PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

TECHNISCHE UNIVERSITÄT ILMENAU

Technische Universität Ilmenau

Department of Computer Science and Automation

Master Thesis

Studies on obstacle detection and path planning for a quadrotor system

To achieve the degree:

Master of Science (M.Sc.)

in Technische Kybernetik und Systemtheorie

Submitted by:	Dalthon Abel Valencia Mamani
Date and Place of Birth:	01.08.1989 Juliaca (Perú)
Matrikel-Nr:	57865
Cod. PUCP:	20156459

Department (TU Ilmenau):	Technische Kybernetik und Systemtheorie
Advisor (TU Ilmenau):	Dr. Kai Wulff
Advisor (TU Ilmenau):	
Responsible Professor (TU Ilmenau):	Univ.-Prof. Dr. Johann Reger
Responsible Professor (PUCP):	MSc.-Ing. Francisco Fabian Cuéllar Córdova
Date and Place:	October 2, 2017, Ilmenau

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

ESCUELA DE POSGRADO

Studies on obstacle detection and path planning for a quadrotor system

TESIS PARA OPTAR EL GRADO ACADÉMICO DE MAGISTER EN INGENIERÍA DE CONTROL Y AUTOMATIZACIÓN

AUTOR

Dalthon Abel Valencia Mamani

ASESOR

Supervisores (PUCP): MSc.-Ing. Francisco Fabian Cuéllar Córdova

Supervisor (TU Ilmenau): Dr. Kai Wulff

Septiembre, 2017

Kurzfassung

Autonome Systeme sind ein interessantes Thema vor kurzem untersucht; für Land- und Luftfahrzeuge; Allerdings ist die Hauptbegrenzung von Luftfahrzeugen das Gewicht, um an Bord zu tragen, da die verbrauchte Energie davon abhängt und Hardware wie Sensoren und Prozessor begrenzt ist. Die vorliegende Arbeit entwickelt eine Anwendung der digitalen Bildverarbeitung zur Erkennung von Hindernissen, die nur eine Monokamera verwenden, es gibt einige Ansätze, aber der vorliegende Bericht will sich auf den Abstandsschätzungsansatz konzentrieren, der in Zukunft mit anderen Methoden kombiniert werden kann, da dieser Ansatz ist allgemeiner.

Der Abstandsschätzungsansatz verwendet Merkmalserkennungsalgorithmen in zwei aufeinanderfolgenden Bildern, passt sie an und schätzt somit die Hindernisposition ab. Die Schätzung wird durch ein mathematisches Modell der Kamera und Projektionen zwischen diesen beiden Bildern berechnet. Es gibt viele Parameter, um die endgültigen Ergebnisse zu verbessern, und die besten Parameter werden mit aufeinanderfolgenden Bildern gefunden und getestet, die alle 0,5 m auf einem geraden Weg von 5 m erfasst wurden. Fraunhofer-Positionsmodule werden mit dem gesamten Algorithmus getestet. Schließlich wird, um den neuen Weg ohne Hindernisse zu etablieren, ein optimales Binär-Integer-Programmierproblem vorgeschlagen, das den Ansatz unter Verwendung von Ergebnissen, die aus der Abstandsschätzung und der Hinderniserkennung erhalten wurden, anpasst. Die daraus resultierenden Daten eignen sich zur Kombination mit Informationen aus konventionellen Sensoren wie Ultraschallsensoren. Der erhaltene mittlere Fehler liegt zwischen 1 % und 12 % in kurzen Abständen (weniger als 2,5 m) und größer mit längeren Abständen.

Die Komplexität dieser Studie liegt in der Verwendung einer einzigen Kamera für die Erfassung von Frontalbildern und dem Erhalten von 3D-Informationen der Umgebung, wird die Berechnung des Hinderniserfassungsalgorithmus off-line getestet und der Wegplanungsalgorithmus wird mit erkannten Keypoints vorgeschlagen im Hintergrund.

Abstract

Autonomous systems are one interesting topic recently investigated; for land and aerial vehicles; however, the main limitation of aerial vehicles is the weight to carry on-board, since the power consumed depends on this and hardware like sensors and processor is limited. The present thesis develops an application of digital image processing to detect obstacles using only a monocamera, there are some approaches but the present report wants to focus on the distance estimation approach that, in future works, can be combined with other methods since this approach is more general.

The distance estimation approach uses feature detection algorithms in two consecutive images, matching them and thus estimate the obstacle position. The estimation is computed through a mathematical model of the camera and projections between those two images. There are many parameters to improve final results and the best parameters are found and tested with consecutive images, which were captured every 0.5m along a straight path of 5m. Fraunhofer position modules are tested with the entire algorithm. Finally, in order to establish the new path without obstacles, an optimal binary integer programming problem is proposed, adapting the approach using results obtained from the distance estimation and obstacle detection. Resulting data is suitable for combining them with information obtained from conventional sensors, such as ultrasonic sensors. The obtained mean error is between 1% and 12% in short distances (less than 2.5 m) and greater with longer distances.

The complexity of this study lies in the use of a single camera for the capture of frontal images and obtaining 3D information of the environment, the computation of the obstacle detection algorithm is tested off-line and the path-planning algorithm is proposed with detected keypoints in the background.

Acknowledgment

I thank my parents, who have always been there to support me in everything; my brothers and sisters, who are the reasons because I continue working; my advisers, who have been watching my progress; my friends, who are one motivation and show me the way to overcome me.

I also want to thank the grant of FONDECYT-CONCYTEC, through the agreement 2015-034 FONDECYT, in the context of which the present thesis was developed S̈tudies on obstacle detection and path planning for a quadrotor system änd the Universidad Catolica del Perú that gave me the opportunity to be part of the double degree program with the Technical University of Ilmenau.

Finally, and not less important, I thank God for everything he has given to me until now and for all that is to come.

Ilmenau, den 02. 10. 2017 Dalthon Abel Valencia Mamani

Abbreviations and Symbols

SISO	Single Input, Single Output system
MIMO	Multiple Inputs, Multiple Outputs system
SIFT	Scale-Invariant Feature Transform
SURF	Speeded-Up Robust Features
BRISK	Binary Robust Invariant Scalable Keypoints
UAV	Unmanned Aerial Vehicles
CAS	Collision Avoidance System
SLAM	Simultaneous Localization and Mapping
PTAM	Parallel Tracking and Mapping
KLT	Kanade-Lucas-Tomasi fature
EKF	Extended Kalman Filter
KF	Kalman Filter
RANSAC	Random Sample Consensus
MOPS	Multiscale Oriented Patches
WF	Weighted Filter
FOV	Field of View
CCD	Charge Coupled Device
FPS	Frames per Seconds
LoG	Laplacian of Gaussian
DoG	Difference of Gaussian
SSD	Sum of Squared Differences
GMRES	Generalized Minimal Residual
RPY	Roll-Pitch-Yaw
BF	Brute Force
FLANN	Fast Library for Approximate Nearest Neighbors
EDS	Ego Dynamic Space Transformation

Contents

List of Figures

List of Tables

1 Introduction

Computer power processing, embedded systems development, sensor technology, actuators technology are increasing every day and, in recent years, the development of autonomous systems has been possible. As a clear example, we have the humanoids or four-legged creatures (Fig. 1.1), cars equipped with sensors that give them the ability to park without the driver intervention; these examples allow us to appreciate the ability to control complex systems that we have today. In addition to all this, you can make the system adaptable to certain surface changes, check the position, and have a certain degree of energy independence since they are powered by a battery.

Figure 1.1: Robots developed by Boston Dynamics (now owned by Google) [Dynamics, 2017]

All these advances are focused on making the control system capable of achieving autonomy, i.e. having the ability to work without the intervention of an operator, at least with respect to the displacement. A new feature is needed for the development of this new capability in systems; this new feature is the ability of the system to recognize its environment and thus to be able to deal with obstacles between its current position and its target.

Detection and obstacle avoidance is a very popular research topic in recent years, especially in cars where prototypes have already been implemented (Fig. 1.4). However, Unmanned Aerial Vehicles (UAV) have also been the subject of investigation; the quadcopter, for example, due mainly to the wide range of possible applications and the relative low cost that implies its implementation, has been becoming popular in research and Collision Avoidance System (CAS) for UAV was born as one new topic.

Figure 1.2: A self-driving car by Google is displayed at the Viva Technology event in Paris, France, June 30, 2016

The main work of CAS is to guarantee that the UAV could reach its target without collision occurrence. Obstacles can move or not but the CAS system must solve the following problems [Pham et al., 2015]:

- How to sense the environment and extract useful information (i.e. obstacle position, speed, angle, etc.) and after that how to decide if the obstacle is enough close to collide with the quadrocopter.
- How to know when the obstacle avoidance phase must be enabled and how to realize a manoeuvre.

Every CAS must be divided at least into these two main components [Pham et al., 2015]: Obstacle Detection (environment sensing) and obstacle avoidance (manoeuvre

realization). Each research area emphasises on different parts of the CAS and this is the reason why there are many approaches to implement a CAS Fig. 1.3.

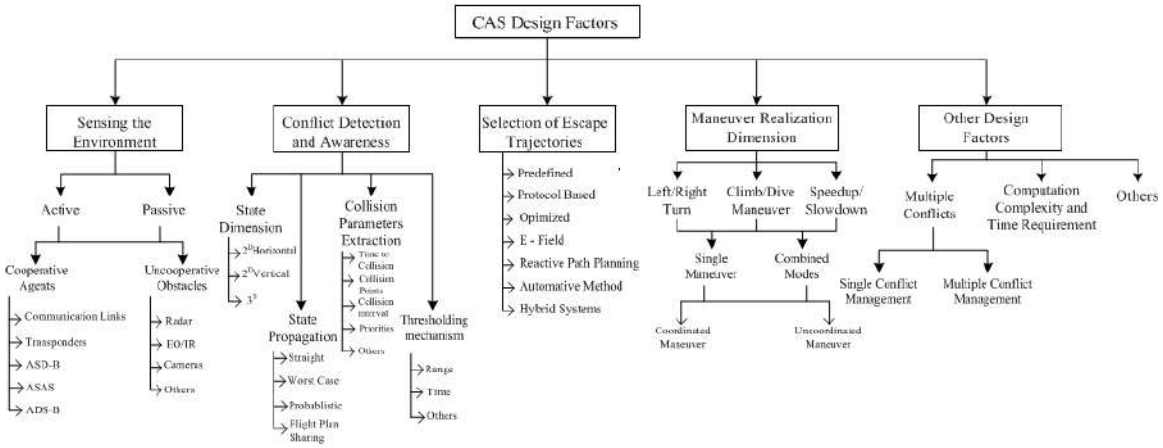


Figure 1.3: Main CAS design factors divisions with its subdivisions [Albaker and Rahim, 2009].

One of the first attempts for autonomous quadcopter was designed in 2007 [Roberts et al., 2007], [Bouabdallah et al., 2007], on the first research the main goal was to design a quadrocopter capable of autonomous indoor operation, from taking-off to landing only with ultrasonic and infrared sensors, without using an external positioning system. In the second research the goal is almost the same, but the research is also focused on making the quadrocopter autonomous for outdoor applications, but using only ultrasonic sensors. In both researches there is no a camera mounted on the quadrocopter because of the low quality in cameras and slow processors at that time. The Simultaneous Localization and Mapping (SLAM) technique is commonly used in robotics, SLAM is an estimation technique, therefore quadcopter localization and environment model are approximations; SLAM with monocular camera has been developed in different methods [Çelik and Somani, 2013], [Sa et al., 2013], [Magree et al., 2013], [Huang et al., 2015] and it is an open problem nowadays because the innovation of technology in sensors and special cameras. All these researches build a mapping from the environment using different methods like corners and line patterns recognition, Parallel Tracking and Mapping (PTAM) algorithm and design of a novel Extended Kalman Filter (EKF) to fuse measurements. The present work pretend extract 3D information from the environment but this information is not processed for SLAM, this task is proposed as a possible future work.

[Çelik and Somani, 2013] presents an indoor navigation integrated with SLAM combined with a suitable feature extraction is possible to obtain absolute distances and

using the infinity point method the pitch and yaw angles can be estimated.
[Sa et al., 2013] uses PTAM especially applied for SLAM; PTAM is a camera tracking method for augmented reality; this method helps to estimate the quadcopter position as well as some parameter like yaw angle and a scale is necessary to convert the PTAM based poses into metric form.
[Magree et al., 2013] uses SLAM algorithm with monocular camera applied for Obstacle Avoidance, the algorithm uses inverse depth parametrization of the feature points for fast depth convergence, and stores convergence points in an altitude map which finally is used for obstacle avoidance.
[Huang et al., 2015] proposes a SLAM system with an improved Kanade-Lucas-Tomasi fature (KLT) tracker, which is an optical flow method, and an EKF is designed for sensor fusion. The quadrotor counts with a monocular camera.
In general, SLAM is a weight technique in the sense of computing, we need to construct a 3D model using sensor information and process images with complex algorithms. For obstacle avoidance, a critical parameter is the processing time because an obstacle avoidance system needs to run in real time, the image processing and mapping building are two of the main problems.
Nevertheless, since 2010 the Kinect RGB sensor is available [Litomisky, 2012], in the beginning it existed some limitation due to field of view, max distance from sensor, quantization, huge amount of data, but several researches has been done in order to deal these limitations [Henry et al., 2010, Besl et al., 1992]. Henry perform a pairwise rough initial aligment using SIFT visual features [Lowe, 1999] and a Random Sample Consensus (RANSAC) algorithm, which is then refined using the depth-based Iterative Closest Point algorithm [Besl et al., 1992].
Combining different algorithms in order to obtain better results is another way to improve an obstacle avoidance system. Jeong-Oog [Lee et al., 2011] proposed a combination of Multiscale Oriented Patches (MOPS) and Scale-Invariant Feature Transform (SIFT), the first mainly helps to detect feature points including orientation (rotation of the feature point) and the second one is able to detect feature points even if the image is scale or is captured from a different angle (size and orientation changes). With this information, the obstacle avoidance system obtains three-dimensional information from obstacles.
An obstacle avoidance system could be implemented by using simple sensors like ultrasonic sensor and infrared sensor [Gageik et al., 2015, Bouabdallah et al., 2007]. For the first research the principal problem is improve the reliable for the whole system, because of the ultrasonic and infrared sensors are not so accurate. Therefore, the data

fusion proposed must fix the reliable problem and allow a better control. For the second research the main idea is develop an obstacle avoidance control.
Some researchers have been studied the obstacle avoidance using the size expansion cue [Mori and Scherer, 2013, Al-Kaff et al., 2016]. The main contribution in these researches was developing an algorithm which returns the apparent size of frontal obstacles applying different approaches and feature matching (SURF and SIFT algorithm respectively) and only with one camera on-board.
Multiple view geometry is commonly use with stereo cameras [Fu et al., 2015], and mixing it with other techniques is possible to estimate a 3D reconstruction with only one camera [Ranft et al., 2013]. In this research, the mono camera on the quadrotor is moved up during the flight in order to obtain two images, apply multiple view geometry, estimate the pose and rotation with visual odometry and finally build a 3D reconstruction of the environment which could be sparse or dense.
Computing the distance between the obstacle and the quadrocopter is another way to detect obstacles. Geometrically, it is possible to build a mathematical model which relate the world axis coordinates, quadrocopter position, camera position and obstacle position, only with rotation information, camera specification, two images from a monocamera and knowing the position of the quadrocopter when these images are got [Saha et al., 2014]. The main problem with this approach could be the high accurate on position estimation we need in order to apply the model, five USB modules are developed for Fraunhofer Institut to fix this problem and they are going to measure the camera position.
Implement a full obstacle avoidance system using low cost sensor and a monocamera is also possible [Gageik et al., 2015]. This research used infrared, ultrasonic and a camera as sensors and an improved method to fusion the data obtained from each sensor called Weighted Filter (WF), which has a low computational burden compared to a Kalman Filter (KF), designing and implementation are also simpler.

1.1 Problem Statement

Application of quadrocopter has taken force to the point of becoming very popular commercially and extend its use in different fields such as cinema, television, security, sports, recreation, etc. In addition, there are applications which innovate in fields such as agriculture (Fig. 1.4) and mining [Patel et al., 2013], [Bemis et al., 2014] and are focused in topics like surveillance and mapping.
There are special sensors for mapping and sophisticated collision avoidance systems for

indoor environments but, due to the energy and process autonomy there are certain limitations, some of them are the weight of all its components that should not be excessive to ensure a longer battery life, the great computational power which will be required with a lot of data to process (maybe a ground station will be necessary) and finally the total cost will increase.

The application in which this work will be focused is the "navigation" that particularly requires the detection of possible obstacles and the planning of trajectory. It is proposed to use as main element a single camera. Obstacle avoidance using computer vision is still a challenge due to the large number of algorithms for image processing, variation of processing time according to the chosen algorithm, physical limitations (the total weight of the quadrocopter is limited so it cannot be mounting very sophisticated sensors) and processing of existing embedded systems (the processing of images involves great use of resources). Furthermore, the whole process itself must be executed in real time to ensure proper operation.

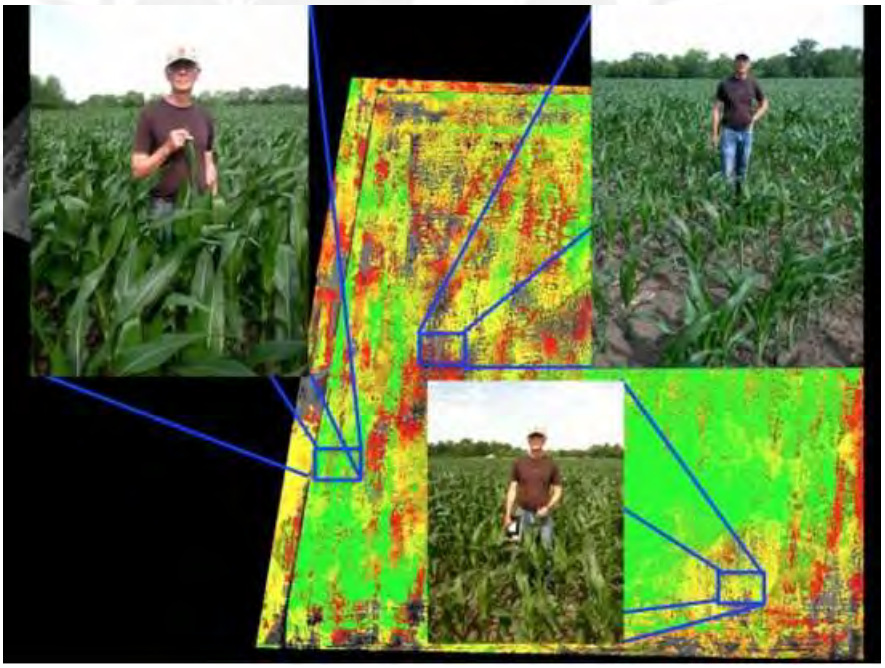


Figure 1.4: DVI image of approximaly 40 acres of corn. This field looks very homogeneous from ground, but aerial imagery indicates dramatic heterogeneity. In the green areas f the field, the corn is nearly chest high, while the corn is only ankle high or knee high in the red or yellow areas, respectively. (Agribotix Agricultural Intelligence)

There are many research directions but the present work is focused on obstacle detection, developing and setting a distance estimation algorithm which delivers as result

3D information from the environment, test and analyse the effect of each parameter in the mathematical model and then set the best conditions to test the entire algorithm in a path of 5 m. Finally, a path-planning algorithm is proposed according to output results of the previous step.

2 Theory and Background

2.1 Camera features

Choosing a camera according to the application is one important step in computer vision, so it is important to recognize some camera features [Howse et al., 2015].

- **Resolution** is the degree of detail that the lens and camera can capture. Details are important in most computer vision applications, so if the system has a poor resolution camera it is going to be difficult to implement computer vision application. The **line pairs per millimetre** (lp/mm) is an empirical measurement, which show all characteristics of the lens, sensor and setup. This parameter varies with the distance between the camera and the target, lens's settings (focal length) and light conditions.

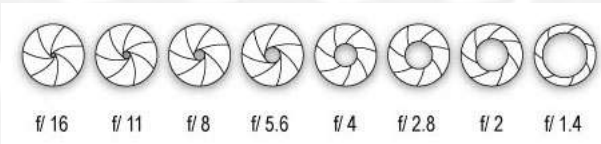


Figure 2.1: Camera apertures degrees [Nikon, 2017].

- **Field of view Field of View (FOV)** is how much of the environment the camera can see. FOV is measured as an angle, but can be expressed as the distance between two points in the edge of the observable field and a given depth from the lens. For example, a FOV of 90 degrees may also be expressed as a FOV of 2m at a depth of 1m. While more far of the centre of the FOV there is more distortion and less resolution, so the FOV should be wide enough to leave a margin around the target.

- **Camera's throughput** is the rate at which it captures image data. This parameter is like a sample rate of the environment. Throughput is measured in Frames per Seconds (FPS) but it is limited by:

– **Shutter speed** (exposure time): The shutter speed is mainly limited by lightning conditions. The lens's aperture (the opening of a lens's diaphragm through which light passes Fig. 2.1) and the camera's ISO speed (how sensitive the camera sensor is to the light that reaches it).

– **The type of shutter**: In a **global shutter** camera every pixel is captured simultaneously. In a **rolling shutter** the rows are captured sequentially and read-out from top to bottom 2.2.

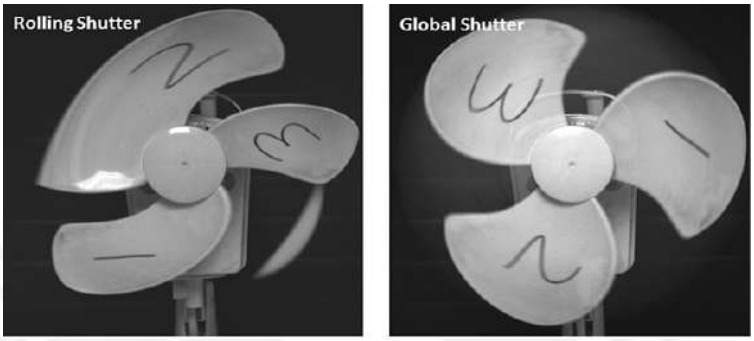


Figure 2.2: Global shutter and Rolling shutter [Andor, 2017].

2.2 Camera model

It has long known that a simple pin-hole is able to create a perfect inverted image on the wall of a darkness room. Even some marine molluscs still have this kind of eyes. On the other hand vertebrate animals have a lens that helps with the image formation in the retina. A digital camera works with a similar principle, there are a plastic lens and a semiconductor chip with light sensitive devices instead of the retina to convert light to a digital image.

A camera model is a mapping, which involves a projection of the 3-dimensional world onto a 2-dimensional surface. The most specialized and simplest camera model, based on central projection and finite centre, is the pin-hole camera. The model was developed in "Multiple View Geometry in Computer Vision" [Hartley and Zisserman, 2003] and can be also applicable to other cameras for example X-ray images, scanned photographic negatives, etc.

In computer vision is common to use the central perspective imaging model shown in Fig. 2.3 . Let the centre of projection be the origin of the Euclidean coordinate system, and consider the plane $z = f$, which is called the image plane or focal plane.

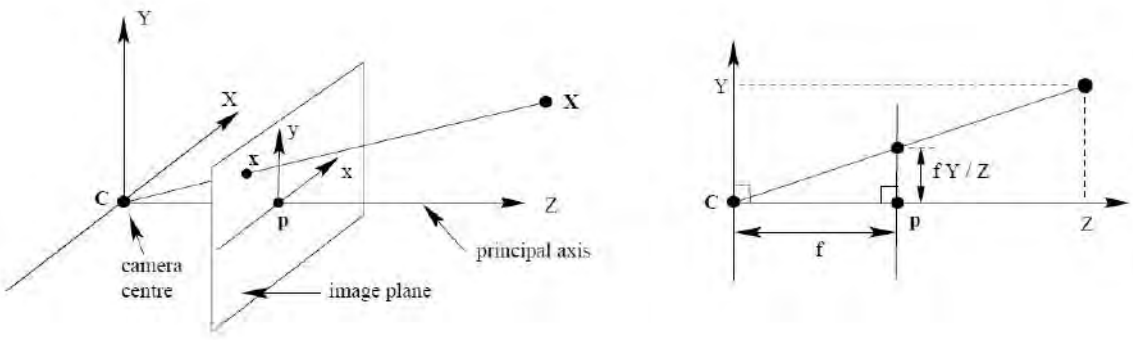


Figure 2.3: The central-projection model (Pin-hole camera model). The image plane is f in front of the camera's origin and on which a non-inverted image is formed [Hartley and Zisserman, 2003].

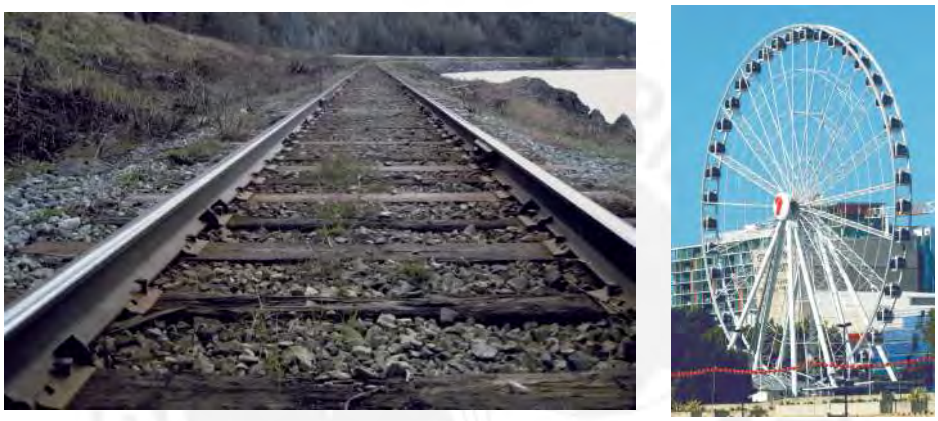
Under the pin-hole camera model, a point in space with coordinates $\mathbf{X}=(X,Y,Z)^T$ is mapped to the point in the image plane where a line joining the point $\mathbf{X}$ to the centre of projection meets the image plane.
The centre of projection is called the *camera centre*. It is also known as the *optical centre*. The line from the camera centre perpendicular to the image plane is called the *principal axis* or *principal ray* of the camera, and the point where the principal axis meets the image plane is called the *principal point*. The plane through the camera centre parallel to the image plane is called the *principal plane* of the camera.
As the Fig. 2.3 shows, by simple relationship between triangles is possible to compute the mapped point (x,y) in the image plane p by

$$x = f\frac{X}{Z}, \quad y = f\frac{Y}{Z} \tag{2.1}$$

which is a perspective projection, from the world to the image plane and has the following characteristics [Corke, 2011]:

- It performs a mapping from 3-dimensional space to the 2-dimensional image plane: $\mathbb{R}^3 \rightarrow \mathbb{R}^2$.
- Straight lines in the world are projected to straight lines on the image plane.
- Parallel lines on the world are projected to lines that intersect at a vanishing point as shown in Fig. 2.4a. In drawing, this effect is known as foreshortening. The exception are lines in the plane parallel to the image plane which do not converge.

- Conics in the world are projected to conics on the image plane. For example, a circle is projected as a circle or an ellipse as shown in Fig. 2.4b.
- The mapping is not one-to-one and a unique inverse does not exist. That is, given $(x,\ y)$ we cannot uniquely determine $(X,\ Y,\ Z)$.
- The transformation is not conformal – it does not preserve shape since internal angles are not preserved. Translation, rotation and scaling are examples of conformal transformations.



(a) Parallel lines (b) Circles

Figure 2.4: The effect of perspective transformation. [Corke, 2011]

The coordinates $(x,\ y)$ can be write in homogeneous form as

$$\begin{pmatrix} f\frac{X}{Z} \\ f\frac{Y}{Z} \\ Z \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.2}$$

Observe that $\mathbf{X}$ is the notation for the world point represented by the homogeneous 4-vector $(X, Y, Z, 1)^T$, $\mathbf{x}$ is the image point represented by the homogeneous 3-vector $(fX/Z, fY/Z, \mathbf{f})$, and the $3x4$ matrix Q is the homogeneous *camera projection matrix*. Then equation (2.2) could be written as

$$\mathbf{x} = Q\mathbf{X}$$

which defines the camera matrix for the pin-hole model of central projection.

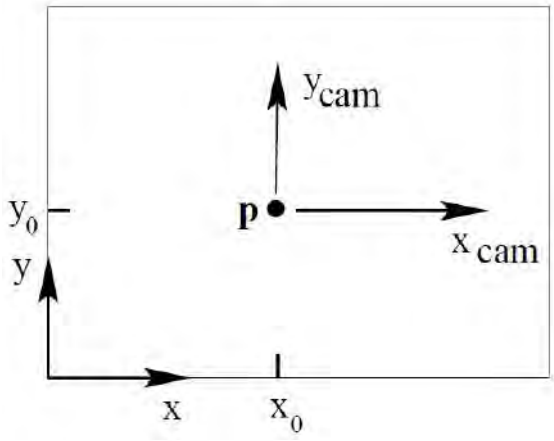


Figure 2.5: Image $(x,\ y)$
and camera $(x_{cam},\ y_{cam})$ coordinate systems [Hartley and Zisserman, 2003].

The equation (2.1) take the origin of the image plane as the principal point. In practice, the principal point is located in a corner as is shown in Fig. 2.5.

$$x = f\frac{X}{Z} + p_x, \quad y = f\frac{Y}{Z} + p_y \tag{2.3}$$

where $(p_x, p_y)^T$ are the coordinates of the principal point. It is possible to express this equation in homogeneous coordinates as

$$\begin{pmatrix} f\frac{X}{Z} \\ f\frac{Y}{Z} \\ Z \end{pmatrix} = \begin{pmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.4}$$

Defining

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

This matrix is called the *camera calibration matrix*. Basically this matrix maps the 3-vector coordinate of the world into a image plane where the principal point is defined by the values of p_x and p_y. The camera calibration matrix (camera model) is formed by some parameters, called *intrinsic parameters*

2.2.1 Intrinsic parameter

The camera calibration matrix can be obtained with some software like Matlab or a C++ code. This matrix contains some parameters, these parameter are called intrinsic parameters because they depend only on the physical characteristics of the camera. The intrinsic parameters are:

- **Focal length:** The focal length is the distance between the image plane and the camera centre. This parameter could be expressed in meters or pixels.
- **Principal point:** The principal point represent the origin in the image plane. As it was said before, in practice, the principal point usually is not the origin. This parameter also can be expressed in meters or pixels.
- **Skew coefficient:** The skew coefficient defines the angle between the x and the y pixel axes. Usually the pixels are square (in Charge Coupled Device (CCD) cameras is possible to have non-square pixels) so the value of this parameter usually will be 90^o.
- **Distortion:** Lens in cameras produce some distortion in images. This means that straight lines in real world has some curvatures in the image. The Barrel distortion is the most common in small cameras and the Pincushion distortion sometimes is present with teleobjectives. See Fig. 2.6.

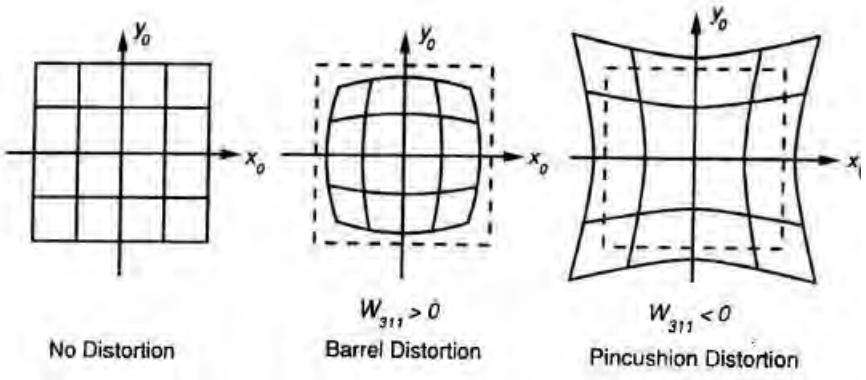


Figure 2.6: Image distortion [Berlin, 2005].

Observe that the parameters which are expressed in pixels are also possible to express them in meters; this happens if the dimensions of a pixel is specified; the most simple cameras have not this parameter. Count with parameters in units of length allows to obtain direct relations between image plane and real world frame.

On the other hand, the distortion is because of using lenses, lenses help to increment the FOV and improve illumination; there are two types of distortion: radial and tangential distortion. In Fig. 2.7 is shown the displacement of the two components of distortion. Radial distortion causes an inward or outward displacement, this is mainly caused by flawed radial curvature of the lens and is strictly symmetric about the optical axis, the radial distortion of a perfectly centred lens is calculated by the .

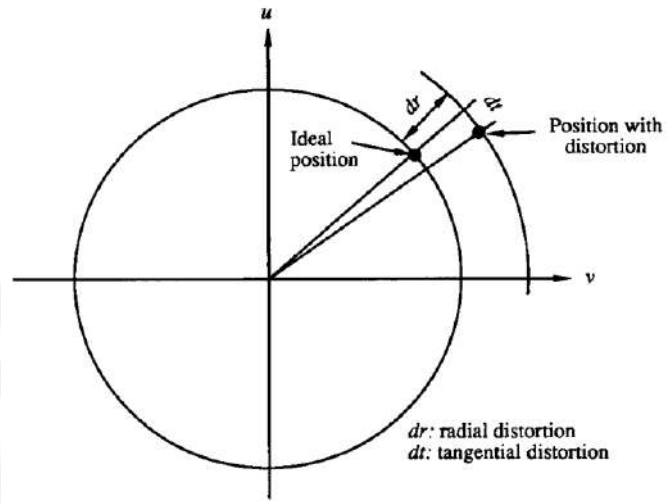


Figure 2.7: Types of distortion [Weng et al., 1992].

If distortion is considered [Heikkila and Silven, 1997], the new point in the image plane will be:

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \begin{bmatrix} \delta x^{(rad)} \\ \delta y^{(rad)} \end{bmatrix} + \begin{bmatrix} \delta x^{(tan)} \\ \delta y^{(tan)} \end{bmatrix}$$

$$\begin{bmatrix} \delta x^{(rad)} \\ \delta y^{(rad)} \end{bmatrix} = \begin{bmatrix} x(k_1 r^2 + k_2 r^4 + ...) \\ y(k_1 r^2 + k_2 r^4 + ...) \end{bmatrix} \quad \begin{bmatrix} \delta x^{(tan)} \\ \delta y^{(tan)} \end{bmatrix} = \begin{bmatrix} 2p_1 xy + p_2(r^2 + 2x^2) \\ p_1(r^2 + 2y^2) + 2p_2 xy \end{bmatrix} \tag{2.5}$$

where $r^2 = x^2 + y^2$, k_1, k_2, ... are coefficient for radial distortion and p_1 and p_2 are coefficient for tangential distortion.

2.2.2 Extrinsic parameters

Until now, all equations are expressed taking the camera centre like origin but, in general, the points in space could be expressed in a different Euclidean frame, called *world coordinates*. It possible to relate both coordinates frames with a rotation and translation movements as it is shown in Fig 2.8.

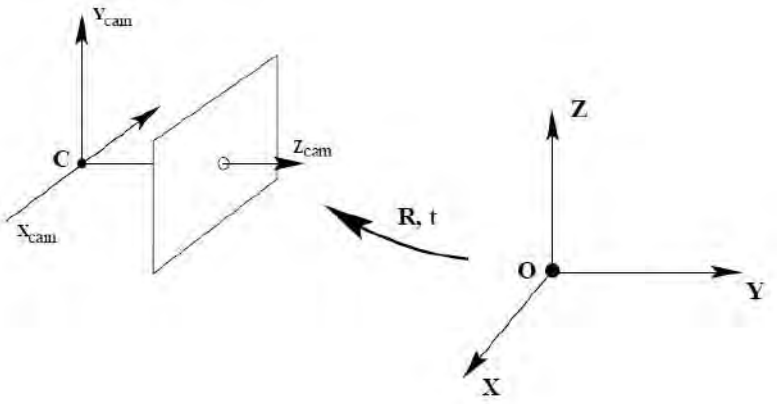


Figure 2.8: Transformation between the world and camera coordinates frames [Hartley and Zisserman, 2003].

Defining

$$\mathbf{x} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

It is easy to see that translation is defined like a additional vector but the rotational 3x3 matrix R can be expressed with a linear transformation using three angles which specify the rotation on each axis (γ, β *and* α).

$$R(\gamma,\beta,\alpha) = R_x R_y R_z = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{pmatrix} \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.6}$$

2.3 Feature detection

Images are large arrays of pixels, with high resolution cameras the amount of pixels increase, but for computer vision applications images have too much data and not enough information, so reducing this amount of data is one goal of feature detection. Features are the gist of the images and depending on the application these can allow to detect object, how fast a object is moving, the shape of the object and how fast the camera is moving.

Feature extraction is really necessary because it helps to reduce large amount of data into only some parameters which can be filtered and reduced as much as the application

needs. It is important to keep in mind that image features is like a summary of all the information present in the image and there is some information which is loose; in addition, the type of feature extractor is chosen depending on the application and doing some assumptions so it will be critical to fulfil this assumption in order to obtain good results. Using an image like input, feature extraction algorithms process the image and returns image features. Image features are commonly scalars (area) or short vectors (corners coordinates, line coordinates). There are several classes of feature: regions, lines and interest points that will be discussed.

2.3.1 Region features

This is the most simple and one of the oldest method of scene understanding, best known as image segmentation. In simple words, this approach allows dividing the scene into areas which represents the same object. Images look different depending on some characteristics of the environment like the illumination, so pixels in the same object could have different colour and this can induce error in the segmentation. Hence, robustness is a important requirement in order to have a accurate segmentation.

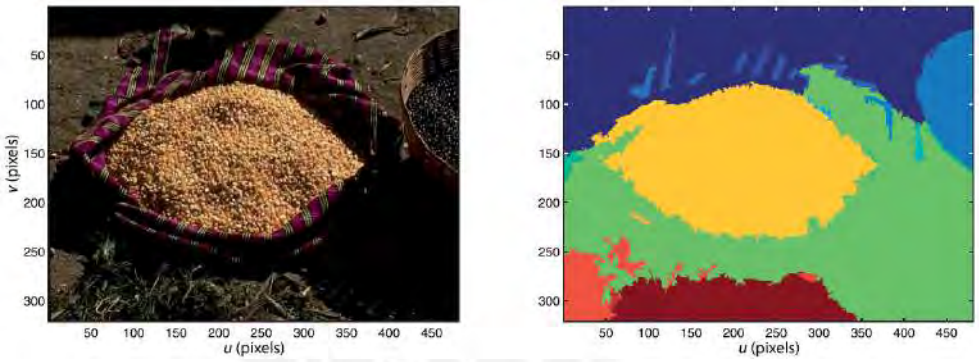


Figure 2.9: Image segmentation [Corke, 2011].

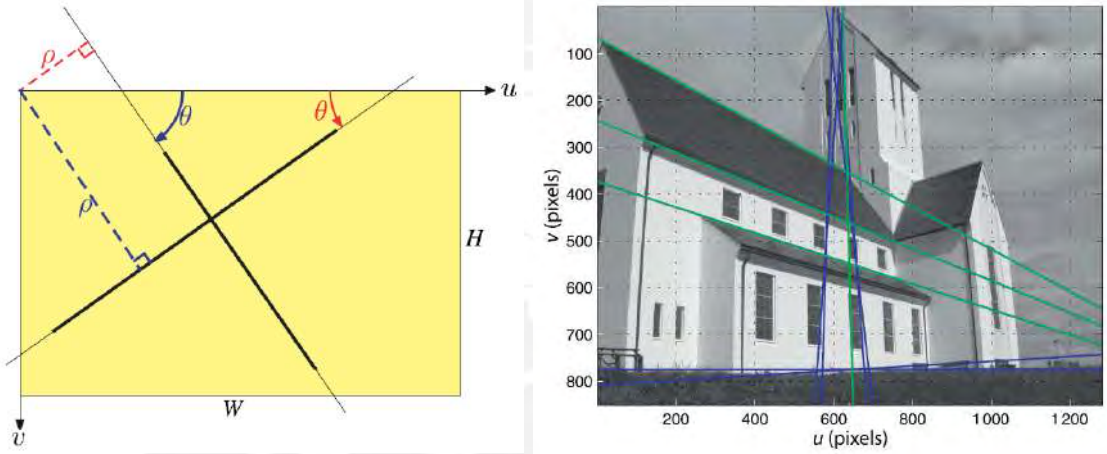
Image segmentation consists of three steps. The first is *classification*, this step is done with each pixel and includes the assignment of one of C classes $c \in \{0 \dots C-1\}$. With $C = 2$ the classification is known as binarization and the pixel is classified as object ($c = 1$) or not-object ($c = 0$). Fig. 2.9 shows a classification with $C = 28$, this is best known as multi-level classification. The second step is *representation* which consists of defining region sets $S_1 \dots S_m$ with adjacent pixels in the same class, this step can be done assigning a set label to each pixel in the image. The third and final step is *description* where the set S_i are represented with scalars or vectors.

2.3.2 Line feature

Lines are common in artificial environments like roads, buildings, doorways, etc. There are algorithms to extract edges from images (Canny edge detector), line feature detectors help to extract straight edges and describe this lines using a minimum number of parameters. The classic representation $v = mu + c$ has a problem in case of vertical lines ($m = \inf$), so it is common to use the (ρ, θ) parametrization.

$$v = -u \tan\theta + \frac{\rho}{\cos\theta}$$

where $\theta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right\rangle$ is the angle from the horizontal axis to the line and $\rho \in [-\rho_{min}, \rho_{max}]$ is the perpendicular distance between the origin and the line. This parametrization is shown in Fig. 2.10a and its application in a real image in Fig. 2.10b.



(a) Line parametrization. (b) Hough transform of real image.

Figure 2.10: Line feature extraction. [Corke, 2011]

2.3.3 Point features

Extract some interesting points of the image is another way to reduce the amount of data; this interest points are also called salient points or keypoints and sometimes but less precisely corner points. A corner point is a basic type of point feature, but the information extracted from one corner point and pixels around this point could be repeated in another corner point; hence, this kind of feature points are not a good option when the application compares two images.

Harris corner detector [Harris and Stephens, 1988] is the most common way to detect corners and the most popular point feature detection algorithms base part of their

performance on this. Analysed blocks of pixels are shown in Fig. 2.11, the cornerness of a block of pixels, $I(x, y)$, could be quantified by comparing it to adjacent blocks in the horizontal, vertical, and diagonal directions and the Harris corner detector can be derived from this comparison as follows.
Let $w(x, y)$ be a binary function which indicates if a pixel (x, y) is inside the block of pixels $(w = 1)$ or outside $(w = 0)$. Now consider the function $E(u, v)$, that is the intensity variation obtained by a small shift of the block of pixels in the direction of the vector (u, v). [Radke, 2013] and [Hassaballah et al., 2016].

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

If the intensity function, $I(x, y)$, is considered as a continuous function and represent it in terms of Taylor series around $(u, v) = (0, 0)$ the results is:

$$\begin{aligned}
E(u,v) &= \sum_{(x,y)} w(x,y) \left(I(x,y) + u\frac{\partial I}{\partial x}(x,y) + v\frac{\partial I}{\partial y}(x,y) - I(x,y) \right)^2 \\
&= \sum_{(x,y)} w(x,y) \left(u\frac{\partial I}{\partial x}(x,y) + v\frac{\partial I}{\partial y}(x,y) \right)^2 \\
&= \sum_{(x,y)} w(x,y) \left(u^2 \left(\frac{\partial I}{\partial x}(x,y) \right)^2 + 2uv \left(\frac{\partial I}{\partial x}(x,y)\frac{\partial I}{\partial y}(x,y) \right) + v^2 \left(\frac{\partial I}{\partial y}(x,y) \right)^2 \right)^2 \\
&= \begin{bmatrix} u \\ v \end{bmatrix}^T \begin{bmatrix} \sum_{(x,y)} w(x,y) \left(\frac{\partial I}{\partial x}(x,y) \right)^2 & \sum_{(x,y)} w(x,y) \left(\frac{\partial I}{\partial x}(x,y)\frac{\partial I}{\partial y}(x,y) \right) \\ \sum_{(x,y)} w(x,y) \left(\frac{\partial I}{\partial x}(x,y)\frac{\partial I}{\partial y}(x,y) \right) & \sum_{(x,y)} w(x,y) \left(\frac{\partial I}{\partial y}(x,y) \right)^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}
\end{aligned} \tag{2.7}$$

The symmetric matrix in Equation (2.7) is called the *Harris matrix* and is positive definite.

$$\begin{bmatrix} \sum_{(x,y)} w(x,y) \left(\frac{\partial I}{\partial x}(x,y) \right)^2 & \sum_{(x,y)} w(x,y) \left(\frac{\partial I}{\partial x}(x,y)\frac{\partial I}{\partial y}(x,y) \right) \\ \sum_{(x,y)} w(x,y) \left(\frac{\partial I}{\partial x}(x,y)\frac{\partial I}{\partial y}(x,y) \right) & \sum_{(x,y)} w(x,y) \left(\frac{\partial I}{\partial y}(x,y) \right)^2 \end{bmatrix} \tag{2.8}$$

The eigenvalues (λ_1, λ_2) and eigenvectors (e_1, e_2) of the Harris matrix can be analysed in order to find corners. Considering $\lambda_1 >= \lambda_2$ there are the following cases, shown in Fig. 2.11:

1. The block is almost constant in all its pixels. The partial derivatives $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$ will both nearly zero for all pixels in the block. The surface will be nearly flat

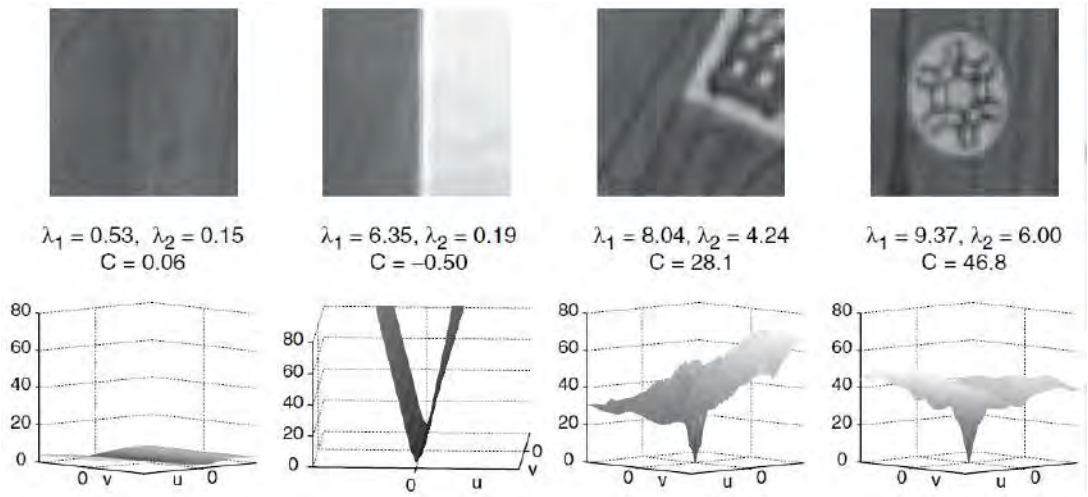


Figure 2.11: From top to bottom: Candidate feature blocks, Harris matrix eigenvalues with Harris quality measure C ($k = 0.04$) and error surfaces $E(u, v)$ around block centre [Radke, 2013].

and thus $\lambda_1 \approx \lambda_2 \approx 0$.

2. The block shows a linear edge. For pixels far from the linear edge the partial derivatives $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$ are both close to zero and for pixels near the edge both derivatives will be perpendicular to the edge direction. Thus, λ_1 is a positive value and e_1 will be normal to the edge direction, while $\lambda_2 \approx 0$ and its corresponding vector e_2 will be along to the edge direction. The surface $E(u, v)$ is similar to the letter V and has its minimal values in the direction of the edge.

3. The block contains a corner or blob. In the first case the edge is not continuous and, after the edge, the pixels are not uniform. In the second case the edge is not linear and inside the blob the pixels are also not uniform. In both cases, the edges define a small not uniform area, thus the surface E(u,v) is similar to a bowl. Both λ_1 and λ_2 will be positive.

The quality measure C is a parameter define as

$$C = det(H) - k\, trace(H)^2$$

where k usually take a value around 0.04 (the lower value of k, the more sensitive de quality measure). C will be large when both eigenvalues are large and C will be near zero if one eigenvalue is small Fig. 2.11. To detect features in the image, it is sufficient to analyse the value of the quality measure (C) at each block in the image,

and select some of them where the quality measure is above a minimum threshold. Finally, the resulting features will called *Harris corners*.

Scale-Scale Feature Detection

In many real applications the camera moves around the environment, thus the scene changes its scale and orientation. In this case, of applications Harris corners detector responds poorly. In addition, the Harris corner detector is very sensitive to texture, this means that small changes in the pixel intensity (details like leaves in a tree) can generate a new feature point, hence detecting features associated with large sectors in the image is not possible.
In order to explain the fundamental principle of scale-space feature detection look at Fig. 2.12a. Fig. 2.12a contains four squares of different size 5x5, 9x9, 17x17 and 33x33, except for the edges there is no too much details in this image and Gaussian blur will be used to eliminate details and make smooth variations along the image. Gaussian blur has a particular expression, convolution of the Gaussian operator and the image, that is applied to each pixel and the result is a blurred image.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

Where:

- L is the blurred image.
- G is the Gaussian blur operator.
- I is an image.
- x and y are the location coordinates.
- σ is the "scale" parameter. This parameter can be taken as the blur degree.
- The * is the convolution operator in x and y.

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

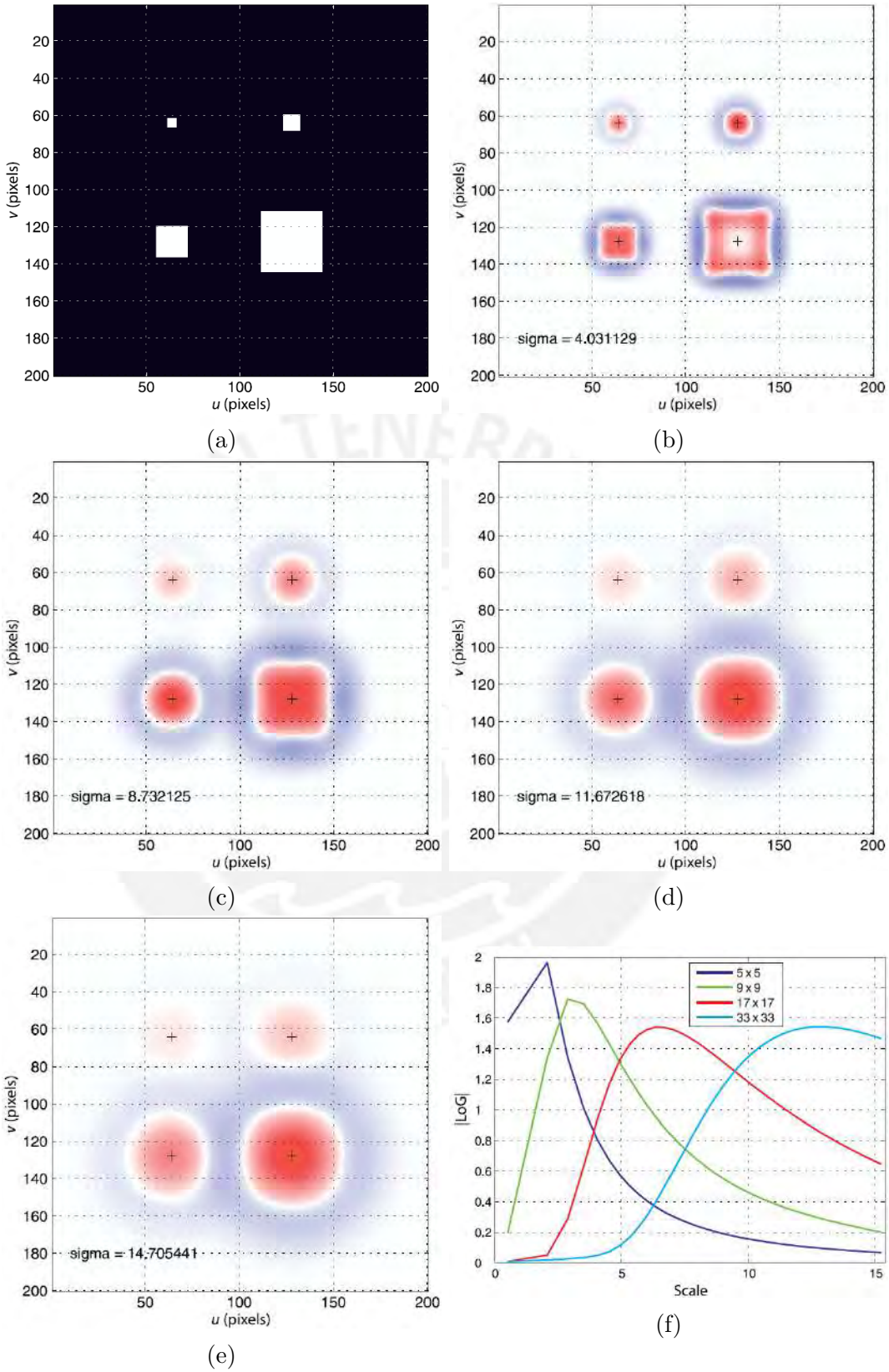


Figure 2.12: Scale-space example. [Corke, 2011]

The Gaussian blur allows to create a scale-space that is a set of blurred images with a increasing scale parameter σ. Afterwards, edges (points of high gradient or null second derivative) are founded applying the Laplacian operator in each Gaussian-smoothed image.

$$\nabla^2 I = \frac{\partial^2 I}{\partial u^2} + \frac{\partial^2 I}{\partial v^2} \tag{2.9}$$

Two sets of images will be created after the Gaussian and Laplacian operators, one set contains the scale-space generated with the Gaussian operator and the other set contain edge information in different scales generated with the Laplacian of Gaussian (LoG) (Fig. 2.12b-e). Fig. 2.12f shows the magnitude of the LoG response as a function of scale, taken in the centre point of each square in Fig. 2.12a. Each curve has a maximum value and note that the scale associated with the maximal is directly proportional to the size of the square.

Considering the set of images, obtained by applying the LoG operator, as a volume the scale-space feature point will be any pixel that is grater that its neighbours in all three dimensions. It is easy to see that LoG is difficult to calculate because there are many images in the output, thus the Difference of Gaussian (DoG) is used. DoG is based on simply the difference of blurred images and is easy to calculate compared to LoG and the result is a good approximation of this.

Popular algorithms for feature detection which are capable to calculate their scale and orientation are based on scale-space concepts. The SIFT (Scale-Invariant Feature Transform) algorithm is based on the maximum point of a sequence of images obtained by the DoG. The SURF (Speeded-Up Robust Feature) algorithm is based on the maximal value in an approximate Hessian of Gaussian sequence. There are many others algorithms but these two are the most important and accurate. The result of SIFT, SURF and others are shown in Fig. 2.13.

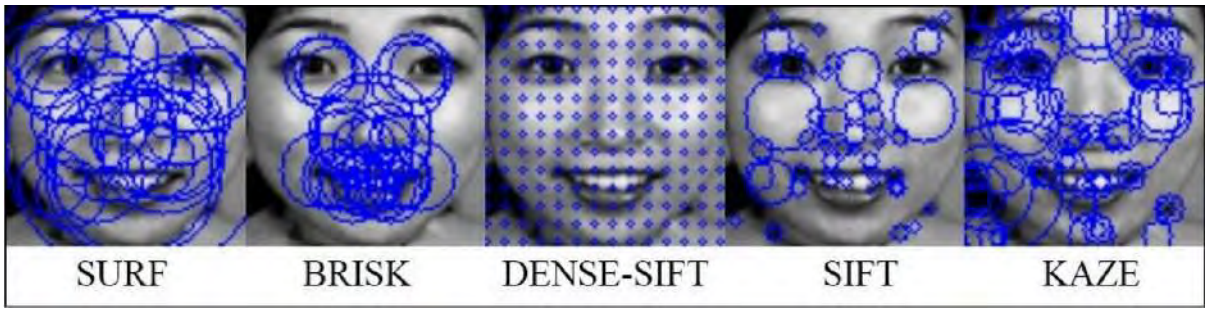


Figure 2.13: Feature extraction methods [Howse et al., 2015].

The matching of two images, compare and extract some common features, is a usual

problem in computer vision. SIFT and SURF are powerful tools in order to obtain good matched feature points because of both get good result even with variation of scale, rotation, illumination and viewpoint. SURF is a improved version of the SIFT. Matching can be used for example for face recognition, recognize objects in an image, objects tracking, etc.
The entire SIFT algorithm can be divided in the following parts [Shack, 2017].

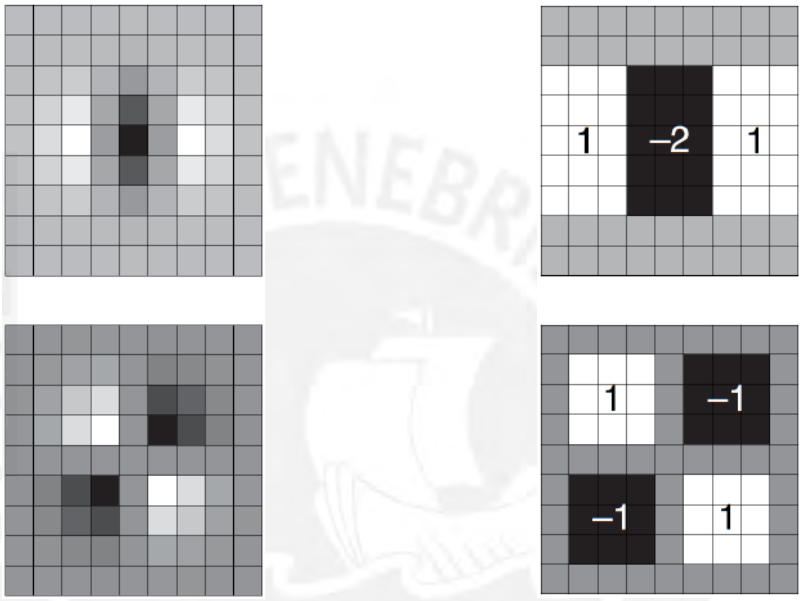
- **Construction a scale-space:** This is the initial preparation and was described before. The scale-space is generated applying a Gaussian blur of different degrees and the final result is a set of blurred images.
- **LoG approximation:** The Laplacian of Gaussian (LoG) was also described before and basically is obtain second derivatives of every blurred images applying the Laplacian operator 2.2. This operation is computationally expensive, thus the Difference of Gaussian (DoG) is used which is a good approximation of the LoG.
- **Finding keypoints:** The keypoints are the maxima and minima in the DoG images obtained in the previous step.
- **Discard bad keypoints:** Edges and low contrast regions are bad keypoints. Eliminating bad keypoints makes the algorithm efficient and robust. A technique similar to the Harris Corner Detector is used in this step.
- **Assigning an orientation to the keypoints:** An orientation is calculated for each keypoint. This makes the rotation invariant because of any further operation is relative to this orientation.
- **Generate SIFT features:** Finally, with scale and rotation invariance in place, one more representation is generated. This way of represent features are called *descriptors* and it will be described later. Descriptors allow to identify easily the feature that is necessary to find.

The SURF algorithm is a fast and robust method for feature detection and invariant representation and comparison of images. With the SURF algorithm, for the detection stage of keypoints, instead of compute the Laplacian operator (2.9), the computation is based on the simple 2D box filters (Fig. 2.14a-b) and integral image ([Viola and Jones, 2004]) to approximate the second order Gaussian derivatives (Laplacian operator) more efficiently. The *integral image* (2.10) is a new way to represent images that allows for

very fast feature evaluation and when it is used for the computation, the speed of applying the box filter is independent of the filter size.

$$ii(x,y) = \sum_{\substack{x' \leq x \\ y' \leq y}} i(x',y') \tag{2.10}$$

where $ii(x,y)$ is the integral image and $i(x,y)$ is the original image.



(a) (b)

Figure 2.14: (a) Example 9 x 9 Gaussian derivatives filters using for computing the Hessian, with $\sigma = 1.2$. The top filter is $\frac{\partial^2 L(x,y,\sigma)}{\partial x^2}$ and the bottom filter is $\frac{\partial^2 L(x,y,\sigma)}{\partial x \partial y}$. Light values are positive, black values are negative, and gray values are near zero. (b) Efficient box filter approximations of of the filters at left. Gray values are zero. [Radke, 2013]

But what happens if the two images to match was taken with a great camera motion around the scene. In this case, circular feature would contain different set of pixels and in consequence descriptors based on these neighbourhoods would use different information, giving suboptimal matches. To fix this problem is required an affine-invariant way of detecting features. First research and theory of affine-invariant regions was proposed by Lindeberg and Gärding [Lindeberg and Gårding, 1997].
It is possible to find affine-invariant elliptical regions around feature points with a straighforward iterative procedure called **affine adaptation** [Radke, 2013]:

1. Detect the feature point position and its characteristic scale.

2. Compute H, which is the local Harris matrix at a given scale (scale-normalized Harris matrix). Scale H so it has unit determinant.

3. Compute the Cholesky factorization $H = CC^T$ where C is a lower-triangular matrix with non-negative diagonal elements.

4. Warp the image around the feature point using the linear transformation C. $I_{new}(x_{new}) = I_{old}(Cx_{old})$.

5. Compute H, the local Harris matrix for the new image and scale it so it has unit determinant.

6. If H is sufficient close to the identity (i.e. its eigenvalues are nearly equal), stop. Otherwise , go to Step 3.

Therefore, featured detected in scale space can be associated with scale-covariant circle or an affine-covariant ellipse.

2.4 Feature Descriptor

After feature detection the next step is to describe these features with vectors of numbers called *descriptors*. In order to obtain a good feature matching, in spite of different types of camera used in many applications, the descriptor from the same 3D location and different views of the same scene must be similar. If D is an algorithm to create a descriptor, further f and f' are detected features of different images, then $D(f) \approx D(\hat{f})$. Feature detection must be covariant respect to a geometric or photometric transformation, ,$\hat{f} = Tf$, and feature descriptors must be invariant to them.
A simple descriptor could be a vector with intensity values from a group of pixels around the feature location. The Sum of Squared Differences (SSD) for two images $f(x, y)$ and $g(x, y)$ it is defined as

$$SSD(d_1, d_2) = \sum_{i=-n_1}^{n_1} \sum_{j=-n_2}^{n_2} (f(x+i, y+j) - g(x+i-d_1, y+j-d_2))^2$$

where images have a resolution of $(2n_1 + 1) \times (2n_2 + 1)$ pixels. With $d_1 = d_2 \approx 0$, that means the change between two images is small and the SSD is computed with corresponding elements, as happen with two consecutive frames of a video, this approach has good results. But considering the robust feature detection approaches

that was seeing before SSD is unsuitable in case of images with significantly different scales, rotation, illumination and point of view.
With the information produced through the feature detector explained before (scale-invariant feature as LoG and DoG) features are expressed as a point in the centre of a circular region whose radius is the characteristic scale (Fig. 2.13). It possible to use these defined circular regions at the end of the affine adaptation process as the basis for an affine-invariant descriptor.

3 Mathematical Modelling for Estimation of Obstacle Position in 3D Space

In order to estimate the distance between the obstacle and the UAV it is a requirement to compute the obstacle feature point $P_o = (x_o, y_o, z_o)$, which is the world coordinates of the feature point in the obstacle. With P_o and the actual position of the UAV $P_{uav} = (x_{uav}, y_{uav}, z_{uav})$ is easy to calculate the distance between these two points applying the Pythagorean theorem

$$d = \sqrt{(x_o - x_{uav})^2 + (y_o - y_{uav})^2 + (z_o - z_{uav})^2}$$

In order to obtain P_o in distance units, it is necessary to express all the needed parameters in meters. The mathematical model is proposed by Jeong-Oog [Lee et al., 2011] who used the projective geometry in two consecutive images. The UAV position must be as accurate as it is possible and in experiments it is estimated using wireless modules.

With some changes in the projection model presented in Fig. 2.3 in chapter 2. The world coordinates frame $\{W\}$ defined by $(\vec{x}, \vec{y}, \vec{z})$, the robot coordinates frame $\{R\}$ and the camera coordinates frame $\{C\}$ defined by $(\vec{n}_x, \vec{n}_y, \vec{n}_z)$ are shown in Fig. 3.2. This figure also shows the projection of the obstacle feature point P_o into the image plane P_I. In order to simplify the mathematical representation, the rotation matrix in $\vec{n}_x$ and $\vec{n}_y$ axis are the identity matrix ($R_{\vec{n}_x} = R_{\vec{n}_y} = I$); this means that the UAV is always parallel to the ground ($\vec{n}_z = \vec{z}$). The camera is rigidly mounted on the UAV, therefore $\{C\}$ and $\{R\}$ are assumed to be aligned.

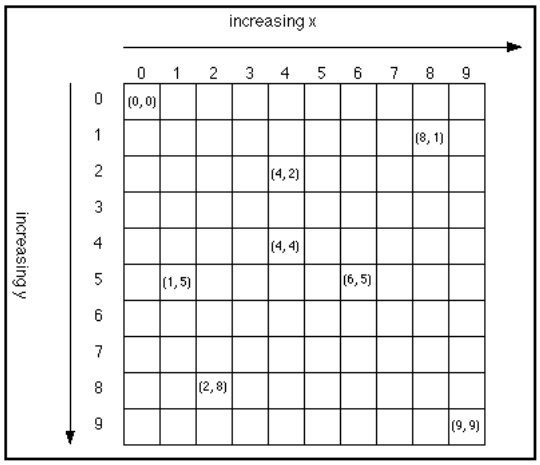


Figure 3.1: Image frame coordinates in OpenCV [OpenCV, 2017a].

The software used for simulations is OpenCV, where the principal point in the image plane is placed in the upper left corner, as it is shown in Fig. 3.1, (similar to Fig. 2.5 in chapter 2, but with some different signs).

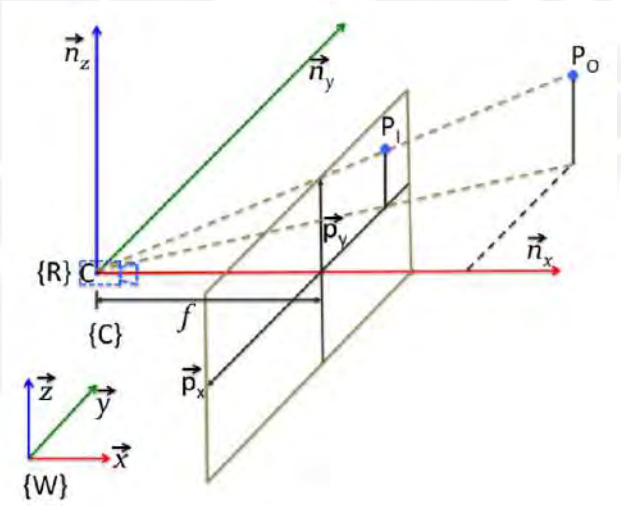


Figure 3.2: The central-projection model and the projection of the obstacle feature point (P_o) on the image plane [Saha et al., 2014].

Redefining the equation (2.3) it is possible to obtain:

$$x = \frac{\tilde{x}}{dx} - p_x, \quad y = -\frac{\tilde{y}}{dy} + p_y \tag{3.1}$$

Equation (3.1) gives a relation between the pixel position in the image plane expressed in meters ($\tilde{x}_I$ and $\tilde{y}_I$), it will be necessary to develop the model, and the pixel position in the captured image by OpenCV. There are two additional parameters, $dx = dy =$

$1.4\mu m$ are the pixel size of the camera used for experiments [Hardkernel, 2017]. In order to obtain a model, it will be necessary to obtain relations between the world coordinates frame $\{W\}$ and camera coordinates frame $\{C\}$ but for computing P_o two of the inputs in the model are $\tilde{x}_I$ and $\tilde{y}_I$ which are the pixel position in the image plane but expressed in meters.

Taking the new coordinates frames shown in Fig. 3.2 and defining $P_o = (X_C, Y_C, Z_C)$ like coordinates of P_o in the camera frame $\{C\}$, the equation (2.1) changes to

$$\tilde{x}_I = -f_x \frac{\overrightarrow{CY_C}}{\overrightarrow{CX_C}}, \quad \tilde{y}_I = f_y \frac{\overrightarrow{CZ_C}}{\overrightarrow{CX_C}} \tag{3.2}$$

Observe that there is a negative sign because of the $\vec{n}$ axis in the camera frame and the $\vec{p_x}$ axis in the image frame are opposite.

Since the UAV is always parallel to the ground, thus there would be only a rotation in the $\vec{n}_z$ axis, according to equation (2.6) the relation between vectors in camera coordinates $\{C\}$ and the world coordinates $\{W\}$ is only R_z.

$$\begin{bmatrix} \vec{n}_x \\ \vec{n}_y \\ \vec{n}_z \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \vec{x} \\ \vec{y} \\ \vec{z} \end{bmatrix}$$

Then

$$\begin{aligned} \vec{n}_x &= a\,\vec{x} + b\,\vec{y} \\ \vec{n}_y &= -b\,\vec{x} + a\,\vec{y} \\ \vec{n}_z &= \vec{z} \end{aligned} \tag{3.3}$$

Defining $a = \cos\alpha$ and $b = -\sin\alpha$. Note that coordinates in the camera frame $\vec{n}_x$, $\vec{n}_y$ and $\vec{n}_z$ can be expressed as points difference in the world frame, for example $\vec{n}_z = \vec{z} = z_o - z_i$ where z_o and z_i are the initial and final points in the world coordinate frame, the other coordinates can also be defined in this simple way if there is no rotation.

$\overrightarrow{CX_C} \in \vec{n}_x$, $\overrightarrow{CY_C} \in \vec{n}_y$, $\overrightarrow{CZ_C} \in \vec{n}_z$ and with equations (3.1) and (3.3) is possible to

obtain the following relations:

$$\begin{aligned} \tilde{x}_I &= \frac{b\,\vec{x} - a\,\vec{y}}{a\,\vec{x} + b\,\vec{y}}\, f_x \\ \tilde{y}_I &= \frac{\vec{z}}{a\,\vec{x} + b\,\vec{y}}\, f_y \end{aligned} \tag{3.4}$$

The world coordinates can be defined $\vec{x} = x_o - x_i$ and $\vec{y} = y_o - y_i$, where x_o, y_o and $\vec{z}_o$ are the world coordinates of the obstacle feature point, in addition, x_i, y_i and z_i represent the position of the camera when the image i is captured.
In order to estimate the obstacle position, the model uses two captured images $i = 1$, 2 and through (3.4) the following equations are obtained:

$$\tilde{x}_{I_1} = \frac{b_1(x_o - x_1) - a_1(y_o - y_1)}{a_1(x_o - x_1) + b_1(y_o - y_1)} f_x \tag{3.5}$$

$$\tilde{y}_{I_1} = \frac{z_o - z_1}{a_1(x_o - x_1) + b_1(y_o - y_1)} f_y \tag{3.6}$$

$$\tilde{x}_{I_2} = \frac{b_2(x_o - x_2) - a_2(y_o - y_2)}{a_2(x_o - x_2) + b_2(y_o - y_2)} f_x \tag{3.7}$$

$$\tilde{y}_{I_2} = \frac{z_o - z_2}{a_2(x_o - x_2) + b_2(y_o - y_2)} f_y \tag{3.8}$$

Dividing equation (3.5) by (3.6), the result is:

$$\frac{\tilde{x}_{I_1}}{\tilde{y}_{I_1}} = \frac{[b_1(x_o - x_1) - a_1(y_o - y_1)]}{a_1(x_o - x_1) + b_1(y_o - y_1)} f_x$$

$$f_y \tilde{x}_{I_1}(z_o - z_1) = f_x b_1 \tilde{y}_{I_1}(x_o - x_1) - f_x a_1 \tilde{y}_{I_1}(y_o - y_1)$$

$$f_x b_1 \tilde{y}_{I_1} x_o - f_x a_1 \tilde{y}_{I_1} y_o - f_y \tilde{x}_{I_1} z_o = f_x b_1 x_1 \tilde{y}_{I_1} - f_x a_1 y_1 \tilde{y}_{I_1} - f_y \tilde{x}_{I_1} z_1 \tag{3.9}$$

Similarly, dividing equation 3.7 by 3.8, the result is:

$$f_x b_2 \tilde{y}_{I_2} x_o - f_x a_2 \tilde{y}_{I_2} y_o - f_y \tilde{x}_{I_1} z_o = f_x b_2 x_2 \tilde{y}_{I_2} - f_x a_2 y_2 \tilde{y}_{I_2} - f_y \tilde{x}_{I_2} z_2 \tag{3.10}$$

Now, dividing equation 3.6 by f_y in both sides and the result is:

$$\frac{\tilde{y}_{I_1}}{f_y} = \frac{z_o - z_1}{a_1(x_o - x_1) + b_1(y_o - y_1)}$$

$$\frac{y_{I_1}}{f_y}\left[a_1(x_o - x_1) + b_1(y_o - y_1)\right] = z_o - z_1 \tag{3.11}$$

Similarly, dividing equation 3.8 by f_y in both sides the result is:

$$\frac{y_{I_2}}{f_y}\left[a_2(x_o - x_2) + b_2(y_o - y_2)\right] = z_o - z_2 \tag{3.12}$$

Now, subtracting equation 3.12 from equation 3.11, the result is:

$$\frac{1}{f_y}\left[\tilde{y}_{I_1}\left\{a_1(x_o - x_1) + b_1(y_o - y_1)\right\} - \tilde{y}_{I_2}\left\{a_2(x_o - x_2) + b_2(y_o - y_2)\right\}\right] = z_2 - z_1$$

$$\tilde{y}_{I_1}\left[a_1x_o - a_1x_1 + b_1y_o - b_1y_1\right] - \tilde{y}_{I_2}\left[a_2x_o - a_2x_2 + b_2y_o - b_2y_2\right] = f_y(z_2 - z_1)$$

$$x_o(a_1\tilde{y}_{I_1} - a_2\tilde{y}_{I_2}) + y_o(b_1\tilde{y}_{I_1} - b_2\tilde{y}_{I_2}) - a_1x_1\tilde{y}_{I_1} + a_2x_2\tilde{y}_{I_2} - b_1y_1\tilde{y}_{I_1} + b_2y_2\tilde{y}_{I_2} = f_y(z_2 - z_1)$$

$$x_o(a_1\tilde{y}_{I_1} - a_2\tilde{y}_{I_2}) + y_o(b_1\tilde{y}_{I_1} - b_2\tilde{y}_{I_2}) = f_y(z_2 - z_1) + a_1x_1\tilde{y}_{I_1} - a_2x_2\tilde{y}_{I_2} + b_1y_1\tilde{y}_{I_1} - b_2y_2\tilde{y}_{I_2} \tag{3.13}$$

Finally, there are three equations: (3.9), (3.10) and (3.13) and three unknown variables: x_o, y_o and z_o. The unknown variables are in the left hand and all the known variables in the right hand. These equations are shown below.

$$f_xb_1\tilde{y}_{I_1}x_o - f_xa_1\tilde{y}_{I_1}y_o - f_y\tilde{x}_{I_1}z_o = f_xb_1x_1\tilde{y}_{I_1} - f_xa_1y_1\tilde{y}_{I_1} - f_y\tilde{x}_{I_1}z_1$$

$$f_xb_2\tilde{y}_{I_2}x_o - f_xa_2\tilde{y}_{I_2}y_o - f_y\tilde{x}_{I_1}z_o = f_xb_2x_2\tilde{y}_{I_2} - f_xa_2y_2\tilde{y}_{I_2} - f_y\tilde{x}_{I_2}z_2$$

$$x_o(a_1\tilde{y}_{I_1} - a_2\tilde{y}_{I_2}) + y_o(b_1\tilde{y}_{I_1} - b_2\tilde{y}_{I_2}) = f_y(z_2 - z_1) + a_1x_1\tilde{y}_{I_1} - a_2x_2\tilde{y}_{I_2} + b_1y_1\tilde{y}_{I_1} - b_2y_2\tilde{y}_{I_2}$$

In order to simplify the notation the following coefficients are defined:

$$c_1 = f_xb_1\tilde{y}_{I_1}$$
$$c_2 = -f_xa_1\tilde{y}_{I_1}$$
$$c_3 = -f_y\tilde{x}_{I_1}$$
$$c_4 = f_xb_2\tilde{y}_{I_2}$$
$$c_5 = -f_xa_2\tilde{y}_{I_2}$$

$$
\begin{aligned}
c_6 &= -f_y \tilde{x}_{I_2} \\
c_7 &= a_1 \tilde{y}_{I_1} - a_2 \tilde{y}_{I_2} \\
c_8 &= b_1 \tilde{y}_{I_1} - b_2 \tilde{y}_{I_2} \\
c_9 &= 0
\end{aligned}
$$

and the constants:

$$
\begin{aligned}
B_1 &= f_x b_1 x_1 \tilde{y}_{I_1} - f_x a_1 y_1 \tilde{y}_{I_1} - f_y \tilde{x}_{I_1} z_1 \\
B_2 &= f_x b_2 x_2 \tilde{y}_{I_2} - f_x a_2 y_2 \tilde{y}_{I_2} - f_y \tilde{x}_{I_2} z_2 \\
B_3 &= f_y (z_2 - z_1) + a_1 x_1 \tilde{y}_{I_1} - a_2 x_2 \tilde{y}_{I_2} + b_1 y_1 \tilde{y}_{I_1} - b_2 y_2 \tilde{y}_{I_2}
\end{aligned}
$$

Now, equations (3.9), (3.10) and (3.13) can be represented with matrix notation:

$$
Ax = B \tag{3.14}
$$

where

$$
A = \begin{bmatrix} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ c_7 & c_8 & c_9 \end{bmatrix}, \qquad x = \begin{bmatrix} x_o \\ y_o \\ z_o \end{bmatrix}, \qquad B = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix}
$$

Equation (3.14) represents the mathematical model which estimates the 3D location of the obstacle feature point $P_o = (x_o, y_o, z_o)$. There are many approaches to compute the solution of (3.14) but in general the approach to choose depends on the form of the matrix A. For simulations A is assumed to be a non symmetric and non sparse matrix because of a rotation of 45 degrees between world coordinates frame and camera coordinates frame is considered, in addition float variables with great size and range are used in order to have more accuracy. With these conditions (3.14) can be solved by a direct method called LU factorization. For a more flexible movement in the UAV a general approach can be used like the Generalized Minimal Residual Generalized Minimal Residual (GMRES).

4 Obstacle Avoidance Implementation

The implementation only includes test off-line, which means that the obstacle avoidance system is tested with previously captured images, position and orientation data, also logged previously. The entire algorithm is programmed in C++ [OpenCV, 2017c] but some parameters necessary for modelling (camera parameters) are computed using Matlab [Mathworks, 2017]. The image processing steps (image capture, feature detection and matching) are performed with OpenCV libraries. Feature detection and matching algorithms are already programmed in OpenCV but code for image capture, filtering of matched decriptors, the obstacle distance estimation and the code for logging data were developed for this work. The position, orientation and captured images are read from a disc location and all the output data (distance estimated and some important parameters as the 3D obstacle position estimated) are logged in text files and output images. Finally, all results are processed in Matlab in order to display more useful graphs, compare different methods and formulate conclusions.

4.1 Obstacle detection block diagram

The obstacle detection system is conformed of several stages, from image capture and position reading to distance estimation. Some stages are only executed once; as examples, the camera calibration and camera model which are performed in the beginning, On the other hand stages as feature detection and distance estimation are executed every time one image is captured. The obstacle avoidance block diagram is shown in 4.1. the present work is focused in the obstacle distance estimation and the path planning. In order to estimate the obstacle distance it is necessary to obtain the camera parameters, form the respective mathematical model, read the actual position and process the captured images.
For a better understanding of 4.1 a small description of every block will be realized, take into account that the purpose of some blocks were already developed.

- **Camera calibration and camera model:** Camera calibration toolbox from Matlab is used and then the pin-hole camera model is formed. Camera calibration allows to compute the focal lengths: f_x and f_y, distortion coefficients: kc
- **Image capture:** The first step is to capture two consecutive images, *img1* and *img2*; these two images must be captured with a certain difference of time to generate a dis-

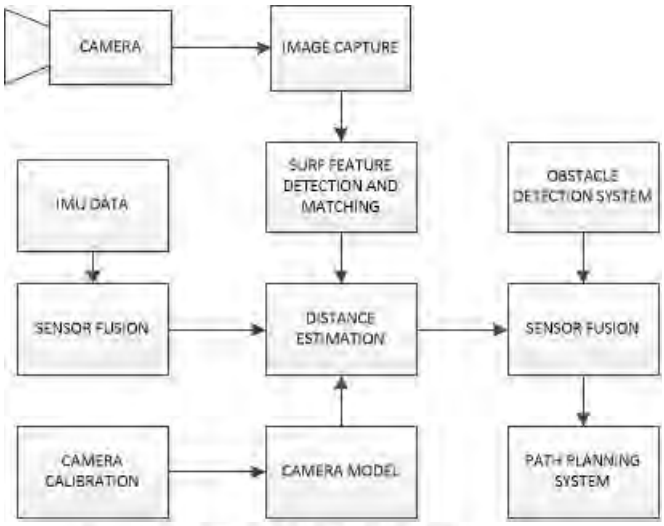


Figure 4.1: Extrinsic parameters.

tance variation between *img1* and *img2*. One task is to find the best distance difference between these two images in order to obtain good obstacle distance estimation.

- **Feature detection and matching:** Chapter 2 describes all about feature detection and matching, in this step both images are processed to find common keypoints and then apply the obstacle distance estimation algorithm. Keypoints are expressed with two coordinates sets for each image: *xim1*, *yim1* for *img1* and *xim2*, *yim2* for *img2*

- **Position modules:** In order to apply the mathematical model developed before, accurate position data is very important. This data is obtained from USB position modules developed for Fraunhofer Institutm they are 5 USB modules, one in the origin, 3 in the end of every axis and the last one on the quadrocopter. The data obtained from position modules are: x_1, y_1, z_1 for *img1* and x_2, y_2, z_2 for *img2*.

- **Distance estimation:** Using the camera model, positions data, matched keypoints and some assumptions the mathematical model to estimate distance is implemented in C++.

- **Ultrasonic obstacle detection:** The quadrocopter has already a obstacle avoidance system with ultrasonic sensors. A Kalman Filter is used for sensor fusion and one more task is to add the data obtained from the obstacle distance estimation block.

- **Path-planning:** After the obstacle is detected the UAV applies a path-planning algorithm to avoid the obstacle.

The obstacle detection and distance estimation is implemented with the algorithm 1:

Algorithm 1 Obstacle distance estimation.

Require: Two captured image $img1$ and $img2$, orientation Roll-Pitch-Yaw (RPY) angles, position data for both images x_1, y_1, z_1, x_2, y_2 and z_2.

```
while img1 ≠ 0, img2 ≠ 0 do
    Read img1, img2, RPY angles, x1, y1, z1, x2, y2 and z2.
    Feature detection and descriptor computation algorithm, example: (SURF).
    Feature matching algorithm, example: Brute Force (BF) algorithm.
    Read size of matched feature match_size.
    Determine the minimal feature distance match_dist_min.
    for i = 0 to match_size do
        if Matched feature distance less than 4 times match_dist_min then
            Estimation of the 3D obstacle position.
            Distance to the obstacle computation.
            Important data logging.
        end if
    end for
end while
```

4.2 Camera calibration with Matlab

In order to test the mathematical model, first, it is necessary to identify some parameters like the focal lengths f_x and f_y, principal point, skew coefficient and distortion coefficients, these are called intrinsic parameter and they were described in the second chapter. All these parameters depend on the camera being used; for this reason it is important to choose one with suitable features for the application.

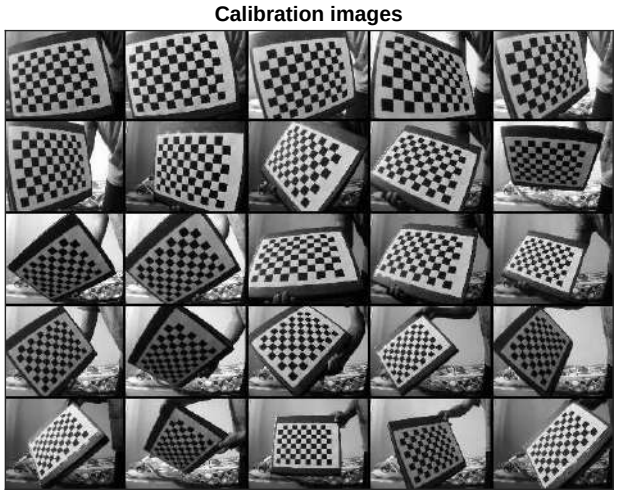
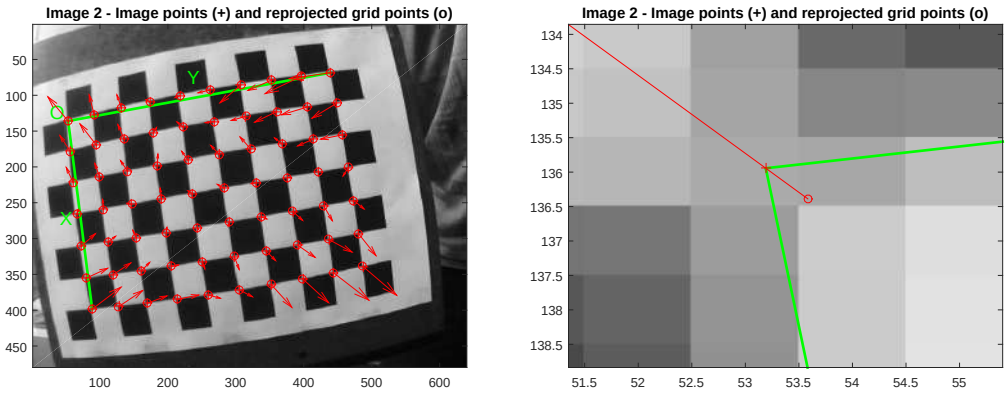


Figure 4.2: Chessboard pattern images for camera calibration

The procedure to obtain all the intrinsic parameters is the so-called *camera calibration*. There are many software applications that can compute the procedure but Matlab has a camera calibration toolbox [Various, 2017] which will be used. This Matlab toolbox is based on the following literature: [Zhang, 1999], [Heikkila and Silven, 1997], [Tsai, 1987], [Sturm and Maybank, 1999] and [Clarke and Fryer, 1998].

In order to use the toolbox for camera calibration in Matlab, it is necessary to capture some images with a planar checkerboard pattern with known dimensions. The checkerboard must be captured with different orientation and positions with respect to the camera, Fig. 4.2 shows sixteen of the forty images used to perform the calibration and the corners of every square is detected automatically.

The planar checkerboard pattern with the detected corners point (red cross) and the respective reprojected points (red circle) is shown in Fig. 4.3a; the checkerboard has eight by eleven squares and the dimension of every square is $dx = 23mm$ and $dy = 23mm$, the red arrows are on the reprojected corner and indicate the direction where the real corner is located, while bigger the arrow bigger the error is. It is also easy to see that the error is minimal in the center of the image, and it grows in distant points.



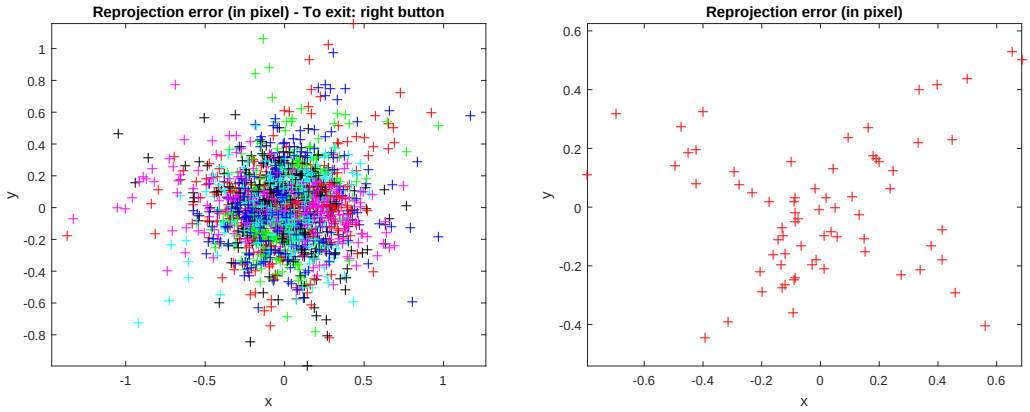
(a) Image 2 with reprojected corners points.

(b) Zoom of the upper left reprojected corner in image 2

Figure 4.3: Chessboard pattern in image 2 after camera calibration

In order to have a better appreciation, Fig. 4.3b shows a better focus of the upper left reprojected corner. There are two close points, the first one is the detected corner, it is located with a red cross and the second one is the reprojected corner calculated with equations and intrinsic parameters, it is located with a red circle. With a ideal camera calibration both points must be the same but in this case there is a small error around 0.5 pixels; the error in every reprojected corner for all the used images for camera calibration is shown in Fig. 4.4a and the error only for image 2 is shown in Fig. 4.4b. Observe that the x and y axis are in pixels.

Fig. 4.4a shows clearly maximal error values, these are around 1.4 pixels on x-axis and 1.2 pixels in y-axis. The error is apparently not too much but next paragraphs are going to analyse the effect in the obstacle detection system.



(a) Reprojection error for all images. (b) Reprojection error for image 2

Figure 4.4: Chessboard pattern in image 2 after camera calibration

After calibration the intrinsic parameters with their respective error are the following:
Focal length (pixels):

$$fc = \begin{bmatrix} 646.1442 \\ 656.1083 \end{bmatrix}$$

Principal point (pixels):

$$cc = \begin{bmatrix} 319.8762 \\ 240.4569 \end{bmatrix}$$

Skew coefficient (degrees):

$$alpha\_c = 0.0000$$

Distortion coefficients:

$$kc = \begin{bmatrix} -0.4441 \\ 0.2135 \\ 0.0018 \\ -0.0014 \\ 0.0000 \end{bmatrix}$$

Focal length uncertainty (pixels):

$$fc\_error = \begin{bmatrix} 1.6381 \\ 1.5401 \end{bmatrix}$$

Principal point uncertainty (pixels):

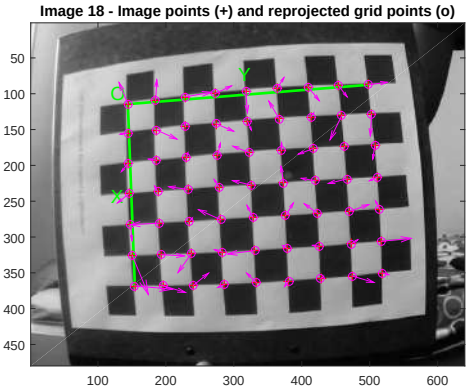
$$cc\_error = \begin{bmatrix} 3.0139 \\ 2.3872 \end{bmatrix}$$

Distortion coefficients uncertainty:

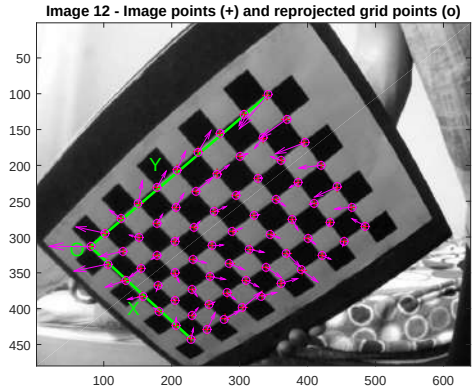
$$kc = \begin{bmatrix} 0.0122 \\ 0.0589 \\ 0.0007 \\ 0.0007 \\ 0.0000 \end{bmatrix}$$

Image size (pixels):

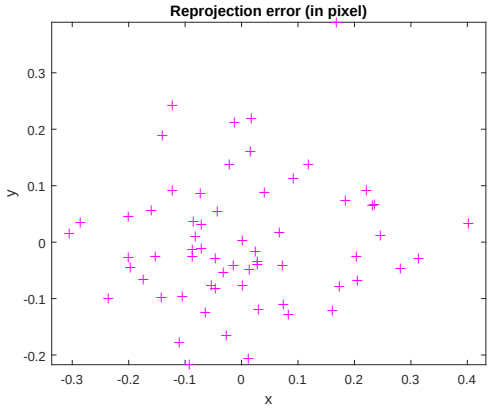
$$n_x = 640, \qquad n_y = 480$$



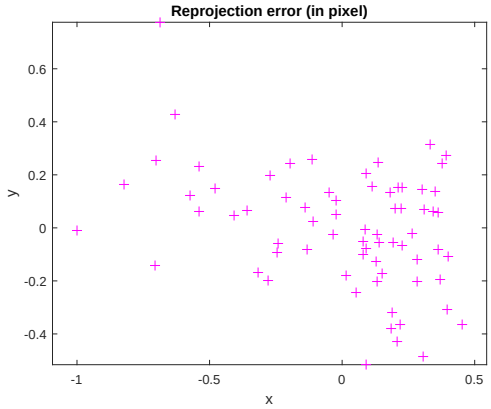
(a) Image 18 with reprojected corners points.



(b) Image 12 with reprojected corners points.



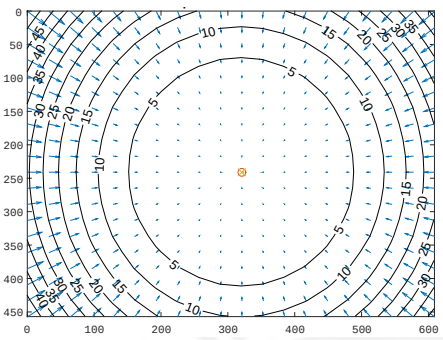
(c) Reprojection error for image 18



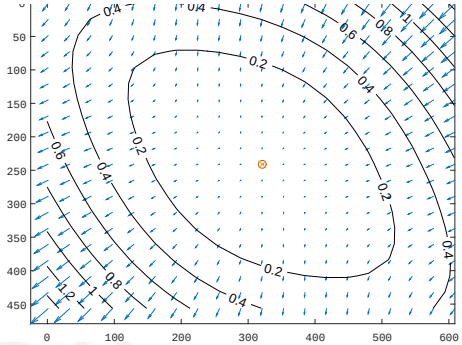
(d) Reprojection error for image 12

Figure 4.5: Distortion and reprojection error.

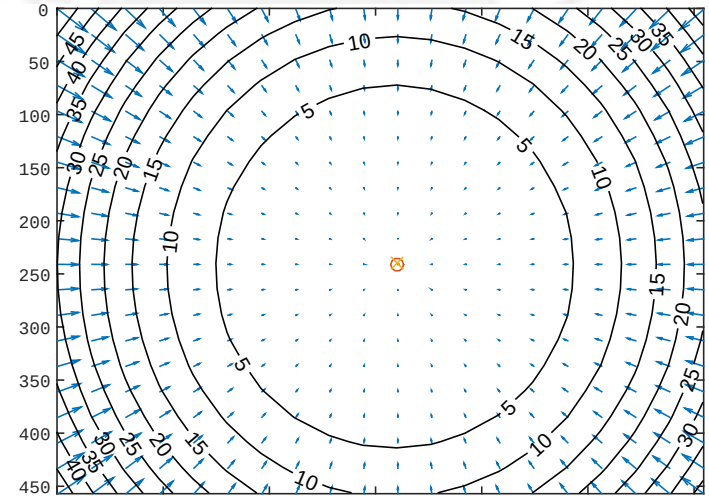
All the intrinsic parameters are explained in Chapter 2. Observe that most of them are in pixels, in Chapter 3 the mathematical model purpose is to estimate the 3D position of the obstacle and it is easy to see that all parameters in the model must be in meters, thus the equation (3.1) performs the conversion of pixels to meters taking into account the size of the pixel defined in the characteristics of the camera.



(a) Radial component.



(b) Tangential component.



```
Pixel error                    = [0.2737, 0.2422]
Focal Length                   = (646.144, 656.108)      +/- [1.638, 1.5
Principal Point                = (319.876, 240.457)      +/- [3.014, 2.38
Skew                           = 0                       +/- 0
Radial coefficients            = (-0.4441, 0.2135, 0)  +/- [0.0122, 0.05887,
Tangential coefficients  = (0.001823, -0.001374) +/- [0.0007237, 0.00074
```

(c) Complete distortion model.

Figure 4.6: Distortion model.

Note that $\tilde{x}$ and $\tilde{y}$ are defined in equation (3.2), where f_x and f_y must be expressed in meters, among the parameters fc is expressed in pixels and the relation between f_x, f_y and fc is:

$$f_x = fc(1)dx, \qquad f_y = fc(2)dy$$

with $dx = dy = 1.4\ \mu m$, finally

$$f_x = 904.6019\ \mu m, \qquad f_y = 918.5516 \mu m$$

Note that f_x and f_y values are almost the same because of the skew coefficient (angle between x and y pixel axes) is 90 degrees, this means that pixels are square.
Distortion was defined in equation (2.5) and expressed with the new notation obtained from Matlab radial and tangential distortion is defined by

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \begin{bmatrix} \delta\tilde{x}^{(rad)} \\ \delta\tilde{y}^{(rad)} \end{bmatrix} + \begin{bmatrix} \delta\tilde{x}^{(tan)} \\ \delta\tilde{y}^{(tan)} \end{bmatrix}$$

$$\begin{bmatrix} \delta\tilde{x}^{(rad)} \\ \delta\tilde{y}^{(rad)} \end{bmatrix} = \begin{bmatrix} \tilde{x}\left(1 + kc(1)r^2 + kc(2)r^4 + kc(5)r^6\right) \\ \tilde{y}\left(1 + kc(1)r^2 + kc(2)r^4 + kc(5)r^6\right) \end{bmatrix} \qquad \begin{bmatrix} \delta\tilde{x}^{(tan)} \\ \delta\tilde{y}^{(tan)} \end{bmatrix} = \begin{bmatrix} 2kc(3)\tilde{x}\tilde{y} + kc(2)(r^2 + 2\tilde{x}^2) \\ kc(3)(r^2 + 2\tilde{y}^2) + 2kc(4)\tilde{x}\tilde{y} \end{bmatrix}$$

Distortion in all images can be easily noted. Squares far of the image centre in the checkerboard are not strictly squares, this can be appreciated in the Fig. 4.5a where the image was captured with the camera parallel to the checkerboard and error arrows are bigger in corners far of the centre. The effects of distortion with a different camera orientation increment the error as it is shown in Fig. 4.5b where the checkerboard is not parallel to the camera, in this case the error arrows also are larger at the edges of the image but corners near to the centre have small errors too.
In order to appreciate in a better way the effects of distortions on the pixels, Figures 4.6a and 4.6b show how this two types of distortion affect the pixels in the image. In addition, Fig. 4.6c shows the total effect of the distortion, the concentric circles indicate the pixel displacement because of the distortion. In general, the distortion increases when more distant is the pixel from the image center, this effect was previously corroborated in Fig. 4.5a and Fig. 4.5b.

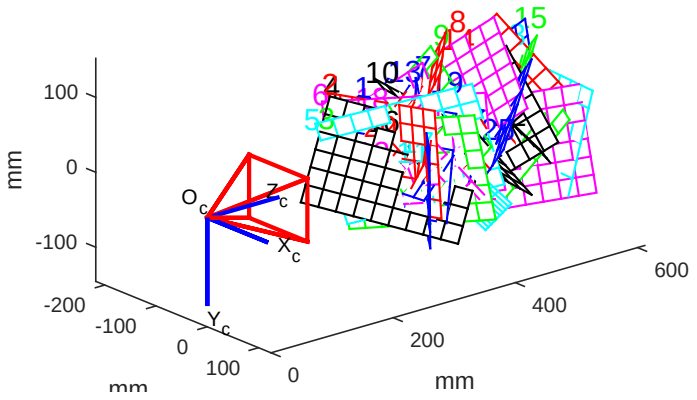
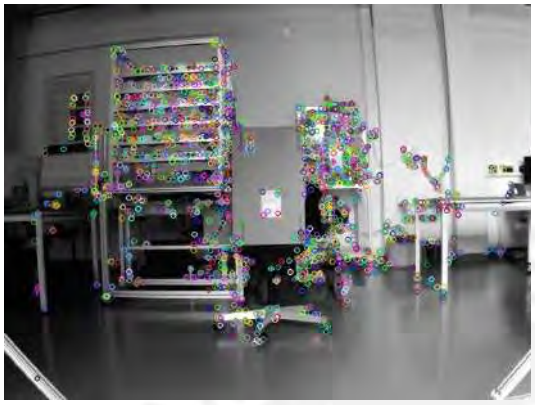


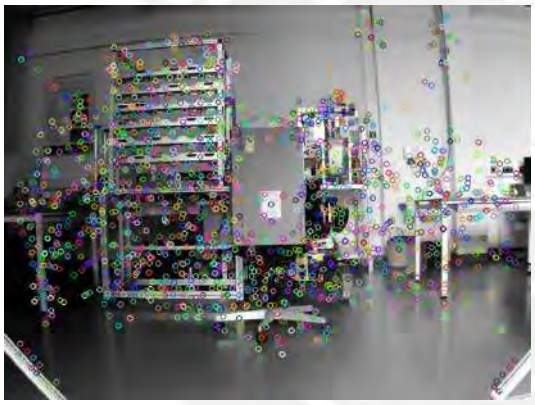
Figure 4.7: Extrinsic parameters.

Finally, Fig. 4.7 shows the 3D positions of the checkerboard with respect to the camera, this figure is a graphic representation of extrinsic parameters of every checkerboard image. The frame (O_c, X_c, Y_c, Z_c) is the camera reference frame. The red pyramid represent the effective field of view of the camera defined by the image plane. All the axis are in millimetres defined by the dimension of the checkerboard.



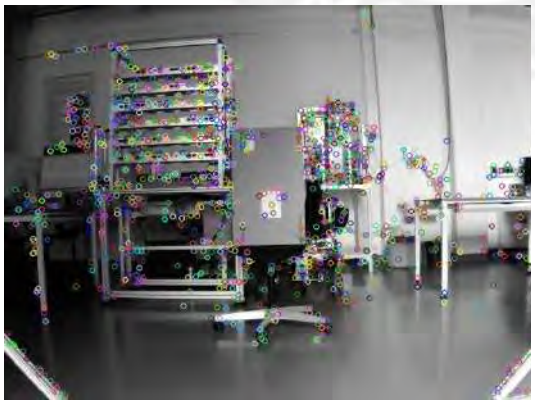
(a) BRISK keypoints.

(b) BRISK keypoints with its radius.



(c) SURF keypoints.

(d) SURF keypoints with its radius.



(e) SIFT keypoints.

(f) SIFT keypoints with its radius.

Figure 4.8: Feature keypoints representation.

4.3 Feature detection and matching

There are many algorithms for feature detection and feature matching. For feature keypoint detection and descriptor computation the available algorithms in OpenCV are: SIFT, SURF and BRISK. The main objective of this section is to compare these algorithms to recognize the one that can find best keypoints for matching and then apply the algorithm 1. Feature keypoints are basically defined by a (x, y) coordinate in the image plane and a scalar which indicates the circular radius to define a sector in the image, the keypoint radius is an important parameter since the larger the radius, the better the obstacle detection, then is is necessary to compare keypoints radius for each possible algorithm. Feature descriptor and matching distance output must be also compared, this parameter indicates how similar the matches descriptors are, the smaller distance matching, the better the matching of descriptors. In other words, feature keypoints define a sector in the image, a sector which can be represented by vectors and then followed along the image.
First, the number of feature keypoints and the computing time are going to be compared. Images of 640x480 pixels and 320x240 will be used, the obstacle is a simple box and there are many objects in the scene to detect as many keypoints as possible. The processing is performed in an Intel i5 at 2.50 GHz CPU. Secondly, results and output images will be explained.

Table 4.1: Feature detection comparison for 640x480 resolution.

	Image 1		**Image 2**		**Image 3**	
	#features	**time (sec)**	**#featuers**	**time(sec)**	**#features**	**time (sec)**
BRISK	2153	1.534	2352	0.681	2370	0.678
SURF	1869	3.267	2019	3.46	2070	3.365
SIFT	1387	1.428	1378	1.436	1476	1.487

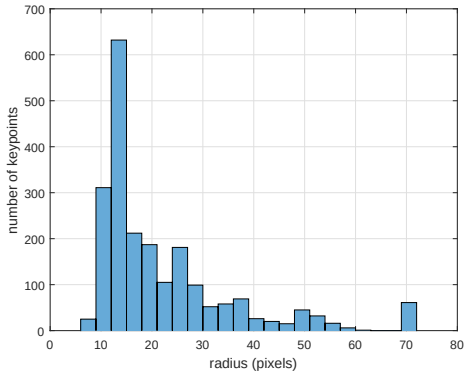
The processed images and detected feature keypoints are shown in Fig. 4.8, left images shows all the detected keypoints while right images shows some keypoints as circles (the radius of these circles are the size of the keypoint, and are greater than 10 pixels). The results are shown in the Table 4.1, using three 640x480 consecutive images captured every 0.5m with a box 2.93m far as the obstacle. All the detected keypoints are marked by small multicolour circles; at first sight the results are very similar but in general BRISK algorithm detects a greater number of keypoints in less time, having more detected keypoints is an advantage; then, this algorithm is the best option. Although there is another parameter which is more important: the size that is defined by the keypoint radius.
As it is known SIFT is the original scale-invariant feature detection algorithm, SURF is a simplified version of SIFT, and finally BRISK is a more simplified algorithm. This means

that computing time for BRISK algorithm is less than SURF and SIFT. In fact, for off-line implementation, very little computing time is not required since in the experiment only the estimation distance algorithm is tested but for future works, using an embedded system this result will be important. In this way, there is no defined maximum time between the two captured images but depending on the speed of flight and image resolution this can be reduced.

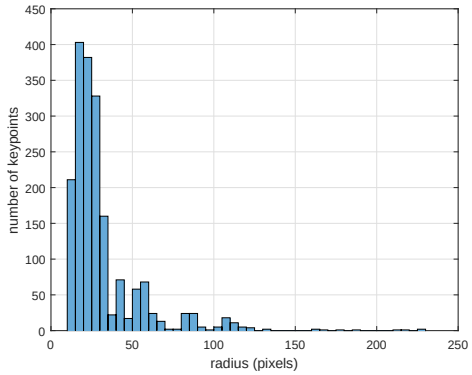
Table 4.2: Feature detection comparison for 320x240 resolution.

	Image 1		**Image 2**		**Image 3**	
	#features	**time (sec)**	**#featuers**	**time (sec)**	**#features**	**time (sec)**
BRISK	1011	1.430	1069	0.595	1048	0.623
SURF	624	0.782	646	0.822	658	0.781
SIFT	541	0.434	577	0.441	620	0.442

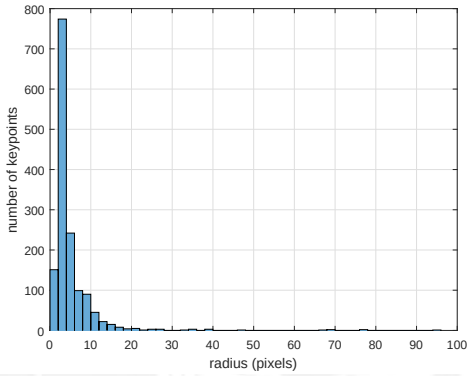
Table 4.1 shows that BRISK is the fastest algorithm, but at the same time the amount of detected feature keypoints is higher. With the first image the computing time is greater than one second. In general, computing times are directly proportional to the amount of feature keypoints, except with the BRISK algorithm. For on-line applications the computing time in Table 4.1 are too long, one way to reduce computing time is to reduce the amount of keypoints and this can be reach it with images of low resolution or clean background in the scene. Table 4.2 shows how much the computing time is reduced with 320x240 images.
It is possible to appreciate in Fig. 4.8f that SIFT algorithm tends to detect feature keypoints with small radius, while SURF, in Fig. 4.8d, and BRISK, in Fig. 4.8b, find keypoints with large radius. For obstacle detection, it is desirable to detect a greater amount of keypoints with radius as long as it is possible. These three algorithm are based in sectors detection and second derivative of the image or its approximation. This is the main reason that all the keypoints are located close to the edges where there is a great change in the pixel value. There are too many small size keypoints, mainly along straight edges, these keypoints are too similar between them so they should be filtered. Finally, it is important to clarify that images are captured with color and JPG format but all the feature detection algorithm are executed with grayscale images, then a JPG to grayscale conversion is applied but results are shown in the color image.



(a) BRISK



(b) SURF



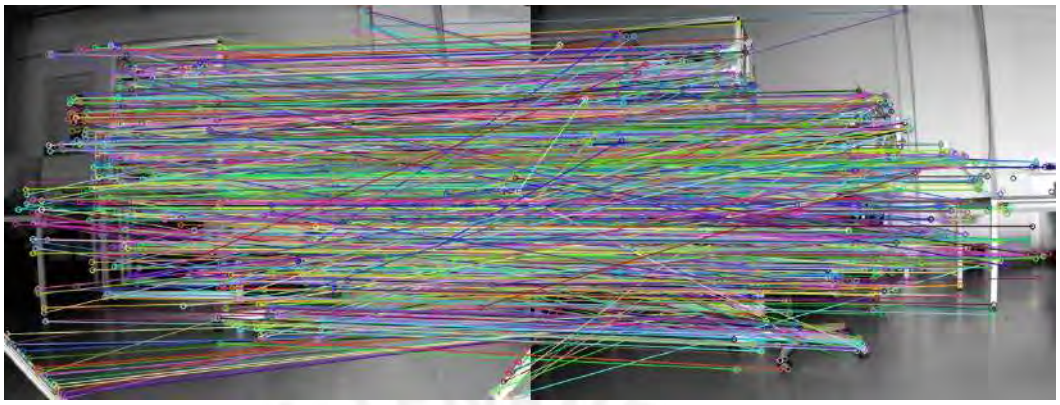
(c) SIFT

Figure 4.9: Feature keypoints radius histograms.

With last results, all algorithms are good candidates for testing the feature matching and distance estimation since many keypoints are detected on the obstacle, although SIFT algorithm detects less amount of keypoints (Table 4.1). Now another parameter will be analysed, Figures 4.9 plot keypoints radius represented as a histogram. BRISK and SURF keypoints histograms are similar, but SURF algorithm is better because most of its keypoints have radius between 10 and 50 pixels, while BRISK radius are between 10 and 30 pixels and SIFT radius are concentrated between 1 and 10 pixels. The keypoint size is important to recognize obstacles or details in the obstacle, the keypoint radius is proportional to the detected obstacle size. Therefore, because of the amount of keypoints, too many detected keypoints, see 4.1, and keypoints radius, keypoint radius between 10 and 30 pixels, SURF algorithm is the best choice; but in simulations, in the next section, to test the developed distance estimation, SIFT algorithm will be applied because of test obstacles has small intentional details to obtain keypoints in different sectors of the image.

OpenCV has two matching algorithms [OpenCV, 2017b]: BF algorithm which takes the descriptor of one feature in the first set and compute de distance with every descriptor of the second set, the closest distance are returned and the Fast Library for Approximate Nearest

Neighbors (FLANN) which contains a collection of algorithms optimized for fast nearest neighbour search in large datasets.



(a) Matching with all keypoints.

(b) Best matched keypoints.

Figure 4.10: Feature matching filtering.

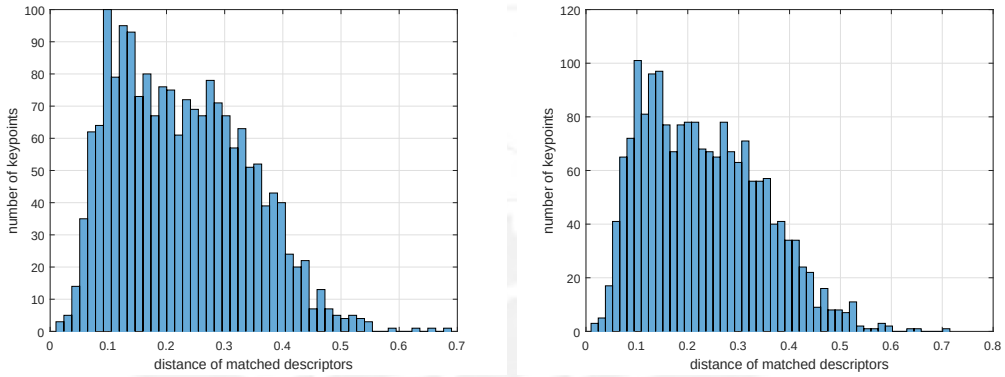
Figure 4.10a shows how matching algorithms are represented between two images with all the detected keypoints, matched keypoints are connected with color lines, appreciate that some descriptors are incorrectly matched since lines which connect matched keypoints are not all parallel. In order to obtain a new set of best matched keypoints and apply distance estimation, incorrect matched keypoints must be filtered. BF and FLANN has a *matching distance* as output parameter; this parameter indicates how close the descriptors of the two test images are and the algorithm 2 is implemented to get all the best matched descriptors, results are shown in Fig. 4.10b.

The K parameter in algorithm 2 is defined to obtain the best matched keypoints that can describe the environment. Fig. 4.11 shows a histogram of matching distances with SURF descriptors, note that the matching distance is normalized between 0 and 1, most of them are small, less than 0.4 but the difference between BF and FLANN matching algorithms is imperceptible. With $K = 3$, the amount of good matched descriptors are enough.

Algorithm 2 Matched descriptors filtering.

Require: Descriptors sets $descriptor_{01}$ and $descriptor_{02}$.

Set the minimum distance with a large value: $dist_{min} = 500$.
Match $descriptor_{01}$ and $descriptor_{02}$ to obtain $matching\_distance$.
Obtain size of $descriptor_{01}$: $descrip01_{size}$.
for $i = 0$ to $descrip01_{size}$ **do** # determine the minimum distance value
  if $matching\_distance < dist_{min}$ **then**
    $dist_{min} = matching\_distance$
  end if
end for
for $i = 0$ to $descrip01_{size}$ **do** # determine the best matched descriptors
  if $matching\_distance < Kdist_{min}$ **then**
    $descriptor_{01}$ and $descriptor_{02}$ are good matches.
    Store amount of good matches: $mathes_{num}$.
  end if
end for

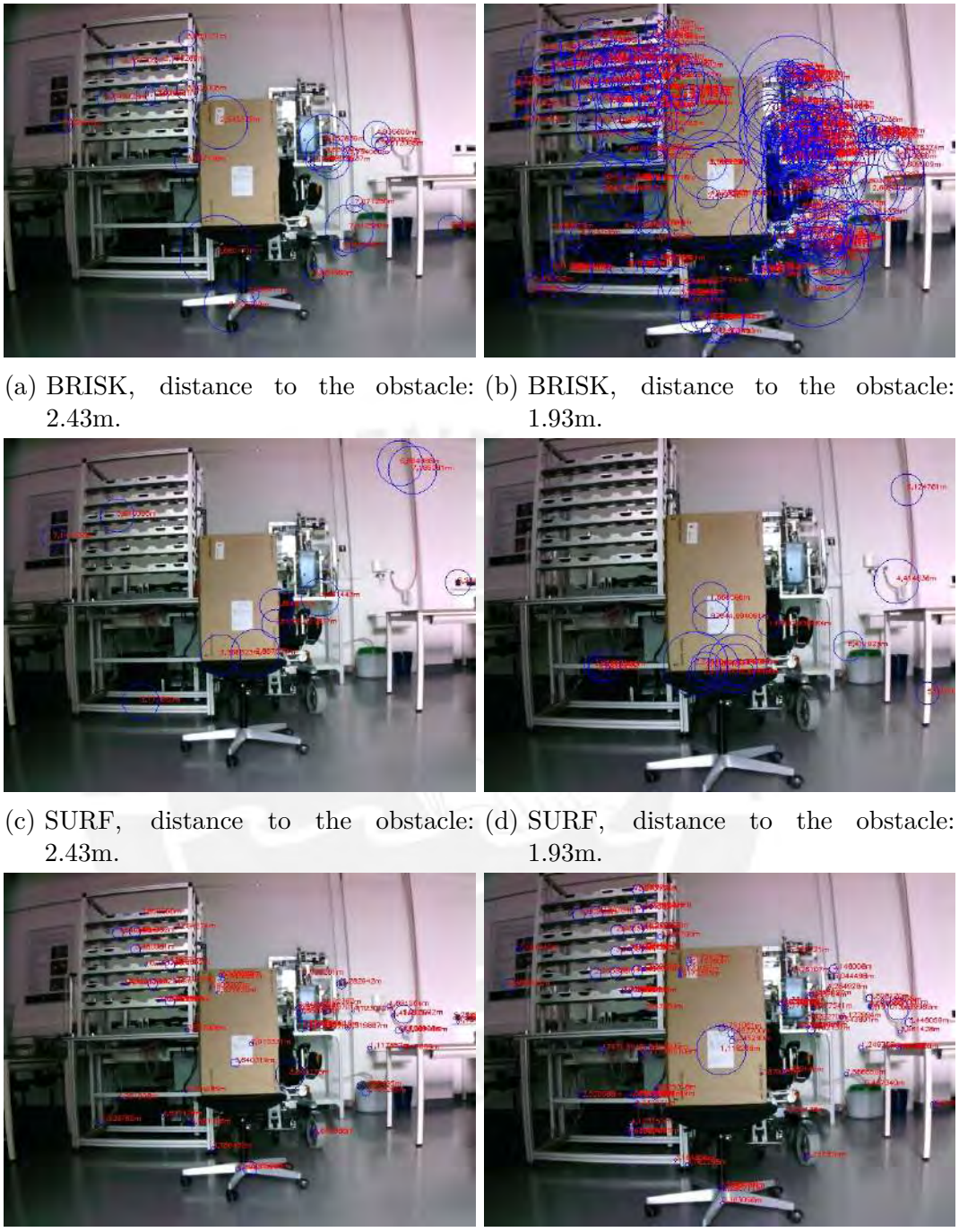


(a) BF (b) FLANN

Figure 4.11: Distance matching comparison with SURF descriptors.

4.4 Obstacle distance estimation

Once good matched candidates for distance estimation are filtered, the mathematical model for distance estimation must be applied and solved by algorithm 3. There are many parameters and methods that can be chosen to improve results: internal variables like feature detection algorithms, descriptor computation algorithms, minimum distance in the filter, matching algorithms; and external possibilities as the obstacle form, obstacle real distance, background in the scene, the distance travelled by the camera to capture a new image, camera properties, accuracy of position and orientation data. Best conditions were analysed in the last section.



(a) BRISK, distance to the obstacle: 2.43m.
(b) BRISK, distance to the obstacle: 1.93m.
(c) SURF, distance to the obstacle: 2.43m.
(d) SURF, distance to the obstacle: 1.93m.
(e) SIFT, distance to the obstacle: 2.43m.
(f) SIFT, distance to the obstacle: 1.93m.

Figure 4.12: Distance estimation with different feature detection algorithm (distance to the obstacle 2.93m and images captured every 0.5 m).

In this section, the main objective is to evaluate some initial results of experiments to verify the best conditions to estimate distances between obstacle and camera. With filtered matched descriptors the three algorithms seen before will be applied, the error estimation will be compute and the step distance between captured images will be change. Finally, setting the

Algorithm 3 LU factorization.

Require: Linear system to solve: $Ax = b$.

1: Get matrices L and U where $A = LU$.
2: Set $y = Ux$.
3: Use forward substitution to solve for y from $Ly = b$.
4: Use back substitution to solve for x from $Ux = y$.

best conditions and with some conclusion about the operation of the distance estimation algorithm, captured images with a large distance to the obstacle will be tested.

Fig. 4.12a shows only one keypoint on the obstacle. Only one keypoint detected on the obstacle is not too much information in order to analyse keypoints positions and estimated distances. However, the estimated distance in this single keypoint is 2.59m which is very close to the real distance which must be greater than 2.43m, note that obstacle distance was measured from the camera to the obstacle center, then distances far from the obstacle center must be larger than 2.43 m. There is no information about real distance of background objects but Fig. 4.12a shows certain degree of depth perception since detected keypoints out of the obstacle have larger estimated distances, this information can be used for path-planning tasks.

Fig. 4.12b doesn't show too much information since there are too many detected keypoints with large sizes. However, estimated distances in the center greater than 2 m can be appreciated. In fact, center estimated distances with values 2.4, 2.5 and 2.1 m have a significant error respect to the real distance 1.93 m. Non-legible distances on the obstacle will be analyse using other types of graphs.

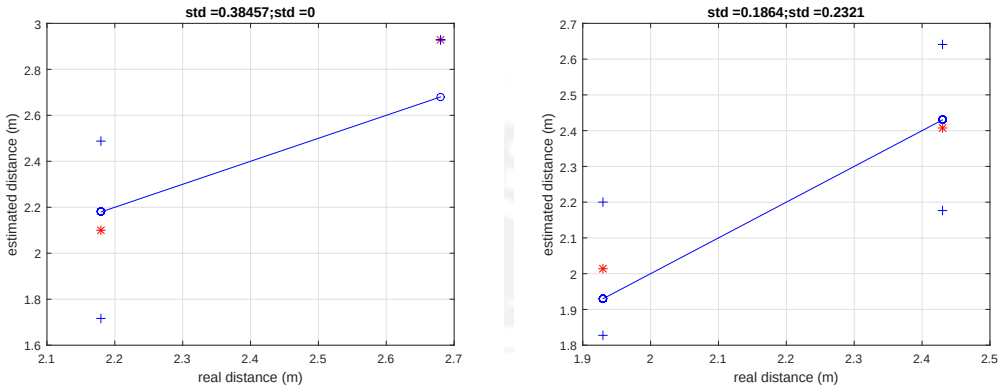
Figures 4.12c and 4.12d shows that SURF algorithm tends to detect large size keypoints, this is good since this obstacle does not have too much details to recognise. Detected sectors are very close to the edges, this is another negative result because their descriptors can be matched wrongly.

Finally, Figures 4.12e and 4.12f show that SIFT algorithm detects more details in the obstacle since there are matches keypoints in small marks in the box. Detect more details is a positive characteristic since more feature keypoints can be matched, another advantage respect BRISK and SURF is that feature keypoints are not close to the box edges so more keypints are good matched.

Next step is to analyse the effect of distance between captured images, the SIFT algorithm will be applied in order to obtain a larger amount of feature keypoints on the obstacle. Figures 4.14a and 4.14b show the estimated obstacle distance with the camera moving with steps of 25 cm, the obstacle is to 2.68 m and 2.18 m respectively but in Fig. 4.14a the points marked in the image indicates estimated distances with too much errors (around 30 cm). In Fig. 4.14b estimated distance are closer to the real one. Therefore, estimated distance are more accurate when the obstacle is closer to the camera in points far from the

image center.

Detected features in figures 4.14c and 4.14d show a similar behaviour to the results previously analysed but if the estimated distances are compared with the previously one, these last results are more accurate, this observation can be verified in 4.13a and 4.13b where the standard deviation is greater with steps of 0.25 m. Note that feature keypoints near the image center always have large estimation errors, this phenomenon happens because keypoint displacement is small in the center.

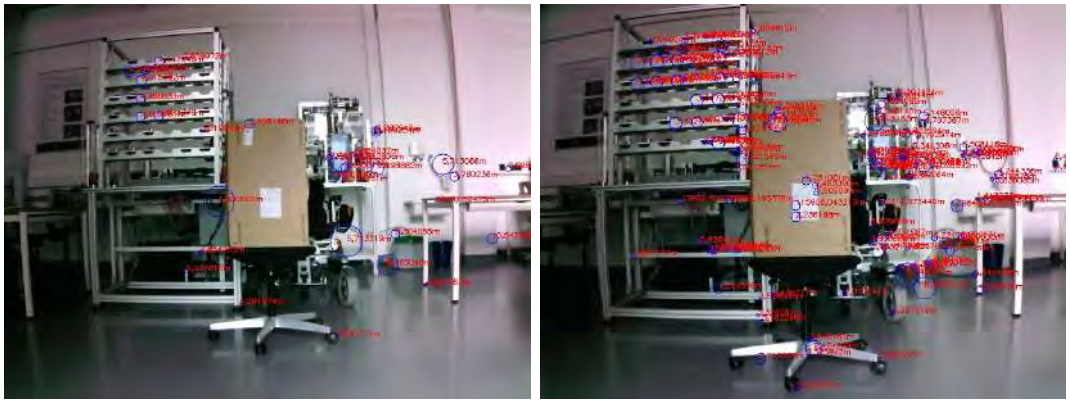


(a) Step distance of 0.25 m. (b) Step distance of 0.50 m.

Figure 4.13: Step distance comparison.

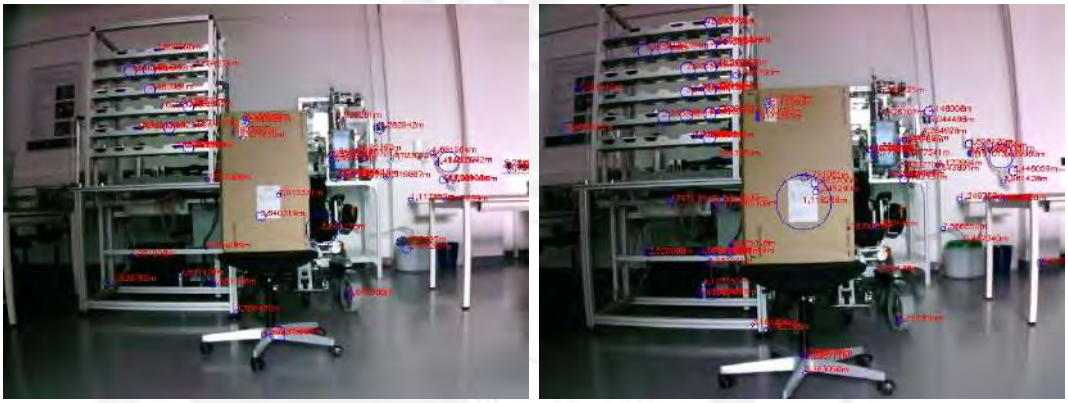
It is important to note that:

- Estimated distances are too variable, seems like the error is random but actually there are many factors that affects the error.
- When the feature keypoint is near to the image center the estimated distance increases, note that the error increases too.
- When the camera moves with steps of 50 cm the error is smaller than when the step is 25 cm.
- Proximity of the obstacle affects the estimated distance error.
- Feature keypoint location affects the estimated distance, the estimation error is variable from the center to the edge in the image.
- Obstacle appearance, image resolution and feature detection algorithm define the quantity of detected feature keypoints.



(a) SIFT, distance to the obstacle: 2.68 m.

(b) SIFT, distance to the obstacle: 2.18 m.



(c) SIFT, distance to the obstacle: 2.43 m.

(d) SIFT, distance to the obstacle: 1.93 m.

Figure 4.14: Distance estimation with different feature detection algorithm, distance to the obstacle 2.93 m, images captured every 25 cm (a and b) and 50 cm (c and d).

4.4.1 Error analysis in distance estimation

Obstacle images seen before was captured along a 1.13 m path. In order to analyse results and setting the best condition, now the obstacle is placed 5m far and the image capture step is 50cm. There are too many factors that introduce error in estimated distances; in this subsection some tables and graphs are going to show the error behaviour and distance estimation. Easily recognizable characters were placed on the obstacle and SIFT algorithm for feature detection is used to control in a better way the keypoints and analyse them later. Table 4.3 shows the matched keypoints in the first image $xi1$, $yi1$, matched keypoints in the second image $xi2$, $yi2$ and the distance estimation.

The estimation is basically based on the keypoint movement, the accuracy depends on how much the keypoint is moving. If the keypoint changes only few pixels the accuracy is not so good due to the error in camera calibration is ± 1 pixels, this error and total distortion are

the main reason why many results are far from the reference value.

Table 4.3: Distance estimation.

Distance = 4.5 m				
xi1	**yi1**	**xi2**	**yi2**	**estimation**
341	235	344	239	0.6250
Distance = 4.0 m				
xi1	**yi1**	**xi2**	**yi2**	**estimation**
342	273	345	277	4.1250
Distance = 3.5 m				
xi1	**yi1**	**xi2**	**yi2**	**estimation**
348	197	352	192	4.3000
348	205	351	201	4.3750
302	210	299	206	3.7500
299	280	295	286	3.3333
304	274	301	279	3.4000
295	279	291	285	3.2500
299	275	295	281	2.9167
Distance = 3.0 m				
xi1	**yi1**	**xi2**	**yi2**	**estimation**
295	286	292	292	3.8333
295	281	292	287	3.4167
Distance = 2.5 m				
xi1	**yi1**	**xi2**	**yi2**	**estimation**
303	218	299	209	1.2222
291	192	284	178	1.7143
296	199	290	187	1.7084
299	284	294	288	5.5000
Distance = 2.0 m				
xi1	**yi1**	**xi2**	**yi2**	**estimation**
358	181	366	166	1.9667
299	209	291	201	1.9375
290	292	280	305	2.0000
284	178	273	162	1.9375
302	260	296	266	1.6667
312	267	308	275	1.6875
297	197	289	186	1.9545

Distance = 1.5 m				
xi1	**yi1**	**xi2**	**yi2**	**estimation**
292	182	278	154	1.4357
370	169	381	136	1.4758
291	201	277	179	1.5864
288	180	273	151	1.4345
308	275	300	278	1.8333
368	308	381	322	1.4286
289	186	274	160	1.4385
351	208	358	188	1.5000
Distance = 1.0 m				
xi1	**yi1**	**xi2**	**yi2**	**estimation**
278	155	262	102	0.9019
298	272	290	281	1.1778
273	151	254	97	0.9241
285	159	272	108	0.9941
350	181	372	141	0.9375
300	278	294	289	1.1273
273	171	254	125	0.9500
367	276	397	288	1.1500
274	160	256	109	0.9843
Distance = 0.5 m				
xi1	**yi1**	**xi2**	**yi2**	**estimation**
218	30	263	351	0.3271
357	160	397	61	0.4040
291	299	262	340	0.7195
378	136	433	18	0.4407
290	281	260	303	0.9318
273	282	225	305	0.9130
294	289	268	320	0.7903
372	166	425	72	0.3936

Fig. 4.15 is the image result after obstacle recognition and distance estimation when the camera is 2 m far from the obstacle. In the experiment the camera was parallel to the obstacle and just between the letters *e* an *g*, then this is the image plane center. Something important to note is that the error increases as the keypoint is moving away from the image plane center, this is because the displacement of the keypoints is greater while they are

further from the center. The keypoints displacement also increases as the camera is closer to the obstacle. Therefore, the distance to the obstacle and the keypoint position in the image plane affect the distance estimation.

Figure 4.15: Distance estimation result for obstacle of 2m distance.

Fig. 4.16 shows the mean value in red asterisks and the standard deviation in blue crosses for every estimated value. The graph shows an irregular behaviour with long distances, as the last paragraph said the error depends on the distance to the obstacle. Error is always present, the distortion, camera calibration error and variations in the keypoints displacement are the responsible. After camera calibration distortion parameters also was calculated and these parameters can be used to correct the keypoints location and, finally, compute another estimated distance value.

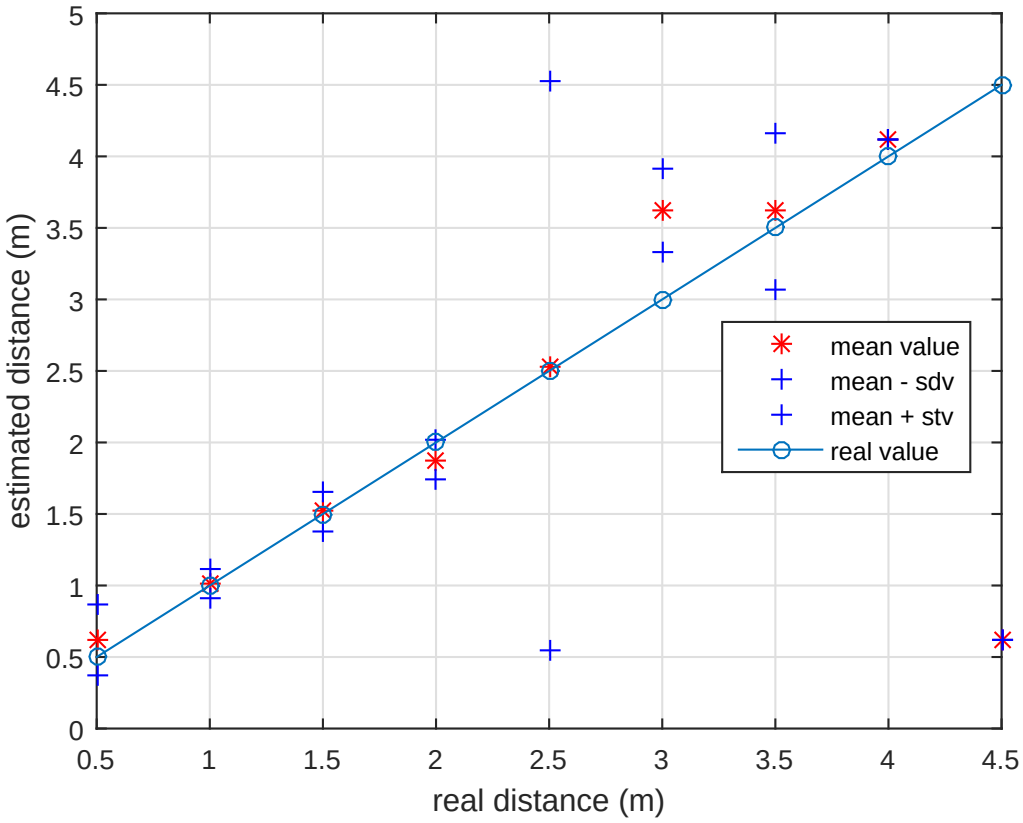
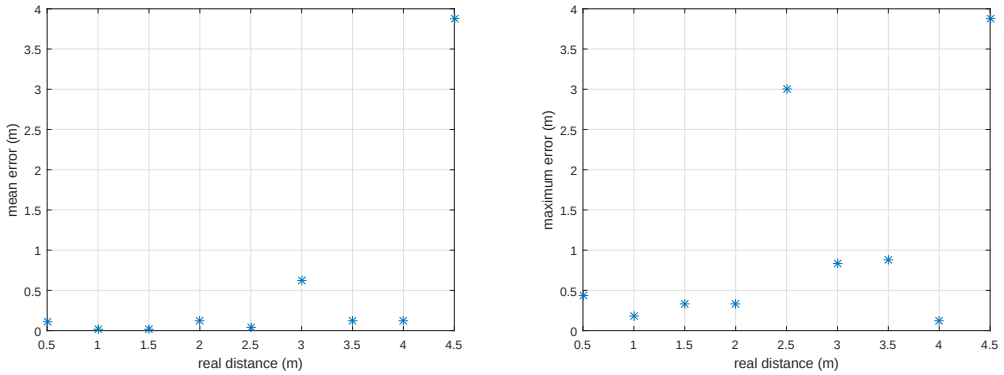


Figure 4.16: Estimated distance vs. real distance.

Graphs in Fig. 4.17 show the mean and maximum error value in every step of the path. Both graphs indicate that error increase with obstacle distance, it is not linear because many other factors that can affect the final error like pixel position, pixel displacement and camera parameters error. Because of the error variation only the mean error value will be analysed. A great mean error is present at 3 m obstacle distance with 20 % while with small distances the error is 20 %, 1.63 %, 1.1 %, 6 % and 1 % for 0.5, 1.0, 1.5, 2.0 and 2.5 m respectively. This demonstrate that estimation is less accurate with great distances.



(a) Step distance of 0.25 m. (b) Step distance of 0.50 m.

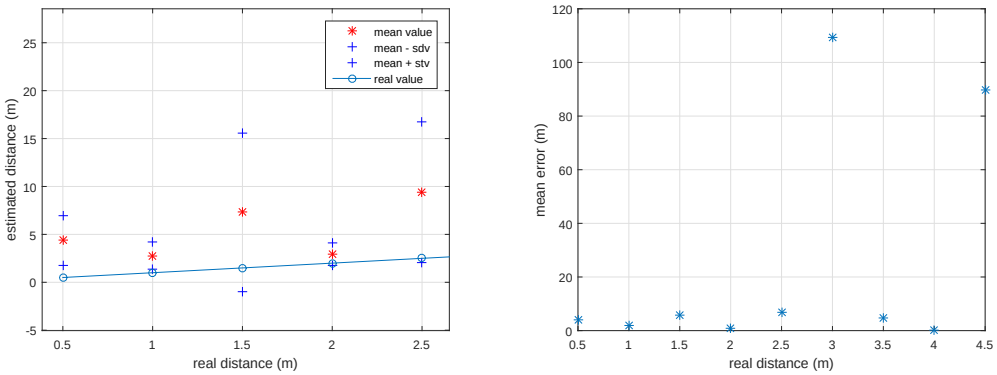
Figure 4.17: Estimation error.

4.4.2 Distance estimation with real time localization system.

Table 4.4 shows mean position and standard deviation recording using USB localization modules. Standard deviation values are between 1.3 cm and 14.3 cm, some measurements have too much variation, this affects estimation as Fig. 4.18 shows with bad estimations and large error.

Table 4.4: Feature detection comparison for 320x240 resolution.

x_{mean}	x_{std}	y_{mean}	y_{std}	z_{mean}	z_{std}
1.5384	0.0789	1.0904	0.0640	0.4976	0.1437
1.5948	0.0222	1.7540	0.0227	0.0938	0.0636
1.5986	0.0334	2.2954	0.0200	0.0916	0.0767
1.5718	0.0327	2.7156	0.0241	0.1362	0.0712
1.7638	0.0463	3.0258	0.0192	0.7090	0.0870
1.7204	0.0290	3.5558	0.0151	0.6384	0.0651
1.6844	0.0258	4.0976	0.0209	0.6578	0.0524
1.6330	0.0256	4.5950	0.0136	0.5308	0.0477
1.6426	0.0837	5.0666	0.0219	0.5938	0.1516
1.7258	0.0811	5.6174	0.0172	0.8518	0.1430



(a) Estimated distance vs. real distance. (b) Mean error.

Figure 4.18: Estimation with real time localization system.

This graphs corroborate that position measurements are very important for distance estimation, localization systems must be as accurate as it is possible.

5 Path-planning

The algorithm 1 in a general way estimates the distance for all keypoints in the image, thus keypoints in the backgroung are also recognized and their distances are estimated. This characteristic can be observed only when there are many keypoints in the background; for this reason, captured images that have many obstacles in background are used. The distance estimation along the whole image can represent a depth mapping of the environment. Fig. 5.1 shows a box as the obstacle and many detected keypoints that can be the input for computing the estimated distance.

The depth mapping of the environment can be transformed into a Ego Dynamic Space, this is a spacial representation where all estimated distances are transformed into effective distances considering the quadrocopter dynamic constraints. The quadrocopter in the lab does not have a fine control, its trajectory is not stable while it is flying, because of this the path-planning will not be implemented, but this section will describe a method according to the results obtained from the obstacle detection stage and estimation of the distance of this obstacle.

Figure 5.1: Detected keypoints in background.

5.1 The Ego Dynamic Space Transformation

The Ego Dynamic Space Transformation (EDS) [Bipin et al., 2015] allow to build a spacial representation where the distances to the obstacles re transformed into distances that depend on the quadrocopter deceleration constraint and response time of the system. Considering the dynamic constrains in the quadrocopter the trajectory planning is not susceptible to collision. EDS takes quadrocopter dynamics into account to ensure feasible motion, equations (5.1), (5.2) and (5.3) describe the transformation.

$$d_{obs} = D_E - avg(e^{E_{ln}}) \qquad d_{obs} = d_{eff} + d_{brake} \tag{5.1}$$

$$\begin{aligned} d_{eff} &= v\Delta t \\ d_{brake} &= \frac{v^2}{2a_b} \end{aligned} \tag{5.2}$$

$$\frac{d_{eff}^2}{2\Delta t a_b} + d_{eff} - d_{obs} = 0 \tag{5.3}$$

Where D_E is the estimated distance to the obstacle, $avg(e^{E_ln})$ represents the average error in the estimation and d_{obs} takes in account the error too. d_{eff} is the effective distance, the maximum distance that the quadrocopter can fly at a constant velocity v during the period Δt, and d_{brake} is the minimum distance to stop the quadrocopter safely before hitting an obstacle applying the maximum deceleration a_b. If $d_{eff} \leq 0$ indicates imminent collision thus the EDS transformation of d_{obs}:

$$EDS\{d_{obs}\} \quad \longrightarrow \quad a_b\Delta t^2 \left(\sqrt{1 + \frac{2d_{obs}}{a_b\Delta t^2}} - 1 \right) \tag{5.4}$$

5.2 Optimal direction

Each value in $EDT\{d_{obs}\}$ correspond to one detected keypoint and if the effective distance is long enough, these keypoints could be a probable next way-point for the quadrocopter. Each keypoint is defined by $S_i, \forall i = 1, ..., M$. SURF algorithm is optimal for detecting all possible way-points because of the kind of keypoints that this algorithm detects, each keypoint S_i contains inherent properties: radius (r_i), center (o_i) and distance to the center along the line of sight in X-axis, respect to the camera, (d_i) which is used for determining the optimum next way-point. Fig. 5.2 shows the resulting keypoints in the background with SURF algorithm.

Figure 5.2: Detected keypoints in background.

The process of selection the optimal way-point is formulated as a binary integer programming problem where the objective function define the optimum criteria for way-point selection and ensure a smooth free collision movement, similar to human intuition, which prefers moving towards the nearest and maximum size free space to avoid frontal obstacle, this cost function is defined as:

$$C_i = \frac{\rho_i}{r_i}; \quad \forall i = 1, ..., M \tag{5.5}$$

$$\min_k C^T k \tag{5.6}$$

Where ρ_i is the distance between the background keypoint centre and the image plane centre, r_i is its radius, and C is the coefficient vector of the cost function that describes the optimum way-point selection and k is a binary integer vector of length M with exactly one index allowed value 1 at any given time. Equation 5.6 need to be solved subject to kinematics and physical geometry constraints of quadrocopter.

Quadrocopter dynamics need to be taken into consideration for ensuring collision free tra-

jectory, equation 5.7 represents the equality and inequality constraints by affine functions of k. The equality constraints in 5.7 is represented by A_eq which is a vector of length M with all elements with value 1 and b_{eq} unit value.

$$\begin{aligned} A_{eq}k &= b_{eq} \\ A_{in}k &= b_{in} \end{aligned} \tag{5.7}$$

The quadrocopter requires a minimum free space along the image plane, depending on the shape and motion towards next way-point. The drift and inconsistent need to be taken into account while the traversable space is determined. These conditions add M inequalities, one for each keypoint.

$$\begin{aligned} \eta &= \delta + max(\sigma_y, \sigma_z) \\ \frac{1}{r_i} &\leq \frac{1}{\eta}; \quad \forall i = 1, \ldots, M \end{aligned} \tag{5.8}$$

Where η is the minimum desired threshold radius of segments S_i and δ constant parameter dependent on the shape of quadrocopter. σ_y and σ_z are standard deviation of error in state estimation along the plane. Finally, the entire optimization problem is:

$$\min_k C^T k; \quad subject\ to \quad A_{eq}k = b_{eq};\ A_{in}k \leq b_{in} \tag{5.9}$$

5.3 Path generation

Once the best direction is selected, the next step is to generate a path between the current position and the selected direction, for autonomous systems smooth trajectories are necessary but in the case of this work the obstacle detection (obstacle distance estimation) and obstacle avoidance (path-planning) must run one after the other. For path generation the Bezier polinomial is proposed, there are two points as data: the current position of the quadrocopter and the final point in the image plane, they can be connected by a line but a straight line could be a problem with close obstacles, thus Bezier curves are applied (5.10).

$$q_i(\tau) = \sum_{i=0}^{n} \binom{u}{v} \tau^i (1-\tau)^{n-i} p_i, \quad \forall \tau = 0, \ldots, 1 \tag{5.10}$$

Where $\binom{n}{i}$ are binomial coefficient and $\binom{u}{v}\tau^i(1-\tau)^{n-i}$ are the Bezier/Bernstein basis polynomials, p_i are scalar coefficient called control points and q_i are the points which must be interpolated.

6 Conclusions

- Data from sensor are critical in order to obtain good distance estimations, position data and orientation. This data is used for distance estimation and building a linear system equation expressed with a matrix. Most of the matrix elements depends on the orientation and focal length; then, for a good result matrix elements must be well defined to not generate sparse systems that are difficult to solve.

- For the experiments the orientation was fixed to 90 degrees, but if this value is changed the final estimated result changes too with variations around 2cm; this happens because orientation is part of many elements in the linear system to solve. For on-line implementation sensors must be sensible enough in order to detect small orientation variations because captured images can be affected. In addition, an accurate control system for smooth movement is crucial.

- There are too many algorithms in order to detect feature and match descriptors, but the main difference among them is the time execution and keypoint size. These differences can help to detect small and big obstacle with SIFT and SURF algorithms. Algorithms seen before were based on similar thinking but with simplified mathematical methods. Hardware resources are always a problem since images size increases with advance of technology and experiments shows that time computing with 640x480 images with SURF algorithm is 3.27 seconds and 0.78 seconds with 320x240 images.

- With a small linear system equation it is possible to apply direct factorization methods in order to compute numerically the solution. In order to apply the correct method, it is necessary to recognize the matrix properties. The condition number for the linear system matrices is around 1000 with well-defined matched keypoints, then the LU factorization is enough to solve the linear system.

- The obstacle detection method developed in this work is a general one, since it is based on general feature detection comparing two consecutive images of the same scene. Because of its generality too many parameters can affect the accuracy of results. For good results, camera properties must be known with small errors, that means that camera calibration is an important step. In experiments, the camera calibration step show that error in projection is ± 1 pixel and the total distortion has a initial value of 5

pixels (this value increases when the pixel is moving away from the image center) that finally will generate wrong displacement in detected image features. Graphs shows that error value is less than 5 % with obstacle distances less than 3 m and has a non-linear behaviour with greater distances.

- Because of all images used in experiments were previously captured and stored, image quality cannot be evaluated. Images format are also important since algorithms for feature detection are based on pixels variation and uncompressed images have more information.

Bibliography

Abdulla Al-Kaff, Qinggang Meng, David Martín, Arturo de la Escalera, and José María Armingol. Monocular vision-based obstacle detection/avoidance for unmanned aerial vehicles. In *Intelligent Vehicles Symposium (IV), 2016 IEEE*, pages 92–97. IEEE, 2016.

BM Albaker and NA Rahim. A survey of collision avoidance approaches for unmanned aerial vehicles. In *international conference on technical postgraduates (TECHPOS), 2009*, pages 1–7. IEEE, 2009.

Andor. Rolling and global shutter, 2017. URL `http://www.andor.com/learning-academy/rolling-and-global-shutter-exposure-flexibility`.

Sean P Bemis, Steven Micklethwaite, Darren Turner, Mike R James, Sinan Akciz, Sam T Thiele, and Hasnain Ali Bangash. Ground-based and uav-based photogrammetry: A multi-scale, high-resolution mapping tool for structural geology and paleoseismology. *Journal of Structural Geology*, 69:163–178, 2014.

Universität Berlin. Global computer vision, 2005. URL `http://robocup.mi.fu-berlin.de/buch/chap9/ComputerVision.htm`.

Paul J Besl, Neil D McKay, et al. A method for registration of 3-d shapes. *IEEE Transactions on pattern analysis and machine intelligence*, 14(2):239–256, 1992.

Kumar Bipin, Vishakh Duggal, and K Madhava Krishna. Autonomous navigation of generic monocular quadcopter in natural environment. In *International Conference on Robotics and Automation (ICRA), 2015 IEEE*, pages 1063–1070. IEEE, 2015.

Samir Bouabdallah, Marcelo Becker, Vincent de Perrot, and Roland Yves Siegwart. Toward obstacle avoidance on quadrotors. 2007.

Koray Çelik and Arun K Somani. Monocular vision slam for indoor aerial vehicles. *Journal of electrical and computer engineering*, 2013:4–1573, 2013.

Timothy A Clarke and John G Fryer. The development of camera calibration methods and models. *The Photogrammetric Record*, 16(91):51–66, 1998.

Peter Corke. *Robotics, vision and control: fundamental algorithms in MATLAB*, volume 73. Springer, 2011.

Boston Dynamics. From left: Boston dynamics original atlas, next-generation atlas, bigdog, wildcat, and alphadog., 2017. URL `https://www.bostondynamics.com/`.

Changhong Fu, Adrian Carrio, and Pascual Campoy. Efficient visual odometry and mapping for unmanned aerial vehicle using arm-based stereo vision pre-processing system. In *International Conference on Unmanned Aircraft Systems (ICUAS), 2015*, pages 957–962. IEEE, 2015.

Nils Gageik, Paul Benz, and Sergio Montenegro. Obstacle detection and collision avoidance for a uav with complementary low-cost sensors. *IEEE Access*, 3:599–609, 2015.

Hardkernel. ocam : 5mp usb 3.0 camera, 2017. URL `http://www.hardkernel.com/main/products/prdt_info.php?g_code=G145231889365`.

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

Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

M Hassaballah, Aly Amin Abdelmgeid, and Hammam A Alshazly. Image features detection, description and matching. In *Image Feature Detectors and Descriptors*, pages 11–45. Springer, 2016.

Janne Heikkila and Olli Silven. A four-step camera calibration procedure with implicit image correction. In *Computer Society Conference on Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE*, pages 1106–1112. IEEE, 1997.

Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In *In the 12th International Symposium on Experimental Robotics (ISER)*. Citeseer, 2010.

Joseph Howse, Steven Puttemans, Quan Hua, and Utkarsh Sinha. *OpenCV 3 Blueprints*. Packt Publishing Ltd, 2015.

Rui Huang, Ping Tan, and Ben M Chen. Monocular vision-based autonomous navigation system on a toy quadcopter in unknown environments. In *International Conference on Unmanned Aircraft Systems (ICUAS), 2015*, pages 1260–1269. IEEE, 2015.

Jeong-Oog Lee, Keun-Hwan Lee, Sang-Heon Park, Sung-Gyu Im, and Jungkeun Park. Obstacle avoidance for small uavs using monocular vision. *Aircraft Engineering and Aerospace Technology*, 83(6):397–406, 2011.

Tony Lindeberg and Jonas Gårding. Shape-adapted smoothing in estimation of 3-d shape cues from affine deformations of local 2-d brightness structure. *Image and vision computing*, 15(6):415–434, 1997.

Krystof Litomisky. Consumer rgb-d cameras and their applications. *Rapport technique, University of California*, 20, 2012.

David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference*, volume 2, pages 1150–1157. Ieee, 1999.

Daniel Magree, John G Mooney, and Eric N Johnson. Monocular visual mapping for obstacle avoidance on uavs. In *International Conference on Unmanned Aircraft Systems (ICUAS), 2013*, pages 471–479. IEEE, 2013.

Mathworks. Matlab, 2017. URL `https://uk.mathworks.com/products/matlab.html`.

Tomoyuki Mori and Sebastian Scherer. First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles. In *International conference on robotics and automation (icra)*, pages 1750–1757. IEEE, 2013.

Nikon. Understanding maximum aperture, 2017. URL `http://www.nikonusa.com/en/learn-and-explore/a/tips-and-techniques/understanding-maximum-aperture.html#`.

OpenCV. The coordinate system of pin-hole camera model., 2017a. URL `http://answers.opencv.org/question/31470/the-coordinate-system-of-pinhole-camera-model/`.

OpenCV. Feature matching opencv, 2017b. URL `http://docs.opencv.org/trunk/dc/dc3/tutorial_py_matcher.html`.

OpenCV. Opencv, 2017c. URL `http://opencv.org/`.

Parth N Patel, Malav A Patel, Rahul M Faldu, and Yash R Dave. Quadcopter for agricultural surveillance. *Advance in Electronic and Electric Engineering*, 3(4):427–432, 2013.

Hung Pham, Scott A Smolka, Scott D Stoller, Dung Phan, and Junxing Yang. A survey on unmanned aerial vehicle collision avoidance systems. *arXiv preprint arXiv:1508.07723*, 2015.

Richard J Radke. *Computer vision for visual effects*. Cambridge University Press, 2013.

Benjamin Ranft, Jean-Luc Dugelay, and Ludovic Apvrille. 3d perception for autonomous navigation of a low-cost mav using minimal landmarks. In *Proceedings of the International Micro Air Vehicle Conference and Flight Competition (IMAV2013), Toulouse, France*, volume 1720, 2013.

James F Roberts, Timothy Stirling, Jean-Christophe Zufferey, and Dario Floreano. Quadrotor using minimal sensing for autonomous indoor flight. In *European Micro Air Vehicle Conference and Flight Competition (EMAV2007)*, number LIS-CONF-2007-006, 2007.

Inkyu Sa, Hu He, Van Huynh, and Peter Corke. Monocular vision based autonomous navigation for a cost-effective mav in gps-denied environments. In *International Conference on Advanced Intelligent Mechatronics (AIM), 2013 IEEE/ASME*, pages 1355–1360. IEEE, 2013.

Suman Saha, Ashutosh Natraj, and Sonia Waharte. A real-time monocular vision-based frontal obstacle detection and avoidance for low cost uavs in gps denied environment. In *International Conference on Aerospace Electronics and Remote Sensing Technology (ICARES), 2014 IEEE*, pages 189–195. IEEE, 2014.

AI Shack. Scale-invariant feature transform, 2017. URL `http://aishack.in/tutorials/sift-scale-invariant-feature-transform-introduction/`.

Peter F Sturm and Stephen J Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In *Computer Society Conference on Computer Vision and Pattern Recognition, 1999. IEEE*, volume 1, pages 432–437. IEEE, 1999.

Roger Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal on Robotics and Automation*, 3(4):323–344, 1987.

Various. Cameta calibration toolbox for matlab, 2017. URL `http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example.html`.

Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

Juyang Weng, Paul Cohen, Marc Herniou, et al. Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on pattern analysis and machine intelligence*, 14(10):965–980, 1992.

Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *International Conference on Computer Vision, 1999. The Proceedings of the Seventh IEEE*, volume 1, pages 666–673. Ieee, 1999.