António Manuel Passos Eleutério

Licenciado em Ciências da Engenharia Electrotécnica e de Computadores

2D Position System for a Mobile Robot in Unstructured Environments

Dissertação apresentada para obtenção do Grau de Mestre em Engenharia Electrotécnica e de Computadores, pela Universidade Nova de Lisboa, Faculdade de Ciências e Tecnologia.

Orientador : Fernando Coito, Professor Associado, FCT-UNL

Júri:

Presidente: Anikó da Costa, Professora Auxiliar, FCT-UNL

Arguente: Paulo Gil, Professor Auxiliar, FCT-UNL

Vogal: Fernando Coito, Professor Associado, FCT-UNL

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Julho, 2014

2D Position System for a Mobile Robot in Unstructured Environments

Success is the product of too many failures.

To my family.

Acknowledgements

I would like to thank to professor Paulo Oliveira and Carlos Cardeira for the guidance and suggestion of this dissertation theme. A very interesting and challenging theme covering various study areas, that contributed to my knowledge growth.

I express my gratitude to my advisor Fernando Coito for his wise advising towards scientific knowledge. All meetings I had with him were very instructive, making me realize difference perspectives of knowledge. Without him this dissertation quality would be inferior.

Prof. Luís Palma was not officially an advisor or supervisor. However, he was always available to an interesting chat, providing support and advices, related to my dissertation, scientific matters, finding a job matters, or other cultural matters, and sometimes even leisure matters. I am grateful to him for all this companionship.

Thank you to all my friends from FCT-UNL. In special, João Pires for his constant support and friendship throughout this degree. My flatmates Carlos Raposo and Fábio Querido for their support in my bad mood times, for their companionship, studying and growing up together. And José Vieira for his friendship and encouraging me in critical times.

Finally, I would like to show my special gratitude to my family, that provided me support and education to surpass this stage of life.

Resumo

Hoje em dia, existem vários sensores e mecanismos para estimar a trajetória e localização de um robô móvel, relativamente ao seu meio de navegação. Normalmente os mecanismos de posicionamento absoluto são os mais precisos, mas também são os mais caros, e requerem equipamento pré-instalado, exterior ao robô. Portanto, um sistema capaz de medir o seu movimento e localização relativamente ao seu meio de navegação (posicionamento relativo) tem sido uma área de investigação fulcral desde o início do aparecimento dos veículos autónomos. Com o aumento do desempenho computacional, o processamento da visão por computador tem-se tornado mais rápido e, portanto, tornado possível incorporá-la num robô móvel. Em abordagens, baseadas em marcadores, de odometria visual, a estimação de modelos implica a ausência total de falsas associações de marcadores (*outliers*) para se ter uma correta estimação. A rejeição de *outliers* é um processo delicado, tendo em conta que existe sempre um compromisso entre a velocidade e fiabilidade do sistema.

Esta dissertação propõe um sistema de posição 2D, para uso interior, usando odometria visual. O robô móvel tem uma câmera apontada ao teto, para a análise de imagem. Como requisitos, o teto e chão (onde o robô se move) devem ser planos. Na literatura, o RANSAC é um método muito usado para a rejeição de *outliers*. No entanto, pode ser lento em circunstâncias crítica. Por conseguinte, é proposto um novo algoritmo que acelera o RANSAC, mantendo a sua fiabilidade. O algoritmo, chamado FMBF, consiste na comparação de padrões de textura entre as imagens, preservando os padrões mais parecidos. Existem vários tipos de comparação, com differentes custos e fiabilidade computacional. O FMBF gere estas comparações, a fim de otimizar o compromisso entre velocidade e fiabilidade.

Palavras Chave: Odometria Visual, *RANdom SAmple Consensus* (RANSAC), *Feature Metrics Best Fit* (FMBF), Marcadores.

Abstract

Nowadays, several sensors and mechanisms are available to estimate a mobile robot trajectory and location with respect to its surroundings. Usually absolute positioning mechanisms are the most accurate, but they also are the most expensive ones, and require pre installed equipment in the environment. Therefore, a system capable of measuring its motion and location within the environment (relative positioning) has been a research goal since the beginning of autonomous vehicles. With the increasing of the computational performance, computer vision has become faster and, therefore, became possible to incorporate it in a mobile robot. In visual odometry feature based approaches, the model estimation requires absence of feature association outliers for an accurate motion. Outliers rejection is a delicate process considering there is always a trade-off between speed and reliability of the system.

This dissertation proposes an indoor 2D position system using Visual Odometry. The mobile robot has a camera pointed to the ceiling, for image analysis. As requirements, the ceiling and the floor (where the robot moves) must be planes. In the literature, RANSAC is a widely used method for outlier rejection. However, it might be slow in critical circumstances. Therefore, it is proposed a new algorithm that accelerates RANSAC, maintaining its reliability. The algorithm, called FMBF, consists on comparing image texture patterns between pictures, preserving the most similar ones. There are several types of comparisons, with different computational cost and reliability. FMBF manages those comparisons in order to optimize the trade-off between speed and reliability.

Keywords: *Visual Odometry* (VO), RANSAC, FMBF, Features.

List of Acronyms

AGV *Automatic Guided Vehicle*

AR *Aspect Ratio*

CS *Corner Signature*

CDS *Corner Distance Signature*

CTS *Corner Triangle Signature*

CTAS *Corner Triangle Angle Signature*

CTDS *Corner Triangle Distance Signature*

DOF *Degrees of Freedom*

DoG *Difference of Gaussian*

EKF *Extended Kalman Filter*

FAST *Features from Accelerated Segment Test*

FAST-ER *FAST - Enhanced Repeatability*

FMBF *Feature Metrics Best Fit*

FOV *Field Of View*

FPS *Frames Per Second*

GPS *Global Positioning System*

IMU *Inertial Measurement Unit*

LASER *Light Amplification by Stimulated Emission of Radiation*

NN *Nearest Neighbour*

RANSAC *RANdom SAmple Consensus*

SI *International System of Units*

SIFT *Scale Invariant Feature Transformation*

SLAM *Simultaneous Localisation And Mapping*

SUSAN *Smallest Univalue Segment Assimilating Nucleus*

VO *Visual Odometry*

VSLAM *Visual SLAM*

WCA *Window Corner Ajustment*

List of Symbols

a	a pixel location of the arc A
A	set of locations of the arc of n contiguous pixels of the FAST segment test
b	distance between the lens of the camera and the ceiling
b_1	distance between the lens and the edge entrance of the camera
b_2	distance between the edge entrance of the camera and the ceiling
B	matrix of the quality values of the accepted corner matches
$\vec{B}$	ascendant sorted vector of the quality values of the accepted corner matches
BCS	RANSAC best consensus set
BM	RANSAC best model
CS	RANSAC consensus set
$C_{0:n}$	set of robot poses, between 0 and n, in relation to the origin
C_k	k^{th} robot pose in relation to the origin
D	a set of data matches used in RANSAC
$D_{\Theta,i,j}$	set of differences between Θ_i and Θ_j
$D_{\Phi,i,j}$	set of differences between Φ_i and Φ_j
E_k	set of corner matches used for the estimation of the k^{th} pose
FOV	camera field of view
FOV_w	width camera field of view
FOV_h	height camera field of view
$g_{w\times h}$	camera calibration factor for frames with resolution $w \times h$
h	frame height
H_k	relative motion between the k^{th} and the $k-1^{th}$ frame
H_α	transformation between the central reference of I_k and the ceiling reference
H_β	transformation between the central reference of I_{k-1} and the ceiling reference
$I_{0:n}$	set of frames between poses 0 and n

I_p	intensity of corner p
$I_{p \to l}$	intensity of pixel $p \to l$
I_k	k^{th} frame
It	generic designation for number of iterations
$It(N)$	number of iterations depending of the number of corners
l	a location of the FAST circle segment test
L	set of locations of the FAST circle segment test
L_Δ	number of corner triangle signatures used in corner metrics
n	minimum size of $p \to A$ for a corner detection
N	generic designation for the number of corners in a frame
N_{ask}	number of asked corners
N_f	number of filtered corners in WCA
N_{ia}	number of points inside the intercepted area between frames
$\bar{N}_{ia}^{\alpha}$	number of points outside the intercepted area between frames
N_k	number of corners in the k^{th} frame
N_α	number of corners in frame α
$N_{\alpha,b}$	number of bright corners in frame α
$N_{\alpha,d}$	number of dark corners in frame α
$N_{\alpha,nr}$	number of non repeated corners in frame α
$N_{\alpha,r}$	number of repeated corners in frame α
$N_{\beta,b}$	number of bright corners in frame β
$N_{\beta,d}$	number of dark corners in frame β
$N_{\beta,nr}$	number of non repeated corners in frame β
$N_{\beta,r}$	number of repeated corners in frame β
m	a length of the image projection of the camera
MC	number of corner match combinations
MC_{bc}	best case of the number of corner match combinations
MC_{wc}	worst case of the number of corner match combinations
MM	RANSAC maybe model
M_a	number of accepted corner matches
M_e	number of elected corner matches
M_{in}	number of corner match inliers
M_{out}	number of corner match outliers

p	generic corner designation
$p \rightarrow a$	a pixel of the arc $p \rightarrow A$ at location a
$p \rightarrow A$	arc of n contiguous pixels of the FAST segment test
$p \rightarrow l$	a pixel of the FAST detector segment test at location l
$p \rightarrow L$	all pixels of the FAST segment test
p_k	k^{th} corner in a frame
$p_{k,i,a}$	corner a that belongs to the i^{th} triangle of the k^{th} corner
$p_{k,i,b}$	corner b that belongs to the i^{th} triangle of the k^{th} corner
p_α	a corner in frame α
p_β	a corner in frame β
P	set of all pixels in a frame
P_b	set of bright corners in a frame
P_d	set of dark corners in a frame
P_s	set of non corners in a frame
P_i^α	coordinates of the i^{th} point of frame α
P_i^β	coordinates of the i^{th} point of frame β
Q	set of quick rejection locations of the FAST segment test
Q_{CDS}	(quality) average of the differences between CDS
$Q_{CDS,i,j}$	average of the differences between CDS, of i^{th} and j^{th} corners of frames α and β
$p \rightarrow Q$	set of pixels with locations Q
RP	number of repeated points in corner signatures simulator
R_k	relative rotation between the k^{th} and the $k-1^{th}$ frame
s_k	k^{th} distance of a CTS comparison
$s_{\alpha,i}$	i^{th} distance of S_α
$s_{\beta,j}$	j^{th} distance of S_β
$S_{p \rightarrow l}$	state of the pixel $p \rightarrow l$ in relation to the candidate corner p
S_α	set of corner distances in frame α
$S_{\alpha,r}$	set of repeated corner distances in frame α
S_β	set of corner distances in frame β
$S_{\beta,r}$	set of repeated corner distances in frame β
t	generic designation of FAST detector threshold
t_D	minimum number of distance comparisons for the approval of a CDS comparison
t_e	minimum number of elected corner matches for frame match approval

t_{ma}	limit number of approved comparisons in FMBF
t_{max}	maximum threshold t necessary to make a corner detectable
t_{mb}	limit of the number of accepted corner matches
t_{md}	threshold value to determine if a corner fits a model in RANSAC
t_k	FAST detector threshold used in the k^{th} frame
t_Δ	minimum neighbour distance to create a CTS
$t_{\Delta s}$	threshold used to reject CTS differences
t_Θ	threshold for a CTAS match validation
t_Φ	threshold for a CTDS match validation
T_k	relative translation between the k^{th} and the $k-1^{th}$ frame
w	frame width
W	generic designation for the set of corner coordinates in a frame
W_k	set of corner coordinates in the k^{th} frame
$W_{k,k-1}$	set of corner coordinates of the k^{th} and the $k-1^{th}$ frames
$W'_{k,k-1}$	set of filtered corner coordinates of the k^{th} and the $k-1^{th}$ frames
X	x coordinate of the robot pose
Y	y coordinate of the robot pose
Z	z coordinate of the robot pose
α	designation for previews frame
β	designation for current frame
$\Delta I_{p\rightarrow L}$	set of colour intensity differences between pixel p and each pixel on $p \rightarrow L$
$\Delta I_{p\rightarrow A}$	set of colour intensity differences between pixel p and each pixel on $p \rightarrow A$
$\Delta s_{i,j}$	difference between the distances $s_{\alpha,i}$ and $s_{\beta,j}$
η_{in}	inliers portion
$\eta_{in}(N)$	inliers portion depending on the number of corners
η_{out}	outliers portion
$\eta_{out}(N)$	outliers portion depending on the number of corners
η_r	generic designation for portion of repeated corners in a frame
$\eta_{\alpha,r}$	portion of repeated corners in frame α
$\eta_{\beta,r}$	portion of repeated corners in frame β
Υ_k	set of t_{max} values of the k^{th} frame
θ	robot orientation
θ_i	i^{th} angle of the set Θ

Θ	set of amplitude angles of a CTS
Θ_i	set of amplitude angles of a CTS of the i^{th} corner, of frame α
Θ_j	set of amplitude angles of a CTS of the j^{th} corner, of frame β
ρ	probability of selecting at least one sample with all inliers, after running RANSAC
υ	minimum number of data matches to fit a model in RANSAC
$\phi_{k,i,a}$	distance between p_k and $p_{k,i,a}$
$\phi_{k,i,b}$	distance between p_k and $p_{k,i,b}$
Φ	set of pairs of distances that creates a CTDS
Φ_i	set of pairs of distances that creates a CTDS of the i^{th} corner of frame α
Φ_j	set of pairs of distances that creates a CTDS of the j^{th} corner of frame β
φ	angle of a rigid body rotation
$\Omega_{w\times h}$	relation between millimetre and pixel in a frame with resolution $w \times h$
Ω'	normalized relation between millimetre and pixel

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the navigation and mapping systems for *Automatic Guided Vehicle* (AGV), based on the literature. An overview is performed about several sensors and techniques available to use in *Simultaneous Localisation And Mapping* (SLAM). The proposed system of the dissertation is revealed, using the *Visual Odometry* (VO) technique alone. The objectives and contributions are presented in order to solve the literature problems. An algorithm approach is proposed to make VO a faster process. Also, the structure of the dissertation is presented, organized by chapters.

1.1 Motivation

Mobile robot navigation and mapping has been a central research topic in the last few years. An autonomous mobile robot needs to track its path and locate its position relative to the environment. Such a task is still a challenge when operating in unknown and unstructured environments. However, from a theoretical and conceptual point of view, it is now considered a solved problem. SLAM is a crucial method for the resolution of this problem. SLAM has the purpose of building a consistent map while simultaneously estimating the robot position within the map. SLAM may be designed for different domains such as to outdoor or indoor, to wheeled or legged, to underwater and to airborne systems. Furthermore, robots are built for different purposes, leading to a wide variety of navigation mechanisms. Therefore, the complexity of the surroundings and the way the robot interacts with the surroundings varies considerably. Due to current technology limits, there is

not a standard approach regarding this matter. In order to achieve SLAM, different types of sensors are used, contributing for the efficiency of the process. Combinations of sensors are chosen, according to the environment and robot tasks purposes.

Wheel odometry, *Inertial Measurement Unit* (IMU), magnetic compasses, active beacons, VO, are very common techniques associated with the estimation of a vehicle position and location. Nevertheless, when combining several sensors for that purpose, visual odometry is usually a favourite choice. The main reason is that VO is very versatile, considering it is not restricted to a particular locomotion method, and the relation between motion estimation accuracy and the system expense is very appealing.

1.2 Proposed System

This dissertation aims to implement an indoor 2D tracking position system for a mobile robot through a VO operation, using a single camera pointed upward to the ceiling. The robot motion is estimated through successive comparisons between picture frames of the ceiling. In each frame, several local patterns are detected, designated as landmarks or features. The consecutive frames need to have overlapped areas, where the features can be compared and matched. By matching features throughout the frame sequence, the robot motion of the last frame in relation to the first frame is estimated.

This system is limited to a 2D path. The ceiling has to be a plane as well as the floor where the robot moves. Running a VO system in real time is a major goal in the literature. The algorithm needs to be fast and reliable enough to accomplish real time. The most burdensome process is the feature matching between frames. Therefore, a new algorithm approach is implemented in this dissertation, which aims to reduce the computational cost of the feature matching process.

1.3 Objectives and Contributions

In the literature, the feature matching, that compares all features from one frame to all other features from the other frame, is extremely computationally expensive. This dissertation proposes an approach using this method, taking advantage of its high accu-

racy, but reducing the computational cost considerably. The features used are specifically designated as corners (described in section 2.3.5), detected by the algorithm *Features from Accelerated Segment Test* (FAST). Each corner is considered a special pixel, positioned in pixel coordinates. Therefore, every corner has a *Corner Signature* (CS) which consists in all distances to the other corners of the same frame. Also, with two corner neighbours of a certain corner, a *Corner Triangle Signature* (CTS) is created. It is required a certain number of CS comparisons with a high similarity to have a frame match. Therefore, instead of comparing all CS of one frame to all CS of the other frame, the CS comparisons run until such number of comparisons with high similarity is achieved. Additionally, for each corner comparison a CS comparison is performed only if a CTS inexpensive comparison is approved. For corner matches quick rejection, a pre corner filtering is performed in pairs of frames, before the feature matching, based on a corner detection parameter (designated as FAST threshold). Also, in order to have an appropriate number of corner detections per frame, a corner controller is presented, based on the last used FAST threshold and number of detected corners.

The work developed in this dissertation, led to a paper presented in an international conference, [CECV14].

1.4 Organization

This dissertation is organized as follows.

Chapter 1 introduces the navigation and mapping systems for AGV, based on the literature. An overview is performed about SLAM and several sensors and techniques used for it. It is revealed the proposed system of the dissertation, using the VO technique alone, and its constraints. The objectives and contributions are presented in order to solve the literature problems. It is proposed an algorithm approach to make VO faster. Also, the structure of the dissertation is presented, organized by chapters.

Chapter 2 contains a literature review about mobile robotics position systems. It performs an overview about several sensors and techniques used in SLAM. The sensors are divided in absolute and relative positioning types, and comparisons are performed between them. The benefits and inconvenients of each sensor are emphasized and several compar-

isons are performed. Several VO techniques presented in the literature are explained and mentioned where they belong in the VO block diagram. Feature detection, feature matching and outlier removal are explained in detail and presented their importance to the success of a VO process.

Chapter 3 presents the VO solution proposed in this dissertation. It describes the camera projection that takes picture frames and justifies why the limited resolution of the frame. The FAST corner detector is detailed explained along with the differences between several *FAST-n*, varying *n*. After detecting the corners in one frame, it is explained how corner signatures are created in order to allow the corner matching between frames. RANSAC used alone for corner matching would bring a high temporal complexity, as stated. Therefore, the FMBF is carefully detailed in order to prove the reduction of such complexity. Using the frames, a map is performed throughout the trajectory. It is also mentioned how the camera calibration was performed, using several pictures taken to a certain plane. And also, the two simulators used to validate the FMBF algorithm are detailed.

Chapter 4 illustrates the experimental results of three particular tests. In the first test, nine frames of the same ceiling are used to build a map. In the second test, twenty four pictures of an other ceiling are used to estimate the robot path. In the third test, a video is used to create a map, based on where the robot went through.

And finally, chapter 5 presents a conclusion based on the performed VO solution. MATLAB is referred as a slow language for online calculations. Also, the experimental results brought a new understanding, related to the motion estimation accuracy. For future work, it is suggested to use a faster programming language and a more robust corner detector. Also, several more VO techniques to raise the estimation reliability, along with other sensor techniques.

Chapter 2

AGV Localisation

This chapter contains a literature review about mobile robotics position systems. An overview about several sensors and techniques used in SLAM is performed. The sensors are divided in absolute and relative positioning types, and comparisons are performed between them. In each sensor the benefits and inconvenients are highlighted and several comparisons are shown. It is explained several VO techniques presented in the literature and mentioned where they belong in the VO block diagram. Feature detection, feature matching and outlier removal are detailed and presented their relevance to the success of a VO process.

2.1 *Simultaneous Localisation And Mapping* (SLAM)

The probabilistic SLAM problem was firstly proposed in the 1986 IEEE Robotics and Automation Conference held in San Francisco. A discussion between many researchers led to the recognition that a consistent probabilistic mapping was a fundamental problem in robotics with major conceptual and computational issues that needed to be addressed [DwB06]. In [SSC87, DW87] was established the statistical basis for describing relationships between landmarks and manipulating geometric uncertainty. Those works had the purpose of showing that there is a high degree of correlation between the estimates of the location of different landmarks in a map, and that these correlations grow with successive observations. Several results, regarding this issue, were demonstrated in visual navigation [AF88] and sonar-based navigation using Kalman filter type algorithms [CL85, Cro89].

Therefore, the solution for localisation and mapping required a joint state composed of the vehicle pose (i.e., position and orientation) and every landmark position, to be updated following each landmark observation [SSC87]. Currently that is how SLAM is performed.

In SLAM the mobile robot can build a map of an environment and at the same time use the map to determine its location. Both the trajectory of the robot and the location of all landmarks are estimated online and without any location awareness in advance. Figure 2.1 illustrates an example SLAM draft.

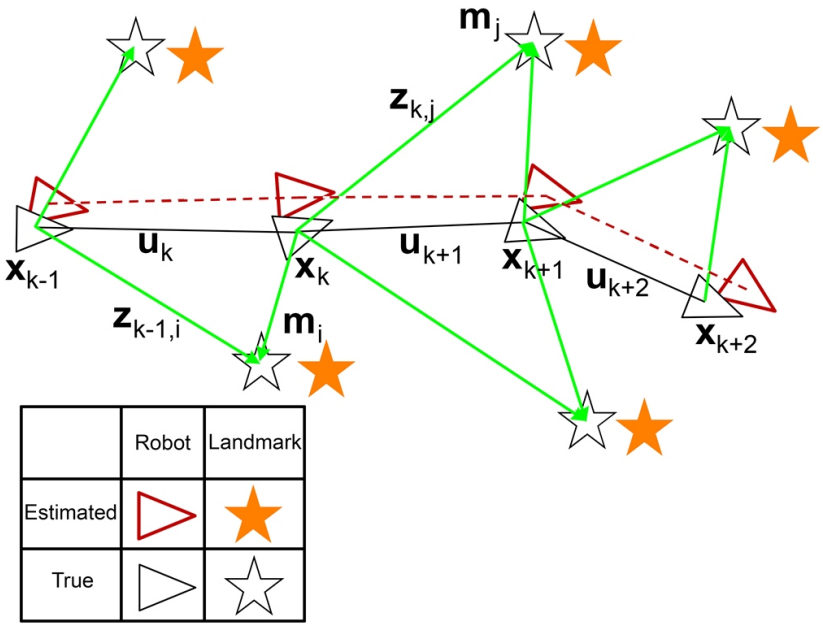

Figure 2.1: Example of the SLAM problem.

In the figure a mobile robot moves across an environment taking relative observations of a number of landmarks using a sensor located on the robot. At time instant k the following quantities are defined:

1. x_k: state vector of the location and orientation of the vehicle.

2. u_k: control vector, applied at time $k-1$ to drive the vehicle to state x_k at time k.

3. m_i: vector describing the location of the i^{th} landmark whose true location is assumed time invariant.

4. $z_{k,i}$: observation taken from the vehicle of the location of the i^{th} landmark at time k. The observation is written simply as z_k when the specific landmark is not relevant

to the discussion, or there are multiple landmark observations at a time k.

The following sets are also defined:

1. $X_{0:k} = \{x_0, x_1, ..., x_k\} = \{X_{0:k-1}, x_k\}$: History of the vehicle locations.
2. $U_{0:k} = \{u_1, u_2, ..., u_k\} = \{U_{0:k-1}, u_k\}$: History of control inputs.
3. $m = \{m_1, m_2, ..., m_n\}$: Set of all landmarks.
4. $Z_{0:k} = \{z_1, z_2, ..., z_k\} = \{Z_{0:k-1}, z_k\}$: Set of all landmark observations.

SLAM requires the probability distribution $P(x_k, m|Z_{0:k}, U_{0:k}, x_0)$ to be computed for all k times. This probability distribution describes the joint posterior density of the landmark locations and the vehicle state, at time k, through the recorded observations and control inputs from the start until time k, and the initial state of the vehicle.

A recursive solution is used to the SLAM problem [DwB06]. It starts with an estimation of $P(x_{k-1}, m|Z_{0:k-1}, U_{0:k-1})$ at time $k-1$, where the joint posterior is computed using Bayes Theorem, following a control u_k and an observation z_k. This computation requires a state transition model and an observation model, describing the effect to the control input and observation respectively.

The most common representation for these models is in the form of a state-space model with additive Gaussian noise, leading to the use the *Extended Kalman Filter* (EKF).

The **motion model** of the vehicle can then be described in terms of a probability distribution of state transitions in the form of:

$$P(x_k|x_{k-1}, u_k) : x_k = f(x_{k-1}, u_k) + w_k, \tag{2.1}$$

where $f(.)$ models vehicle kinematics and where w_k are additive, zero mean uncorrelated Gaussian motion disturbances with covariance Q_k.

The **observation model** describes the probability of making an observation z_k when the vehicle location and landmark locations are known, and is described in the form:

$$P(z_k|x_k, m) : z_k = h(x_k, m) + v_k, \tag{2.2}$$

where $h(.)$ describes the geometry of the observation and where v_k are additive, zero mean uncorrelated Gaussian observation errors with covariance R_k.

2.2 Absolute and Relative Positioning

There are two basic position estimation methods, absolute and relative positioning, for a mobile robot. If possible, they are usually employed together for better reliability [BK87, SCDE95]. Relative positioning is based on monitoring robot poses, computing the offset from a known starting position. Absolute positioning methods rely on systems external to the robot mechanisms. They can be implemented by a variety of methods and sensors, but all of them present relevant inconvenients. For example, navigation beacons and landmarks have a significant accuracy, but require costly installations and maintenance. Or map-matching methods, also very accurate, are usually slow, preventing general commercial applications. Therefore, with these measurements it is necessary for the work environment either be prepared or be known and mapped with high precision [BKP92].

GPS can be used only outdoors and has a poor accuracy of 10 to 30 meters for non military devices. Such limitation is imposed by the US government deliberately, through small errors in timing and satellite position to prevent a hostile nation from using GPS in support of precision weapons delivery.

2.3 Sensors and Techniques

There is a wide variety of sensors and techniques for mobile robot positioning:

1. Classical odometry (wheeled), where the number of wheel revolutions is monitored;
2. IMU, which measures acceleration and orientation;
3. Magnetic compasses, that measures the Earth magnetic field;
4. Active beacons, which uses triangulation and trilateration;
5. Visual odometry, where visual frame features are matched;

6. LASER, which analyses scanned features.

Nevertheless, only the first five methods are worth devoting some attention for the background of this work.

2.3.1 Odometry

Odometry is one of the most used positioning systems for mobile robots. It is inexpensive, accurate in short term and allows high sampling rates. The principle operation of odometry is the aggregation of incremental motion information over time. The vehicle localisation is estimated by calculating the performed displacement from the initial position. Due to the incremental nature of this measure type, the measure uncertainty increases over time [BEFW97]. In particular, orientation errors cause significant lateral errors, which grow proportionally with the distance traveled by the robot. With time those errors may become so large that the estimated robot position is completely wrong after 10 m of travel [GT94]. Nevertheless, most researchers invest in odometry considering that navigation tasks could be simplified by improving the odometric accuracy. For example, in order to obtain more reliable position estimations, [SD+99] and [GT94] propose fusing odometric data with absolute position measurements, mentioned previously.

Odometry is based on simple equations, which translates wheel revolutions into robot linear displacement relative to the floor, in order to provide the robot local reference relative to the global reference, as illustrated in figure 2.2.

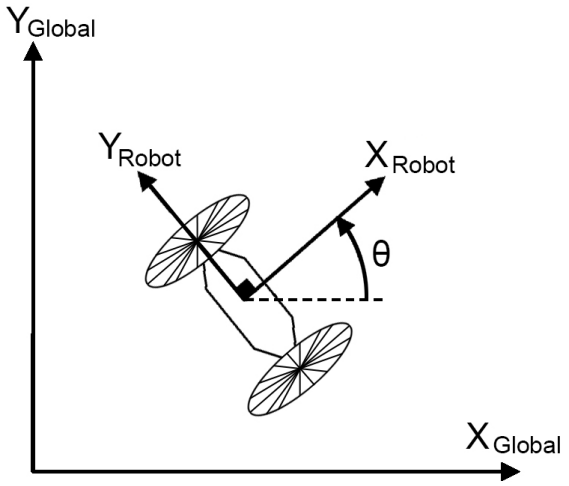

Figure 2.2: Global reference and robot local reference.

However, wheel slippage may prevent the proportional translation into linear motion, among other smaller causes. There are two types of errors to consider, designated as systematic and non-systematic [Bor96]. Systematic errors result from kinematic imperfections of the robot, such as, differences in wheel diameters or uncertainty associated with the exact wheelbase. Non-systematic errors result from the interaction of the wheels with the floor, such as, wheel slippage or bumps and cracks. In [Bor96] a method, called UMBmark, was designed to measure odometry systematic errors. UMBmark requires that the mobile robot follows an experience with stipulated conditions, in order to obtain a numeric value that expresses the odometric accuracy. In addition, similar to the latter, the extended UMBmark method was designed to measure non-systematic errors. In both cases, a calibration procedure was developed to reject or reduce odometry errors, providing a more reliable odometry process.

2.3.2 Inertial Measurement Unit

IMU is also an option when it comes to estimate a mobile robot position. It uses gyroscopes and accelerometers to measure rate of rotation and acceleration, respectively [BEFW97]. IMU systems have the advantage of being independent from external references. However, they have other serious disadvantages, as show in [BDW95] and [BDw94]. Studies found that there is a very poor signal-to-noise ratio at lower accelerations. Measurements need to be integrated twice, for accelerometers, to obtain the position, which makes these sensors extremely sensitive to drift. Also, they are sensitive to uneven ground. If there is a small disturbance in a perfectly horizontal position, the sensor detects a component of the gravitational acceleration g, adding more error to the system.

Gyroscopes provide information related to the rate of rotation of a vehicle only, which means their data need to be integrated once. They are more accurate than accelerometers, but also more costly. Gyroscopes can help compensate the foremost weakness of odometry, that is, large lateral position error. Small momentary orientation errors cause a constantly growing lateral position error. Therefore, detecting and correcting immediately orientation errors would be of great benefit.

Figure 2.3 illustrates the 6 DOF an IMU sensor can read.

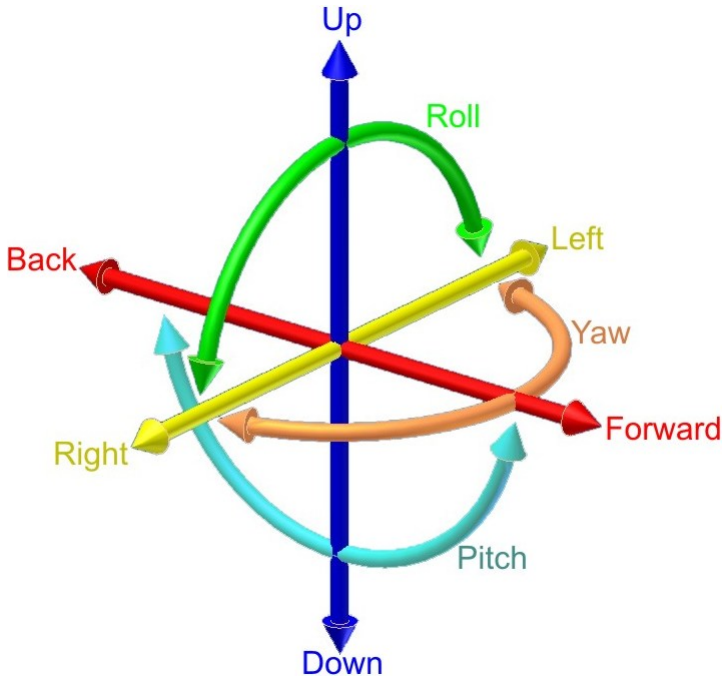

Figure 2.3: 6 DOF read by the IMU sensor.

2.3.3 Magnetic Compasses

A magnetic compass is a sensor that provides a measure of absolute heading. This is important in solving navigation needs for mobile robots, considering there are no accumulated dead-reckoning errors. However, one unavoidable disadvantage of any magnetic compass is that the Earth magnetic field is often distorted near steel structures [BKP92] or power lines. For this reason, these sensors are not reliable for indoor applications.

There are different sensor systems, base on a variety of physical properties related to the Earth magnetic field. For example: mechanical magnetic compasses, fluxgate compasses, hall-effect compasses, magnetoresistive compasses, and magnetoelastic compasses, among other. The most used sensor for mobile robot applications is the fluxgate compass. By maintaining in a certain level attitude, it measures the horizontal component of the Earth magnetic field. As advantages it has low power consumption, intolerance to shock or vibration, quick start-up, and low cost. If the robot is projected to operate in uneven terrain, the sensor should be gimbal-mounted and mechanically dampened, in order to prevent large errors from the vertical component of the geomagnetic field [BEFW97].

2.3.4 Active Beacons

Active beacons are widely used for ships and airplanes, as well as on commercial mobile robotic systems. This navigation system provides accurate positioning information with minimal processing, as active beacons can be easily detected. It results that this approach stands high sampling rates with high reliability, but the cost of installation and maintenance is also very high. There are two different types of active beacon systems: trilateration and triangulation [BEFW97].

Trilateration is based on distance measurements to calculate the beacon sources, which leads to the estimation of the vehicle position. Usually there are three or more transmitters mounted at known locations in the environment and a receiver on board the vehicle. In reverse, there may be receivers mounted in the environment and one transmitter on board the vehicle. The famous *Global Positioning System* (GPS) is a particular trilateration example.

In triangulation there are three or more active transmitters mounted in know locations, as in trilateration. However, instead of estimating the position of the robot using the distances between transmitters and receiver, it uses the angles $\alpha_k \in \{\alpha_1, \alpha_2, \alpha_3\}$ between the sensor and the three longitudinal axis, performed between the transmitters $T_k \in \{T_1, T_2, T_3\}$ and the sensor S. The sensor keeps on rotating on board, and registers the three angles. From these three measurements the robot coordinates and orientation can be computed. Figure 2.4 illustrates the draft of the triangulation active beacon type.

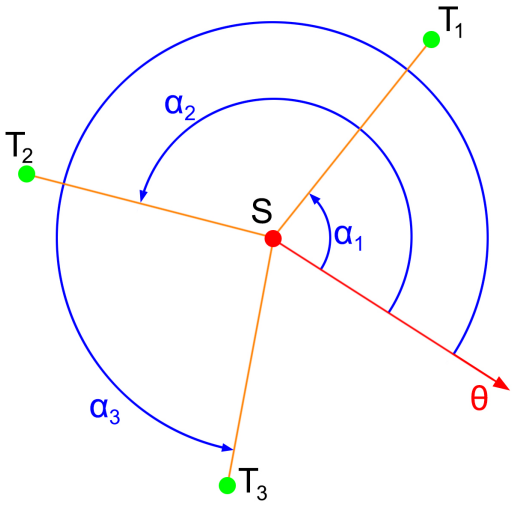

Figure 2.4: Triangulation draft of the active beacons system.

2.3.5 Visual Odometry

Nowadays, computer vision is an important application to mobile robotics. SLAM has often been performed using other sensors rather than regular cameras. However, recent improvements in both sensors and computing hardware have made real-time vision processing much more practical for SLAM applications, as computer vision algorithms mature. Furthermore, cameras are inexpensive and provide high information bandwidth, serving as cheap and precise localisation sensor.

The term VO was created in 2004 by Nistér in his landmark paper [NNB04]. This term was chosen due to its similarity to wheel odometry. VO operates by successively estimate the pose of the vehicle through examination of the changes that motion induces on the images of its onboard cameras. A stereo head or a single camera may be used. As requirements, there should be sufficient illumination in the environment and a static scene with enough texture to allow apparent motion to be extracted. This system aims to construct the trajectory of the vehicle with no prior knowledge of the scene nor the motion for the pose estimation (relative positioning system). Consecutive picture frames should be captured by ensuring that they have sufficient scene overlap. VO works by detecting features (also named as keypoints or interest points) in the frames and matching them over overlapping areas from consecutive frames. The feature detection is usually performed with features such as corners or blobs. Figure 2.5 illustrates an example of blobs detection.

Figure 2.5: Example of blobs detection. From [FS11].

The feature matching may be performed by matching features in pairs of frames, or tracking features throughout several frames.

VO is not restricted to a particular locomotion method. It can be used in any robot with sufficiently high quality camera [FS11]. Compared to wheeled odometry, it has no slippage problems in uneven terrain or other adverse conditions. Nevertheless, outdoor terrains is in some way more challenging than indoor environments. Outdoors are unstructured, and simpler features such as corners, planes and lines that are abundant in indoor environments rarely exist in natural environments [AK07].

There is no ideal and unique VO solution for every possible working environment. The optimal solution should be chosen accurately according to the specific navigation and the given computational resources. Furthermore, many approaches use VO in conjunction with information from other sources such as GPS, inertia sensors, wheel encoders, among others [CECV14, How08, MCM07, KCS11]. Over the three decades of VO history, only in the third decade real-time flourished, which has led this system to be used on another planet by two Mars exploration rovers for the first time. In Mars, rovers used IMU and wheeled encoder-based odometry that performed well within the design goal of at most 10% error. However, the rover vehicles were also driven along slippery slopes tilted as high as 31 degrees. In such conditions VO was employed to maintain a sufficiently accurate onboard position estimation [Mai05].

In [AK07] an approach using VO, IMU and GPS was used in rough terrain, accomplishing localisations over several hundreds of meters within 1% of error.

In [NNB04] it is described a real-time method for deriving vehicle motion from monocular and stereo video sequences, in autonomous ground vehicles. Obstacle detection and mapping is performed using visual input in stereo data, as well as, estimating the motion of the platform in real-time.

2.3.5.1 The problem

The problem of estimating a vehicle egomotion from visual input alone was first triggered by Moravec [Mor80], supported by NASA Mars exploration program. The work of Moravec stands out for describing one of the earliest corner detectors, known today as the

Moravec corner detector, and for presenting the first motion estimation pipeline, whose main functioning blocks are still used today. Figure 2.6 presents that pipeline for a VO system.

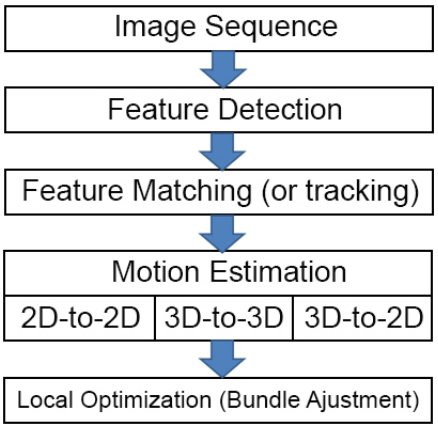

Figure 2.6: A block diagram showing the main components of a VO system.

The image sequence can be stereo or monocular. Moravec used monocular for Mars rovers. His VO system performed a 3D motion estimation with 6 *Degrees of Freedom* (DOF), equipped with what he termed as a slider stereo: a single camera sliding on a rail. The robot moved in a stop-and-go fashion, digitizing images and analysing them using extracted corners. Corners were detected in the images by his operator. The motion was then computed through a rigid body transformation to align the triangulated 3D points seen at two consecutive robot positions [FS11].

Most of the research done in VO has been produced using stereo cameras. Superior results were demonstrated in trajectory recovery for a planetary rover, due to absolute scale possession [MS87].

In both camera options, egomotion estimates alone results in accumulation errors which grows with the distance travelled, leading to increased orientation errors [OMSM00]. This error accumulation generates a drift of the estimated trajectory from the real path. The VO drift can be reduced through combination with other sensors as mentioned previously. However, it is of common interest increase the visual reliability. Therefore, approaches like window bundle adjustment [KAS11, OMSM00] and loop closure were developed [FS11]. The window bundle adjustment consists on performing a pose estimation using, not just the last pose, but rather the previous m poses. In [KAS11] it is proved

that windowed bundle adjustment can decrease the final position error by a factor of 2-5 on ranges up to 10km, in a VO experiment. Loop closure implies realizing when the robot returns to a previously visited area. This approach is more common in *Visual SLAM* (VSLAM), considering it is necessary to keep track of the map landmarks, in order to recognize a visited area.

2.4 VO Motion Estimation

In the case of a monocular system, the set of images taken at time k is denoted by $I_{0:n} = \{I_0, ..., I_n\}$. In the case of a stereo system, there are a left and a right image at every time instant, denoted by $I_{l,0:n} = \{I_{l,0}, ..., I_{l,n}\}$ and $I_{r,0:n} = \{I_{r,0}, ..., I_{r,n}\}$. For simplicity, the camera coordinate is assumed to be also the robot coordinate. Two camera positions at adjacent time instants $k-1$ and k are related by the rigid body transformation $H_{k,k-1} \in \Re^{4\times4}$ of the following form:

$$H_{k,k-1} = \begin{bmatrix} R_{k,k-1} & T_{k,k-1} \\ 0 & 1 \end{bmatrix}, \tag{2.3}$$

where $R_{k,k-1} \in \Re^{3\times3}$ is the rotation matrix, and $T_{k,k-1} \in \Re^{3\times1}$ the translation vector. The set $H_{1:n} = \{H_{1,0}, ..., H_{n:n-1}\}$ contains all subsequent motions. Camera poses are $C_{0:n} = \{C_0, ..., C_n\}$, containing the transformations of the camera with respect to the initial coordinate frame at $k = 0$. The current pose C_n is computed by concatenating all the transformations $H_{k,k-1}$ $(k = 1, ..., n)$, and therefore, $C_k = C_{k-1}H_{k,k-1}$, whereas C_0 is the camera pose at the instant $k = 0$, which can be set arbitrarily by the user.

VO aims to compute the relative transformations $H_{k,k-1}$ from the images I_k and I_{k-1}, and then to concatenate the transformations in order to recover the full trajectory $C_{0,n}$ of the camera [FS11]. Figure 2.7 illustrates an example of the VO problem with stereo camera.

In the figure, each pair of blue rectangles represents a pair of images captured by the stereo camera, therefore, a robot pose. The black stars represent natural features in the environment, which are represented as orange starts when projected in the digitalized images of the cameras.

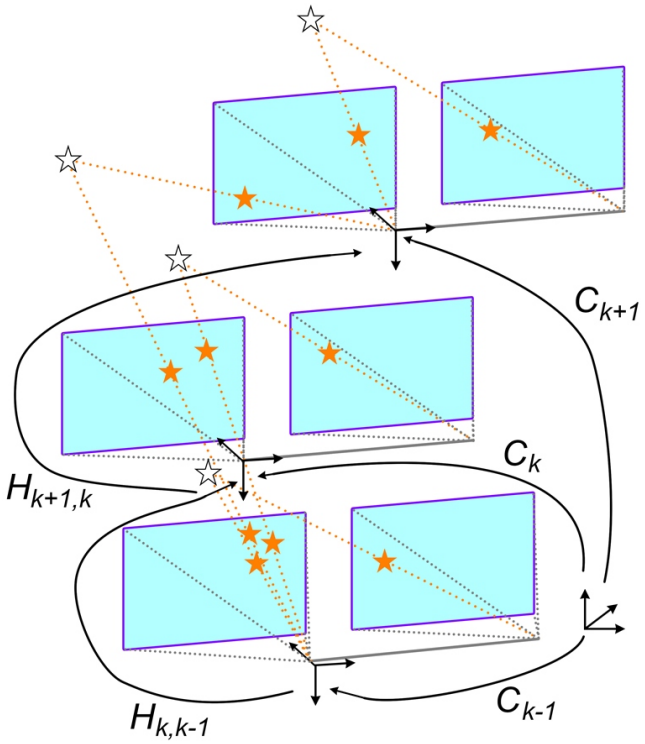

Figure 2.7: Illustration of the VO problem. From [FS11].

2.5 Detecting and Matching Features

Besides feature based methods, to compute the relative motion $H_{k,k-1}$, may also be used appearance based methods, and hybrid methods. Appearance based methods use the intensity information of all the pixels in two input image (or subregion). A known example of this method is using all the pixels in two images to compute the relative motion using a harmonic Fourier transform [MGD07]. This method has the advantage of working with low texture, but it is computationally extremely expensive. Hybrid methods use a combination of the other two.

In feature detection, the image is searched for salient keypoints that are likely to match well in other images. Point detectors, such as corners of blobs, are important for VO because of their accurate positioning measurements in the image. A blob is an image pattern different from its immediate neighbourhood in intensity, colour, and texture.

Feature detectors are characterized by: accurate localization (in position and scale), repeatability (i.e., features should be redetected across frames), computational efficiency, robustness (to noise and blur), distinctiveness (distinguish features for easier feature

matching), and invariance (to illumination, rotation, scale and perspective distortion).

To accomplish feature matching (or tracking), features are characterized by a set of metrics, named feature descriptor. For example, the simplest descriptor of a feature is its appearance (i.e., the intensity of the pixels in a path around the feature point).

The simplest way for matching features between two frames is to compare all feature descriptors from one frame with all feature descriptors from the other frame. Such comparisons are performed by similarity between pairs of feature descriptors. However, exhaustive matching becomes quadratic with the number of features, which is computationally expensive, and could became impractical when the number of features is large. Other feature matching methods like multidimensional tree search or hash table could be used [BL97]. Figure 2.8 illustrates an example of feature matching in two frames.

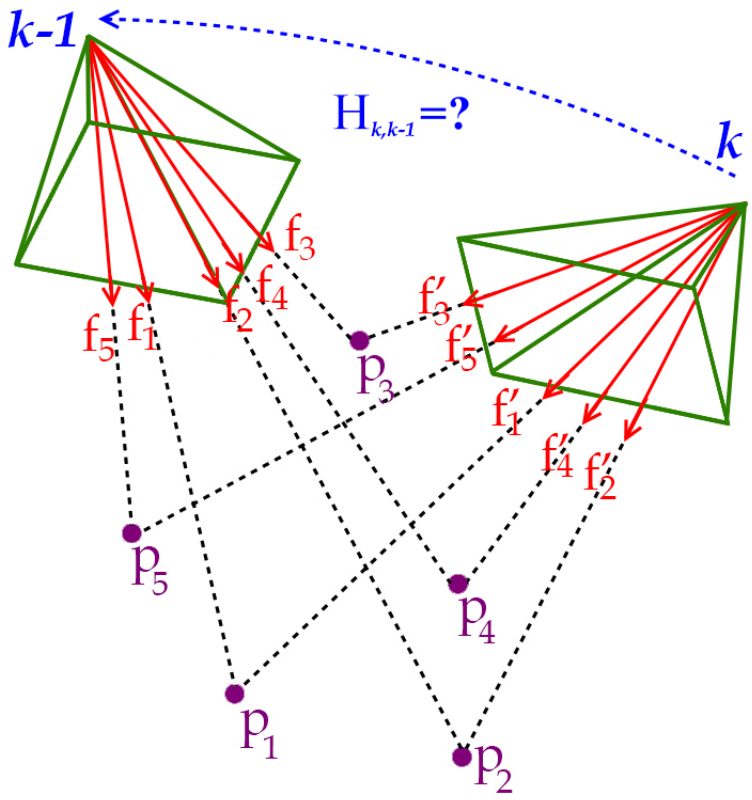

Figure 2.8: Example of feature matching between two frames.

In the example the camera poses are in samples k and $k-1$, the features are of type f_i and f'_i, the corresponding points in the environment are of type p_i, and the pose transformation is $H_{k,k-1}$ ($i \in \{1,2,3,4,5\}$).

Feature tracking is an alternative to feature matching. It consists in detecting features

in the first image and search for their corresponding matches in the following images. This method is suitable for situations when the amount of motion and appearance deformation between adjacent frames is small. However, the features can suffer from large changes in long images sequences, providing a bad outcome [FS11].

2.6 Feature Detectors

In the VO literature there are many point feature detectors. The most common corners detectors are: Moravec [Mor80], Förstner [FG87], Harris [HS88], Shi-Tomasi [ST94], and FAST [RD05]. And the most common blob detectors are: SIFT [TW09], SURF [BTG06], and CENSURE [AKB08]. Here is performed an overview of Moravec and Harris, which have been widely used in VO applications, and FAST that is used in this work. Although FAST is not the most robust in 3D localization, the literature claims this feature detector is currently the fastest one, among the other corner detectors [FS12]. Therefore, FAST is chosen for this work due to its speed, and due to the fact that this work aims to accomplish 2D localization. With 2D localization, significantly less complex than 3D, there is not a significant difference in the corner detectors in terms of robustness.

2.6.1 Moravec Corner Detector

Moravec operator is considered a corner detector, since it defines interest points as points where there is a large intensity variation in every direction. The concept "point of interest" as distinct regions in images was defined by Moravec. In his work, it was concluded that to find matching regions in consecutive image frames, these interest points could be used. This proved to be a vital low level processing step in determining the existence and location of objects in the vehicle environment [Mor80].

The Moravec corner detector algorithm is stated below.
The input parameters of this algorithm are:

I - Grayscale image;

$w(x, y)$ - Binary window;

T - Threshold.

The output parameter of this algorithm is:

C - Cornerness map.

Input : I, $w(x, y)$, T

Output: C

for *each pixel (x,y) in* I, **do**

`// Calculate the itensity variation from shifts` (u, v) `of:`

`//` $(1, 0), (1, 1), (0, 1), (-1, 1), (-1, 0), (-1, -1), (0, -1), (1, -1)$

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

`// Calculate the cornerness map`

$C(x, y) = min(V_{u,v}(x, y))$

`// Threshold the interest map`

if $C(x, y) < T$ **then**

  $C(x, y) = 0$

Perform non-maximal suppression to C `// To find local maxima`

`// All the remaining non-zero points in` C `are corners.`

Algorithm 2.1: Moravec.

Harris corner detector [HS88] was developed as a successor of Moravec corner detector. The difference lies on a Gaussian window $w(x, y) = exp(-\frac{x^2+y^2}{2\sigma^2})$, instead of a binary window vulnerable to noise, and in the use of a the Taylor expansion to compute all shifts, instead of using a set of shifts at every 45 degree. Results proved Harris to be more efficient and robust to noise [HS88, DNB+12].

2.6.2 FAST Feature Detector

It is stated that the FAST detector is sufficiently fast allowing online operation of the tracking system [RD05]. In order to detect a corner, a test is performed at a pixel p by examining a circle of 16 pixels (a Bresenham circle of radius 3) surrounding p. A corner is detected at p if the intensities of at least 12 contiguous pixels are all above or all below the intensity of p by some threshold t. This is illustrated in figure 2.9.

The highlighted squares are the pixels used in the corner detection. The pixel C is the centre of a detected corner. The dashed lines passes through 12 contiguous pixels which

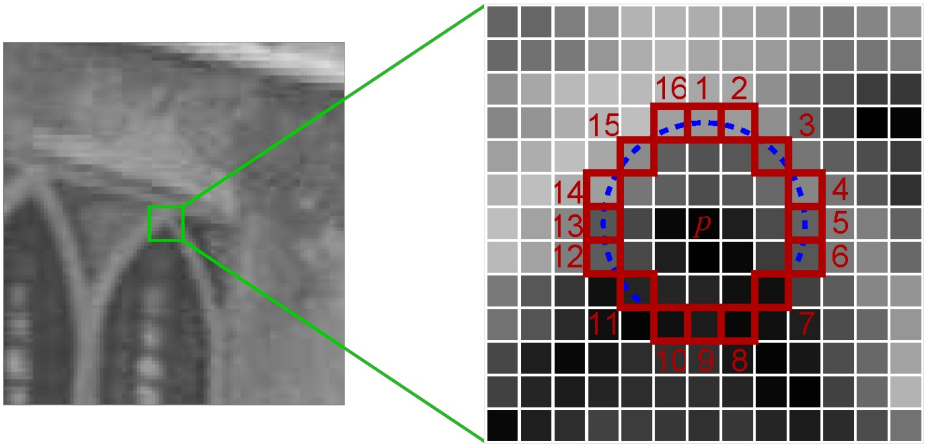

Figure 2.9: FAST detection in an image patch. From [RD05]

are brighter than C by more than the threshold.

The test of this condition is optimized by examining pixels 1, 9, 5 and 13, to reject candidate pixels more quickly, considering a corner exists only if three of these test points are all above or below the intensity of p by the threshold.

A corner is categorized as positive, where the intensities of the contiguous pixels of the segment test are greater that the intensity of the pixel in the centre, and negative, on the contrary.

Figure 2.10 shows a corner detection using FAST, where positive and negative corners are distinguished as red and yellow.

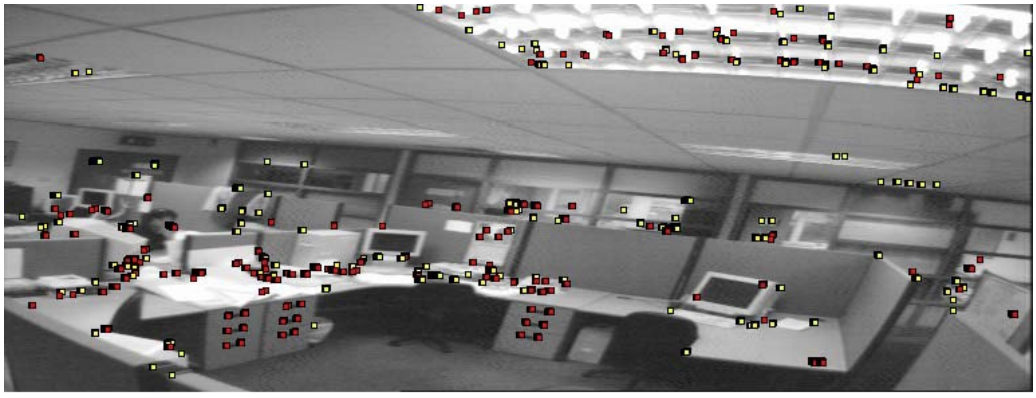

Figure 2.10: FAST detection in a frame. From [RD05]

Table 2.1 presents the time taken to perform feature detection on a PAL field (768×288 pixels) on a test system, using FAST, SUSAN and Harris, detectors.

The FAST algorithm was tested with different sizes for the contiguous arc of the

Table 2.1: Time taken to perform a feature detection with FAST, SUSAN and Harris. From [RD05].

Detector	FAST	SUSAN	Harris
Time (ms)	2,6	11,8	44

segment test. The arc with size 9 was proved to have more repeatability [RPD10].

More recently, FAST-ER algorithm was developed as a FAST successor. With a new heuristic for feature detection and using machine learning, it was demonstrated to represent significant improvements in repeatability, yielding a detector that is very fast and has a very high quality. The FAST-ER feature detector authors claim in [RPD10] (2010) it is able to fully process live PAL video using less than 5% of the available processing time. By comparison, most other detectors cannot even operate at frame rate (Harris detector 115%) [RPD10].

2.7 Outlier removal

Usually the corner matches are contaminated by outliers, i.e., wrong corner associations. Image noise, occlusions, blur, and changes in viewpoint and illumination are possible causes for outliers, that the mathematical model of the feature detector does not take into account [FS12]. Most feature matching techniques assume linear illumination changes, pure camera rotation and scaling, or affine distortions. Therefore, for an accurate estimation of the camera motion, it is crucial to remove the outliers. RANSAC is the most widely used method for outlier removal and is explained as follows.

2.7.1 RANSAC

For outlier removal the solution consists in taking advantage of the geometric constrains of the motion model. RANSAC was established as the standard method for model estimation of data contaminated with outliers. This method consists in computing model hypotheses from randomly sampled sets of data points and verify theses hypotheses on the other data points. Then, the selected solution is the model hypotheses that shows the highest consensus with the other data.

Figure 2.11 illustrates the estimation of a line that approximates the representing model of the inliers, contained on a set of data.

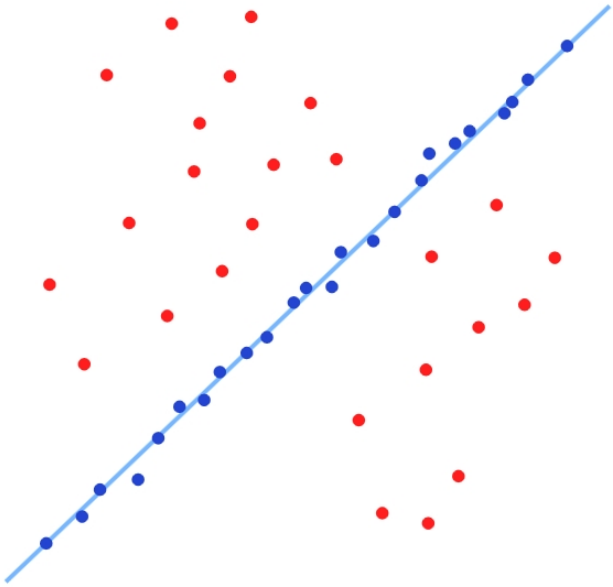

Figure 2.11: Estimated inliers of a data set contaminated with outliers.

For the stereo camera motion estimation, as used in VO, the estimated model is the relative motion between two camera positions, and the data points are the candidate features correspondences.

The RANSAC algorithm works as shown below.

The input parameters of this algorithm are:

It - the number of iterations performed by the algorithm;

D - a set of data matches;

v - the minimum number of data matches required to create the model;

d - maximum distance between a point and a model, to be considered an inlier.

The output parameter of this algorithm is:

C - Cornerness map.

Input : It, D, υ, d

Output: C

while *number of iterations* It *is not reached* **do**

Select a sample of υ points from D

Fit a model to these points

Compute the distance of all other points to this model

Count the number of points whose distance from the model is lower than d

(create the inlier set)

Store these inliers

The solution of the problem is the set with the maximum number of inliers //

Estimate the model using all the inliers.

Algorithm 2.2: RANSAC.

The number of iterations necessary to guarantee that a correct solution is found can be computed as $It = \frac{log(1-P)}{log(1-(1-\epsilon)^s)}$ where ϵ is the percentage of outliers in the data points, s is the number of data points from which the model can be instantiated, and P is the requested probability of success [FB81]. Usually It is multiplied by a factor of 10 to make sure of the solution success. As illustrated, RANSAC is a probabilistic method and is non-deterministic considering it exhibits a different solution on different runs. Nevertheless, the solution tends to be stable when the number of iterations is increased.

2.8 Summary

SLAM aims to build a map of the environment and at the same time use the map to determine the location of the robot. Several sensors may be used for this purpose, such as: classical odometry, IMU, magnetic compasses, active beacons, visual odometry, LASER, among other. The sensors may be of type absolute or relative positioning. The absolute positioning sensors are more expensive and less flexible, considering it requires pre installed equipments in the environment. Therefore, relative positioning sensors are of major importance.

Classical odometry (wheeled) was one of the first robots estimation techniques. However, despite being limited to a particular locomotion (wheeled) it is vulnerable to drift,

which cause large position estimation errors. Magnetic compasses are efficient sensors when only the magnetic field of the Earth is present, otherwise it loses its functionality. Active beacons, as GPS, are very efficient and accurate, however it is very expensive due to its external equipments. IMU are very used nowadays, due to its low cost and considering it has 6 DOF information. Visual Odometry is also very used and its number of supporters is growing nowadays. It is a cheap system that requires only a camera and is accurate when using robust algorithms.

Visual Odometry requires analysing frames, comparing them, and estimate the motion performed by the robot. In each frame, several features (image patterns) are detected for later comparison between frames. The features detection is very importance for the good performance of the VO process. Therefore, there are many feature detectors in the literature, such as: Moravec [Mor80], Harris [HP88], FAST [RD05], among others. In order to estimate a good motion model, the features from the frames need to be matches, and it is important to remove the outliers that contaminate data. RANSAC is an iterative algorithm available in the literature, for this purpose. A motion model of the robot is then is estimated with the inliers.

Chapter 3

Visual Odometry and Mapping

This chapter illustrates the VO solution proposed in this dissertation. A perspective camera projection is used for the VO system. Picture frames are taken, with a certain tested resolution, for odometry analysis. The FAST corner detector is used to extract corners from these frames. FAST is detailed along with the differences between several *FAST-n*, varying *n*. For corner matching between frames, this work uses the common iterative method RANSAC. However, using RANSAC alone for corner matching would bring a high temporal complexity. Therefore, after detecting the corners in each frame, it is presented how corner signatures are created in order to allow the corner matching between frames. The algorithm FMBF is carefully detailed in order to prove the decrease of such complexity. Using the frames and the robot locations, a map is performed throughout the path. The camera calibration is performed using several pictures of a certain plane, with SI metrics written on it. The corner signature and picture frames simulators are used to validate the FMBF algorithm.

3.1 A Visual Odometry approach

3.1.1 Introduction

As mentioned previously, VO is the process of estimating the egomotion of a rigid body using only video input. The VO proposed approach consists on estimating the egomotion of a robot using a *monocular* scheme for video input. The robot is aimed to work on a

horizontal surface only, in an indoor environment, with a singular and perspective camera rigidly attached and pointed to the ceiling. The ceiling and the floor are both parallel planes, and are the means of interaction of the robot with the world. Considering the robot is wheeled and the camera has no proper motion, its image projection is also a plane, parallel to the ceiling.

Figure 3.1 presents a perspective view of the robot motion in one dimension. The ceiling is represented as a black line pattern, scanned by the camera (grey) with a field of view represented by the light green triangle. The body of the robot is shown as blue rectangle. The red arrow indicates the performed motion.

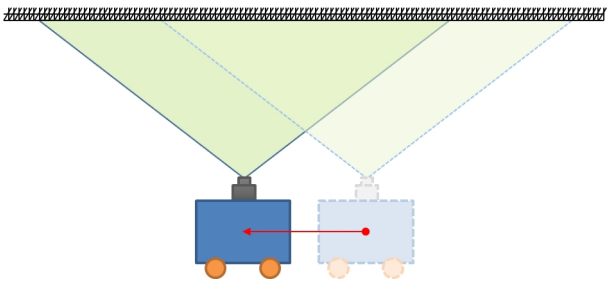

Figure 3.1: Perspective view of the robot, performing a one dimension movement.

While the robot moves, image features are extracted and matched throughout the picture frame sequence, enabling the estimation of the robot egomotion on the floor. There is no feature *triangulation* and the full process is performed in 2D data. Hence, this system produces a 3 *Degrees of Freedom* (DOF) pose of the robot (x, y, yaw).

Figure 3.2 presents an example of two successive picture frames taken from a ceiling.

The first frame corresponds to the green rectangle and the second frame to the blue rectangle. The second frame is taken after a movement that changes all the coordinates of the 3 DOF. The red shaded region is the area common to both frames. It is in this area that extracted features from both frames are matched and, through calculations, conceive the estimation of the frames geometric transformation. That transformation represents the position and orientation of the robot when the second frame is taken, relative to the position and orientation of the robot when the first frame is taken. In other words, it is the estimation of a segment of motion. Therefore, by taking a sequence of several frames, several segments of motion can be estimated and enchained, creating a full trajectory of the robot.

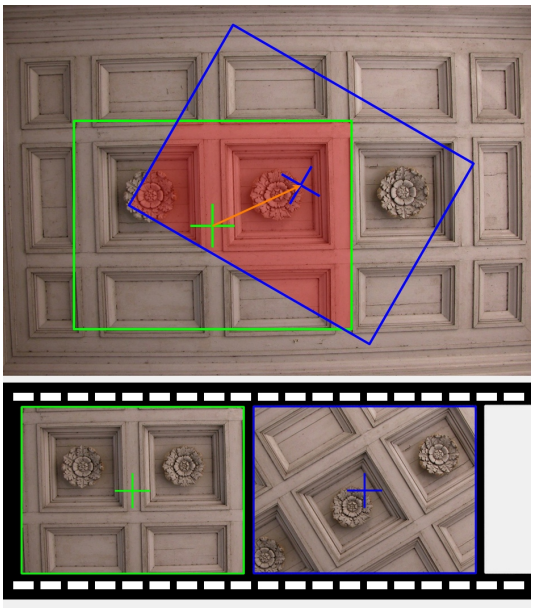

Figure 3.2: Example of two successive picture frames taken to a certain ceiling. Top: geometric dispersions of both frames of the ceiling. Bottom: collected frame data.

3.1.1.1 Context

This VO approach uses a feature based type method. This means, features are extracted from each frame of the sequence, for egomotion estimation purpose [NNB04, CSS04]. Other authors use featureless motion estimation [MGD07]. Instead of detecting features, the harmonic 2D Fourier transform is applied in all the pixels from the two frames, in order to compute the relative motion. In feature based methods, features point out landmarks of the environment that are repeated over frames. When a robot movement is performed, there are landmarks that show up in the current frame and in the previous frame, in different locations. The pixel coordinates of the same landmark are known in both frames. Therefore, considering landmarks are physically static in the environment, through simple kinematics, the transformation matrix is calculated.

The point features, such as corners or blobs are effective because of their accurate position measurements in the frames. FAST feature detector [RD05] is the feature detector technique used in this work. This tool provides corner features with high efficiency and repeatability level. Blobs are more distinctive than corners and better localized in scale. Nevertheless, corners are faster to detect, and in this work no scale variance is used, because the robot moves through a plane (the floor) and collects features from another plane (the

ceiling). Therefore, the distance between the planes is constant and it is assumed to be known in advance.

For feature correspondence in different frames, some authors use feature tracking method [Mor80, SLL01, MCM07], while others use feature matching [NNB04, Low03, How08, CECV14]. Feature tracking consists in detecting features in one frame and track them in the following frames, based on local search techniques. This method is recommended for small-scale environments, where motions between frames are short (it was used in the first VO researches). Feature matching consists on detecting and matching features independently based on similar metrics. This method is currently more employed, considering the increasing research on large-scale environments, and it is followed in this work.

After extracting corners from frames, the available metrics are *corner coordinates* and *corner symmetry*. For the corner matching method to work, a set of algorithms is used, to obtain extra corner metrics. Those metrics are CS which are divided by *distance* (*Corner Distance Signature*, CDS) and *triangle* (*Corner Triangle Signature*, CTS). Also, CTS is split in angle (*Corner Triangle Angle Signature*, CTAS) and distance (*Corner Triangle Distance Signature*, CTDS). CDS is a set of all the Euclidean distances (in pixels units) between the other corners of the frame and the corner respective to the signature. CTAS is a set of angle amplitudes of triangles, made by the respective corner and nearby chosen neighbours. CTDS is a set of pairs of distances made by the respective corner and the neighbours of the same triangle which performs CTAS. The corner symmetry describes if the respective corner is either much brighter or much darker than the arc of pixels from FAST detector. See section 3.1.3 and 3.1.4 for the explanation of these concepts.

False corner associations, called *outliers*, contaminating data is an obstacle that needs careful attention. To achieve corner matching, the outliers need to be removed for a posterior correct motion estimation. *RANdom SAmple Consensus* (RANSAC) is an accurate algorithm for this task [FB81]. However, a large number of outliers are involved after corners detection, increasing significantly RANSAC computational cost. Therefore, a new matching method was developed to remove most of the outliers. This method is here denominated as *Feature Metrics Best Fit* (FMBF). Similar to a *Nearest Neighbour* classifier

(NN) methodology [Bha10], FMBF finds the best fit of feature metrics for the corners that are extracted from consecutive frames. Section 3.1.5.2 describes this process in detail.

A block diagram in figure 3.3 represents the VO pipeline in this case study.

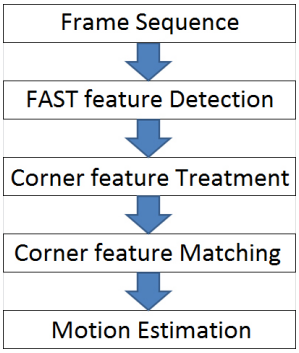

Figure 3.3: VO block diagram.

3.1.2 Picture frame capture

Picture frames are captured by a perspective camera model. Other authors with 3D approaches use omnidirectional camera models [GD00]. Such camera models have a field of view beyond 180 degrees, and could be useful for 3D mapping, considering the wide amount of the environment information at once.

Experimental results use a frame resolution of 320×240 pixel (i.e., 320 of width and 240 of height). Using this FAST approach (detailed in section 3.1.3) the experiments of this work prove that higher resolutions are not viable. Corners repeatability suffers a substantial decline and time consuming rises. For 3D approaches, where scale-variance is unavoidable, this problem is managed with *FAST - Enhanced Repeatability* (FAST-ER) [RPD10].

The ceiling observed by the camera is assumed to be a plane, with minimal roughness. There are no pre processing algorithms for image noise, camera shacking or blur treatment.

The frames sampling rate changes with the algorithms time consumption. For different algorithms combinations and adjustments the sampling rate varies.

Figure 3.4 shows the projection of a perspective camera model. The scene plane is measured in SI units and the image plane is measured in pixels. Section 3.3 describes the conversion between those units.

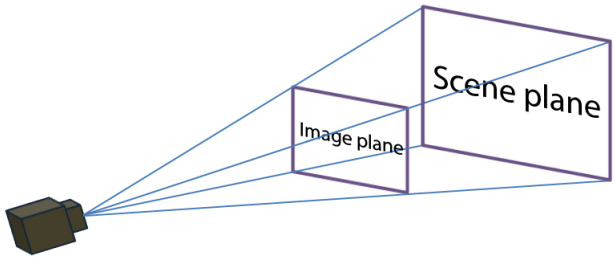

Figure 3.4: Perspective projection: scene plane (SI) and image plane (px) .

3.1.3 FAST feature detection

The FAST feature detector is a method that detects corner features in frames. There are several definitions of corners such as the result of geometric discontinuities, or small patches of texture. FAST computes a corner response and defines corners to be large local maxima of such response. This method requires frames with one colour channel, meaning grey scale intensity is considered on pixels. In order to test if a pixel is elected as a corner, a circle of 16 pixels (with a radius of 3) around a candidate corner p is analysed. The circle $L \in \{1, ..., 16\}$ with locations $l \in L$ consists on pixels positioned relative to p, denoted by $p \to l$. A pixel $p \to l$ can be darker, brighter or similar to p, according to the difference between its intensity $I_{p\to l}$ and p intensity I_p, by a threshold t. These three states are presented as:

$$S_{p\to l} = \begin{cases} d, & I_{p\to l} \leq I_p - t \quad \text{(darker)} \\ s, & I_p - t < I_{p\to l} \leq I_p + t \quad \text{(similar)} \\ b, & I_p + t \leq I_{p\to l} \quad \text{(brighter)} \end{cases} \tag{3.1}$$

The locations of an arc of n contiguous pixels, inside the circle, is denoted by $A = \{1, ..., n\}$, and each pixel location by $a \in A$. If the circle around p contains an arc $p \to A$ with all pixels darker then $I_p - t$ or brighter then $I_p + t$, p is considered a corner. Otherwise, no corner is considered. Let P be the set of all pixels in a frame. Three subsets of P are denoted as P_d, P_s and P_b, where:

$$P_d = \{p \in P, \forall a \in A : S_{p\to a} = d\}, \tag{3.2}$$

and P_s and P_b are defined similarly. P_d is the set of pixels called negative corners, P_b is

the set of pixels called positive corners, and P_s are non corners.

FAST was originally made only for $n = 12$ arc size (FAST-12) [RD05]. Compared to other known corner detectors, such as DoG [Low04], SUSAN [SB97], Harris [HP88], Shi-Tomasi [ST94], FAST-12 is faster, but its repeatability is not higher. Years later, approaches of FAST-n [RPD10] were studied, with $n \in \{9, ..., 16\}$, and results prove that FAST-9 is the most efficient of FAST-n.

Figure 3.5 illustrates a 9 point segment test corner detection in an frame patch. The highlighted blue squares correspond to the 16 pixels of the circle, used in the corner detection. The pixel at p is the centre of a candidate corner. The arc is presented by the dashed green line that passes through 9 contiguous pixels.

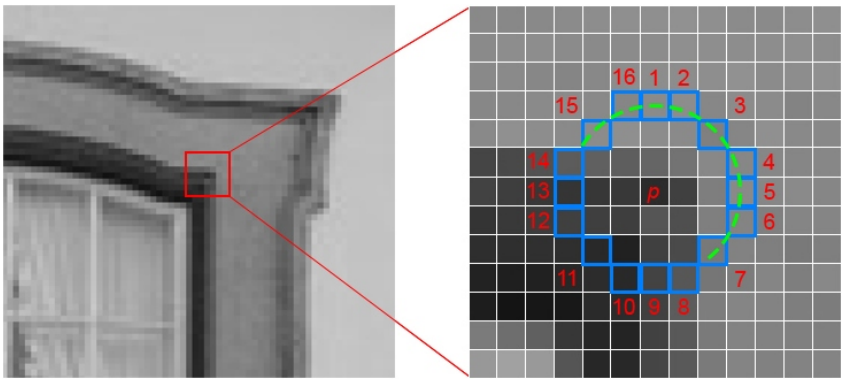

Figure 3.5: FAST feature detection in a frame patch.

Let Q be a subset of locations of L, $Q = \{1, 5, 9, 13\}$, shown in figure 3.6. To determine if p is a corner, the pixels $p \rightarrow Q$ are examined first. If $S_{p\rightarrow 1}$ and $S_{p\rightarrow 9}$ are both s, then p can not be a corner. If p can still be a corner, pixels $S_{p\rightarrow 5}$ and $S_{p\rightarrow 13}$ are examined similarly. This is a quick rejection that makes the algorithm faster.

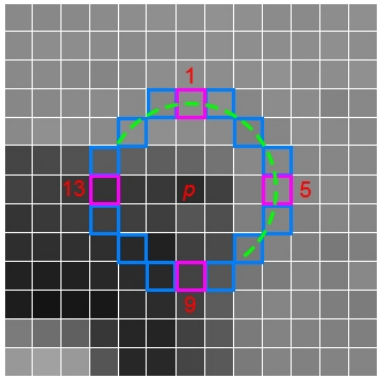

Figure 3.6: Frame patch and the pixels with Q locations.

For a certain n, the number of detected corners decreases with t increasing. Hence, it is important to use an appropriate t to regulate the number of detected corners, depending on prior purposes. When a corner is found there are two implemented approaches. The faster one is to test each pixel of $p \rightarrow L$ in sequence, and stop when an arc $p \rightarrow A$ is found. The other approach is to keep testing the remaining $p \rightarrow L$ after $p \rightarrow A$ is found, for further analyses. With the complete information of $p \in \bar{P}_s$ corner circles in one frame, it is possible to control the number of detected corners on the next frame. This process is explained in section 3.1.7.1 in detail.

Figure 3.7 shows two successive frames (distinguished by different robot position) and the point corners at red (negative corners) and yellow (positive corners), extracted from FAST-9 with a threshold $t = 22$.

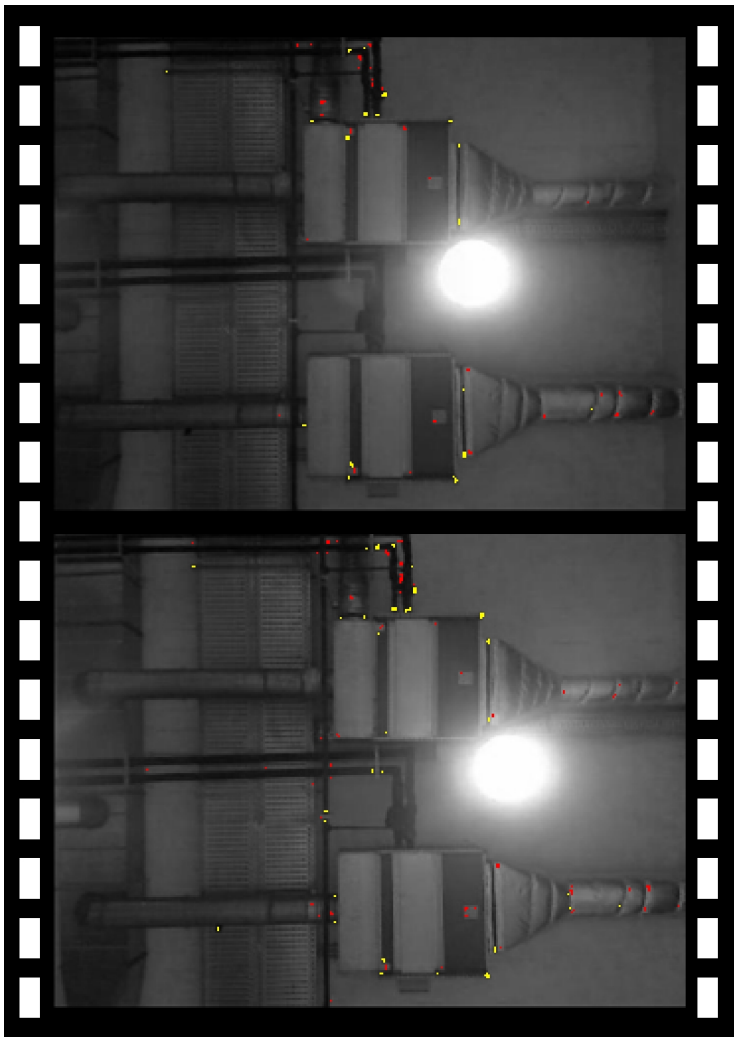

Figure 3.7: Corner points extracted from FAST-9 with $t = 22$. Top: 108 corners. Bottom: 165 corners.

By decreasing the threshold to $t = 18$, figure 3.8 presents the increase of the extracted corners, using FAST-9.

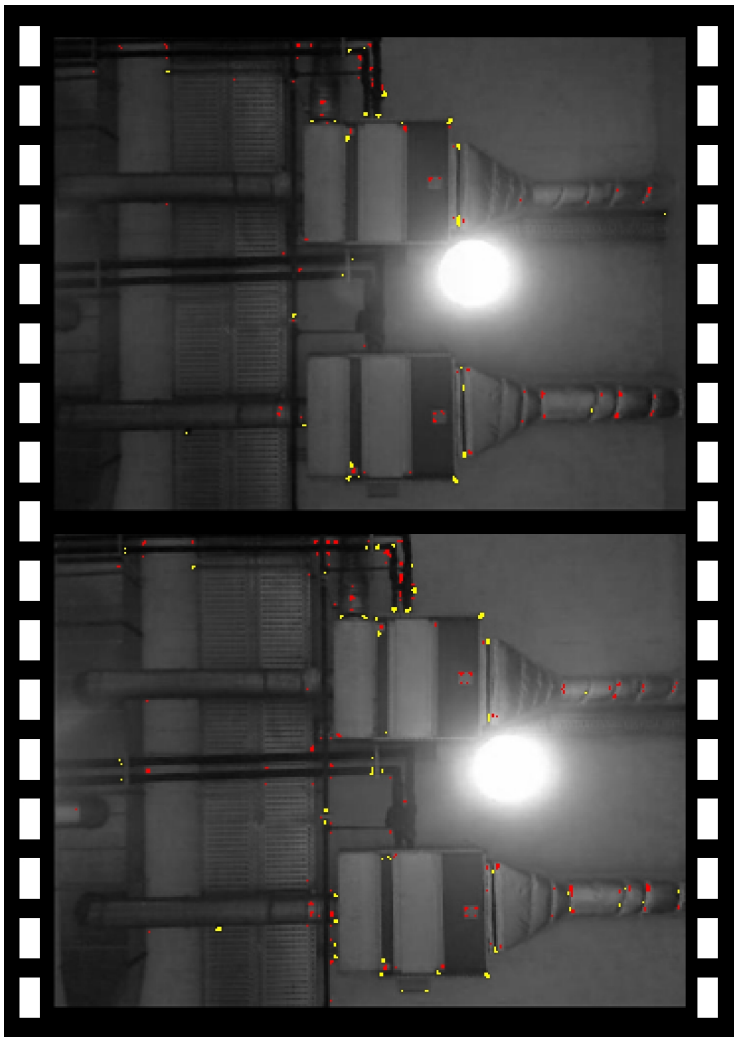

Figure 3.8: Corner points extracted from FAST-9 with $t = 18$. Top: 222 corners. Bottom: 375 corners.

The different number of extracted corners between two successive frames occurs due to two possibilities. First, due to image noise or FAST detection imperfections (i.e., repeatability uncertainty). And second, due to the motion of the image projection, as figure 3.9 illustrates. The red corner points are extracted in both frames. The other corner points are extracted only in the respective colour frame.

This corners variation in successive frames brings a problem for feature matching. A detailed justification is presented in section 3.1.5.

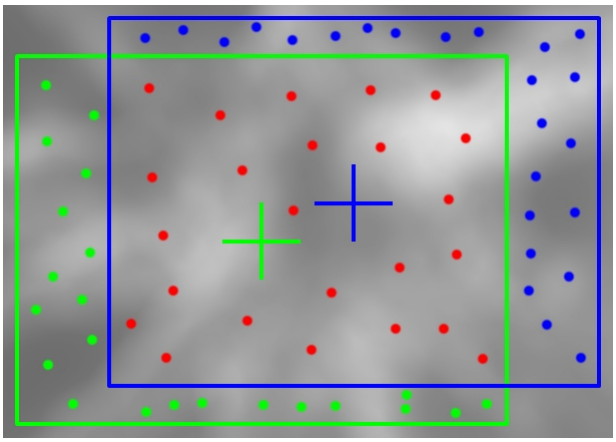

Figure 3.9: The problem of disappearing corners between frames due to the robot motion.

3.1.3.1 Frame resolution dependency

The smallest unit of digital images are pixels. Pixels are unsplittable, and shaped as squares. As a consequence, usually image transformations change the information of the original image pixels (except for rotations with angles multiples of 90°, and translations with integer displacement values, where the full image is maintained). The following equation describes how a transformation is performed, for every pixel in two images, where (x, y) are the pixel coordinates of the original image and (x^*, y^*) are the pixel coordinates of the transformed image. The rotation angle is φ and, T_x and T_y is the translation displacement in its respective axes.

$$\begin{bmatrix} x^* \\ y^* \\ 1 \end{bmatrix} = \begin{bmatrix} cos(\varphi) & -sin(\varphi) & T_x \\ sin(\varphi) & cos(\varphi) & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.3}$$

To create the human illusion that the original image is preserved, interpolations may be applied, such as nearest-neighbour, bilinear, biquadratic, among others. If images has high resolution and the current zoom makes pixels too small to be individually noticed, that illusion works. In return, at a pixel scale, the differences in both images are visible for humans, and countable for computers.

This specific image translation is not virtual but physical (camera translation). Nevertheless, it complies with the same principles regarding the loss of pixel information.

In images with low resolutions, a transformation does not affect pixel colour patterns

as much as it affects in images with higher resolutions. Colour patterns are more likely to remain the same, considering pixels are large enough to tolerate the image transformation. Another reason is that those pixels represent a larger portion of the world, whereas one pixel of low resolution has the colour of the average of several pixels of high resolution, which makes them less vulnerable to Gaussian noise.

With the FAST circle test segment of size 16 pixels, colour patterns are likely to remain the same for certain resolutions. However, further explanations demonstrate that the lower the resolution, less accurate this system is. As mentioned earlier, experiments show the resolution needs to be 320×240 pixel. Figure 3.10 shows how Gaussian noise and a perspective rotation may interfere in corner detection of the same location, for an experiment with a 640×480 pixel resolution.

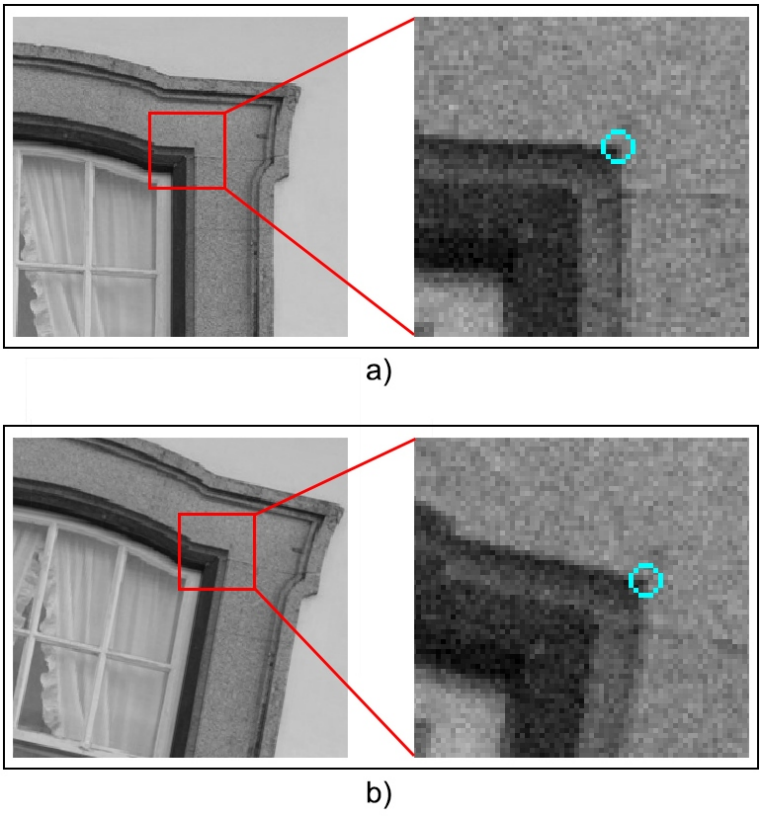

Figure 3.10: Corner detection interfered by Gaussian noise and perspective rotation. In b) a rotation is performed from a).

Observing the figure, in the right side of each of the both plots, many random pixel patterns are observed in areas where only one colour should be seen. This effect results from the camera Gaussian noise. Both of the blue circles represent the segment test that

should validate the middle pixel point. However, the noise distorts the tests which leads to different output in both cases. This means the probability of classifying both middle pixel points as the same corner is low. The figure also demonstrates a perspective rotation, between (a) and (b), that disrupts edges natural patterns. Figure 3.11 shows a more detailed example of this disruption. An edge pattern on an image, with 1 pixel thick, is presented before (a) and after (b) a perspective rotation.

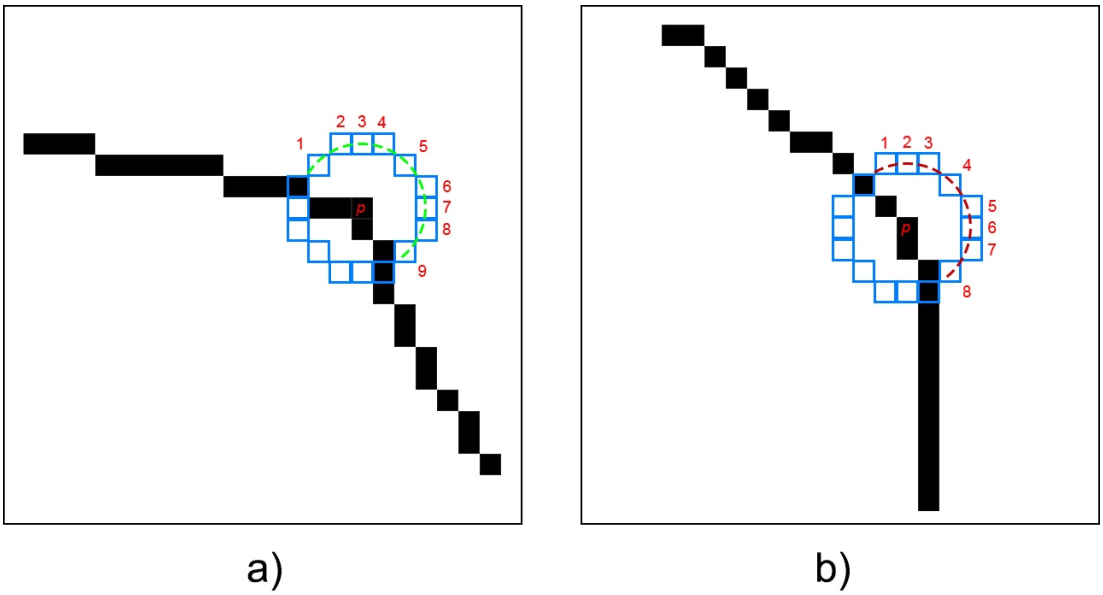

Figure 3.11: Influence of a perspective rotation in corner detection with high resolution. In b) a rotation is performed from a).

In the figure, both plots have segment test masks represented by 16 blue squares. The edge in (a) has a valid corner p, as an arc of 9 pixels $p \rightarrow A$ is observed. When taking a picture of that edge after a rotation, the picture of the same edge could be obtained has shown in (b). In (b) the edge is distorted, compared to (a), preventing the existence of an other arc $p \rightarrow A$, leading to no corner detection. In conclusion, this is why the corner repeatability is highly affected for hight resolutions.

3.1.4 Corner features pre-processing

Before matching the corner features, corner signatures are created from corner coordinates extracted from FAST. This is requisite because FMBF works by comparing corner signatures.

3.1.4.1 Corner Distance Signature

A corner distance signature is a sorted set of all Euclidean distances between a corner point and all other corner points from the same frame. Through the set of corner coordinates in one frame, all corner distance signatures are calculated using the Pythagorean theorem. Figure 3.12 represents a corner distance signature (blue lines) of a corner indicated as an orange cross.

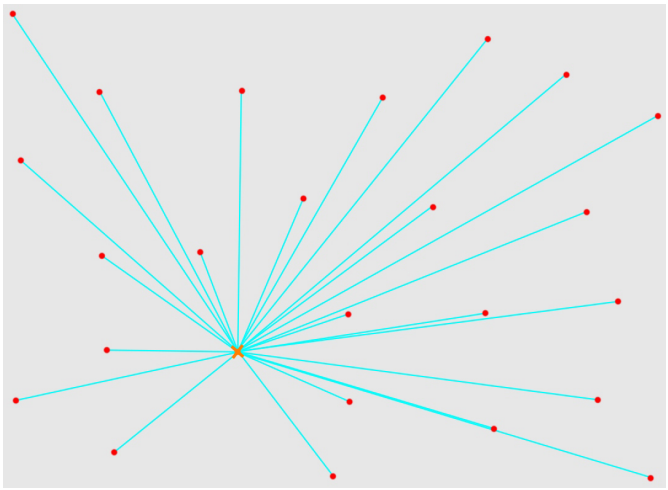

Figure 3.12: Representation of a corner distance signature (blue lines).

The process of calculating N corner distance signatures in the worst case has a $O(N^2)$ (quadratic) cost. Let $W = \{p_1, ..., p_N\}$ be the set of all corner coordinates where $p_k \in \mathbb{Z}^2$ is the k^{th} corner point on the frame. The cost can be reduced to less then a half, considering the distances between p_a and p_b and between p_b and p_a are the same. Therefore, this process is performed as illustrated in figure 3.13, for $N = 7$. The p_k corner points are represented by grey squares, identified by its k index, and the necessary calculations are represented by the blue lines.

The number of necessary iterations are $It(N) = \sum_{n=1}^{N-1} n$. Simple deduction leads to the following equation:

$$It(N) = \frac{N \times (N-1)}{2} \tag{3.4}$$

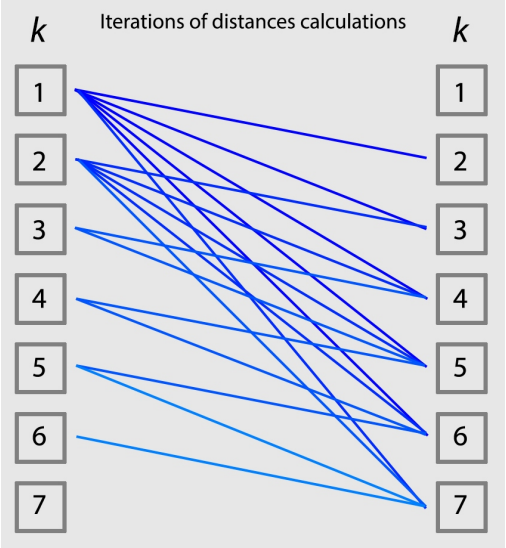

Figure 3.13: Necessary iterations for the calculation of 7 corner distance signatures.

3.1.4.2 Corner Triangle Signature

In a corner triangle signature, a CTAS is a set of angle amplitudes Θ, with size L_Δ. The i^{th} angle $\theta_i \in \Theta$ is from a triangle consisting of: the corner point p_k associated to the corner signature and two other neighbouring corner points (corner neighbour a, $p_{k,i,a}$, and corner neighbour b, $p_{k,i,b}$). The angle formed at the vertex p_k of this triangle is referred as the angle θ_i.

A CTDS is a set of pairs of distances, Φ, also with length L_Δ. The i^{th} pair of distances $\{\phi_{k,i,a},\ \phi_{k,i,b}\} \in \Phi$ corresponds to the distances between the corner neighbours and the corner in concern, where $\phi_{k,i,a}$ is the distance between p_k and $p_{k,i,a}$, and $\phi_{k,i,b}$ is the distance between p_k and $p_{k,i,b}$.

For every CTS in a match between two frames, there is a minimum distance for neighbours acceptance t_Δ. Figure 3.14 illustrates a corner concerned to the respective CTS and several corner neighbours. Within the red circle, with radius t_Δ, the corner neighbours are not accounted. The rest of the red points (where $L_\Delta = 3$) correspond to the neighbours where calculations are performed to obtain Θ and Φ.

All of the L_Δ triangles have the vertex of p_k in common, but each of them are made with different neighbours. The index i corresponds to the ascending order of the distances between each pair of neighbours and p_k. That is, θ_1 corresponds to the triangle of the nearest pair (first) of neighbours, θ_2 to the second, and so forth.

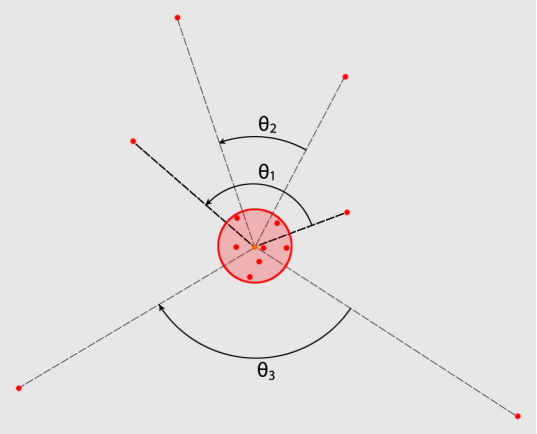

Figure 3.14: Three angles of a corner triangle signature.

3.1.5 Corner features matching

Feature matching is the process of finding the correspondence for each feature in consecutive frames (if possible). Generically, two consecutive frames, the first with N_α and the second with N_β number of corners, have $MC = N_\alpha \times N_\beta$ matches combinations. In other words, MC is the number of all the possible matches for all corners, between two frames. Figure 3.15 gives the notion of the tremendous amount of matches combinations only for 11 corners in each frame ($MC = 121$). Red points are corners and blue line are representations of the possible match combinations.

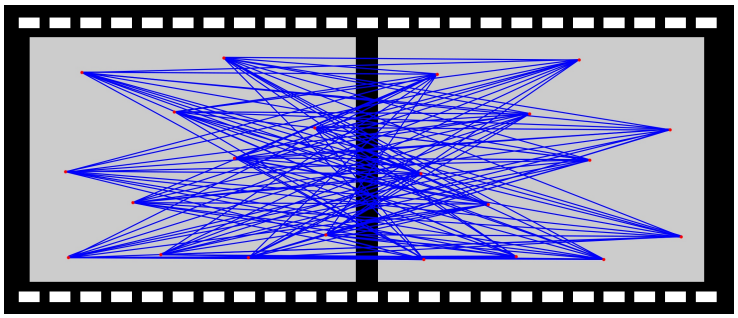

Figure 3.15: Abstract case of matches combinations.

FAST reduces this number, considering that corners are separated by corner symmetry (negative and positive ones). The corner matches combinations are then:

$$MC = N_{\alpha,d} \times N_{\beta,d} + N_{\alpha,b} \times N_{\beta,b} \tag{3.5}$$

The number of negative corners in frame α is $N_{\alpha,d}$, and the others are defined similarly.

As previously illustrated, d notation is for negatives (darker) and b for positives (brighter). Obviously $N_\alpha = N_{\alpha,d} + N_{\alpha,b}$ and $N_\beta = N_{\beta,d} + N_{\beta,b}$. Figure 3.16 presents the reduction of the combinations with corners separated by corner symmetry. Negative corners are shown as red and positive corners are shown as yellow. $N_{\alpha,d} = 5$, $N_{\beta,d} = 5$, $N_{\alpha,b} = 6$ and $N_{\beta,b} = 6$. In this example, the corner matches combinations are reduced approximately to the half, $MC = 61$.

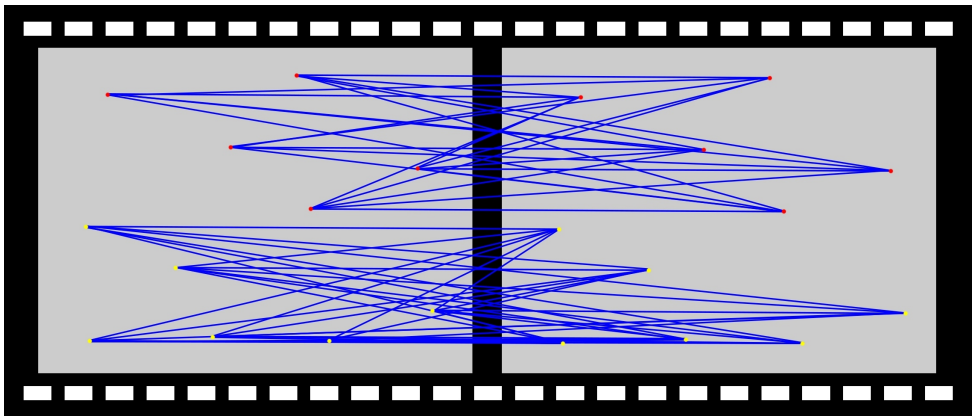

Figure 3.16: Matches combinations divided by corner symmetry.

However, the intended corner matches are presented in figure 3.17. In this example, those 11 matches are the matches that need to be found by any matching method. This means, in 61 possible cases, 50 are outliers (that need to be rejected) and 11 are the wanted inliers.

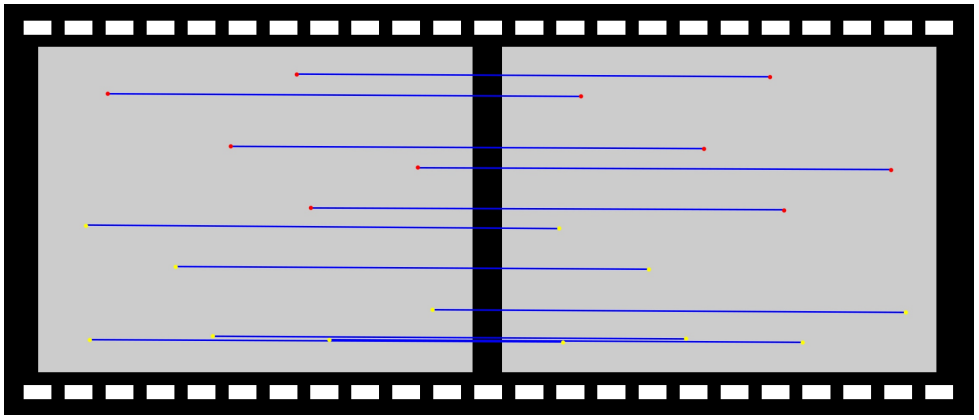

Figure 3.17: Correct intended corner matches.

For simplicity, the previous artificial example has 100% corner repeatability between frames, only a few corners are presented, positive and negative corners are separated

physically, the corners are homogeneously distributed and no image motion (rotation or translation) is performed.

Figure 3.18 presents an example which differs in a rotation and in the existence of non repeated corners, compared to the previous example.

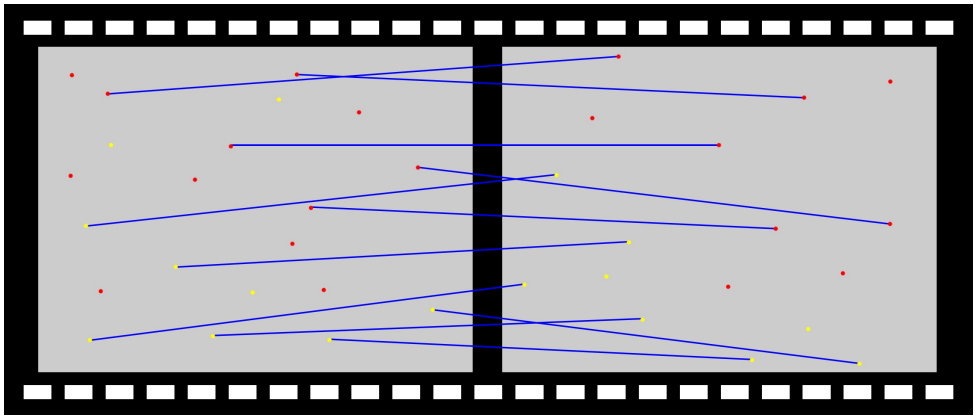

Figure 3.18: Correct matches with non repeated corners.

Let M_{in} be the number of inliers, M_{out} the number of outliers (between frame α and β); N_r the number of repeated corners and $N_{\alpha,nr}$ the number of non repeated corners, in frame α (similarly to frame β). The values of M_{in} and N_r are always equal. Their difference lies on notation: M_{in} for number of corner matches and N_r for number of corners.

Table 3.1 presents the number of corners (by their characteristics) and the matches of the previous example.

Table 3.1: Number of corners and matches of an artificial example.

$N_{\alpha,d}$	$N_{\alpha,b}$	$N_{\beta,d}$	$N_{\beta,b}$	$N_{\alpha,nr}$	$N_{\beta,nr}$	N_r	N_α	N_β	M_{in}	M_{out}	MC
7	9	8	7	5	4	11	16	15	**11**	**108**	119

The portion of outliers is approximately $\eta_{out} = 91\%$, whereby inliers are only $\eta_{in} = 9\%$.

Generically, the repeated corners portion of frame α is $\eta_{\alpha,r} = \frac{N_r}{N_\alpha}$, the portion of inliers is $\eta_{in} = \frac{M_{in}}{MC}$ and the portion of outliers is obviously $\eta_{out} = \frac{MC - M_{in}}{MC}$.

In the best case scenario, positive and negative corners are divided equally: $N_{\alpha,d} =$

$N_{\alpha,b} = \frac{N_\alpha}{2}$, $N_{\beta,d} = N_{\beta,b} = \frac{N_\beta}{2}$. Hence, using equation 3.5,

$$\begin{aligned} MC_{bc} &= N_{\alpha,d} \times N_{\beta,d} + N_{\alpha,b} \times N_{\beta,b} \\ &= \frac{N_\alpha \times N_\beta}{2} \qquad \text{(best case)} \end{aligned} \tag{3.6}$$

For the sake of simplicity let's consider both frames with equal number of corners:

$$N = N_\alpha = N_\beta,$$

Consequently:

$$\eta_r = \eta_{\alpha,r} = \eta_{\beta,r},$$

The number of combinations for both case scenarios are:

$$MC_{bc} = \frac{N^2}{2} \quad \text{(best case)} \qquad \text{and} \qquad MC_{wc} = N^2 \quad \text{(worst case)},$$

This leads to:

$$\begin{aligned} \eta_{out}(N) &= \frac{MC_{bc} - M_{in}}{MC_{bc}} \\ &= \frac{\frac{N^2}{2} - N_r}{\frac{N^2}{2}} \\ &= \frac{\frac{N^2}{2} - \eta_r \times N}{\frac{N^2}{2}} \\ &= 100\% - 2 \times \frac{\eta_r}{N} \qquad \text{(best case)} \end{aligned} \tag{3.7}$$

And

$$\begin{aligned} \eta_{out}(N) &= \frac{MC_{wc} - M_{in}}{MC_{wc}} \\ &= \frac{N^2 - N_r}{N^2} \\ &= \frac{N^2 - \eta_r \times N}{N^2} \\ &= 100\% - \frac{\eta_r}{N} \qquad \text{(worst case)} \end{aligned} \tag{3.8}$$

Figure 3.19 presents a plot with the outlier portions η_{out} depending on the number of corners N, for different five instances. The graph shows how outliers behave varying the repeatability portions, 50%, 80% and 100%. It also shows how the worst (WC) and the best case scenarios (BC) affects the outlier quantities.

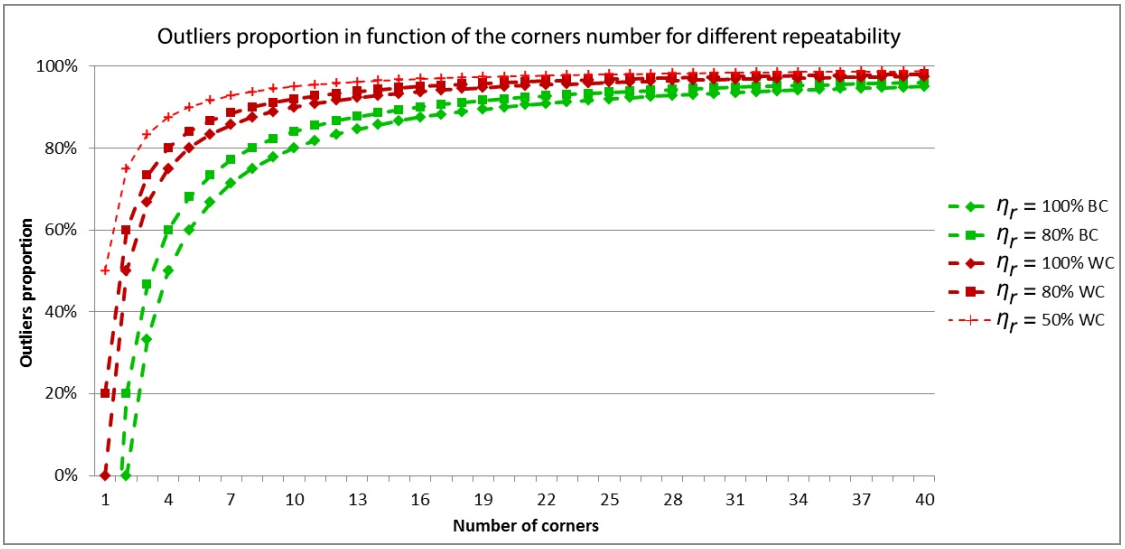

Figure 3.19: Graph of the outliers portion in function of the corner number for different repeatability.

3.1.5.1 RANSAC

RANSAC (RANdom SAmple Consensus) is an iterative method used to reject outliers. It estimates the parameters of the mathematical model of the inliers contained on a set of observed data, contaminated with outliers. RANSAC is non-deterministic given the fact that it produces reasonable results with a certain probability, which increases as more iterations are allowed.

Generically, the algorithm selects data points randomly to estimate the model parameters. The model is evaluated by the number of inliers that fits in it. A point is considered an inlier if the distance from the point and the model is within a threshold. Many models are tested this way It times (iterations). In the end the model with the largest number of inliers is selected and a more accurate model based on all the inliers is recomputed. The used algorithm is specifically presented in Algorithm 3.1.

The input parameters of this algorithm are:

D - a set of data matches;

υ - the minimum number of data matches required to create the model;

It - the number of iterations performed by the algorithm;

t_{md} - a threshold value for determining when a corner match fits a model;

```
Input  : D, υ, It, t_md
Output: BM, BCS
It ⇐ 0
BM ⇐ null
BCS ⇐ null
while k ≤ It do
    CS ⇐ RandomSelectData(D, υ) // Selects υ random matches from D
    MM ⇐ TransformationModel(CS) // Estimates the model of CS
    for i ⇐ 1 to Length(D) do
        if Contains(CS, D(i)) = 0 then // If match of D is not in CS
            if Distance(MM, D(i)) < t_md then // If match fits model
                CS = Concat(CS, D(i)) // Adds match to CS
    if Length(CS) > Length(BCS) then
        BCS = CS
        BM = MM
BM ⇐ TransformationModel(BCS) // Estimates the model of BCS
```

Algorithm 3.1: RANSAC used algorithm.

In each iteration, a *consensus set* (CS) is randomly generated with υ matches from the data. A *maybe model* (MM) is created from this CS. Then, for every match in data not in CS, if match fits MM with a distance smaller than t, that match is added to CS. This distance is calculated based on motion calculus which is explained in detail in section 3.1.6, as well as the use of plausible values for the threshold t_{md}. The *best consensus set* (BCS) and the *best model* (BM) is updated in each iteration if a new model is found with more inliers then the previous *best model*. In the end BCS is considered to have all the inliers of D, therefore BM is estimated with BCS for a more accurate model. In this

work, the model used in RANSAC (MM and BM) is the matrix rigid body transformation $H_{k,k-1}$, of equation 2.3.

The number of iterations of the algorithm It is selected so that the probability ρ of selecting at least one sample with all inliers is high enough. In this context, a sample is a set of υ number of matches. In the literature, and in this work, it is used $\rho = 0.99$. The authors of RANSAC [FB81] state that It is calculated by the following:

$$It = \frac{log(1-\rho)}{log(1-(1-\eta_{out})^{\upsilon})} \tag{3.9}$$

where η_{out} is the probability of selecting an outlier and υ the minimum number of data matches required to create the model, as mentioned previously.

To create a model for this VO approach, the minimum of υ is 2. Section 3.1.6 presents an explanation in detail.

Analysing carefully the Algorithm 3.1, the first instruction in each iteration is crucial to the performance of the iteration. When all the matches in a sample are inliers a correct *maybe model* is created. Considering, in the *for* loop, each data match is verified if it fits in the model, more inliers will be found, and the model quality will increase. Therefore, υ needs to be as low as possible so that the probability of choosing υ inliers is high. Considering that the probability of finding an outlier match is as stated by equation 3.8 for the worst case, finding an inlier is $\frac{\eta_r}{N}$ probable. With $\upsilon = 2$, the probability of finding 2 inliers when selecting random matches is much lower $(\frac{\eta_r}{N})^2$. This is related to the fact that It grows exponentially with υ in equation 3.9.

Using the plot of figure 3.19, for $N = 20$ in the best case, $\eta_{out} = 0.9$. This means $It = \frac{log(1-0.99)}{log(1-(1-0.9)^2)} = 458$. To ensure that the corner repeatability is high enough, N is usually aimed to 30, which makes It exponentially higher than with $N = 20$.

Using equation 3.9 and making It depending on N, figure 3.20 shows the graph of $It = \frac{log(1-\rho)}{log(1-\eta_{out}(N)^{\upsilon})}$, in order to understand the behaviour of the number of iterations while increasing the number of corners. Worst and best cases are presented with a common repeatability of 70%.

This exponential behaviour is a serious problem for computational cost. Hence, in order to reduce the number of match combinations MC, other methods should be used

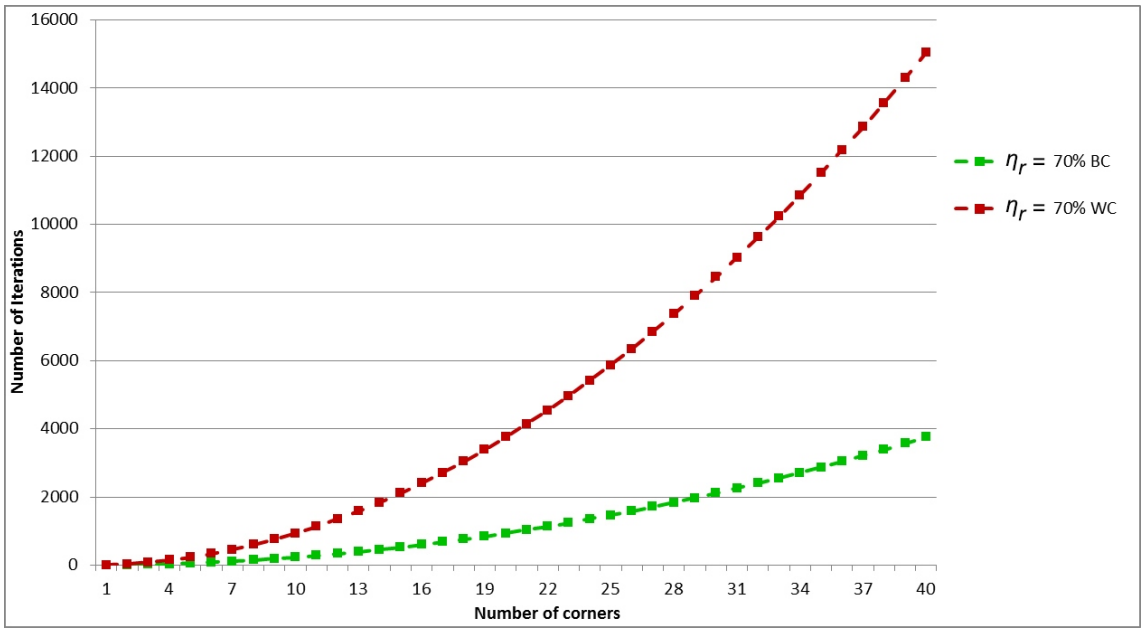

Figure 3.20: Graph of the number of iterations in function of the number of corners.

prior to RANSAC.

3.1.5.2 Feature Metrics Best Fit

Feature Metrics Best Fit (FMBF) is a method developed in this work, whose purpose is to find a match between two frames with the best fit of the corner signature metrics. The probability of a corner distance signature (CDS) being unique for a certain corner is very high, and it increases with the number of corners per frame. Therefore, the first approach of this method consisted in comparing all CDS of frame α with all CDS of frame β, separately for negative and positive corners. This is highly accurate, but it also has a high computational cost, considering comparing two CDS is already $O(N)$ in the best case scenario. Therefore, in order to reduce the number of CDS comparisons, a new option is inserted. The frame match *could* finalize when a certain number of accurate corner matches t_{ma} is approved. For an accurate admission of the best matches, several stages may be tested. Those stages are applied to each corner match as presented bellow:

1. Comparison of CTAS, the set of angles Θ;

2. Comparison of CTDS, the set of pairs of distances Φ;

3. Comparison of CDS;

4. Admission of the best matches.

Notice that when mentioning corner comparisons, they are always perform between two distinct frames.

Unlikely to CDS that is meant to approve corner matches, CTS is designed to reject corner matches in order to save computational cost. Hence, later on it is shown that stage 1 and 2 can be used for quick inaccurate corner matches rejection. This prevents such corner matches from wasting time in stage 3, which is the most time consuming stage.

Stage 1 consists on approving a corner match if the set of angles Θ are close between two corners. Let's consider Θ_i the set of angles of the i^{th} corner of frame α, and Θ_j the set of angles of the j^{th} corner of frame β. The set of differences, of the set of angles, is $D_{\Theta,i,j} =\mid \Theta_i - \Theta_j \mid$. A certain corner is approved at stage 1, if all the differences in $D_{\Theta,i,j}$ are lower than a certain threshold t_Θ.

Stage 2 consists on approving a corner match if the set of pairs of distances Φ is not very different between two corners. Let's consider Φ_i the set of pairs of distances of the i^{th} corner of frame α, and Φ_j the set of pairs of distances of the j^{th} corner of frame β. To perform a Φ comparison, it is necessary to have a set of pairs of differences, which corresponds to the differences between Φ_i and Φ_j, i.e., $D_{\Phi,i,j} =\mid \Phi_i - \Phi_j \mid$. Similarly to stage 1, stage 2 is approved, for a certain corner match, if $D_{\Phi,i,j}$ have all differences lower than a certain threshold t_Φ, chosen by the user.

For better performances of stages 1 and 2, it is essential to use neighbours ($p_{k,i,a}$ and $p_{k,i,b}$) with distances higher than t_Δ. This is due to the discrete nature of frame information. In the real world, the points around a middle point have an infinite number of possible positions, regardless the radius. Instead, with circles in digital images, the number of positions is limited by the radius. Figure 3.21 shows an usual way of performing circles with pixels. It is not possible to have all the pixels performing the exact same distance from the middle pixel. Therefore, to perform complete circles, the pixels around the middle pixel need to be positioned where their distances are the most approximately as possible to the intended radius.

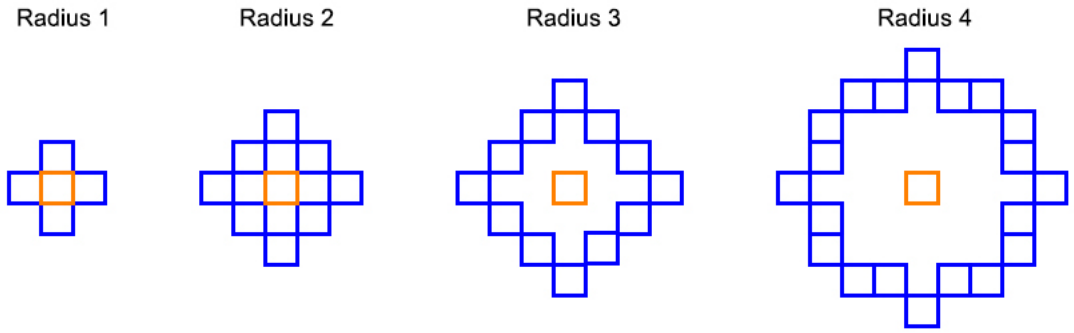

Figure 3.21: Several discrete circles with different radius approximations.

This means, when the distance between corners and their neighbours is 1 pixel, the neighbours can only assume angles multiples of 90°. If that distance is 2 pixels, the neighbours can only assume angles multiples of 45°, and so forth. Hence, with high t_Δ values, the probability of having different corners with the same CTS decreases, which leads to a more efficient rejection. Obviously, t_Δ can not be larger than the frame dimensions, otherwise it is not possible to perform a CTS. Figure 3.22 presents a practical extreme example of this situation. Notice that the image is not a real extraction of corners. Coloured pixels are intentionally placed in the image in order to illustrate a more clear example.

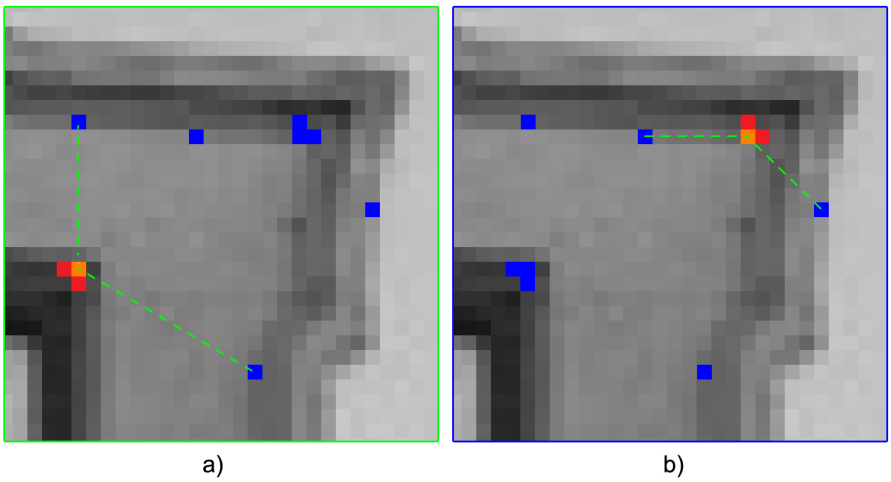

Figure 3.22: An example that leads to a wrong CTS match: a) frame α, b) frame β.

In each plot of the figure, the orange square represents the corner of the CTS in study. If $t_\Delta = 0$ was always used, CTS would be performed with the closest neighbours represented by the red square. Hence, the CTS of the corners in study on both plots would have

$\Theta = \{90°\}$ (it is assumed $L_\Delta = 1$ for simplification). This means different corners with the same CTS. Nevertheless, by choosing an experimental value for t_Δ, for example $t_\Delta = 2$ the CTS of each corner in study would be performed by the blue neighbours indicated by the dashed lines.

It is important to mention that high L_Δ is not viable. A corner triangle signature with many triangles makes them match between corners more difficult to accomplish due to the existence of non repeatable corners. In other words, real corner matches might not be detected because stage 1 or 2 do not approve them due to broken CTS. Figure 3.23 illustrates an artificial example of how a non repeated corner influences CTS.

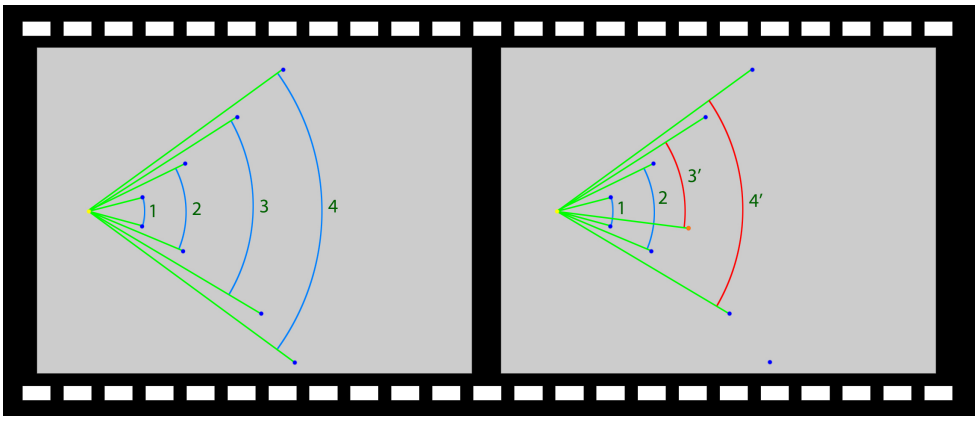

Figure 3.23: Influence of a non repeated corner in CTS.

In each frame, the yellow point represents the corner in study. Its neighbours are as blue points, carefully placed for easy understanding. The arcs attach the neighbours that perform the triangles of CTS. The blue ones exists in both frames, in return the red ones do not. The corner represented at orange shows up only in the second frame, changing the CTS. Due to its location, it changes all the triangles performed with neighbouring corners that are further then the neighbours of triangle 2 (in this example triangle 3 and 4 are affected).

The example of the figure is very simplistic. There is only one non repeated corner and only in one frame. In reality non repeated corners are scattered in the frames. Therefore, to avoid this incident, generally the used L_Δ values are 1 or 2.

In stage 3 CDS comparisons are performed and are analysed by the average of the differences between corner distances (in pixels). That average value represents the quality of a CDS match and is denoted as Q_{CDS} (defined in equation 3.10). Hence, the best matches are the ones which have lower Q_{CDS}. If Q_{CDS} is lower than a threshold t_{CDS}, the match counts as an approved one. Each CDS comparison requires comparing at least t_D (threshold chosen by the user) distances of both of this metrics. If that is not possible, the corner match is discarded.

The calculation of a CDS comparison is not trivial and its details are explained below. Let $S_\alpha = [s_{\alpha,1}, ..., s_{\alpha,N_\alpha-1}]^\intercal$ be a vector of a CDS, of a certain corner, in frame α. For the following explanation it is not necessary to mention a particular corner. The notation S_β is similarly calculated, for frame β.

Ideally, the process of comparing two CDS would be as follows:

$$\begin{aligned} Q^*_{CDS} &= average(|S_\alpha - S_\beta|) \\ &= average\left(\left|\begin{bmatrix} s_{\alpha,1} \\ ... \\ s_{\alpha,N_\alpha-1} \end{bmatrix} - \begin{bmatrix} s_{\beta,1} \\ ... \\ s_{\beta,N_\beta-1} \end{bmatrix}\right|\right) \\ &= average\left(\begin{bmatrix} |s_{\alpha,1} - s_{\beta,1}| \\ ... \\ |s_{\alpha,N_\alpha-1} - s_{\beta,N_\beta-1}| \end{bmatrix}\right) \end{aligned} \tag{3.10}$$

This way, the computational cost for this operation would be always and only linear. However, due to possible corner repeatability lower than 100%, this process becomes more complex. Figure 3.24 shows an example where the same corner in different frames has different CDS, due to non repeatable corners present in the frames.

The corner in analysis is illustrated as yellow points in both frames. The blue and orange lines represent the distances of CDS. The red points are the neighbour corners used to create the signature. Those neighbours are designated by numeration, according to their distance to the yellow corner, in an ascendant order. The corners respective to the orange lines do not exist in the consecutive frames. In return, the corners respective to the blue lines are repeated in both frames. This means all the orange distances could

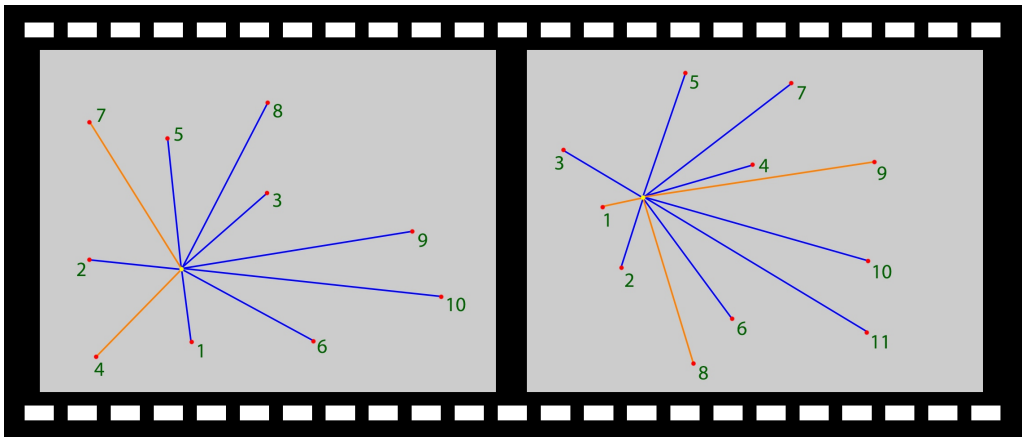

Figure 3.24: Correct matches with non repeated corners.

be called noise in a corner distance signature. Therefore, the orange distances must be discarded so the comparison between each signature could be performed only with the blue distances only. For the sake of abstraction, let's consider a set of distances $\{s_1, ..., s_n\}$ whose indices are attributed according to their assortment ($s_k < s_{k+1}$). Table 3.2 presents the corner neighbours sorted by their distances to the yellow corner, and their correspondences between frame α and β.

Table 3.2: Correspondence of 2 CDS.

Distances	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{12}	s_{13}	s_{14}
Corners α	N/A	1	2	3	4	5	6	7	8	N/A	N/A	9	10
Corners β	1	2	3	4	N/A	5	6	N/A	7	8	9	10	11

The set of repeated distances from frame α is denoted $S_{\alpha,r}$, and set of repeated distances from frame β is denoted as $S_{\beta,r}$. In this theoretical example, $S_{\alpha,r} = \{s_2, s_3, s_4, s_6, s_7, s_9, s_{13}, s_{14}\}$ and $S_{\beta,r} = \{s_2, s_3, s_4, s_6, s_7, s_9, s_{13}, s_{14}\}$, leading to $Q_{CDS} = average(|S_{\alpha,r} - S_{\beta,r}|) = 0$. However, assuming the algorithm discards the non repeated distances correctly, $Q_{CDS} = 0$ happens theoretically only in particular conditions (robot motions with angle rotations multiples of 90°, and translations with integer displacement values). In practice, those conditions are highly unlikely to occur, $S_{\alpha,r} \approx S_{\beta,r}$, and the best corner matches are the ones with the lower Q_{CDS}.

The process of discarding the non repeated distances is explained below. Let $s_{\alpha,i}$ be the i^{th} distance of S_α and $s_{\alpha,j}$ be the j^{th} distance of S_β. Let's call the action of finding

$s_{\beta,j}$, nearest distance to $s_{\alpha,i}$, a *marriage*, and the undo of this action a *divorce*. This means every marriage has an associated difference (between distances) $\Delta s_{i,j} = |s_{\alpha,i} - s_{\beta,j}|$. The mechanism consists on making as many marriages as possible with $\Delta s_{i,j}$ as low as possible. Before a marriage is performed, the distance $s_{\beta,j}$ might be already married. If that is the case, the older marriage is compared to the candidate marriage and the one with lower $\Delta s_{i,j}$ prevails. This means divorce of the older marriage, if the new marriage prevails. In figure 3.25 natural numbers are attributed to the distances of both CDS. This leads to unreal differences $\Delta s_{i,j}$, but it is easier to understand.

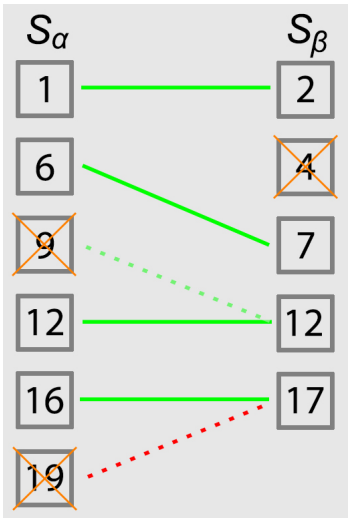

Figure 3.25: An example of CDS matching calculation.

The green lines represent the marriages that prevail, and the green dashed line represents a divorce. The red dashed line presents a marriage attempt that has failed.

The algorithm always runs through the array S_α, i.e., S_β is searched for distances in order to match with individual S_α distances. In the example of the figure occur the following phases:

1. Marriage between 1 and 2;
2. Marriage between 6 and 7 (4 never gets married);
3. Marriage between 9 and 12;
4. Divorce of 9, because of marriage between 12 and 12;
5. Marriage between 16 and 17;

6. Failure of 19 to get married with 17, because 17 has already a marriage with lower $\Delta s_{i,j}$.

The rejections of non repeated distances are mostly done by marriages and divorces. Nevertheless, there is still the possibility of two non repeated distances (i.e., distances derived of non repeated corners) getting married. This prejudices the CDS match by decreasing its quality (Q_{CDS} rises). In table 3.3 is illustrated 2 CDS and the differences between each distance. For the sake of simplicity, there are no divorces and no single distances (every distance is married).

Table 3.3: A CDS match with unfavourable marriages.

s_α	1	9	15	20	35	50	55	57
s_β	2	4	17	26	40	51	54	56
Δ_s	1	5	2	6	5	1	1	1

As the table shows, the differences of the unfavourable marriages (orange) are higher than the others. This means an easy rejection of those differences. When the marriages process ends, the average of all differences is calculated, but Q_{CDS} is not yet attributed. Instead, a method equivalent to a low-pass filter (of signal analysis study field) cuts off the high differences. Every difference higher than the average multiplied by a threshold $t_{\Delta s}$ is discarded. A new average is calculated and Q_{CDS} assumes its value. Figure 3.26 shows an example of this method. Every difference above the red dashed line is discarded.

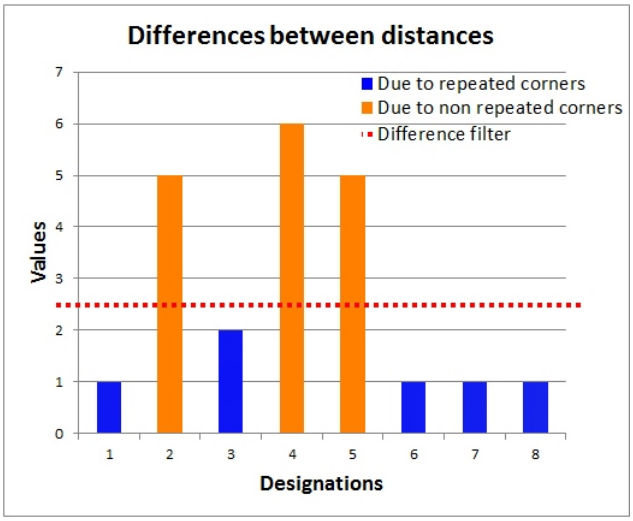

Figure 3.26: Filtering of unfavourable differences.

In step 4 the set of approved corner matches is analysed and M_a best matches are accepted according to their Q_{CDS}. Let matrix B be a structured set of quality values of all matches, and $Q_{CDS,i,j} \in B$. Hence, $Q_{CDS,i,j}$ is the quality of the match between the i^{th} corner of frame α and the j^{th} corner of frame β, where $i \in \{1, ..., N_\alpha\}$ and $j \in \{1, .., N_\beta\}$. If such match was previously discarded for some reason, $Q_{CDS,i,j}$ assumes a *null* value. $\vec{B}$ is the set of quality match values of matrix B, structured as $\vec{B} = \{Q_{CDS,1,1}, ..., Q_{CDS,N_\alpha,1}, ..., Q_{CDS,1,2}, ..., Q_{CDS,N_\alpha,N_\beta}\}$. Figure 3.27, illustrates an example of B and $\vec{B}$ with size 9.

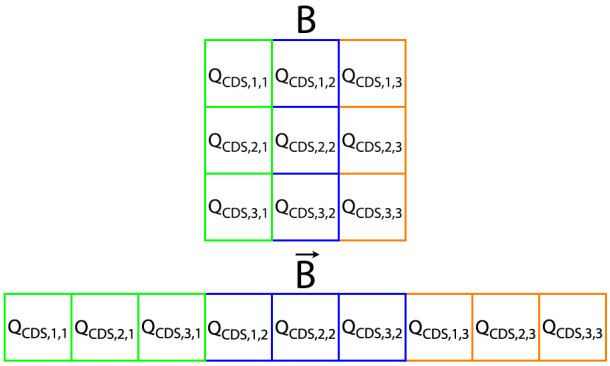

Figure 3.27: Example of a conversion of B to $\vec{B}$.

For the algorithm purpose, $\vec{B}$ is then sorted in an increasing order. This means, the first value is the best match quality (lowest Q_{CDS}) and the last value the worst match quality (highest Q_{CDS}). The algorithm collects the values of $\vec{B}$ from the beginning until t_{mb} values are found, or a *null* value is found. The threshold t_{mb} is chosen by the user. This collection is performed with no repeated indexes i or j of $Q_{CDS,i,j}$, to guarantee no corners repetition in different corner matches. Therefore, the number of accepted corner matches is always $M_a \leq t_{mb}$.

FMBF has 4 running **options**:

1. Run stage 1, 2, 3 and 4;
2. Run stage 2, 3 and 4;
3. Run stage 3 and 4;
4. Run stage 3 not limited to t_{ma} and run 4.

In option 1, 2 and 3, FMBF runs the stages 1, 2 and 3, until t_{ma} approved random corner matches are found. In option 4, all corner match combinations are tested, in order to make sure a valuable frame match is calculated.

This options system performs a trade-off between computational speed and efficiency, as illustrated in figure 3.28.

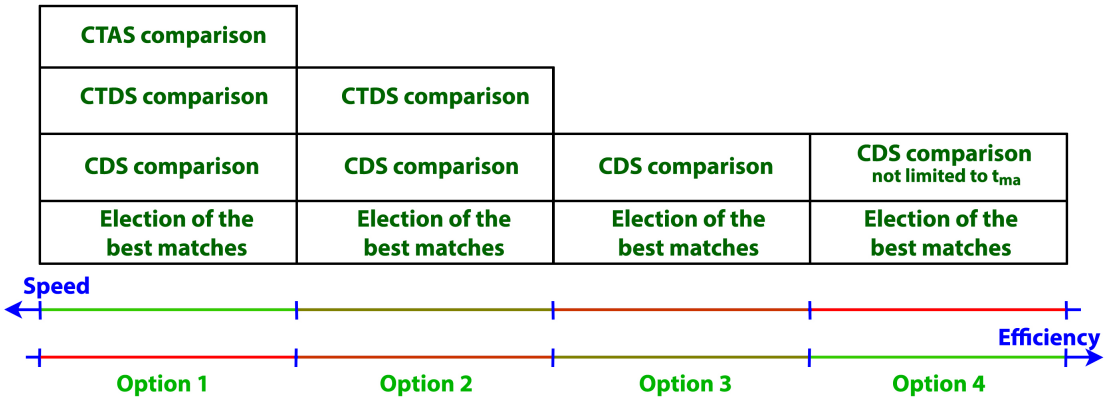

Figure 3.28: Trade-off between speed and efficiency of FMBF options.

If option 4 is always used, the algorithm will calculate many and accurate matches, because it runs throughout all the existing match combinations (MC number of CDS comparisons). However, it is very slow. Instead, if option 1 is always used, it works very fast in certain frame matches with hight corner repeatability. Still, it exceedingly rejects correct matches in certain frame matches, preventing its success.

RANSAC is used after FMBF in order to make sure the approved corner matches have a motion model in common. The accepted corner matches have small portions of outliers. Therefore, the number of iterations It is low and RANSAC presents a small computational cost. The fact that RANSAC may reject some portion of the accepted corner matches means that the number of the RANSAC elected corner matches, M_e, can be lower or equal to M_a. To reckon the calculated frame match as correct, M_e needs to be higher than a certain threshold t_e.

In order to fulfil a correct motion estimation, with a balance between speed and efficiency, the following ascendant steps are used:

Step 1: Run option 1 in FMBF. **If** M_a is **NOT** higher **than** t_e, go to step 3.

Step 2: Run RANSAC. **If** M_e is higher than t_e, go to step 9.

Step 3: Run option 2 in FMBF. **If** M_a is **NOT** higher **than** t_e, go to step 5.

Step 4: Run RANSAC. **If** M_e is higher than t_e, go to step 9.

Step 5: Run option 3 in FMBF. **If** M_a is **NOT** higher **than** t_e, go to step 7.

Step 6: Run RANSAC. **If** M_e is higher **than** t_e, go to step 9.

Step 7: Run option 4 in FMBF.

Step 8: Run RANSAC.

Step 9: Move on to the motion estimation.

The set of matches provided by RANSAC is denominated as E.

3.1.6 Motion estimation

The motion estimation process estimates a transformation model of the robot motion, base on all the provided feature matches. This motion is a relative motion, considering it is performed between pairs of frames. If one frame motion estimation fails, the next estimated motions will be incorrect relatively to the origin of the path. Motion estimations may fail if the corner repeatability between frames is not enough. Some authors use windowed bundle adjustment to prevent this from happening [OMSM00]. Performing frame matches between the current frame and several previous frames increases the quality of the motion estimation. In the future that approach could be used in this work, to increase its robustness. For the moment this work is focused on obtaining the best results with fewer and essential methods.

The set of robot poses denoted by $C_{0:n} = \{C_0, ..., C_n\}$ contains all the transformations with respect to the initial coordinate frame at $k = 0$. Let $I_{0:n} = \{I_0, ..., I_n\}$ be the set of n frames taken by the camera. Hence, at the moment k, the robot pose C_k corresponds to the same moment as when frame I_k is taken. C_k assumes the following form:

$$C_k = \begin{bmatrix} cos(\theta) & -sin(\theta) & X \\ sin(\theta) & cos(\theta) & Y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.11}$$

where θ is the robot orientation, and X and Y are the x and y coordinates of the robot pose respectively.

Two robot positions at adjacent time instants $k-1$ and k are related by the rigid body transformation $H_{k,k-1} \in \Re^{3\times3}$ of the following form:

$$H_{k,k-1} = T_{k,k-1}R_{k,k-1} \tag{3.12}$$

where $T_{k,k-1}$ is the translation matrix of the format:

$$T_{k,k-1} = \begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

and $R_{k,k-1}$ is the rotation matrix of the format:

$$R_{k,k-1} = \begin{bmatrix} cos(\varphi) & -sin(\varphi) & 0 \\ sin(\varphi) & cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

where φ is the rotation angle.

Therefore, the set $H_{1:n} = \{H_{1,0}, ..., H_{n,n-1}\}$ contains all subsequent motions. To simplify the notation, from now on, H_k is used instead of $H_{k,k-1}$. The current pose C_n can be computed by concatenating all the transformations H_k $(k = 1, ..., n)$. Hence, $C_n = C_{n-1}H_n$, where C_0 is the robot pose at the origin (instant $k = 0$). In other words, H_k is the position of the reference C_k according to the reference C_{k-1}. C_0 can be set arbitrarily and corresponds to the identity matrix of size 3.

The estimation of $\widehat{H}_k$ is performed through 3 references: the frames I_k and I_{k-1} central references and the ceiling reference created by the corners of the elected matches. By using the centre of each I_k to perform motion calculations, it is assumed to be coincident with the centre of the robot itself. With this 2D position system, 2 points (corners) is enough to create a ceiling reference. One for the reference position, and the other for its orientation. This is why in RANSAC $\upsilon = 2$. Let's consider H_β the transformation between the central reference of I_{k-1} and the ceiling reference, and H_α the transformation between the central reference of I_k and the ceiling reference. Therefore, $\widehat{H}_k = H_\alpha {H_\beta}^{-1}$. Figure 3.29 illustrates $\widehat{H}_k$ by the orange arrow and, H_α and H_β by the yellow arrows.

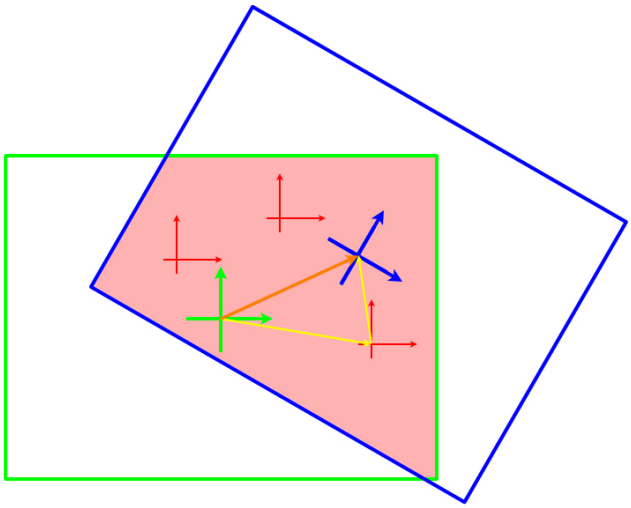

Figure 3.29: Reference relation between a pair of frames.

The ceiling reference can be stipulated anywhere inside the red area, as long as it is the same for H_α and H_β calculations. The previous figure shows 3 ceiling reference examples as red. The higher is the number of points, the more accurate is the estimation of $\widehat{H}_k$.

As shown in figure 3.30, the set of points (of the elected matches) is divided in two sets. The average of each set performs a virtual point. It is with those 2 virtual points that the ceiling reference is computed (a point and a vector).

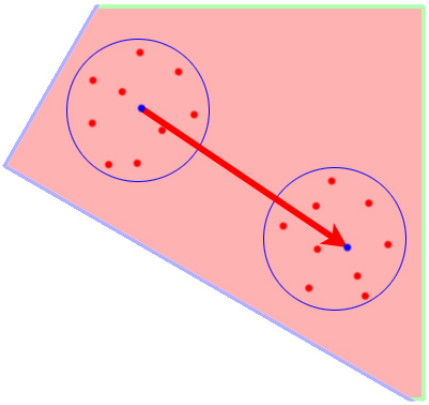

Figure 3.30: Ceiling reference creation.

The model that RANSAC estimates in each iteration is $\widehat{H}_k$, for the providing corner matches. It is known that a corner, in frame α coordinates p_α, transformed by H_k, is the same corner, in frame β coordinates p_β. That is: $p_\beta = p_\alpha H_k$. Therefore, an estimation $\widehat{p}_\beta = p_\alpha \widehat{H}_k$ is performed, and p_β (from the match that links p_α and p_β) is compared to $\widehat{p}_\beta$.

This comparison is made by measuring the distance between both points, which needs to be under a certain threshold t_{md}, in order for the match $p_\alpha \longleftrightarrow p_\beta$ to fit the model $\widehat{H}_k$.

3.1.7 Process chain

The FAST detector method becomes slower with higher number of detecting corners, i.e., lower threshold t. High number of corners increases the number of signature comparisons, leading to high computational costs. It also decreases the corner repeatability between frames, reducing the corner signature comparisons efficiency. Therefore, it is important to control the number of detected corners.

A corner controller was initial developed in this work. Ideally, it would control the number of corners as the block diagram represented in figure 3.31, also proposed in this work.

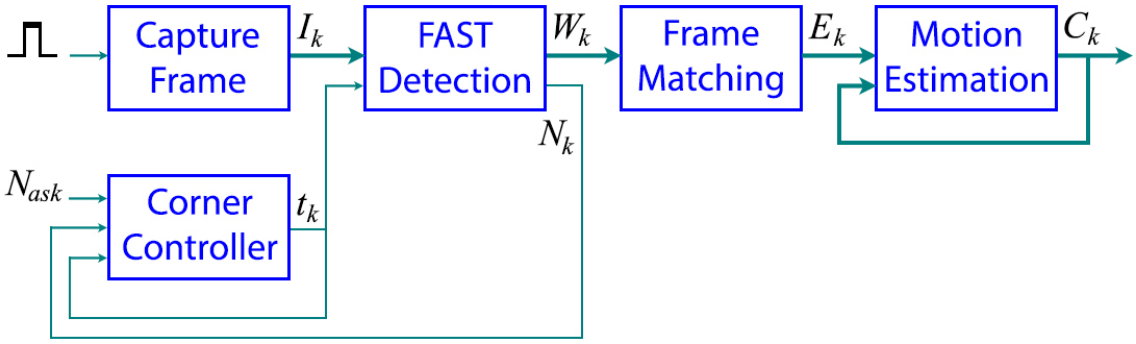

Figure 3.31: Ideal block diagram.

As shown in the figure, the controller aims to provide the extraction of N_{ask} number of *asked* corners in FAST detection. It is not possible to create a model representing the relation between t_k and N_k, for a given I_k. However, the controller provides t_k, estimated to influence the number of corners N_k, based on how t_{k-1} has influenced N_{k-1}, in I_{k-1}.

The FAST detector provides a set of 16 colour intensity differences, for each detected corner, denominated as $\Delta I_{p\to L}$. Those differences are the ones performed between the pixels of the circle segment test and the pixel in the middle, $\Delta I_{p\to L} = \{|I_p - I_{p\to l}| : \forall l \in L\}$. Hence, for each corner in a given frame, it is possible to know the maximum threshold t_{max} necessary to make the corner detectable. In other words, each corner as a t_{max} and is detected if t belongs to $\{0, ..., t_{max}\}$.

3.1.7.1 Choice of detecting a corner

The corner detector used in this work is FAST-9, therefore the size of the arc $p \rightarrow A$ is at least 9. This means, a detected corner has an arc $p \rightarrow A$ of at least 9 pixels. Nevertheless, for the calculation of t_{max} the size of $p \rightarrow A$ is always the minimum, 9. Therefore, let's consider a corner with an arc $p \rightarrow A$ of exact size 9. Hence, there is a subset of 9 contiguous differences in the set $\Delta I_{p \rightarrow L}$, all above the t used to detect the corner. Such subset is denominated as $\Delta I_{p \rightarrow A}$. This means the minimum of $\Delta I_{p \rightarrow A}$ is the t_{max} of the corner. Despite FAST detecting the corner with a certain $p \rightarrow A$, it does not mean there are not other arcs of size 9 in the circle $p \rightarrow L$, allowing the detection of the same corner. In fact, there are 16 possibilities for $\Delta I_{p \rightarrow A}$ to be positioned in the set $\Delta I_{p \rightarrow L}$. This means 16 minimum (from $\Delta I_{p \rightarrow A}$ of each possible position) need to be taken into account. An arc $p \rightarrow A$ can be positioned according $A = \{f_i, ..., f_i + 8\}$ and $i \in \{1, ..., 16\}$, where:

$$f_i = \begin{cases} i, \ i \leq 16 \\ 16 - i, \ i > 16 \end{cases}$$

In figure 3.32 the structure of the segment test is illustrated with its pixel enumerated by i, in the left. Only 5 arc positions are indicated with lines (red and green). In the right side are the representations of the arc position possibilities.

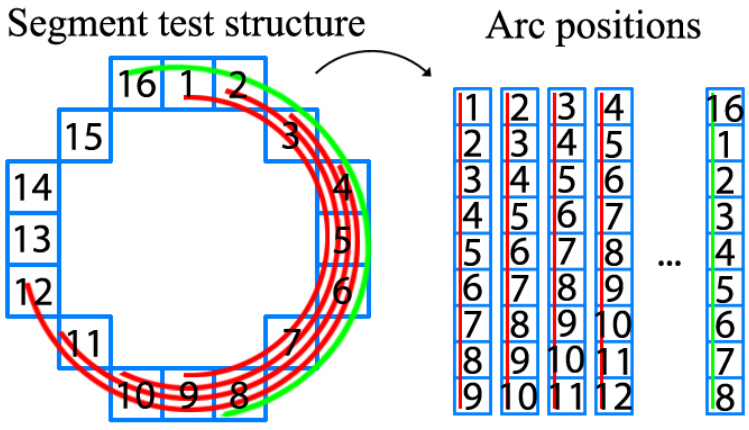

Figure 3.32: Determine t_{max} through $\Delta I_{p \rightarrow L}$.

The value of t_{max} is thus the maximum of all 16 minimums.

3.1.7.2 Controlling the number of corners

Applying the previous process in every detected corner of frame I_k, a set of t_{max} values is obtained, denominated as Υ_k. That set is sorted in a descendent order. Its size is obviously N_k, considering each value is associated with each corner of I_k. If N_{ask} is smaller than N_k, then the output t_{k+1} of the corner controller is the ${N_{ask}}^{th}$ value of Υ_k. Figure 3.33 illustrates an example, where $N_k = 15$ and $N_{ask} = 12$.

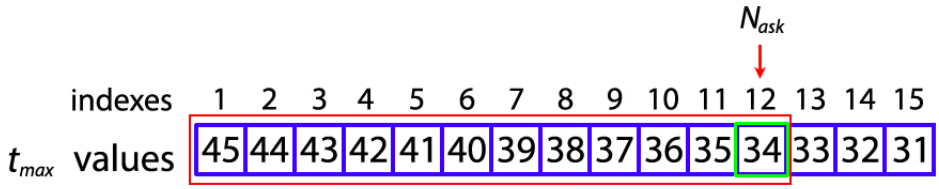

Figure 3.33: Example of choosing t_{k+1} in Υ_k, using N_{ask}.

In this example Υ_k is an integer number sequence from 31 to 45, for simplification. $N_{ask} = 12$, therefore the chosen threshold is 34. Hence, if the FAST detector runs in the same frame with a threshold of 34, the corners respective to the t_{max} indexed from 1 to 12 would be detected. This means, t_{k+1} is projected for I_k. Obviously, FAST is applied in the next frame I_{k+1} and not in I_k, which means the process is not going to work fully as expected. Nevertheless, it is assumed that the intersected area between I_k and I_{k+1} is large enough to allow this process to work with enough precision.

If N_{ask} is larger than N_k, than a mathematical function is used. Through many experiences, it is know that the number of corners varies exponentially with the decreasing of the threshold. Therefore, the used function is:

$$t_{k+1} = t_k - \mathrm{a}^{b(N_{ask} - N_k)} \tag{3.15}$$

After running some tests, the default values are: $a = 1,1$ and $b = 0,5$. This function is important to prevent the abrupt decreasing of the number of detected corners, in the next frame. Therefore, a and b need to be chosen so that a high number of corners is detected.

3.1.7.3 Windowed Corner Adjustment

The previous described controller tries to control the number of corners per frame in order to reduce the computational cost, as mentioned previously. However, once in a while there are occasions where N_k is significantly different from N_{ask}, slowing down the algorithm considerably. It may result from function 3.15 not fitting accurately to the real number of corners variation, or due to successive frames having low repeatability. Therefore, *Windowed Corner Adjustment* (WCA) is developed in this work to solve that problem.

In WCA a low threshold is chosen so that a high number of corners is detected, i.e.: N_{ask} is high. Those corners are then filtered, before the frame matching process. Filtering consists on selecting a number of corners N_f according to their t_{max} values (however, the number of selected corners can be slightly different than N_f). Such filtering is performed considering pairs of sets $\Upsilon_k \longleftrightarrow \Upsilon_{k+1}$, in order to reject the corners with t_{max} much different in each pair. With WCA, N_f number of corners (approximately) are used directly in the frame matching process. Without WCA, that number is N_{ask}. Using WCA, the experiments show that N_{ask} should be twice or three times N_f, to guarantee that N_f number of corners are selected.

In the pair $\Upsilon_k \longleftrightarrow \Upsilon_{k+1}$ the lower limit of the filter is the minimum between the values of the ${N_f}^{th}$ position, in Υ_k and Υ_{k+1}. The higher limit of the filter is the minimum between the values respective to the 1^{st} position, in Υ_k and Υ_{k+1}. Figure 3.34 illustrates an example of this process, with 3 pairs of sets Υ_k and $N_f = 12$.

The values of Υ_k are inside the blue grids. The orange values are the ones that are discarded and the green ones are accepted, in respective pairs. At the left of each set is the indexation of their values. The red index is N_f.

In this example, Υ_2, in the pair on the middle, and Υ_3, in the pair on the right, are filtered with 11 number of corners, even with $N_f = 12$. That occurs because Υ_2 and Υ_3 have their last values higher then their pairs, and their 12^{th} lower then their pairs.

Without WCA, the corner controller is used to reduce the computational cost of the FAST detector and, consequently, the feature matching process. With WCA, the corner controller is used just to reduce the computational cost of the FAST detector, considering

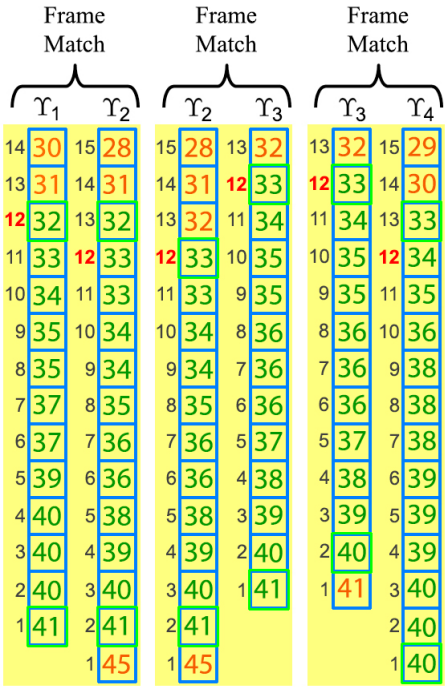

Figure 3.34: Example of WCA filtering 3 pairs of sets Υ_k with $N_f = 12$.

that the corners are filtered by WCA, before the matching process.

The WCA is an important breakthrough of this work, because it allows detecting a large portion of corners, and provides a selected set of strong candidate matches to the frame matching process. This reduces the frame matching process time consuming and increases the frame matching quality.

The block diagram with WCA is therefore illustrated in figure 3.35.

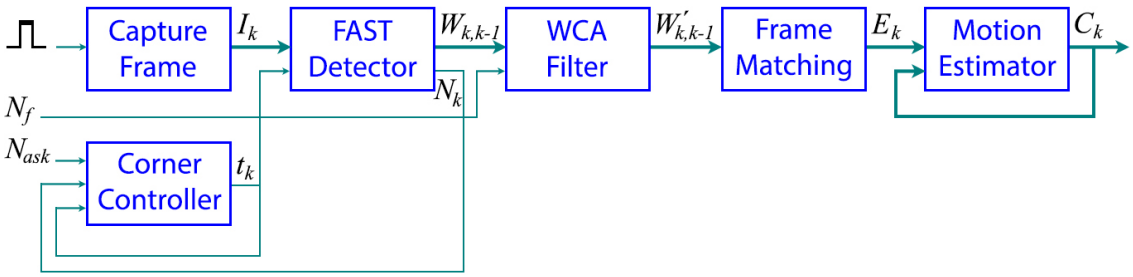

Figure 3.35: Block diagram with WCA.

Using WCA filter is always convenient. Therefore, the above block diagram is used in all tests of the experimental results of this work.

3.2 Mapping

This VO system is designed to work only in 2D. Therefore, while the robot is moving in a 2D plane, the respective taken frames of the ceiling are recorded in order to create a 2D map. The first frame is placed in the origin and the others are placed using their pose C_k, related to the origin. Figure 3.36 shows a sequence of taken frames.

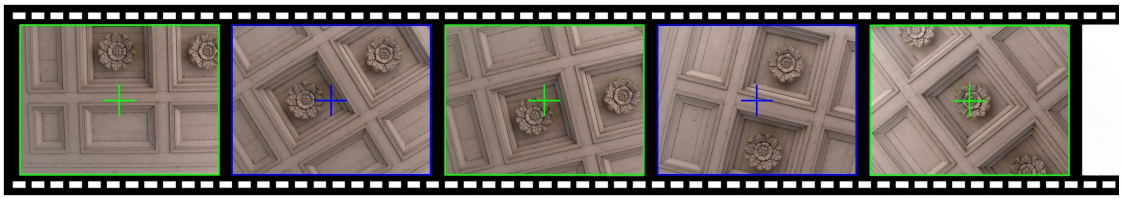

Figure 3.36: Frame sequence taken from a ceiling.

Figure 3.37 illustrates the map performed with the previous frame sequence.

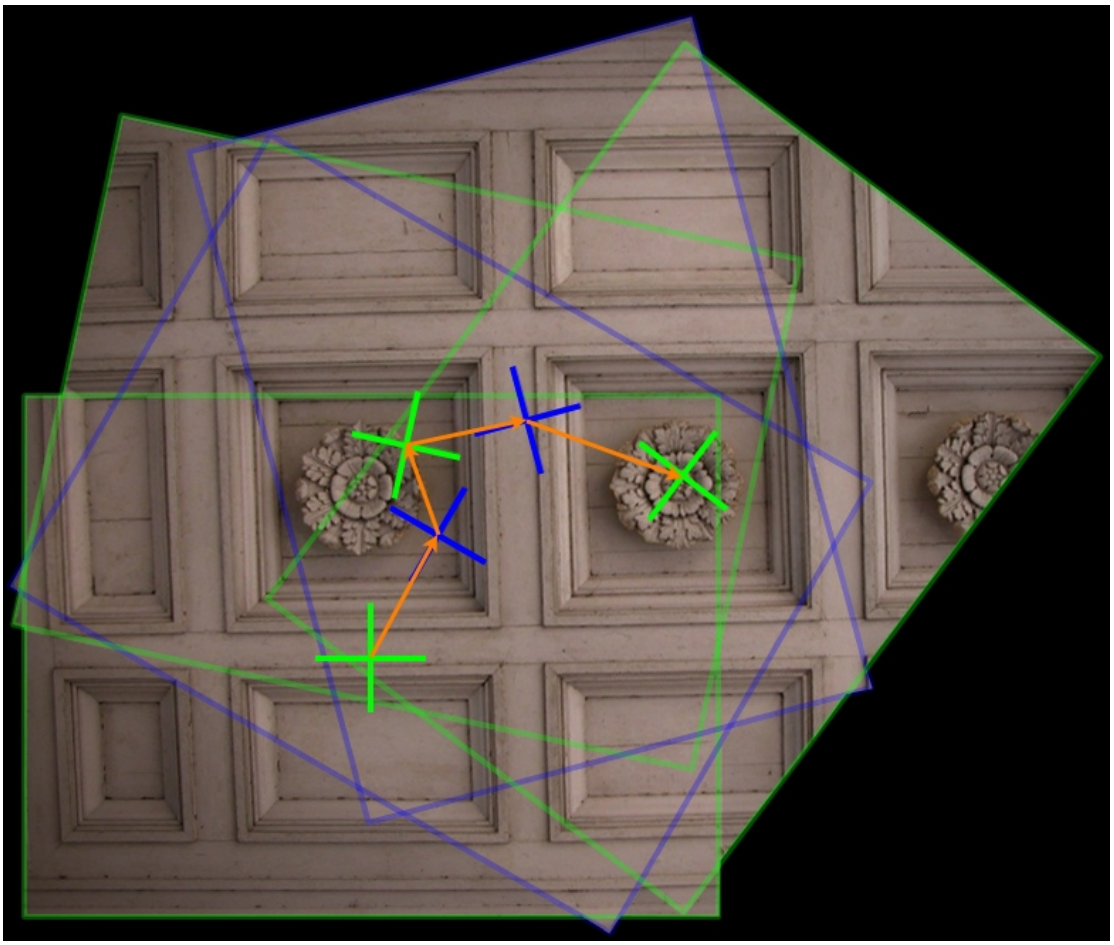

Figure 3.37: Map performed by a frame sequence.

3.3 Camera calibration

In order to obtain the robot motion in SI units, a conversion is performed from pixels units. To perform such conversion, it is important to understand the relation between the distance from the lens of the camera to the ceiling b, and the camera *Field Of View* (FOV) angle. Figure 3.38 shows the geometry of the field of view of a generic perspective camera.

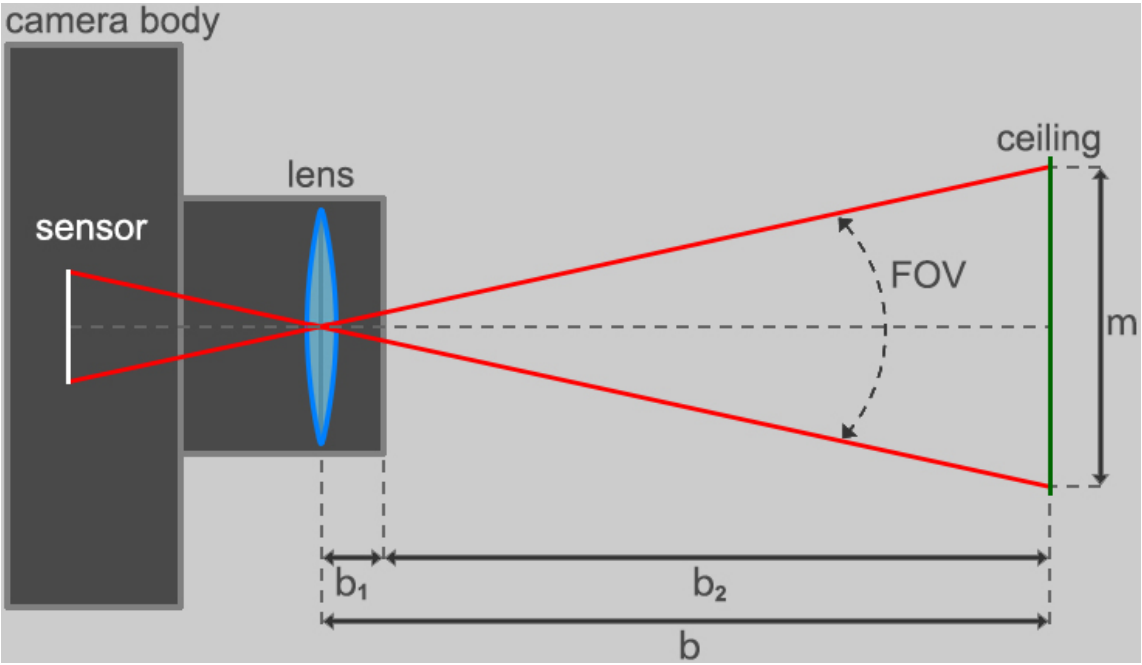

Figure 3.38: Field of view of the camera.

Through the geometry of the figure, doing the maths, FOV is calculated by:

$$FOV = 2\,atan\left(\frac{m}{2b}\right) \tag{3.16}$$

The distance b is split in b_1 and b_2 ($b = b_1 + b_2$). Quantity b_1 is the distance between the lens and the edge of the camera entrance, and b_2 is the distance between that edge and the ceiling. And m is a dimension of the picture (either width or height). Hence, the millimetres/pixels relation is performed by knowing m in pixels and in millimetres. The values of m and b_2 are possible to be measured directly, and vary from scene to scene. But, FOV and b_1 are camera features. Hence, the purpose of the camera calibration is to calculate those features, for a certain camera. Knowing those features and the distance

b_2, m is calculated by:

$$\begin{aligned} m &= 2\,tan\left(\frac{FOV}{2}\right) \times b \\ &= 2\,tan\left(\frac{FOV}{2}\right) \times (b_1 + b_2) \end{aligned} \tag{3.17}$$

Every camera taking rectangular images have different FOV for width and height dimensions. It is possible to calculate one FOV by knowing the other, through the *Aspect Ratio* (AR) of the image. The camera used in this work is a webcam with 2.0 megapixels, and the resolution used for this calibration is 640×480 pixels, leading to an AR of 4/3. For better metrical accuracy both FOV are calculated experimentally.

In order to calculate FOV and b_1, several pictures are taken to a parallel plane, with different distances b_2 from it. The plane has measurements written in SI units, allowing to find the image dimensions in millimetres. Figure 3.39 corresponds to a picture taken with $b_2 = 1200$ mm.

Figure 3.39: A calibration picture.

In the figure, the number in the middle corresponds to the distance b_2, and the numbers in sequences correspond to width and height scales, respectively. Those scales are in centimetres, but the conversion is performed to millimetres. Therefore, in this picture 500 mm corresponds to 310 px (between 50 and 0 in the picture) and 600 mm corresponds to 373 px (between 60 and 0 in the picture). In conclusion, considering the resolution of 640×480 pixels, this camera visualizes an area of $1032,3 \times 772,1$ mm from a distance of 1200 mm.

One more picture taken at a different distance then 1200 mm would enable the calculation the camera features. However, for a more accurate calibration, 10 pictures are taken. Table 3.4 shows the results of several taken pictures varying b_2.

Table 3.4: Behaviour of the field of view.

b_2 (mm)	1000	1100	1200	1300	1400	1500	1600	1700	1800	1900
M_w (px)	310	286	260	242	225	210	197	186	174	165
M_h (px)	373	343	312	291	270	252	237	223	209	198
m_w (mm)	1032	1119	1231	1322	1422	1524	1624	1720	1839	1939
m_h (mm)	772	842	923	990	1067	1143	1215	1292	1378	1455
AR	1,34	1,33	1,33	1,34	1,33	1,33	1,34	1,33	1,34	1,33
$\Omega_{640\times480}(mm/px)$	1,61	1,75	1,92	2,06	2,22	2,38	2,53	2,69	2,87	3,03
$\Omega'(mm/px)$	258	280	308	330	356	381	406	430	460	485

The measures M_w px and M_h px correspond to 500 mm and 600 mm respectively. Through linearisation, m_w and m_h are calculated, corresponding to 640 px and 480 px, respectively. The indexes w and h means *width* and *height*. Considering the measures M_w and M_h are performed by human eye, the AR is used to chose those values more accurately. That is, when collecting M_w and M_h values, they may be adjusted in a few pixels so that AR is as close as possible to 4/3 $(1,3(3))$. The relation between millimetre and pixel is calculated by dividing m_w per 640 (or m_h per 480, considering the differences are insignificant) and is denoted by $\Omega_{640\times480}$. However, such relation works only for the resolution used in the calibration. A normalized relation millimetre/pixel, denoted by Ω', is calculated by the average of the division between m_w per 4 and m_h per 3, $\Omega' = \frac{1}{2}\left(\frac{m_w}{4} + \frac{m_h}{3}\right)$. Therefore, $\Omega_{640\times480} = \frac{1}{160} \times \Omega'$, i.e., in order to obtain millimetre/pixel for the resolution 640×480, Ω' is divided by factor of 160. Let's consider $g_{w\times h}$ a generic factor for a certain resolution $(w \times h)$. Hence, g is obtained by $g_{w\times h} = \frac{w}{4}$ (or $g_{w\times h} = \frac{h}{3}$).

And generically: $\Omega_{w\times h} = \frac{1}{g_{w\times h}} \times \Omega'$.

With the values of the previous table b_2, m_w and m_h, the graph presented in figure 3.40 is created.

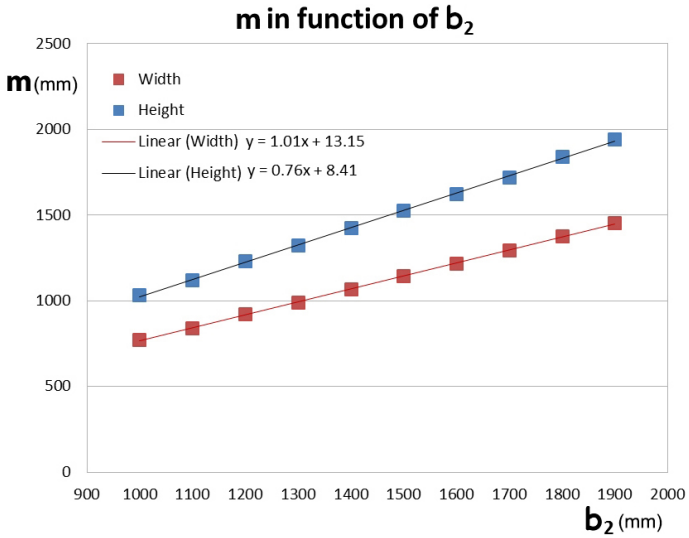

Figure 3.40: Calibration graph.

As expected, a linear behaviour is found between m and b_2, both in height and width. Both functions are linear, of $y = mx + b$ type, where y is m and x is b_2.

With the values of the previous table b_2 and Ω', the graph presented in figure 3.41 is created.

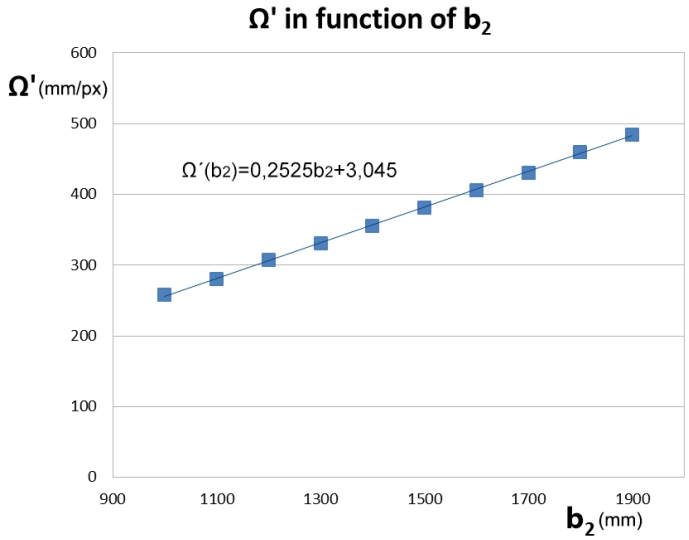

Figure 3.41: Millimetre per pixel in function of the distance b_2.

Therefore, the function of the previous graph,

$$\Omega'(b_2) = 0,2525b_2 + 3,045, \tag{3.18}$$

is the one used to convert from pixels to millimetres. Considering the motion of the robot is (X, Y) in pixels, $(\frac{1}{g_{w\times h}} \times \Omega'(b_2) \times X,\ \frac{1}{g_{w\times h}} \times \Omega'(b_2) \times Y)$ is the motion of the robot in millimetres, for a certain b_2.

The straight lines from the graph of figure 3.40 indirectly represents the previously mentioned features of the camera. However, those lines are only mathematical abstractions. It is FOV that is commonly used feature a camera in this field of study. Therefore, let's consider FOV_w the FOV measured in a scale of width and FOV_h similarly for the scale of height. Using equation 3.16,

$$FOV_w = 2\,atan\left(\frac{1}{2}\frac{m_w}{b_{1w} + b_2}\right) \tag{3.19}$$

and

$$FOV_h = 2\,atan\left(\frac{1}{2}\frac{m_h}{b_{1h} + b_2}\right) \tag{3.20}$$

The calculation of b_{1w} is performed by finding x when the *width* line $y = 1,01x+13,15$ intercepts the x axis, and b_{1h} is similarly calculated with the *height* line $y = 0,76x+8,41$. Hence, $b_{1w} = 13,02$ mm and $b_{1h} = 11,07$ mm. The difference between b_{1w} and b_{1h} occurs due to measure uncertainties, and are insignificant. In order to calculate both FOV, two random points (m_w, b_2) and (m_h, b_2), from width and height lines respectively, are chosen. This leads to: $FOV_w = 53,59°$ and $FOV_h = 41,61°$.

3.4 Simulation

Considering VO is composed by a sequence of processes, it is convenient to validate the under developing process sequence, during the development period. As is well known, the processes can be validated by checking every included instruction, through the whole complex system. However, that is not an efficient way of validation. First of all, it is time consuming for the developer. Furthermore, it is not accurate and viable. Therefore, simulators are of major importance to perform the validation of the system. Simulators

can simulate a certain imposed *reality* and provide automatic analyses regarding certain processes. Analysis of the processes behaviours, error dispersion and error identification can be used in order to understand the algorithm reactions to the *reality* imposed by the simulators. It should also be noted that working with simulators facilitates the maturation of the underlying ideas while research moves on.

3.4.1 Corner Signatures Simulator

The signatures simulator is used to validate the feature matching algorithm, concerning corner signatures. Two virtual frames are generated with the coordinates of several points, simulating corner locations. The second frame (β) has a virtual rotation and a translation relatively to the first frame (α). The conditions imposed for that simulation are based on the following values: number of points of the first frame N_α, number of points of the second frame N_β, number of requested repeated points RP and frame projective transformation $H(X, Y, \varphi)$. Hence, those values may vary for different tests in order to facilitate, or make it harder for the matching algorithm.

Frame α is generated including N_α random points with different coordinates. Let P_i^α be the coordinates of the points of frame α, with the format $[x\ y]^T$, with $i \in \{1, ..., N_\alpha\}$, $x \in \{1, ..., w\}$, $y \in \{1, ..., h\}$, w is frame width and h is frame height. P_i^β is similar, for frame β. The projective transformation matrix $H(X, Y, \varphi)$ is calculated as follows:

$$H(X, Y, \varphi) = T(X, Y) \times R(\varphi) \tag{3.21}$$

Where:

$$T(X, Y) = \begin{bmatrix} 1 & 0 & X \\ 0 & 1 & Y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.22}$$

and

$$R(\varphi) = \begin{bmatrix} cos(\varphi) & -sin(\varphi) & 0 \\ sin(\varphi) & cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.23}$$

Frame β is generated including N_β points. From those points, the number of repeated points is denoted as N_r, and their coordinates are calculated according to the following equation:

$$\begin{bmatrix} {P_i}^\beta \\ 1 \end{bmatrix} = H(X, Y, \varphi)^{-1} \begin{bmatrix} {P_i}^\alpha \\ 1 \end{bmatrix} \tag{3.24}$$

The rest of the points are randomly generated.

The following figures present an example of a simulation. In this example the imposed motion is: $X = 15$ mm, $Y = 40$ mm and $\varphi = 30°$, and the imposed conditions are: $N_\alpha = 12$, $N_\beta = 14$ and $RP = 10$. Figure 3.42 illustrates the geometry relation between frame α and β.

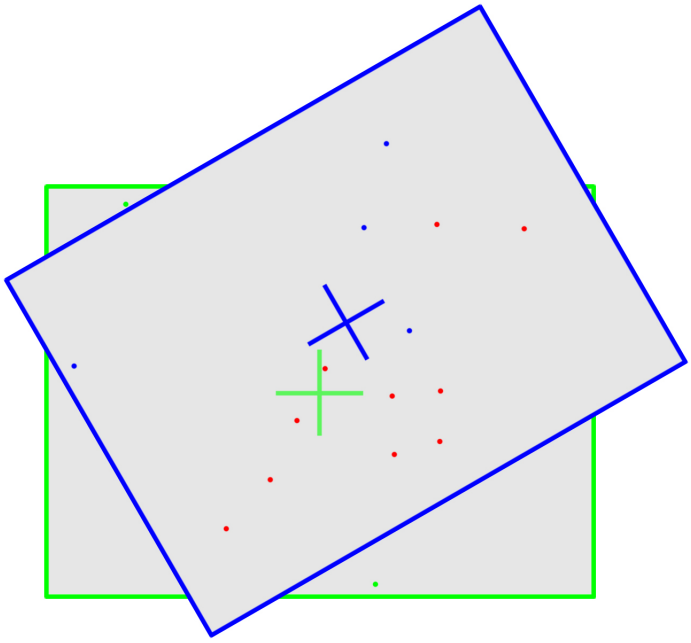

Figure 3.42: Geometry of both frames in a random example of a signature simulation.

The number of points inside the intercepted area, between both frames, is denoted as N_{ia} and $\bar{N}_{ia}^\alpha$ is for the number of points outside that area, in frame α. In the figure, the repeated points are shown as red, the other points belong only to the frame correspondent to their colour. By chance, two points of frame α are out of the intercepted area between both frames ($\bar{N}_{ia}^\alpha = 2$), which provides the possibility of generating all requested repeated points (considering $N_\alpha - RP = 2$). In general, if $RP > N_{ia}$ then $N_r = N_\alpha - \bar{N}_{ia}^\alpha$, otherwise

the number of repeated points is as requested ($N_r = RP$). This limitation is previously mentioned in section 3.1.2, and illustrated in figure 3.9. After generating all N_r points in frame β, more points are randomly generated until performing N_β points.

Figure 3.43 presents the output of the frame matching algorithm performed in the above virtual frames. As mentioned previously, only the coordinates of the points are used in this simulator, the pictures are used just for visualization purposes.

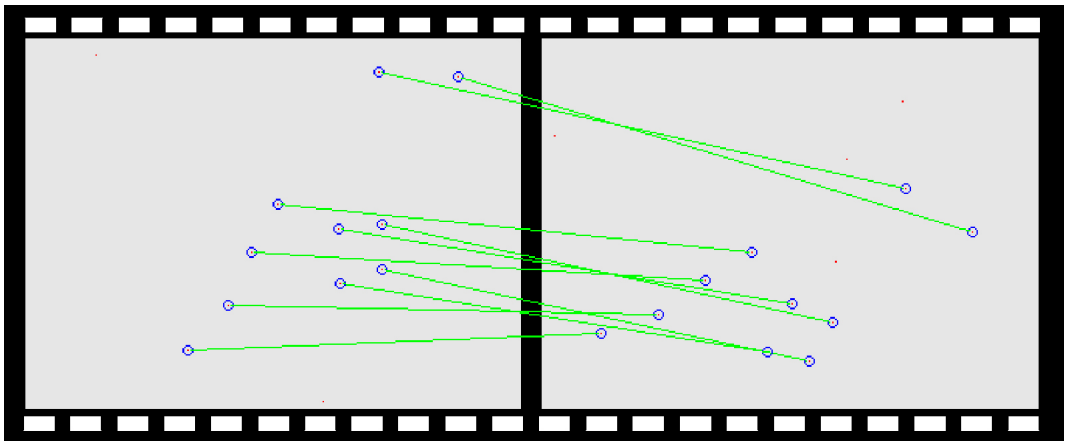

Figure 3.43: A signature simulation match.

The estimated matches are indicated as green lines, linking pairs of points highlighted with blue circles.

Table 3.5 presents the results of the previous simulation.

Table 3.5: Results of the corner signatures simulation.

N_α	14		
N_β	10		
RP	10		
N_r	10		
$CornerMatches$	$Correct = 10$	$Undetected = 0$	$Incorrect = 0$
DOF	$\mathbf{X}(px)$	$\mathbf{Y}(px)$	$\varphi(^\circ)$
$Simulated\ Motion$	$15,00$	$40,00$	$30,00$
$Estimated\ Motion$	$14,98$	$40,27$	$30,11$
$Absolute\ Errors$	$0,02$	$0,27$	$0,11$
$Relative\ Errors$	$0,13\%$	$0,68\%$	$0,37\%$

As illustrated in the table, the number of repeated points agree with the number of requested repeated points ($RP = N_r$). This occurs because the virtual overlapped areas between frames contains exactly the number of repeated points (RP).

This example has high corner repeatability, which leads to a correct motion estimation.

The errors are considered small and insignificant.

3.4.2 Picture Frames Simulator

The picture frames simulator is used to validate the feature matching algorithm, concerning all corners metrics. Its goal is to manipulate the path of the robot, taking virtual frames of a certain ceiling. The simulation consists on choosing a large picture to work as a background, where virtual robot positions are selected by mouse clicking. Such positions are drawn in the background linked with straight lines. Figure 3.44 illustrates an example of a path selection option.

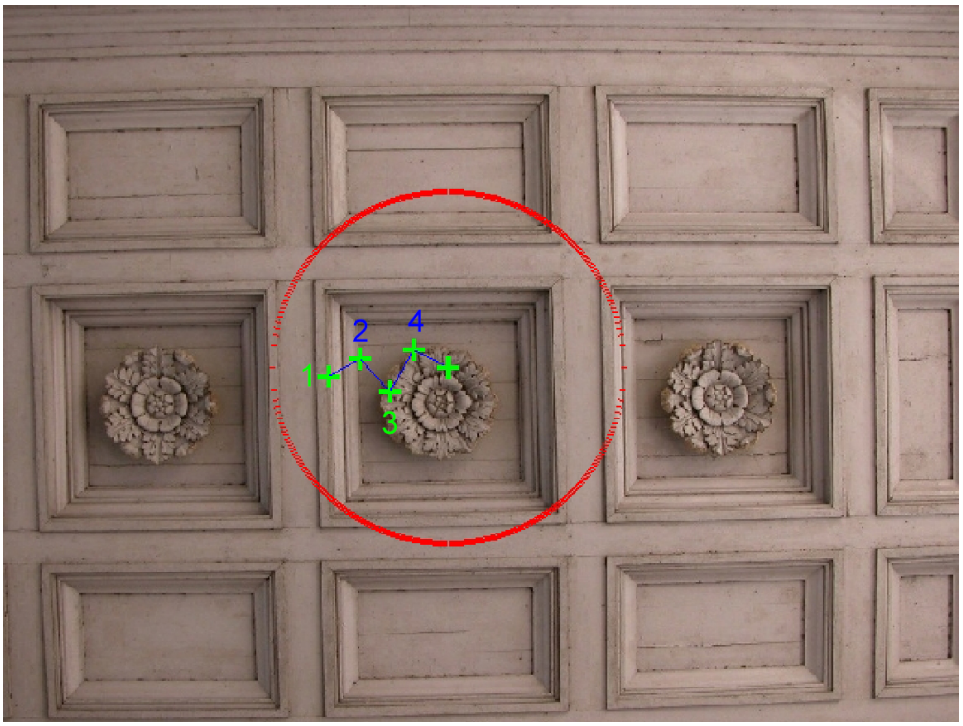

Figure 3.44: Path selection option.

It is known that the distance between frames is a limitation of the matching algorithm. Therefore, while the user is choosing the path, a red circle is drawn in order to identify the area where the next position could be placed. Outside the circle it is not allowed to place the next position. The radius of the circle is by default half of the frame height, to guarantee that the intercepted area between frames is large enough. The last selected position is used only to state the orientation of its previous position. In the example of the previous figure it is simulated the robot moving always forward. Figure 3.45 presents

the location and orientation of the taken virtual frames, in the previous example.

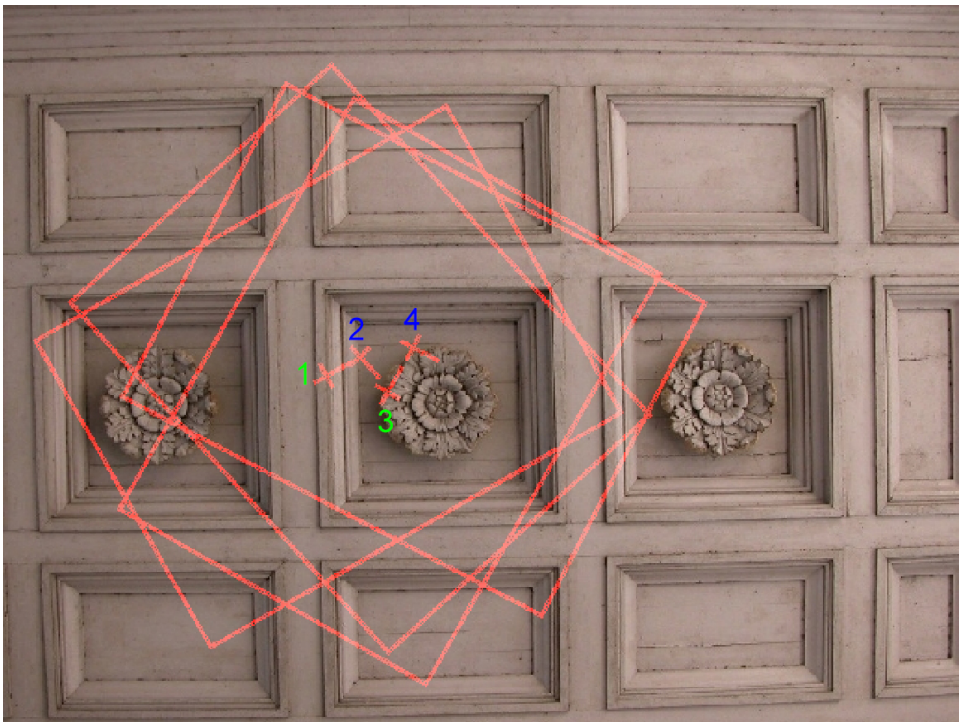

Figure 3.45: Locations and orientations of the virtual frames.

Figure 3.46 displays the four previous virtual frames.

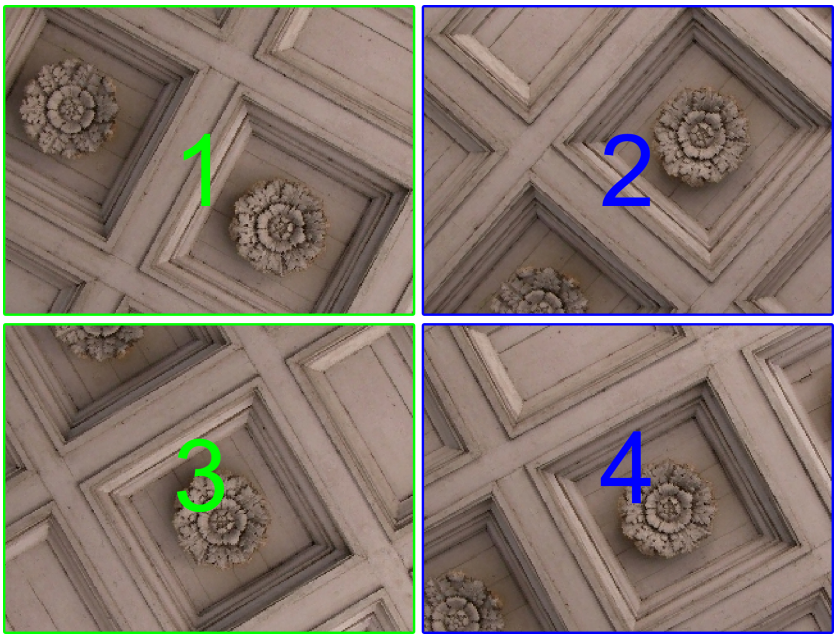

Figure 3.46: Virtual frame sequence.

Figure 3.47 shows the output of the frame matching algorithm for frame 1 and 2.

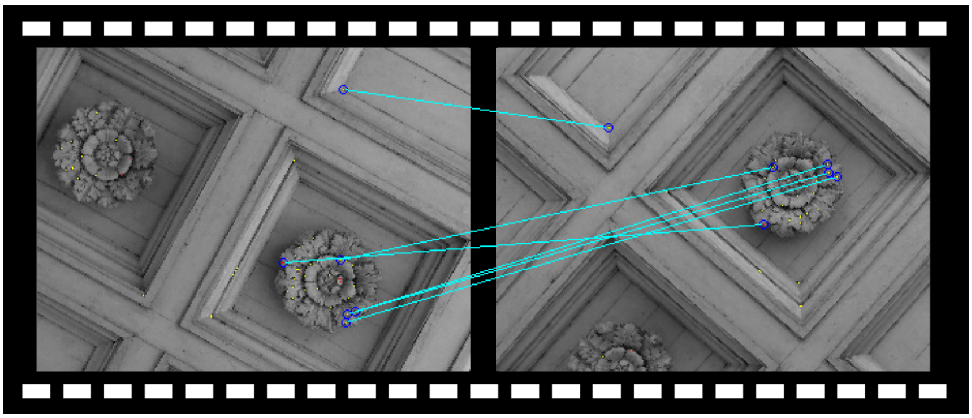

Figure 3.47: Matching between frame 1 and 2.

Figure 3.48 shows the output of the frame matching algorithm for frame 2 and 3.

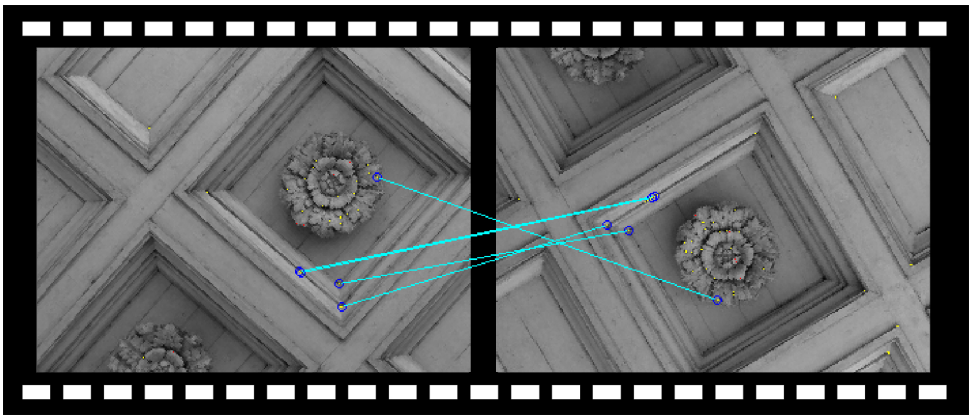

Figure 3.48: Matching between frame 2 and 3.

Figure 3.49 shows the output of the frame matching algorithm for frame 3 and 4.

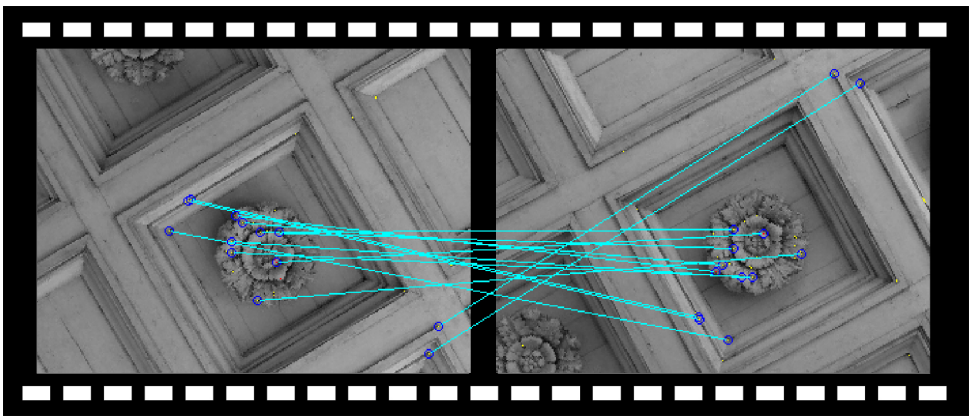

Figure 3.49: Matching between frame 3 and 4.

In the end of the simulation 3 pictures are presented with: groundtruth, the path

estimation and both. Figure 3.50 shows those 3 pictures correspond to the previous example.

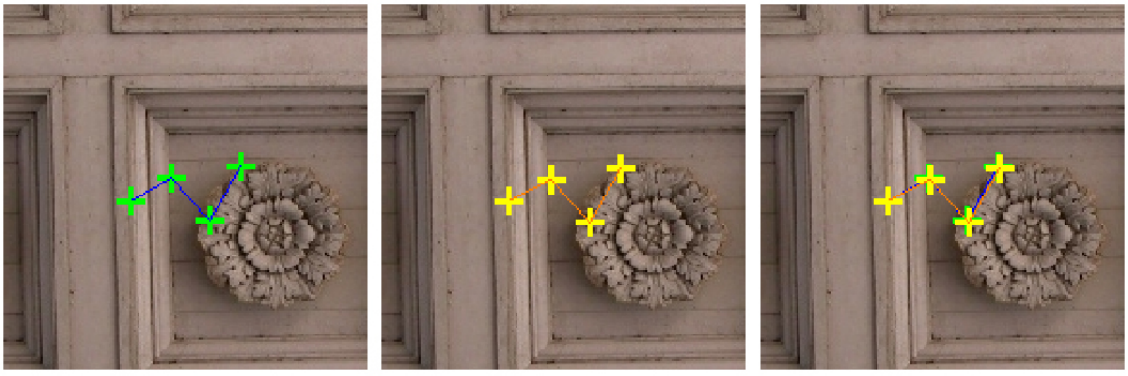

Figure 3.50: Groundtruth and path estimation.

Table 3.6 presents the results of the picture frames simulation example. The attempts is the number of option attempts performed by the FMBF algorithm, and M_e is the number of matches approved, used for motion estimation.

Table 3.6: Results of the picture frames simulation.

Frame Match	**1 ⟷ 2**			**2 ⟷ 3**			**3 ⟷ 4**		
Attempts	3			2			3		
M_e	6			5			13		
N_α	51			32			31		
N_β	32			50			29		
DOF	$\mathbf{X}(px)$	$\mathbf{Y}(px)$	$\varphi(^\circ)$	$\mathbf{X}(px)$	$\mathbf{Y}(px)$	$\varphi(^\circ)$	$\mathbf{X}(px)$	$\mathbf{Y}(px)$	$\varphi(^\circ)$
Sim. Motion	24, 19	0, 00	−77, 47	29, 73	0, 00	107, 98	32, 25	0, 00	−87, 81
Est. Motion	24, 73	−1, 17	−77, 14	29, 76	−0, 31	108, 19	32, 35	−0, 16	−87, 86
Abs. Errors	0, 54	1, 17	0, 33	0, 03	0, 31	0, 20	0, 10	0, 16	0, 06
Rel. Errors	2, 23%	–	0, 43%	0, 01%	–	0, 19%	0, 31%	–	0, 07%

It is verified that there is always a small error associated to the motion estimation, due to the discrete nature of the digital images (already mentioned in section 3.1.4.2). If there are matches, used for one single motion estimation, that happen to be wrong, a large irretrievable motion error damages the whole robot path estimation. In other words, errors are either significantly large or small. This is the reason why the frame matching method needs to be robust, in order not to fail even if it means being slow for certain difficult matching cases. In fact, simulations showed that every frame match having more

than 4 accepted matches ($M_e > 4$) is worth trust. On the other hand, $M_e \leq 4$ has a high probability of being wrong. After several tests using the simulator, performing low relative errors, the next step is to perform experiments in reality.

3.5 Summary

The implemented VO approach requires a camera pointed to a ceiling. Both the ceiling and the floor must be planes. The system takes frames from the ceiling and detects corners with FAST detector. In order to perform frame matches, corners are compared and matched. There are many outliers in the corner match which need to be removed. RANSAC is a widely used method for removing outliers. However, it is an iterative method that performs too many iterations in order to provide an efficient output. Therefore, the FMBF algorithm was created to reduce the number of outliers in the corner matches provided to RANSAC.

Before using FMBF, corners need a treatment. Corner Distance Signatures are created. CTS comparisons are very efficient, although they are very time consuming. Corner Triangle Signatures are also created. These ones are composed by Corner Triangle Distance Signatures and Corner Triangle Angle Signatures. A CTS comparison is not very efficient, but it is very fast. Therefore, CTS are used to reject the most obvious outliers before using CDS for accurate inlier selections. FMBF attempts to find the balance in the trade-off between speed and efficiency.

The provided corner matches inliers are then used for motion estimation. The corners are considered to be static references in the environment, and it is through them that the motion is calculated.

In order to control the FAST detector speed, a corner controller is used. Hence, the number of detected corners are approximately as asked.

WCA is a very important tool developed in this work. Before the frame match, the corner matches are pre filtered by WCA with insignificant time consuming. WCA performs a filtering in pairs of frames, based on all the maximum threshold each corner has.

To create a map, the estimated poses are used to concatenate the taken frames.

The camera is calibrated base on several frames taken to a certain plane. The frames

are taken perpendicularly to the plane and the plane has several written metric on it. Therefore, based on those metrics in millimetres and on the image projection in pixels, a function is created to convert pixels in millimetres.

Chapter 4

Experimental results

This chapter shows the experimental results of three particular tests. In the first test, nine frames of the same ceiling are used to build a map. In the second test, twenty four pictures of a different ceiling are used to estimate the robot path. In the third test, a video is used to create a map, based on where the robot went through.

4.1 Introduction

Digital image manipulation is used abundantly in this approach, therefore, matrix calculations, as well as complex algorithms using arrays and arrays indexes. MATLAB is a powerful tool that facilitates matrix manipulations and signal processing. That is the reason why this approach is developed in MATLAB language. However, it is know certain tasks written in C language would be significantly faster then this implementation. In future work this should be taken into consideration.

4.2 FAST Detector time dependency

The authors of FAST detector claim it takes 2,6 ms to detect corners in a frame of 768×288 px of resolution [RD05]. However, in this work, this tool is developed in MATLAB with no concerns about its speed. MATLAB language is slow and the programming of this implementation is performed with no optimizations considering memory allocation management, or instructions reduction to improve the computational cost. Which is why

the same task takes several seconds. In the experimental results, the time the FAST detector spends is not presented. It is theoretically considered to be no more than 2,6 ms, considering the used frames always have 320×240 px of resolution.

Several experiments were performed to the FAST detector with different thresholds in a MATLAB environment. A relation between FAST time consuming and the number of detected corners is observed based on those experiments. Figure 4.1 illustrates a graph of that relation. The resolution of the frames used in those experiments is 320×240 pixels.

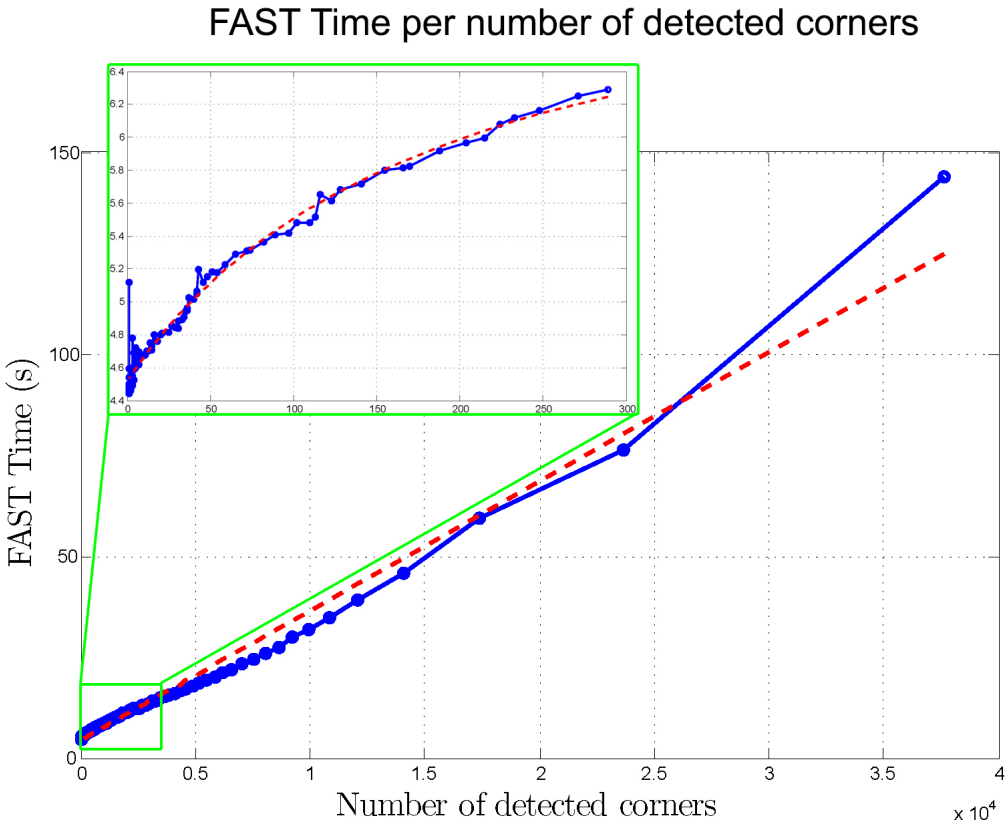

Figure 4.1: Graph of FAST Time per number of detected corners.

By analysing the behaviour of the FAST time consuming in all number of detected corners, it can approximately described by $y = 3,2x \times 10^3 + 4,49$. However, not all the range of detected corners is used. In fact only the interval $[20; 200]$ is relevant for this work. Therefore, a smaller window is presented in the graph to show that interval in more detail. A new behaviour is observed and it approximately fits the following expression $y = \frac{7,4x+866,9}{x+191,8}$.

In conclusion, the FAST detector, built for this work, in MATLAB language consumes

between 5 and 6 seconds.

4.3 Experiences

For the experimental results two collections of ceiling pictures are used. Both are taken from two different laboratories, denoted by lab 1 and lab 2. There are three types of tests for the experimental results, in both labs. The first one (test A) consists in 9 pictures from lab 2, to create a map. The second one (test B) consists in 24 taken pictures in lab 1, to perform a VO process. And the third one (test C) consists in a video scene recorded in lab 2, to perform a map. A VO process only outputs the robot positions of the path. To perform a map it is necessary to run a VO process and concatenate the taken frames afterwards, considering the robot motion.

4.3.1 Picture Map

The 9 pictures are taken by a robot, completely still, in 9 different positions. Therefore, there is no image distortion between frames. Instead, faster lightness variation could cause reduction of repeatability. Figure 4.2 presents the respective frame sequence, sorted by their numbers.

This test used the input parameters of table 4.1.

Table 4.1: Input parameters of test A.

t	24
N_f	30
t_D	40%
t_{ma}	60
t_{mb}	14
$t_{\Delta s}$	8
t_{Δ}	20 px
t_{CDS}	0,8 px
L_{Δ}	1
t_{Θ}	$2,3^{\circ}$
t_{Φ}	4 px
t_{md}	0,8
η_{out}	80%
t_e	4

In this test no corner controller is applied. The FAST threshold is $t = 24$ for every frame. The matching differences is a proportion ($t_D = 40\%$) of the minimum of N_{α}

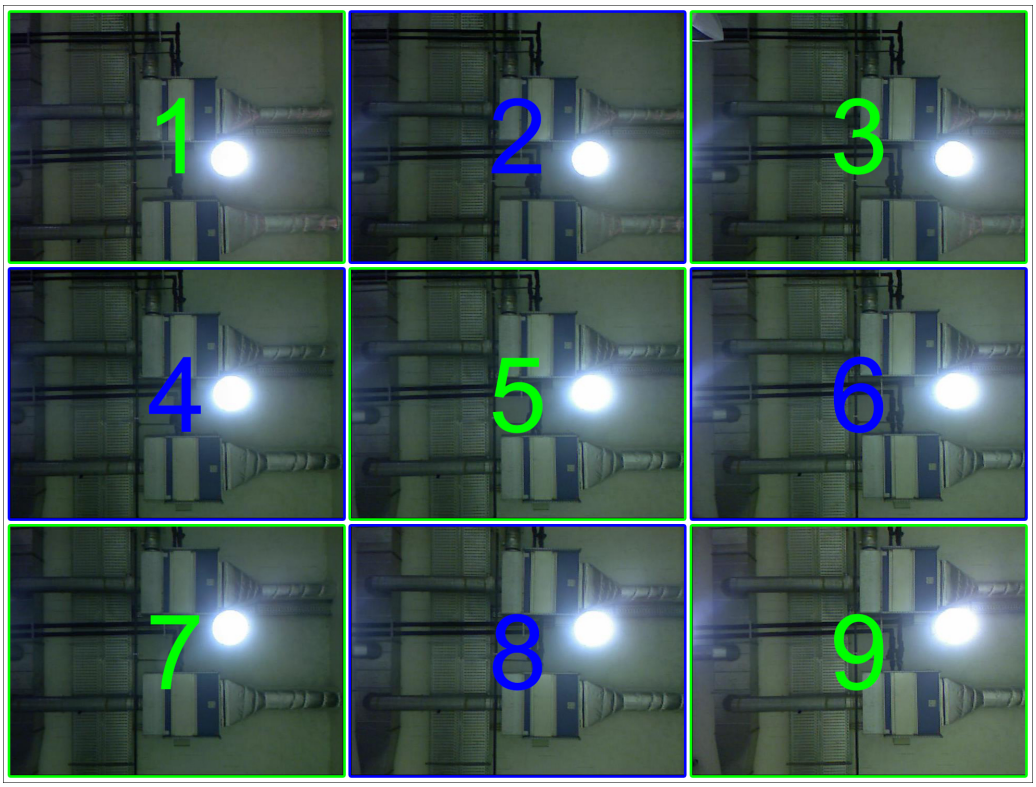

Figure 4.2: Frame sequence for test A.

and N_β, in each frame matching (i.e., $t_D = 0.4 \times min(N_\alpha, N_\beta)$). RANSAC is used with $\eta_{out} = 0,8$, whereas $It = 113$. FMBF provides an outlier portion less than 80%. Therefore, as previously mentioned, RANSAC is used to guarantee that all corner matches are inliers of the chosen model. Considering 113 iterations consumes 10% of the average time of FMBF, $\eta_{out} = 80\%$ can be afforded. It is worth mention though, η_{out} could be reduced in other situations, for time optimization in a milliseconds scale.

Table 4.2 presents the number of detected corners in each frame of this test.

Table 4.2: Number of detected corners in each frame, of test A.

k	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
N_k	93	98	183	73	125	203	54	103	193

Figure 4.3 presents the graph performed with the values of the table.

Considering that the corner controller was not applied in this test, the FAST threshold is constant ($t = 24$). This graph shows how the number of detect corners can vary in frames of the same scenario, differed by displacement, with the same threshold t.

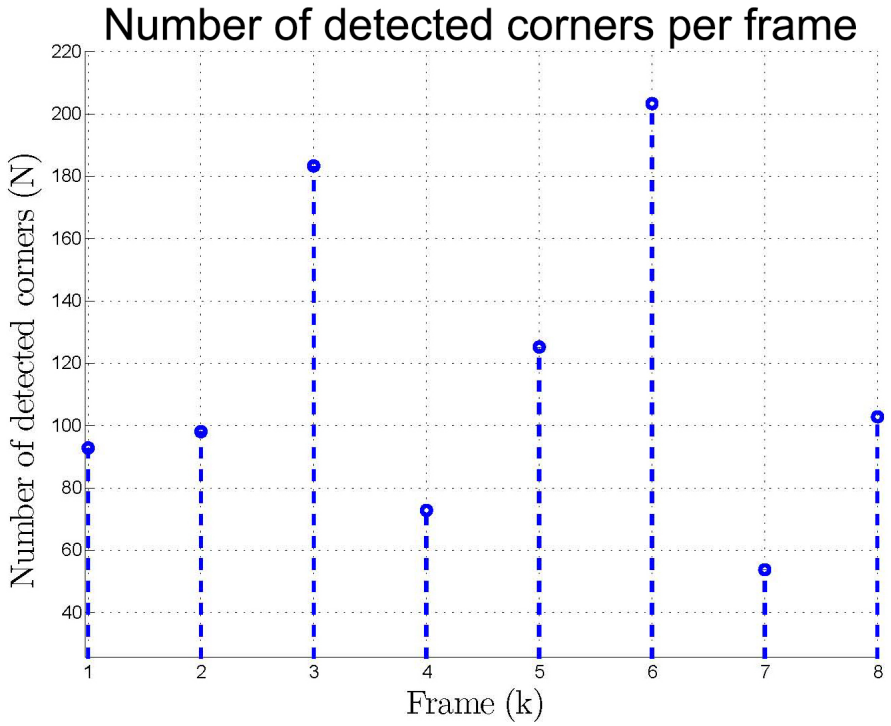

Figure 4.3: Graph of the number of detected corners per frame from test A.

Table 4.3 presents the selected corners by WCA and corner match combinations.

Table 4.3: Number of selected corners and corner match combinations of test A.

Frame Match	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
N_α	29	31	92	32	31	108	30	35
N_β	39	51	32	59	51	30	57	90
$\|\Delta N\|$	10	20	60	27	20	78	27	55
MC	1131	1581	2944	1888	1581	3240	1710	3150

The absolute value of the difference between N_α and N_β is denoted as $|\Delta N|$. The corner match combinations are considered for the worst case, whereas $MC = N_\alpha \times N_\beta$.

Figure 4.4 illustrates a graph, with the selected corners in frame α and β provided by WCA, whose values are presented in the table.

As the graph shows, most of the times, the number of the selected corners are above N_f (which is intended). Concerning this test, the frame match 3 and 6 are a problem, as seen later on. The numbers N_α and N_β are significantly different, which means high MC, and therefore more time consuming in frame matching.

Table 4.4 presents the number of attempts, the time consuming of CS, FMBF and corner matching.

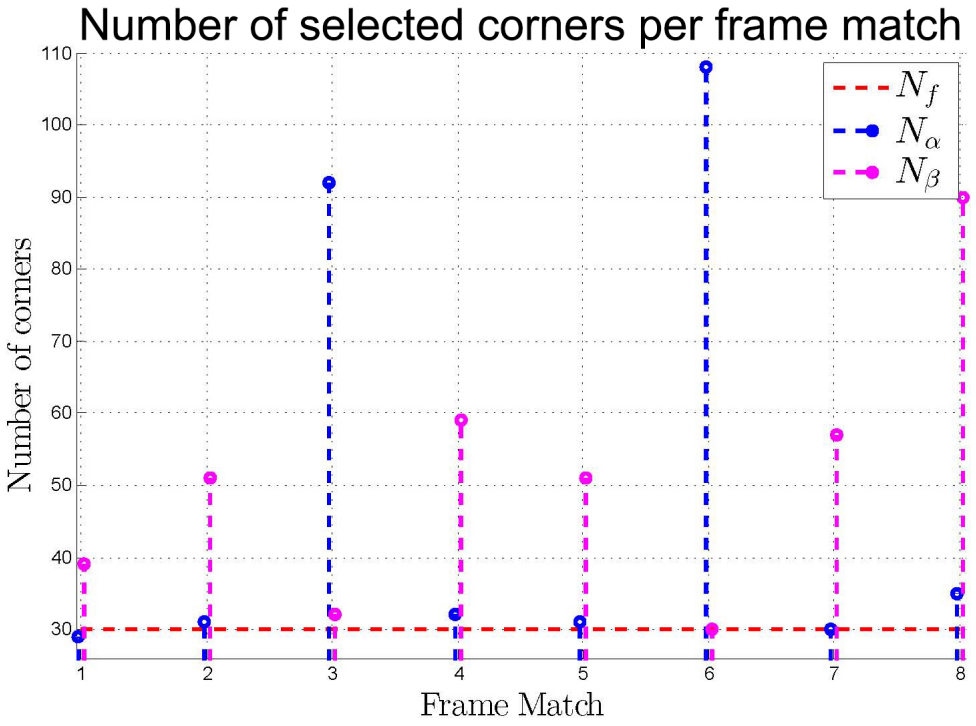

Figure 4.4: Graph of the number of selected corners in frame α and β, of test A.

Table 4.4: Frame match quality and time consuming of test A.

Frame Match	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Attempts	1	1	4	3	4	4	3	1
CS *Creation Time* (ms)	$28,6$	$21,3$	$42,3$	$25,0$	$20,8$	$62,2$	$23,2$	$41,5$
FMBF *Time* (s)	$0,58$	$0,48$	$33,27$	$1,41$	$5,32$	$41,41$	$0,90$	$0,68$
RANSAC *Time* (s)	$0,18$	$0,17$	$0,17$	$0,18$	$0,17$	$0,17$	$0,17$	$0,17$
Matching Time (s)	$0,79$	$0,67$	$33,49$	$1,60$	$5,52$	$41,64$	$1,09$	$0,89$

As mentioned before, attempts is the number of option attempts performed by the FMBF algorithm. CS Creation Time has a smaller scale due to its significance. The Matching Time is the sum of all 3 times.

Figure 4.5 shows the frame matching time consuming, the number of attempts, the difference between N_α and N_β in module, and the number of corner combinations. Those parameters are presented in table 4.3 and 4.4.

As seen in the graph, there is a direct relation in the behaviour of MC and $|\Delta N|$. They both vary equality in the same proportion. The algorithm tried to select N_f in both frames. There is always one of them that works as a reference, where a threshold t_{max} is chosen (considering N_f). Therefore, the corners of that reference frame is selected close to

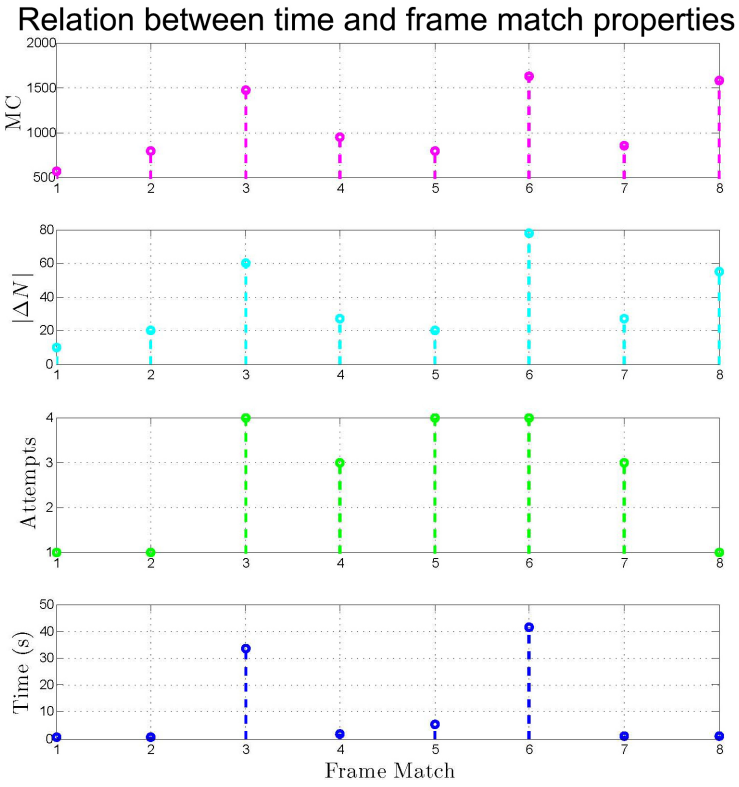

Figure 4.5: Graph of the relation between time and several frame match properties (A).

the number N_f. The corners of the other frame is selected with this same threshold t_{max}. Hence, if both frames have their corners significantly different, then the difference between the number of selected corners ($|\Delta N|$) is high. In this case MC is also high, because it is equal to the multiplication of a number close to N_f with a number significantly higher than N_f. If both frames have N_α and N_β close to N_f, than MC is lower, because it is equal to the multiplication two numbers close to N_f.

In the graph it is also verified that Time is related to MC as expected. In most of the cases, when MC is high, Time is high. With this test there is not enough information to find a relation between the Attempts and others. However, it certainly affected Time, considering each *attempt* is running a whole new FMBF option.

Table 4.5 presents the accepted matches by FMBF M_a, the elected corner matches by RANSAC M_e, and the rejection portion given by $\frac{M_a - M_e}{M_a}$.

Table 4.5: Frame match quality and time consuming of test A.

Frame Match	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
M_a	14	14	14	14	14	14	14	13
M_e	6	5	5	6	13	11	9	5
RANSAC *Rejection* (%)	51,1	64,3	64,3	57,1	7,1	21,4	35,7	61,5

Figure 4.6 illustrates a graph of the values presented in the table.

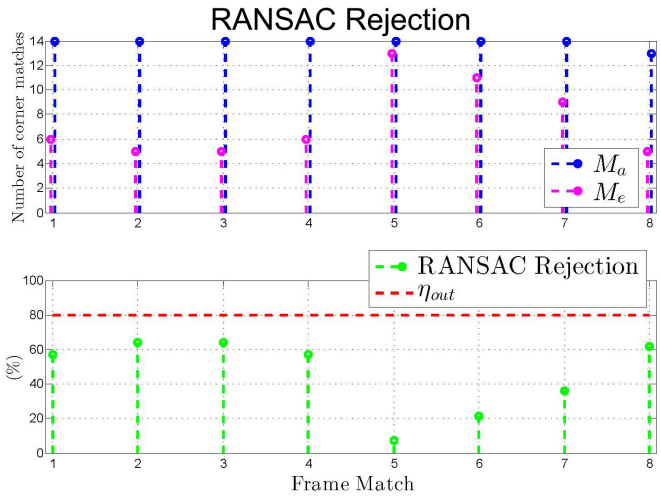

Figure 4.6: Graph of the RANSAC rejection in test A.

The higher the number of elected corner matches, the higher the guarantee that the frame match is correct. Nevertheless, the RANSAC Rejection portion is not strictly associated with the accepted corner match outlier portion. RANSAC could be rejecting high or low portions considering the threshold t_{md}. In any case, the rejection should be demanding, to guarantee that there are not wrong corner matches (outliers).

Table 4.6 presents the robot motion in the previous test.

Table 4.6: Results of the robot motion in test A.

	Motion		
Frame Match	**X** (px)	**Y** (px)	θ $(^\circ)$
1	$-17,38$	$-1,22$	$0,89$
2	$-16,22$	$1,10$	$0,97$
3	$37,59$	$-18,87$	$1,60$
4	$-15,60$	$-1,00$	$-0,99$
5	$-19,00$	$-2,00$	$-0,00$
6	$36,55$	$-18,10$	$0,31$
7	$-18,04$	$0,38$	$0,88$
8	$-17,55$	$-1,73$	$-0,69$

Figure 4.7 illustrates the geometry of the path performed by the robot, in pixels.

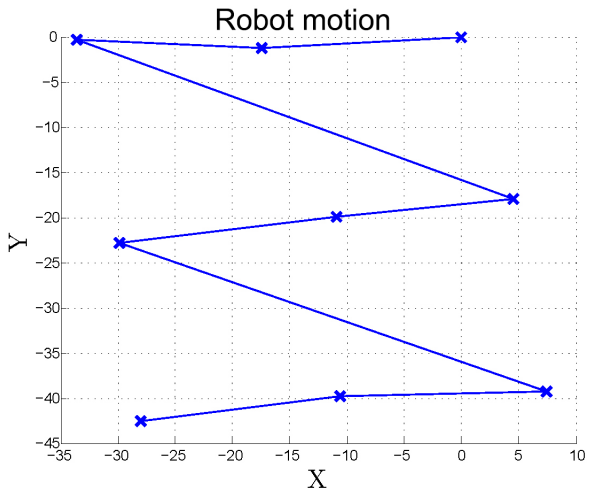

Figure 4.7: Geometry of the path performed by the robot in test *A*.

Here, it is obvious why the frame matches, number 3 (match of position 3 and 4) and number 6 (match of position 6 and 7), are more problematic. Their motion distance are larger than the others, as illustrated in figure 4.7.

Figure 4.8 shows the map performed with the frames acquired by the robot in this test.

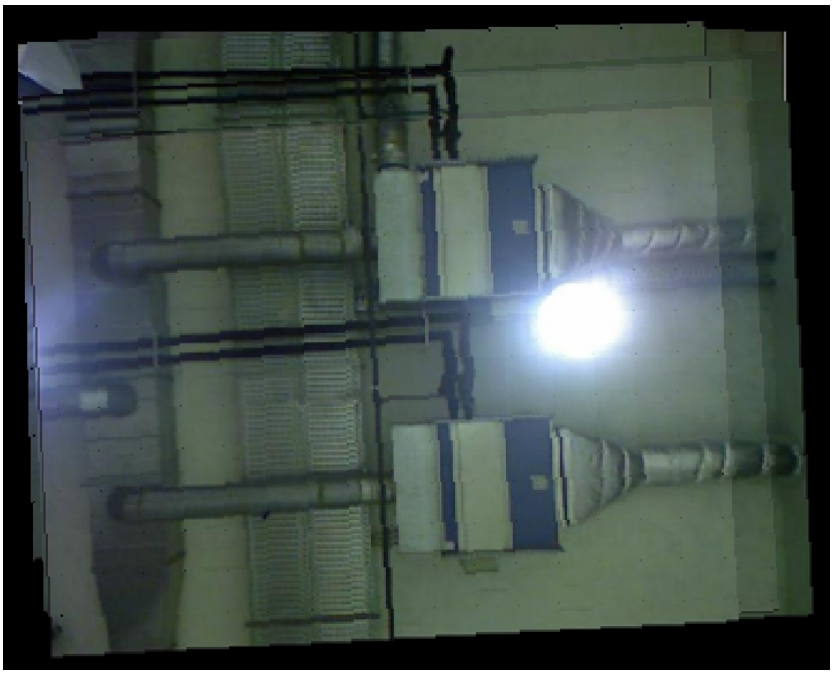

Figure 4.8: Map performed by the taken frames of the test *A*.

There are small imperfections in the map, concerning brightness, due to lightness difference in frames. Also, there are small frame displacement imperfection due to uncertainty errors performed by the algorithm.

4.3.2 Frame Visual Odometry

The second set of 24 pictures are also taken by the a robot, completely still, in 24 different positions. The ceiling used for this test was decorated by some textures, in order to perform contrast from its white colour. Figure 4.9 presents a picture of the ceiling, to show the mentioned textures.

Figure 4.9: Picture of the Lab 1 ceiling.

This test used the input parameters of table 4.7.

This test exemplifies a VO process with a large frame sequence. Therefore, the corner controller is applied so that N_k attempts to follow N_{ask}. In this test the image contrast is high, therefore it is expected to have corner signatures "cleaner" than in the previous test, i.e., with less *distance noise*. Hence, t_D is higher for faster frame matching ($t_D = 0.5 \times min(N_\alpha, N_\beta)$).

Table 4.7: Input parameters of the frame test *B*.

N_f	30
N_{ask}	60
t_D	50%
t_{ma}	60
t_{mb}	14
$t_{\Delta s}$	8
t_Δ	20 px
t_{CDS}	0,8 px
L_Δ	1
t_Θ	$2,3^\circ$
t_Φ	4 px
t_{md}	0,8
η_{out}	80%
t_e	4

Figure 4.10 presents the corner controller effect.

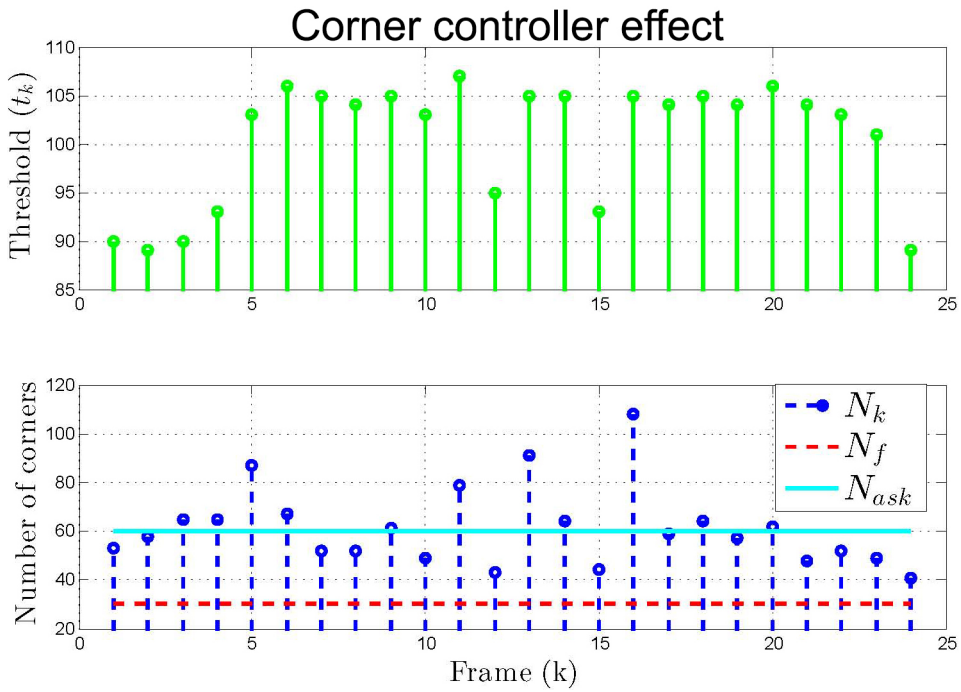

Figure 4.10: Corner controller effect of test *B*.

Most importantly, the number of detected corners N_k can not be bellow N_f, otherwise frame match is not performed. Nevertheless, the lower is N_k the lower is the time that FAST consumes. As illustrated, N_k does not vary linearly with the threshold t_k, and there are some oscillations in N_k, around N_{ask}.

Figure 4.11 illustrates a graph, with the WCA selected corners in frame α and β.

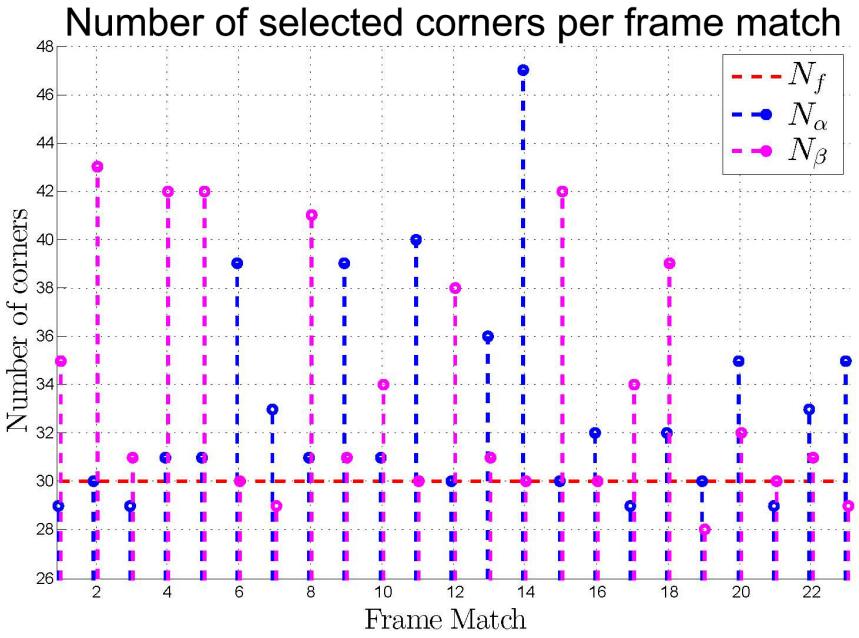

Figure 4.11: Number of collected corners per match of test *B*.

In this test, the number of selected corners in some cases happen to be bellow N_f. However, the differences are small portions that do not affect the frame match.

Figure 4.12 shows the frame matching time consuming, the number of attempts, the difference between N_α and N_β in module, and the number of corner combinations.

In comparison between this test and the previous one, this graph shows smaller differences ΔN, which lead to smaller MC. Therefore, there are less number of attempts and the option 4, which is the most time expensive, is never reached. As a consequence, time is lower.

Figure 4.13 presents the accepted matches by FMBF M_a, the elected frames by RANSAC M_e, and the rejection portion given by $\frac{M_a - M_e}{M_a}$.

The RANSAC Rejection is lower, which means the frame matches have better quality compared to the previous test, considering both tests have the same input parameters.

Figure 4.14 presents a graph with the robot estimated positions performed in this test as blue, and *ground truth* as green, which corresponds to the real robot positions.

Figure 4.15 shows a graph of the error between the real and the estimated robot positions. Such error corresponds to the distance between the position of C_k in the ground truth and the estimated position of $\widehat{C_k}$.

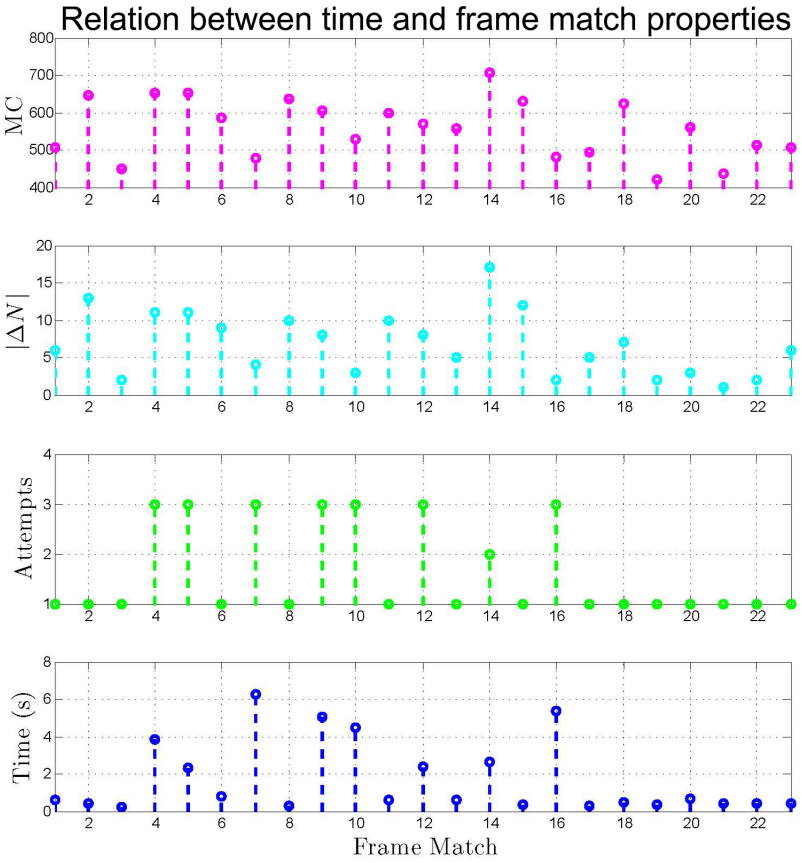

Figure 4.12: Graph of the relation between time and several frame match properties (B).

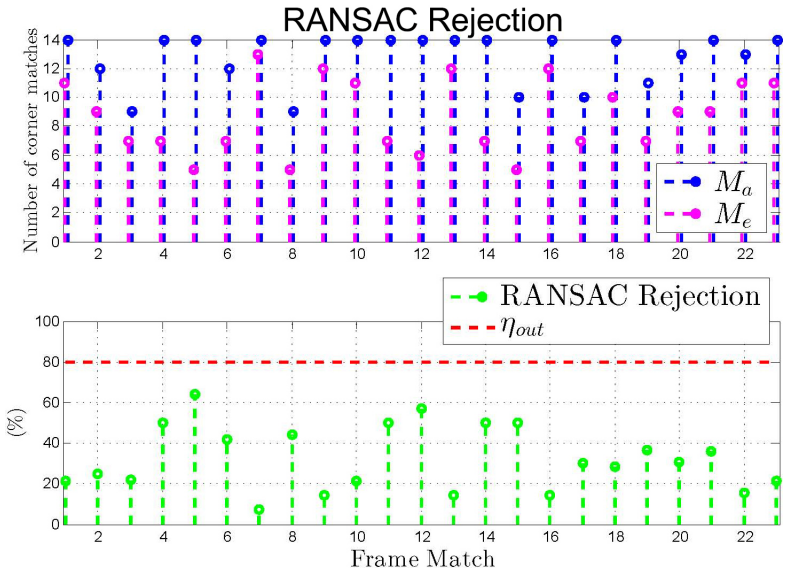

Figure 4.13: Graph of the RANSAC rejection in test B

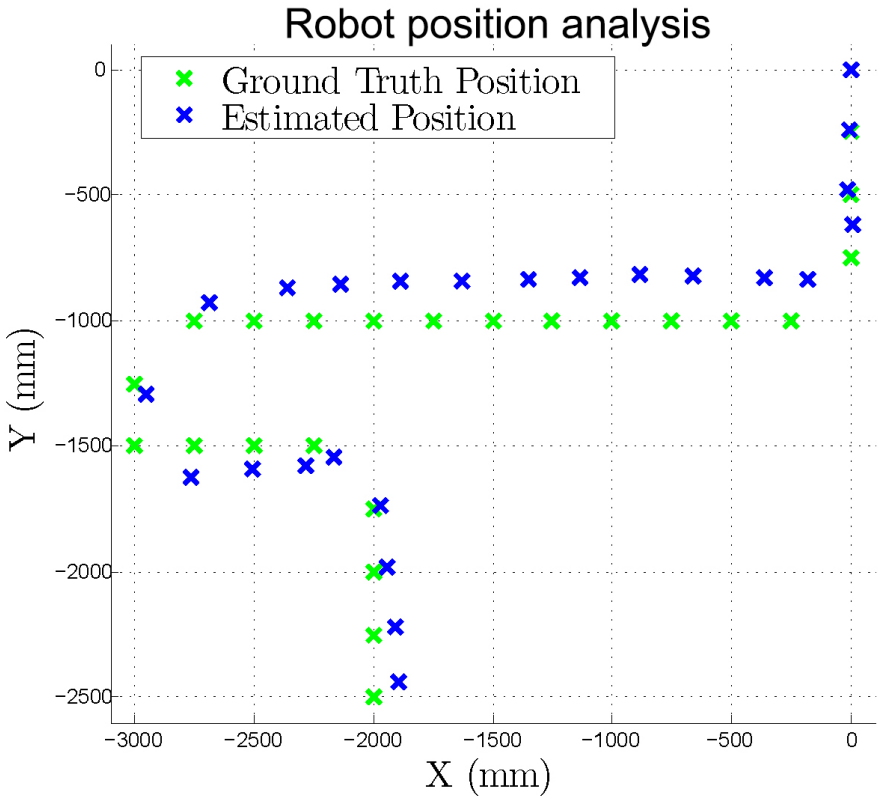

Figure 4.14: Robot position graph analysis of test *B*.

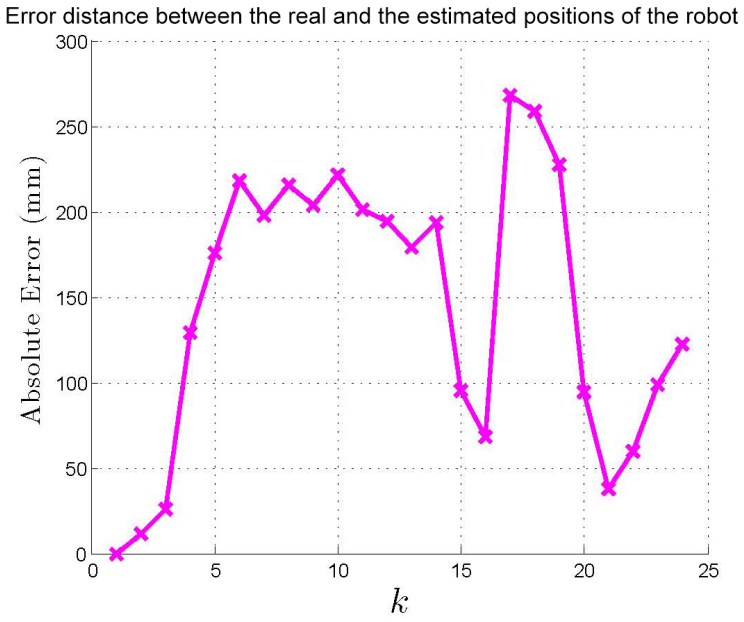

Figure 4.15: Error of the absolute poses in test *B*.

In general, the pose errors are related to three uncertainties. The camera calibration uses measurements of a physical wall, and also pixels measurements from the taken pictures of the wall. The frame pixel units always have uncertainties associated, in each robot motion estimation. To collect the pictures from the ceiling, the robot was placed in its positions by hand, as well as the physical measurements of those positions were made by hand.

Figure 4.16 presents a frame match example of this test (between 14 and 15 frames).

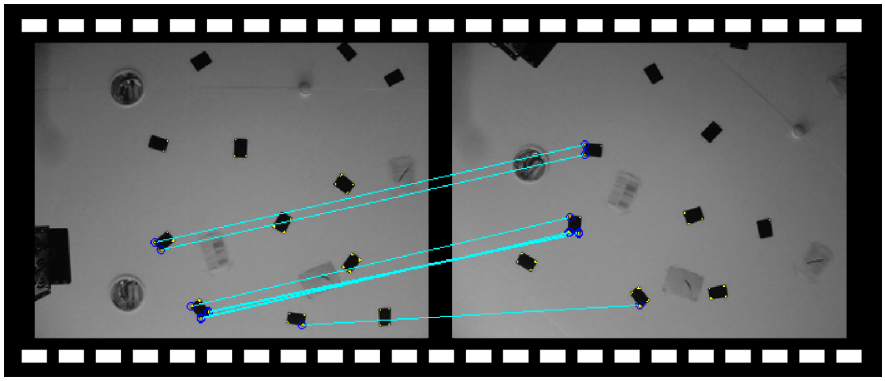

Figure 4.16: Frame match between frames 14 and 15, of test *B*.

Figure 4.17 presents the map performed by the taken frames.

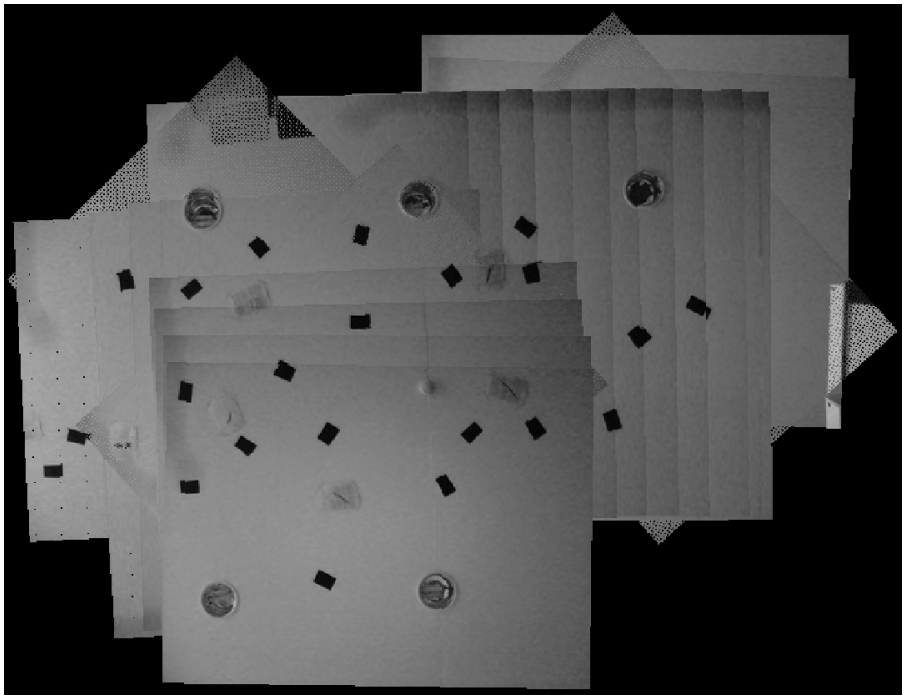

Figure 4.17: Map performed by the taken frames of test *B*.

In this map the lightness differences and displacement imperfection are more evident. The light of this laboratory comes through a window. Therefore, the closer the robot is to the window, the brighter are the frames.

4.3.3 Video Mapping

Computing the robot motion in SI units is just a matter of measuring the distance β_2 and using equation 3.18, $\Omega'(\beta_2)$. In this section the tests do not have ground truth and their validation is performed by manual inspection of each frame match.

To test this system limitations, a 2 minute video of the laboratory 2 ceiling was used for 5 specific parameter configurations. The video scene has 3000 frames, therefore, 25 *Frames Per Second* (FPS). However, the tests were performed with smaller frame sampling rates. One type of test was performed with 1 frame per 6 frames of the scene ($\frac{25}{6}$ FPS), which means 500 frames in total. The other type of test was performed with 1 frame per 10 frames of the scene ($\frac{25}{10}$ FPS), whereby 300 frames in total.

Table 4.8 illustrates the input parameters that are always the same in all tests.

Table 4.8: Input parameters common to the 5 parameter configurations.

$t_{\Delta s}$	8
t_{Δ}	20 px
t_{CDS}	0,8 px
L_{Δ}	1
t_{Θ}	$2,3^{\circ}$
t_{Φ}	4 px
t_{md}	0,8
η_{out}	80%
t_e	4

In fact, those input parameters are recommended to be used as default. Advanced users may perform adjustments in other different occasions. Table 4.9 illustrates the rest of the input parameters, for 5 parameter configurations.

By testing such amounts of frames increases the odds of frame match failures. In fact, a new kind of failure was found, consisting in corners of the elected corner matches being too close to each other, providing a wrong motion estimation. Such failure is designated as *Close Corners* (CC). On the other hand, *Wrong Match* failure (WM) means that there are wrong corner matches in the correspondent frame match. Figure 4.18 shows an example

Table 4.9: Most important input parameters.

Test	**1**	**2**	**3**	**4**	**5**
Frame Sampling Rate (FPS)	$\frac{24}{10}$	$\frac{25}{6}$	$\frac{25}{6}$	$\frac{25}{6}$	$\frac{25}{6}$
Number of frames	300	500	500	500	500
N_f	30	30	30	30	30
N_{ask}	90	90	90	90	90
t_D	50%	50%	55%	45%	45%
t_{ma}	60	60	60	100	200
t_{mb}	14	14	14	14	14

of the CC problem.

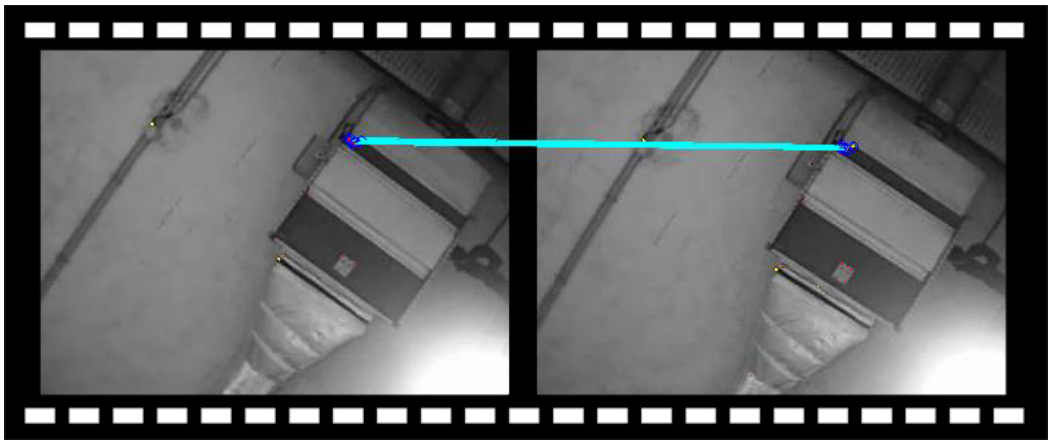

Figure 4.18: Example of Close Corners problem.

Technically, CC is more of a vulnerability occurring with low probability rather than a failure. In fact, CC is the limit case of the corners dispersion. The more the corners are scattered in the frames, lower is the error in the motion estimation. Nevertheless, CC is designated as a failure, for simplicity.

Table 4.10 presents the failures and time results of the 5 parameter configurations.

Table 4.10: Failures and time results.

Test	**1**		**2**		**3**		**4**		**5**	
Failure Type	*WM*	*CC*	*WM*	*CC*	*WM*	*CC*	*WM*	*CC*	*WM*	*CC*
Failure Number	3	2	2	4	5	3	0	1	0	0
Average Time per match (s)	3, 85		2, 46		2, 81		2, 73		3, 44	
Total Time (s)	1154		1232		1401		1364		1720	

In this video the robot moves through a larger area, whereby the drift error becomes significant. The current estimated pose has a substantial accumulated motion error equal

to the sum of the errors performed in all previous motions. From all 5 parameter configurations, the fifth one is the most accurate and, therefore, it is the test in study.

Figure 4.19 shows the corner controller effect on this test.

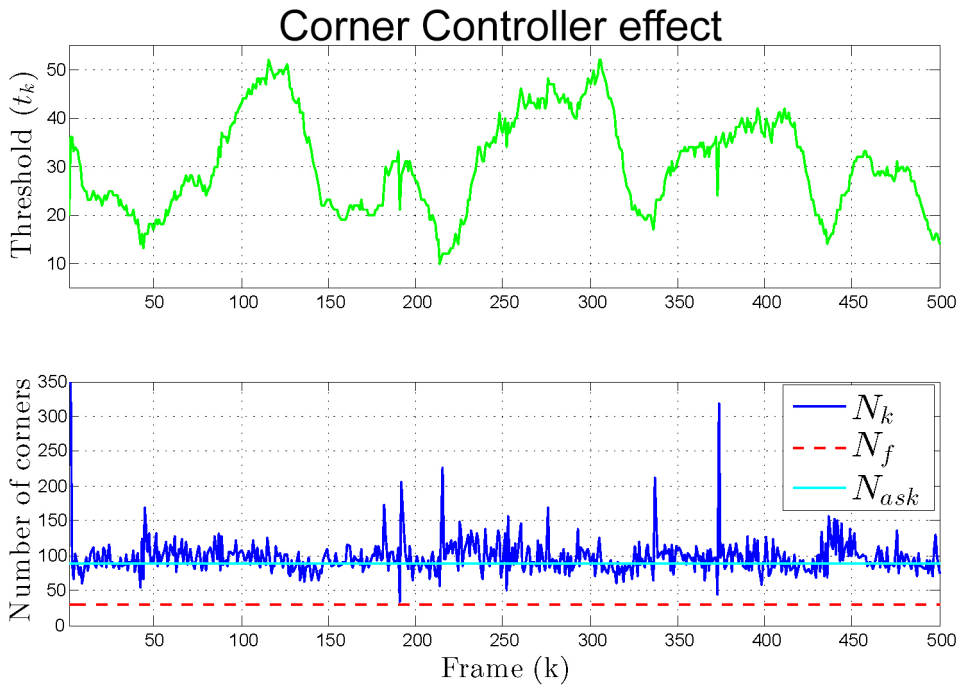

Figure 4.19: Corner controller effect of test C.

Figure 4.20 presents the RANSAC Rejection on this test.

Figure 4.21 presents the frame matches time consuming of this test.

Figure 4.22 presents a graph with the robot positions performed in this test.

Figure 4.23 shows the map performed by the taken frames of this test.

The corner controller effect and the RANSAC Rejection as an appropriate behaviour. In order for this system to work in a sequence of 500 frames, it needs to be more tolerant, to increase the chances of solving all the difficult frame matches. That is why t_{ma} needs to be high. As a consequence the time consuming rises, considering the options 1, 2 and 3 of FMBF approvals always need to run t_{ma} comparisons each. However, the inevitable drift is observed in the map of figure 4.23. An evident drift is seen throughout the path of the red dashed line, whereas the robot groundtruth is approximately the green dashed line. The yellow squares indicate locations that should be the same.

In order to reduce the drift, bundle adjustment and loop closure are recommended techniques. Using a window of several frames to perform frame matching reduces the

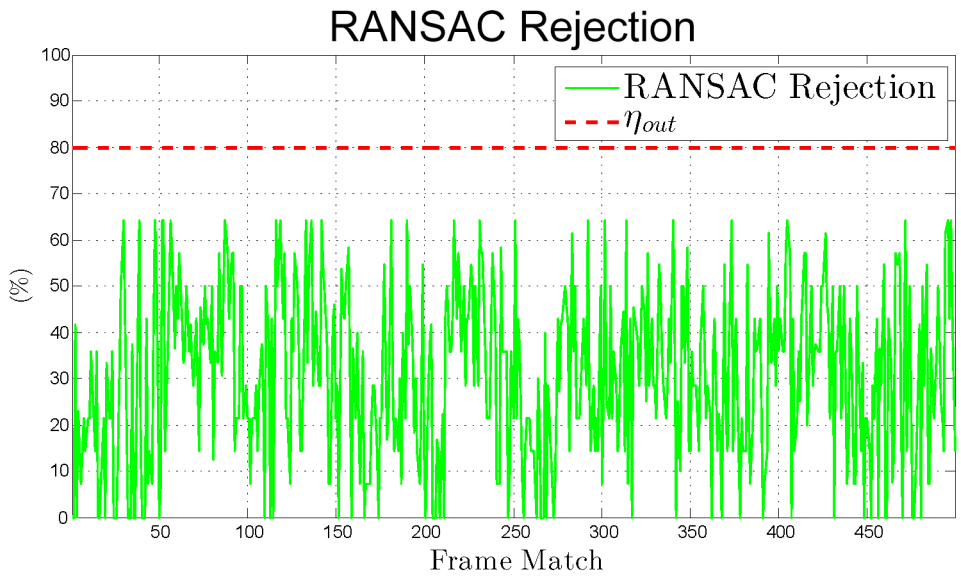

Figure 4.20: RANSAC Rejection of test C.

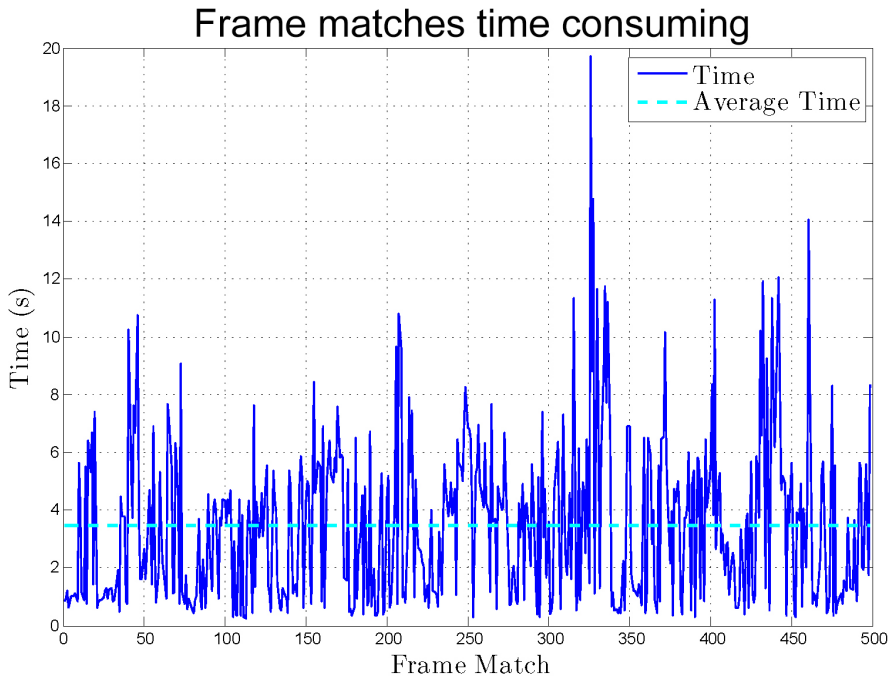

Figure 4.21: Time consuming of the frame matches of test C.

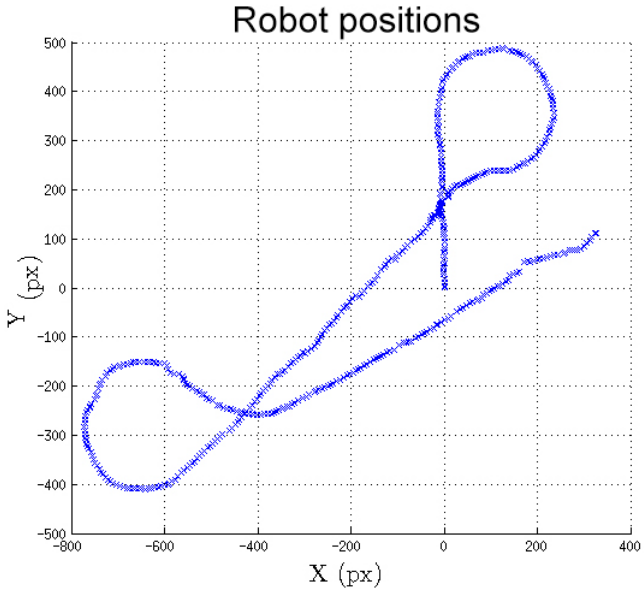

Figure 4.22: Robot positions graph of test C.

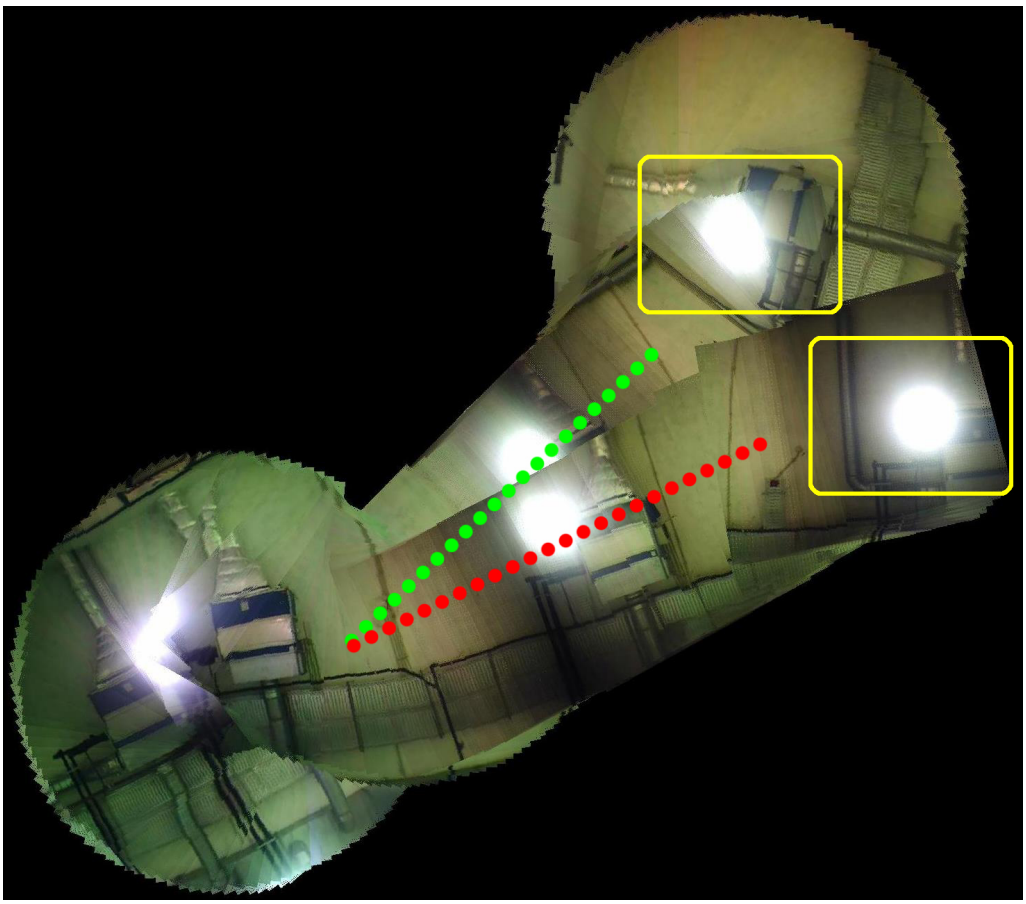

Figure 4.23: Map performed by the taken frames of test C.

matching errors, delaying the drift. Loop closure updates the robot location when it passes through a previous known location, therefore removing the drift since then. For further suggestions to reduce drift see section 5.2.

4.4 Error relation

It is important to be aware of this system limitations, regarding the motion between pairs of frames. As mentioned in chapter 3, pairs of frames need to have an overlapped area in order to perform the frame matching. Also, the size of that area is important for an accurate motion estimation. Therefore, this system was tested with several random motions between pairs of frames, in order to understand the error behaviour depending on displacement and orientation of the robot. To eliminate the human uncertainty, instead of using real motion, these tests were performed in the corner signature simulator. The used inputs are the same used in test C, of section 4.3.3. Figure 4.24 illustrates a 3 dimensional graph, where the error is depending on the displacement and the orientation of the robot.

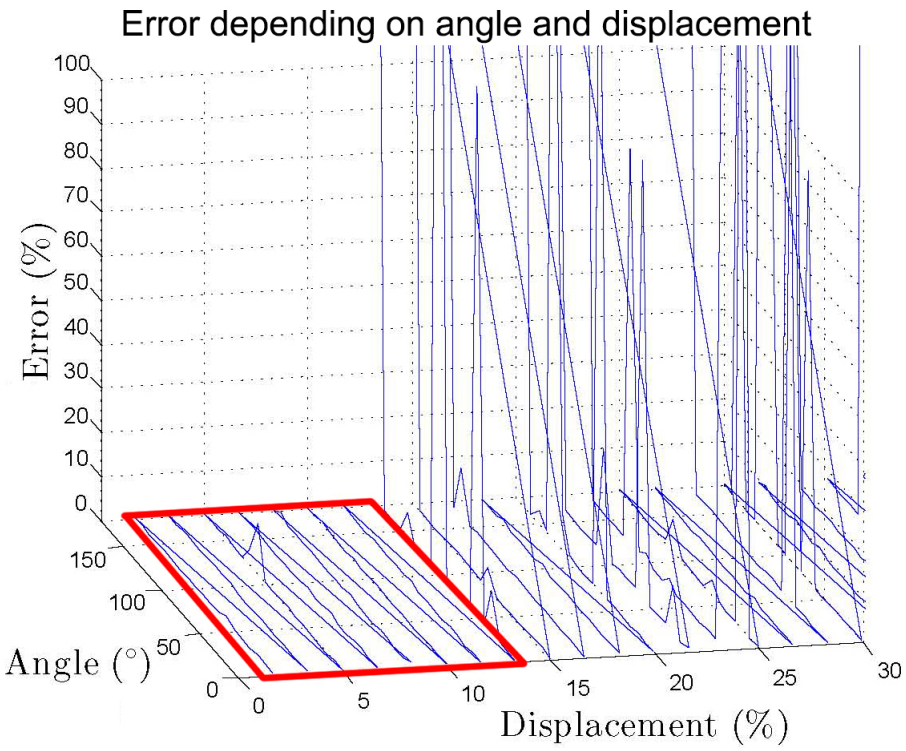

Figure 4.24: Graph of the error depending of the robot displacement and orientation, between pairs of frames.

The displacement is the distance, in pixels, between poses. It is considered that the maximum possible robot displacement between frames is half of the lower dimension of the frame. The frame dimensions are 340 × 240 px, whereas the lower dimension is 240, and the half is it is 120. Therefore, both the error and the displacement are normalized in percentage, whereby 120 px is 100%.

The figure shows an area, highlighted with a red rectangle, where the error is admissible, lower than 10%.

Figure 4.25 shows the graph only from the displacement perspective.

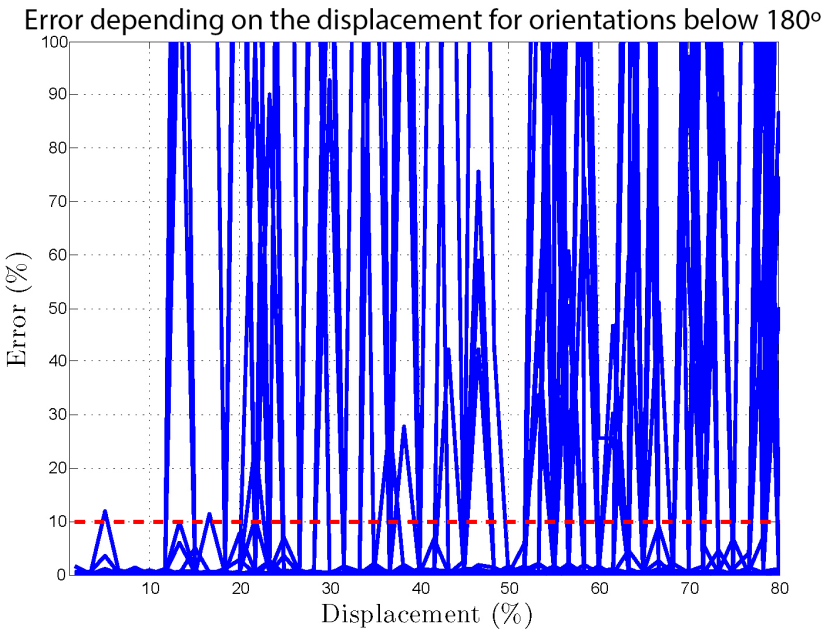

Figure 4.25: Graph of the error depending of the robot displacement for orientations below 180°.

The error behaviour, from the displacement perspective is not admissible, considering it is larger than 10%. In this case, all tests between 0° and 180° are presented. Decreasing the angles range to the half, between 0° and 90°, the graph of figure 4.26 is verified.

With motion estimations, with orientation between 0° and 90°, the displacement can be used between 0% and 13%. And if the orientation angle is reduced to 30°, figure 4.27 shows the error behaviour of this situation.

With motion estimations, with orientation between 0° and 30°, the displacement can be used between 0% and 22%.

This means there is a trade-off between the maximum possible displacement and

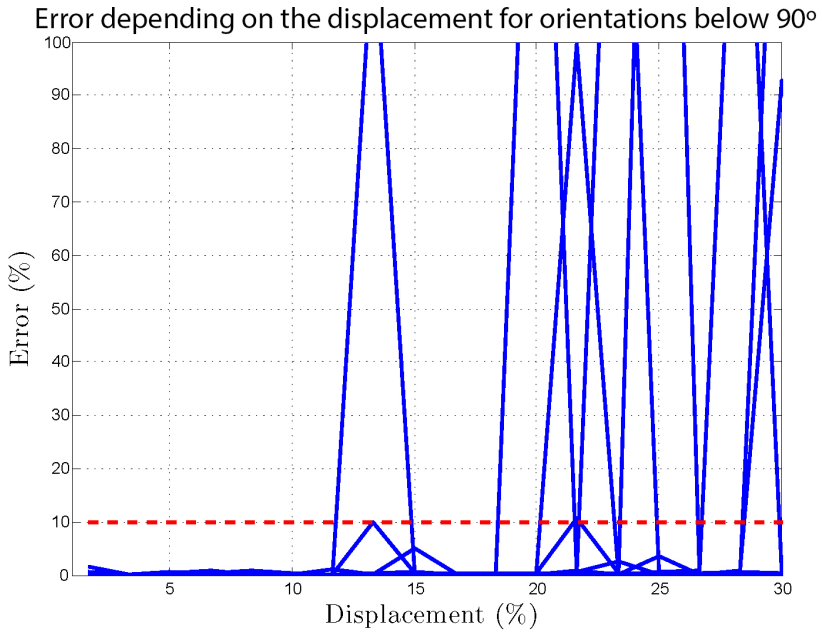

Figure 4.26: Graph of the error depending of the robot displacement for orientations below 90°.

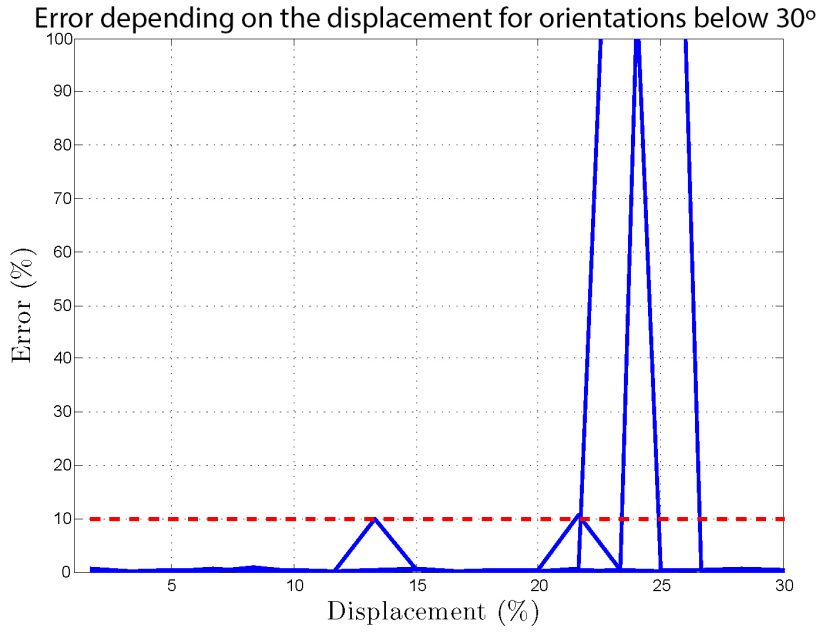

Figure 4.27: Graph of the error depending of the robot displacement for orientations below 30°.

orientation of the robot between two frames. Such limitation should be respected, in order to accomplish correct motion estimations.

4.5 Summary

In this implementation, FAST is very slow compared to the speed that its authors claim to have. A relation between the FAST detector time consuming and the number of detected corners was created, through several tests. And follows the behaviour: $y = \frac{7{,}4x+866{,}9}{x+191{,}8}$ in the interval $[20; 200]$.

In the experimental results three types of tests are performed. The first one is picture map, which used frames taken completely sill, from lab 2. In shows that WCA works, attempting to select N_f corners in each frame. The robot motion estimations is shown as well as the map. The map was built through the robot estimated poses. There are several imperfection in the map due to the brightness and to light differences. Also, displacement imperfections, due to uncertainty errors performed by the motion estimation.

The test frame visual odometry uses frames taken completely still, from lab 1. It shows that the corner controller attempts to provide the number of asked corners. This test is validated with the groundtruth. The absolute error of each frame transformation is approximately 150 mm average.

The video mapping consists in a video of 3000 frames. However, the FPS are reduced. From all 5 parameter configurations, the fifth one, with 300 frames, is the most accurate and used for further analysis. In this test it is verified a new vulnerability called Close Corners. This contributes to drift raises. In the map it is visible a large drift, contributing to wrong pose estimations. The drift is caused by small error accumulations throughout the motion estimations.

An error relation is created to provide the system limitations regarding the orientation and the displacement. The range of angles between 0° and 180° should not be used between robot motions. With ranges of angles between 0° and 90° the displacement can be used until 13%. And with ranges between 0° and 30° the displacement can be used until 22%.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This dissertation implemented a VO system for a mobile robot path estimation. With a camera pointed to the ceiling, frames were analysed and matched in order to estimate the robot motion. The FAST detector was used in each frame to detect the landmarks called corners. RANSAC is a widely used method in the computer vision literature, for model estimation. However, deduction proved it becomes slow with the growth of data. Therefore, a new method named FMBF was developed in this work to accelerate the motion estimation process. Corner Signatures are created to be used in the FMBF algorithm. These signatures consist in Corner Distance Signatures and Corner Triangle Signatures. Corner Distance Signature comparisons are very efficient, but very slow, while Corner Triangle Signature comparisons are fast, but not so efficient. FMBF is a method that attempts to find the balance in the trade-off between speed and efficiency, using Corner Distance Signatures and Corner Triangle Signatures comparisons, between pairs of frames. FMBF outputs corner matches with a low outlier portion. Such low outlier portion makes RANSAC running significantly faster.

In order to reduce the corner detection and the corner matching time consuming, a corner controller was developed in this work to control the number of detected corners. However, experiments proved that when the corner controller had deviations, the match combinations raised quadratically, increasing the outlier portion significantly. Therefore, the method *Window Corner Ajustment* (WCA) was created.

Before the frame matching process, the corner matches are pre filtered by WCA with a reduced time consuming. WCA performs a filtering in pairs of frames, based on all the maximum threshold each corner has.

The selected corner matches are then used for motion estimation. The corners were considered to be static references in the environment, and it was through them that the motion was calculated.

The maps was created using the estimated poses for frames concatenations.

In the video mapping test it was verified a new vulnerability called Close Corners. This contributes to drift raises. In the map it was visible a large drift, contributing to wrong pose estimations. The drift was caused by small error accumulations throughout the motion estimations. In this test, a new vulnerability was observed. As mentioned, and observed in figure 4.18, *Close Corners* may cause high motion estimation errors. This vulnerability also triggered a new understanding. The more the corners are scatted all over the frames, better is the estimation motion accuracy, and the more the corners dispersion were close to the *Close Corners* situation, worse is the estimation motion accuracy. Therefore, VO drift increases when the corners dispersion tends to a *Close Corners* situation.

Using MATLAB language it is not possible to perform an online path estimation with a reasonable speed, as seen in the visual odometry test of section 4.3.2. MATLAB was used for fast programming, as a high level programming language it is. Also, the programming code is not fully optimized in terms of temporal complexity. Therefore, this whole system can be used in a indoor robot, working in a stop-and-go style. It is advised to be used only for a short distance corresponding to several dozens of frames.

The work developed in this dissertation was already used in experiments, combined with other motion sensors, which led to a paper presented in an international conference, [CECV14].

5.2 Future Work

Considering the MATLAB language mentioned speed problem, for future work, it is proposed to implement this approach in *C* language, as well as, optimize the code in terms

of memory allocation, reducing the number of variables and always use pointers wherever it is possible.

To save time in corner detection, next step should be to use the FAST-ER detector instead of FAST detector.

In terms of the VO system reliability, the drift should be reduced using window bundle adjustment and loop closure detection. By using window bundle adjustment, several frame matches, between the current frame and previous frames, are accounted to the current motion estimation, reducing the local motion error. Using loop closure, all the drift performed since the previous recognized location is eliminated. Also, to reduce *Close Corners* situations, the vector of the ceiling reference (figure 3.30) should be accepted only if higher than a certain stipulated threshold. And, the virtual points that create the ceiling reference (3.1.6) need to be chosen so that the vector has a length as high as possible.

In order to create a robust system for motion estimation, in future work, it is important to combine VO with other motion sensors, such as IMU and wheel odometry, to increase the redundancy of the system, fusing the information of all techniques.

After all these important steps, the next step of the future work is a 3D application. The ceiling and the floor could be irregular, not limited to parallel planes, increasing the robot functionalities in terms of degrees of freedom.

References

[AF88] N. Ayache and O. Faugeras. Building, registrating, and fusing noisy visual maps. *International Journal of Robotics Research*, 7:45–65, December 1988.

[AK07] M. Agrawal and K. Konolige. Rough Terrain Visual Odometry. *International Conference on Advanced Robotics (ICAR)*, August 2007.

[AKB08] M. Agrawal, K. Konolige, and M. Blas. Censure: Center surround extremas for realtime feature detection and matching. *Proc. European Conf. Computer Vision*, pages 102 – 115, 2008.

[BDw94] B. Barshan and H. F. Durrant-whyte. Orientation Estimate for Mobile Robots using Gyroscopic Information. *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, 3:1867–1874, September 1994.

[BDW95] B. Barshan and H. F. Durrant-Whyte. Inertial navigation systems for mobile robots. *IEEE Transactions on Robotics and Automation*, 11(3):328–342, June 1995.

[BEFW97] J. Borenstein, H. R. Everett, L. Feng, and D. Wehe. Mobile Robot Positioning: Sensors and Techniques. *Journal of Robotic Systems*, 14(4):231–249, April 1997.

[Bha10] N. Bhatia. Survey of nearest neighbor techniques. *International Journal of Computer Science and Information Security*, 8(2):302–305, 2010.

[BK87] J. Borenstein and Y. Koren. Motion Control Analysis of a Mobile Robot. *Journal of Dynamic Systems, Measurement, and Control*, 109(2):73–78, Jun 1987.

[BKP92] R. H. Byrne, P. R. Kiarer, and J. B. Pletta. Techniques for Autonomous Navigation. *Sandia Report SAND92-0457*, March 1992.

[BL97] J. S. Beis and D. G. Lowe. Shape Indexing Using Approximate Nearest-Neighbour Search in High-Dimensional Spaces. *Conference on Computer Vision and Pattern Recognition*, pages 1000 – 1006, 1997.

[Bor96] J. Borenstein. Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation*, 12(6):869–880, December 1996.

[BTG06] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. *Proc. ECCV*, pages 404 – 417, 2006.

[CECV14] F. Coito, A. Eleuterio, Fernando Coito, and S. Valtchev. Tracking a Mobile Robot Position Using Vision and Inertial Sensor. *DoCEIS14 - 5th Doctoral Conference on Computing, Electrical and Industrial Systems*, 423:201 – 208, April 2014.

[CL85] R. Chatila and J. P. Laumond. Position referencing and consistent world modeling for mobile robots. *IEEE Journal of Robotics and Automation*, 2:138–143, March 1985.

[Cro89] J. L. Crowley. World Modeling and Position Estimation for a Mobile Robot Using Ultrasonic Ranging. *IEEE Journal of Robotics and Automation*, 2:674–680, May 1989.

[CSS04] P. Corke, D. Strelow, and S. Singh. Omnidirectional visual odometry for a planetary rover. *International Conference on Intelligent Robots and Systems (IROS)*, 4:4007–4012, October 2004.

[DNB+12] N. Dey, P. Nandi, N. Barman, D. Das, and S. Chakraborty. A Comparative Study between Moravec and Harris Corner Detection of Noisy Images Using Adaptive Wavelet Thresholding Technique, September 2012.

[DW87] H. Durrant-Whyte. Uncertain geometry in robotics. *IEEE International Conference on Robotics and Automation*, 4:851–856, March 1987.

[DwB06] H. Durrant-whyte and T. Bailey. Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. *IEEE Journal of Robotics and Automation*, 13(2):99 – 110, June 2006.

[FB81] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, June 1981.

[FG87] W. FÃ¶rstner and E. GÃ¼lch. A fast operator for detection and precise location of distinct points, corners and circular features. *Intercommission Conference on Fast Processing of Photogrammetric Data*, 1987.

[FS11] F. Fraundorfer and D. Scaramuzza. Visual Odometry - Part I: The First 30 Years and Fundamentals. *IEEE Robotics and Automation Magazine*, 18(4):80 – 92, December 2011.

[FS12] F. Fraundorfer and D. Scaramuzza. Visual odometry - part ii: Matching, robustness, optimization, and applications. *IEEE Robotics and Automation Magazine*, 19(2):78 – 90, June 2012.

[GD00] C. Geyer and K. Daniilidis. A unifying theory for central panoramic systems and practical applications. *Proc. European Conf. Computer Vision*, pages 445 – 461, 2000.

[GT94] C. Gourley and M. Trivedi. Sensor Based Obstacle Avoidance and Mapping for Fast mobile Robots. *Proceedings of the IEEE International Robotics and Automation*, 2:1306–1311, May 1994.

[How08] A. Howard. Real-time stereo visual odometry for autonomous ground vehicles. *Intelligent Robots and Systems*, pages 3946–3952, September 2008.

[HP88] C. G. Harris and J. M. Pike. 3D Positional Integration from Image Sequences. *In Proc. Alvey Vision Conference*, 6(2):87–90, May 1988.

[HS88] C. Harris and M. Stephens. A Combined Corner and Edge Detector. *In Proceedings of The Fourth Alvey Vision Conference*, pages 23.1 – 23.6, September 1988.

[KAS11] K. Konolige, M. Agrawal, and J. SolÃ . Large Scale Visual Odometry for Rough Terrain. *International Symposium on Robotics Research*, 66:201 – 212, 2011.

[KCS11] L. Kneip, M. Chli, and R. Siegwart. Robust Real-Time Visual Odometry with a Single Camera and an IMU. pages 16.1–16.11, 2011.

[Kne12] L. Kneip. Laurent kneip. `http://www.laurentkneip.de/research.html`, 2012. [Online; accessed 10-July-2014].

[Low03] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.*, 20(2):91 – 110, 2003.

[Low04] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[Mai05] M. W. Maimone. Visual Odometry on the Mars Exploration Rovers. *IEEE International Conference on Systems, Man and Cybernetics*, pages 903–910, October 2005.

[MCM07] M. Maimone, Y. Cheng, and L. Matthies. Two Years of Visual Odometry on the Mars Exploration Rovers. *Journal of Field Robotics*, 24(3):169–186, March 2007.

[MGD07] A. Makadia, C. Geyer, and K. Daniilidis. Correspondence-free Structure from Motion. *International Journal of Computer Vision*, 75(3):311–327, December 2007.

[Mor80] H. P. Moravec. Obstacle Avoidance and Navigation in the real world by a seeing robot rover. 1980.

[MS87] L. Matthies and S.A. Shafer. Error modelling in stereo navigation. *IEEE Journal of Robotics and Automation*, 3(3):239 – 248, June 1987.

[NNB04] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry. *IEEE Computer Society Conference of Computer Vision and Pattern Recognition (CVPR)*, 1:652–659, July 2004.

[OMSM00] C. Olson, L. Matthies, M. Schoppers, and M. Maimone. Robust stereo ego-motion for long distance navigation. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:453 – 458., Jun 2000.

[RD05] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. *Tenth IEEE International Conference on Computer Vision (ICCV)*, 2:1508–1515, October 2005.

[RPD10] E. Rosten, E. Porter, and T. Drummond. Faster and better: a machine learning approach to corner detection. *IEEE transactions on pattern analysis and machine intelligence*, 32(1):105–19, January 2010.

[SB97] S. M. Smith and J. M. Brady. SUSAN - A New Approach to Low Level Image Processing. *International Journal of Computer Vision*, 23(1):45–78, May 1997.

[SCDE95] E. Stella, G. Cicirelli, A. Distante, and I. Elaborazione. Position Estimation for a Mobile Robot using Data Fusion. *IEEE International Symposium on Intelligent Control*, pages 565–570, Aug 1995.

[SD+99] S. Scheding, , G. Dissanayake, E. M. Nebot, and H. Durrant-Whyte. An experiment in autonomous navigation of an underground mining vehicle. *IEEE Transactions on Robotics and Automation*, 15(1):85–95, February 1999.

[SLL01] S. Se, D. Lowe, and J. Little. Vision-based mobile robot localization and mapping using scale-invariant features. *IEEE International Conference on Robotics and Automation (ICRA)*, 2:2051–2058, 2001.

[SSC87] R. Smith, M. Self, and P. Cheeseman. Estimating Uncertain Spatial Relationships in Robotics. *IEEE International Conference on Robotics and Automation*, 4(8):167–193, March 1987.

[ST94] J. Shi and C. Tomasi. Good features to track. *Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, Jul 1994.

[TW09] M. Toews and W. Wells. SIFT-Rank: Ordinal Description for Invariant Feature Correspondence cite as. *Computer Vision and Pattern Recognition*, pages 172 – 177, June 2009.