Vision Based Pose Computation from Landmarks: an application to Quadrotors

André Filipe Marques da Silva

ISR & Instituto Superior Técnico, Technical University of Lisbon, Portugal.
andremsilva@ist.utl.pt

Abstract

Worldwide, UAVs have become an important tool for the realization of different tasks that, otherwise, would have to be performed by humans in, sometimes, difficult and dangerous conditions. Even so, there is plenty of space for improving the technical capabilities and functionalities of these vehicles. Particularly, this thesis aims at the development of a module for providing pose error measurements to help stabilize the vehicle's position when it is airborne. This module is composed by lightweight hardware components that can easily be attached to an UAV and uses visual information acquired by a camera for error estimation. The first part of this thesis presents the theoretical background needed to the rest of the work developed. Consisting of some basic computer vision concepts such as camera models and image transformations and the description and analysis of features detectors and descriptors algorithms. The second part presents the developed solution, its implementation and the experimental results. The developed module was attached to a robotic arm and, by closing the loop with a control feedback law, the tests performed shown that the end effector's pose error is correctly estimated and can be driven to zero.

Keywords: *Unmanned Aerial Vehicles; Visual Pose Estimation; Features Detectors; Features Descriptors*

1 INTRODUCTION

1.1 Context, Motivation and Problem Description

The main objective of this project is to develop advanced robotic tools and techniques for the inspection of critical infrastructures, like bridges and dams. Considering the case of dams, for example, these infrastructures require a periodic monitoring program. This program, normally consists of a team of technicians who have to examine, possibly using climbing equipment or mobile platforms, all the infrastructure looking for damages, such as cracks. A more practical and safer alternative to this possibly dangerous activity, would be to use an autonomous unmanned aerial vehicle (UAV) that could capture images from the whole infrastructure surface and send them to a base station. For a UAV to navigate over all the infrastructure surface, it will need a navigation, guidance and control system. One of the most basic functionalities of these systems is to stabilize the vehicle around a desired position. The present paper aims at developing a module that helps on this activity.

Consider the following example of usage: the vehicle is teleoperated into a desired pose where a wall is seen by the camera. Due to winds and other factors, the vehicle will be dragged from the desired pose. Using only a camera and the reference image captured, the developed solution will have to be able to calculate the error of translation and rotation, from the current pose to the desired one. Providing this information to a control feedback law that can the vehicle to the desired pose.

1.2 Related Work

Altüg et al. used a camera located on the ground to estimate the pose of a quadrotor in order to achieve a stable flight [1]. Later, they extended their work by using a second camera on-board the quadrotor[2]. Earl and D'Andrea estimated the quadrotor by using a kalman filter with on board gyroscopes measurements and off board vision sensor measurements [3]. Regarding the challenge of flying a quadrotor indoors, Romero et al., proposed a simple vision system using off board computation hardware [4]. All these solutions require a ground station to process the visual information, making the vehicle dependent upon a continuous connection with ground. Concerning on board only solutions, Mondragón et al., Martinez et al. and Bourquardez et al. works estimate the pose of an UAV at real time using an on board camera [5] [6] [7]. Similarly to the approach chosen on this paper, they explore the information obtained by the projective transformation of planar opbjects into a calibrated camera. But the plane that is captured is previously known, such as a landmark or an helipad for example.

1.3 Main Contributions

We developed a modulw which we called Gumstix-and-Camera module. It is composed by ultra-compact and lightweight hardware: a Gumstix Computer-On-Module, for performing all the computations, and a Caspa VL camera, for image capturing. Together with the developed algorithm , this module is able to compute the camera pose error in relation to a reference image, without the need of special visual marks or of any other type. The only restriction is that the scene captured at the reference image is planar. Finally, the Gumstix-and-Camera module is attached to a robotic arm and is used to close the control loop driving the arm end effector into the pose corresponding to the reference image, making the pose error converge to zero.

1.4 Outline

The remaining of this paper is organized as follows. Section 2 provides basic concepts from computer vision necessary for the work developed. Section 3 discusses the algorithms available to

detect and describe features on images. Section 4 describes the algorithm developed and the hardware used to test it. Section 5 shows the results of the tests realized. On the final sections, the results are discussed and final remarks are provided along with possible future work.

2 BACKGROUND

Camera pinhole model. Using the Camera Pinhole model [8], a point coordinates in 3-D, $\boldsymbol{X} = (X, Y, Z, 1)^T$ can be related to its correspond 2-D image coordinates, $\boldsymbol{x} = (x, y, , 1)^T$ by

$$\boldsymbol{x} = \frac{1}{Z}\mathsf{K}[\mathtt{I}|\mathbf{0}]\boldsymbol{X}. \tag{1}$$

where

$$\mathsf{K} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

is the *camera calibration matrix*

Planar Homography Consider two images of an object that are captured by a camera from two different positions. The image of the object is not the same in both images. Nevertheless, the two images may be related by a *2-D transformation* [9], i.e., a transformation that occurs in the 2D plane. A particular case of 2-D transformation is called *homography*, *projective transform* or *perspective transform*. It has the form

$$\boldsymbol{x}' = \boldsymbol{H}\boldsymbol{x} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \boldsymbol{x} \tag{3}$$

The matrix $\boldsymbol{H}$ is homogeneous and, as so, is defined up to a scale factor. Two matrices that differ only by a scale represent the same transformation.

Consider two images of points p on a 2D plane P in 3-D space (Figure 1). Assuming that the optical center of the camera never passes through the plane, it is possible to define a transformation that maps a image $\boldsymbol{x}_1$ of a point $p \in P$ into the second image $\boldsymbol{x}_2$ of the same point. To this transformation we will call *planar homography* [10].

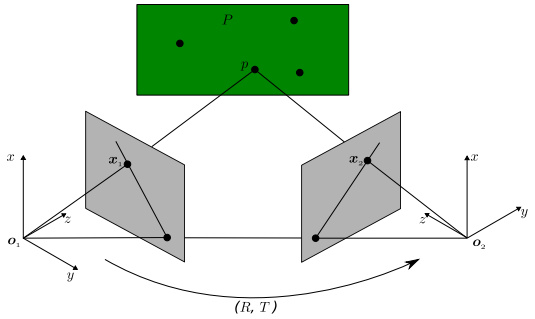


Figure 1: Planar homography

Consider two orthonormal reference frames with origins o_1 and o_2 located at the optical centers of the cameras that took the two different images of the same scene. For simplicity, and without loss of generality, assume the world frame to be one of the cameras. If we call the 3-D coordinates relative to the camera frame of a point p, $\boldsymbol{X}_1 \in \mathbb{R}^3$ and $\boldsymbol{X}_2 \in \mathbb{R}^3$, then the following relation holds

$$\boldsymbol{X}_2 = \boldsymbol{R}\boldsymbol{X}_1 + \boldsymbol{t} \tag{4}$$

where $\boldsymbol{R}$ is a 3x3 rotation matrix and $\boldsymbol{t}$ is a 3x1 translation vector that relate the two reference frames. Defining the unit normal vector of the plane P with respect to the first camera frame as $\boldsymbol{n} = [n_1, n_2, n_3]^T$ and the distance from the plane P to the optical center of the first camera as $d > 0$ the following relation holds

$$\boldsymbol{n}^t\boldsymbol{X}_1 = n_1X + n_2Y + n_3Z = d \quad \Leftrightarrow \quad \frac{1}{d}\boldsymbol{n}^t\boldsymbol{X}_1 = 1, \quad \forall \boldsymbol{X}_1 \in P \tag{5}$$

Applying equation (5) in equation (4) we get

$$\boldsymbol{X}_2 = \boldsymbol{R}\boldsymbol{X}_1 + \boldsymbol{t} = \boldsymbol{R}\boldsymbol{X}_1 + \boldsymbol{t}\frac{1}{d}\boldsymbol{n}^t\boldsymbol{X}_1 = \left(\boldsymbol{R} + \frac{1}{d}\boldsymbol{t}\boldsymbol{n}^t\right)\boldsymbol{X}_1. \tag{6}$$

Defining the matrix

$$H = \left(\boldsymbol{R} + \frac{1}{d}\boldsymbol{t}\boldsymbol{n}^t\right) \tag{7}$$

which represents the transformation from $\boldsymbol{X}_1$ to $\boldsymbol{X}_2$ leads to

$$\boldsymbol{X}_2 = H\boldsymbol{X}_1. \tag{8}$$

Matrix H is called the *planar homography matrix* and, as it can be seen, depends on the motion parameters $(\boldsymbol{R}, \boldsymbol{t})$ and the structure parameters $(\boldsymbol{n}, d)$. Because the translation vector is scaled by the distance d in the term $\frac{1}{d}\boldsymbol{t}\boldsymbol{n}^t$, it can only be expected to extract the direction. but not the module of the vector.

Considering image coordinates we can write

$$\lambda_1\boldsymbol{x}_1 = \boldsymbol{X}_1, \quad \lambda_2\boldsymbol{x}_2 = \boldsymbol{X}_2, \quad \boldsymbol{X}_2 = H\boldsymbol{X}_1 \tag{9}$$

and obtain

$$\lambda_2\boldsymbol{x}_2 = H\lambda_1\boldsymbol{x}_1 \quad \Leftrightarrow \quad \boldsymbol{x}_2 \sim H\boldsymbol{x}_1 \tag{10}$$

This last equation is known as *planar homography mapping*[10].

Having the correspondence between points of two images, the planar homography matrix can be calculated by using RANSAC [11], for examble, to eliminate outliers. Then to decompose the planar homography matrix into $\boldsymbol{R}$ and $\frac{1}{d}\boldsymbol{t}$ the method described in [10] can be used. The form of the four solutions provided by this method are summarized in Table 1.

Table 1: The four possible solutions to the decomposition of the planar homography matrix

Solution 1	$\boldsymbol{R}_1 = \boldsymbol{W}_1\boldsymbol{U}_1^T$ $\boldsymbol{n}_1 = \mathsf{S}(\boldsymbol{v}_2)\boldsymbol{u}_1^T$ $\frac{1}{d}\boldsymbol{t}_1 = (H - \boldsymbol{R}_1)\boldsymbol{n}_1$	Solution 3	$\boldsymbol{R}_3 = \boldsymbol{R}_1$ $\boldsymbol{n}_3 = -\boldsymbol{n}_1$ $\frac{1}{d}\boldsymbol{t}_3 = -\frac{1}{d}\boldsymbol{t}_1$
Solution 2	$\boldsymbol{R}_2 = \boldsymbol{W}_2\boldsymbol{U}_2^T$ $\boldsymbol{n}_2 = \mathsf{S}(\boldsymbol{v}_2)\boldsymbol{u}_2^T$ $\frac{1}{d}\boldsymbol{t}_2 = (H - \boldsymbol{R}_2)\boldsymbol{n}_2$	Solution 4	$\boldsymbol{R}_4 = \boldsymbol{R}_2$ $\boldsymbol{n}_4 = -\boldsymbol{n}_2$ $\frac{1}{d}\boldsymbol{t}_4 = -\frac{1}{d}\boldsymbol{t}_2$

Two solutions can be eliminated by applying the positive depth constraint, i.e. by imposing $\boldsymbol{n}^T e_3 = n_3 > 0$. For the remaining two, a third image of the same plane or a second plane on the same images can be used [12]. Either way, a new homography matrix

will be available. After decomposing this second homography and getting two new possible solutions, it is only a question of finding the common solution on the two sets of solutions. A third option is available, one can calculate a rotation matrix from an IMU readings, compare it with the two possible solutions and choose the closest one.

3 Image Features

As seen in section 2, if a planar scene is being captured by a camera, it is necessary to have the correspondence between the 2-D coordinates of points to extract the information about the camera pose change between images.

Common requirements of the applications that require a visual tracking system are the system to be robust and fast enough to be computed in real time. Although some systems work with optical flow, we are only interested in feature-based visual tracking because optical flow is prone to long-term drift, as the motion estimates and therefore the errors are integrated over time, and is only feasible for smooth motion.

Although different, all feature-based approaches start with two essential steps: interest points detection and features description.

Mainly, three types of points detectors can be found among the literature: Corner Detectors [13], [14] and [15], Blob Detectors [16], [17] and[18] and Affine-Invariant Detectors like [17].

On the field of feature descriptors there are a few approaches that stand out. There are some earlier ones such as [19], that use derivatives to achieve rotation invariance, or [20] that make use of derivatives of Gaussians of different order. But, is the work of Lowe with SIFT [17] that really stands out. It is invariant to changes in scale and rotation by assigning a local reference frame in relation to a dominant scale and rtotaion previously computed around the feature point, using local histograms on a square grid. As a faster alternative to SIFT, Bay et al. [18] introduced Speeded-Up Robust Features (SURF). It uses similar SIFT approaches but makes use of integral images [21] and approximations of the expensive Gaussian filters using response box filters to speed up the computations to constant time. More recently, Calonder et al. [22], developed the Binary Robust Independent Elementary Features (BRIEF), which uses only binary strings as an efficient alternative for feature point description. It makes use of the Hamming distance which is very efficient to compute when compared to the L_2 norm usually used. Other approaches such as [23] and [24] exist that use trained classifiers but they have the drawback of previously requiring a training phase.

Taking into account the good results documented in the literature regarding SURF and BRIEF , these are the main two approaches that are tested in the project.

SURF

Interest points detection In the task of finding an object in an image, instead of searching for the object as a whole, it is usual to search for objects interest points. This type of approach is chosen for several reasons, from which the main ones are the computational cost of searching in such a high-dimensional data as the one stored in images, and the high level of redundancy incorporated, because pixels do not move independently and have a high level of correlation. There are several methods to define and detect interest points, ranging from the ones that consider corners as interest points to the ones that consider blobs instead. SURF's implementation used the *fast hessian* detector [18] that detects blob-like features (Figure 2)..

Figure 2: An example of the type of features that the Fast Hessian Detector detects

Features detection is performed at different scales because interest points may be compared between images where they are seen at different scales. The scale space is implemented as an image pyramid. Without box filters, usually, image pyramid is built by repeatedly smooth the image with a Gaussian and then sub-sample it in order to achieve a higher level of the pyramid. Fortunately, with box filters and integral images, there is not the need to filter the image iteratively and sub-sample it. Instead, it is the filter that is up-scaled and applied at exactly the same speed on the original image.

In order to classify a point a of interest or not a non-maximum suppression in a $3\times3\times3$ is applied in scale and image space. Each sample is compared to its 8 neighbors in the current image and the 18 neighbors of the "bigger" and "smaller" images in scale space. If it has the biggest score (Hessian matrix determinant) of its neighbors then it is considered an interest point, otherwise it is discarded.

Once it is classified as an interest point, the location is refined to subpixel accuracy by fitting a parabola to the sample point and its immediate neighbors [25].

Interest point orientation assignment To achieve rotation invariance, Haar wavelet filters are used wavelet responses are calculated around a circular neighborhood that depend of the scale at which the feature was detected.

The filter responses are weighted with a Gaussian centerd at the point and represented in Cartesian coordinates system. Using a orientation sliding window (Figure 3), the sum of all responses within the window are calculated. The orientation of the window that has the greatest summed value is the orientation that is assigned to the point.

Descriptor vector The standard SURF descriptor consists of a vector with 64 entries. To build this vector, a square region, centered at the interest point and with the orientation selected in the previous step is used. This region is split up into 4×4 square sub-regions. Using Haar wavelets filters, the filter responses at 5×5 equally spaced sample points are calculated in the x and y direction. Note that these directions are defined in relation to the square region orientation. But instead of rotating the image itself, the filter responses are calculated in the unrotated image and then interpolated.

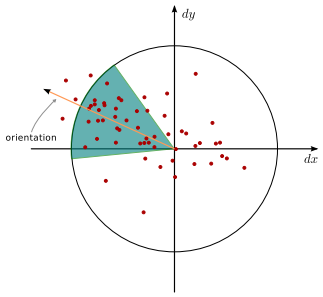


Figure 3: Sliding window used to assign the orientation

After weighting the filter responses using a Gaussian with $\sigma = 3.3$ centerd at the interest point, four sums in each sub-region are calculated: the sums of dx and dy and, to have information about the polarity of the intensity changes, the sums of $|dx|$ and $|dy|$.

An interest point in one image, is considered to *match* another interest point in other image if they are close enough in the *nearest neighbor* sense. For this purpose the FLANN library [26] is used. It is a C++ library for performing fast approximate nearest neighbor searches in high dimensional spaces.

BRIEF The main difference between BRIEF and SURF are described next.

BRIEF descriptor The BRIEF descriptor differs from the SURF descriptor in the sense that it directly computes binary strings from image patches. Each bit is obtained by comparing intensities of pairs of points. Also, by constructing the descriptor this way, it enables the possibility of using the Hamming distance to compare strings. The use of this distance has the advantage of only requiring a bitwise XOR operation and a bit count which can be performed much faster than other standard distance measures.

Intensity test The test τ to determine each descriptor's bit is defined as follows:

$$\tau(\mathbf{p}; \boldsymbol{x}, \boldsymbol{y}) := \begin{cases} 1 & \text{if } \mathbf{p}(\boldsymbol{x}) < \mathbf{p}(\boldsymbol{y}) \\ 0 & \text{otherwise} \end{cases}, \tag{11}$$

where $\mathbf{p}$ is the $S \times S$ image patch and $\mathbf{p}(\boldsymbol{x})$ is the smoothed pixel intensity at coordinates $\boldsymbol{x} = (u, v)^T$. So, if the pixel intensity at $\boldsymbol{x}$ is lower than the one at $\boldsymbol{y}$ the correspondent bit will be 1, otherwise it will be 0.

Spatial arrangement of tests The authors of [22] found that the patch positions at which the tests will be performed that leads to the highest recognition rate are the ones that are sampled from an isotropic Gaussian distribution. The locations following this distribution are calculated on the initialization phase of the algorithm, from there, the locations will be the same for every feature descriptor that is calculated.

Descriptor bitstring Finally, having the location pairs $(\boldsymbol{x}, \boldsymbol{y})$ defined, the n_d-dimensional descriptor string is defined as

$$f_{n_d}(\mathbf{p}) := \sum_{1 \le i \le n_d} 2^{i-1} \tau(\mathbf{p}; \boldsymbol{x}_i, \boldsymbol{y}_i). \tag{12}$$

The authors of [22] conclude that, for image pairs with short baseline, using $n_d = 256$ yields near optimal results and that for all other cases $n_d = 512$ perform better.

Feature detector Calonder et al., do not propose any special feature detector, any option available on the literature can be used. The chosen one was FAST (Features from Accelerated Segment Test) [27].

FAST is based on the Accelerated Segment Test (AST). This test, which is simplified version of SUSAN (Smallest Univalue Segment Assimilating Nucleus) [28], consists on defining a circle of radius r around the candidate point (Figure 4). Then, if at least n contiguous pixels are all brighter or all darker than the center pixel by at least t, the point is considered a feature. Specifically, FAST's tests shown that better results are achieved by using $r = 3$ (circunference of 16 pixels) and $n = 9$. The order by which the circle pixels are tested can speed up considerably the overall performance. This order is chosen by building a decision tree from the ID3 algorithm [29] applied to a training set of images. Contrasting to SURF's Fast Hessian Detector, that detects blob-like features, FAST detects corner-like features.

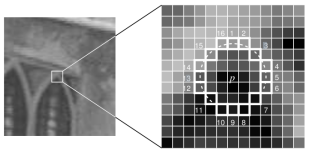


Figure 4: Illustration of the test circle. The pixel at p is the center of a candidate corner. The arc indicated by the dashed line passes through 12 contiguous pixels which are brighter than p by more than the t. Figure adapted from [27].

Scale and Rotation Invariance On the original work from Calonder et al., BRIEF is not invariant to scale change or rotation. To achieve that, a pyramidal approach was used. When the template image is being processed, several copies of the original image are generated by scaling it and rotating within a 360 degrees span. To each of these generated imaged FAST and BRIEF algorithms are applied. This will lead to several descriptors of the same features but rotated and scaled, thus, achieving scale and rotation invariance.

4 Implementation

Concerning the main algorithm developed, it is divided into two main parts. The first one, consists in detecting the template plane present on a video stream. The two features detectors and descriptors described before were implemented in order to do so. By doing this, it will be possible to compare them, regarding computational efficiency, robustness and precision (see Chapter 5). The second main part of the developed algorithm consists of using the coordinates of the frame already detected on the video stream to estimate the pose of the camera that is acquiring it. This is done by calculating the homography matrix that maps the coordinates of the template plane detected on the current frame into the coordinates of the template plane on the reference frame. With this matrix, the homography decomposition method is applied and two possible solutions, each one consisting of a translation vector and a rotation matrix, are calculated. The true solution is chosen by, either using the IMU data of the vehicle where the camera is attached, or by using a third image containing the reference plane too but captured at a different pose.

All code was written using the C++ language for computational efficiency. Because C++ does not natively provide an API to easily work with matrices, the OpenCV library [30] was also used. It is a computer vision library that is free for academic and profession use and provides an API with several computer vision algorithm already implemented.

Regarding the hardware where the code will run, because one of the final objectives of the developed algorithm is to be used on board of aerial vehicles, it was necessary to define a commitment between cost, computation power and size. It is required that the hardware is lightweight enough, but at the same time powerful enough to, hopefully, run the code at real time speed. Considering these requirements, the hardware that seemed more appropriate was the Gumstix Overo Fire. It belongs to the COM (Computer On Module) hardware class. Basically, it has an ARM Cortex-A8 CPU, RAM memory, storage memory and networking capabilities, all on a single board with an approximate size of a gum stick. It runs the Linux OS, and many expansion boards to extend the capabilities are available. For capturing video, firstly the e-cam50 camera module was considered, but due to better technical specifications the Caspa VL was chosen.

Although, as already stated, the main objective is to use the developed application on board of aerial vehicles, and a quadrotor is available at ISR labs. This was not the test bench used. Getting a quadrotor in the air requires a team of several people, and this is not practical to perform every time there is the need test some modifications on the code. Instead, a robotic arm was used. It is more practical to operate, requires only a single person and is conceptually identical to estimate the pose of a quadrotor, or the pose of the camera attached to the arm's end effector. The robotic arm available on the university facilities is the PUMA 500 model from Unimation with 6-DOF.

The remaining of this chapter is organized as follows. Section 4 contains a flowchart of the developed algorithm and explain the order of the steps performed. Sections 4, 4 and 4 describes with more detail all the hardware used. Finally, Section 4 shows all the hardware is connected, it provides a global view of the testing setup architecture.

Developed Algorithm The developed algorithm works as follows: the program starts acquiring frames from the camera. When the camera is at the desired pose the user sends a command and the template image is stored and learned by the chosen method (either SURF, BRIEF or TAG features are extracted). Then the main loop starts. A new frame is acquired and the features are extracted. Now, the features of the current frame are matched with the template frame features. Matched features coordinates are undistorted using the camera calibration matrix and distortion coefficients. The homography matrix is calculated using RANSAC and then decomposed into rotation matrix and translation vector. From the two possible solutions, the correct one is chosen based, either on a previously second frame taken from the same template or on the vehicle IMU. From the correct solution, the control command is calculated using the chosen control law and fed into the vehicle actuators. This procedure is repeated until the user requests the program to be stopped.

Gumstix Overo Fire The Gumstix series computers are very small, general purpose computers for embedded systems applications shiped with Linux 2.6 operating system. The name "gumstix" is due to the fact that each one of these computers are a COM (Computer On Module) with dimensions really close to a common stick of gum. The model used in this particular work is the Gumstix Overo Fire COM. It consists of a OMAP3530 processor from Texas Instruments (TI) based on ARM Cortex-A8 architecture with a working frequency of 600Mhz a DSP. Also, it features a 256MB RAM unit, a 256MB flash drive and Bluetooth and WiFi connectivity.

To provide the COM basic communication interfaces and expand its functionalities small expansion boards exist that are attached to the main COM board. The model chosen was the Tobi expansion board. It supports numerous interfaces such as 10/100baseT Ethernet, DVI-D (HDMI), USB OTG mini-AB, USB host standard A, Stereo audio in/out, USB Serial Console among others.

All these flexible features and processing capacities make the Gumstix Overo Fire a powerful and versatile COM very suitable for our project.

Caspa VL Camera To integrate imaging capacities with the Gumstix Overo Fire, a Caspa VL Camera module is connected to it. It consists of an ultra-compact camera which weights only 22.9 g. It can capture images with up to 720×480 pixels of resolution with a framerate of 60 fps. Due to its technical features and to the fact that it is an affordable and lightweight hardware, it was chosen to be used in this project.

The mechanical support that hold the Overo Fire together with the Caspa VL Camera is hand made. This module provides the protection and robustness needed to easily attach and detach the Gumstix to a vehicle.

This set will, from now on, be designated by *Gumstix-and-Camera module* (G-C module) and is shown in Figure 5.

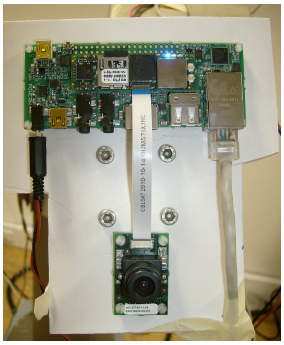


Figure 5: The Gumstix-and-Camera module

PUMA 560 Robotic Arm To test the developed algorithm with real data in a real context. the Gumstix-and-Camera module was attached to a 6-DOF PUMA (Programmable Universal Machine for Assembly) 560 robotic arm (Figure 6). With this platform, it is possible to test all the algorithms, to analyze their performance, improve them and correct any errors without jeopardizing the hardware.

When working with PUMA it was possible to understand that its operation conditions are not ideal anymore. Maybe due to the fact that it is a 15 years old robotic arm and no regular

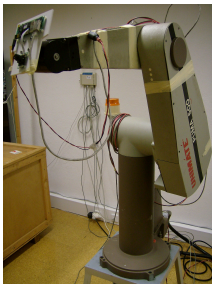


Figure 6: The Puma 500 Robotic Arm with the G-C module.

maintenance has been made. For example, when a single joint is ordered to move the all arm "shakes" a bit. Also, if the end effector is ordered to go to a specific position, and then joints readings are requested, the reported position has an error in the order of centimeters.

Overall Architecture The architecture of the test bench created is illustrated in Figure 7. As already stated, the controller computer is only used as interface to the robotic arm. Is the PUMA PC that does the calculations regarding the inverse kinematics (conversion from the desired arm pose in the Cartesian Space to the individual joints desired positions). This PC receives the desired positions from the main laptop. By its turn, the main laptop is connected to the G-C module. Two options exist, the main laptop can do all the calculations for estimating the camera pose and the G-C module is only used as a simple network camera that sends the captured images over the network, or, all the calculations are performed on the G-C modules and the main laptop only operates as an interface. Either way, after each frame is processed, the mail laptop send the desired next position to the PUMA PC that, by its turn, order the PUMA arm to move.

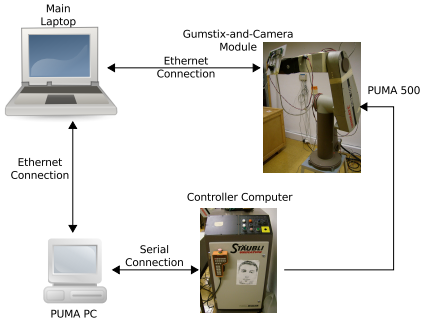


Figure 7: Gumstix Overo Fire attached to the Tobi expansion board

5 Experimental Results

Considering computation times, table 2 presents the time needed to process one frame of 752×480 pixels resolution either by using SURF or BRIEF on a Dual Core at 2.2GHz PC and on the Gumstix. SURF requires much more time than BRIEF. On PC real-time speed is achieved, but on Gumstix, even when choosing brief, only an average of 3-4 fps.

Regarding SURF, 85% of processing time is spent calculating the features descriptors, thus representing the bottleneck. Almost all the operations on this task are floating point, which are not very efficiently performed on the Gumstix CPU.

Table 2: Computation times

Algorithm	PC	Gumstix
SURF	256 ms	5 s
BRIEF	27 ms	330 ms

5.1 Simulation

Synthetic Dataset We want to simulate what a real camera sees if it travels along a certain trajectory. By defining the signals of rotation and translation over time, we can calculate the respective homography that describes the transformations suffered by the image seen by the camera.

To build the dataset a template image was chosen. Then, starting from this image, the homographies that correspond to the rotations and translations chosen are calculated using equation 7. To each of these homographies the following transformation is applied to take into account that the image is uncalibrated:

$$T = \mathsf{K}H\mathsf{K}^{-1}. \tag{13}$$

These consecutive perspective transformations are applied to the template image. And, this way, the movie frames are generated.

The purpose of these movies is to test the correctness and the robustness of the solution given by the algorithm developed. Which is possible because the true exact ground truth is available (actually is chosen) and can be used to calculate the error of the solution found by the algorithm developed.

The actual dataset is composed of 10 movies, 5 different trajectories each applied to two different template images. One of the template images is shown in Figure 8.

The the purpose of four different trajectories is to evaluate the behavior of the algorithm under various conditions:

- **Pure Rotation dataset:** The camera is rotated along its y axis.
- **Scale Change dataset:** The camera is moved along its z axis without rotating. This causes the template on the image to get consecutively small. To test the scale invariance of the descriptors.
- **Pure Rotation Z dataset:** The camera is rotated along its z axis. To test rotation invariance of the descriptors.
- **Perspective Distortion dataset:** Starting perpendicular to the object, the camera moves down in an arc resulting in strong perspective distortion

It worth to mention that because the axis-angle representation was chosen, to avoid the degenerate case, none of the trajectories contain a pose where there is no rotation. For example, in the Scale Change dataset there is a constante rotation along the z axis of 10 degrees.

Some examples of the movie frames are shown in Figure 8.

(a) Scene 1

Figure 8: Some examples of dataset frames.

Results Analysis In order to evaluate the precision of the algorithm, some performance indexes had the be defined. Considering the translation vector error, and taking into account that, by construction, the scale factor is always unknown, we can only evaluate the correctness of the direction. So the angle between the true and calculated translation vector was used as the error measurement. For the rotation, the axis-angle representation was chosen. The error between the true and the calculated rotation angle, as well as the angle between the true and the calculated rotation axis, are used as error measurements.

The three graphics for each of the four trajectories are represented in Figures 9 and 10. The results were pretty similar for both template images so, the errors represented are the average of both templates errors. Both SURF and BRIEF descriptors were tested. If the plots are not continuous and there are points missing it means that the algorithm failed to detect the template in the correspondent frame.

Regarding the Pure Rotation dataset on Figures 9(a), 9(c) and 9(e) it can be seen that the translation error is relatively small, between 1 and 7 degrees for both BRIEF and SURF. The rotation axis error is higher when the angle of rotation tends do zero. But, at the same time, because the rotation angle itself tends to zero along with the rotation angle error, this becomes negligible.

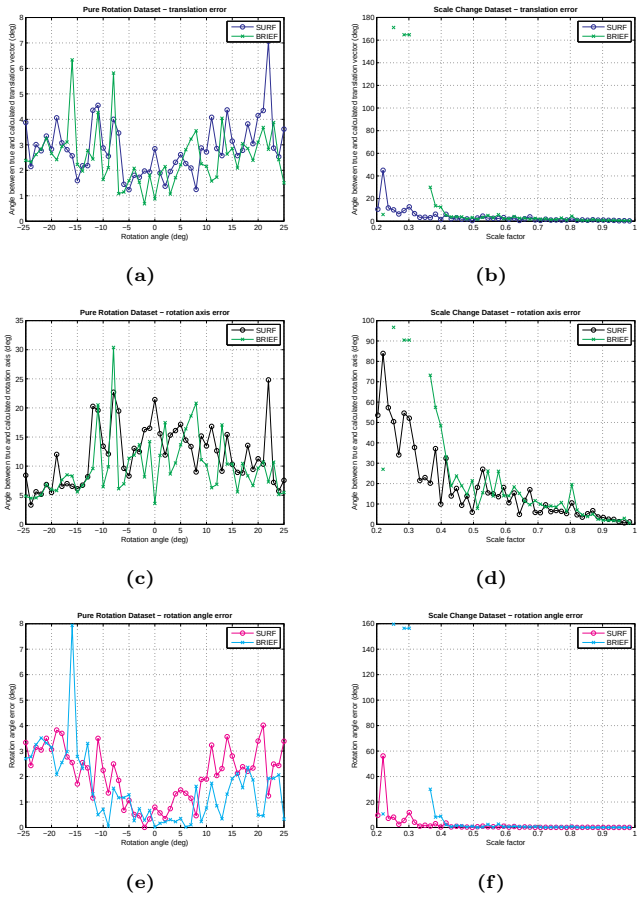
In the Scale Change data set (Figures 9(b), 9(d) and 9(f)), both BRIEF and SURF remain more or less precise until the template is reduced by a scale factor of 0.4. The translation error oscillates between 1 and 7 degrees and, once again, the rotation axis error is negligible. From a scale factor of 0.4 and beyond, BRIEF fails to detect the template and the SURF estimation becomes more unstable, although still reliable.

Considering the Pure Rotation Z dataset (Figures 10(a), 10(c) and 10(e)), BRIEF and SURF are both very invariant to rotations, with translation error between 0 and 4 degrees and rotation error almost always zero. Still, SURF performed slightly better around rotation angles of 90 degrees.

The Perspective Distortion dataset in Figures 10(b), 10(d) and 10(f) shows that with angles greater than 50 degrees the estimations became unreliable. For angles between 0 and 50 degrees, translation errors are no greater than 13 degrees, rotation axis errors are almost always lower than 15 degrees, just like the rotation angle error. SURF estimation are little more accurate.

5.2 Closed Loop Control with Real Videos tests

In order to fully demonstrate that the proposed solution allows the correct pose determination of a camera, a feedback control law was used to close the loop and drive the camera to the desired pose.



(a) **(b)** **(c)** **(d)** **(e)** **(f)**

Figure 9: Error of pose detection under several different conditions.

In this section, the control law used is described, some implementation considerations and modifications explained and the final results are presented.

Vision-Based control for rigid body stabilization and implementation on the PUMA 500 robotic arm The following is a summary of Cunha et al. [31] to describe the control law used. Here the notation is a little different from the rest of the paper in order to be in agreement with the original work.

Considering a fully-actuated rigid-body, with an attached coordinate frame $\{B\}$, and an inertial frame $\{F\}$ attached to the feature plane, let $(\boldsymbol{p}, \boldsymbol{R}) = ({}^{B}\boldsymbol{p}_{F}, {}^{B}_{F}\boldsymbol{R})$ represent, respectively, the translation and rotation of the $\{F\}$ with respect to the body frame. The equation that describes the motion of the body over time can be written as:

$$\dot{\boldsymbol{p}} = -\boldsymbol{v} - \mathsf{S}(\omega)\boldsymbol{p} \tag{14}$$

$$\dot{\boldsymbol{R}} = -\mathsf{S}(\omega)\boldsymbol{R} \tag{15}$$

where $\boldsymbol{v}$ and $\omega \in \mathbb{R}^3$ are the linear and angular velocities, respectively. If the rigid-body desired pose is represented by $(\boldsymbol{p}^*, \boldsymbol{R}^*)$ which are considered constant over time, we can introduce the following error variables

$$\boldsymbol{p}_e = \boldsymbol{p} - \boldsymbol{p}^*, \qquad \boldsymbol{R}_e = \boldsymbol{R}\boldsymbol{R}^{*T} \tag{16}$$

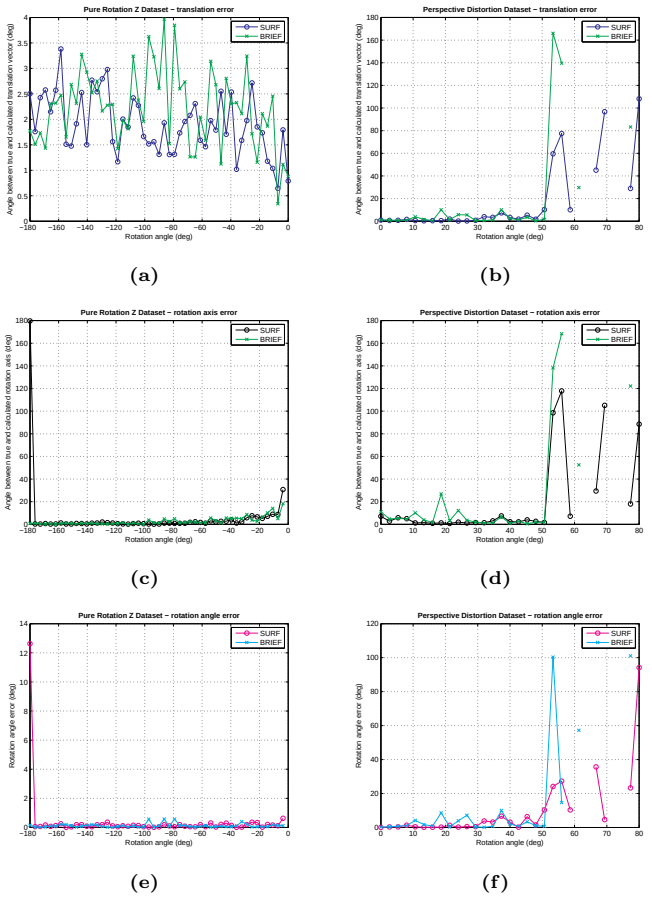


Figure 10: Error of pose detection under several different conditions.

and write the corresponding state equations

$$\dot{\boldsymbol{p}}_e = -\boldsymbol{v} - \mathsf{S}(\omega)(\boldsymbol{p}_e + \boldsymbol{p}^*) \tag{17}$$

$$\dot{\boldsymbol{R}}_e = \mathsf{S}(\omega)\boldsymbol{R}_e \tag{18}$$

Denoting the current and desired calibrated coordinates of the image features by $\boldsymbol{y}$ and $\boldsymbol{y}^*$, respectively, and by α the homography matrix scale factor (see Chapter 2), the control law developed can be written as

$$\boldsymbol{v} = \begin{cases} k_1(\alpha\boldsymbol{y} - \boldsymbol{y}^*) & \text{until } ||\alpha\boldsymbol{y} - \boldsymbol{y}^*|| < \gamma \\ k_2(\alpha\boldsymbol{y} - \boldsymbol{y}^*) - \hat{z}^*\mathsf{S}(\omega)\alpha\boldsymbol{y} & \text{afterwards} \end{cases} \tag{19}$$

$$\omega = \begin{cases} k_3\mathsf{S}(\boldsymbol{y}^*)\alpha\boldsymbol{y} & \text{until } ||\alpha\boldsymbol{y} - \boldsymbol{y}^*|| < \gamma \\ k_4\mathsf{S}^{-1}(\boldsymbol{R}_e\boldsymbol{M} - \boldsymbol{M}\boldsymbol{R}_e^T) & \text{afterwards} \end{cases} \tag{20}$$

with the update law for the estimate of the desired depth $\hat{z}^*$ given by

$$\dot{\hat{z}}^* = \begin{cases} 0 & \text{until } ||\alpha\boldsymbol{y} - \boldsymbol{y}^*|| < \gamma \\ k_z\boldsymbol{y}^*\mathsf{S}(\omega)\alpha\boldsymbol{y} & \text{afterwards} \end{cases} \tag{21}$$

where γ, k_1, k_2, k_3, k_4 and k_z are positive scalars. $\hat{z}^*$, is a estimator for the desired depth. This control law relies on an adaptive scheme and can be divided in two sequential objectives. First, it will drive the translation vector $\boldsymbol{p}$ to an arbitrarily small neighborhood of $\boldsymbol{p}^*$. Then, it will ensure the convergence of $(\boldsymbol{p}, \boldsymbol{R})$ to $(\boldsymbol{p}^*, \boldsymbol{R}^*)$ using a controller that enforces feature visibility by guaranteeing that the camera not only points towards the features, but also remains in front of them.

As stated on the original work, the described control law guarantees that the desired equilibrium point is an almost global attractor.

For the implementation on the PUMA robotic arm, a few considerations had to be made due to some PUMA limitations. The control law had to be discretized because the arm can only receive inputs in position and not in velocity.

Denoting the inertial arm frame by $\{A\}$, the position and orientation of the body frame $\{B\}$, with respect to $\{A\}$, can be written as

$${}^A\boldsymbol{p}_B = {}^A\boldsymbol{p}_F - {}^A_F\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{p}, \tag{22}$$

$${}^A_B\boldsymbol{R} = {}^A_F\boldsymbol{R}\boldsymbol{R}^T \tag{23}$$

respectively, where $({}^A\boldsymbol{p}_F, {}^A_B\boldsymbol{R})$ denotes the pose of $\{F\}$ expressed in $\{A\}$. Using (14) and (15), the kinematics for ${}^A\boldsymbol{p}_B$ and ${}^A_B\boldsymbol{R}$ can be written as

$${}^A\dot{\boldsymbol{p}}_B = {}^A_B\boldsymbol{R}\boldsymbol{v} \tag{24}$$

$${}^A_B\dot{\boldsymbol{R}} = {}^A_B\boldsymbol{R}\mathsf{S}(\omega) \tag{25}$$

Using the Euler method, the solution of (24) can be approximated by

$${}^A\boldsymbol{p}_B(k+1) = {}^A\boldsymbol{p}_B(k) - \Delta t({}^A_B\boldsymbol{R}(k)\boldsymbol{v}(k)), \tag{26}$$

where Δt is the sampling time. The zero-order hold discretization of (25) is given by

$${}^A_B\boldsymbol{R}(k+1) = {}^A_B\boldsymbol{R}(k)\ \mathsf{rot}(\Delta t\omega(k)) \tag{27}$$

Also, the true current pose of the camera had to be estimated by reading the PUMA joints encoders to know the current transformation from the arm frame to the camera frame. This was due to the fact that PUMA accepts position inputs expressed in the arm frame $\{A\}$ instead of in the feature frame $\{F\}$.

Results Analysis The results are presented next. Figure 11 shows the template frame used, the image seen at the start pose, and the image seen at the final pose to where the PUMA end effector converged. As it can be seen, the image captured at the final pose is very similar to the reference template frame.

(a) Template frame **(b)** Image at start pose **(c)** Achieved image at final pose

Figure 11: Template frame and some examples of dataset frames.

The evaluation over time of the error measure $||\alpha\boldsymbol{y} - \boldsymbol{y}^*||$ used on the control law is shown in Figure 12. Overall, the error tends

to zero but, there are moments where its norm increases. This is caused by the way that the controller outputs are sent to the PUMA arm. As already stated, the controller is discretized. At each sample time, the controller outputs are calculated considering the sample rate and the PUMA actuators are activated to move the arm to next position according to the controller outputs. The trajectory that the arm travels between two sample positions is not externally controlled and depends on the internal joints controllers. So, the temporary increases on the error norm were predictable. If the arm could accept commands in linear and angular velocities, the control law could be used on its original version and it is expectable that the error norm curve thus becomes monotonically decreasing. It is also important to note that the final error is not zero. At the final pose, controller outputs were still being calculated and sent to PUMA, but the end-effector was not moving. This is caused by the low precision commands accepted by the PUMA arm. Nevertheless, according to the actuators readings, the translation error norm between the desired and the achieved pose is only 1.9 cm.

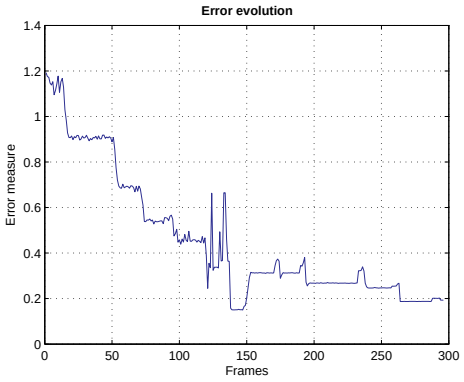


Figure 12: Error norm evolution over time

6 Conclusions

A solution to the problem of taking a rigid-body into a desired pose was developed. This solution only requires a image of planar scene acquired at the desired pose and a video stream captured by a camera on board of the vehicle containing the planar scene inside the FOV. Real-time performance was achieved when using a regular PC. Using the lightweight Gumstix-and-Camera hardware module an average of 3-4 fps are possible.

To achieve the described solution, Concepts from computer vision such as the Camera Pinhole Model, image transformations like planar homographies and methods for the extraction of 3-D information from the planar homograhy matrix were learned. To solve the problem of finding correspondence between image features, several algorithms were studied.

These algorithm are designed as image features extraction and description algorithms and mainly two were studied and implemented: SURF and BRIEF. From the tests performed, it was possible to conclude that both can be used to detect the reference planar scene on a video stream. Nevertheless, both have its own advantages and disadvantages. In general, SURF is more robust in extreme situations, like strong image scale changes or perspective distortion. Besides that, there are conditions where BRIEF totally fails detecting the scene and SURF estimation is not robust. But, at least SURF provides a estimate, which, if a control feedback law is used, can be sufficient to take the rigid-body into a pose where robustness is recovered. On the other hand, BRIEF wins with the incredible speed at which the required computations are performed. Plus, the fact that BRIEF fails sooner than SURF may not be a problem, since if the feedback control law is working properly the rigid-body pose error will probably not be very big and BRIEF will always succeed on detecting the reference scene. That is why, when considering applications requiring real time performance, BRIEF is chosen over SURF.

A test bench for testing the developed algorithm was built, consisting of synthetically generated video and true video captured from a PUMA robotic arm with the G-C module attached to its end effector. After long years of inactivity, the PUMA robotic arm was reanimated and its existence given a purpose. This allowed to verify the correctness of the algorithm, its performance and robustness. With this test bench it was possible to confirm that, when using an appropriate control law, the pose error tends to zero, thus demonstrating the correctness of the algorithm.

7 Future Work

The frame rate of the algorithm running on the G-C module can be improved if the integrated DSP is used to compute the Hamming distance, required for matching the BRIEF descriptors. Also, if Moore's law remains valid, new and more powerful Gumstix's like COMs will be available.

Further testing the algorithm with a real vehicle, like the quadrotor available at the ISR labs, would be interesting by not discretizing the control law and using its original version.

Regarding, the features detection. A different technique was studied, where circular TAG's are used instead of the regular features extractors and descriptors algorithms. This has the drawback of requiring the scene to contain a TAG but, under the scope of the AIRTICI project this may not be an issue. On the other hand, this has the advantage of not being as computationally intensive as SURF or BRIEF algorithms. The TAG detection algorithm originally developed by Reverse Engineering company personell was studied and adapted to calculate homographies between images. Unfortunately, due to time restrictions, it was not integrated on the main developed algorithm.

References

[1] E. Altüg, J. Ostrowski, and R. Mahony, "Control of a quadrotor helicopter using visual feedback," *Robotics and Automation*, 2002.

[2] E. Altüg, J. Ostrowski, and C. Taylor, "Quadrotor control using dual camera visual feedback," *Robotics and Automation*, 2003.

[3] M. Earl and R. D'Andrea, "Real-time attitude estimation techniques applied to a four rotor helicopter," *Robotics and Automation*, 2003.

[4] H. Romero, R. Benosman, and R. Lozano, "Stabilization and location of a four rotor helicopter applying vision," *American Control Conference*, 2006.

[5] I. F. Mondragón, P. Campoy, C. Martínez, and M. A. Olivares-Mendez, "3d pose estimation based on planar object tracking for uavs control," *IEEE International Conference on Robotics and Automation*, 2010.

[6] C. Martínez, I. F. Mondragón, M. A. Olivares-Méndez, and P. Campoy, "On-board and ground visual pose estimation techniques for uav control," *Intelligence Robotics Systems*, 2011.

[7] O. Bourquardez, R. Mahony, N. Guenard, F. Chaumette, T. Hamel, and L. Eck, "Image-based visual servo control of the translation kinematics of a quadrotor aerial vehicle," *IEEE Transactions on Robotics*, 2009.

[8] R. Hartley and A. Zisserman, *Multiple View Geometry in computer vision*. Cambridge University Press, 2003.

[9] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2010.

[10] Y. Ma, S. Soatto, J. Kosecká, and S. Sastry, *An Invitation to 3-D Vision*. Springer, 2004.

[11] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *American Control Conference*, 1981.

[12] O. Faugeras and F. Lustman, "Motion and structure from motion in a piecewise planar environment," *International Journal of Pattern Recognition and Artificial Intelligence*, 1988.

[13] C. Harris and M. Stephens, "A combined corner and edge detector," *Proceedings of the 4th ALVEY vision conference*, p. 147 151, 1988.

[14] P. R. Beaudet, "Rotationally invariant image operators," *International joint conference on pattern recognition*, 1978.

[15] L. Kitchen and A. Rosenfeld, "Gray-level corner detection," *Pattern Recognition Letters*, pp. 95 – 102, 1982.

[16] T. Lindeberg, "Scale-space theory: A basic tool for analysing structures at different scales," *Journal of Applied Statistics*, pp. 224 – 270, 1994.

[17] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, pp. 91 – 110, 2004.

[18] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features surf," *Computer Vision and Image Understanding*, 2008.

[19] C. Schmid and R. Mohr, "Local greyvalue invariants for image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1997.

[20] W. T. Freeman and E. H. Adelson, "The design and use of steerable filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1991.

[21] F. C. Crow, "Summed-area tables for texture mapping," *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, p. 207 212, 1984.

[22] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," *ECCV'10*, 2010.

[23] M. Özuysal, P. Fua, and V. Lepetit, "Fast keypoint recognition in ten lines of code," *IEEE conference on computer vision and pattern recognition*, 2007.

[24] S. Taylor, E. Rosten, and T. Drummond, "Robust feature mathing in 2.3us," *IEEE conference on computer vision and pattern recognition*, 2009.

[25] M. Brown and D. lowe, "Invariant features from interest point groups," *BMVC*, 2002.

[26] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," *International Conference on Computer Vision Theory and Applications (VISAPP'09)*, 2009.

[27] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *European Conference on Computer Vision*, 2006.

[28] S. M. Smith and J. M. Brady, "Susan - a new approach to low level image processing," *International Journal of Computer Vision*, 1997.

[29] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, 1986.

[30] Opencv: Open source computer vision. [Online]. Available: http://http://opencv.willowgarage.com/

[31] R. Cunha, C. Silvestre, J. Hespanha, and A. P. Aguiar, "Vision-based control for rigid body stabilization," *Automatica*, 2011.