Lucas & Kanade in 1981 [24]. It was further used and improved by numerous researchers. Namely by Jean-Yves Bouguet [3], whose pyramidal implementation of the classical Lucas-Kanade algorithm is used in this Thesis and described in detail.

2.4.1 Basic Tracking Algorithm

Let $I(x, y)$ and $J(x, y)$ be two successive images from the image stream. If there is a feature detected in image I at point $\boldsymbol{p} = [p_x, p_y]^T$ (Sec. 2.3), then $A(x, y) = I(x, y)$ for $\forall x, y \in [\boldsymbol{p} \pm (\boldsymbol{\omega} + 1)]$ is a window of I, where $\boldsymbol{\omega}$ is its size in pixels around point $\boldsymbol{p}$. Similarly $B(x, y) = J(x + g_x, y + g_y)$ for $\forall x, y \in [\boldsymbol{p} \pm \boldsymbol{\omega}]$ is a window, moved by a guessed shift $\boldsymbol{g} = [g_x, g_y]^T$ from the point $\boldsymbol{p}$, in which the feature will be searched for. The search is performed by minimizing a basic matching equation, similar to the Eq. (2.18), as

$$\epsilon(\boldsymbol{\nu}) = \epsilon(\nu_x, \nu_y) = \sum_{\boldsymbol{\omega}} [A(x, y) - B(x + \nu_x, y + \nu_y)]^2. \tag{2.27}$$

The minimization happens as partial derivation with respect to the displacement vector $\boldsymbol{\nu} = [\nu_x, \nu_y]^T$

$$\frac{\partial \epsilon(\boldsymbol{\nu})}{\partial \boldsymbol{\nu}} = -2 \sum_{\boldsymbol{\omega}} (A(x, y) - B(x + \nu_x, y + \nu_y)) \cdot \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix}. \tag{2.28}$$

Yet again, as in the Sec. 2.3, small displacements between the frames are considered. This allows:
First to a substitution of $B(x + \nu_x, y + \nu_y)$ by its Taylor expansion, resulting in

$$\frac{\partial \epsilon(\boldsymbol{\nu})}{\partial \boldsymbol{\nu}} \approx -2 \sum_{\boldsymbol{\omega}} (A(x, y) - B(x, y) \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix} \boldsymbol{\nu}) \cdot \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix}. \tag{2.29}$$

Second to approximate the gradient operator $\begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix}$ by its almost identical counterpart expressed from A and inherently I, giving a different notation

$$\nabla I = \begin{bmatrix} I_x \\ I_y \end{bmatrix} \doteq \begin{bmatrix} \frac{\partial B}{\partial x} \frac{\partial B}{\partial y} \end{bmatrix}^T. \tag{2.30}$$

The subtraction $A(x, y) - B(x, y)$ express the difference in a frame around the same spot, it can be understood as a temporal image derivative, allowing for another notation simplification such as

$$\delta I(x, y) \doteq A(x, y) - B(x, y). \tag{2.31}$$

After applying both notation changes (Eq. (2.30), Eq. (2.31)) and dropping the -2 multiplier, the Eq. (2.29) becomes

$$\frac{\partial \epsilon(\boldsymbol{\nu})}{\partial \boldsymbol{\nu}} \approx \sum_{\boldsymbol{\omega}} (\nabla I \boldsymbol{\nu} - \delta I) \nabla I^T \tag{2.32}$$

$$\left[\frac{\partial \epsilon(\boldsymbol{\nu})}{\partial \boldsymbol{\nu}} \right]^T \approx \sum_{\boldsymbol{\omega}} \left(\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \boldsymbol{\nu} - \begin{bmatrix} \delta I I_x \\ \delta I I y \end{bmatrix} \right). \tag{2.33}$$

Defining

$$G = \sum_{\boldsymbol{\omega}} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}, \text{and } \boldsymbol{b} = \sum_{\boldsymbol{\omega}} \begin{bmatrix} \delta I I_x \\ \delta I I y \end{bmatrix}, \tag{2.34}$$

the displacement vector, that minimize Eq. (2.27), Eq. (2.33) becomes

$$\boldsymbol{\nu} = G^{-1} \boldsymbol{b}. \tag{2.35}$$

2.4.2 Iterative Extension of Basic Tracking Algorithm

The previously described basic tracking algorithm is valid for small displacements between the two consecutive images. For a practical and accurate solution, that is smaller than some limit of error function, one iteration is not sufficient. For example the shift between the images may be too high for a reasonable Taylor expansion. Taking into account previous lines, recursive description of the algorithm leads to several changes.

Assuming that the iteration number is expressed by an index k, and final number of all iterations K. When some iteration $k-1$ is finished, B_k gets translated within guessed displacement $\boldsymbol{\nu}^{k-1}$ yielding

$$B_k(x, y) = B(x + \nu_x^{k-1}, y + \nu_y^{k-1}). \tag{2.36}$$

Eq. (2.27) becomes

$$\epsilon^k(\boldsymbol{\eta}^k) = \epsilon^k(\eta_x, \eta_y) = \sum_{\boldsymbol{\omega}} [A(x, y) - B_k(x + \eta_x^k, y + \eta_y^k)]^2, \tag{2.37}$$

where the vector $\boldsymbol{\eta}^k$ is the contribution to the final displacement vector $\boldsymbol{\nu}$ by the k-th iteration. Next the guessed displacement $\boldsymbol{\nu}^k$ for the $k+1$ iteration that shifted $\boldsymbol{B}_k$ prior to calculation, becomes

$$\boldsymbol{\nu}^k = \boldsymbol{\nu}^{k-1} + \boldsymbol{\eta}^k. \tag{2.38}$$

Throughout the iterative process $A(x, y)$ and so I_x,I_y remains constant and are computed only for the first iteration, but $B_k(x, y)$ changes. So all equations (2.27 to 2.35) needs to be recomputed. It is still a more efficient calculation though, since the original method in [24] required also the derivatives I_x,I_y to be recomputed at each iteration.

2.4.3 Pyramidal Extension of the Iterative Feature Tracking Algorithm

Complex scene offers usually a lot of options for feature detection and tracking. However it is not possible to decide from one image, whether some features are false positives. Features might be recognized at depth discontinuities or reflections. This problem can naturally be avoided by decreasing the window size ω. A smaller window would also increase the accuracy, but would come with a not so obvious problem. There is a requirement of small inter frame window change. It originates directly from Eq. (2.27). It is preferable to have a displacement vector $\boldsymbol{\nu} < \boldsymbol{\omega}$, the window size. Otherwise, the probability to successfully track the feature decrease rapidly. Another option, to increase the window size, is also not good. As mentioned earlier, a big window comes with the problem of smoothed details or false feature sliding at the edge of occlusion. To enable the tracking of fast moving features, pyramidal implementation of the feature tracking algorithm [3] was proposed.

The method starts by creating multiple image versions of I and J with the resolution decreased by 4. Let L_m define the number of layers and $L = 0, ..., L_m$ each individual layer. Any feature at coordinates in the original image $\boldsymbol{p} = \begin{bmatrix} p_x & p_y \end{bmatrix}$ will have coordinates in other layers defined as

$$\boldsymbol{p^L} = \frac{\boldsymbol{p}}{2^L}. \tag{2.39}$$

The size of the window ω remain constant however. If for example $\omega = 15$, the tracking window is 30×30 and the result is no more than 5 reasonable layers: $1280 \times 960, 640 \times 480, 320 \times 240, 160 \times 120, 80 \times 60, 40 \times 30$. Tracking starts at the deepest layer L_m, with an initial guessed displacement vector $\boldsymbol{\nu}_{\boldsymbol{L_m}} = [0\ 0]^T$. The result of iterating through layer L_m is passed to layer L_{m-1} as $\boldsymbol{\nu}^{[\mathbf{0}]}_{\boldsymbol{L_{m-1}}}$ the initial guess in the first iteration of Eq. (2.36). This process then runs again in the next layer, where the tracking process iterates for required number iterations as explained in Sec. (2.4.2), updating $\boldsymbol{\nu}^{\boldsymbol{k}}_{\boldsymbol{L}}$ (Eq. (2.38)) and so on, until layer 0.

2.4.4 Notes on Feature Detection and Tracking

- Detection and tracking as described is an intertwined problem as is visible from the similarity of respective Eq. (2.18) and Eq. (2.27). This relation is further explained in [34]. A loose explanation is: Good features are those that are easy to track, and vice-versa.
- The basic translation model as shown in this chapter is sufficient for tracking features, that change only a little. For more complicated cases (clear rotation / affine transformation...) extended model can be used [2].
- Features are detected only in the first frame, later on, the difference between two neighboring images is analyzed. This can lead to a significant change in the features character, practically turning them into outliers. [34] propose a measure of dissimilarity, that takes into account changes with respect to the first frame.

2.5 Geometric Jacobian

For any robotic manipulator, there exists a base, to which the manipulator is mounted and referred, and an end effector that represents the manipulator's other end. The task space is linked to the base. In this task space, a position $\boldsymbol{t}^0_n = (t_x, t_y, t_z)$ and an orientation (α, β, γ) of the end effector is represented by a vector $\boldsymbol{p}^0_n = [t_x \quad t_y \quad t_z \quad \alpha \quad \beta \quad \gamma]^T$. (Note that within this section, any further superscripts ^{0} represents mapping to the base frame, and subscripts $_n$ related to the last n-th link, or end effector frame).
Rigid manipulator elements are connected and articulated by joints. The joints configurations are represented in joint space by the vector $\boldsymbol{q} = (q_1 \quad q_2 \quad ... \quad q_n)^T$. The vectors $\boldsymbol{p}^0_n$ and $\boldsymbol{q}$ are related with the direct kinematics equation

$$\boldsymbol{p}^0_n = f_t(\boldsymbol{q}), \tag{2.40}$$

that prescribes the pose of the end effector in the base frame based on the joint variables.
A differentiation of Eq. (2.40) with respect to time results in

$$\dot{\boldsymbol{p}}^0_n = \boldsymbol{J}_a(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{2.41}$$

where $\boldsymbol{J}_a$ is the analytical Jacobian. Perhaps against intuition, the components of $\dot{\boldsymbol{p}}^0_n$ relative to the end effector orientation express the rate of change of the parameters characterizing the adopted minimal representation [35], not the angular velocity of the end effector. In order to

find a relation between joint speeds and linear and angular velocities a GJ is needed.

The geometric Jacobian is constructed as

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J_v} \\ \boldsymbol{J_\omega} \end{bmatrix}. \tag{2.42}$$

It is a $6 \times n$ matrix, where n is the number of joint variables. It relates the linear and angular velocities in a following manner

$$\boldsymbol{v}_n^0 = \boldsymbol{J_v}\dot{\boldsymbol{q}} \tag{2.43}$$

$$\boldsymbol{\omega}_n^0 = \boldsymbol{J_\omega}\dot{\boldsymbol{q}}. \tag{2.44}$$

Consider the transformation matrix

$$T(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{R}(\boldsymbol{q}) & \boldsymbol{t}(\boldsymbol{q}) \\ \boldsymbol{0}_3 & 1 \end{bmatrix} \tag{2.45}$$

that transforms from end effector frame to base frame, identically as Eq. (2.40) does. The matrix $\boldsymbol{R}(\boldsymbol{q})$ is the rotation matrix between the related frames.

The linear velocity Jacobian $\boldsymbol{J_v}$ is derived from the linear velocity of the end effector. The linear velocity based on the chain rule for differentiation, is

$$\boldsymbol{v}_n^0 = \dot{\boldsymbol{t}}_n^0 = \sum_{i=1}^{n} \frac{\partial t_n^0}{\partial q_i}\dot{q}_i. \tag{2.46}$$

Thus, the $i-$th column of the linear velocity GJ is

$$J_{v_i} = \frac{\partial \boldsymbol{t}_n^0}{\partial q_i}. \tag{2.47}$$

Further it can be shown [36], that

$$J_{v_i} = \begin{cases} \boldsymbol{z}_{i-1}^0 & \text{for prismatic joint} \\ \boldsymbol{z}_{i-1}^0 \times (\boldsymbol{t}_n^0 - \boldsymbol{t}_{i-1}^0) & \text{for revolute joint,} \end{cases} \tag{2.48}$$

where $\boldsymbol{z}_{i-1}^0$ is the representation of the axis of $(i-1)$-th joint in the base frame - the axis of rotation or direction of extension for a prismatic joint.

Angular velocity Jacobian. GJ related to angular velocity is obtained in a different way. Eq. (2.49) [36]

$$\omega_n^0 = \omega_1^0 + \boldsymbol{R}_1^0\omega_2^1 + \boldsymbol{R}_2^0\omega_3^2 + \boldsymbol{R}_3^0\omega_4^3 + \ \ldots \ + \boldsymbol{R}_{n-1}^0\omega_n^{n-1} \tag{2.49}$$

shows, that angular velocities can be added vectorially, if they are expressed with respect to the same frame. Therefore, angular velocity of the end effector frame can be written as a sum of angular velocities, expressed in the base frame, contributed by individual joints. Let ω_i^{i-1} be

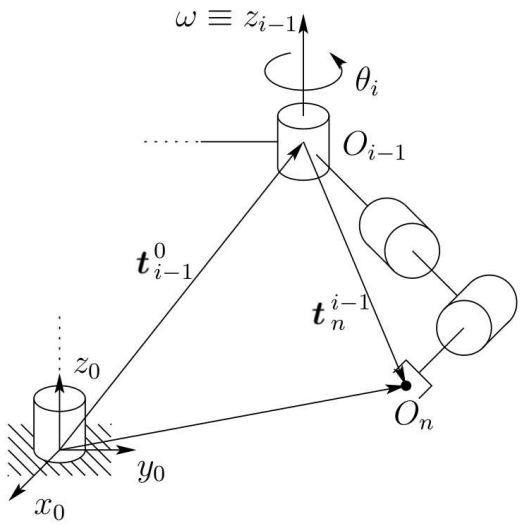

Figure 2.8: Manipulator end effector motion due to revolute joint i. Modified from [36]

the angular velocity with respect to the frame $i-1$, caused by rotation of joint i. Then ω_i^{i-1} is expressed in the $(i-1)$-th frame by

$$\omega_i^{i-1} = \dot{\boldsymbol{q}}_i \boldsymbol{z}_{i-1}^{i-1}. \tag{2.50}$$

A sketch of the motion of the end effector due to revolute joint i is shown at Fig. 2.8. The axis of rotation $\boldsymbol{z}_{i-1}^{i-1}$ of joint i can be transformed to the base frame using

$$\boldsymbol{z}_{i-1}^{0} = \boldsymbol{R}_{i-1}^{0} \boldsymbol{z}_{i-1}^{i-1}. \tag{2.51}$$

Finally, Eq. (2.49) is expressed by joint rates as

$$\boldsymbol{\omega}_n^0 = \rho_1 \dot{\boldsymbol{q}}_1 \boldsymbol{z}_0^0 + \rho_2 \dot{\boldsymbol{q}}_2 \boldsymbol{z}_1^0 + \rho_3 \dot{\boldsymbol{q}}_3 \boldsymbol{z}_2^0 + \ \ldots \ + \rho_n \dot{\boldsymbol{q}}_n \boldsymbol{z}_{n-1}^0 = \sum_{i-1}^{n} \rho_i \dot{\boldsymbol{q}}_i \boldsymbol{z}_{i-1}^0, \tag{2.52}$$

where ρ is 1 or 0 for revolute or prismatic joints respectively. Then, taking into account Eq. (2.44), the GJ for angular velocity is written as

$$\boldsymbol{J}_{\boldsymbol{\omega}} = \begin{bmatrix} \rho_1 \boldsymbol{z}_0^0 & \rho_2 \boldsymbol{z}_1^0 & \rho_3 \boldsymbol{z}_2^0 & \ldots & \rho_n \boldsymbol{z}_{n-1}^0 \end{bmatrix} \tag{2.53}$$

Finally the **GJ** is a combination of $\boldsymbol{J_v}$ and $\boldsymbol{J_\omega}$. For joint i it is defined as

$$\boldsymbol{J}_i = \begin{bmatrix} \boldsymbol{J_{v_i}} \\ \hline \boldsymbol{J_{\omega_i}} \end{bmatrix} = \begin{cases} \begin{bmatrix} \boldsymbol{z}_{i-1}^0 \times (\boldsymbol{t}_n^0 - \boldsymbol{t}_{i-1}^0) \\ \boldsymbol{z}_{i-1}^0 \end{bmatrix} & \text{for revolute joint} \\ \begin{bmatrix} \boldsymbol{z}_{i-1}^0 \\ \boldsymbol{0} \end{bmatrix} & \text{for prismatic joint} \end{cases} \tag{2.54}$$

2.6 Software, External Libraries and Hardware

The objective of this Thesis is to test the applicability of VS for the purpose of attitude control of a robot. The mathematical principles described in the previous chapters are implemented in a software bundle, that is based on an already available external libraries and software packages. The hardware demonstrator, on which the software runs, is specifically designed and build for this Thesis and constructed from commercially available hardware and a 3D printed frame. External libraries, software and hardware parts used are shortly described in this chapter.

2.6.1 Software and Libraries

The software runs on a *Linux* operating system, Ubuntu 16.04 and is written in *C++*. Therefore the software should run under most modern Linux systems. *ROS*, *OpenCV* and *ViSP* library are used in the Thesis extensively.

ROS. Robot Operating System is a "*flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms.*"[31]
ROS's middleware offers a message passing interface for communication based on a publish/subscribe message system and services. This interface enables communication between robot specific features and ROS tools. The communication interface, robot specific features and tools together form the core components of ROS. Some specific capabilities of ROS are mapping, navigation, pose estimation etc. Tools like *rviz* or *rqt* allows a 3D visualization of data or framework for graphical user interface respectively.
ROS was developed with collaboration in mind. It allows various individuals and teams to collaborate and share without the need to reinvent the tools by each team individually.
Thesis utilize the ROS distributed system in a manner that each function is represented by one ROS node, namely the image acquisition and processing, feature detection, feature tracking and visual servoing.

OpenCV. OpenCV stands for Open Source Computer Vision Library [16]. It is a software library aimed at computer vision and machine learning, dedicated to accelerated utilization of machine perception. It offers a wide range of algorithms capable of object identification, tracking, 3D model extraction, image stitching, scenery recognition etc. Thanks to the cross platform compatibility and programming language it supports, OpenCV is one of the most used libraries. Within the scope of this Thesis, it is used among other things for camera calibration and as a 3rd party library for the feature tracking module within ViSP.

ViSP. *"ViSP standing for Visual Servoing Platform is a modular cross platform library that allows prototyping and developing applications using visual tracking and visual servoing techniques at the heart of the researches done by Inria Lagadic team. ViSP is able to compute control laws that can be applied to robotic systems. It provides a set of visual features that can be tracked using real time image processing or computer vision algorithms. ViSP provides also simulation capabilities. ViSP can be useful in robotics, computer vision, augmented reality and*

computer animation."[38]
ViSP platform offers broad list of modules that are independent of hardware, are portable and possible to mutually combine. For the purpose of this Thesis modules dedicated to keypoint detection, feature tracking and visual servoing are used.

2.6.2 Hardware

The hardware demonstrator is built as a hand-held device, able to operate entirely disconnected from power or network. Image acquisition by a camera, camera pose control, data processing and power will be build into the demonstrator. These parts are shown on Fig. 2.9 and briefly described in following paragraphs.

ODROID C2 is a single board computer running on a 64bit architecture. The computational power is handled by an ARM Cortex A53 1.5Ghz processor. Other parameters are 2Gb DDR3 SDRAM, 4x 2.0 USB, eMMC flash storage slot and 40 pin GPIO. It is compatible with various Unix systems.

oCam-1MGN-U is a global shutter monochrome camera. It provides 1280×960 pixel resolution at 45 fps from the OnSemi MT9M031 CMOS image sensor. Sensor and pixel size is 1/3 inch (8.46 mm) $1.4\mu m \times 1.4\mu m$, respectively. Lens has 3.6 mm focal length and 65° FoW. The camera supports the USB 3.0 interface.

3.5inch Touchscreen Shield. The screen used is a 3.5" (88.9 mm) touch resistive TFT LCD with 480×320 pixels resolution. It is connected to the Odroid C2 via 40 GPIO connector and includes 4 configurable buttons.

Arduino & Servo. The camera is mounted in Roll-Pitch-Yaw gimbal. Each axis is moved by a basic servo motor. Servo motors are controlled by Arduino micro. Although the Odroid C2 features 40 GPIO pins, an additional Arduino is necessary since those pins are used by the LCD.

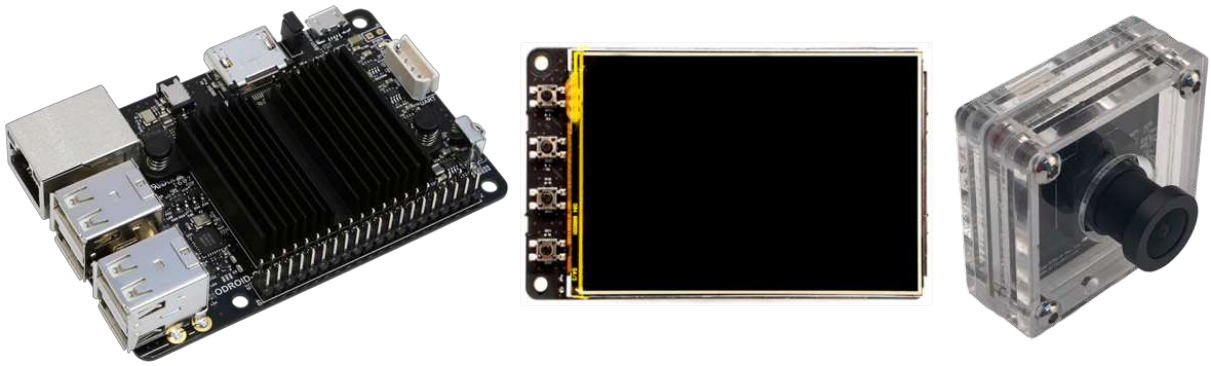

Figure 2.9: Hardware components. From left: Odroid C2, 3.5inch Touchscreen Shield, Camera. [30]

Chapter 3

Demonstrator design

The Principles behind VS allows to control practically any arbitrary amount of joints, provided the apparatus has detected a sufficient number of available features. For practical reasons this controlled space is represented by 6 DoF.

The attitude control in a form of VS was investigated with application to a Spacecraft in mind. The orbit of most Spacecrafts is practically stable and predetermined by the designed parameters. On the other hand, many Spacecrafts need a precise control of the attitude. Applying this scenario to robotics, mean a control of only 3 DoF that are related to the orientation, are demanded from the manipulator. Manipulators build for this purpose come basically in two configurations, serial or parallel. Within some extent commercially available products representing these two configurations would generally suffice for the purpose. Examples of both are to be seen on Fig. 3.1.

Parallel manipulators like the *Gough - Stewart platform* offer certain advantages compared to the serial counterparts. Primarily it is the higher rigidity and precision. This is mainly because the controlled links in parallel manipulator share the dynamic and static load. As a comparison, the serial manipulators have with first link that have to carry all the following ones. Serial manipulators on the other hand offer bigger operable space and are simpler to control and build. For this reason a variant of a serial manipulator with three revolute joints, with perpendicular axis of rotation, was chosen for the Thesis. Commercially available products, where the attitude control is based on a gyroscope signal offer sufficient functionality in this manner. To use one, would mean among other things to re-engineer the motor controller. This fact together with budget concerns and uncertain rigidity resulted in the demonstrator being designed in-house. Next Sec. 3.1 explains the design limits and decisions that led to the resulting product.

Figure 3.1: Example of serial (left) and parallel (right) manipulators [21],[28].

3.1 Design process

The design process was limited by multiple constrains. The demonstrator that is an outcome of the process is shown on Fig. 3.2, where individual parts are unrealistically colored for a better recognition. Most of the constrains that led to the design together with the solutions are presented in the next part of this section.

Hemisphere coverage. Used camera offers 65° FoW. Simple servomotors were picked since they offer $\approx 180°$ range. Stepper or DC motors were considered and refused, due to the total simplicity of control of servo motors. Servo motors additionally offers a high static torque thanks to the gearbox and included position controller.

Sufficient rigidity. It is an unwritten rule to do mechanical prototyping with the help of additive manufacturing - 3D print. The commonly available ABS or PLA materials are more than adequate for the purpose. If the product is designed properly, risk of structural failure is minimal.

Compact dimensions. The consumer products as the camera holder on Fig. 3.1 are minimalistic, lightweight and nice to look at. It is unfeasible to attain similar characteristics in a project of this scale. Nonetheless the demonstrator was designed with at least some level of the aesthetics in mind and with as little internal volume as was possible.

User friendliness. Another desired similarity with consumer products was the intuitive usability. The demonstrator is designed with no hatches covers etc, as solid as possible with only two power switches for the `Odroid C2` and the `arduino` with servomotors.

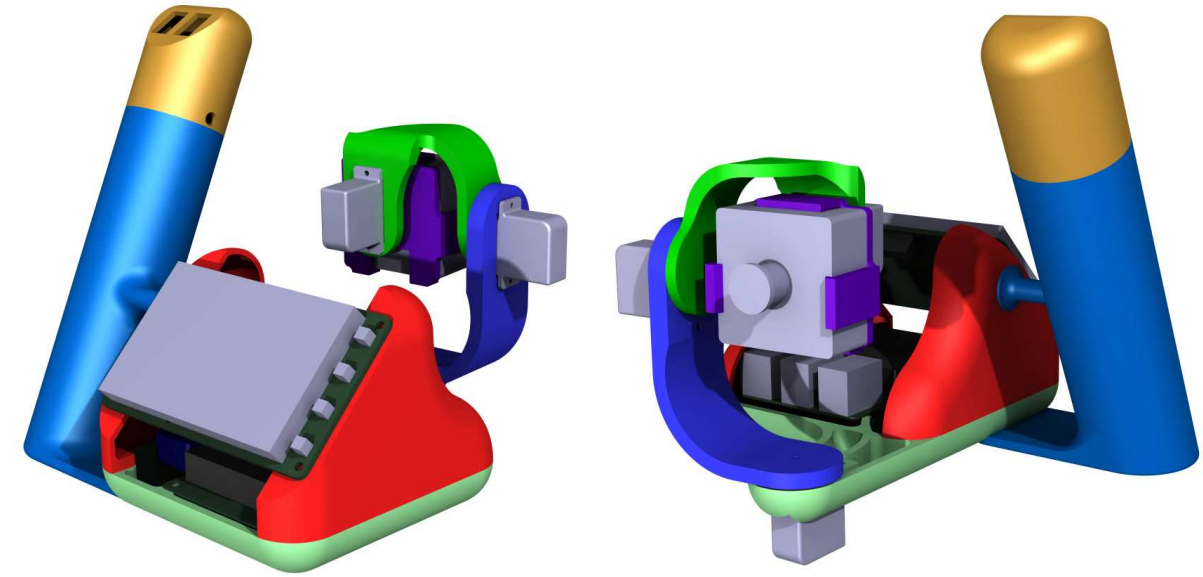

Figure 3.2: 3D model of the demonstrator.

All-in-one design without the need of external hardware. VS is relatively computationally demanding process. Also any manipulator with the camera mounted to spherical wrist would suffice as a proof of concept. Both these facts aims to use of external computer connected to the manipulator. This option was undesired and so a small computing board - `Odroid C2` with display was build into the demonstrator.

Free movement of camera cable. One considered option was to combine the camera and the computing power into one package that is mounted to the end effector. This option was discarded for the dimensions constrains and unnecessary load on servomotors. The camera is the only electronics mounted to the end effector. Therefore movement of the Camera $\leftrightarrow$ `Odroid C2` cable cannot be limited by any obstruction.

The manipulator mounted to the structure is designed for a Yaw-Pitch-Roll configuration. Target orientation is also defined by intrinsic rotations as $(z-x'-z'')$ sequence. In this sequence the coordinate frame is rotated first around the z axis (Yaw), second the new x' axis (Pitch) and finally around the again new z'' axis (Pitch). There is another 11 variants how the camera might be pointed. In the case of $(z - x' - z'')$, the manipulator structure does not block the camera cable pointing backwards from it. Exploded view and real photos of the demonstrator are shown on Figures (3.3 3.4).

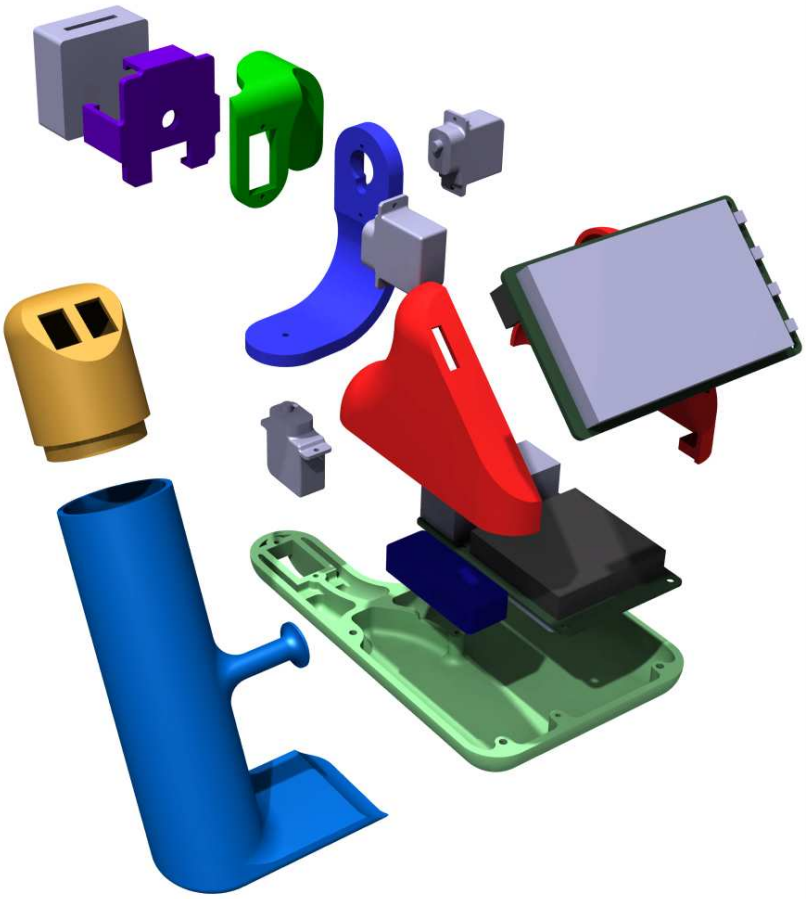

Figure 3.3: Exploded view of the 3D model of the demonstrator.

Figure 3.4: The demonstrator.

Chapter 4

Visual Servoing Implementation

This chapter describes former analytical computation of geometrical Jacobian needed for successful program functionality and the implementation of Visual servoing into a functional program. The computation of Jacobian in Sec. 4.2 is based on the direct kinematics model shown in Sec. 4.1. Section 4.3 describes the structure of software written for this thesis and explains the functionality of individual nodes.

4.1 Direct Kinematic Model

For the purpose of robotic task within this thesis, establishing of a kinematic model based on the design described in chapter (3) was needed. The design of the manipulator is a common yaw - pitch - roll representation of a spherical wrist. Applying *Denavit - Hartenberg* (D-H) convention for selecting frames of reference, coordinate frames were assigned as shown on Fig. 4.1.

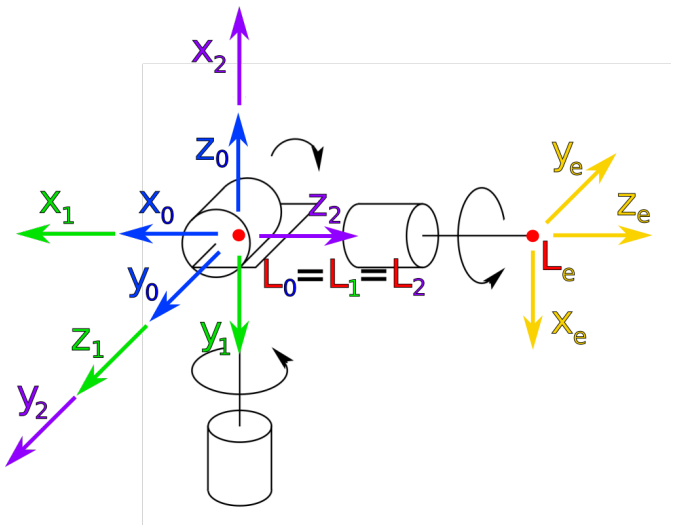

Figure 4.1: Manipulator coordinate frames. Background wrist image was modified from [36].

D-H convention was likewise applied to derive transformation matrix between base and end effector frame. Individual transformation matrices and final homogeneous transformation matrix from end effector to base frame are following

$$
\begin{aligned}
{}^{0}\boldsymbol{T}_{1} &= \begin{bmatrix} cos(q_1) & 0 & -sin(q_1) & 0 \\ sin(q_1) & 0 & cos(q_1) & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
{}^{1}\boldsymbol{T}_{2} &= \begin{bmatrix} cos(q_2) & 0 & sin(q_2) & 0 \\ sin(q_2) & 0 & -cos(q_2) & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
{}^{2}\boldsymbol{T}_{e} &= \begin{bmatrix} cos(q_3) & -sin(q_3) & 0 & 0 \\ sin(q_3) & cos(q_3) & 0 & 0 \\ 0 & 0 & 1 & L \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{4.1}
$$

$$
{}^{0}\boldsymbol{T}_{e} = {}^{0}\boldsymbol{T}_{1}\,{}^{1}\boldsymbol{T}_{2}\,{}^{2}\boldsymbol{T}_{e} = \begin{bmatrix} c1c2c3 - s1s3 & -c1c2s3 - s1c3 & c1s2 & c1s2L \\ s1c2c3 + c1s3 & -s1c2s3 + c1c3 & s1s2 & s1s2L \\ -s2c3 & s2s3 & c2 & c2L \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.2}
$$

where $s1$, $c1$, $s2$, $c2$, $s3$, $c3$ stands for $sin(q_1)$, $cos(q_1)$, $sin(q_2)$ etc. This mathematical construct was further used to determine the GJ, that is a part of VS.

4.2 Geometric Jacobian of Spherical Wrist w.r.t. End Effector Frame

The method described in Sec. 2.5 simplifies the determination of GJ for any manipulator. Joint axes $\boldsymbol{z}$ and coordinates of frame origins $\boldsymbol{t}$ are all the quantities needed. They are easily obtained from corresponding transformation matrices $\boldsymbol{T}$ of the forward kinematics model of the manipulator. Joint axes $\boldsymbol{z}$ are represented by first three elements in the third column, coordinates of frame origins $\boldsymbol{t}$ are the first three elements in the fourth column.
Spherical wrist consists of three revolute joints, selecting the correct row of the equation (2.54) leads to GJ of the form

$$
{}^{0}\boldsymbol{J}_{e} = \begin{bmatrix} \boldsymbol{z}_0^0 \times (\boldsymbol{t}_e^0 - \boldsymbol{t}_0^0) & \boldsymbol{z}_1^0 \times (\boldsymbol{t}_e^0 - \boldsymbol{t}_1^0) & \boldsymbol{z}_2^0 \times (\boldsymbol{t}_e^0 - \boldsymbol{t}_2^0) \\ \boldsymbol{z}_0^0 & \boldsymbol{z}_1^0 & \boldsymbol{z}_2^0 \end{bmatrix}. \tag{4.3}
$$

Note the subscript n representing the last frame was changed according this thesis specific task to e as end effector. This GJ form however relates the change in joint variables to the end

effector velocities expressed in the base frame. Instead, end effector velocities expressed in it's own frame are needed for functional VS system. Desired GJ has form

$$ {}^{e}\boldsymbol{J}_{e} = \begin{bmatrix} \boldsymbol{z}_0^e \times (\boldsymbol{t}_e^e - \boldsymbol{t}_0^e) & \boldsymbol{z}_1^e \times (\boldsymbol{t}_e^e - \boldsymbol{t}_1^e) & \boldsymbol{z}_2^e \times (\boldsymbol{t}_e^e - \boldsymbol{t}_2^e) \\ \boldsymbol{z}_0^e & \boldsymbol{z}_1^e & \boldsymbol{z}_2^e \end{bmatrix}. \tag{4.4} $$

4.2.1 Computation of Geometric Jacobian

Elements of the Jacobian matrix (Eq. (4.4)) was extracted from the direct kinematic model in Sec. 4.1, Those elements that were not directly available were calculated from reversed variants of transformation matrices. For homogeneous coordinate transformation matrix between two orthonormal frames, the reverse of transformation matrix is calculated as

$$ \boldsymbol{T}^{-1} = \begin{bmatrix} \boldsymbol{R}^T & -\boldsymbol{R}^T\boldsymbol{t} \\ \boldsymbol{0}_3 & 1 \end{bmatrix} \tag{4.5} $$

where $\boldsymbol{R}$ and $\boldsymbol{t}$ are respectively the rotation matrix and translation of frame origin of the matrix $\boldsymbol{T}$.
First derived matrix element is $\boldsymbol{t}_e^e$, it is a translation of origin of end effector frame to the end effector frame, and therefore $\boldsymbol{t}_e^e = \boldsymbol{0}$. Other Jacobian matrix elements were determined as follows

$$ \begin{aligned} {}^{0}\boldsymbol{T}_e &= {}^{0}\boldsymbol{T}_1 {}^{1}\boldsymbol{T}_2 {}^{2}\boldsymbol{T}_e \\ {}^{0}\boldsymbol{T}_e^{-1} &= {}^{e}\boldsymbol{T}_0 \qquad\qquad \longrightarrow \boldsymbol{z}_0^e,\ \boldsymbol{t}_0^e \end{aligned} $$

$$ \begin{aligned} {}^{1}\boldsymbol{T}_e &= {}^{1}\boldsymbol{T}_2 {}^{2}\boldsymbol{T}_e \\ {}^{1}\boldsymbol{T}_e^{-1} &= {}^{e}\boldsymbol{T}_1 \qquad\qquad \longrightarrow \boldsymbol{z}_1^e,\ \boldsymbol{t}_1^e \end{aligned} $$

$$ {}^{2}\boldsymbol{T}_e^{-1} = {}^{e}\boldsymbol{T}_2 \qquad\qquad \longrightarrow \boldsymbol{z}_2^e,\ \boldsymbol{t}_2^e $$

After filling elements of Jacobian matrix (Eq. (4.4)) and several algebraic simplifications (e.g. $sin(q_1)^2 + cos(q_1)^2 = 1$). The Jacobian matrix becomes

$$ {}^{e}\boldsymbol{J}_{e} = \begin{bmatrix} sin(q_2)sin(q_3)L & cos(q_3)L & 0 \\ cos(q_3)sin(q_2)L & -sin(q_3)L & 0 \\ 0 & 0 & 0 \\ -cos(q_3)sin(q_2) & sin(q_3) & 0 \\ sin(q_2)sin(q_3) & cos(q_3) & 0 \\ cos(q_2) & 0 & 1 \end{bmatrix}. \tag{4.6} $$

This symbolic representation of GJ is a function of joint variables $\boldsymbol{q}$. After implementation it serves as a framework for particular Jacobian matrix calculated each iteration from current joint variables.

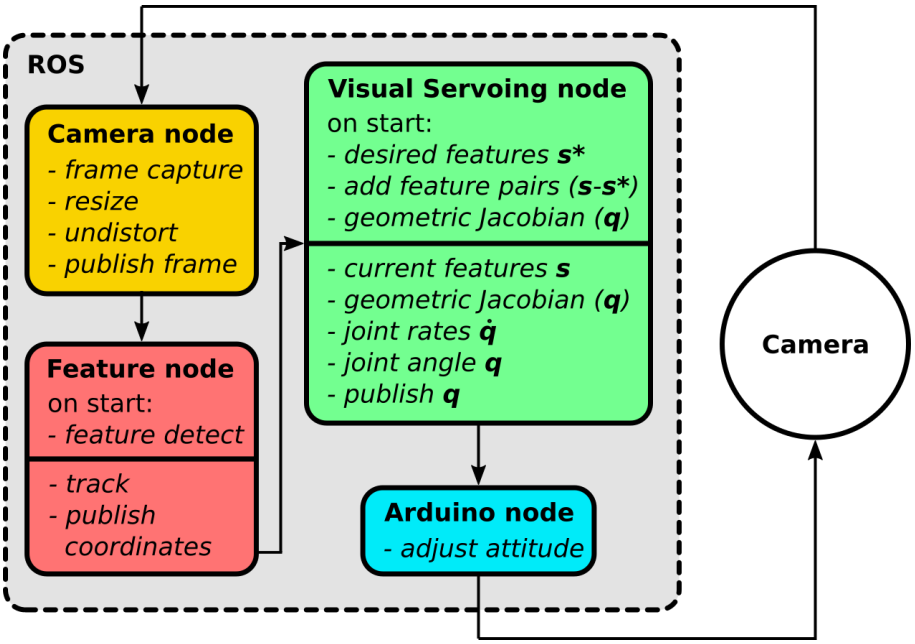

Figure 4.2: Project architecture.

4.3 Software Architecture

Big part of the work performed is realized in a form software components. Components have form of individual `ROS` nodes. Each node contain set of logically related functions. Program structure is serial, i.e. individual node's published message serves as an input to next node. Figure (4.2) shows high level overview of the project architecture with main functions listed. Complete project contain four nodes. Functionality description of each is following

`Camera_node` acquires frame in a `Y800` or `GREY` format. Frame is transported to `OpenCV` compatible format using camera manufacturer proprietary `Withrobot` libraries. Camera does not offer to resize image sufficiently, only to cut out section of it, practically decreasing the field of view. Therefore frames are captured in 640×480 pixels resolution and additionally resized to desired 480×320. Frame is undistorted based on the precomputed camera intrinsic parameters and published with message type `sensor_msgs/Image` on `camera/image` topic.

`Feature_node` subscribes to `camera/image`. Features are detected in the first frame on initialization based on the algorithm explained in Sec. 2.3. Detection happen in small rectangular area in the frame center. Tracking is performed on all consecutive $2, 3, 4$ ... n incoming frames identically to principle in Sec. 2.4. Pixel coordinates of tracked features in the image plane are published in an array on topic `Tracked_features_coordinates` in a message type `std_msgs::Float32MultiArray`.

`Visual_Servoing_node` is subscribed to `Tracked_features_coordinates` topic. Feature co-

ordinates are first recomputed from pixels in image frame to meters in camera frame based on the relative position of camera and end effector. In this project's specific case the camera is fixed to end effector and so the transformation matrix ${}^{c}\boldsymbol{V}_{e}$ is constant. On start of the node the feature coordinates contained in the first message are used to construct the set of desired features $\boldsymbol{s}^{*}$, and to add $(\boldsymbol{s}-\boldsymbol{s}^{*})$ feature pairs to VS task. Geometric Jacobian matrix is calculated based on the initial joint variables $\boldsymbol{q}$. Each following iteration the set of current features $\boldsymbol{s}$ is updated from incoming messages, Jacobian is calculated for it is needed to calculate the joint rates $\dot{\boldsymbol{q}}$ from the control equation (2.13). From joint rates the new joint variables are computed and published on a `joint_variables` topic with a `geometry_msgs::Vector3` message type.

`Arduino_node` is the last link in the cycle, it is subscribed to `joint_variables` topic and set the servomotors position according on received joint variable. Thus the attitude of the camera is modified and the cycle repeats.

Chapter 5

Experiments and Functionality Assessment

This chapter describes the series of experiments that were done to evaluate the VS for the purposes of VBAC. In the first Sec. 5.1 the two similar feature detectors are tested and compared. The following Sec. 5.2 consist of an experiment aimed at feature tracking stability and the last Sec. 5.3 describes the functionality testing of the completed demonstrator. It must be noted in advance, that the system performance is influenced by too many external factors (e.g. lightning conditions, reflectivity of the observed surface/map, focus...), rendering the repeatability of the achieved results practically impossible. Further experiments are performed to assess the functionality of the VS as a platform, not the strictly scientific value of the methods used. For this reason only variables relevant to the individual experiment are mentioned.

5.1 Feature Detector Evaluation

The feature detection implemented for the purposes of this thesis relies on an autocorrelation matrix (Eq. (2.24)) for detection of features that will be processed in the following modules. To have a first impression on the capability of the system, two mainstream approaches were compared - Harris and Shi & Tomasi. From the description in Sec. (2.3.1) can be seen, that the detectors does not give much space for the parameter tuning, as they differ only in the approach to the autocorrelation matrix. *ViSP* libraries however allow for an additional overlay over the basic detectors, most notably allowing to select the maximum number of features detected and a quality level. The maximum number of features is self explanatory. The Quality parameter is multiplied by the best corner measure, either the minimal eigenvalue for Shi& Tomasi or the quality response R for Harris. Any potential feature below the result is discarded. Using those parameters is highly subjective, and depends strictly on the application. In the following subsections these parameters are used variably. The performance of both detectors was investigated and is explained in the following subsection.

Figure 5.1: Feature Detection: Camera pose with respect to the image.

5.1.1 Repeatability

A first experiment consists of repeated detection of a big number of features. The goal is to see the coherence between individual detections. For the experiment the camera was fixed relative to the image as is shown in Fig. 5.1. The detection algorithm ran 1000 times for both Shi&Tomasi and Harris. Because most detection parameters are shared for Shi&Tomasi detector, Harris parameter k is the only differentiating option. Effect of its default value $k = 0.04$ and $k = 0.4$ are tested and compared to results obtained by the Shi&Tomasi algorithm. Fig. 5.2 shows the histogram of detected features distribution. It can be seen that Harris detector tend to produce a higher number of features. This tendency rises with the increase of the Harris parameter. The Shi&Tomasi detector is more consistent, as can be seen from the standard deviation in Table 5.1.

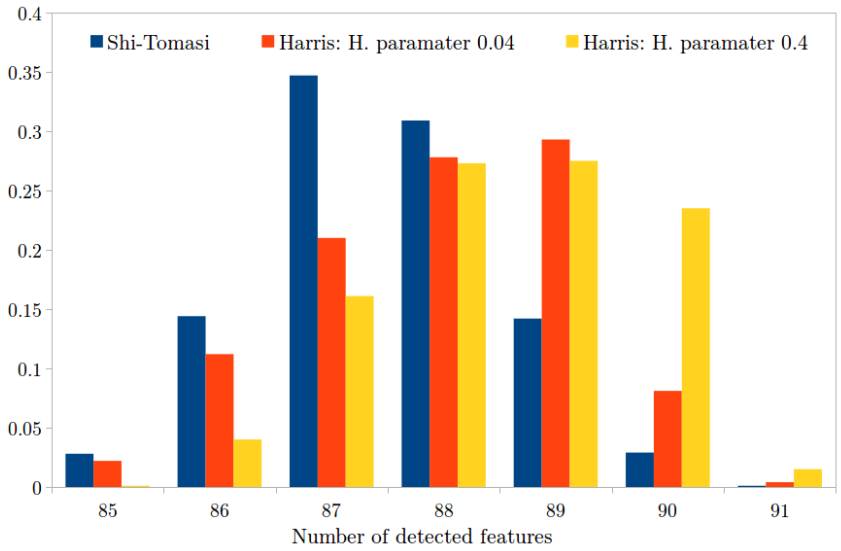

Figure 5.2: Feature Detection: Histogram of the number of detected features.

An examination of the images with features overlay, obtained by the camera, shows a high

	Shi & Tomasi	Harris , $k = 0.4$	Harris , $k = 0.04$
mean [/]	87.487	87.969	88.548
standard deviation [/]	1.0821284591	1.225373633	1.171780299

Table 5.1: The mean and standard deviation of number of features detected by Shi&Tomasi and Harris detector after 1000 runs.

reliability of the detection algorithms. Fig 5.3 and 5.4 show two extreme cases, the first figure shows the minimum 85 features detected by Shi&Tomasi detector, the second figure shows 91 features detected by the Harris detector. These figures exemplify the inner working of the algorithm, i.e. in localities with a high contrast (high image gradients) features are detected, contrary to the bleak or dark portions. Notice the majority of detected features is re-detected at the identical coordinates. This behavior is expected, because both algorithms operate with the same autocorrelation matrix. Several features that are missing in the first figure are missing in the dark locations. This difference in detection repeatability is assumed to result from small random variations in the lighting conditions that translated into a slightly inferior autocorrelation matrix at the location of missing feature and finally resulted in a quality below the threshold.

This experiment primarily shows the applicability of the detection algorithms and the richness of information contained in a small resolution, gray scale image. Further it confirms the expectation, that the detection algorithms are practically identical in performance. Based on this result, only Shi&Tomasi detector is used in the rest of the document.

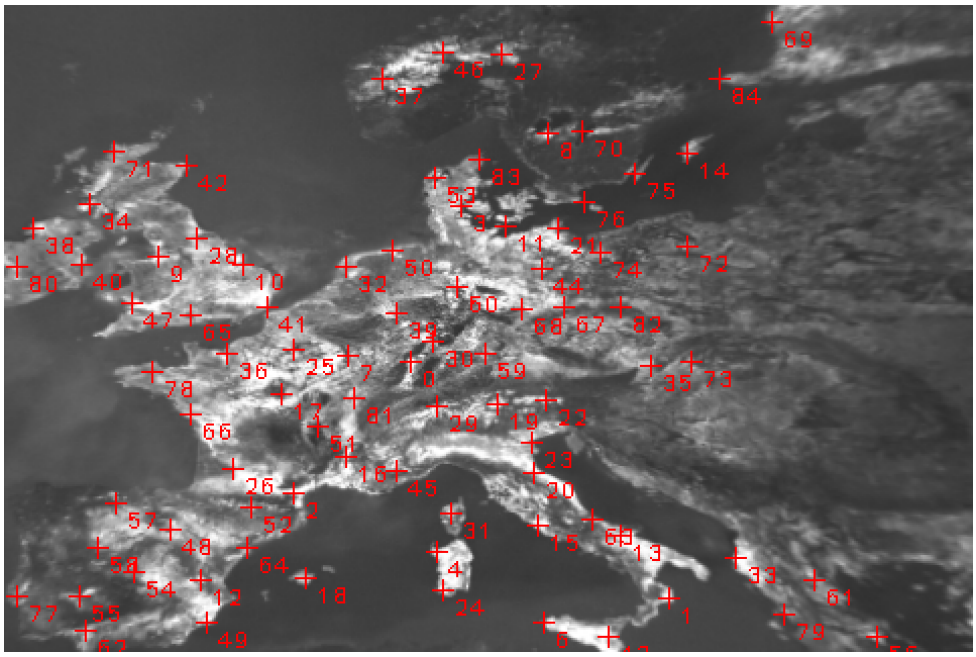

Figure 5.3: Feature Detection: The minimal amount of detected features, Shi&Tomasi, 85 features, frame 35.

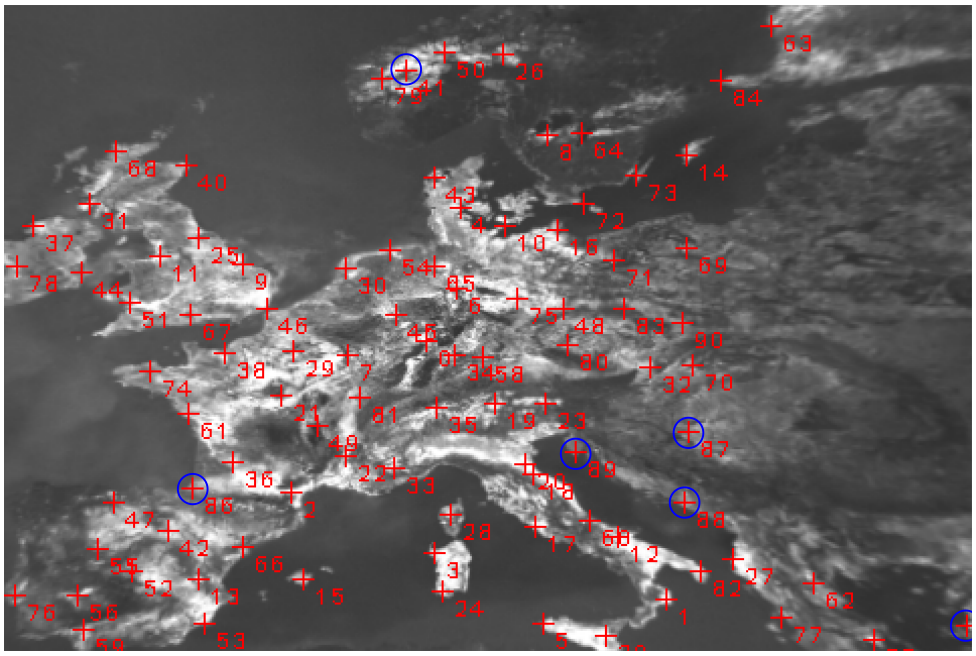

Figure 5.4: Feature Detection: The maximal amount of detected features, Harris $k = 0.4$, 91 features, frame 63. Several unstable features are highlighted.

5.2 Visual Servoing Performance

The next step in the experiment chain is the evaluation of the tracking and control law performance. This experiment aims to verify the robustness of the software architecture in a complicated scenario. To evaluate the control law (Eq. (2.13)) performance, joint velocities $\dot{\boldsymbol{q}}$ are updated throughout the test based on a constant initial GJ. This is done to test the stability of the tracked features $\boldsymbol{s}$ and the precision of the joint velocities $\dot{\boldsymbol{q}}$ without the noise and error that is inherently related to the jittery servo rotations. The full evaluation of the demonstrator with focus on the attitude control is described in another Sec. 5.3.

The experiment scenario consists of fluent arbitrary movements, orientations and distance adjustments of the camera with respect to the observed map. An arbitrary path of the camera is mixed with several returns to an initial fixed pose to evaluate the potential drift of the features and error in the computed joint velocities. The processing is done on a PC, not demonstrator itself, to remove the the performance bottleneck and allow the VS to perform unhindered. In ideal scenario, the error $\boldsymbol{e} = (\boldsymbol{s} - \boldsymbol{s}^*)$ between the current and the desired features, and therefore the resulting joint angular velocities, would return to zero when the camera returns to the initial pose. The features were detected only in the 50×50 pixel area in the middle of the image to expand the potential pose change as much as possible. Tab. 5.2 summarizes the parameters used for the experiment.

Feature detection and Tracking	
maximum features [/]	5
detection square area [pixel]	50
quality parameter [/]	0.09
detection window [pixel]	12
tracking window [pixel]	12
pyramid levels [/]	5
Visual Servoing	
constant gain [/]	1
control update frequency [hz]	22.5

Table 5.2: The VS experiment parameters.

The experiment lasted for 110 s. During the test, the initial pose was reached four times. Fig 5.5, visualize the current (red) and desired (green) feature sets overlaid over the camera view. The desired features are build from the first message that the `Visual_Servoing_node` receives and are displayed for comparison. Subfigures are linked to respective time points in Fig. 5.6, which shows the joint rates calculated by the control algorithm. Note their absolute magnitude is not crucial for precise work, because a specific gain λ might be applied to increase/decrease the velocities and the dynamics of the system.

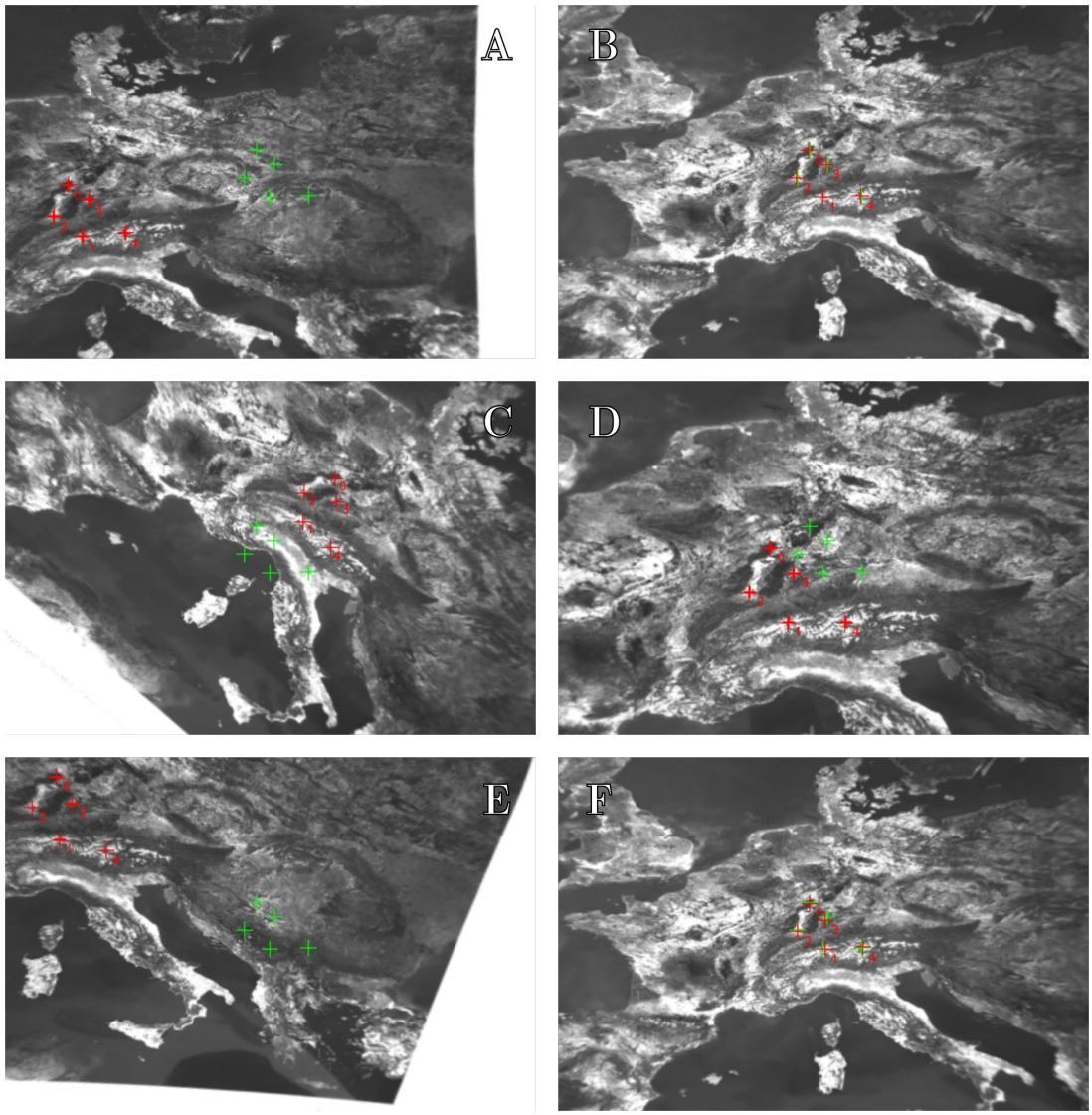

Figure 5.5: VS experiment: A composite image sequence of the camera view with current (red) and desired (green) features.

Fig. 5.6 shows the computed joint velocities $\dot{q}_1$, $\dot{q}_2$, $\dot{q}_3$ with respect to time through the test. Several remarkable points in time are marked and discussed. Fig. 5.7 shows the distance of individual features to the current averaged CoG. This figure shows features behavior, that might not be visible from the joint angular velocities, e.g. drift of one feature would be distinguishable if a particular line would change the value and slope independently of the others. Unless the image would rotate with all features being well tracked.

Point ***A*** is an arbitrary pose change, governed primarily by a change in the position instead

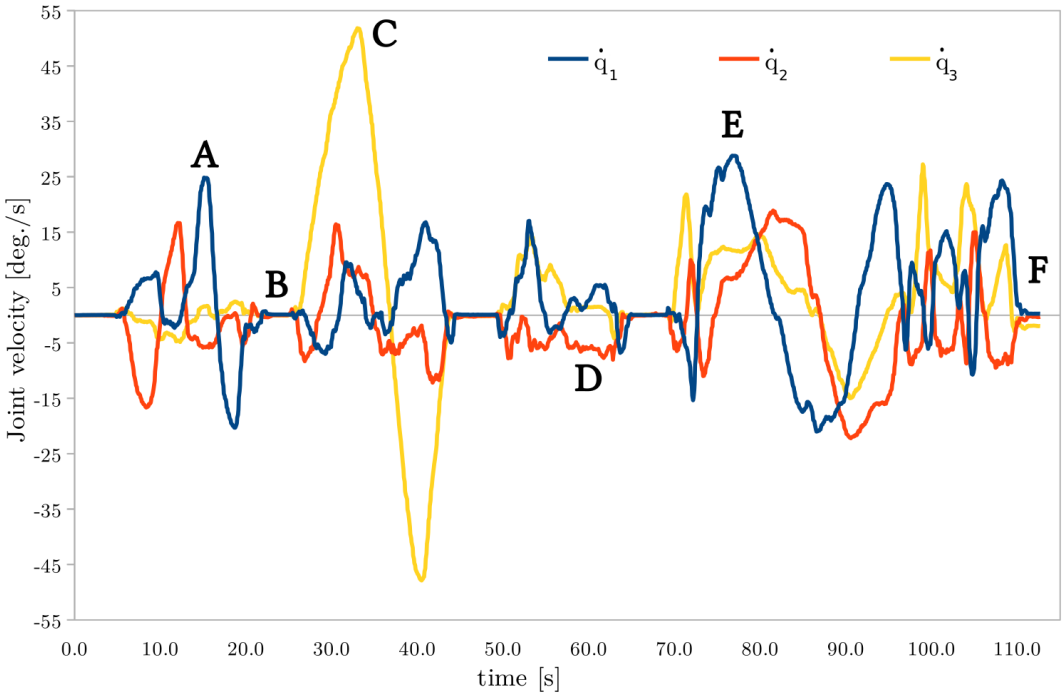

Figure 5.6: VS experiment: Joint angular velocities $\dot{q_1}$, $\dot{q_2}$, $\dot{q_3}$ computed by the control law with constant GJ.

of orientation. It can be seen from the image sequence in Fig. 5.5: ***A***, that the features are well tracked up to the edge of image. Due to the calibrated camera, the individual error caused only by the distortion at the image edges is minimized. Point ***B*** shows the first return to the initial pose, identically to another three returns in later time. The flat region in all joint angular velocities signalize a minimal feature drift, what is a very important and desired property in the VS. In the point ***C***, the camera is moved and mainly rotated by $\pm 50°$ counterclockwise and soon after clockwise, this change is visible as a nonuniform slope change in Fig. 5.7. Fig. 5.5: ***D*** shows major scale change, it can be seen from Fig. 5.7 that the camera was first brought closer, then further from the image. ***E*** depicts another limit pose change. The uneven distribution of curves in Fig. 5.7 at this time point shows a change in feature spatial distribution. This effect is a combination of a small distortion and considerable change in the view angle. ***F*** shows the final return to the initial position. Poses in subfigures ***C*** and ***D*** were particularly selected to test the rotation and scale invariant property of the tracking algorithm. The invariant property is confirmed by no observable drift of the features and the minimal final error in the point ***F***.

This complicated scenario show the suitability of the visual servoing algorithm for the purposes of VBAC. Suitability is further substantiated by the subpixel error values at the end of the test (point ***F***), resulting in a mean error $e = 0.17 \approx 0.022°$, considering 65° FoV.

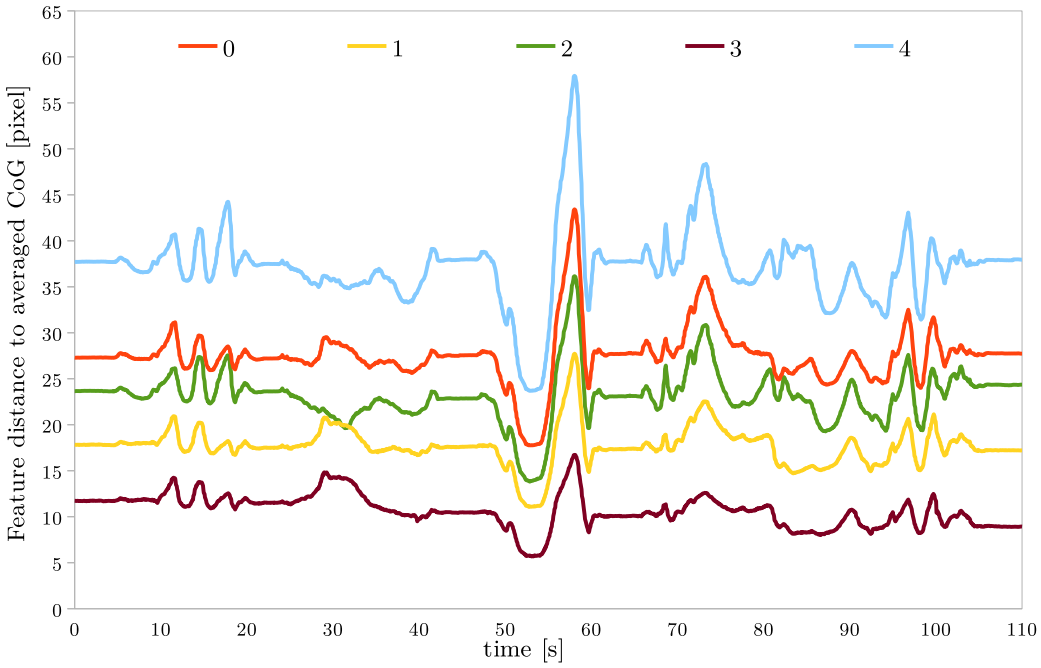

Figure 5.7: VS experiment: The distance of individual features to the current CoG with respect to time.

5.3 Demonstrator Evaluation

The last and most important experiment is focused on the verification of a complete demonstrator package, by analyzing the functionality in another complicated scenario. It was shown in the previous Sec. 5.2, that the VS method is capable of a high precision attitude control. Applying VS on a real hardware is a necessity in proving the concept. Therefore, the focus of this experiment lies primarily on the main capability of the demonstrator hardware to reliably minimize the error $\boldsymbol{e}$ between the current and the desired features. The experiment consisted of an acquisition of features by the demonstrator and consequent arbitrary moving of it with focus on limit orientation changes. Joint velocities, angles and feature coordinates were again recorded for later analysis. Experiment lasted for 100 s until a pose, that the system was not able to adjust to, was reached.

Parameters of feature tracking and visual servoing were selected by many iterations and testing. Again, the values are adjusted to produce best possible results in this specific scenario, and might not perform adequately in another. Despite that, for the clarity reasons, used parameters are listed in Table 5.3.

A set of figures, that shows the range of positions and orientations of the demonstrator during the experiment is shown in Fig. 5.8. Notice the red camera casing that repeatedly points to the middle of the poster, where the features are detected.

Feature detection and Tracking	
maximum features [/]	5
detection square area [pixel]	50
quality parameter [/]	0.07
detection window [pixel]	12
tracking window [pixel]	14
pyramid levels [/]	5
Visual Servoing	
adaptive gain [/]	0.4 to 9, slope -10
control update frequency [hz]	12

Table 5.3: Demonstrator evaluation experiment parameters.

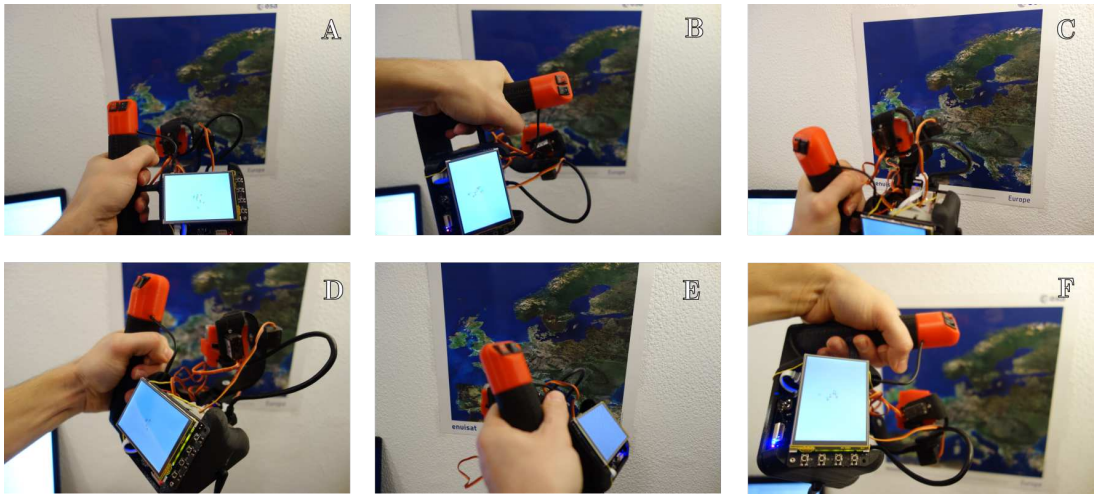

Figure 5.8: Demonstrator evaluation: Examples of the achieved positions and orientations during the VS experiment.

Unlike in the previous Sec. 5.2, the change of orientation of the camera is desired, and thus the joint angular velocities were computed from the constantly updating GJ matrix ${}^{e}\boldsymbol{J}_{e}$. Jacobian is updated based on the joint angles computed in the previous iteration. Because of the manipulator, and consequently camera orientation is controlled, the internal view changes comparably less than in the previous constant Jacobian scenario. Nonetheless, the internal view does change more than was anticipated. The most distinctive moments of the experiment, visualized as the demonstrator internal view, are shown in Fig. 5.9. Fig. 5.11 and Fig. 5.12 shows the joint angles and angular velocities respectively, computed during the experiment. The initial values of joint angles originates from the direct kinematic model of this particular

demonstrator.

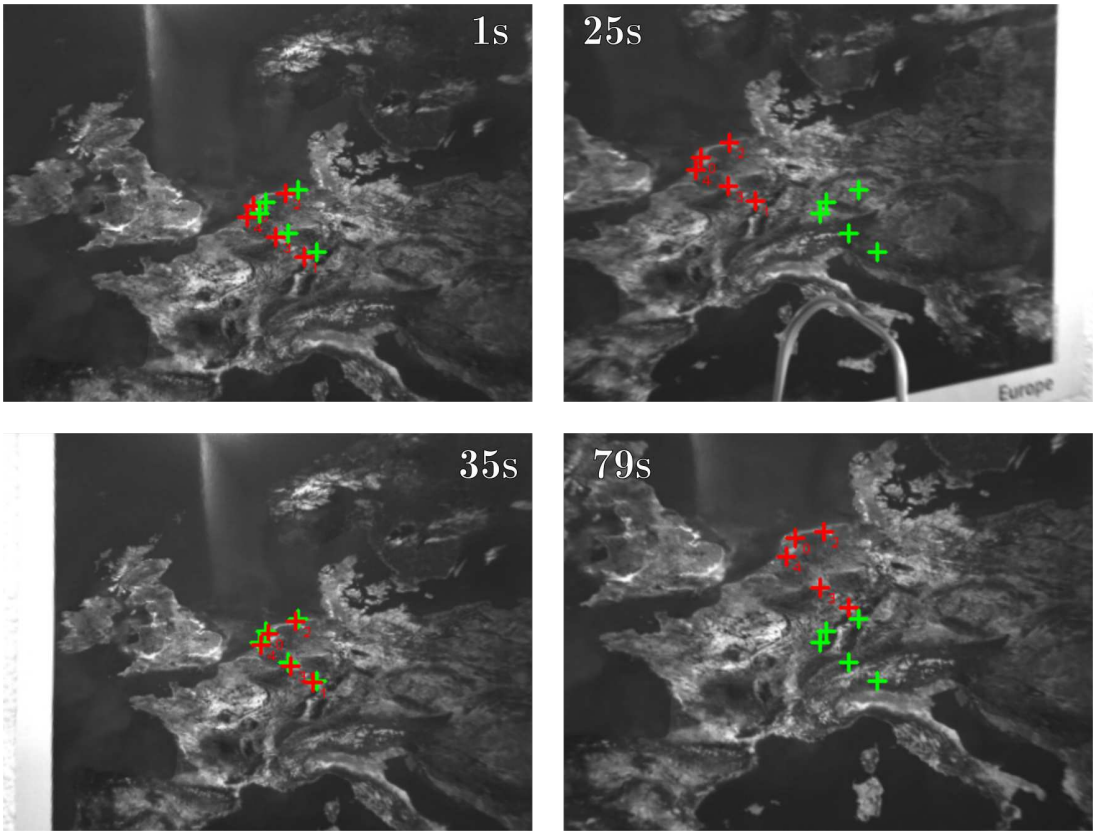

Figure 5.9: Demonstrator evaluation: The demonstrator internal view, distinctive moments at 1s, 25s, 35s and 79s

5.3.1 Discussion

The experiment started close to the position shown in Fig. 5.9: 1s. Small error is already visible, as the demonstrator was not perfectly static. The manifestation of this small movement is already visible in the computed joint angular velocities in first seconds of Fig. 5.12. The following rotation as is shown in Fig 5.8: $\boldsymbol{B}$, can be traced to moment at 7 s, from the high peak in angular velocity of $\dot{q}_3$ and the change of the angle q_3. After the desired rotation was reached, the angular velocity returned to zero as would be expected. It can be traced from the roll angle q_3, that the orientation of the camera was similarly changed another two times. Fig. 5.9: 25s is an example of the delay in the system. The tracked features in red, are shown in the moment, when the control mechanism and servo motors did not yet managed to adjust for the change in orientation. The effect of such an error in current and desired features is translated to big peak in angular velocities, as is visible at 25 seconds in Fig. 5.12 and results in a fast change in angles. After the system adapted to the peak change, and external motions become minimal, the demonstrator was able to minimize the error and reached a flat region for all controlled angles

in the interval 33 s to 37 s. Another example of the ongoing delay between feature tracking and is clearly visible at 79 s where the demonstrator was rotated in a downward and right direction faster than the control was able to accommodate. The algorithm however speedily corrected the error. After examination of the Fig. 5.11 and 5.12, the figures at 5.8 can be traced backward to the particular time intervals of the experiment.

The aforementioned continuous delay is caused partially by the time needed for image processing, namely image resizing and the distortion removal. Next the delay is caused by processing the data received by `ROS Visual_Servoing_node` from the `Feature_node`, arduino serial communication and servo control and the demonstrator fast movements. These reasons combined, results in a lagging period. All `ROS` nodes running in athe control loop were able to operate at $\approx 12\ hz$. Higher rate was not achievable due to relatively low computational power of the `Odroid C2` board, missing optimization and distributing the tasks into separate nodes. Besides the processing lag, the Arduino itself and cheap servos are considered to be the next significant reason for broad error distribution. The servo motors that are used at the demonstrator offer a simple position control. But a jitter motion is present due to practically no motion dynamics, discrete position update, or any speed control. Despite the update rate, the Arduino subjectively underachieved and updated the servo positions slower. The aforementioned unexpected magnitude of change in internal view is better summed in the scatter plot of the in frame averaged error in Fig 5.10. The error distribution of non-negligible magnitude is however not considered to be a significant issue. It is necessary to take into account the almost full hemisphere of the orientation changes of the demonstrator during the experiment and the demonstrator motion speed of $\approx 60°/s$, that the control system had to cover for. This experiment showed that the demonstrator performs its function, and that the VS system can be build into a small package for the purposes of VBAC.

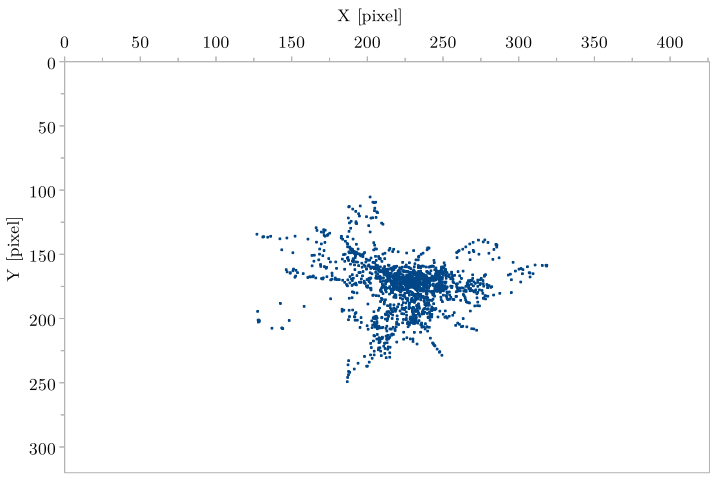

Figure 5.10: Demonstrator evaluation: A scatter plot of the CoG of error in the image plane.

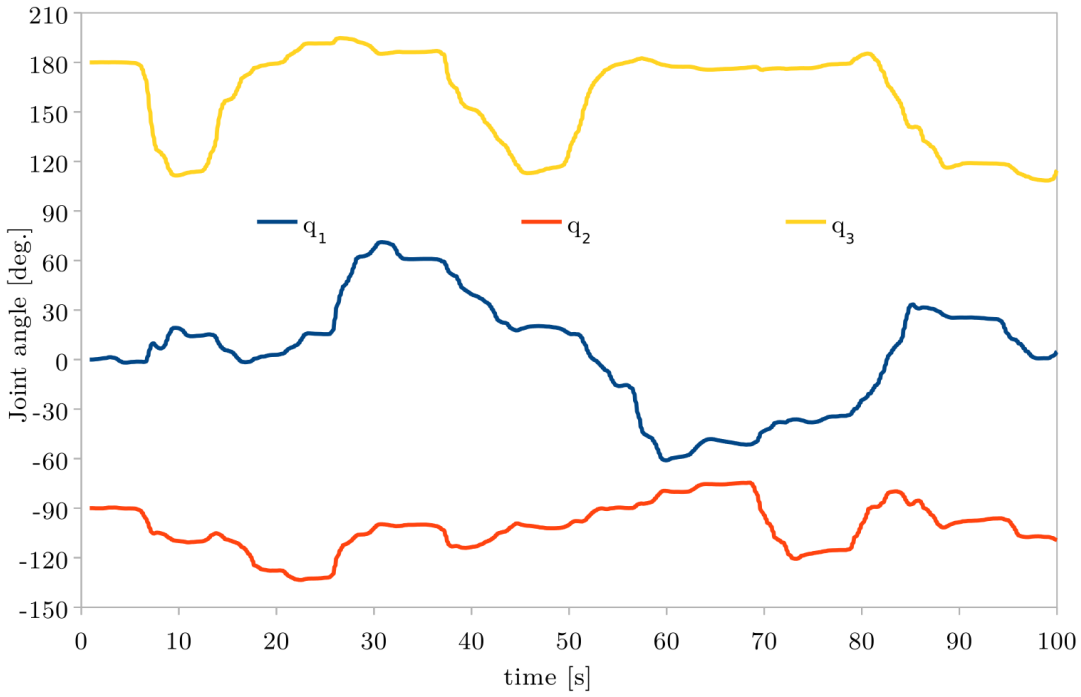

Figure 5.11: Demonstrator evaluation: Computed joint angles applied to the demonstrator during the experiment.

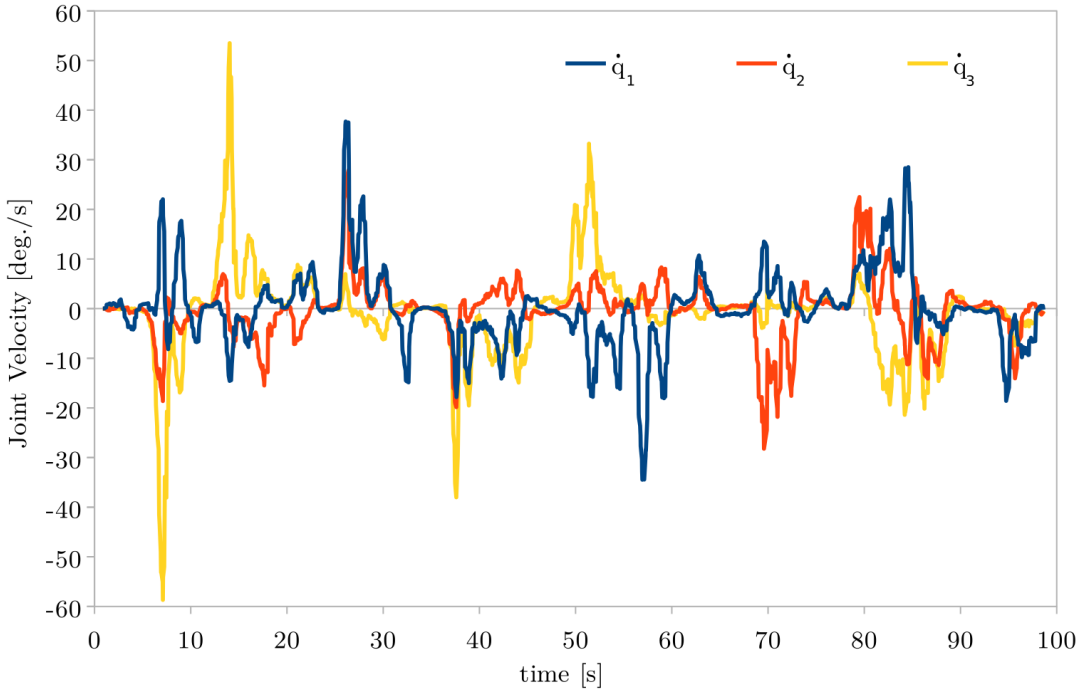

Figure 5.12: Demonstrator evaluation: joint angular velocities computed by the control law.

Chapter 6

Conclusion

This Thesis investigated the Visual Servoing method for the purpose of Vision Based Attitude Control. The conclusion of the Thesis with a summary of the addressed topics is shortly discussed. It is then followed by an outline of potential improvements and future work.

6.1 Summary

The attitude control is a broad and important topic in the Space applications and Earth observation. The VBAC provides an alternative to well established techniques based on the Sun/Stars/Horizon sensors or the gyroscope/magnetic field measurements and precalculation. The VBAC is dependent only on one or more cameras that are usually already present on a Spacecraft, simplifying the construction, lowering the general complexity of the Spacecraft as well as the development related financial load. The pointing precision, e.g. the measure of goodness of attitude control, can offer sub-degree results and is mainly influenced by the camera resolution.

This thesis investigated the whole concept of the VBAC, from the camera image to attitude control commands. The practical work on this topic was executed as a proof of concept, i.e. a self sufficient demonstrator was designed, build and tested. Furthermore, the principles of VS were implemented into a functional program pack that controls the demonstrator.

Chapter 1 motivates the need for VBAC in general and provides a series of examples of previous work done on the topic, both in the air and Space. Main goals and outline of the work are further listed.
Chapter 2 offers an insight into the individual theoretical steps behind the investigated VBAC method. The chapter opens with description of basic camera model - the pinhole camera. The gross volume of the chapter focuses on the VS, the link between the camera image and the robot motion and the individual theoretical components needed for VS - feature detection, tracking and robot kinematic relations. The hardware and software components that are used are introduced at the end of the chapter.
Chapter 3 illustrates the demonstrator design process and requirements. The complete demon-

strator structure is shown.
Chapter 4 details the kinematic model of the demonstrator, a relevant GJ and the software architecture build for the thesis purpose.
Chapter 5 describes three experiments that were performed to test the individual components of the VS and the demonstrator. The first experiment investigates the capability of the implemented detectors to robustly and repeatably detect the features in the image. Both Shi&Tomasi and Harris detectors are considered to provide good results, because both detectors are capable to repeatably detect more than sufficient amount of features (from 85 to 91) in the 480×320 image with standard deviation of 1.08 for Shi&Tomasi detector. The inner working of the detectors is confirmed in the section figures, where it can be seen that the features are prominently detected in the places of a higher contrast, whereas the bleak and low contrast area of east Europe is feature-free. The next experiment was performed to test the performance of the VS and the system tracking capability. In the experiment, the joint velocities were successfully logged and feature distance to averaged CoG computed. The joint velocities are computed based on the constant GJ. The experiment shows that a limit orientation and pose changes have negligible effect on the features traceability. Equally, the flat regions of the velocities clearly shows where the camera was returned to the initial position. The experiment was concluded after 110 s with very low resulting mean subpixel error $e = 0.17 \approx 0.022°$. This low error value shows the VS is capable to operate at very precise manner, that can be further improved by increasing the resolution and adjusting the scene for higher contrast. The last experiment evaluated the complete demonstrator package. Again, as in the previous experiment, arbitrary orientation and pose changes were performed. The performance of the demonstrator was lower than desired, because the sum of individual processes delayed the update frequency significantly. Low update frequency together with comparatively fast induced orientation changes of the demonstrator, caused abrupt peaks in the computed joint velocities. Nonetheless, the demonstrator shows that it is able to successfully cover for the orientation changes, despite the aforementioned low update frequency and jittery servo control. Altogether the VS system and demonstrator are considered a to perform well enough for the purposes of VBAC. The performance can be improved though, specifically if further work and improvements will take place. These potential improvements and the future work are discussed in the next section.

6.2 Improvements and Future Work

The primary goal of the Thesis, to design a demonstrator able to perform VBAC, was fulfilled. Still, the implementation and the demonstrator itself, offers a wide opportunity for improvement.

The demonstrator hardware would benefit from sturdier joints, either by milling the frame parts, using bearings or primarily from using different motors. As was previously mentioned, the servomotors are cheap and offers very simple position control, but for the price of flimsy mounting to servo axis and jittery stepped motion. A geared dc motor accompanied by a precise rotary encoder would be a complicated, but a significant improvement.
The demonstrator can be further improved by optimizing the performance of the system. One of the options is to combine individual node components into one program, consequently dropping

the complexity and inter-node communication delays. This option was considered in the beginning, but refused for the debugging and clarity reasons. Another improvement option might be found in the image acquisition and processing. The camera is capable of 45 fps, but does not offer native resolution smaller than 640×480 pixels, only a cutout. Therefore, to keep the FoV at 65°, the image is resized at each individual frame. Resizing and distortion removal causes significant computational load and slows down the demonstrator. This problem can be alleviated by either different camera or by finding a mean to improve the image processing performance. The VBAC functionality can be further enhanced by a regular feature reinitialization. This step can practically cancel the buildup tracking error and would allow for continuous attitude control where a very big change in the scene is expected. Example for such application is an increased time of surveillance of ground by a Satellite in orbit.

The majority of the mentioned improvements represents an inclusion of unnecessary complexity to the demonstrator system and therefore were not implemented at the moment. However for any practical usage of such a VBAC system or device, other than demonstration, a set of improvements is indisputable. The list of planned future work consists at the moment of two parts. First minor step is the performance improvement in the image processing as previously mentioned. Second, and major planned progress milestone is to test the VBAC on a precision table capable of micro-degrees orientation changes, using the full camera resolution and a PC for processing. Those steps are expected to improve the pointing accuracy of the VBAC system by at least one order and will provide a mean of precise measurement of the system capability.

List of Figures

List of Tables

List of Acronyms

ATV	Automated Transfer Vehicle
CoG	Center of Gravity
D-H	Denavit - Hartenberg
DoF	Degree of Freedom
EiH	Eye in Hand
EtH	Eye to Hand
FoV	Field of View
fps	Frames per Second
GJ	Geometric Jacobian
GPS	Global Positioning System
IBVS	Image Based Visual Servoing
ISS	International Space Station
PBVS	Position Based Visual Servoing
ROS	Robotic Operating System
SIFT	Scale-invariant Feature Transform
SSD	Sum of Squared Differences
UAV	Unmanned Aerial Vehicle
VBAC	Vision Based Attitude Control
ViSP	Visual Servoing Platform
VS	Visual Servoing

Bibliography

[1] Andrea Antonello and Panagiotis Tsiotras. Vision-based attitude determination using a slam algorithm during relative circumnavigation of non-cooperative objects.

[2] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004.

[3] Jean-Yves Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5(1-10):4, 2001.

[4] Matthew Brown, Richard Szeliski, and Simon Winder. Multi-image matching using multi-scale oriented patches. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 510–517. IEEE, 2005.

[5] Peter J Burt. Local correlation measures for motion analysis: a comparative study. In *Proc. Pattern Recognition and Image Processing Conf., Las Vegas, 1982*, 1982.

[6] Andrés Castaño and Seth Hutchinson. Visual compliance: Task-directed visual servo control. *IEEE transactions on Robotics and Automation*, 10(3):334–342, 1994.

[7] Francois Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. *The confluence of vision and control*, pages 66–78, 1998.

[8] François Chaumette. *Robot Visual Control*, pages 1188–1194. Springer London, London, 2015.

[9] François Chaumette and Seth Hutchinson. Visual servo control. i. basic approaches. *IEEE Robotics & Automation Magazine*, 13(4):82–90, 2006.

[10] François Chaumette and Seth Hutchinson. Visual servoing and visual tracking. pages 563–583, 2008.

[11] François Chaumette, Seth Hutchinson, and Peter Corke. Visual servoing. In *Springer Handbook of Robotics*, pages 841–866. Springer, 2016.

[12] D Connor and J Limb. Properties of frame-difference signals generated by moving images. *IEEE Transactions on Communications*, 22(10):1564–1575, 1974.

[13] Emilio De Pasquale. Atv jules verne: a step by step approach for in-orbit demonstration of new rendezvous technologies. In *Proc. SpaceOps Conference, Stockholm*, 2012.

[14] Cédric Demonceaux, Pascal Vasseur, and Claude Pégard. Uav attitude computation by omnidirectional vision in urban environment. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2017–2022. IEEE, 2007.

[15] Bernard Espiau, François Chaumette, and Patrick Rives. A new approach to visual servoing in robotics. *ieee Transactions on Robotics and Automation*, 8(3):313–326, 1992.

[16] Itseez et al. OpenCV - Open Source Computer Vision Library. `https://http://opencv.org/`, 2017.

[17] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Manchester, UK, 1988.

[18] Richard Hartley and Andrew Zisserman. Multiple view geometry in computer vision second edition. *Cambridge University Press*, 2000.

[19] Seth Hutchinson, Gregory D Hager, and Peter I Corke. A tutorial on visual servo control. *IEEE transactions on robotics and automation*, 12(5):651–670, 1996.

[20] Myung Hwangbo and Takeo Kanade. Visual-inertial uav attitude estimation using urban scene regularities. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2451–2458. IEEE, 2011.

[21] Direct Imaging. Beholder EC-1 3 Axis DSLR Handheld Gimbal. `https://www.directimaging.com.my/wp-content/uploads/2016/09/beholder-Ec-1-1-.jpg`, 2017.

[22] Gregor Klančar, Sašo Blažič, Drago Matko, and Gašper Mušič. Image-based attitude control of a remote sensing satellite. *Journal of Intelligent & Robotic Systems*, 66(3):343–357, 2012.

[23] Toru Kouyama, Atsunori Kanemura, Soushi Kato, Nevrez Imamoglu, Tetsuya Fukuhara, and Ryosuke Nakamura. Satellite attitude determination and map projection based on robust image matching. *Remote Sensing*, 9(1):90, 2017.

[24] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[25] Ezio Malis. Improving vision-based control using efficient second-order minimization techniques. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 2, pages 1843–1848. IEEE, 2004.

[26] Ezio Malis, Francois Chaumette, and Sylvie Boudet. 2 1/2 d visual servoing. *IEEE Transactions on Robotics and Automation*, 15(2):238–250, 1999.

[27] Éric Marchand, Fabien Spindler, and François Chaumette. Visp for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Robotics & Automation Magazine*, 12(4):40–52, 2005.

[28] Micromo. Esa-hexpod. `https://www.micromo.com/media/wysiwyg/Applications/Aerospace/hexpod1.jpg`, 2017.

[29] Hans P Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical report, STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1980.

[30] Odroid - Hardkernel co.,Ltd. `http://www.hardkernel.com`.

[31] ROS - Robot Operating System. `http://www.ros.org/about-ros/`.

[32] Arthur C Sanderson and Lee E Weiss. Adaptive visual servo control of robots. In *Robot vision*, pages 107–116. Springer, 1983.

[33] Abd El Rahman Shabayek, Cédric Demonceaux, Olivier Morel, and David Fofi. Vision based uav attitude estimation: Progress and insights. *Journal of Intelligent & Robotic Systems*, 65(1-4):295–308, 2012.

[34] Jianbo Shi et al. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[35] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer, 2016.

[36] Mark W Spong and Mathukumalli Vidyasagar. *Robot dynamics and control*. John Wiley & Sons, 2008.

[37] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[38] ViSP - Visual Servoing Platform. `https://visp.inria.fr/`.

[39] Guo-Qing Wei and Song De Ma. Implicit and explicit camera calibration: Theory and experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):469–480, 1994.

[40] Aurelien Yol, Eric Marchand, Francois Chaumette, Keyvan Kanani, and Thomas Chabot. Vision-based navigation in low earth orbit. In *Int. Symp. on Artificial Intelligence, Robotics and Automation in Space, i-SAIRAS'16*, 2016.

[41] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.

Proclamation

Hereby I confirm that I wrote this thesis independently and that I have not made use of any other resources or means than those indicated.

Würzburg, January 2018