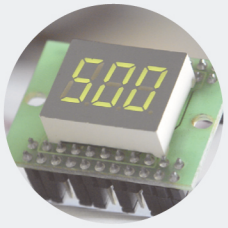
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






Real Time Stereo Cameras System Calibration Tool and Attitude and Pose Computation with Low Cost Cameras

NELSON CASIMIRO CASTRO DE CAMPOS
novembro de 2017

POLITÉCNICO
DO PORTO

Real Time Stereo Cameras System Calibration Tool and Attitude and Pose Computation with Low Cost Cameras

Nelson Casimiro Castro Campos
1090427@isep.ipp.pt

Master Degree of Electrotechnical and Computer Engineering
Specialization in the area of Autonomous Systems
Department of Electrotechnical Engineering

Instituto Superior de Engenharia do Porto

2017

Dissertation for Satisfaction of Requirements from Master Degree of Electrotechnical and Computer Engineering

Candidate: Nelson Casimiro Castro Campos
1090427@isep.ipp.pt

Advisor: Alfredo Manuel Oliveira Martins
aom@isep.ipp.pt

Master Degree of Electrotechnical and Computer Engineering
Specialization in the area of Autonomous Systems
Department of Electrotechnical Engineering

Instituto Superior de Engenharia do Porto

November 7, 2017

Special Thanks

First of all, I would like to thank my parents and sister for their unconditional support, for understanding and patience, not only during the realization of this thesis but also throughout my life where I have never lacked anything. You all are unique. I would not have a hundred lives to repay you for all the efforts you have ever done for me. With pride, I can say that you have made me the person I am today. Thank you for all the affection and for following sincerely my joys.

I would also like to thank my uncles and grandparents who have always been a great pillar in my life. Present in the most difficult moments but also by my side to celebrate all the victories and joys along this route.

No less important, I leave a huge thank you to my lifemate Ana and her parents for their presence in such a dedicated way in my life. Thank you for your presence, support and strength in the difficult moments that we have gone through together, always without neglecting what unites us and for feeling my victories as yours.

A big thank you to my friend Miguel Morim Moreira for being on my side during this long journey practically from the beginning. For the successes we have achieved together, for the difficulties that made us grow as persons, friends and good professionals. I wish you all the best that life has because you deserve it.

Thanks also to my advisor, Engineer Alfredo Manuel Oliveira Martins for his great work and assertive, organized and well structured support that got me through to good port. His dedication and hard work contributed to the, at least in my opinion, great success of this work.

Thanks also to André Faria, João Ribeiro, Guilherme Amaral, Joel Oliveira, Marco Gonçalves and other friends and colleagues who, in one way or another, were accompanying my work and taking a "little" of their own time to help me and try to find out news about my progress.

Agradecimentos

Em primeiro lugar, gostaria de agradecer aos meus pais e irmã pelo apoio incondicional, pela compreensão e paciência, não só durante a realização desta tese mas também ao longo de toda a vida em que nunca me faltou nada. Vocês são únicos. Não me chegariam cem vidas para vos retribuir todos os esforços que sempre fizeram por mim. Com orgulho, posso dizer que me tornaram na pessoa que sou hoje. Obrigado por todo o carinho e por fazerem questão de me acompanhar em todas as alegrias.

Gostaria também de agradecer aos meus tios e avós, que sempre foram um grande pilar na minha vida. Estiveram presentes nos momentos mais dificeis, mas também ao meu lado para festejar todas as vitórias e alegrias ao longo deste percurso.

Não menos importante, um enorme obrigado à minha companheira Ana Furtado e aos seus pais pela forma como, também eles, estão presentes de uma forma tão dedicada na minha vida. Obrigado pela presença, apoio e força nos momentos que ultrapassámos juntos, sempre sem descurar o que nos une e por sentirem as minhas vitórias como vossas.

Um grande obrigado ao meu amigo Miguel Morim Moreira por estar também do meu lado durante este longo percurso, praticamente desde o início. Pelos sucessos que alcançámos juntos, pelas dificuldades que nos fizeram crescer enquanto pessoas, amigos e profissionais. Desejo-te tudo do melhor que a vida tem porque tu mereces.

Obrigado também ao Eng. Alfredo Manuel Oliveira Martins pelo seu ótimo trabalho e apoio assertivo, organizado e bem estruturado, que me levaram a bom porto. O seu esméro trabalho contribuiu para que o meu se pudesse concretizar de forma, pelo menos a meu ver, tão bem sucedida.

Obrigado também ao André Faria, João Ribeiro, Guilherme Amaral, Marco Gonçalves, Joel Oliveira, bem como a todos os outros amigos e colegas que, de uma forma ou de outra foram acompanhando e tirando um "pouco" do seu tempo para me apioarem e tentarem saber do meu percurso.

Abstract

The Engineering in autonomous systems has many strands. The area in which this work falls, the artificial vision, has become one of great interest in multiple contexts and focuses on robotics. This work seeks to address and overcome some real difficulties encountered when developing technologies with artificial vision systems which are, the calibration process and pose computation of robots in real-time. Initially, it aims to perform real-time camera intrinsic (3.2.1) and extrinsic (3.3) stereo camera systems calibration needed to the main goal of this work, the real-time pose (position and orientation) computation of an active coloured target with stereo vision systems.

Designed to be intuitive, easy-to-use and able to run under real-time applications, this work was developed for use either with low-cost and easy-to-acquire or more complex and high resolution stereo vision systems in order to compute all the parameters inherent to this same system such as the intrinsic values of each one of the cameras and the extrinsic matrices computation between both cameras. More oriented towards the underwater environments, which are very dynamic and computationally more complex due to its particularities such as light reflections.

The available calibration information, whether generated by this tool or loaded configurations from other tools allows, in a simplistic way, to proceed to the calibration of an environment colorspace and the detection parameters of a specific target with active visual markers (4.1.1), useful within unstructured environments. With a calibrated system and environment, it is possible to detect and compute, in real time, the pose of a target of interest. The combination of position and orientation or attitude is referred as the pose of an object.

For performance analysis and quality of the information obtained, this tools are compared with others already existent.

Key-words: Stereo Vision, Calibration, Intrinsics, Extrinsics, Attitude,

Pose, Underwater, Low-cost, Artificial vision

Resumo

A engenharia de sistemas autónomos actua em diversas vertentes. Uma delas, a visão artificial, em que este trabalho assenta, tornou-se uma das de maior interesse em multiplos contextos e focos na robótica. Assim, este trabalho procura abordar e superar algumas dificuldades encontradas aquando do desenvolvimento de tecnologias baseadas na visão artificial. Inicialmente, propõe-se a fornecer ferramentas para realizar as calibrações necessárias de intrínsecos (3.2.1) e extrínsecos (3.3) de sistemas de visão stereo em tempo real para atingir o objectivo principal, uma ferramente de cálculo da posição e orientação de um alvo activo e colorido através de sistemas de visão stereo.

Desenhadas para serem intuitvas, fáceis de utilizar e capazes de operar em tempo real, estas ferramentas foram desenvolvidas tendo em vista a sua integração quer com camaras de baixo custo e aquisição fácil como com camaras mais complexas e de maior resolução. Propõem-se a realizar a calibração dos parâmetros inerentes ao sistema de visão stereo como os intrínsecos de cada uma das camaras e as matrizes de extrínsecos que relacionam ambas as camaras. Este trabalho foi orientado para utilização em meio subaquático onde se presenciam ambientes com elevada dinâmica visual e maior complexidade computacional devido à suas particularidades como reflexões de luz e má visibilidade.

Com a informação de calibração disponível, quer gerada pelas ferramentas fornecidas, quer obtida a partir de outras, pode ser carregada para proceder a uma calibração simplista do espaço de cor e dos parametros de deteção de um alvo específico com marcadores ativos coloridos (4.1.1). Estes marcadores são úteis em ambientes não estruturados.

Para análise da performance e qualidade da informação obtida, as ferramentas de calibração e cálculo de pose (posição e orientação), serão comparadas com outras já existentes.

Key-words: Visão stereo, Calibração, Intrínsecos, Extrínsecos, Attitude,

Pose, Sub-aquático, Baixo-custo, Visão artificial

Contents

List of Figures

List of Tables

Acronyms List

BRIEF Binary Robust Independent Elementary Features

CCD Charge-Coupled Device

CFA Color Filter Array

CkB Checkerboard

CMOS Complementary Metal-Oxide-Semiconductor

CPU Central Processing Unit

DOF Degrees Of Freedom

DoG Difference of gaussians

EKF Extended Kalman Filter

FAST Features from Accelerated Segment Test

FLANN Fast Library for Approximate Nearest Neighbours

FOV Field Of View

fps frames per second

FPS Frames per second

GUI Graphical User Interface

GMM Gaussian mixture models

GNSS Global Navigation Satellite System

GPS Global Positioning System

GT Ground Truth

HD High Definition

HQ High Quality

HSV Hue-Saturation-Value

Hz Hertz

IMU Inertial Measurement Unit

IR Infra-Red

ISEP Instituto Superior de Engenharia do Porto

LDA Latent Dirichlet Allocation

LoG Laplacian of Gaussian

LQ Low Quality

LSA Autonomous Systems Laboratory

LSH Locality Sensitive Hashing

MOCAP Motion Capture

MP Mega-Pixels

OPI Object of Potential Interest

ORB Oriented FAST and Rotated BRIEF

PCA Principal Component Analysis

RAM Random Access Memory

RGB Red-Geen-Blue

RMS Root Mean Square

ROI Region Of Interest

ROS Robotic Operative System4

RT Real Time

SD Standard Definition

SIFT Scale-Invariant Feature Transform

SLAM Simultaneous Localization and Mapping

SVD Singular Value Decomposition

SURF Speeded-Up Robust Features

Chapter 1

Introduction

This work aims to solve some known issues within artificial vision systems such robots pose computation and vision systems calibration in real-time. The main goal is to achieve a reliable and precise real-time pose retrieve (position and orientation) with a stereo artificial vision system. To perform such task, the stereo system must be calibrated. Thus, additionally, this work provides some calibration tools needed when there are no previous knowledge about cameras calibration and environment characteristics.

The artificial vision, has become one of great interest on robotics. Robotic technologies can save lives by replacing humans in dangerous tasks. For example, fully autonomous and multi-environment robotic teams can be designed to cooperate in the search and rescue of people simultaneously by means such as air, land and water. Robots can also perform meticulous and routine tasks replacing humans, thus avoiding possible risks that would be incurred when performed by a human, for example, in situations of high precision or difficult human access.

Artificial vision is present on most of the referred robotic applications. Thus, this work seeks to address and overcome some real difficulties encountered in artificial vision systems, taking advantage of the technological advances in computer vision systems.

This work allows the use of low cost cameras but with high quality images to form stereo vision systems with easy and real-time calibration processes and reliable data capture to recover images depth and compute real-time object's pose, mainly in underwater environments where there are the need of knowing a robot pose and the communication with it to know its sensors data and pose can be tricky and slow to acquire due to the hindrances imposed by the water environment.

Nowadays, the calibration process of single cameras can be hard and high time consuming since, most of the times, it requires high user intervention on manual patterns

selection from captured images in order to ensure correct pattern detection and reliable calibration data. This also obliges the user to have deep knowledge about what he is going to do. So, this work overcome this issue, making life easier for the user, streamlining the calibration process, allowing the automatic fully calibration of a stereo vision system with only some parameters adjustment to offer wider environment and pattern usage like active markers. It also allows the use of customized patterns on object's real-time attitude and pose computation.

With the birth of digital cameras, it was possible for computer systems to collect information from the environment in the same way that humans would do through eye vision since cameras can reproduce visual scenes into digital images. The evolution of these vision equipments (cameras) to high resolution images, brought more detailed images but has also become increasingly heavy and difficult to use the computationally vision to assist the tasks that autonomous systems intend to perform. Fortunately, processors are now able to manipulate information and apply algorithms at higher speeds, sometimes allowing real-time or near-real-time operation of this systems.

1.1 Motivation

This work intends to allow the use of high quality (HQ) but low-cost cameras at Laboratório de Sistemas Autónomos (LSA) from Instituto Superior de Engenharia do Porto (ISEP) to determine target object's real-time attitude and pose computation. To get reliable data from a stereo vision system for real-time attitude computation, it is mandatory to make prior system calibrations. It should be as customizable and easy as possible and ROS-compliant since it is widely used on LSA applications under multiple conditions and environments. A ROS-compliant software is able to work under the Robotic Operative System (ROS) with hardware abstraction and message-passing between processes as m later.

LSA contributors are continuously developing useful technologies under several different and challenging environments and applications such as monitoring, security, environment research, human search and rescue, structures mapping/inspection, seabed mapping, prospecting and study, among others. Most of these applications are developed using vision systems integrated with other sensors. As everything, even made by precise machines, cameras are always different from each other and as presented later, this sensors carry some distortion imposed by the lens (3.2) which vary from camera to camera and need to be known. Therefore, it is necessary to calibrate the cameras before its use. This is, build the model which better relates the world objects and coordinates

with the ones mapped into the 2D image plane of the camera.

In case of a single camera, based on the pinhole model (3.2), the system will need to know only the intrinsic parameters (3.2.1) of the camera lens/sensor to remove image its distortion and its extrinsics (3.3) face to a point to represent the world objects on the camera/image(s) coordinate systems before use those images.

The developed calibration toolbox takes advantage of several image analysis algorithms. Almost instantaneously feature and shapes detection were already developed and remain valid to this application. In some cases, they stay unchanged even from its creation or with few improvements. One important step on computational vision was the possibility of obtaining images from two or more cameras simultaneously and being able to process and match points between them in order to obtain depth information from point triangulation. This allowed computational systems to have, with more or less precision, a notion of depth until then impossible with artificial vision. Even so, despite all the advances, there are only a few tools that directly make these developments (calibration and pose computation) available to the user so that they can take advantage of these same algorithms in a fast and simple, even so, a useful and precise way to overcome the difficulties of calibrating a single/stereo vision system and progress rapidly using robotic vision systems.

If more than one camera is used, to triangulate points between them and obtain depth perception, the relations from one camera to the other(s), or a selected referential, must be known or computed (its rotations and translations). This relations, known as camera extrinsic parameters (3.3) relates each camera image coordinates with a known reference coordinate system (most of the times, relative to another camera).

The process to compute the extrinsic parameters is often very complex, in standalone applications, high time-consuming and, most of the times, working only in post-processing mode with tools based or integrated on Matlab which are very expensive software for mathematical purposes.

Therefore, the presented work is Linux based but can be used on other architectures since it uses python and OpenCV graphical modules and is also ROS-Compliant. It is capable use different targets to compute, autonomously and in real-time after a few steps, all the intrinsic values/model of each camera and the extrinsic parameters/model between cameras.

With a fully calibrated vision system, the depth of the world can be recovered with some accuracy. The attitude and pose of objects can also be computed either from real-time LQ/HQ camera or from pre-recorded image/video files. The presented pose computation tool is capable to autonomously detect the target and compute its position

and orientation after a quick and simple environment/target parameters adjustment. This tool can then be used as a ground-truth system for other applications and infer robot's pose when inertial data can not be retrieved or retrieved with noisy/deviated measures.

This work was mainly designed for underwater environments and vehicles (under special and hard light conditions) although it can work in any environment due to the wide possibility of visual detection customization and easy future upgrades since it was designed to be a modular application.

1.2 Objectives

As described before, the calibration process and pose computation present known issues that slows down future works related or not with this one in vision aided systems. Thus, the essential addressed goals to achieve are:

- Simple and fast camera calibration process;
- Stereo system calibration with customizable targets;
- Target detection and recognition;
- Real-time image acquisition, feature detection and processing;
- Real-time pose computation;
- Path, cloudpoint of the 3D projected pattern, attitude and pose export to real-time ROS-topics/files;
- Application of information filters to smooth visual misdetection or imperfections.

1.3 System requirements

Besides the proposed objectives to achieve, there are parallel ideologies to make this work suitable for future improvements and upgrades. Therefore, this work seeks for:

- Modular programming to facilitate future updates and upgrades;
- Friendly Graphical User Interface (GUI);
- User maximum freedom to choose between several modules/methods to overcome some environment peculiarities;

- Wide image/video streaming acquisition compatibility;
- Reduce time spent processing data by reducing the images to Regions of Interest (ROIs);
- Environment noise filtering through unimportant pixels discard;
- Allow multiple image processing algorithms;
- Detect customizable targets (different from chequerboards);
- Customizable targets match between cameras;
- Configuration parameteres/matrices import/export;
- Reliable calibration and pose computation data output in several ways to provide wide tool compatibility;

1.4 Structure

This document is organized in eight chapters. The first one (**1**), presents the context in which this tool is thought, its motivation, the goals to achieve during this work, parallel ideologies and the advantages that this work brings to the LSA.

In the state of art chapter (2), are mentioned some similar calibration tools, some of its advantages and disadvantages and a comparison between them.

The theoretical foundations chapter (3) presents the theoretical concepts needed to support the developed work and explain the reason of some choices across this work.

The calibration toolbox chapter (4) is divided into several subjects to make an overview of the developed calibration tools, visual marker and camera holder. Firstly, it is presented the intrinsics calibration toolbox, its window menus and its output matrices/files. Secondly, the designed active marker, its structure and its peculiarities. Then, the extrinsics calibration toolbox is presented, explained its handling and instantaneous image outputs over the several configuration tabs. There are also presented the developed frame to hold the experimental cameras with the intent to stabilize the cameras relative relation with known dimensions and baseline.

The pose computation system chapter (5) similar to the previous one, presentes the attitude computation tool, its ROS and file outputs.

Then, in the project implementation chapter (6), there are presented the applied methodology and the presented/developed algorithms to achieve the calibration tasks and outputs, the stereo triangulation and the pose and attitude computation principles.

In results chapter (7), there are presented experimental setups and the obtained data.

In the conclusions and future work chapter (8), there are presented the achievements of this work and some improvements to do in future work.

Chapter 2

Related Work

This chapter exposes some existing calibration tools to single cameras and/or stereo camera systems and some of the already existent motion capture systems (MOCAP).

Stereo vision systems are formed by images of the same scene with different viewpoints/perspectives, in most cases, achieved by two or more cameras static or not with its own extrinsic parameters (position and orientation face to a referential) and can be related to each other by mean of rotation and translation values/matrices (explained on chapter 3.3.1).

Motion capture is the ability to retrieve motion and position coordinates of objects on the 3D world into the digital world, useful in many applications such as object's attitude or pose and movement studies, cinema industry in scenes/objects reconstruction, augmented reality in real-time display, etc. This process is usually a high-processing costly task due to the quantity of information to analyse in the images and hard to perform in real-time applications with more than one camera.

For a logic sequence, there will be presented the calibration toolboxes which comprise the necessary tools to compute camera intrinsic parameters/matrices and cameras extrinsic parameters/matrices. Then, motion capture tools from visual targets, ROS-compliant or not. As mentioned before, a ROS-compliant tool is able to work/interact under the Robotic Operative System (ROS) with hardware abstraction, low-level device control, message-passing between processes and clever package management [1], among several other benefits.

The toolboxes/softwares based on checkerboard (CkB) patterns use Harris corner ([56]) to detect squares intersections in images [32]. This technique may also be used to find the initial points required for the later Hough transform ([67]), the corner detector, as its response is positive for corners, almost nil for the uniform zones and is negative

for the edges. Once the corner detector has been applied, a threshold level is set for the image to obtain only the relevant corners. On all of the objects of the image, a dimensional analysis is performed, which rejects blobs that are too small or too big. From the points remaining, centres of gravity are obtained to define the checkerboard pattern.

After that, Hough Transform is performed over the information of the intersection between the lines of the image. So, having the previously obtained points, corners and edges from Harris Corner Detector, the intersections computation will be faster and focused only around those regions of interest ROIs.

Other novel techniques are presented too, based on the same theoretical foundations but for different kinds of calibration patterns. Different algorithms were developed to detect circle grids which seems to improve in the pattern detection and reprojection errors quality, leading to better calibration results but still based on the same math principles for calibration parameters computation.

2.1 ROS non-compliant tools

Matlab is a very powerful software with lots of advantages but also with some disadvantages since it is a closed-source software, it has a high cost license and sometimes with non-customizable tools/toolboxes (due to patented algorithms).

Some of the calibration methods assume unbiased observations, zero-mean independent error and identically distributed random noise in the observed image coordinates. These conditions are never met, leading to less accurate calibration results than expected.

2.1.1 Camera Calibration Toolbox (toolbox_calib) - Matlab

Toolbox_calib tool was designed by Heikkila, Jean-Yves Bouguet and O. Silven [4] and was the most known [9], widely used toolbox and the base for most of all the other existing toolboxes. Integrated on Matlab, it was for a long time very rudimentary and working only in offline mode where the user could load images containing the checkerboard calibration pattern to compute calibration parameters (post processing method). It is also not very user friendly since the user must use the Matlab console to perform the intrinsics calibration task but it gets the job pretty well done. It can also provide extrinsic calibration parameters but for a single camera at a time. Based on articles [10] to [18], it applies all those described logic and algorithms.

Figure 2.1: Toolbox_calib menu [9].

The user must capture or load all the chosen images and then, manually select the checkerboard's four boundaries for each image which makes the job high time-consuming and exhaustive for the user. After that, user has access to reprojection errors from images within a plot graph and, if he knows what is doing and understands the presented data meaning, he can exclude "bad images" and run the calibration process again, in order to improve calibration quality.

2.1.2 Camera Calibration - Matlab

The camera calibrator toolbox is based on the previous one (2.1.1) but can now be divided on two since it presents more functionalities. A single camera calibrator to model the intrinsic parameters of a single camera at a time and a stereo system camera calibrator to calibrate a multi-view/stereo system with checkerboard (CkB) on its field of view (FOV).

2.1.2.1 Single Camera Calibrator - cameraCalibrator()

This toolbox [2], based on toolbox_calib (2.1.1) refers to the calibration as being the process of estimating parameters of the camera using captured images of a specific calibration pattern. This tool is limited to detect only standard CkB patterns. It is able to compute camera intrinsics and extrinsics for cameras up to a field of view FOV of 95 degrees).

Images can be loaded or acquired directly from a online USB camera, using an extra support-package but is not yet ROS-compliant. The pattern detection is now automatic and based on OpenCV libraries [6] but not performed in real-time.

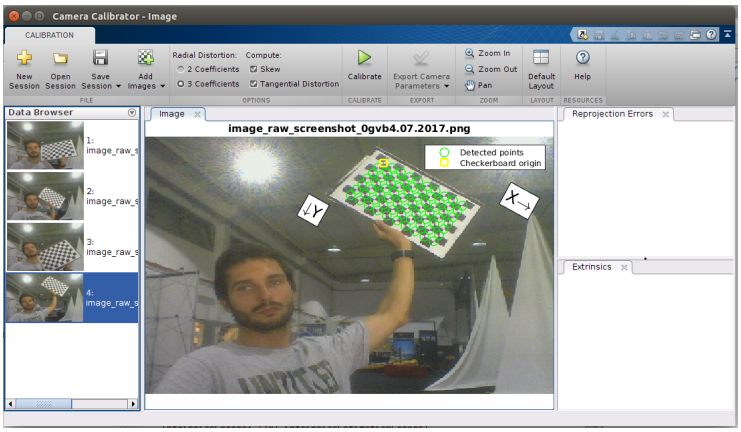


Figure 2.2: "Camera Calibration" tool pattern detection for a single camera.

The pattern detection [3] just need the CkB square size input and then all the process is done autonomously and fast, even to coloured images. It seems to not have pattern restrictions in order to distinguish horizontal from vertical axis. Then, the user can choose how many radial distortion coefficients wants to compute (2 or 3 values) and if he wants to compute skew and tangential distortion coefficients (2 values).

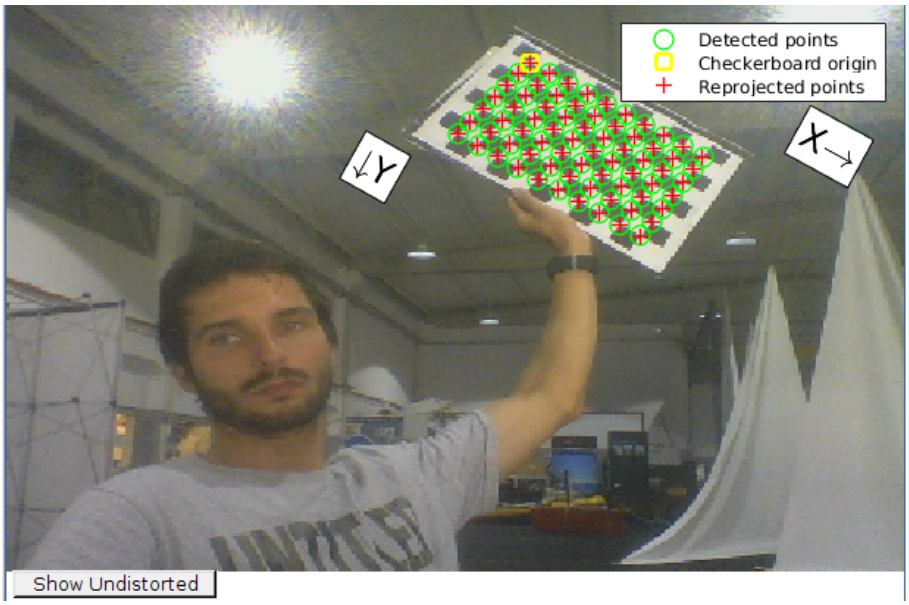


Figure 2.3: "Camera Calibration" tool with reprojected points.

Parameters computation and reprojection points on images may be a little bit slow

when a large number of images are loaded or the pattern occupies a large field of the image since it performs a sub-pixel search for features to improve corner detection on the pattern. The images are then presented with pattern origin reference, the image with or without distortion and the reprojected points on detected patterns but it seems to lose definition in comparison to the original input ones.

Graphical output results can be seen on righ-side banner tabs as presented below on figures 2.4 and 2.5.

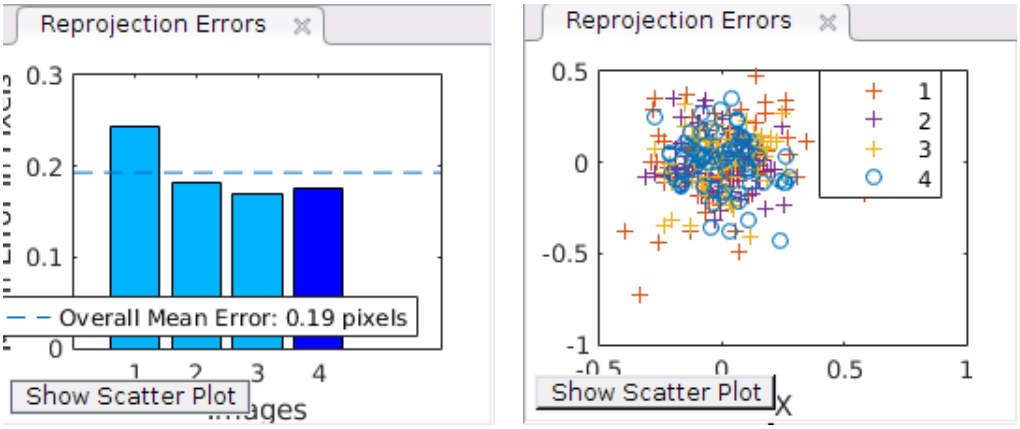


Figure 2.4: Reprojection errors display from this tool [7].

Reprojection errors from all the images are displayed by mean of:

- Bar Graph fig:2.4(a);
- Scattter Plot fig:2.4(b);

This tool also presents extrinsic parameters to the calibrated camera which are the relative position between the camera and the pattern.

Thus, single camera extrinsics can be displayed assuming two ways:

- Camera-centric view (static camera and moving patterns fig:2.5(a));
- Patter-centric view (static pattern and moving camera along fig:2.5(b));

Results can be exported directly to Matlab variables or to a script which performs all the tasks done by the graphical interface such as read all the images, compute calibration and output its calibration results to variables which allows the user to continue coding with the obtained parameters from images.

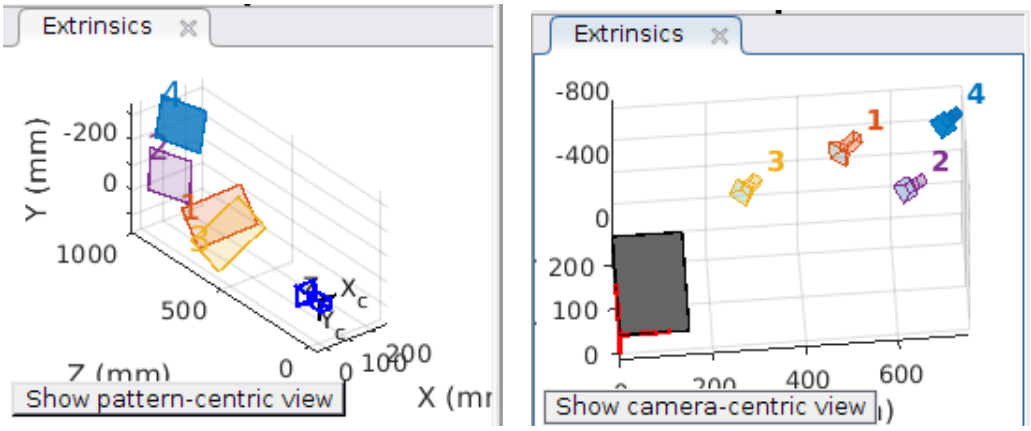


Figure 2.5: Extrinsics display from this tool [7].

2.1.2.2 Stereo Camera Calibrator - stereoCameraCalibrator()

This one [8] is a duplication of the previous tool and makes exactly the same but this time for two cameras. This time it can only be performed in offline mode, loading images from both cameras with related names/indexes (fig:2.6). This means that, the user must do a prior capture of the images in each camera at the same time or with statical patterns on its FOV and only then run the stereo/extrinsic calibration process. There are no references about the algorithms used to compute extrinsic parameters and relative position/orientation between cameras.

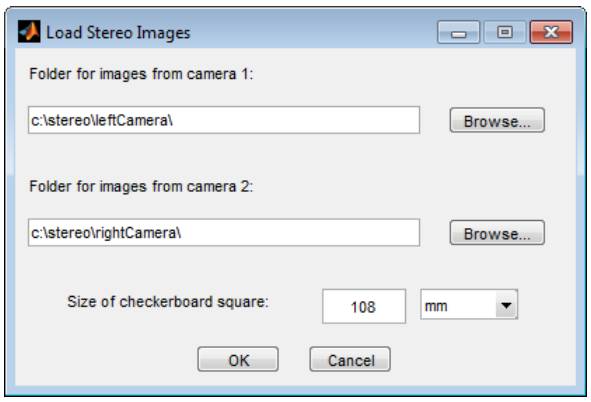


Figure 2.6: Camera Calibration tool [8].

With this tool, the camera system can be used to recover depth from images. First, this application estimates the parameters of each one of the two cameras and then computes extrinsic parameters between them, the position and orientation of "camera

2" relative to "camera 1". This tool has the advantage of allowing the user to optimize detection options (user can give initial-guess intrinsic and distortion parameters).

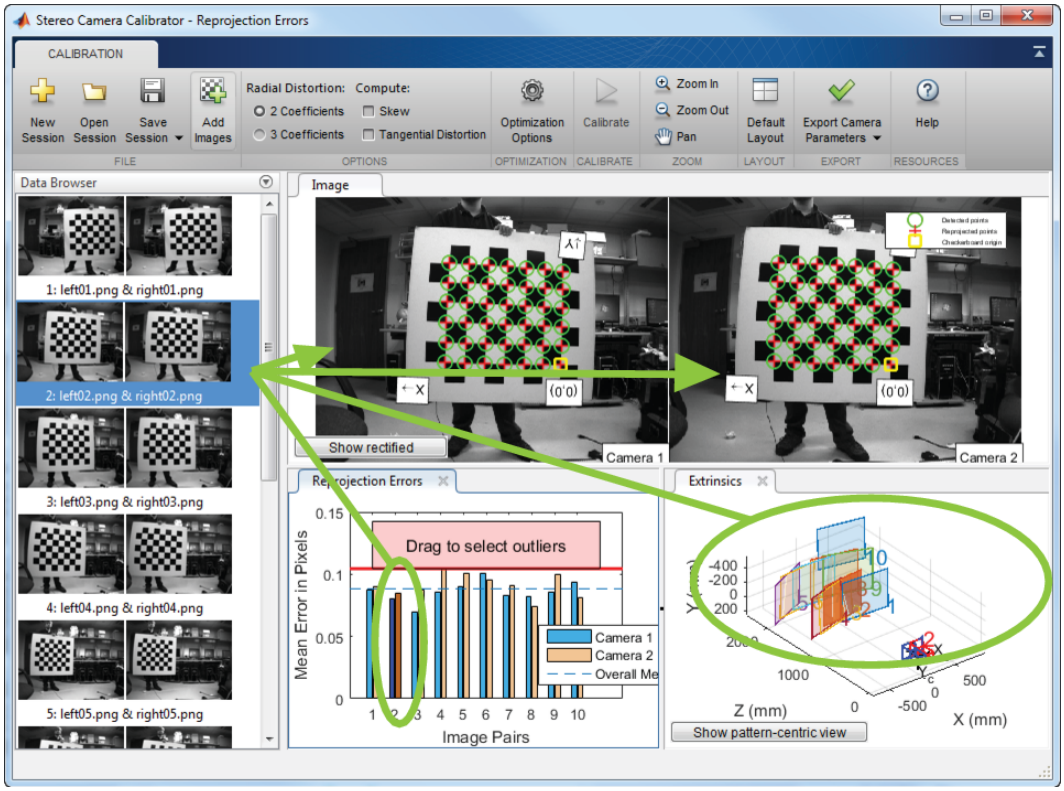


Figure 2.7: Camera Calibration tool with reprojected points [8]

2.1.3 OCamCalib: Omnidirectional Camera Calibration Toolbox - Matlab

This toolbox [20] is intended for catadioptric and fisheye cameras up to 195 degrees and, again, it was partially inspired in toolbox_calib (2.1.1) by Jean-Yves Bouguet [21].

The OcamCalib Toolbox for Matlab allows the user (also lay users) to calibrate any central omnidirectional cameras (any panoramic camera having a single effective viewpoint). Older versions of this toolbox must have high human intervention on calibration process. The calibration information capture must be done with a checkerboard on its field of view FOV and then, a manual corner points extraction must be performed by the user. Newer versions made this operation completely automatic thus, no manual extraction is needed despite still done in post-processing/offline mode.

Figure 2.8: Ocamcalib toolbox [20].

This toolbox provides two functions (CAM2WORLD and WORLD2CAM) which express the relation between a given pixel point and its projection onto the unit sphere [20]. This relation depends directly on the mirror shape and on camera intrinsic parameters. OCamCalib toolbox assumes that the camera-mirror system has a single viewpoint or camera-centric view. It is able to provide an optimal solution even when the single view point assumption is not perfectly verified (it happens when the camera optical center is not exactly in the focus of the hyperbola).

2.1.4 GML C++ Camera Calibration Toolbox

This C++ toolbox allows the use of multiple patterns/checkerboards on the same image and is a modification of the single camera calibrator (2.1.2) toolbox from MATLAB. The usage of multiple patterns makes the calibration procedure more stable and improves calibration accuracy [28], also less images for calibration are needed.

GML Camera Calibration toolbox is a free tool for camera's intrinsics and extrinsics calibration with the disadvantages of only working on windows platforms and a standard procedure with with more than 25 photos to calibrate individual cameras is needed.

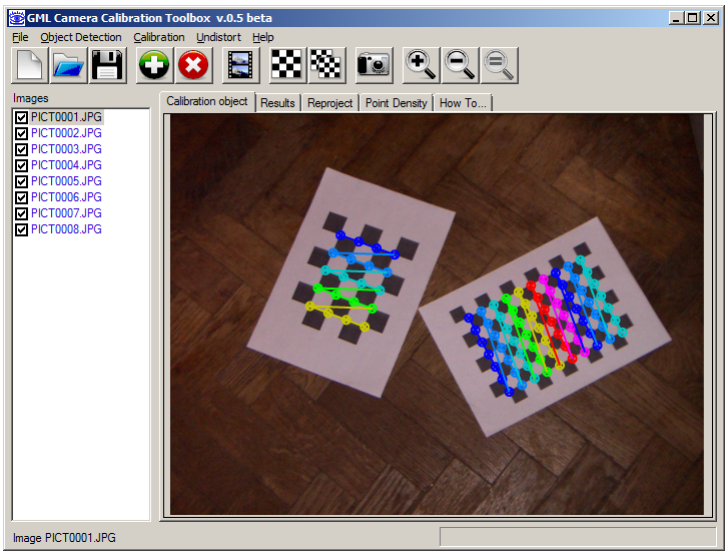


Figure 2.9: GML C++ Camera Calibration [28].

It only is able to detect checkerboards as calibration patterns. The calibration process of corners extraction is completely automatic and there are implemented several checkerboard detection algorithms to recover the relative orientation between pattern's coordinate systems.

2.1.5 camera-calib - Camera intrinsic calibration

This GUI program [30] is multiplatform (Linux and Windows). Although it runs and detects the CkB in real-time as well from loaded or pre-recorded image/video files, it is not ROS-compliant.

It alleges [30] to be able to support a wide range of image sources such as:

- All cameras supported by OpenCV (webcams, firewire,...);
- All cameras supported by FFmpeg (IP cameras,...);
- Video files (in any format);
- Rawlog files (an MRPT format of robotic datasets);
- The stereo Bumblebee camera, for calibration of each single camera in a run each;
- The intensity channel of a SwissRanger ToF 3D camera;
- The RGB intensity and IR channels of Microsoft Kinect.

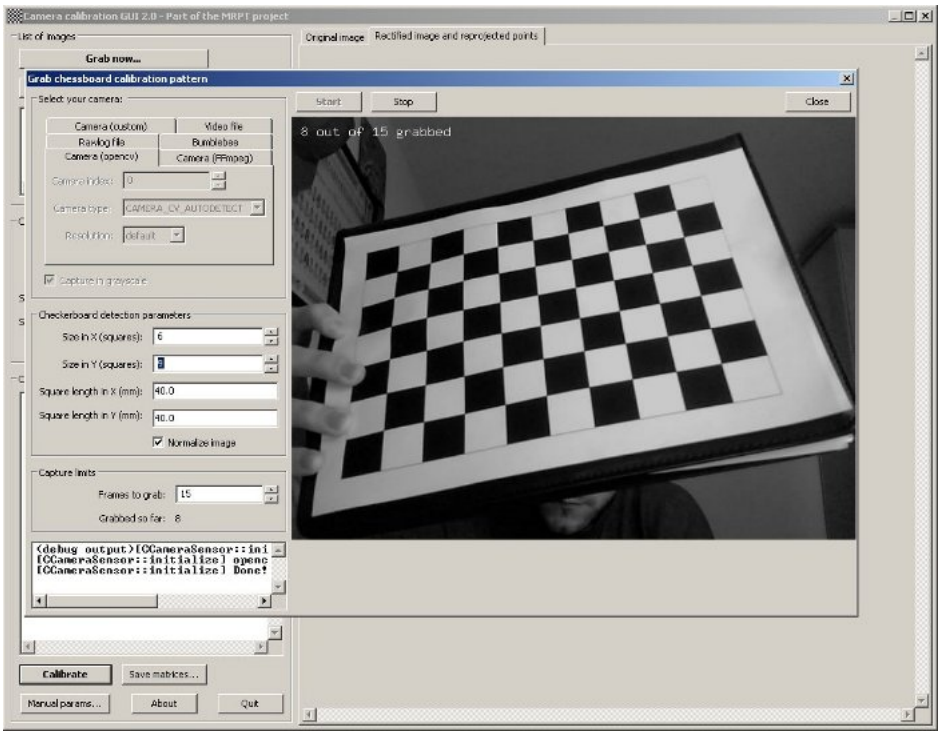


Figure 2.10: camera-calib GUI application [30].

It also shows the reprojected points on the pattern, the undistorted images and a 3D view of the reconstructed camera poses (pattern-centric view) but it seems to miss a lot of real-time detections.

2.1.6 Camera Calibration Tools

Camera Calibration Tools [31] is a closed-source Windows application which can be used to capture calibration images only from a USB cameras. It is able to detect only a checkerboard calibration pattern to compute the intrinsic and a single extrinsic camera parameters (camera centric-view).

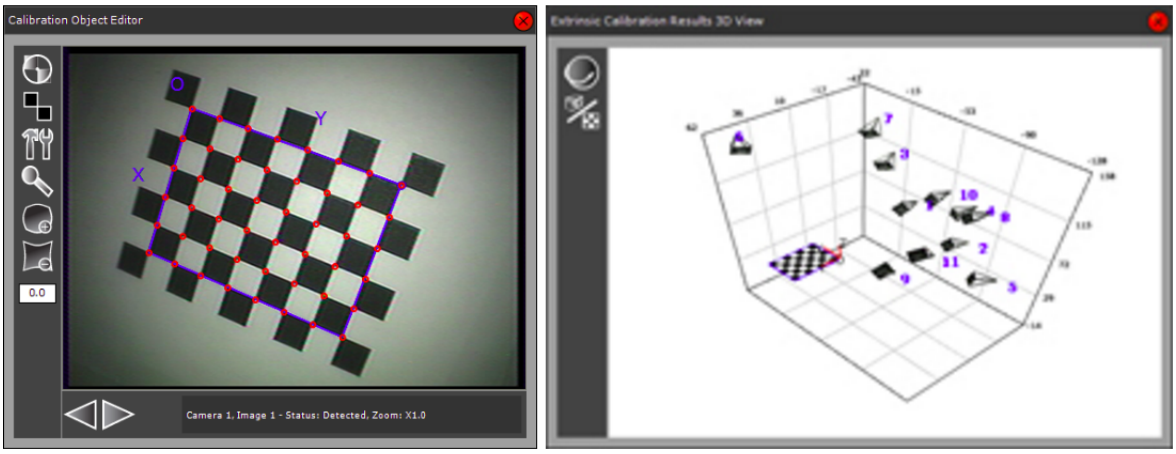


Figure 2.11: Camera calibration tools GUI application [30].

This application is largely inspired by the Matlab toolbox_calib application (2.1.1) and provides similar functionalities in a standalone application. It also includes support for USB video devices and files. New functionalities such as automatic calibration pattern detection, hand-eye (pattern-centric) calibration, camera network calibration and self-calibration are meant to develop for future releases but it seems to be stuck since 2011.

2.2 ROS compliant tools

2.2.1 Kalibr

Kalibr [22] is a ROS-compliant toolbox designed by Paul Furgale, Jérôme Maye, Jörn Rehder, Thomas Schneider and Skybotix AG at the Autonomous Systems Lab (ASL) from ETH Zurich school in Switzerland.

It proposes to solve following calibration problems:

- Multiple camera calibration: intrinsic (3.2.1) and extrinsic (3.3) calibration of camera-systems [24];
- Camera-IMU calibration: spatial and temporal calibration of an IMU and camera-system [25] [26];
- Rolling Shutter Camera calibration: full intrinsic calibration (projection, distortion and shutter parameters) of rolling shutter cameras [27].

Figure 2.12: Kalibr Toolbox from ETHZ-ASL [22].

This toolbox can't analyse and detect patterns from images under real-time constraints so, it only works in offline mode with loaded image data files. It is also recommended to lower the frame rate of the cameras to around 4Hz in order to reduce redundant information in images leading to lower runtime of the calibration. The image data can be provided only as a ROS bag file containing the image streams from all cameras. To get a good estimate of the system parameters, the calibration routine goes through all images and pick images based on information theoretic measures. A single calibration process can combine different projection and distortion models. It assumes that the cameras are static and the calibration pattern is moved in the FOV of all the cameras.

Summing up, this seems to be a good toolbox for camera systems calibration but it stays limited to ROS-compliant applications and to offline/post-processing data analysis with low capture frame rates.

The calibration output is very friendly and make the calibration integration easier to user's posterior projects [23]:

- Report in PDF format. Contains all plots for documentation;
- Result summary as a text file;
- Results in YAML format. This file can be used as an input for the camera-imu calibrator.

2.2.2 camera_calibration

This is a ROS-compliant package/tool [19] to calibrate single (narrow/wide) or stereo cameras using a CkB but it seems to be very restrict on user manipulation during calibration.

The sync between cameras should be done before the input on the toolbox or be triggered to capture images simultaneously to ensure the correct operation and results but recent updates made it able to calibrate stereo vision pairs that are not exactly synchronized. The perfect trigger synchronization is hard even when performed directly

by hardware since there will always exist tiny differences between camera's acquisition timestamps because the trigger signal can't be done simultaneously and camera acquisition times may differ. So, the application can now get a input value for timestamp approximation between image pairs.

2.2.2.1 Dual Checkerboards for Narrow Stereo Camera

With this tool, the user will be able to use large **and** small checkerboards at the same time to calibrate a stereo camera system. The large checkerboard must be used at a range of about 4 meters and the small one at ranges of 0,7 to 1,5 meters.

Figure 2.13: ROS Camera Calibration Package [19].

It has the restriction to hold the small CkB horizontally, with no tilting during calibration. Patterns must have different dimensions so the system can tell them apart. No images are displayed until the program detects the CkBs successfully.

After initiated the program, the user must move and tilt the movable CkB around the camera field of view (FOV) to ensure a good calibration.

User can see real-time or near-realtime (depending on camera resolution) the pattern and corner detection represented over the real image to ensure the correct detection on camera calibration process. Then, automatically and in a aleatory way, images are taken to detect corners and use it to posterior intrinsics computation. Meanwhile, X, Y and Size quality side-bars are displayed to get the user to know his progress.

Figure 2.14: ROS Camera Calibration pan and tilt movements [19].

If the checkerboard is successfully detected, it can give the user good axis and distortion calibrations [19] depending on:

- Left and right edges of the field of view, user is improving X axis calibration;
- Top and bottom edges of the field of view, user is improving Y axis calibration;
- Detected at various angles to the camera, user is improving "Skew";
- The checkerboard fills the entire field of view , user is improving "Size calibration";
- The checkerboard tilted to the left, right, top and bottom (X, Y, and Size calibration)

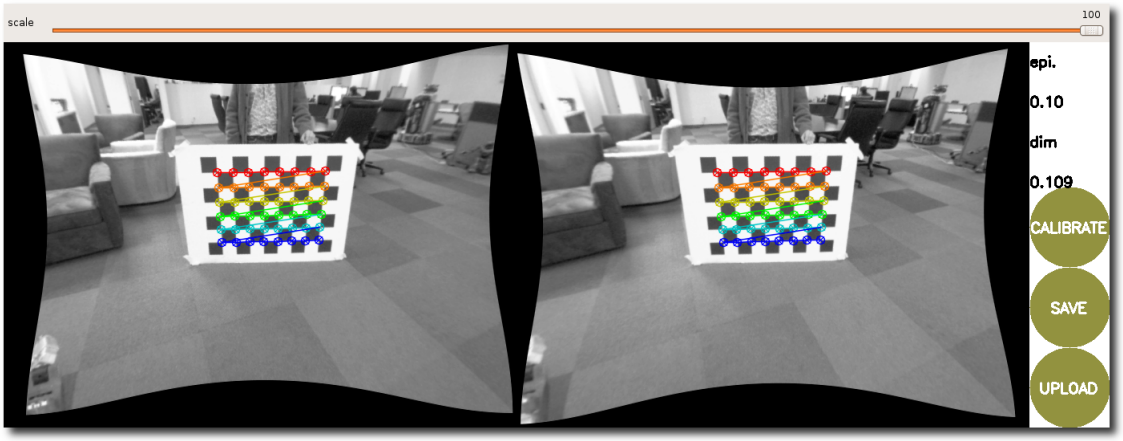


Figure 2.15: ROS camera calibration corrected images [19].

Only when all the right-side banner is highlighted, the system has the enough data to compute intrinsic values of each camera and the needed information to compute the relation between cameras. The user is then authorized to proceed and save the calibration output, yet with no access to reprojection error values.

2.2.3 industrial_extrinsic_cal

This toolbox/package [29] is still in development but already computes the camera intrinsic parameters using a rail guide, based on a calibration target with a circle grid.

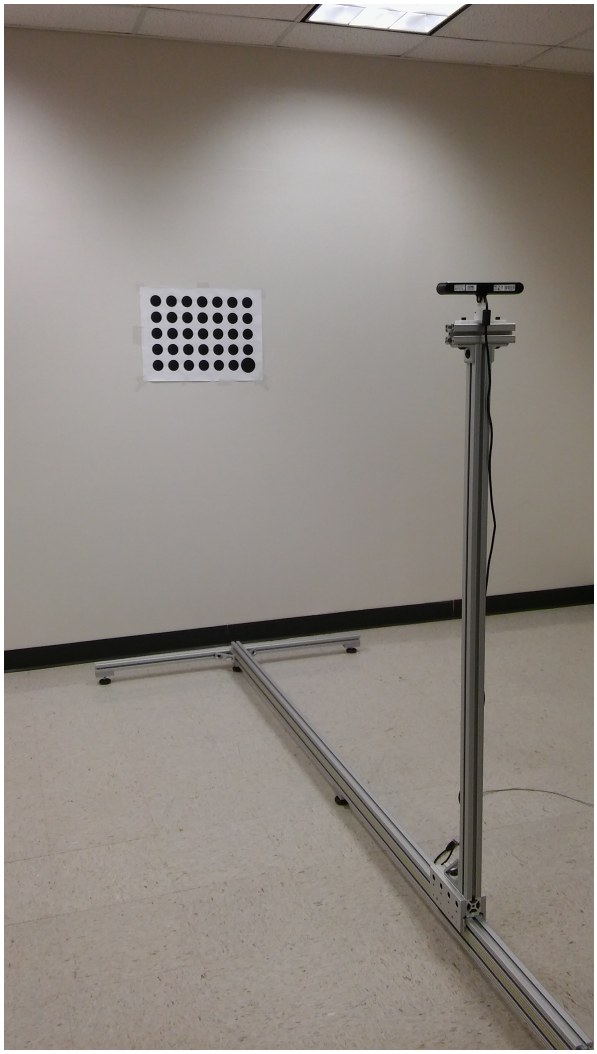


Figure 2.16: ROS Camera Calibration with circles grid pattern [29].

Even there are already existent toolboxes, for some industrial and high precision applications, the results may not be enough. Depending on the number and quality of the images collected by the user, the calibration method can result in widely varying focal length values, therefore this calibration tool uses a different kind of calibration procedure and distinct pattern with a circle grid where its centers are placed similar to the edges/corners detected on a standard CkB.

Its developers allege that, when performed precisely, this routine is both quicker (due to fewer images needed) and more accurate because the parameters have lower co-variance but it relies on knowing the distance that the camera is moved between successive images, being fully dependent on human measurement accuracy inputs leading to variable calibration outputs and sometimes to high time-consuming processes to achieve good calibration parameters.

2.3 Toolboxes comparison

The above calibration tools were presented starting from the most rudimentary to the most complex and complete ones. It is clear that the automatic pattern detection, which is not present on "toolbox_calib" toolbox, became a crucial goal to streamline the calibration process and so on a standard feature on this tools (present on all the other subsequent tools).

The table 2.1 contains synthesized information about all the presented tools/applications in order to make a easier overview of the available calibration toolboxes. Although it could sometimes be a discriminating factor, the operating system of each of the tools is not taken into account for this synthesized comparison where we emphasize its the specifications and versatility.

Table 2.1: Toolboxes/applications comparison.

Calibration Tool	**Real time (RT)**	**ROS compl.**	**Single cam calib.**	**Stereo cam system calib.**	**Calib. Patterns**	**Outputs**
toolbox_calib	No	No	Yes	Yes	CkB	Camera intrinsic and extrinsics matrices.Calibration errors.
cameraCalibrator	No	Not yet	Yes	No	CkB	Reprojection errors, camera intrinsics graphs and matrices.
stereo CameraCalibrator	No	Not yet	No	Yes	CkB	Each camera reprojection errors, extrinsics graphs and matrices.
OCamCalib	No	No	Yes	No	CkB	Camera intrinsic and extrisics matrices.
GML C++	No	No	Yes	No	Multiple CkB	Camera intrinsics and extrinsics matrices for each pattern detection.
camera-calib	Yes	No	Yes	No	CkB	Reprojection of corners, undistorted images, camera poses reconstruction and intrinsic matrices.
Camera Calibration Tools	Not yet	No	Yes	No	CkB	Camera poses reconstruction and camera intrinsic and extrinic matrices.
Kalibr	No	Yes	Yes	Yes	CkB	Camera intrinsic and extrinic matrices in PDF, txt and YAML files.
camera_calibration	Yes (LQ)	Yes	Yes (RT)	Yes (RT)	Multiple CkB	Camera intrinsic and extrinic matrices
industrial extrinsic_cal	No	No	Yes	No	Circle Grid	Camera intrinsics

2.4 Motion capture systems (MOCAP)

MOCAP technologies, originally used in sports and biomechanics, became successfully employed in a wide variety of sectors like medicine, animation and cinematographic industry, advertising, gaming development industry and education. This systems are usually expensive but very profitable since it can streamline the movement animation or study, creating more lifelike movements than manual animation or giving more precise movement flows in order to study animal kinematics for example.

Object's or body's movement can be retrieved through the placement of markers/sensors in a way to reproduce reliably it's shape changing, usually placed on or near each body joint. Each joint movements are translated in positions or rotated in angles relative to each other or to a global coordinate system.

These systems are continuously capturing joint velocities, accelerations, impulses and/or relative angles, providing an accurate digital representation of the movement.

Most of the existent applications, similarly to this work, were developed to work

with visual markers, therefore with visual camera systems with more than one camera. Although it is possible to achieve precision measurements (with resolution below millimetre), some of this kinds of MOCAP systems can be computationally high-cost because of the size and quantity of information to analyse in the captured images, making it a hard task for applications under real-time constraints with more than one camera. It is always mandatory the use of more than one camera with different view-points to recover depth from images but it has some other known issues in addition to those already referred. Most of the times, two cameras are not enough due to markers occlusions which lead the tracking algorithms to lose the joints continuous pose and mismatching once recovered.

Therefore, other new IMU-based or markerless approaches can now be found overcoming some of the visual problems related with the quantity of information to process but they are not perfect too. IMU-based systems can have direct access to joint's relative coordinates, velocities, accelerations and angles, therefore providing faster access to the kinematic information but its information quality depends directly on the sensor information quality and its deviation errors. Bad sensor's characteristics such as high noisy measures, low sensitivity, bias deviation, high bias instability and low response frequency or high sensor latency can lead to time-drifted/deviated measures from the reality if there are no implemented filters to modulate the sensor behaviour (which may also have high processing costs). This kind of systems also have the disadvantage of needing a sensor for each joint in contrary to visual systems where a single stereo camera system could detect several joints in the 3D space, making it costly (the more sensors needed, the more expensive it will be). It can also be impracticable in some cases with stationary objects/scenes where the "random walk" noise (random noise) is highly felt, leading to wrong and drifted measures of real static objects since the error in these kind of sensor is cumulative, which does not happen in the vision systems.

The combination of position and orientation/attitude is known as pose of an object, even though this concept may be used to refer the orientation meaning only when the position zero or not measured.

With each joint pose information, softwares like Motion Builder from AutoDesk [102] can be used to directly render that information from the cameras, transposing the observed motion to the animation/digital world and easily modelling body's behaviour.

2.4.1 Mechanical-based (markerless) MOCAPs

Mechanical-based MOCAP systems use physical measures to capture motion. They measure the amount of motion from specific sensor that measure for example mechanical

encoder angles, magnetic fiels or inertial joint values like velocities, accelerations and orientations.

The encoder ones like Animazoo Gypsys 7 from METAmotion [83] (figure 2.17(a)) may cost a few hundred euros but its life-time is reduced due to sensor's detrition effects along the time. The precision and price of this systems depends directly of the encoders resolution.

Magnetic field ones like MotionStar wireless from Ascensio-Tech [85] (figure 2.17(b)) may also be within some hundred euros but its measures are easily uncalibrated and external noisy sensible. The precision and price of this systems depends directly of the field sensors resolution/calibration and noise isolation.

Figure 2.17: Motion capture with mechanical encoders [84] and magnetic field sensors [86].

IMU-based stands for inertial measurement units, that is, a physical "sensor" capable to measure accelerations and angular velocities. Usually, this units are the union of several sensors and have long life-time because they almost do not suffer of detrition effects. IMU-based systems (figure 2.18) to capture motion, or compute joints pose, are endowed of inertial sensors such as gyroscopes/inclinometers, accelerometers and magnetometers/compasses which directly retrieve the joint's angular velocity, acceleration and orientation respectively.

There are some known IMU-based tools/products to capture sensor's information but most of them in closed-source architectures:

- Pedalvatar [71];
- LPMOCAP [72];
- MVN BIOMECH from XSENS [73].

The price of this kind of systems depends directly on the price of the inertial sensors and its quantity. Just to have an idea, inertial measurement units or individual inertial sensors for this purposes can vary from a few euros to thousands of euros each, depending on it's measurement data quality.

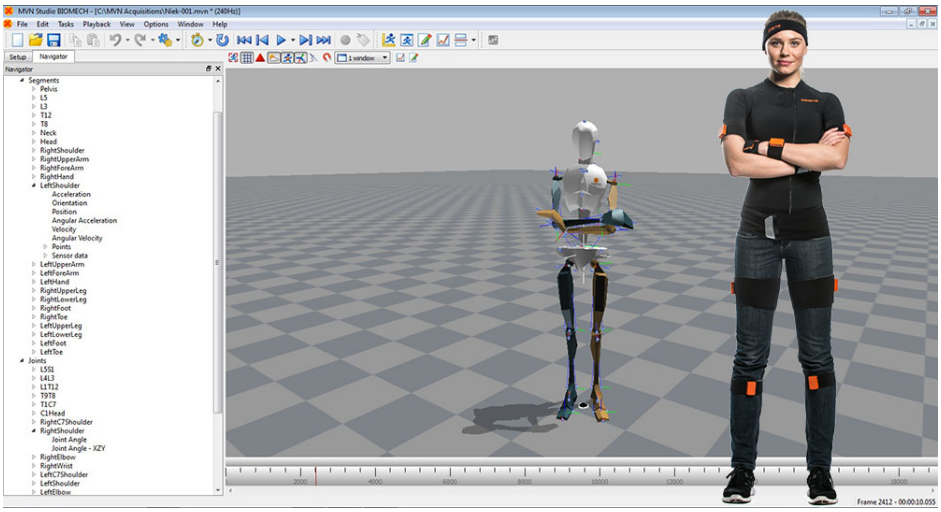


Figure 2.18: IMU-based MVN-BIOMECH system from XSENS [74].

The quantity of information captured with this sensors is reduced to a few bytes for each inertial measurement unit since there are only velocities, accelerations and orientations to recover. This makes this kind of sensor's information easy to handle under real-time constraints even at high acquisition rates.

2.4.2 Vision-based MOCAPs

Under the vision-based MOCAP systems, there are three kinds of motion capture technologies: Infra-Red (IR) aided, IR-vision-only and the RGB-vision-only systems.

As well as this work, vision-based kind sensors usually have the disadvantage of large quantities of information to process, depending directly on camera's resolution sensor, since each image can easily reach millions of pixels and several Mega-bytes size.

Infra-Red (IR) aided 2.4.2.1 approaches, are the combination of a single standard vision RGB camera with infra-red (IR) sensors to capture depth information from structured IR light, then merged with the vision sensor. This kind of systems usually doesn't prime for accuracy, making them only suitable for indoor, closer and low-precision applications like motion gesture extraction where the precise dimensions or amount of movement is not quantified or important but only observed object's shape.

IR-vision-only systems with two or more IR cameras like VICON and Qualisys 2.4.2.2.1, are meant to be used indoor with cameras endowed of active infra-red light emitters and markers made of retro-reflective materials (figure 2.19) since this camera sensors capture only information from specific materials which reflect the IR light straight-back to its emitter. This technology reduces the amount of information to process to only marker's x and y coordinates on image and marker's radius, usually within a few bytes size. Although the information is reduced to a few bytes, real-time applications are only possible with low resolution cameras or dedicated processing units to each camera. The markers radius may vary but it must be unique on a single scene. Different marker's sizes would lead to wrong measures and undistinguishable shapes since a big and distant marker could be observed with the same radius of a small and closer one. Although the depth information can not be recover with a single camera (of an unstructured environment), knowing the real size of a marker and its observed size will result in markers differentiation. This systems also have limited accuracy distance range due to IR emitted lights low range or noisy susceptibility in outdoor scenes if the light is not structured.

Figure 2.19: Retro-reflective markers [78].

Much more cheaper than all the presented tools are the RGB-vision-only motion capture systems. System price can be lower, depending directly on the amount of cameras to use and the quality of the cameras (it's specifications like frame rate acquisition, resolution, external trigger, etc). This kind of systems are known to be high computationally demanding since each camera's coloured frame is often over some Megabytes, leading to longer processing times, the higher the image resolution. Once again, similarly to the IR

based approaches, the depth information of unstructured scenes can only be recovered by two or more cameras but, although coloured images can be heavy to process, this brings lots of advantages. Scene features can be filtered to the desired color objects and apply specific algorithms under real-time constraints until then possible in the whole coloured image spectrum. The color presence makes it possible to use less cameras than IR systems since markers may have distinguishable colors and sizes, overcoming some partial vision occlusion problems. This systems can also be markerless and track specific features of images like fingers or faces but it obliges the whole image search for features to track and match on the subsequent frames.

Figure 2.20: Face tracking marker and markerless [79].

2.4.2.1 IR-aided (markerless) MOCAPs

Infra-Red (IR-aided) systems like Intel SR300 (67€), "Kinect" (99€) on figure 2.21(a), developed by Microsoft and Prime Sense have or Senz3D (269€) from Creative have, at least, one standard RGB camera and one IR-structured pair (emitter and receiver/sensor) to add depth perception to the camera image information. IR-structured emitters project fased IR light rays on the scene and sense only the backscattered rays. This kind of systems are able to track markerless objects under real-time constraints but its precision tends to be weak for distant objects or in high noisy scenes due to undesired reflections or false positive measures.

Figure 2.21: Kinect sensor [75] from Microsoft and Senz3D from creative [80].

There are also some laser-aided systems with high precision measures but are usually very expensive and computationally heavy due to high information quantities. Powerful and more precise ones can also work with harmful rays to human vision which make them impracticable to use on most of the human animated tasks.

2.4.2.2 IR-vision-only (with retro-reflective visual markers) MOCAPs

As referred before, this systems use marker's visual information to observe object's movements. Due to object's markers occlusion issues, most of the systems use more than two cameras to prevent angle occlusion situations and faster tracking loss recover. IR-vision cameras can not recover depth information unless, there are more than one camera or point of view to perform stereo triangulation. Usually, each camera has its own processor to handle with the image information and outputs just the x and y markers coordinates and radius at impressive frame rates and precision under a millimetre but it also leads to very expensive hardware setups. In addition, all the camera's information must be correlated and merged in a separate processing unit or by expensive closed-source softwares to ensure optimal tracking accuracy and solve lost tracked markers mismatching.

2.4.2.2.1 VICON and QUALISYS (Products)

The famous MOCAP VICON or QUALISYS systems, use only IR sensible cameras with active infra-red light emitters synchronized with the IR image capture. Usually they are applied on green or white screen scenes to avoid noise measures. Therefore, IR-vision cameras can filter or be calibrated to detect only the specific retro-reflective materials like ball markers or strips, leading to quicker processing algorithms employed over clean infra-red images. Each camera observes only the marker's coordinates and radius, leading to low data latency and reliable tracking output. On the other hand,

this kind of sensors have limited accuracy range, usually optimized for a known short distances range of some meters from each camera due to the emitted IR short range.

VICON cameras have resolutions between 1.3 MP at 250 FPS with 3.4ms latency and 16 MP at 120 FPS with 8.3ms latency with starting prices from 2300£each. QUALISYS and VICON cameras are only suitable to work in indoor applications under stable light conditions and with no noise interferences from the environment, most of the times, caused by sun rays scattered over wide wavelength range.

There is no available exact prices information or any other functional information about VICON or QUALISYS cameras and softwares, unless contacted directly by company sellers.

2.4.2.2.2 OptiTrack (Product)

Similar to VICON and Qualisys, Optitrack is also one of the solutions within this kind systems endowed of IR-vision cameras able to capture retro-reflective markers. Its price varies between 279$ for a 0.3 mega-pixels (MP) camera at 120 frames per second (FPS) and 8.3 millisecond latency able to capture 1.58cm markers up to 11 meters, to the most expensive one of 5999$ for a 4.1 MP camera at 120 FPS and 5.5 milliseconds latency, able to capture 1.58cm markers up to 30 meters (figure 2.22).

Figure 2.22: Optitrack IR-vision camera sensor [77].

In addition, this system will need a software to process camera's information with prices between 999$ to 2999$, plus annual maintenance costs from 499$ to 1499$, depending on the number of cameras to handle and software available features.

2.4.3 RGB-vision-only MOCAPs

The present work falls in the category of RGB-vision-only MOCAPs. RGB-vision systems can track visual coloured features easily with no time spent attaching markers and avoid potential failure due to misplacement of markers. They are no range limited since standard cameras can accurately track distant objects almost imperceptible for human eye (with proper lens). However, the depth resolution decreases with the distance and the information to process is usually enormous.

Systems like Simi Shape 3D from SIMI [81] and AMI from HOLONEER [82] were developed under this architecture to capture motion from cameras with relative low resolutions. High resolution cameras would lead to delays on image analysis.

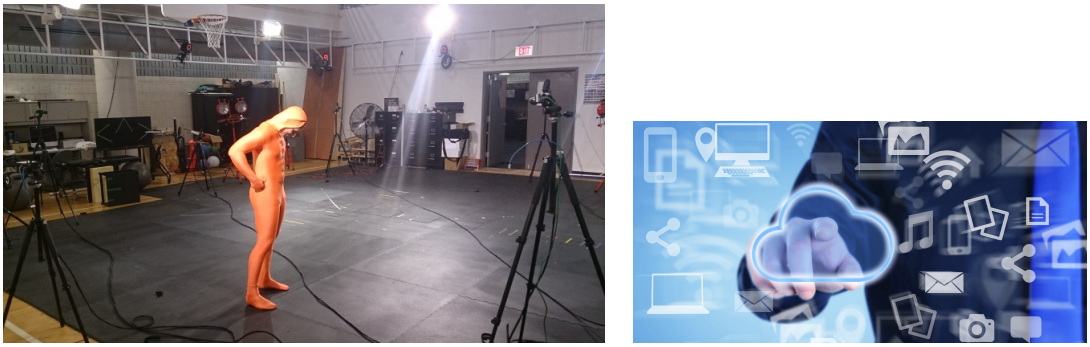


Figure 2.23: Simi 3D Shape [81] from SIMI and Ami product from HOLONEER [82].

Simi product requires special suites like the one on figure 2.23(a) and green screen scenes to allow high performance detections and accuracy. Alternatively, ami works under every scene but is mainly used for hand and finger tracking in order to replace computer mouse functionalities.

When recording multiple high-performance cameras, computers are faced with enormous quantities of video data, up to several gigabytes per second [81]. Therefore, cameras usually capture from 0.3 to 2.2 mega-pixels images with frame rates from 30 to a maximum of 500 FPS for the low resolution ones. Additionally, this systems must have high speed communication rates and have powerful processing units dedicated only to the image processing of each individual camera. Thus, information can be processed and merged in software to compute image depth and study movement flow accurately.

2.5 Motion capture systems comparison

The mechanical-based systems have the advantages of being cheaper and precise but with spoil and uncalibration issues. Although its amount of information is reduced to angles measurements, its integration may also be non-comfortable, obstruct/interfere with movement and not easily customizable/scalable.

IR vision-aided systems are relatively cheaper and work well, which makes them suitable for reduced-budget architectures. Their measurement's quality is not as reliable for accurate applications such as ground-truth (concept explained on chapter 3.7) where its precision should be millimetric. The information capture and process is performed by an host computer, which leads to additional computational costs since it also has to merge the information of the RGB cameras with the information from the IR sensor and, only then, integrated on further applications like position computation or object's attitude.

Table 2.2: MOCAP systems comparison

MOCAP	**Accuracy**	**Setup mount and calibration**	**Real-time processing**	**System scalability**	**Cost**
Mechanical based	High	Fast but not comfortable and not customizable	Yes	Yes	Depends directly of mechanic sensors quality
IR-aided (markerless)	The worst (above several millimetre)	Plug-in and play	Yes	No	Less then a hundred euros
IR-vision (with markers)	The best (under millimetre)	Time-consuming and regular calibrations	Yes	Yes	Over hundreds of thousand euros (exponential with number of cameras)
RGB-vision (markerless)	Middle (a few millimetres)	Time-consuming and regular calibrations	Yes (mostly in LQ cameras)	Yes	Over dozens euros (exponential with number of cameras)

On the other hand, despite the information minimization from IR-vision-only systems which allows its use on high performance and precision applications, its cost blows off most of the regular budgets since each camera can reach several hundreds of euros. The outcome quality of this systems depends directly of the camera resolution and frame rate, therefore its cost. The use of markers makes the motion capture job easier since the detection is done over images with visible retro-reflective makers only. Its noise effects are almost null which save lots of processing-time on detection task. Although there is no significant complains about it, the correct marker's position on robots can influence directly the precision of the movement.

RGB-vision-only MOCAP system's are on enormous disadvantage when referring to data processing times but this is much more affordable for tight budgets than the other

options. RGB-vision cameras offer wide possibilities for algorithms employment due to color presence in images and the depth recovery is based on the same stereo principles of IR-vision-only systems, therefore the same accuracy when comparing equivalent cameras. Visual markers can be used with this kind of systems to get great results but this RGB cameras also allow the accurate tracking over markerless objects.

2.6 Contributions from this work

This work aims to give the user a real-time motion capture tool and calibration toolboxes with a ROS-compliant architecture, using not only a checkerboard but also customized targets with active coloured markers.

The calibration toolboxes are often high time consuming and hard to handle. Thus, the developed tools try to overcome some of this issues and allow the user to calibrate intrinsic and extrinsic camera parameters in real-time, outputting the detection of the target in real-time and the calibration result to files and ROS topics.

Chapter 3

Theoretical foundations

3.1 Camera Image Sensor

World as we see it, is full of details and objects which are, to our eyes, no less than the light or photons reflected by them in into our eye which, in turn, transforms all that light into an image to our brain. So, similar to other technologies developed based on living beings morphologies, the human being was capable of reproduce the way as we see the world in electronic systems.

First of all, it is necessary to capture photons/light with photo-sensible electronic sensors (most common are CMOS and CCD sensors) [33]. These sensors have a matrix full of tinny individual photo-sensible electronic cells called as "Picture Elements", more known as pixels which detect quantity of received light.

When image capture is triggered, the sensor is uncovered and stays exposed/light-receptive just for the necessary amount of time to collect the world incoming light. Usually, is used a very short period of time to minimize scene object movement effects and capture them as if they were static. Each one of these pixels detects a single color and the more photons it receives, the more intense will be the color. Then, when exposure ends, each pixel is individually quantified and stored as a numerical value to form a image file/array.

The basic pixel elements are only capable of collecting amount of light and not color information. In order to obtain this, a set of coloured filters are usually used to allow one pixel to be sensitive to one color or wavelength only. The complete color spectrum is obtained by neighbouring pixels interpolation techniques.

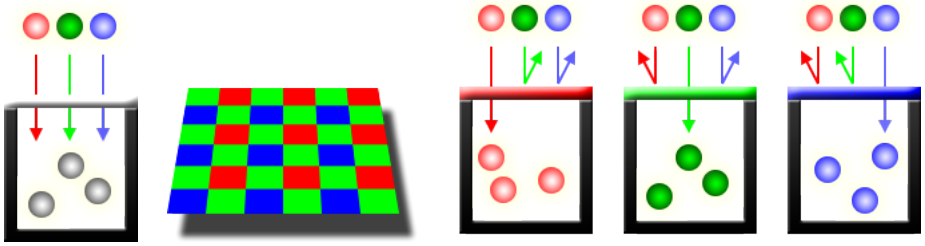


Figure 3.1: Monochromatic Photon-sensible sensors with no filters and then color filtered [87].

These sensors have now a Color Filer Array (CFA). So, the principles stay the same from the monochromatic images but now, each pixel has its individual filter to receive only the wanted photons/light within a wavelength/color (red, green or blue).

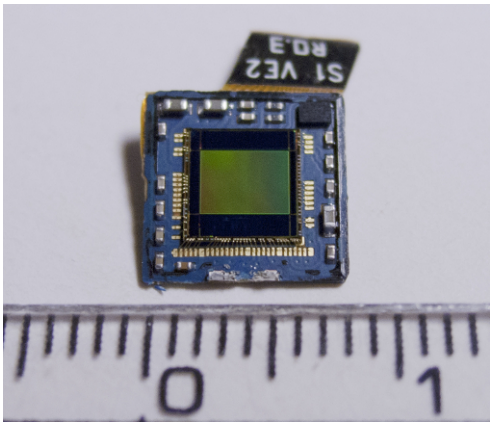


Figure 3.2: Camera Sensor (CCD) [88].

3.1.0.1 Color interpolation - Bayer Pattern Interpolation

These image sensor has a grid of red, green and blue pixels arranged so that the first row is "RGRGRGRG" (Red-Geen-Red-Green-...), the next one is "GBGBGBGB" (Geen-Blue-Green-Blue-...), and that sequence is repeated in subsequent rows as seen in figure 3.1 and known as Bayer pattern. This one is known as the most common CFA.

For every channel, missing pixels (pixels from other channels) are obtained by interpolation in the demosaicing process to build up the complete image. This is, the real color of each pixel shouldn't be its own single color but a mix of all RGB channels to obtain real object colors. So, it estimates missing information of pixel using informa-

tion from surrounding pixels. In other words, color interpolation or color demosaicing algorithms are the restoration of image color from image detectors based on CFAs [41].

3.1.1 Camera Models

As images are 2D projections of real world scenes, the information we can get from it can be geometric (positions, points, lines, etc) or photometric (intensity and colour). So, it is necessary to understand how the image is formed.

A specific camera model will be presented, based on perspective projection, the "Pinhole Camera Model".It is the most widely used and simple but there are other camera models [36] based on:

- Orthographic projection;
- Scaled orthographic projection;
- Para-perspective projection;

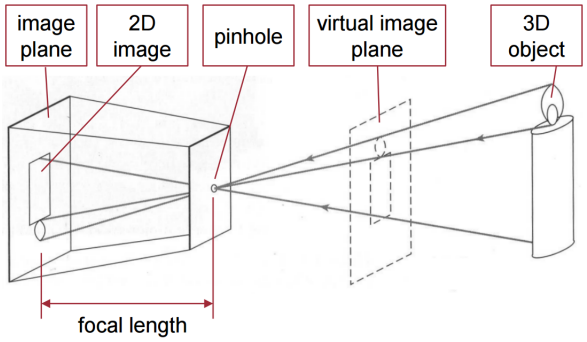


Figure 3.3: Camera pinhole model [36].

The camera model describes how a 3D point in the world is mapped onto the image plane in a 2D point. The pinhole model principles can be explained by the figure 3.3.

3.2 Pinhole camera model

The mapping of 3D points onto the image/sensor plane can be modelled by the intrinsic parameters of the camera. The camera meaning refers to the set of image sensor and lens. Later it will be presented the intrinsic calibration parameters such as:

- Focal length (f_x, f_y) - the distance between the optical centre of the pinhole to the image plane/sensor (assuming there are no lens);

- Optical Center (c_x, c_y) - The center of the perspective projection or optical center (the point in which all the light rays intersect). In this case it's the pinhole (assuming there are no lens) and the center of image may not be aligned, resulting in x and y pixel offset values;

- Skew coefficient (s)- The skew coefficient defining the angle between the x and y pixel axes. Skew coefficient is non-zero if the image axes are not perpendicular.

- Aspect Ratio (η) - The aspect ratio of an image describes the proportional relationship between its width and its height;

3.2.1 Intrinsic Sensor Calibration Parameters

Light reflected from points in the 3D world travel along a single straight path through the pinhole onto the view plane so, the object is imaged upside-down on the image plane [37]. The optical center, center of projection of the sensor image or pinhole location (this last notation will be upgraded later with the introduction of a lens) may not be aligned with the center of the image so, its offset (c_x, c_y) has to be measured in image coordinates (pixels). It can be easily obtained knowing objects dimensions/relations from the world onto the image.

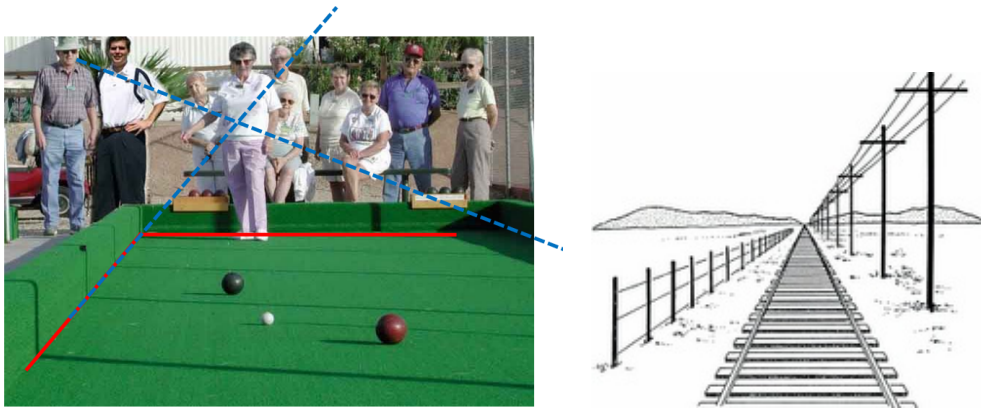


Figure 3.4: Depth perception loss [89] and intersections of parallel lines [90].

By default, right-handed coordinate system is used on 2D image where the Z-axis is normal to its plane with positive values from the camera to the scene direction.

Meanwhile the light travels through the pinhole, all the perception of depth was lost because it is geometrically projected onto a 2D plane image and, with it, also lost the perception of surface angles/lines. This means that straight parallel lines in real

world like trains rails, may seem to intersect and perpendicular lines seems no longer perpendicular on 2D plane [38] although, still straight lines. The figure 3.4(a) presents all the concepts referred above. The points in images where the straight parallel lines seems to intersect are called as vanishing points.

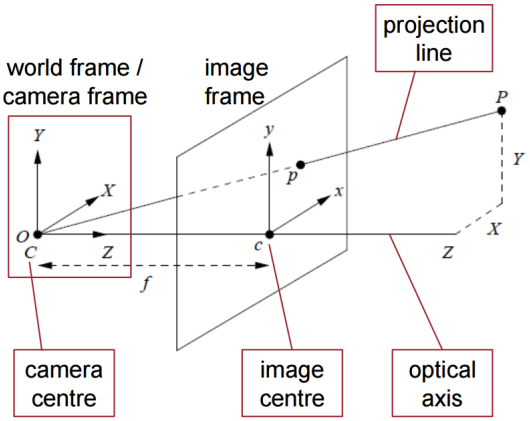


Figure 3.5: Camera Pinhole Model [43].

Knowing some features from the scene, it is possible to begin the intrinsics computation. The perspective projection from 3D world to the image can be established by the expressions 3.1 to 3.6, supported by the figure 3.5. A known point P is located in the world at $P = [X, Y, Z]^T$ and seen in the image coordinates at $p = [x, y]^T$.

It is possible to relate both points and extract the focal length values (f_x, f_y) based on triangles similarity (perspective geometry) through the equation 3.1:

$$\begin{cases} \frac{X}{Z} = \frac{x}{f_x} \\ \frac{Y}{Z} = \frac{y}{f_y} \end{cases} \iff \begin{cases} x = \frac{f_x * X}{Z} \\ y = \frac{f_y * Y}{Z} \end{cases} \tag{3.1}$$

To avoid non-linear division operations, the previous relation can be reformulated using the projective geometry framework in homogeneous coordinates [42], into a projection mapping matrix as:

$$\lambda * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} * \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.2}$$

Where $\lambda = Z$ is the homogeneous scaling factor and P in homogeneous coordinates is $P_h = [X, Y, Z, 1]^T$.

In practice, f_x and f_y can differ for a number of reasons [39]:

- Flaws in the digital camera sensor;
- The image has been non-uniformly scaled in post-processing;
- The camera's lens introduces unintentional distortion;
- The camera uses an anamorphic format, where the lens compresses a widescreen scene into a standard-sized sensor;
- Errors in camera calibration.

As mentioned before, the image center and the optical center of the sensor may not be aligned so, the previous equations must account with this offset in x and y axes.

Using the previous notation on equation 3.1 we can update it to:

$$\left\{ \begin{array}{l} \frac{X}{Z} = \frac{x}{f_x} \\ \frac{Y}{Z} = \frac{y}{f_y} \end{array} \right. \iff \left\{ \begin{array}{l} x = \frac{f_x * X}{Z} \\ y = \frac{f_y * Y}{Z} \end{array} \right. \iff \left\{ \begin{array}{l} x = \frac{f_x * X + c_x}{Z} \\ y = \frac{f_y * Y + c_y}{Z} \end{array} \right. \tag{3.3}$$

Once again, now using the projective geometry framework in homogeneous coordinates (3.3.4) like in equation 3.2, the center of the image can be added to the projection mapping matrix:

$$\lambda * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} * \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.4}$$

Until now it was implicitly assumed that the pixels of the image sensor was perfectly square and not skewed. However, both assumptions may not always be valid. For example, an NTSC TV system defines non-square pixels with an aspect ratio of $10 : 11$. In practice, the pixel aspect ratio is often provided by the image-sensor manufacturer. Second, pixels can potentially be skewed (figure 3.6), especially in the case that the image is acquired by a frame grabber. In this particular case, the pixel grid may be skewed due to an inaccurate synchronization of the pixel-sampling process. Axis skew causes shear distortion in the projected image [40].

Even rarely encountered, both previously mentioned imperfections of the imaging system can be taken into account in the camera model, using the parameters η and s, which model the pixel aspect ratio and skew of the pixels, respectively.

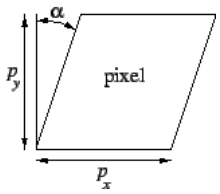


Figure 3.6: Pixels Skew.

Following the equation 3.4, the projection mapping can be now updated with this two error factors to:

$$\lambda * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & s & c_x & 0 \\ 0 & \eta f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} * \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} K & 0_3 \end{bmatrix} * P_h \tag{3.5}$$

In practice, when employing recent digital cameras, it can be safely assumed that pixels are perfectly square ($\eta = 1$) and non-skewed ($s = 0$).

From now on, the projection matrix that incorporates the intrinsic parameters and relates a point in the world to the camera image in homogeneous coordinates, which contains the **Camera Matrix**, is denoted as P and K respectively throughout this work.

$$K = \begin{pmatrix} f_x & s & c_x \\ 0 & \eta f_y & c_x \\ 0 & 0 & 1 \end{pmatrix} \tag{3.6}$$

3.2.2 Lens Distortion

Until now, the presented foundations above were done assuming that the world objects reflected light came cleanly through the pinhole to the image sensor but it is not entirely true in our days. To ensure an accurate image capture and more detailed geometry replication, the image sensor is "protected" by a lens where light converges before reaching

image sensor. This lens may be static or adjustable to modify focus of the image and the zoom the light rays into the image plane. **The point where all those light rays focus will, from now on, be considered the optical center with a focal length imposed by the lens (and not the pinhole) in all the presented concepts**.

This passage from the air through the lens and back again to the air, changes light travel due to refractions and lens misalignment, causing distortion effects that affects world replication into an image. Fortunately, all these "issues" have already been solved and can be minimized to get the original forms and details of the objects, lines, etc in the world. It only has to be done once for each camera if its lens is never moved after calibration.

There are two kinds of lens distortion [35]:

- Radial distortion (figure 3.7(a)) - occurs when light rays are refracted or bent more near the edges of a lens than they do at its optical center. The smaller the lens, the greater the distortion. The radial distortion coefficients model this type of distortion;

- Tangential distortion (figure 3.7(b)) - occurs when the lens and the image plane are not parallel/aligned. The tangential distortion coefficients model this type of distortion.

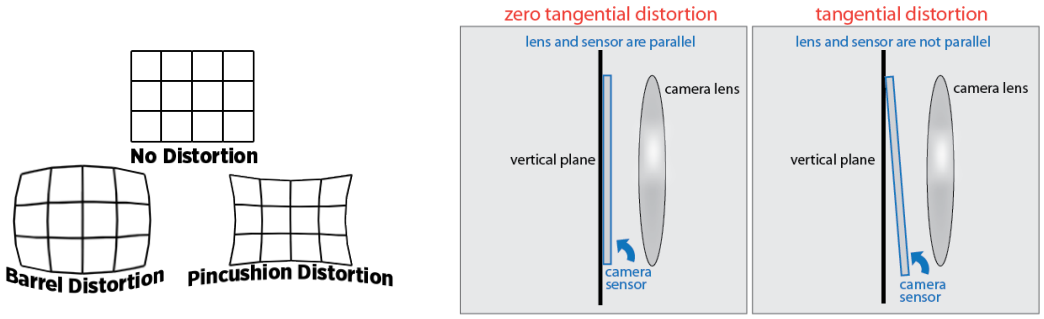


Figure 3.7: Lens radial [91] and tangential [92] distortion.

Once again, with previous knowledge of world object's dimensions/relations, this distortion parameters can be computed with already developed techniques using, for example, straight lines, perpendicular or parallel lines on the scene.

This distortion parameters or coefficients can be described mathematically below.

Let $(x_u, y_u)^T$ and $(x_d, y_d)^T$ be the undistorted/corrected and the measured distorted/non-corrected pixel positions on image plane, respectively.

The relation between an undistorted and a radial distorted pixel can be done by equations 3.7 and 3.8.

$$x_u = x_d(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.7}$$

$$y_u = y_d(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.8}$$

And the relation between an undistorted and a tangential distorted pixel can be done by equations 3.9 and 3.10.

$$x_u = x_d + (2k_4 y_d + k_5(r^2 + 2x)) \tag{3.9}$$

$$y_u = y_d + (k_4(r^2 + 2y_d) + 2k_5 x_d) \tag{3.10}$$

The estimation of the distortion parameters can be performed by minimizing a cost function that measures the curvature of lines in the distorted image. To measure this curvature, a practical solution is to detect feature points belonging to the same line on a calibration rig, e.g., a checkerboard (CkB) calibration pattern is used. Each point belonging to the same line in the distorted image forms a bended line instead of a straight line. By comparing the deviation of the bended line from the theoretical straight line model, the distortion parameters can be computed.

3.3 Extrinsic Sensor Calibration Parameters

Opposed to the intrinsic parameters that describe internal parameters of the camera (camera matrix and lens distortion coefficients) and belong unchanged (unless there are lens adjustment) to model the way as world objects are mapped/represented into a 2D image, the extrinsic parameters model location of the 3D world objects to the image plane and contrariwise. Camera extrinsic may change face to a world coordinate reference. This calibration is a mandatory step in 3D computer vision in order to extract metric information from 2D pairs of images. It is also the information about camera position and orientation to a relative/absolute coordinate system, even if it is placed in a camera optical center. The localization of general objects and cameras can be trivially computed and placed in a coordinate system but, to obtain camera sensor extrinsic calibration values, it is mandatorily done with the camera image itself since its image sensor is inside an enclosure with a lot of needed infinitesimal and impossible measures from outside.

So, before further explanations about extrinsics it is required to introduce rigid body transformations concepts.

3.3.1 Rigid body transformations (translations and/or rotations)

There are several ways to perform objects transformations but in all of them it is only applicable if they are considered as rigid objects. More specifically, any objects that keep its real world euclidean distances between all its particles upon a movement (translation and/or rotation). Due to elastic properties of real objects, it is not entirely true but more a reality approximation. Even so, this elastic deformations are almost negligible and we assume its absence from now on.

3.3.2 Rotations of rigid bodies

Not the perfect one of the many ways but the easier to quantify a rigid body rotation is upon euler angles over x, y and z axes. It can be parametrized [43] with 3DOF (Degrees of freedom) as presented on matrices 3.11, 3.12 and 3.13 and supported on figure 3.8. It is easily obtained by simple mathematical geometry principles and angles over the desired axes.

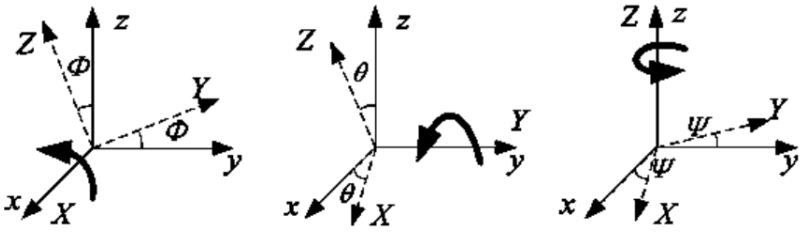


Figure 3.8: Euler angles [43].

The object rotations over x axis are represented by the angle ϕ, θ over y axis and ψ over z axis. Each axis rotations upon Euler angles was reduced to simple matrices:

$$R(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \tag{3.11}$$

$$R(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{3.12}$$

$$R(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

Based on the previous assumptions, the rotations over all the axes can be represented within a single rotation matrix $R = Rz(\psi) * Ry(\theta) * Rx(\phi)$.

$$R(\phi,\theta,\psi) = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \cos\psi\sin\phi\sin\theta - \cos\theta\sin\psi & \sin\psi\sin\phi\sin\theta + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\psi\sin\theta\cos\phi + \sin\phi\sin\psi & -\sin\phi\cos\psi + \sin\psi\sin\theta\cos\theta & \cos\theta\cos\phi \end{bmatrix} \tag{3.14}$$

This parametrization methodology is non-perfect due to the existence of singularities which forbid its use on systems whose angles may reach ±90 degrees, leading to parallel axis or alignment, where one DOF is lost because there are two axis rotating on the same plane. When this happens, we are facing an event known as Gimbal lock [44] and rotating an object within two different but coincident axis, will lead to the same mathematical calculations of body's rotation. There are six singularities/situations in all the euler rotation space where this event may occur, as described in figure 3.9.

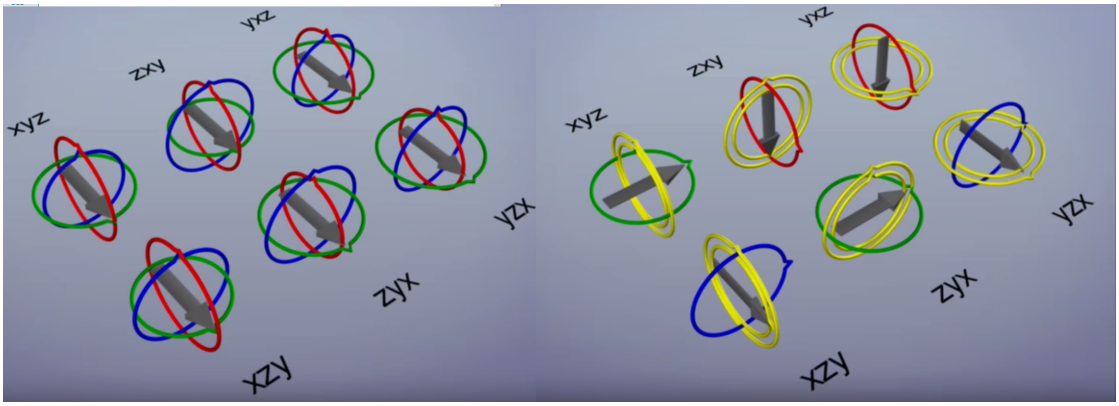


Figure 3.9: Euler singularities occurs when two axis get parallel [44].

This issue would be concerning if our system could face the referred ±90 degrees singularities where we would be forced to use a heavier and harder notation such as quaternions notations instead, where this problem does not occur.

3.3.3 Translations of rigid bodies

Objects may not only be rotated but also translated, at the same time or not. If only a translation is performed, it is a simple case where is "added an offset" in the desired axes to the current object coordinates.

The translation in the 3D space may be expressed by a 3D vector or column matrix as:

$$T(x, y, z) = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{3.15}$$

3.3.4 Homogeneous Coordinates

When a translation happens simultaneously to a rotation, it represents the distance over all the particles will rotate over.

For easier handle with rigid body transformations, the rotation and translation relations tend to be expressed by a single matrix which encapsulates both of them. Therefore, this matrix must include rotation (3.14) and translation (3.15) matrices, in homogeneous coordinates, represented by M_{CR} (3.16):

$$M_{CR} = \begin{bmatrix} R(\phi, \theta, \psi)_{3\times3} & T(x, y, z)_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}_{4\times4} = \Big[R|T\Big]_{4\times4} \tag{3.16}$$

So, to transform or map a point P in the world to another location P' or to another coordinate reference system, it can be performed as:

$$P' = M_{CR}P \tag{3.17}$$

Once presented the notions about rigid body transformations, it is easier to understand what extrinsics of a camera really are. They are no less than the rotations and the translations of the camera sensor relative to a coordinate system. Sometimes, the coordinate system may be centred on the camera itself leading to the translation and rotation of scene objects be represented face to its referential.

3.4 Objects projection

After the approach about the concepts of camera models, intrinsic camera parameters, lens distortion, rigid body transformations and extrinsic camera parameters, we are fully

ready to project a point from the 3D real world onto 2D coordinates in the built image plane.

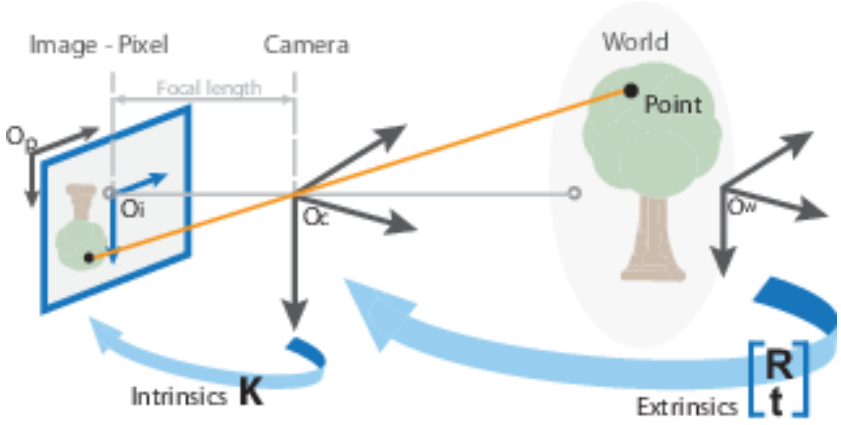


Figure 3.10: Point projection from the world to the image [103].

First, we need to get the projection of 3D point's coordinates ($P_{(x,y,z)^T}$) in the world coordinate reference system to 3D coordinates on the camera reference system $P'_{(x,y,z)^T}$ using its extrinsics (camera rotation and translation face to the world reference system):

$$P' = M_{CR}P = [R|T]\,P \tag{3.18}$$

Then, the projection of 3D points in the camera coordinate system to the 2D image plane ($p_{(x,y)^T}$) can be done using the intrinsics/camera matrix K:

$$p = KP' \tag{3.19}$$

So, using the previous knowledge from equations 3.17 ($P' = M_{CR}P$) and then 3.16 ($M_{CR} = [R|T]_{4\times4}$), the p point on image can be computed as:

$$p = KM_{CR}P = K\,[R|T]\,P \tag{3.20}$$

Resulting in:

$$p = \begin{bmatrix} f_x & s & c_x \\ 0 & \eta f_y & c_x \\ 0 & 0 & 1 \end{bmatrix}_{3\times3} \begin{bmatrix} R(\phi,\theta,\psi)_{3\times3} & T(x,y,z)_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}_{4\times4} P_{4\times1} \tag{3.21}$$

$$\iff p = \begin{bmatrix} f_x & s & c_x \\ 0 & \eta f_y & c_x \\ 0 & 0 & 1 \end{bmatrix}_{3\times3} \begin{bmatrix} R(\phi,\theta,\psi)_{3\times3} & T(x,y,z)_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}_{4\times4} P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.22}$$

3.5 Stereo Triangulation

Until now, we could only estimate 2D coordinates of 3D objects or its 3D coordinates in the camera reference system with no depth information in the world. With the introduction of new triangulation concepts and epipolar constraints, one or more calibrated cameras with distinct viewpoints but with overlapping FOVs can be used to compute the depth from points triangulation with more or less error. The error of this triangulation is directly dependent of cameras baseline (distance between camera's center of projection). This depth is a 3D reality/structures reconstruction of the seen objects, lying down on mathematical coordinates triangulation of the same points projected onto the several images, as referred above on chapter 3.4. This method of stereo triangulation can also be fulfilled with a single camera if it can perceive the same scene from different perspectives.

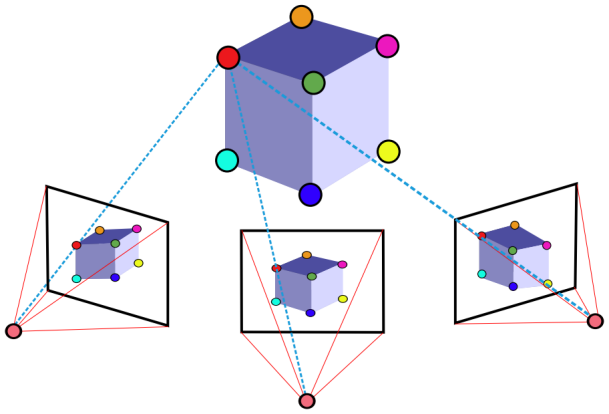


Figure 3.11: Points triangulation [45].

The triangulation problem is, in theory, a trivial problem. Knowing the previously presented camera parameters such as camera matrix (K), its distortion and the extrinsics of each camera (its position and orientation) in a global coordinate system to all the

cameras, it is possible to use each camera view/projection of a point to triangulate it and know its 3D coordinates in the global coordinate system as presented on figure 3.11.

There are several mathematical algorithms to solve the 3D reconstruction but we will focus only on the most common ones which have been proved reliable. So, it is indispensable to introduce other concepts such as:

- Epipolar geometry;
- Epipolar constraints;
- Planar homography;
- Homography decomposition.

3.5.1 Epipolar Geometry and Epipolar constraints

When available two calibrated images from distinct viewpoints/perspectives of the same scene, it allows the use of epipolar geometry based computations/constraints.

Epipolar geometry refers to the use of image's epipoles, its epipolar lines and camera's baseline to constraint the match of points from both cameras or positions (different positions for a single camera). With the projection of the same point in two different image planes, it is possible to build a plane between the **homogeneous coordinates** (see chapter 3.3.4) of both images optical centres and the point itself. This concepts can be supported by the figure 3.12.

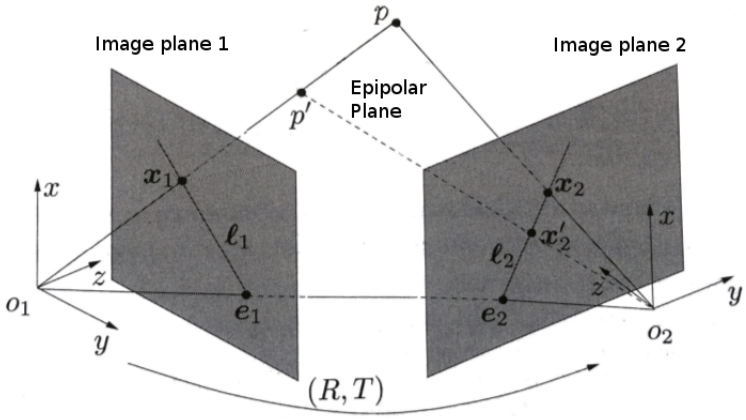


Figure 3.12: Epipolar Geometry [45].

Supporting on figure 3.12, we may observe the line which connects both image optical centres which is called as baseline. The intersection of that line with the image planes

(1 and 2) occurs at the epipoles e_1 and e_2, respectively. The point p is projected on each image plane as x_1 and x_2. The lines connecting each optical centre (o_1 and o_2) to the point in the world (p) are seen in the other images/cameras as semi-straight lines (the epipolar lines (l_1 and l_2), from the epipole (e_n) to the projected point on its image plane) coincident with the epipolar plane [46].

So, based on figure 3.12, knowing all camera parameters (intrinsics and extrinsics), we guarantee a calibrated stereo system and we are free to assume that:

• If a known p point is projected on left camera as x_1, the **epipolar line** which connects both optical centres and intersects image planes at the epipoles, is also known;

• The projection of p on the second image plane lays on the epipolar line l_2. This means that, for each new point which is projected on both images, it will always lay on the other's epipolar line (which resulted from a new epipolar plane formed by the optical centres and the new point);

• With this, we can verify or not the matching of two points in different images through epipolar constraints and assume to the correct triangulation and match of the both 2D points into 3D coordinates.

Epipolar constraints can be imposed to relate points from distinct viewpoints through a precise geometric relationship modelled by the **Essential Matrix**.

3.5.2 Planar Homography

Assuming a pinhole camera mode we can assure that any two images of the same planar surface in space are related by an homography (figure 3.13(a)). The Homography is a transformation (a 3×3 matrix) that maps the points in one image to the corresponding points in another image [93]. This has many practical applications, such as image rectification and extraction of camera motion from observed objects. Once estimated an homography between images of different cameras or view-points, it is possible to extract camera rotations and translations if the object has known dimensions and location. This matrix is useful for solutions like navigation, 3D models construction, mosaics construction (figure 3.13(b)) or augmented reality, so that they are rendered with the correct perspective and appear to a single shot of the original scene [47].

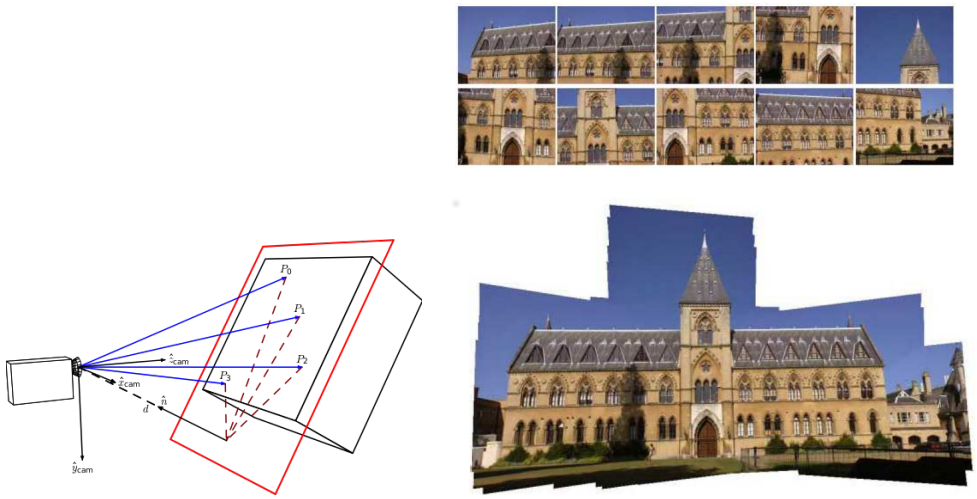


Figure 3.13: Planar homography extraction and mosaicking from several images [47].

If there are available some known points in the world whose locations are also known and they lay on the same planar surface, we can apply planar homography principles. Planar homography establishes the correspondence between projected points on world's plane to image's points from different perspectives. This technique is limited to coplanar points but it is useful when there are no checkerboard patterns to calibrate stereo vision systems. Therefore, coplanar points may be detected and used, not only, for the computation of this homography in order to get rendered images but also to relate the images/cameras through its extrinsics.

It is evident that the detected points must be static or extracted from all the cameras at the same time otherwise there would be points mismatching, leading to false calibrations/rendering.

Using previously geometry knowledge about vectors and previously formulated equations **??**, we may proceed to the planar homography formulation.

From the equation **??** ($X_2 = RX_1 + T$) , x_1 and x_2 points are obtained in the respectively images and they are under the epipolar geometry constraints.

Now, assume a N vector perpendicular to the homography plane P and d as the distance from the optical centre to this plane ($d > 0$). From this, we obtain [43]:

$$N^T X_1 = n_1 X + n_2 Y + n_3 Z = d \iff \frac{1}{d} N^T X_1 = 1 \tag{3.23}$$

Using equation 3.23 in the equation **??**, we get to:

$$X_2 = RX_1 + T = RX_1 + T\frac{1}{d}X_1 = \left(R + \frac{1}{d}TN^T\right) \tag{3.24}$$

Thus, the planar homography matrix can be represented by:

$$H \doteq \left(R + \frac{1}{d}TN^T\right) \tag{3.25}$$

3.5.3 Homography decomposition

As said before, the planar homography can be used for (assuming always calibrated images or known intrinsics calibration of each camera):

- Computation of this homography in order to get rendered images when known the extrinsics (R_n, T_n) of each camera;
- The extrinsics computation of each image/camera relative to the same planar surface;

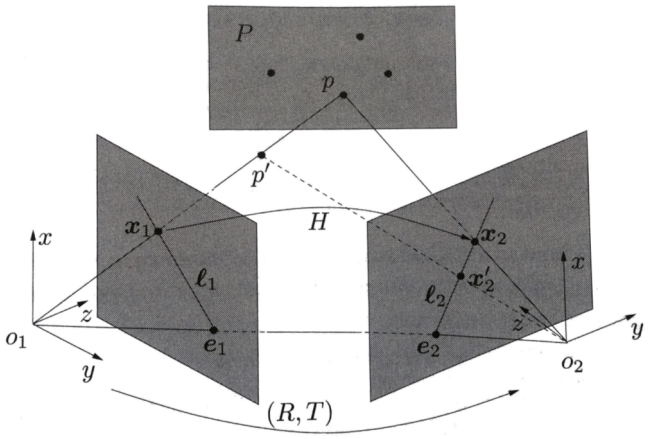


Figure 3.14: Planar Homography [43].

Therefore, we need to detect some points coordinates from the same planar surfaces in different perspectives/viewpoints in order to obtain camera's rigid transformations. The following formulations assume relative rigid transformations between images/cameras.

So, based on the figure 3.14, the decomposition of the homography is suitable to obtain through the Singular Value Decomposition factorization (SVD). This factoriza-

tion method decomposes the H matrix in three mathematically equivalent based on its eigenvalues and eigenvectors [48].

Thus, we obtain M:

$$M = SVD(H) = U \sum = V^T \tag{3.26}$$

Where $\sum$ is a diagonal matrix with the H eigenvalues, $\sum = (\sigma_1^2, \sigma_2^2, \sigma_3^2)$ and $\sigma_2^2 = 1$.

Assume $V = [v_1, v_2, v_3]$ and $H^T H = V \sum V^T$ where H preserves the size of any orthogonal vector to it. Then, we get:

$$H^T H v_1 = \sigma_1^2 v_1, \quad H^T H v_2 = v_2, \quad H^T H v_3 = \sigma_3^2 v_3 \tag{3.27}$$

It is assumed that v_2, N and T are orthogonal and that v_2 preserves its size when projected by H.

Thus, we get two vectors:

$$u_1 = \frac{\sqrt{1-\sigma_3^2} v_1 + \sqrt{\sigma_1^2 - 1} v_3}{\sqrt{\sigma_1^2 - \sigma_3^2}}, \quad u_2 = \frac{\sqrt{1-\sigma_3^2} v_1 - \sqrt{\sigma_1^2 - 1} v_3}{\sqrt{\sigma_1^2 - \sigma_3^2}} \tag{3.28}$$

And:

$$S_1 = span(v_2, u_1), \quad S_2 = span(v_2, u_2) \tag{3.29}$$

Assuming that v_2 is orthogonal to vectors u_1 and u_2, that $v_2 \hat{} u_1$ is normal to S_1 and that $v_2 \hat{} u_2$ is normal to S_2, so, $v_2, u_1, \widehat{v_2} u_1$ and $v_2, u_2, \widehat{v_2} u_2$ form orthogonal bases. Then, the following equations are also true:

$$Rv_2 = Hv_2, \quad Ru_i = Hu_i, \quad R(\widehat{v_2} u_i) = \widehat{Hv_2} Hu_i \tag{3.30}$$

Leading to:

$$\begin{aligned} U_1 &= [v_2, u_1, \widehat{v_2} u_1], \quad & W_1 &= [Hv_2, Hu_1, \widehat{Hv_2} Hu_1]; \\ U_2 &= [v_2, u_2, \widehat{v_2} u_2], \quad & W_2 &= [Hv_2, Hu_2, \widehat{Hv_2} Hu_2]; \end{aligned} \tag{3.31}$$

From here, we obtain **four possible solutions** for rigid body transformations or possible extrinsics:

Solution 1	$R_1 = W_1 U_1^T$ $N_1 = \widehat{v_2} u_1$ $\frac{1}{d} T_1 = (H - R_1) N_1$	Solution 3	$R_3 = R_1$ $N_3 = -N_1$ $\frac{1}{d} T_3 = -\frac{1}{d} T_1$
Solution 2	$R_1 = W_2 U_2^T$ $N_2 = \widehat{v_2} u_2$ $\frac{1}{d} T_2 = (H - R_2) N_2$	Solution 4	$R_4 = R_4$ $N_4 = -N_2$ $\frac{1}{d} T_4 = -\frac{1}{d} T_4$

Table 3.1: Possible solutions from planar homography.

From the four possible solutions, we need to exclude the three wrong ones. First, we need to exclude the solutions that place cameras behind the object, which happens in two of them. From the two remaining ones, is discarded the solution which places the triangulation points from images further from the 3D known points.

3.6 Image processing

All the referred concepts are suitable and high implemented within computer vision applications. Progressive improvements in the stereo concepts and cameras relations understanding have been simplifying the processing costs which the high resolution and frame rate of actual cameras entail.

Images have a lot of valuable features but also a lot of useless information. So, to avoid high cost and processing delay or dragging, several techniques were developed and may be applied in order to reduce the time and energy necessary to handle computer vision information.

The most simple methods lay down on the extraction of Regions Of Interest (ROIs) or Objects of Potential Interest (OPIs) from the whole image. This objects or regions are usually extracted from high resolution images, which means, tons of pixels information to process and, if we are dealing with a coloured image, we must take into account that each of these pixels has three channels of color (most of the times in the RGB color-space) as referred above, which triple the quantity of information to handle. So, the image is firstly analysed in a "sketchy" way to detect those OPIs or ROIs and then in a deeper and more detailed way (it can be understood as a zoomed, focused or sub-pixel way) to ensure information extraction reliability and precision.

3.6.1 Background subtraction

In dynamic scenes but with statical cameras, one of the major and first steps in the information filtering from a image, is the background extraction or, also known, as foreground detection.

Background information can be modelled to include all the pixels that remain with static color and intensity or thresholded changes/ranges of it over the time. Even to the human eye, the pixels may seem static, there are slight changes of reflected illumination leading to changes of color and intensity. Background models can help filtering pixels out of thresholded ranges, excluding those high dynamic noise movements or undesirable and repetitive motions such trees or sea-waves. The foreground information, are all the other non-static pixels.

When we want to detect dynamic OPIs from the foreground, this will translate not in the removal of the background pixels from an image but in a information filtering to analyse. Pixels within thresholded/ranged values of background/foreground will be taken into account and considered valuable.

Analysing only the foreground information will reduce the quantity of information/pixels to focus on and can improve significantly the time and energy spent on features characterization.

There are several background extraction methods based on:

- Frame differencing;
- Image mean filtering;
- Adaptive image mean filtering;
- Gaussian average;
- Background mixture models;

3.6.2 Frame differencing

The simplest way to implement this is to initialize background model taking an image of the scene ($t = 0$) and take subsequent frames/images obtained at the time t, denoted by $Im(t)$, to compare with the background model image, denoted by $B(t = 0)$. Using simple arithmetic calculations, we can extract the foreground objects simply by using the image subtraction technique of computer vision. That is, for each pixel in $Im(t)$, take the pixel value denoted by $P[Im(t)]$ and subtract it with the corresponding pixel at the same position on the background model image denoted as $P[B]$.

The mathematical equation to detect pixels belonging to foreground ($F(t)$) can be written as:

$$P[F(t)] = P[Im(t)] - P[B(t=0)] \tag{3.32}$$

This image subtraction would show only some intensity for the pixel locations which have changed between the two frames. Though we have seemingly removed the background, this approach will only work for cases where all foreground pixels are moving and all background pixels are static, which usually doesn't happen. Small changes in object's luminance or position would lead to tons of false objects placed on the foreground. So, the background model must be constantly updated and a "threshold" or range must be imposed to this image differencing to improve the subtraction and background pixels exclusion (image thresholding). This thresholding is simply done by:

$$\text{minThreshold} \leq |P[F(t)] - P[F(t-1)]| < \text{maxThreshold} \tag{3.33}$$

Or,

$$\text{minThreshold} \leq |P[Im(t)] - P[B(t)]| < \text{maxThreshold} \tag{3.34}$$

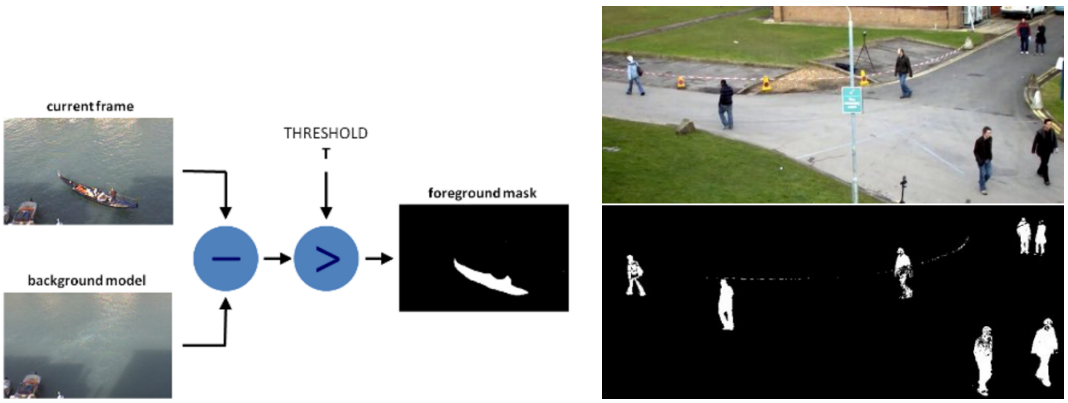


Figure 3.15: Background subtraction and high frequency noise [49] [50].

The accuracy of this approach depends directly of the motion scene frequency. Sometimes, the initialization is not enough and high frequency (faster) movements such as passing clouds shadows and trees shake may require higher thresholds and frequent background model updating as presented on figure 3.15(b).

3.6.3 Image mean filtering

Here, the background model results from a series of preceding averaged images. For calculating the background image at the instant t, it can be mathematically formulated as:

$$B(x, y, t) = \frac{1}{N} \sum_{i=1}^{N} V(x, y, t - i) \tag{3.35}$$

Where N is the number of preceding images taken for averaging. This averaging refers to corresponding pixels in the given images. This number of given images N, directly depends on image frame rate. After calculating the background $B(x, y, t)$, we can then re-use the image differencing concepts similar to frame differencing 3.32 and subtract it from the image $V(x, y, t)$ at time $t = t$ and then threshold it. Thus, the foreground is the outcome of:

$$|V(x, y, t) - B(x, y, t)| > \text{Threshold} \tag{3.36}$$

3.6.4 Adaptive image mean filtering

This is an improvement of the previous one because it doesn't stores images to model background but, instead, weights pixels intensities/colors. This method was the implemented one on this work because it showed to be the most consistent, versatile and sensitive to foreground update.

First is acquired an image to initialize background model. Then, for each new image, a global coefficient/weight α must be defined which will weight each one of the new image pixels and then, this weighted image is added to previous background model pixels also weighted by $(1-\alpha)$. The α coefficient is the background learning factor and dictates the speed of background model update. Its value should be between 0 and 100%. The higher the value, the more quickly the system learns the changes and updates the background model. Therefore, for a static background, lower values like 0.001 are fine but, if the background has high frequency motions like moving trees, higher values must be set like 0.01.

$$B(x, y, t) = \alpha F(t) + (1 - \alpha) B(x, y, t - 1) \tag{3.37}$$

Then, similar to the previous methods, using image differencing, each pixel of the new background model $B(x, y, t)$ is subtracted by the new frame $F(t)$ and then thresholded to extract only foreground objects:

$$|B(x, y, t) - F(t)| > \text{Threshold} \tag{3.38}$$

3.6.5 Gaussian average and Background mixture models

Usage of global (same value for all pixels in the image) or time-independent/static thresholds may limit the accuracy of the above three approaches, suitable to scenes where we get a easy image of background alone, like an image of a room without visitors, image of the road without vehicles etc. In those cases, moving objects extraction is an easy job once they appear on the scene. In most of the cases, you may not have such an image, so we need to extract the background from whatever images we have. It become harder when there is object's shadows. Since shadow is also moving, simple subtraction would mark that also as foreground. So, there are other existing methods already available on OpenCV libraries to handle this issues and model background:

- Gaussians mixture models (GMM);
- Bayesian inference;

Gaussian mixture-based Background/Foreground Segmentation Algorithms were introduced in [51] and use a method to model each background pixel by a mixture of K Gaussian distributions ($K = 3$ to 5). The weights of the mixture represent the time proportions that those colours stay in the scene. The probable background colours are the ones which stay longer and more static.

Bayesian mixture-based models combine statistical background image estimations and per-pixel Bayesian segmentation. It was introduced in [52] and uses first few (120 by default) frames for background modelling. It employs probabilistic foreground segmentation algorithm that identifies possible foreground objects using Bayesian inference. The estimates are adaptive, newer observations are more heavily weighted than old observations to accommodate variable illumination. Several morphological filtering operations like closing and opening are done to remove unwanted noise.

Since both methods principles were not used and it has extensive and complex mathematical explanations, it's formulas won't be presented here.

3.6.6 Color Spaces

Coloured images brought several advantages for artificial vision and there are several ways to represent a color. Therefore the concept of color space must be introduced. A color space is a specific organization of colors. A single color can be represented on the following color spaces:

- Gray-Scale;
- RGB;
- HSV;
- HSL;
- YCrCb;
- CIE L*a*b*;
- CIE XYZ;
- CIE L*u*v*;
- Bayer.

If we are handling with a black and white sensor, the output image will be retrieved in a single channel since pixels have no color information. Each pixel is represented only by values of intensity. Even so, if we have a coloured image (usually represented by three channels minimum) and need to "see" it in a single channel Y (gray image), the conversion is a combination of channel weights based on sensor standard principles. Image sensors standard color space is RGB and it's conversion is [54]:

$$\text{RGB[A] to Gray-scale: } Y = 0.299 * R + 0.587 * G + 0.114 * B$$

The opposite, color extraction/recover from gray-scale images, is no longer possible since there is no way to guess colors:

$$\text{Gray to RGB[A]: } R = Y,\ G = Y,\ B = Y,\ A = max(ChannelRange)$$

In the RGB color space, the widely used, each pixel color is represented by three channels (R=Red, G=Green, B=Blue). This color space has de disadvantage of non-continuity on color representation since it is the linear combination of those three channels, resulting on a cubic color space which leads to ambiguous situations such as the

similar colors presence on opposite boundaries of color space as seen on figure 3.16(a). This issue makes the RGB color space unusable under some feature coloured detection algorithms.

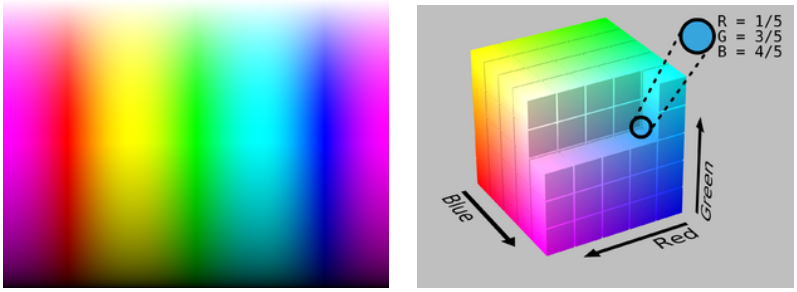


Figure 3.16: RGB color space representations [54].

Another known issue from RGB color space is pixel color saturation. The pixel color saturation happens when the image sensor gets more reflected rays than it can handle. So, a coloured light bulb for example, may reflect to much light in it's centre and less bright colors around it, leading to centered white spots surrounded by the light bulb color, when observed by cameras.

Thus, other color-space representations were developed such as HSV. This was one of the most used on this work due to it's special way to characterize a color without ambiguous situations, continuous color representation and easy pixel saturation detection. This last one was an hindrance found along this work since it was designed for active light coloured balls detection.

So, the workaround to handle with this issue was converting the RGB image to the HSV color space. Where HSV stands for Hue-Saturation-Value and splits the color into "Hue" which is the pixel color/tone, "Saturation" is the amount of white color received and "Value", also called lightness, describes how dark the color is.

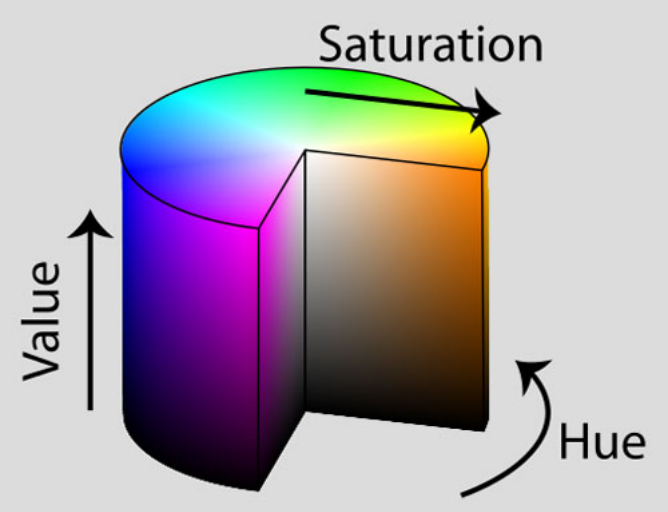


Figure 3.17: HSV color space representations [54].

As said before, active coloured balls frequently cause the image sensor saturation so, the tests of active coloured balls detection in the different color spaces and individual channels filtering let us to conclude that they could be easily detected using just the "Value" channel of the HSV color space. Thus, even the user can choose from the different color spaces to use, the default one is the HSV and all the background extraction and balls detection is made upon the "Value" channel. All the other color space representations won't be explained in detailed since there are no advantages for this work.

The RGB conversion to the HSV representation can be done by:

$$V \leftarrow max(R,G,B)$$

$$S \leftarrow \begin{cases} \frac{V-min(R,G,B)}{V} & \text{, if } V \neq 0 \\ 0 & \text{, otherwise} \end{cases}$$

$$H \leftarrow \begin{cases} \frac{60(G-B)}{V-min(R,G,B)}, & \text{if } V = \text{R} \\ 120 + \frac{60(B-R)}{V-min(R,G,B)}, & \text{if } V = \text{G} \\ 240 + \frac{60(R-G)}{V-min(R,G,B)}, & \text{if } V = \text{B} \end{cases}$$

If $H < 0$ then $H \leftarrow H + 360$. On output $V \in [0,1]$, $S \in [0,1]$ and $H \in [0,360]$. In computation systems, it is often represented in the range $[0,255]$.

3.6.7 Feature detection and tracking

A feature is a piece of image information which is relevant for solving computational tasks related to a certain application. The feature concept is very general and the choice of features in a particular computer vision system may be highly dependent on the specific problem to handle, leading, most of the times, to an algorithm specialization on specific features detection deteriorating other features detection. It is a specific "image region" which differs from its immediate neighbourhood. It is often associated with a change of an image property or several properties simultaneously, though it is not necessarily localized exactly where this change takes place. Pixels intensity, color gradients and textures are the most commonly observed/wanted properties. Thus, there are lots of interesting features on a single image such as lines, corners, circles or other specific shapes always made of a set of special points, which can be retrieved through the pixel(s) neighbourhood overhang.

Then, extracted features are stored in the system and represented not only by its image coordinates but also by a manner to identify and compare them for useful purposes later explained. Therefore, features must be differentiable by its feature descriptors as seen on figure 3.18.

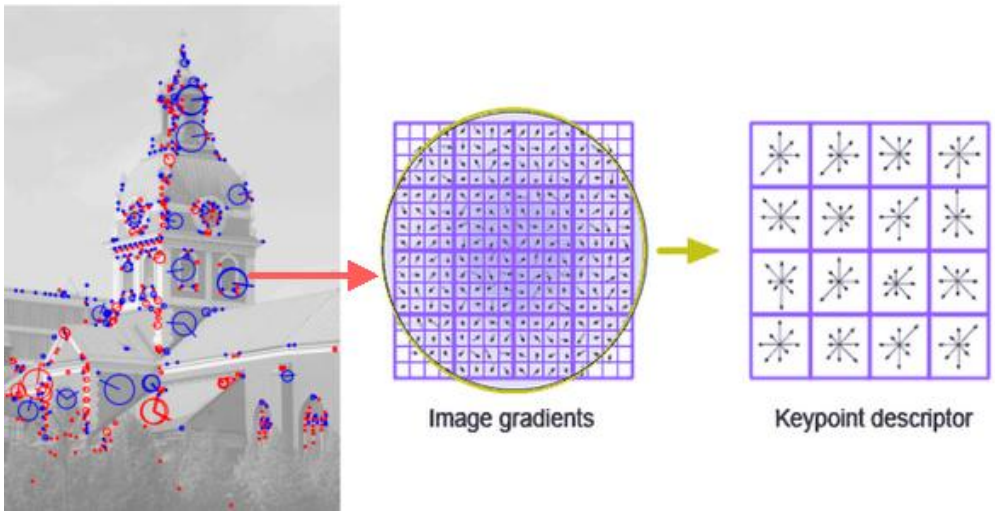


Figure 3.18: Feature detections and feature descriptors example [55].

Most common methods for feature detection are:

- Harris corner detector [56];
- FAST corner detector [57];
- SIFT - Scale-Invariant Feature Transform [59];
- BRIEF - Binary Robust Independent Elementary Features [63];
- ORB - Oriented FAST and Rotated BRIEF [65];
- SURF - Speeded-Up Robust Features; [61]
- Hough Transform - lines and circles detector [67];

Efficient detectors will lead to a robust system information extraction and reliable information that can be used for vision purposes such as object matching and tracking, motion extraction, stereo calibrations, object's world localization, etc. This work requires objects detection, matching and tracking methods to follow the detected active light ball patterns. The more consistent it is, the more accurate and reliable will be our work. In order to achieve intrisics and extrinsics real-time calibration and pattern attitude computation also under real-time constraints, there were studied several methods for feature detection and chose one or a mixture of them as explained later.

3.6.8 Harris corner detector

This feature detector turn simple ideas into mathematical formulas. It basically finds the difference in intensity for a pixels displacement (u, v) in all directions. This is expressed as:

$$E(u,v) = \sum_{x,y} \underbrace{w(x,y)}_{\text{window function}} \; [\underbrace{I(x+u, y+v)}_{\text{shifted intensity}} - \underbrace{I(x,y)}_{\text{intensity}}]^2 \tag{3.39}$$

Window function may be represented either by a rectangular window or gaussian window which weights pixels underneath it. $E(u, v)$ function must be maximized for corner detection. That is, maximize the second term. Applying Taylor Expansion to equation 3.39 and applying some mathematical principles, we get to the equation:

$$E(u,v) \approx [u \;\; v] M \begin{bmatrix} u \\ v \end{bmatrix} \tag{3.40}$$

where:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \tag{3.41}$$

I_x and I_y are image x and y derivatives respectively.

Then, a pixel score equation is created to determine if a window can contain a corner or not:

$$R = det(M) - k(trace(M))^2 \tag{3.42}$$

Where:

- $det(M) = \lambda_1 \lambda_2$;
- $trace(M) = \lambda_1 + \lambda_2$;
- λ_1 and λ_2 are the eigenvalues of M.

So, these eigenvalues decide whether a region is corner, edge or flat:

- If $|R|$ is small, which happens for small λ_1 and λ_2 values, the region is flat;
- For $R < 0$, when λ_1 ¿¿ λ_2 or vice versa, the detected region is an edge;
- And if R is large, when λ_1 and λ_2 are large and $\lambda_1 \sim \lambda_2$, the region is a corner;

It can be represented by the figure 3.19(a).

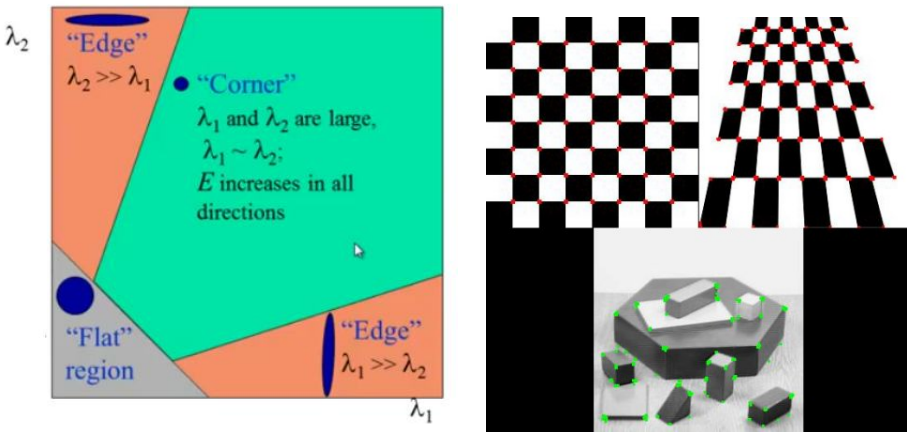


Figure 3.19: Harris corner detector [56].

So the outcome of Harris corner detection is a grayscale image with those score equation outputs. Thresholding them for suitable values, output image contains only the corners in the image.

3.6.9 FAST corner detector

One of the most demanding cases where feature real-time detection is crucial would be on Simultaneous Localization and Mapping (SLAM) applications which have low or limited computational resources and often a lot of information to handle in sensory fusion.

So, the previous presented corner detection method is not fast enough, literally. The workaround was a different method called Features from Accelerated Segment Test (FAST), proposed by Edward Rosten and Tom Drummond [57]. To perform corner detections in a faster way, this algorithm presents some steps:

1. Select a pixel p in the image to be characterized as an interesting point or not (let its intensity be I_p);

2. Select appropriate threshold value th.

3. Consider a circle of sixteen (16) pixels around the testing pixel 3.20;

4. Now, the pixel p is a corner if there are a set of n contiguous pixels in the circle (of sixteen pixels) which are all brighter than $I_p + th$, or all darker than $I_p - th$ (shown as white dash lines on figure 3.20 where n default value was twelve (12));

5. Then, a high-speed test was proposed to optimize and exclude a large number of non-corners. This step examines only the four pixels at one (1), nine (9), five (5) and thirteen (13) positions (first 1 and 9 are tested if they are too brighter or darker. If so, then 5 and 13 pixels are checked too). If p contains a corner, then at least three of these must all be brighter than $I_p + th$ or darker than $I_p - th$. If neither of these situations happens, then p cannot be a corner. The full segment test criterion can then be performed on passed candidates by examining all pixels in the circle.

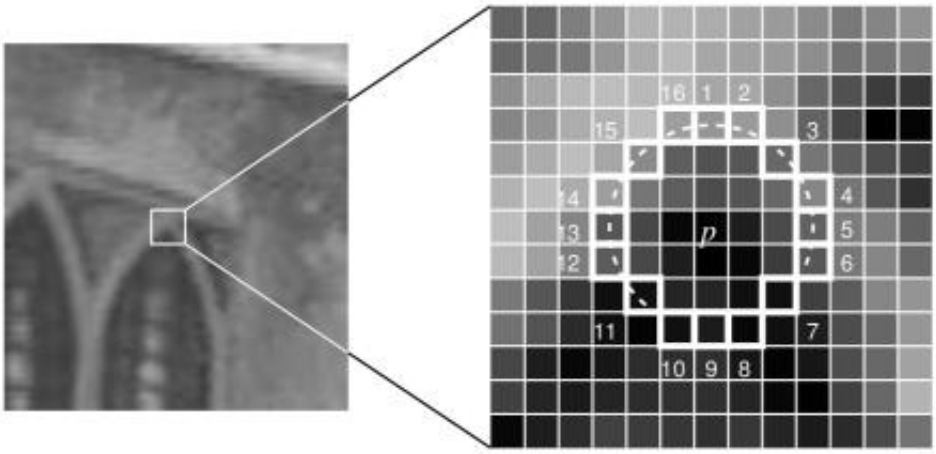


Figure 3.20: FAST corner detector [57].

This is a high performance detector but also with several weaknesses:

- It does not reject as many candidates for $n < 12$;
- The choice of pixels analysis is not optimal because its efficiency depends on ordering questions and distribution of corner appearances;
- Outcome of high-speed tests are thrown away;
- Bad features distribution, this is, multiple features are detected adjacent to one another.

3.6.10 SIFT - Scale-Invariant Feature Transform

Until now, the presented feature detectors are rotation-invariant, this is, we can find the same corners even if the image is rotated. It may be true because corners remain corners in rotated images but, the same doesn't happens when there is image scaling. A corner may not be a corner anymore if the image is scaled as presented on figure 3.21 where a corner in a small image became a flat line when zoomed. So, Harris and FAST corner detectors are not scale-invariant [58].

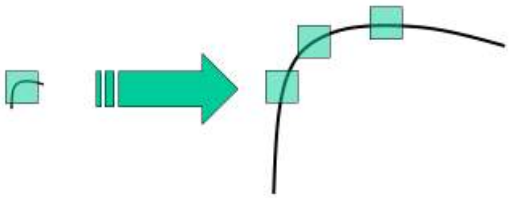


Figure 3.21: SIFT - invariant scale and rotation corner detector. [58].

Fortunately, a new algorithm called Scale Invariant Feature Transform (SIFT) was developed in 2004 by D.Lowe from University of British Columbia. This algorithm extracts keypoints and compute its descriptors [59]. This is a **patented and non-free algorithm** thus it is not open-source available.

There are mainly five steps involved in SIFT algorithm [58]:

- Scale-space Extrema Detection;
- Keypoint Localization;
- Orientation Assignment;
- Keypoint Descriptor;
- Keypoint Matching.

3.6.10.1 Scale-space Extrema Detection

Based on figure 3.22(a), we can perceive that we can't use the same window to detect keypoints under different scales. It fits for small corners but won't for larger corners where larger windows are also needed. Therefore, scale-space filtering is used where a LoG (Laplacian of Gaussian) is found for the image with distinct σ values. LoG acts just like a blob detector which detects different size blobs due to σ changes. Summing up, σ parameter is no less than a scaling factor.

By the figure 3.22(a), it is instantaneously that a gaussian kernel with low σ values detects high number of small corners while a guassian kernel with high σ values fits well for larger corners detection. So, finding the local maxima across the scale and space, it gives us a list of (x, y, σ) values where are potential keypoints at (x, y) under a σ scale.

This LoG is quite processing costly so, SIFT algorithm, uses the Difference of Gaussians which is an approximation of LoG. Difference of Gaussians translates into the difference of Gaussian blurring of an image with two different σ values (σ and $k\sigma$). This

process employed for different image octaves in Gaussian Pyramid as presented on figure 3.22(a).

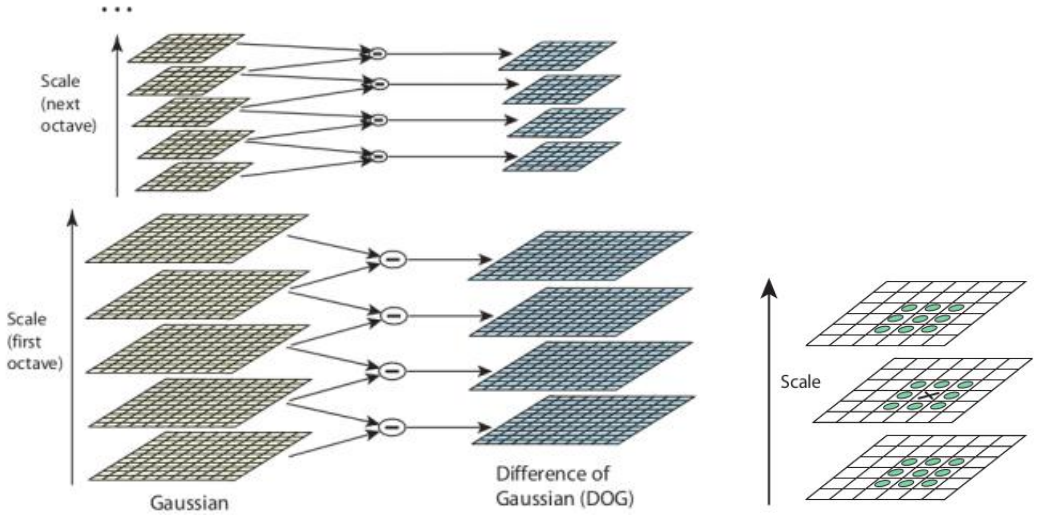


Figure 3.22: SIFT detector [58].

Once this Difference of Gaussians (DoG) are found, a search for local extrema over scale and space is performed on images. As displayed on figure 3.22(b), one pixel in an image scale is compared with its eight (8) neighbours as well as the nine (9) pixels in next scaled window and the nine (9) pixels in previous scale windows. It is a potential keypoint if there is a local maxima which also means, that keypoint is best represented under that scale value.

3.6.10.2 Keypoint Localization

After the found potential keypoints locations, they have to be refined to improve results accuracy. Taylor series expansion of scale space are used to get a more accurate location of space-scale extrema and, for intensity values at this extrema under a threshold value, it is rejected. This is also known as contrastThreshold in OpenCV.

DoG is also high auspicious for edges detection, so edges also need to be removed. Therefore, a concept similar to Harris corner detector is used using a 2x2 Hessian matrix ($Hess$) to compute the principal curvature. We know from Harris corner detector that for edges detection, one eigenvalue (λ) is larger than the other so, here they used a simple function if the ratio between them is greater than a threshold (known as edgeThreshold in OpenCV), that keypoint is discarded. Thus, it eliminates any low-contrast keypoints

and edge keypoints remaining only the strongest interest points.

3.6.10.3 Orientation Assignment

Similar to other methods, an orientation is assigned to each keypoint to guarantee image rotation invariance. A region/neighbourhood is taken around the keypoint location depending on the scale value and then the magnitude and direction gradient is calculated in that same region. Then, there is created an orientation histogram with 36 bins covering 360 degrees. The highest peak in the histogram is taken and any peak above 80% of its value is also considered for orientation calculations. It creates keypoints with same location and scale, but different directions contributing to stability of features matching.

3.6.10.4 Keypoint Descriptor

Now keypoint descriptor is created based on the 16x16 neighbourhood bins around the taken keypoint. It is sub-devided into sixteen (16) sub-blocks of 4x4 size. Then, for each one of those sub-blocks, a eight (8) bin orientation histogram is created leading to a total of 128 bin values available. Each one of these bins is represented as a vector to form the keypoint descriptor as seen on figure 3.18. In addition, several measures are taken to achieve robustness against illumination changes, rotations, etc.

3.6.10.5 Keypoint Matching

To relate features from different frames, image keypoints are matched by identifying their nearest neighbours. It may seem to work perfectly but, in some cases, a second closest-match may be very near to the first. It usually happens due to noise or other random reasons. When facing this issue, a ratio of closest-distance to second-closest distance is taken and if it is greater than 80%, they are rejected. It assures nearly 90% elimination of false matches while discards only 5% of correct matches [59].

3.6.11 SURF - Speeded-Up Robust Features

SIFT is nice for keypoint detection and description but it is relatively slow making it unusable on some real-time applications where there huge quantities of visual information. So, another new method was introduced in 2006, resulting from an SIFT improvement,

the "SURF: Speeded Up Robust Features" [61]. As name suggests, it is a speeded-up version of SIFT [60].

In SIFT, Laplacian of Gaussian (LoG) was approximated by Difference of Gaussian (DoG) for scale-space computing. Now, SURF goes even further and approximates LoG with Box Filters.

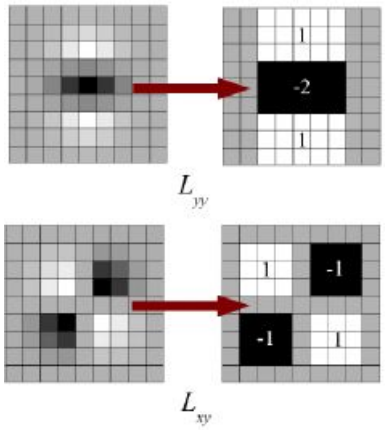


Figure 3.23: SURF - Speeded-Up Robust Features detector [60].

The proposed approximation is presented on Figure 3.23. Convolution with box filter can be easily achieved with the help of integral images and it can be done in parallel for different scales with high yield. The SURF also relies on determinant of Hessian matrix for both scale and location.

Haar-wavelet responses in horizontal (x) and vertical (y) directions for a $6s$ (six s) neighbourhood (where s is the scale at which the interest point was found) are used for orientation assignment, suitable guassian weights are also applied to it and then displayed as seen on figure 3.24. The prevailing orientation is estimated by the sum of all responses within a 60 degrees sliding orientation window. Wavelet response can be easier found out using integral images at any scale s. For many applications, where rotation invariance is not required and there is no need of orientation information, the process can be even faster. SURF allows the orientation bypass which "ignores" features orientation improving speed and showing high robustness up to $\pm 15^{\circ}$ rotations. This is known as the Upright-SURF or U-SURF.

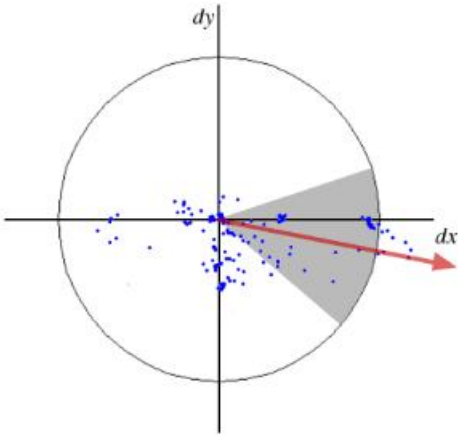


Figure 3.24: SURF orientation [60].

For feature description, SURF uses again Haar-wavelet x and y direction responses, simplified with the use of integral images. A neighbourhood of $20sby20s$ size is taken around the keypoint [61]. It is then subdivided into 4x4 subregions. For each subregion, horizontal and vertical wavelet responses are taken and a vector v is formed:

$$v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|) \tag{3.43}$$

When v is represented as a vector, turns SURF feature descriptor with a total of 64 dimensions. Lower the dimension, higher the speed of computation and matching, but provide better distinctiveness of features [60].

SURF feature descriptor can go even further and be extended to a 128 dimension version where sums of d_x and $|d_x|$ are computed separately for $d_y < 0$ and $d_y \geq 0$ ranges for better distinctiveness. Similarly, the sums of d_y and $|d_y|$ are split up according to the sign of d_x, thereby doubling the number of features without much computation complexity addition.

Another improvement is the use of Laplacian sign (trace of Hessian Matrix) for underlying interest point which adds no computation charges since it is already done upon detection. The Laplacian sign distinguishes bright blobs on dark backgrounds from the opposite situation.

The matching stage only compares features if they have the similar type of contrast as presented on figure 3.25. This translates into a reduction of information quantity which allows faster matching without reducing the descriptor's performance.

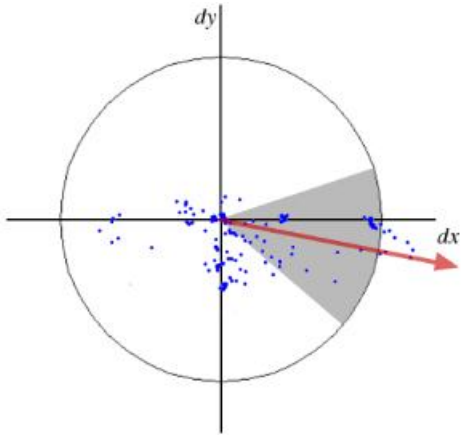


Figure 3.25: SURF matching [60].

Empirical evidences shows it is three times faster than SIFT while its performance is similar and better accuracy [61]. SURF is good at handling images with blurring and rotation, but not so good handling viewpoint and illumination changes.

3.6.12 BRIEF - Binary Robust Independent Elementary Features

SIFT uses exactly 128-dim vector for descriptors using floating point numbers, which basically takes 512 memory bytes. SURF also takes a minimum of 256 bytes (for 64-dim descriptor). This is a restrictive condition since creating such vectors for thousands of visual features requires high computational resources such as large memory capacity to store all those feature descriptors and high processing speeds to handle/match them between frames.

BRIEF method purposes that all these descriptor dimensions may not be actual needed for matching, using short descriptors, reminding that this is only a feature descriptor so it doesn't provide any method to find the features.

The improvements of this method lay on several methods to apply dimensionality reduction like Principal Component Analysis (PCA), Latent *Dirichlet* Allocation (LDA) or Locality Sensitive Hashing (LSH) to original descriptors such as SIFT descriptors presented in floating point numbers to binary strings. These binary strings are be used to match features using Hamming distance (it measures the minimum number of substitutions required to change one string into the other). This leads to a performance improvement since finding hamming distance is just applying a logic *XOR* and a *bit count* (hashing) but it wouldn't solve the initial problem of memory since it would be receiving the feature descriptors given by SIFT or SURF occupying large quantities of memory

too. It came to improve feature characterization given by floating point values of the descriptor vector through very few bits per value without loss of recognition performance leading to lower storage memory needs and faster feature matching.

So, while effective, these approaches to dimensionality reduction always require previous computing of the full descriptor, before further processing can take place. Thus, BRIEF purposes yield good compromises between speed, storage efficiency and recognition rate. It provides a shortcut to find an efficient feature point descriptor using binary strings directly extracted from image patches without computing any descriptors. It takes smoothed image patches (p) and selects a set of $n_d(x, y)$ location pairs along the same line. Then, simple intensity difference tests (τ) of pixels are done on these location pairs [63]:

$$\tau(p; x, y) := \begin{cases} 1 & \text{, if } \mathrm{I(p)} < \mathrm{I(q)} \\ 0 & \text{, otherwise} \end{cases}$$

Where $p(x)$ is the pixel intensity in a smoothed version of p at $x = (u, v)^T$. This is applied for all the n_d location pairs to get a n_d-dimensional bitstring. So once we get this, Hamming Distance can be performed to match these descriptors from different images.

Summing up, BRIEF is no more than a faster method for feature descriptor calculation and matching. It also provides high recognition rates with equal or better outcomes unless there is large in-plane rotation [62].

3.6.13 ORB - Oriented FAST and Rotated BRIEF

This detector merges other two methods presented above, the FAST detector and BRIEF descriptors to enhance storage memory usage, higher detection and feature matching yields. It was born in 2011 at "OpenCV Labs" by the hands of Ethan Rublee, Vincent Rabaud, Kurt Konolige and Gary R. Bradski. They designed a new and alternative approach to get even faster and accurate results face to SIFT and SURF detectors [64]. One of the main advantages is that this is a non-patented algorithm, thus, open-source and free for any purpose, achieving similar or even better results face the other two mentioned algorithms.

To solve the initial problems of memory usage present on SIFT and SURF, they used FAST feature detector to find faster keypoints with low memory storage required, with the disadvantage of being a detector with no orientation component. Then, as it does not produce a measure of cornerness, it is employed a Harris corner measure to

find the ordered top N points among them. FAST also does not compute multi-scale features. So, in order to become scale invariant, they use a pyramid of the imageto produce multiscale-features at each level in the pyramid (filtered by Harris measures).

To overcome the orientation lack and ensure rotation invariance detector, they purposed a simple but effective measure of corner orientation, the intensity centroid [65]. The intensity centroid assumes that a corner's intensity is offset from the corner center, and the direction of the vector from this the corner center to its intensity centroid may be used to impute an orientation.

First, to compute the intensity weighted centroid of the patches with the found corner at the center, intensity moments of a patch are used:

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \tag{3.44}$$

With this moments, we can compute centroid:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \tag{3.45}$$

Now, we are able to construct the needed vector from the corner's center to the centroid $\overrightarrow{OC}$ and obtain the orientation of the patch from:

$$\theta = atan2(m_{01}, m_{10}) \tag{3.46}$$

Where atan2 is the quadrant-aware version of arctan. To improve the rotation invariance of this measure, make sure that moments are computed with x and y remaining within a circular region of radius r which was empirically chosen to be the patch size, so that that x and y run from $[-r, r]$. The figure 3.26 can elucidate the presented concepts.

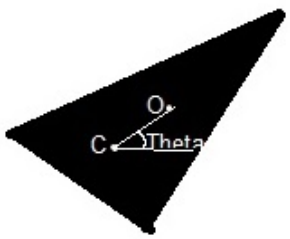


Figure 3.26: ORB centroid and orientation computation.

Then, BRIEF descriptors are used despite its low rotations tolerance, steered ac-

cording to the keypoint's orientation previously computed. This is done for any feature set of n binary tests (equation 3.47) at location (x_i, y_i) (equation 3.48), defining pixels coordinates by $S = 2 \times n$ matrix (equation 3.49).

$$\tau(p; x, y) := \begin{cases} 1 & , \text{if } \mathbf{p}(\mathbf{x}) < \mathbf{p}(\mathbf{y}) \\ 0 & , \text{if } \mathbf{p}(\mathbf{x}) >= \mathbf{p}(\mathbf{y}) \end{cases} \tag{3.47}$$

$$f_n(\mathbf{p}) := \sum_{1 \leq i \leq n} 2^{i-1} \tau(\mathbf{p}; \mathbf{x}, \mathbf{y}) \tag{3.48}$$

Where $\mathbf{p}(\mathbf{x})$ is the intensity of $\mathbf{p}$ at a point $\mathbf{x}$.

$$\mathbf{S} = (\mathbf{x}_1, ..., \mathbf{x}_n \mathbf{y}_1, ..., \mathbf{y}_n) \tag{3.49}$$

Then, using the orientation of patch (θ) and the corresponding rotation matrix $\mathbf{R}_\theta$ is found, it can be applied to S to steer it (then represented by $\mathbf{S}_\theta$).

$$S_\theta = R_\theta S \tag{3.50}$$

The steered BRIEF operator becomes [66]:

$$g_n(\mathbf{p}, \theta) := f_n(\mathbf{p}) | (\mathbf{x}_i, \mathbf{y}_i) \in \mathbf{S}_\theta \tag{3.51}$$

Then, ORB purposes a angular space discretization in $2\pi/30rad$ (twelve degrees) increments to construct a lookup table with BRIEF patterns to compute feature descriptor. If keypoints orientation (θ) is consistent across frames, the correct set of points $\mathbf{S}_\theta$ will be used to compute its descriptor.

One of the pleasing properties of BRIEF descriptor is that each bit feature has large variance and a mean value near to 0.5 but once it is oriented along keypoint direction, to give steered BRIEF, it loses this property and the means are shifted to a more distributed pattern. Thus, the oriented corner keypoints present a more uniform appearance to binary tests.

High variance makes a feature more discriminative, since it responds differentially to inputs. Another desirable property is to have the tests uncorrelated, since then each test will contribute to the result. ORB runs a greedy search among all possible binary tests to find the uncorrelated ones that have both high variance and means close to 0.5. This lead to a new approach known as rBRIEF (Rotation-Aware Brief).

As rBRIEF is a binary pattern, Locality Sensitive Hashing (LSH) is used for nearest

neighbor search in order to match descriptors.

In LSH, the points are stored in several hash tables and hashed in different buckets. Given several feature descriptors, its matching buckets are retrieved and its elements are compared using a brute force matching. The power of that technique lies in its ability to retrieve nearest neighbours with a high probability given enough hash tables. For binary features, the hash function is simply a subset of the signature bits: the buckets in the hash tables contain descriptors with a common sub-signature. The distance is the Hamming distance.

So, a multi-probe LSH improves the traditional LSH by looking at neighbouring buckets in which a query descriptor falls. While this could result in more matches to check, it actually allows for a lower number of tables (thus less RAM usage) and a longer sub-signature, therefore smaller buckets.

3.6.14 Edges detectors

Before introducing further concepts, it is mandatory to introduce edge detectors. Edge detection is one of the most useful operations to perform image processing. It helps on the reduction of the amount of pixels to process and maintains the structural aspect of the image.

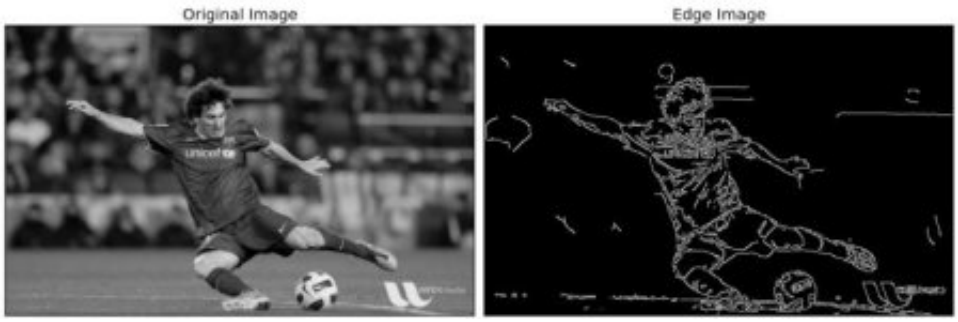


Figure 3.27: Edge detector.

The most known edge detectors are the Laplacian, Canny detector and the Sobel edge detector. In this work, only are implemented and available to perform the first two.

3.6.14.1 Canny edge detectors

The Canny Edge detector was developed by John F. Canny in 1986 [104] and aims to satisfy three main criteria:

- Low error rate: Meaning a good detection of only existent edges;
- Good localization: The distance between edge pixels detected and real edge pixels have to be minimized;
- Minimal response: Only one detector response per edge.

First the image must be filtered by a gaussian filter to reduce image's noise, usually with a 5x5 kernel matrix. Then, The intensity gradient filter must be found on the image applying a pair of convultion masks in x and y directions to find its first derivative in each axis over the image (Im) like:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * Im \tag{3.52}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * Im \tag{3.53}$$

And then compute the gradient strength and its direction with:

$$G_y = \sqrt{G_x^2 + G_y^2} \tag{3.54}$$

$$Angle(\theta) = arctan\left(\frac{G_y}{G_x}\right) \tag{3.55}$$

Gradient direction is always perpendicular to edges. It is rounded to one of four angles representing vertical, horizontal and two diagonal directions. After getting gradient magnitude and direction, a full scan of image is done to remove any unwanted pixels which may not constitute the edge and select the edges by the non-maximum suppression method. This is, at every pixel, pixel is checked if it is a local maximum in its neighbourhood in the direction of gradient. If so, it is considered for next stage, otherwise, it is ignored.

Finally, with the hysteresis thresholding, the remaining pixels are confirmed to be really edges or not. For this, we need two threshold values, minVal and maxVal. Any edges with intensity gradient more than maxVal are sure to be edges and those below minVal are sure to be non-edges, so discarded. Those who lie between these two thresholds are classified edges or non-edges based on their connectivity. If they are connected

to "sure-edge" pixels, they are considered to be part of edges. Otherwise, they are also discarded.

3.6.14.2 Laplacian edge detectors

Unlike Canny edge detector, the Laplacian edge detector uses only one kernel. It calculates second order derivatives in a single pass. Here's the kernel used for it:

$$L = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} * Im \tag{3.56}$$

Or, use either:

$$L = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} * Im \tag{3.57}$$

Or for a better approximation, a kernel matrix of 5x5 dimensions with 24 at the center and everything else is -1. Laplacians are computationally faster to calculate (only one kernel *vs* two kernels) and sometimes produce better results than Canny detector but it is not 100% perfect too. Since we are working with second order derivatives, the Laplacian edge detector is extremely sensitive to noise.

3.6.15 Hough Transform - lines and circles detector

Until now, all the presented methods lay mostly on edge/corner detectors because of its distinguishable shape/properties and simpler recognition across frames.

Although it is useful to the feature matching between camera views on this work, this won't solve all the detection problems since it doesn't suit to detect our desired target (a known dimensions pattern with active lighted balls). This is, since balls also have known shapes and relative disposition, we will also need a circle and ellipsis detector.

So, in 1962, Paul Hough purposed its own method to detect and recognize complex patterns, named as Hough Transform, then patented by IBM [67]. Initially thought to detect standard geometric shapes (initially only to lines), this was the base for other important improvements. Later, using the same principles, Richard Duda and Peter Hart extended it to detect curves, useful on circles and ellipsis detection [68] using simple mathematical representations/boundaries.

It is just applicable in binary images (not gray-scale images) after an edge extraction, where only the magnitude of local changes is known, which will leave only object's boundaries. Then, as object's boundaries must be on a binary format (pixels with zero or one values), there is often applied a threshold value to the edged image. The pixels which exceeds it are marked as "ones", otherwise, are marked as "zeros", resulting on the desired binary edged image of zeros and ones.

A circle can be mathematically parametrized in Cartesian coordinates by [70]:

$$(x - \mathbf{a})^2 + (y - \mathbf{b})^2 = \mathbf{r}^2 \tag{3.58}$$

Using only three parameters, its center coordinates at (**a**,**b**) and its radius **r**, a curve detection is applicable if there are just a few information about the location of a boundary, thus its shape can always be described as a parametric curve by the equation 3.58. This lead to its main advantage of being relatively unaffected by gaps, overlapping or noise in the curves [69] since it doesn't need all the boundary points of the circle [70].

Then, for each edge pixel, we can ask the question: "If it lays on a circle, what is the locus for the parameters of that circle?" (locus=place). The answer is, for fixed x and y, lets vary a, b and r within a right circular conic parameter space as seen of figure 3.28.

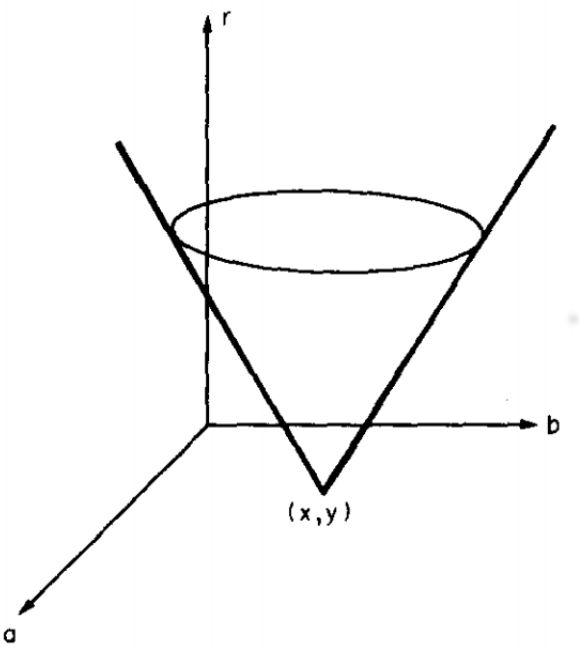


Figure 3.28: Conic parameter space [70].

If a set of edge pixels in the edged image are arranged on a circle with parameters a_0, b_0, and r_0, the resultant loci (location) of parameters for each such point will pass through the same point (a_0,b_0,r_0) in parameter space. Thus many such right circular cones will intersect at a common point [70].

3.6.16 Feature Matching

As said before, the detected features laid on a object of potential interest (OPI), are described/identified by its descriptor. This uniqueness or unequivocal identification makes possible its comparison with other features using feature matching. It is useful on 3D reconstruction, objects recognition, motion tracking, system calibrations, etc.

Most of the matchers are based on probabilistic FLANN (Fast Library for Approximate Nearest Neighbors) or brute-force techniques, comparing point's neighbourhood or a set of points with all the points in another set, using their descriptors to output the best or the k best features that match the compared one. There are several ways to compare descriptor, depending on the used detector:

- Norm L1 - taxicab/Manhattan distance;
- Norm L2 - euclidean distance;
- Hamming Distance;

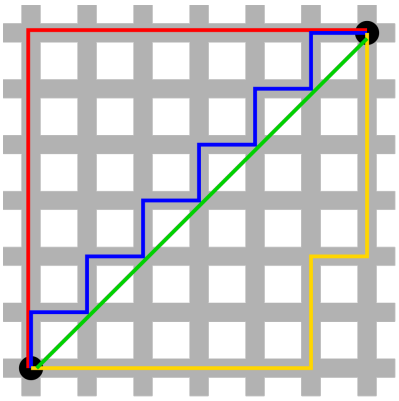


Figure 3.29: Distances [94].

Norm L1 , the taxicab or manhattan distance (also known as rectilinear distance or snake distance), represented by the paths on red, yellow and blue on the figure 3.29, is

the distance between two points results from the sum of the absolute differences of their Cartesian coordinates:

$$d(p,q) = ||\mathbf{p} - \mathbf{q}|| = |\mathbf{p}_x - \mathbf{q}_x| + |\mathbf{p}_y - \mathbf{q}_y| = \sum_{k=1}^{n} |\mathbf{p}_k - \mathbf{q}_k| \tag{3.59}$$

Norm L2, the euclidean distance, represented by the green path on figure 3.29, is the straight-line distance between two points. This may seem obviously faster but it isn't, since it uses square root of sums of square Cartesian coordinates differences to compute the euclidean distance between points (more expensive computationally than the sums used on Manhattan distance):

$$d(p,q) = \sqrt{(p_x - q_x)^2 + (p_2 - q_2)^2} \tag{3.60}$$

Hamming distance [95] is used to compute distances between two strings of equal length where the output is the number of positions at which the corresponding symbols are different, useful when features are described by strings.

3.7 Attitude/pose computation and ground-truth systems

Attitude of an object is its relative orientation to a reference coordinate system. Orientation presented on chapter 3.3.2 and translations presented on chapter 3.3.3 are now used to compute object's pose. Pose of a object is its relative attitude (often represented on euler or quaternions angular space) and its relative XYZ position (represented on system coordinates, usually metric or imperial) as seen on figure 3.30.

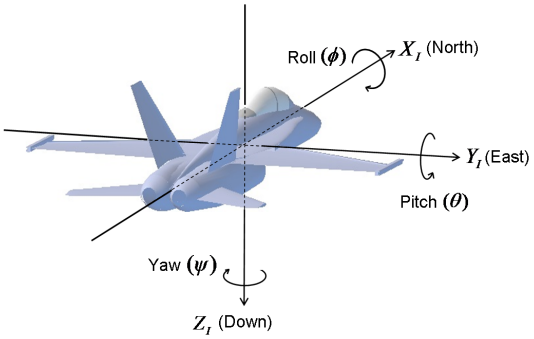


Figure 3.30: Plane attitude [101].

In the autonomous systems context, the ground-truth (GT) meaning refers to some reference measures/values with high accuracy and residual noise errors or even to simulated and perfect measures without any errors [96]. GT values can be provided by external sensors/systems whose accuracy is proven and known noise errors/uncertainty values.

GT systems are often integrated or compared with other systems and assumed to provide the true values, this is, the values that approximate the reality and which other sensors should obtain under the same conditions.

This work use the Pixhawk flight controller from 3DR [99] as the ground truth reference values which are already proven to have high accuracy measures due to well dimensioned Extended Kalman Filters (EKF) [97] and dual-sensor acquisition.

Pixhawk has two gyroscopes (16 bit ST Micro L3GD20H and MPU 6000), two accelerometers (14 bit ST Micro LSM303D and MPU 6000), one magnetometer (14 bit ST Micro LSM303D) and one barometer (MEAS MS5611) .

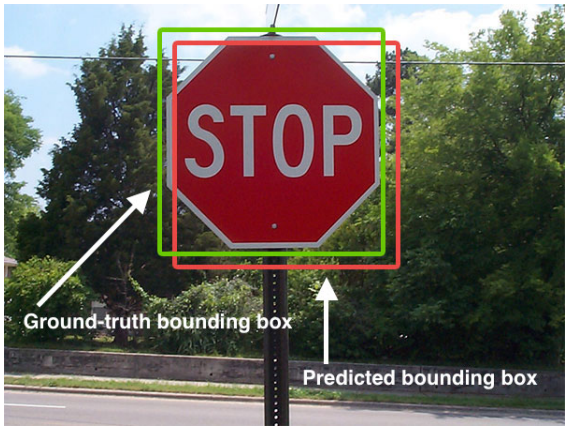


Figure 3.31: Ground truth and predicted state based on sensor characteristics [100].

EKF [97] is a high performance feedback filter suitable for discrete linear systems, proposed to solve the non-linear problem in practical applications like sensor measurements. It includes a state prediction step and a measurements correction step. It uses the iterative algorithm to obtain the optimal state estimation [98]. Non-linear systems like inertial, GPS, pressure sensors produce different outputs for the same input due to noise effects thus, it must be linearised first by the Taylor expansion [97] and only then used by Kalman filters to predict and update sensor measurements (figure 3.31).

In the context of this work, the inertial data from pixhawk sensors (giving instantaneous attitude and relative pose) is used to compare with the marker's attitude and

pose computed by our RGB-Vision application. Thus, a new marker frame was designed to attach the pixhawk and the active coloured balls.

The designed target allows the use of several mathematical principles such as planar homography to recover camera's baseline and easy attitude computation from the orthogonal matrix representation of rotations.

3.7.1 Orthogonal matrix representation of rotations

If we have a rigid body rotating about a fixed point $o \in \mathbb{E}^3$, then, the rotation can be expresse by three orthogonal vectors $r_1, r_2 and r_3$. The three vectors are simply unit vectors along the three principal axes x, y, z of the object frame relative to the world coordinate system as shown on figure 3.32. As the vectors belong to the 3D coordinate system from the world, they will be given by three values (x,y and z).

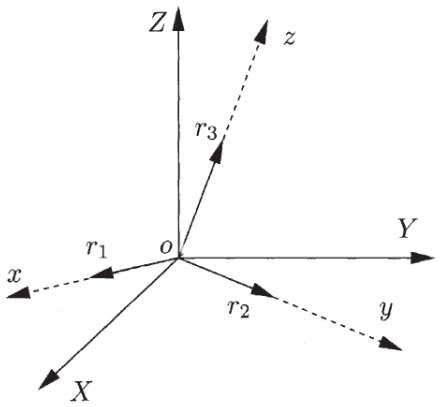


Figure 3.32: Orthogonal rotation matrix [106].

As the vectors are orthogonal, they can be used to form a orthogonal rotation matrix like:

$$R_{wc} \doteq [r1, r2, r3] \in \mathbb{R}^{3\times3} \tag{3.61}$$

From linear algebra we get that, a orthogonal matrix times its transpose equals to the identity matrix with the same dimensions and its inverse equals to its transpose matrix:

$$R_{wc}^T R_{wc} = R_{wc} R_{wc}^T = I \tag{3.62}$$

$$R_{wc}^T = R_{wc}^- 1 \tag{3.63}$$

Chapter 4

Calibration Toolbox

To achieve reliable attitude recover from artificial stereo vision it is mandatory to ensure correct system calibrations. Therefore, this calibration toolbox intends to complement the main goal of this work, the pose recover from stereo systems under real-time constraints. In order to allow its use under a wide variety of light conditions and environment fitting customization, the calibration toolbox provides easy-to-use tools with no user previous knowledge about calibration and allows the use of several kinds of customizable calibration patterns. It is also designed to work in real-time with instantaneous outputs using augmented-reality to give the user a better understanding about the calibration progress.

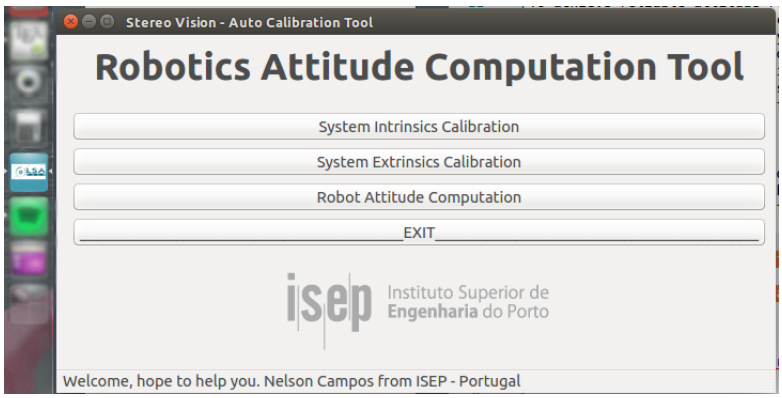


Figure 4.1: Toolbox menu.

The first two options in this toolbox (figure 4.1) are dedicated to the calibration of the stereo system. The third option is the attitude computation tool where the calibration parameters can be loaded or used from the previous steps.

4.1 Intrinsic Calibration Toolbox

The calibration of each individual camera lens and sensor parameters is known as the intrinsic calibration. As referred on chapter 3.2.1, this is the way to map world points in the image plane of each camera. This parameters are static, unless there are lens adjustments or displacements.

This toolbox uses the widely known chequerboard to recover camera parameters/-matrices and can also use the active marker presented on section 4.1.1 since both have known shapes and dimensions. Thus, it is necessary to inform the system about the chequerboard/target dimensions as illustrated on figure 4.2

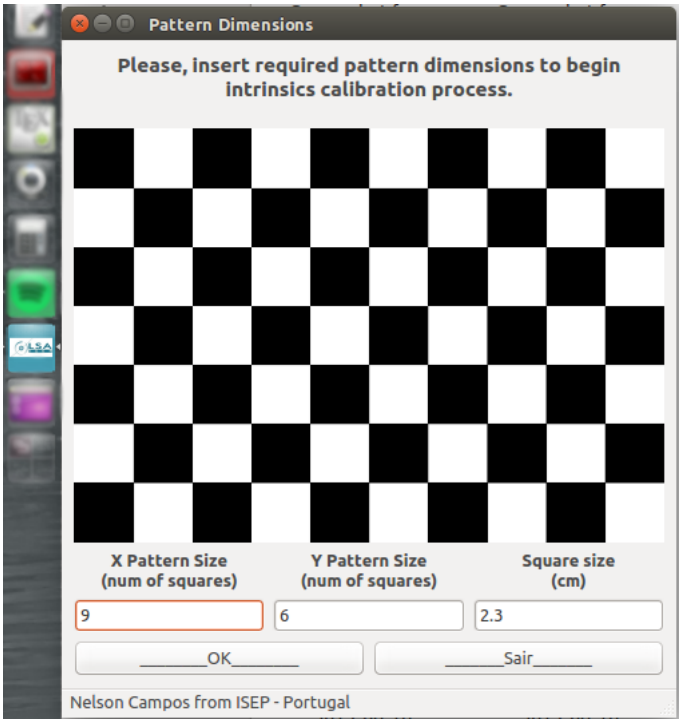


Figure 4.2: Chequerboard menu.

Known the target dimensions to detect, it is time to choose which camera the user wants to calibrate in the next step. In the camera choice menu, the user must name the camera and select between the available cameras as seen on figure 4.3.

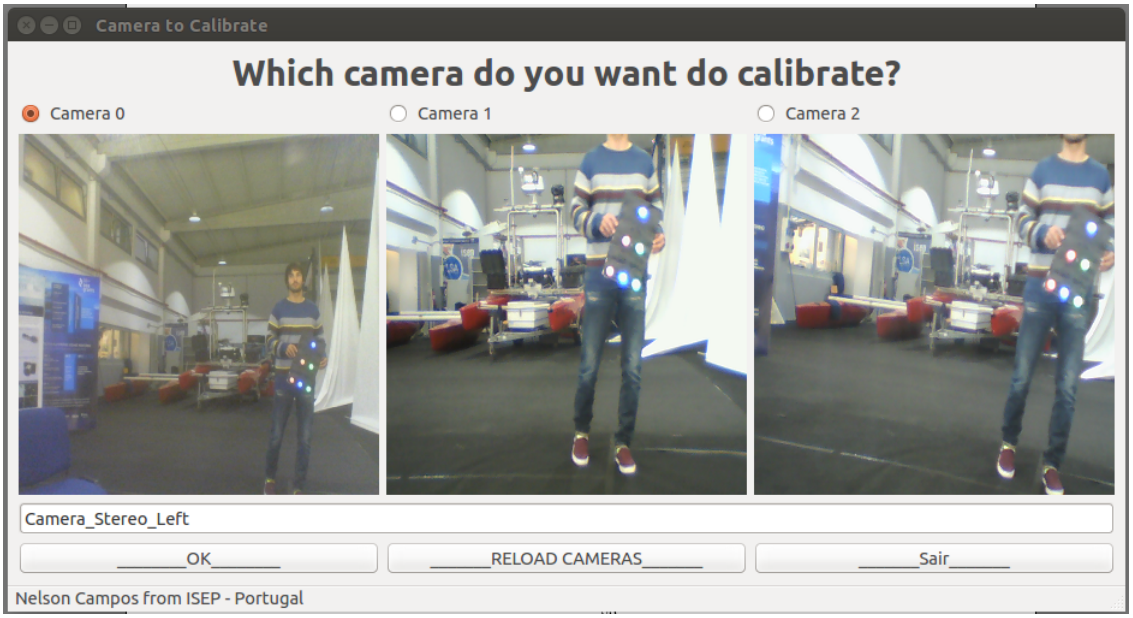


Figure 4.3: Camera choice menu.

The main menu of this calibration tool, presented on figure 4.4, allows the real-time detection and its representation using augmented reality which allows the user to follow instantaneously the calibration process.

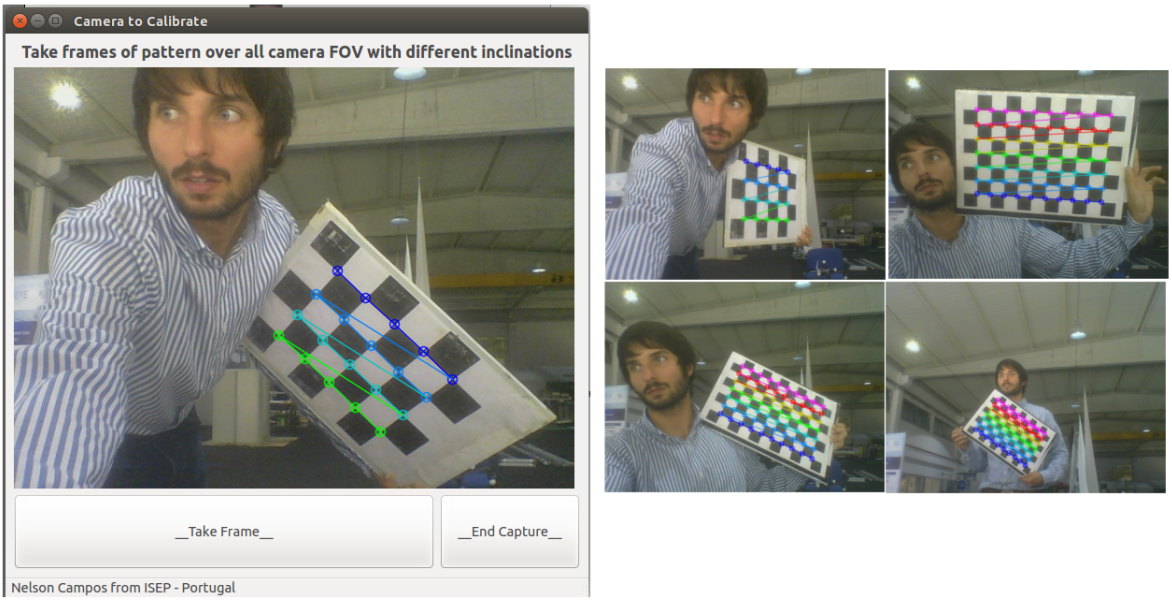


Figure 4.4: Chequerboard real-time detection.

To allow real-time detection, processing and augmented reality representation, it was mandatory to implement the background extration using the adaptive image mean filtering method (presented on chapter 3.6.4).

The calibration process will only take into account the frames taken by the user but it can also be done autonomously, taking frames of the chequerboard and analysing it only when successfully detected. Once the capture is complete, the user can end the capture process and the camera calibration outputs are displayed in a new window. It presents the camera matrix, the distortion coefficients matrix and the re-projection error root mean square (RMS) value in pixels (in good calibration results, its value should be between 0.1 and 1.0 pixels). This calibration process also outputs the rotation and translation vectors of each taken pattern view, usefull to present the user the relative position and orientation (each individual extrinsic parameters) of the pattern in a camera-centric or a pattern-centric view as already available by other tools like cameraCalibrator 2.1.2.1 and presented on figure 2.5. This output could also be available on this tool but it need some improvements and tests first.

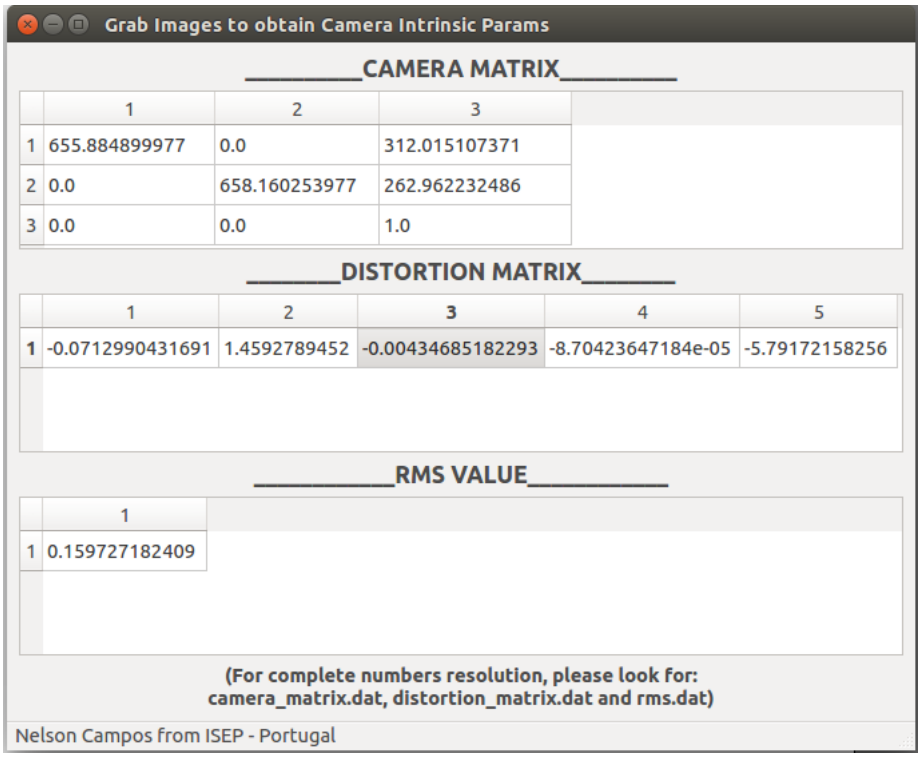


Figure 4.5: Intrinsics calibration output.

The re-projection error is computed projecting the computed 3D chequerboard points into the image plane using the final set of calibration parameters (camera matrix, distortion coefficients, rotation vectors and translation vectors) and comparing the known position of the corners (since the chequerboard or other patterns have known dimensions and shape).

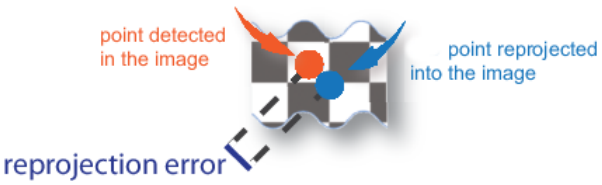


Figure 4.6: Reprojection errors [107].

The calibration matrices are also saved in a new folder with the camera name and with the taken frames with and without the detected patterns drawn and the non-detected patterns. The backup of the taken frames are useful to allow the calibration comparison with other tools and even its manual exclusion by the user to prevents bad detections which would lead to poor calibration results.

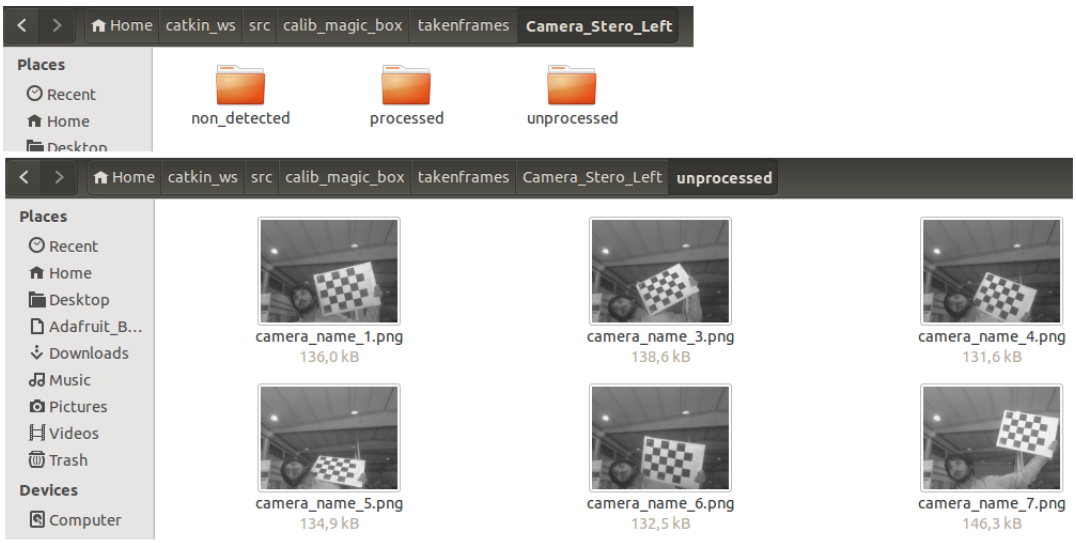


Figure 4.7: Images backup.

4.1.1 Active coloured marker

The calibration process using chequerboards is still not optimized in order to allow its use in systems with frame rates higher than 30FPS, since the detection of the chequerboard is computationally expensive and ends up limiting its raw application in high-definition cameras or with high frame rates. In this work, there were developed and implemented some workarounds such as fast background extraction algorithms and ROI extraction/examination algorithms which allows the use of high-definition cameras and higher acquisition frame rates. In parallel, there were developed some algorithms to allow the detection of different targets. Therefore, there was designed a clever target with active markers and known dimensions. Active marker stands for self-illuminated points or balls instead of the retro-reflective markers used by other systems (2.4.2.2) where its detections depends on cameras with external light emitters making them not suitable for low-cost applications or high range scenes where its precision decreases rapidly with the distance to the cameras.

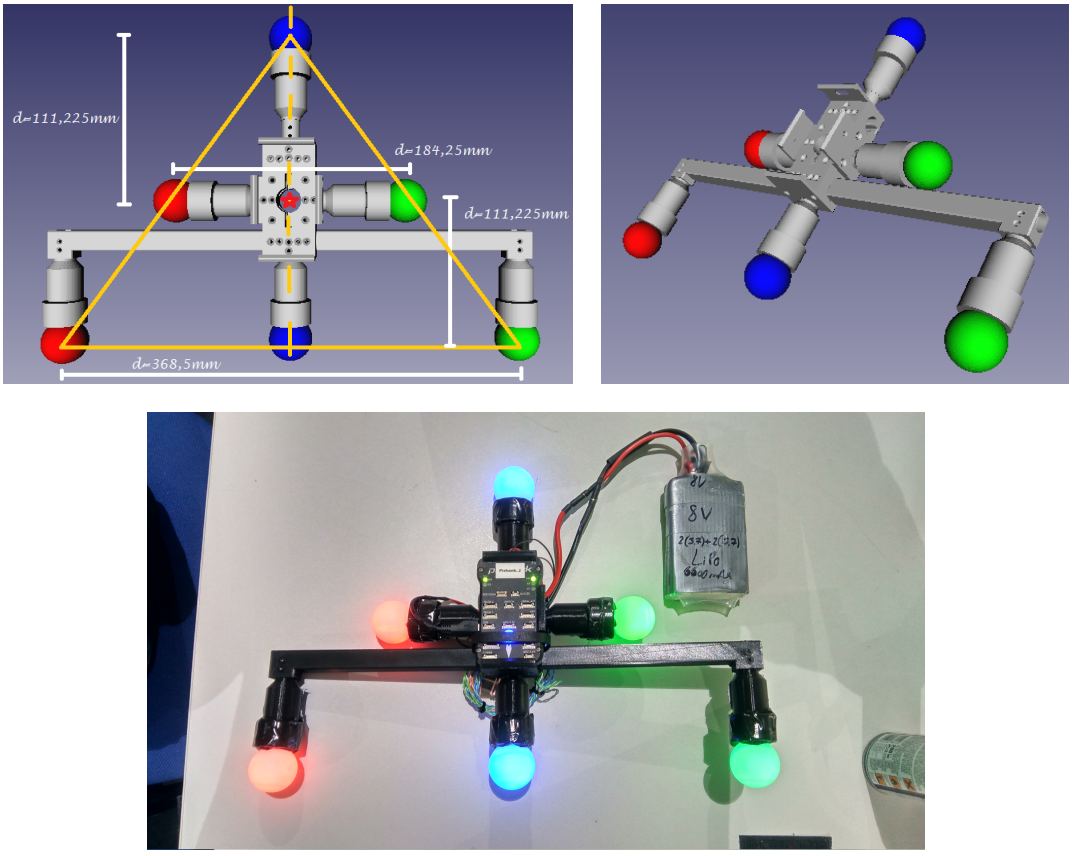


Figure 4.8: Active marker.

Chequerboards are also not very practical for underwater applications since the positioning of the pattern can be difficult and almost undetectable due to water dirtiness, leading to bad pattern detection performances.

As referred before, the presence of high quantities of information in the image scene is also a known hindrance in pattern detection. Thus, the possibility of calibrate each camera and even the stereo system using a single active marker bring multiple advantages:

- Easy integration on underwater applications;
- Easy coupling on moving underwater robots;
- Faster background extraction algorithms due to easy detection of active-markers and outliers exclusion;
- More reliable detections under super-structured (lots of features available) or not structured environments at all;
- Outdoor applications with low noise interference;
- Faster tracking algorithms.

The designed marker, on FreeCAD software, has known dimensions and is easily customizable in shape and color since its parts can be changed, removed or upgraded with simple screws. It also allows the coupling of a pixhawk, a autopilot with a inertial measurement unit (IMU), for data comparison or fusion with the artificial vision attitude computation.

The main advantages of this specific marker are:

- A planar surface can be extracted, since all the markers lay on the same plane (coplanar);
- Coplanar markers decrease computational requirements upon detection and tracking;
- Customizable pattern (color and shape);
- Cancellation of position and orientation ambiguities since each ball has known distances and distinct colors;
- Easy sensor's integration;

- Light trigger to sync its maximum luminance with the camera shutter;
- Strong frame and core for easy coupling with mobile robots;
- Wireless data capture and pose information acquisition from the pixhawk (dry applications).

There are also some problems with this target. Its dimensions are known but there are existence of tinny errors on the positioning of the lights, since they were placed by hand and the dimensions of the markers are much bigger than the corners detected on checkerboards for example. Checkerboards are printed on planar surfaces and its dimensions tend to be preciser than the active coloured ones, even there could exist some warping on planar surface. This can lead to errors in calibration process and, therefore, in pose computation.

4.2 Extrinsic Calibration Toolbox

Stereo vision systems must also be calibrated as a whole to relate points in the several cameras and recover depth from the combination of single cameras. As mentioned before, the extrinsic parameters of each camera place them in a coordinate system, even if its origin is placed in a camera to simplify mathematical operations and reduce computational costs.

Therefore, the camera's positions and orientations from the stereo system can be calibrated face to each other using the "Extrinsic Calibration Toolbox" developed along this work.

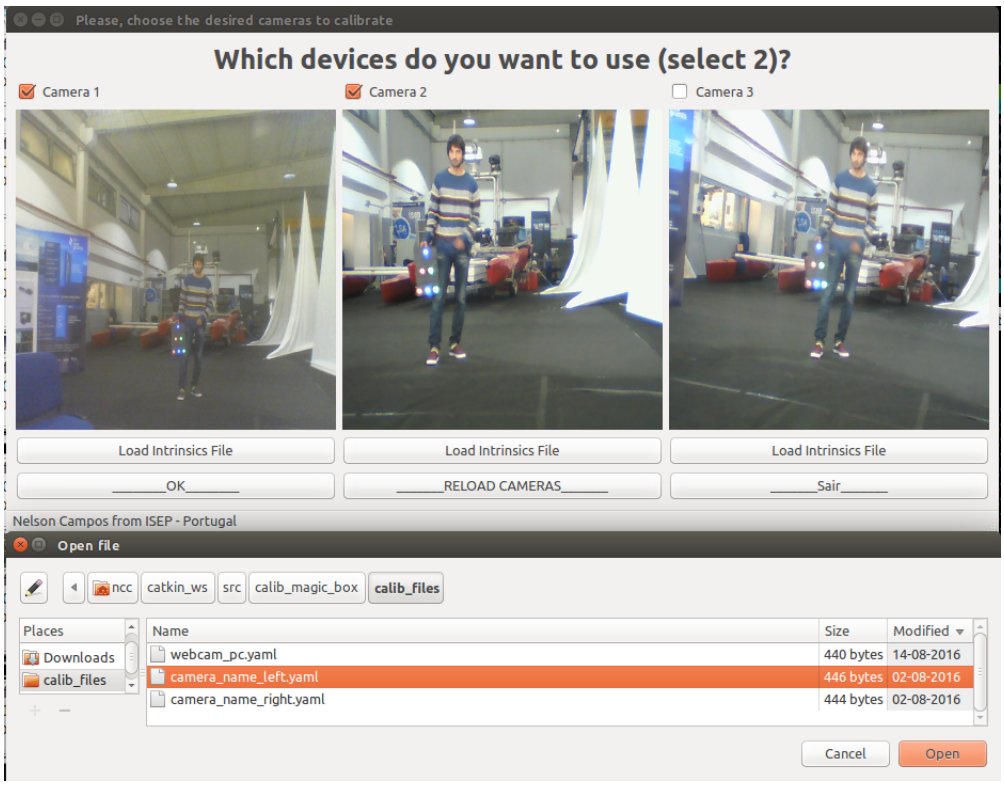


Figure 4.9: Camera choice menu with load of yaml files.

First, the user must select two cameras and load its intrinsic parameters from "yaml" files (figure 4.9), a human friendly data serialization, standard format for all programming languages. This yaml files can be exported by the first developed tool, the "Intrinsic Calibration Toolbox" (chapter 4.1), allowing the correct 3D world point mapping into a 2D undistorted image plane on each camera.

After selected the desired cameras to calibrate, user can view the selected cameras but with no stereo detections yet. Thus, environment and target configurations and calibrations must be done to ensure correct and reliable target detections. One of the main steps upon the environment calibration is, again, the "Background calibration" (figure 4.10) presented on chapter 3.6.4. This step is not imperative but is important to exclude non-important information which would lead to delays and unnecessary processing costs. In this case, we know that the information we need is always in the foreground because it will use the active marker to calibrate the stereo pair and not the information available on the environment (which could even be absent in some cases). This step is mainly important to the next step/tab of the extrinsics calibration, the detector calibration.

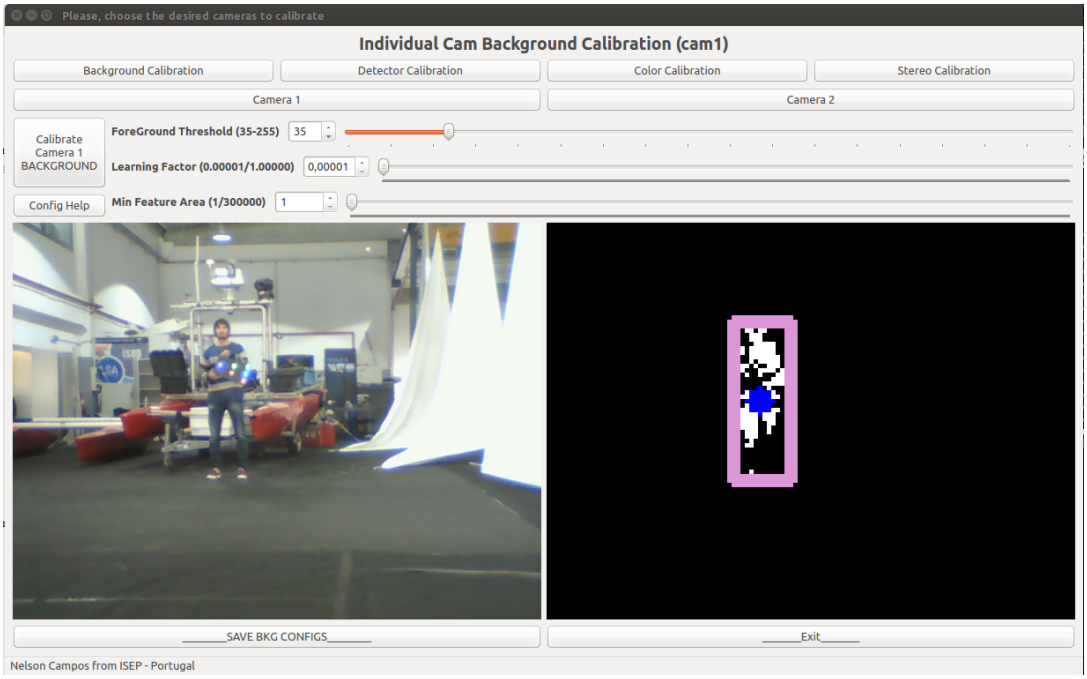


Figure 4.10: Background calibration tab.

Within detector calibration tab, if there is applied a background extraction from the previous step, there are created regions of interest (ROI) which contains only the foreground objects. As referred before, foreground objects are all the moving objects and background model is constantly updated. This means that static objects are, after some time, considered as background objects and, therefore, excluded of the detection analysis.

The detected ROIs resulting from the foreground extration are marked in real-time

over the presented image and the target detection is made only over the foreground objects. This allows the user to adapt in real-time the feature extraction to fit its requirements, improve detection quality as seen on figure 4.11(a) and better adjust the detector to ensure the correct detections and exclusion of outlier detections 4.11(b).

To provide wide target detection and environment fitting, the user can filter foreground objects by its size and sensitivity. This is, filter small or big objects and objects by area with low or high frequency movements like passing cloud shadows or trees respectively using intensity threshold filters. Then, since we need to apply edge detectors based on image first or second order derivatives (as presented on chapter 3.6.14.2), which are susceptible to image noise, the user can apply customizable blur filters to the image in order to filter its noise and reduce false target detections.

Due to known characteristics of our active coloured target, user can then adjust threshold values over the "value" or luminance of the image on the "Target detector" tab, in order to highlight only the coloured markers and exclude low brightness objects (figure 4.11(a)).

To simplify image information even more, the coloured markers can be reduced to its edges or contours applying one of the several edge detectors available as presented on chapter 3.6.14 and seen on figure 4.11(b).

Again, to reduce image noise, the edge detector output can be smoothed to obtain even better results upon the main step of the target detector, the circle detector. Purposed initially by Paul Hough in 1962 and improved later to fit other applications (presented on chapter 3.6.15), is used as circle detector in order to extract the circles formed by the coloured circular markers present on the target.

Figure 4.11: Detection calibration tab. Luminance thresholding and circle detection over object's edges.

If there were no ROI, the detection process might not be possible to perform un-

der real-time constraints due to huge amount of information to process since Hough transform upon circles detection can be computationally heavy.

The next step of the extrinsics calibration is the color calibration since our target has known colors, as presented on chapter 4.1.1. Here, user can isolate red, green and blue balls (figure 4.12) using different color spaces (chapter 3.6.6).

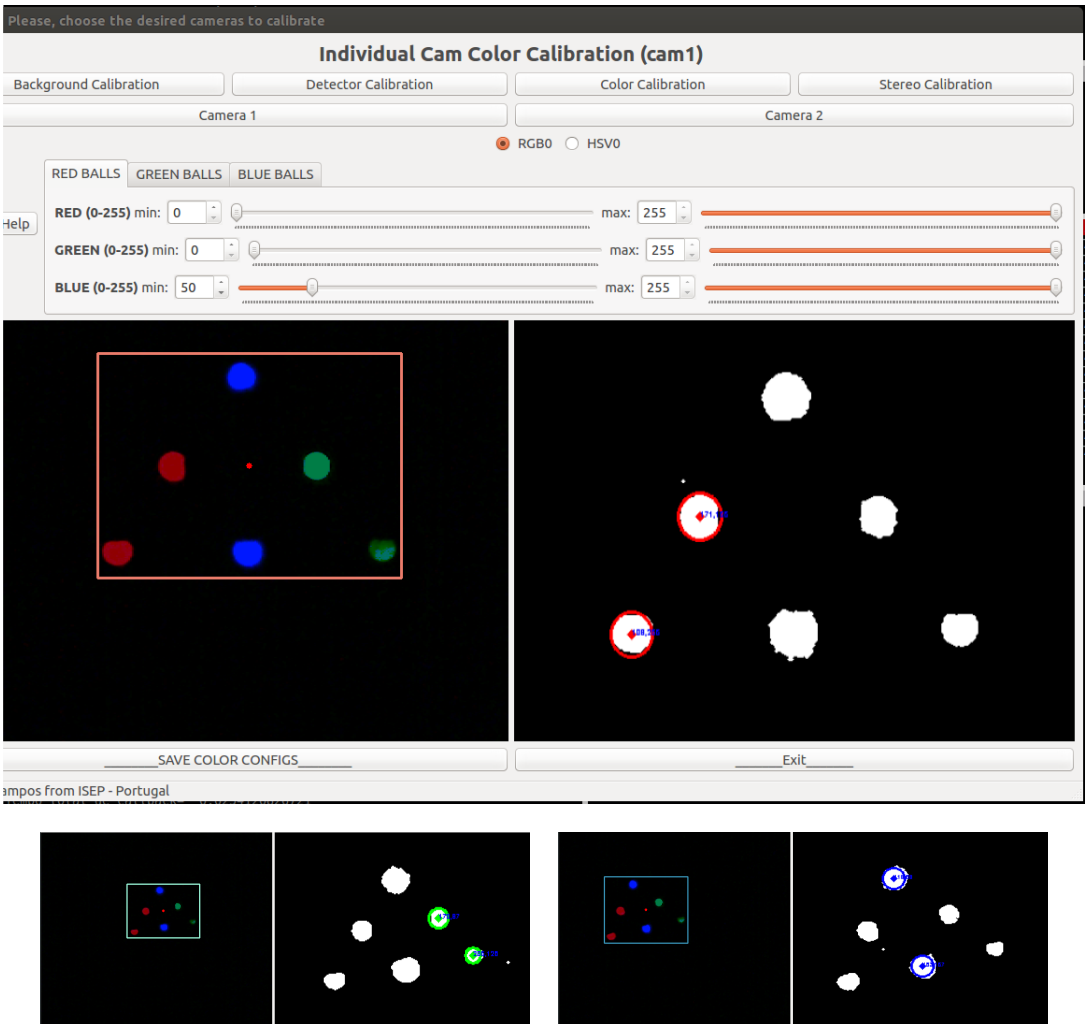


Figure 4.12: Color calibration tab.

With the isolated color markers, it can then be "connected" to form a target plan surface and vectors, needed to perform stereo calibration and recover extrinsic parameters using planar homography (3.5.2) and the orthogonal matrix representation of rotations (presented on chapter 3.7.1). The "Calibrate System" button allows the toolbox to enter/exit the calibration mode, collect data to perform the planar homography and

compute the best homography matrix that best relates both cameras. The best homography matrix is chosen by reprojecting the detected markers again into the world and, every time there is a new homography matrix which minimizes the reprojection error (as presented on figure 4.6) of all the markers in the world, it is saved and set as the best homography matrix for the system.

All the previous configuration steps are imperative to achieve get a correct extrinsic calibration (figure 4.13) and pose computation, the main goal of this work. Thus, all the configured parameters are then saved into a yaml config file.

Figure 4.13: Stereo calibration tab.

4.2.1 Stereo cameras frame-holder

In order to proceed to the stereo calibration and obtain reliable results, the cameras must be static and don't be moved face to each other, either in position/location or in orientation. The camera's baseline (the distance between cameras) must remain unchanged along the stereo calibration process. Every time there's a slight misalignment or relative movement between them, the extrinsics calibration must to be done from the beginning.

Figure 4.14: Camera's frame.

Thus, there was developed a frame, also on FreeCAD, to stabilize the camera's baseline and relative orientation between them (figure 4.14). This frame allows the adjustment of cameras distance and relative orientation with a simple screw untight.

Chapter 5

Attitude Computation System

The pose computation system is, visually, simultaneously the most simple and the most complete and complex tool from the three developed ones on this work. The algorithm, environment configuration and filter choices made upon intrinsics and extrinsics calibration toolboxes can be loaded to be applied now on detection, tracking and pose computation of the visual target.

The target structure must also be passed on that configuration file with its real dimensions, marker relative location and colors respectively.

The synchronization between frame's acquisition from both cameras must be done before the input in the toolbox since it will discard frames with a time drift higher than a adjustable delta, usually, no more than a period between frames ($TimeFrameCam1 - TimeFrameCam2 <= \frac{1}{framerate}$).

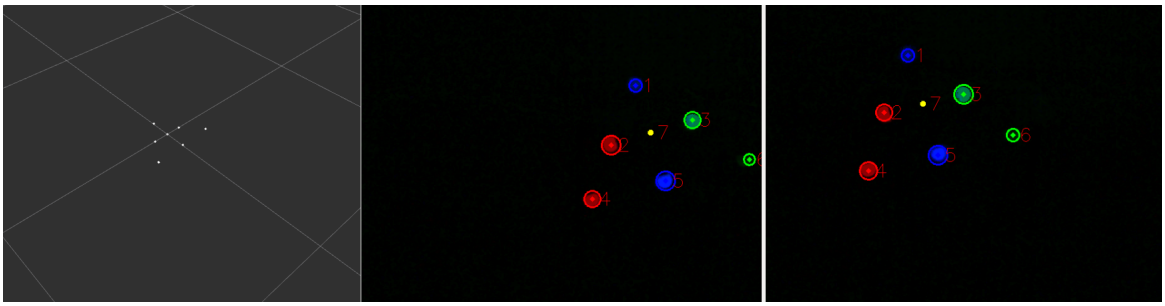


Figure 5.1: 3D point triangulation from computed homography.

The target attitude or orientation computation results from the principles of orthogonal vectors for creation of a rotation matrix, presented on chapter 6.5.2.

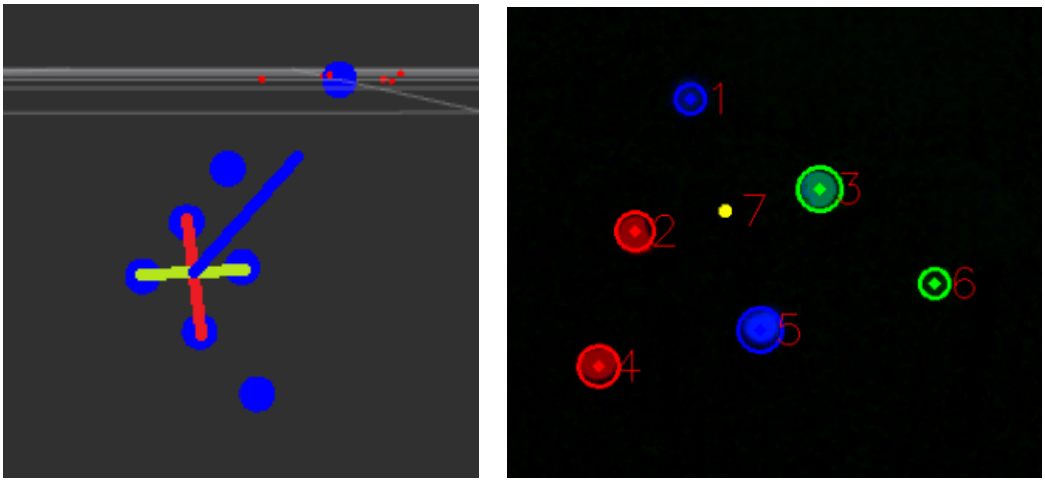


Figure 5.2: Vectors from target and markers enumeration.

Three vectors must be extracted from the target, a rigid body, where all its particles conserve relative distances. Each one of those vectors will represent the target rotation along the x, y and z axes respectively.

Vector can be described by a direction and a magnitude which can be extracted using linear algebra and analytic geometry principles using the subtraction of two points (a vector $\vec{AB}$ can be formed by the subtraction of two points as $B - A$).

So, to form the vector v_x in the orthogonal rotation matrix, it is used the vector formed by the balls 3 and 2 (3-2), making x axis positive in the $2- > 3$ direction. To compute the v_y vector, it is used the balls 5 and 1 (5-1), making the y axis positive in the direction $1- > 5$. The remaining v_z vector to represent the z axis is formed by the cross product of the previous two, resulting in a new vector, orthogonal to them, with direction respecting the right-hand rule, this is, positive direction from cameras towards the target.

Then, the rotation matrix can be built as:

$$R_{target} = \begin{bmatrix} v_{x1} & v_{y1} & v_{z1} \\ v_{x2} & v_{y2} & v_{z2} \\ v_{x3} & v_{y3} & v_{z3} \end{bmatrix} \tag{5.1}$$

Any further concepts about the attitude and pose computation will also be deeply explained on chapter 6, the project implementation.

With the computed homography on the extrinsics calibration toolbox or loaded in a yaml format, the toolbox is fully ready to use calibrated cameras and able to start the

pose computation process as explained before and displayed on figure 5.2(a). The user can see, in real-time on the toolbox, the detection and match between both cameras as well as balls enumeration.

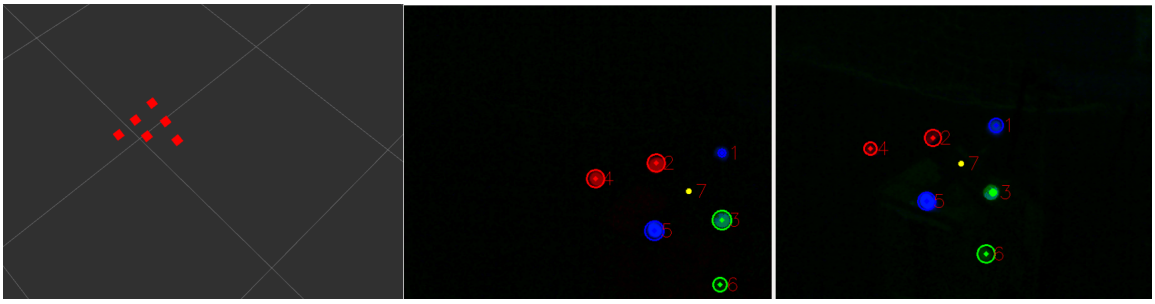


Figure 5.3: 3D point attitude from triangulation.

In background, upon a successful target detection in both cameras, the toolbox starts publishing three new topics:

- Origin **point** of the coordinate system formed by the cameras upon homography computation and the target triangulated points (**pointcloud2**) for that homography (which remain static if there are no new calibration) (figure 5.2(a));
- 3D **pointcloud2** resulting from the triangulation of target points (figure 5.3);
- 3D **pose**, with position on the target centroid and orientation in quaternions in the world reference frame;

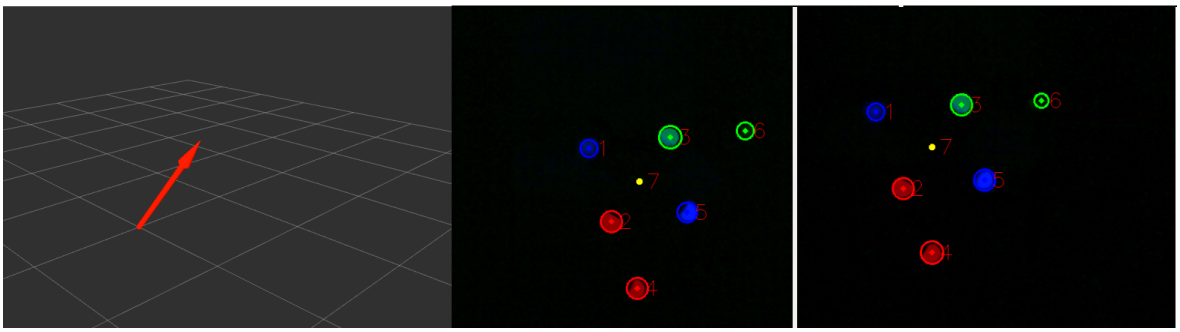


Figure 5.4: 3D pose retrieved from the target.

Chapter 6

Project implementation

Although intuitive and simple to use, the calibration toolbox and the attitude computation tool apply a number of complex and computationally heavy algorithms. Thus, the visual information to be treated would have to be minimized as much as possible to allow it to operate under real time constraints, give user instantaneous outputs about the detection and retrieve useful information to control tasks.

6.1 Background extraction

The very first step to ensure information reduction upon processing algorithms is the background extraction. This is, collect only all the objects belonging to the foreground. Foreground objects are all the moving objects, even if its movement is residual. Thus, initially, there are taken some scene samples from the FOV of each camera in which, the visual target should not appear to build a solid initial model of the foreground, as presented on chapter 3.6.4.

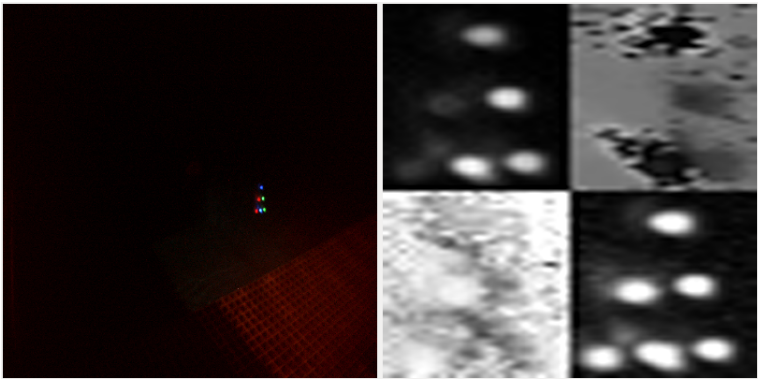


Figure 6.1: Studies about the several channels and color spaces.

Figure 6.2: Edge detector studies using the value channel from the HSV color space.

To speed up the background modelling, the images are resized to low resolution and coarse analysis since there is no need to preserve tinny details on this step. Then, blur filters are applied to the images to prevent image noise.

Now we are facing one of the main discoveries along this work (figures 6.1 and 6.2). As our target has is own light source, we can use this to our advantage. Lights are clearly seen on the "value" channel from the HSV color space so, the background model can be built using it, instead of using and storing all the channels from the RGB or HSV color spaces.

If there is a background model already, the "value" channel from the new frames will be processed, filtered by intensity and weighted by the learning factor defined by the user. The weighted frame is then used to update the background model. Then, the updated background model is superimposed to the actual value channel of the frame and the result is the foreground mask or foreground objects.

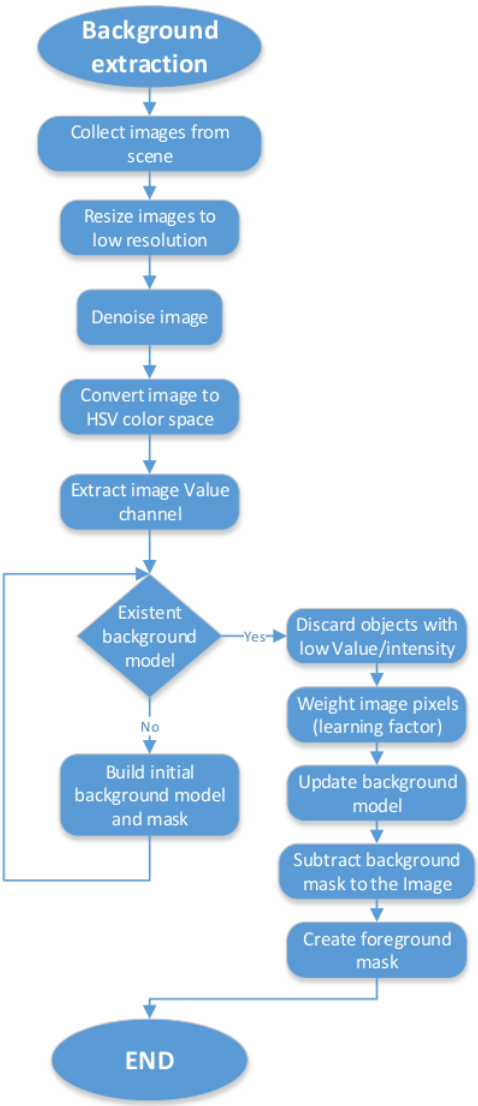


Figure 6.3: Background extraction function.

Upon the entering of the visual target on the FOV of the cameras, it will be easily highlighted from the rest of the image. The learning factor will then dictate if the target remains on the foreground or considered as background after some time static since the background is being updated constantly.

6.2 Region of Interest

Once extracted foreground objects, its analysis can begin. User can choose to straight discard small size objects (often caused by noise in the image) and low "value" pixels (low probability of belong to the foreground), leading to even more data reduction to process and find features later.

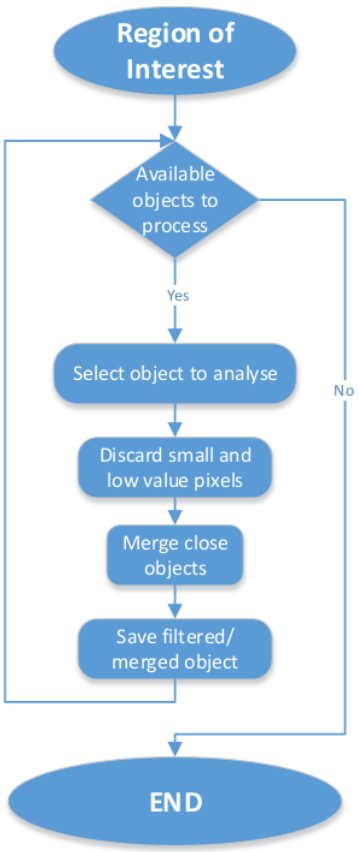


Figure 6.4: Region of interest extraction function.

Due to several causes such as occlusions, shadows or luminance differences, objects can be separated in different ROIs. Thus, the user can define distance thresholds to merge close objects which reduce the number of iterations when performing features detection. The pixels and ROIs that doesn't meet the imposed requirements, such as area or distance, are discarded.

6.3 Features Extraction

The feature extraction from images is, usually, the most heavy task on computer vision since there are applied algorithms that often go pixel by pixel and execute complex math to enhance "simple" properties which could be trivial and immediate when performed by the human eye and brain. Nevertheless, each image can contain millions of pixels. Is right here where the need of information reduction on the image becomes crucial and evident.

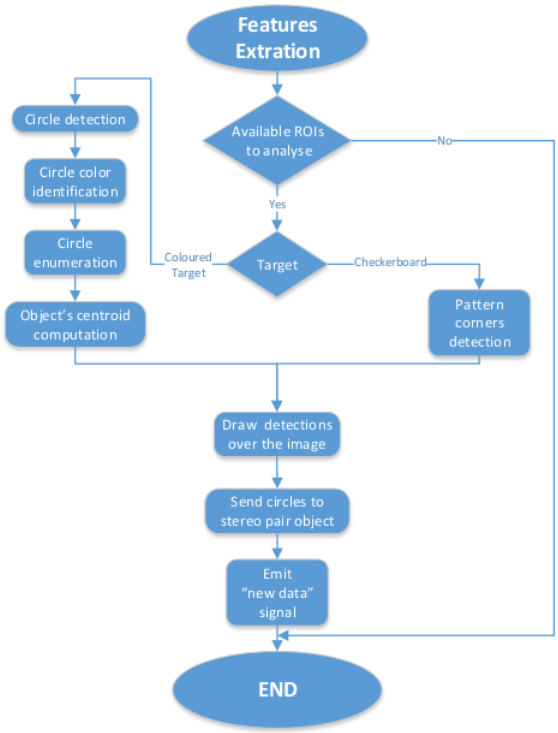


Figure 6.5: Features extraction function.

In this work, for feature extraction, depending on user's calibration target choice, there are applied different principles and algorithms since there are evident differences between them. User can choose to work either with a standard checkerboard or with a customized and active light pattern. The detected features are then drawn over the image in real-time to allow user's better understanding along calibration or attitude computation procedures and sent to a stereo pair object which contains the most actual feature detections and original images. Once there is new data available on the stereo pair object, it is emitted a signal to trigger the stereo computation if the data from both

cameras is coherent and not drifted.

6.3.1 Chequerboard Detection

If it is used a checkerboard, the algorithm will search for pattern corners using corner detectors as explained on chapters 3.6.8 and 3.6.9. Checkerboar patterns present several features laying on the same plane and with relative locations well known, useful to model the way as the camera sees the world, the distortion and focus imposed by the lens (intrinsic parameters), to relate several camera's image points and recover depth from 2D images (extrinsic parameters).

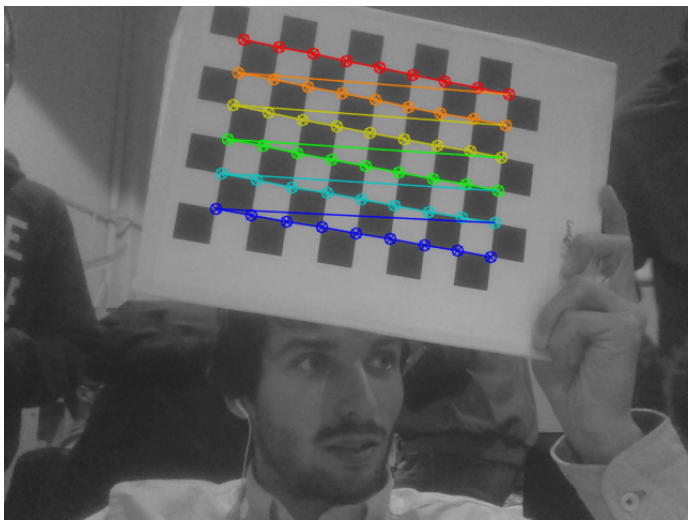


Figure 6.6: Checkerboard detection.

6.3.2 Coloured Target Detection

Otherwise, if it is used a customized pattern, the algorithm will search for circles using the selected algorithms by the user on the toolbox (presented on chapter 3.6.15) and filter them by color. The filtered circles are then associated and identified to compute pattern's centroid as seen on chapter 6.5.1.

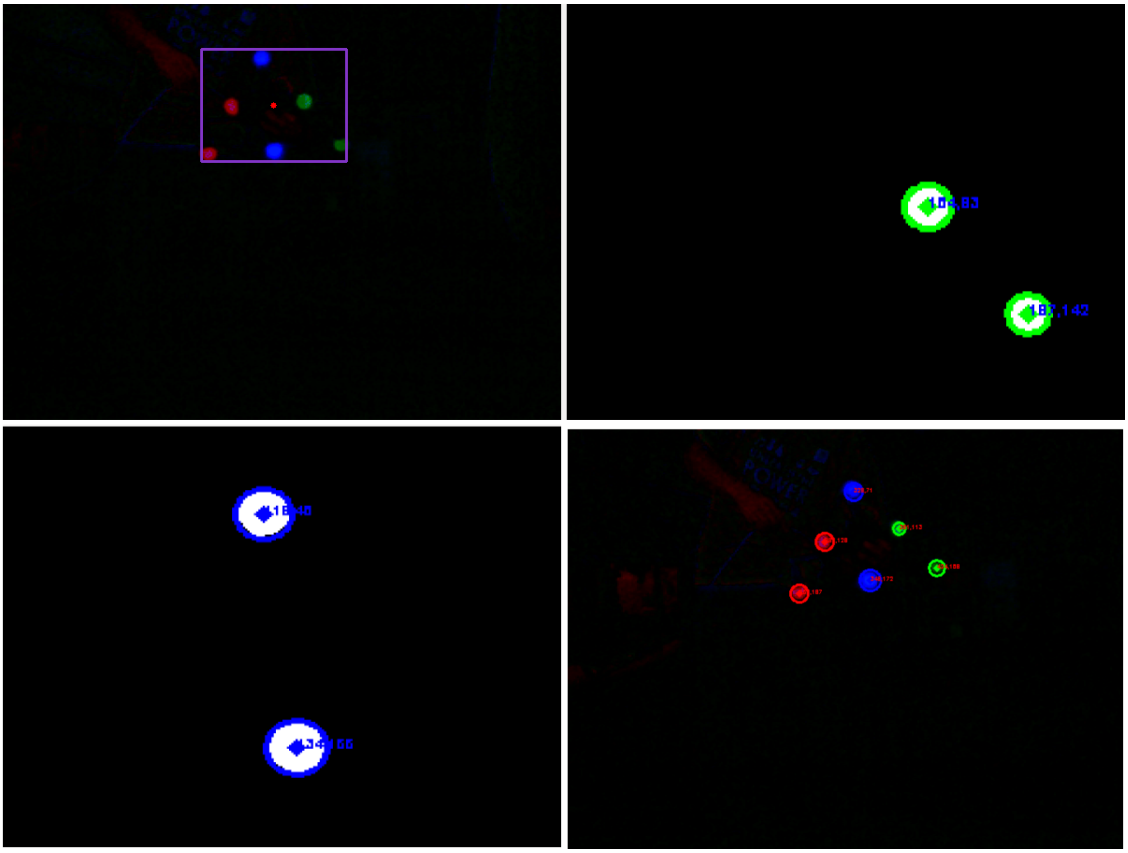


Figure 6.7: Coloured circles detection and filter.

6.4 Stereo Computation

When emitted the "new data" signal, it tells the system that one of the cameras has sent new features to the stereo pair object. If the timestamp difference between the features from both cameras doesn't exceed a specified value (by default the period time between frame acquisition), the information on the stereo object is considered and therefore processed, otherwise it is discarded.

Then, the loaded intrinsic parameters are used for the image distortion removal. The stereo computation will be done only, and only if, the stereo system is already calibrated. If not, firstly, the system must be calibrated using the chosen calibration target where is computed the homography matrix which is then decomposed to get the relative rotation and translation between cameras, needed for the stereo triangulation of points in both cameras.

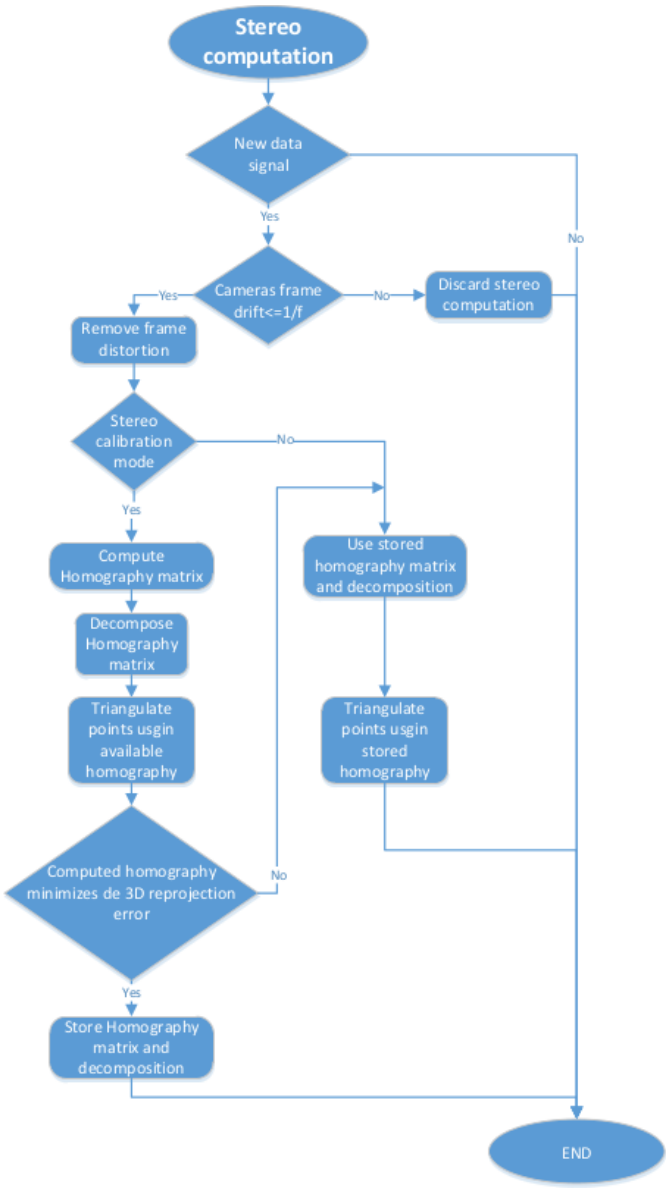


Figure 6.8: Stereo computation function.

As referred, the detection can be sensitive to error and the homography matrix can vary for diverse reasons so, the best homography matrix must be chosen between the several ones computed upon new frames from both cameras. The "best homography matrix" which better relates both cameras is chosen by reprojecting the detected markers again into the world and, every time there is a new homography matrix which minimizes the reprojection error of all the markers in the world even better than the stored one

(as presented on figure 4.6), it is stored. The process is repeated until the user changes the working mode to "calibrated".

6.5 3D attitude and pose computation

The 3D triangulation and pose computation is only possible to perform if there are calibrated cameras or undistorted images of the same scene from different perspectives or view-points and the rotation and translation relations between images are known.

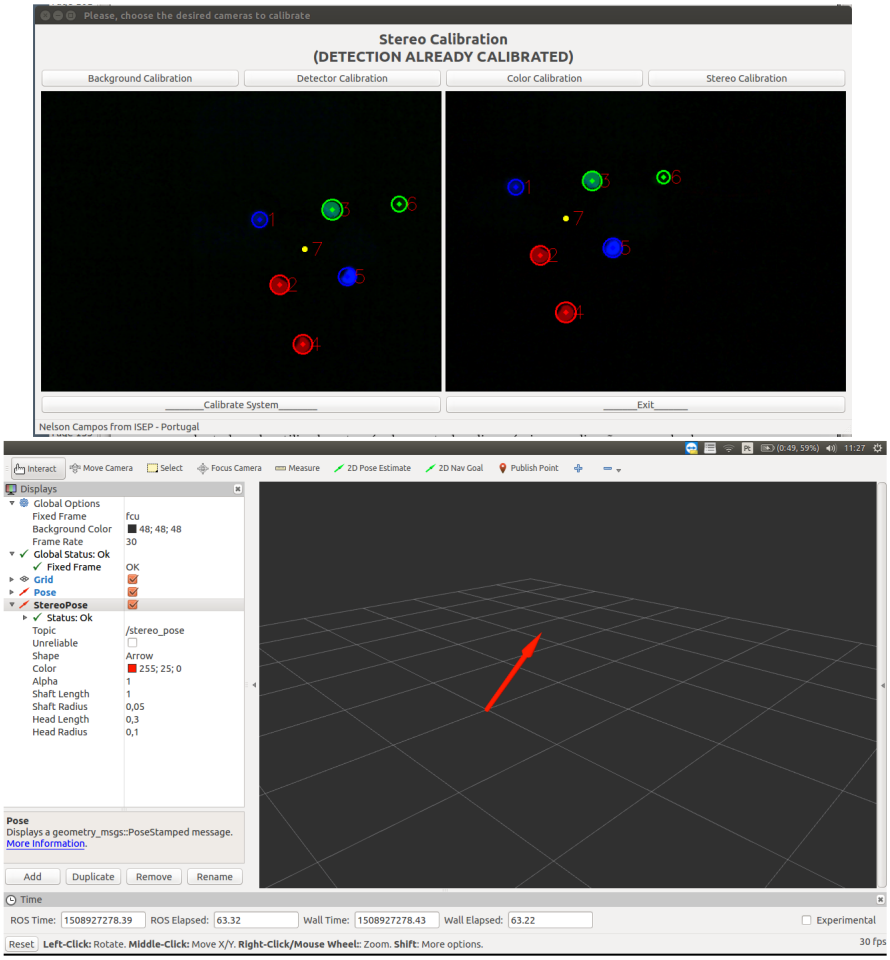


Figure 6.9: Attitude and pose computation output.

If we are handling with a static scene and static cameras, the timestamp difference between images is not important since the error is not cumulative in this kind of systems and the output of the application does not depend on the time but of correct detection

of the target. Thus, pose computation can't be done until there is a calibrated system (intrinsics and extrinsics).

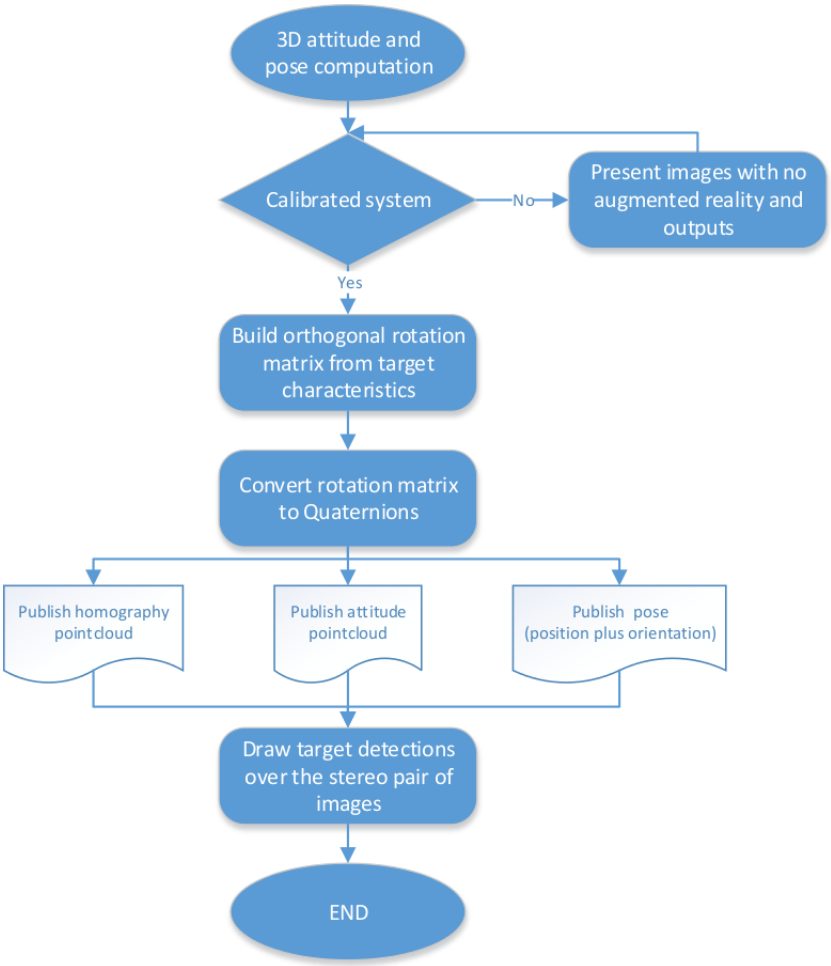


Figure 6.10: Attitude and pose computation function.

If the system is already calibrated and the target dimensions are known, it is possible to compute its rotation matrix using the algebraic principles already presented on chapter 3.7.1 and the assumptions on chapters 6.5.1 and 6.5.2.

6.5.1 Target identification

To recover attitude from the the visual target, it was firstly developed an algorithm to disambiguate the balls location using the known properties of the target, presented on chapter 4.1.1.

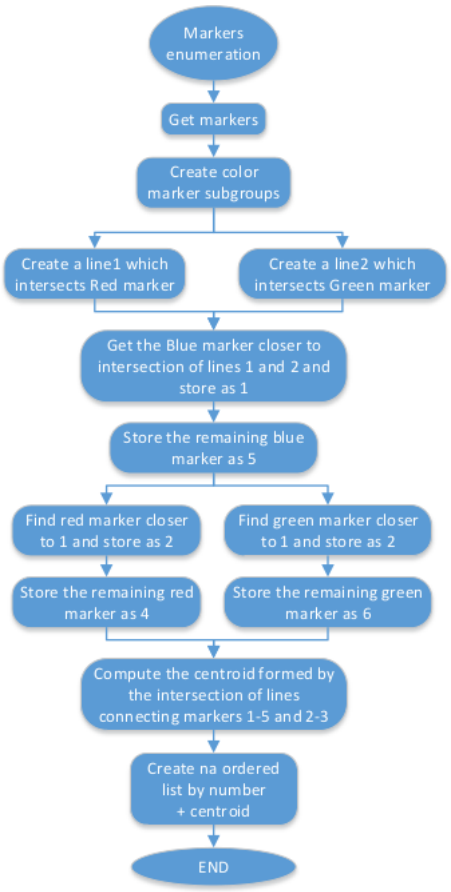


Figure 6.11: Markers enumeration algorithm.

The target identification and consequent markers enumeration reasoning is performed in real-time, presented on figure 6.11 and its result displayed on figure 6.12.

First the whole group of detected balls is divided in three subgroups (representing the red, green, and blue color balls from the RGB color space). At this point, all the misdetections have been excluded and the input to the target identification is a group of six balls (two of each color).

From the three subgroups, we draw two imaginary lines which intersect the red balls (line1) and green balls (line2) and its intersection will be used to find the blue ball which is closer to it and numbered as 1. The remaining blue ball is numbered as 5.

The red and green balls which are closer to the ball 1 are numbered as 2 and 3 respectively. The remaining red and green balls are numbered as 4 and 6.

Finally, two new lines can be drawn to obtain the target centroid, useful for place the world reference coordinate system upon calibration and position component from

pose. The intersection of the two lines connecting balls 1-5 and 2-3 result in a new point (7), considered as the target centroid. The position error of this new point is directly dependent of the errors from the detection of all the balls involved (1,1,2 and 3) and it is not used for any other purposes than the presented above.

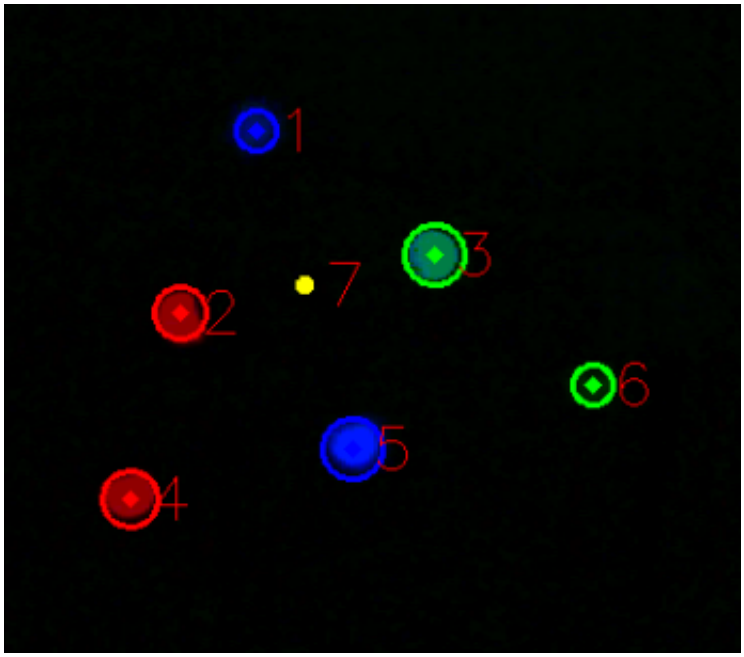


Figure 6.12: Ball enumeration algorithm representation.

6.5.2 Orthogonal vector for rotation matrix

The construction of the orthogonal matrix of rotation presented on chapter 3.7.1, can now be done using direction vectors built using the properties of the designed target. Since the target was designed to be plan in which all the markers should lay on and the markers are placed in a specific, direction vector can be built using 3D position of balls. As presented before, a vector $\vec{AB}$ can be formed by the subtraction of two points as $B - A$. So, to form the vector v_x in the orthogonal rotation matrix, it is used the vector formed by the balls 3 and 2 (3-2), making x axis positive in the $2- > 3$ direction. To compute the v_y vector, it is used the balls 5 and 1 (5-1), making the y axis positive in the direction $1- > 5$. The remaining v_z vector to represent the z axis is formed by the cross product of the previous two, resulting in a new vector, orthogonal to them, with direction respecting the right-hand rule, this is, positive direction from cameras towards the target.

Chapter 7

Results

To study the developed solution, it will be compared to a ground-truth system, the pixhawk auto-pilot. It is widely used for attitude control on robotics due to its sensors quality and filters reliability.

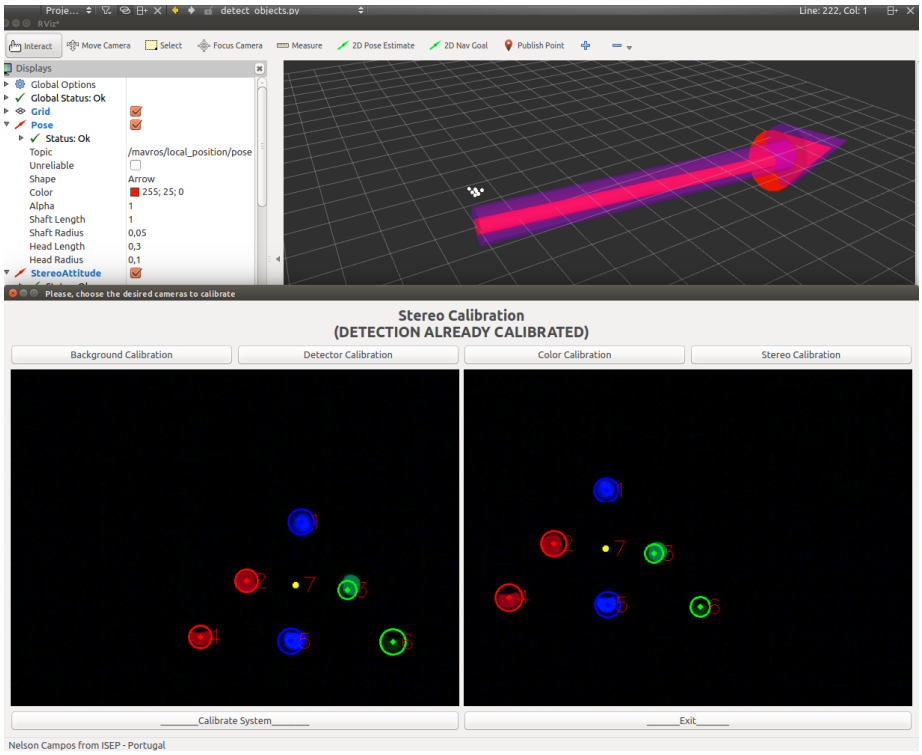


Figure 7.1: Pose output comparison.

Several sensors can be fused in order to improve pose computation and Kalman filters are already applied to smooth the measures and filter outliers from sensors by predicting and updating sensor's measures weighted proportionally to its quality and deviation errors.

Since this auto-pilot computes its attitude and pose based, mostly, on mechanic inertial sensors and, like every sensor, it is noise sensitive, whose error is cumulative along time. This is, sensor's data fusion will also include its random-walk errors or sensor white noise bias provoked by unknown and random causes which lead to growing attitude and pose deviations from reality. To minimize those errors, other sensors must be included on sensor fusion such as GNSS systems, pressure and temperature sensors (sensors measurements may vary depending on temperature so, it should be compensated).

Visual systems error is static and, if untouched, will also remain calibrated which does not happen with inertial systems which often require calibration procedures. Visual systems have lots of other known problems like objects occlusion, high memory usage, light sensitive algorithms which limit vision systems appliances and non-repeatability on features detection over frames.

Vision systems tasks are, most of the times, limited to low acquisition frame rates in computer vision applications (typically up to 60fps or 60Hz in real-time) imposed by the data size which usually reaches several mega bytes per image. Inertial systems work at much higher acquisition rates starting from like 200Hz and its data can be instantaneously processed.

7.1 Static target setup

To characterize random-walk noise from both systems, it will be analysed its response for a static target in the world where the relations between camera-target and target-pixhawk do not change over time. So, a static setup was mounted and the position and orientation deviations were analysed for approximately 4000 seconds (approx. 1 hour).

In a perfect situation, with a perfectly static target, cameras should present constant values for ball detections on image over time and its position and attitude should not deviate a single millimetre/degree.

Figure 7.2 presents the absolute distance between the real position and the observed position for both systems along the time.

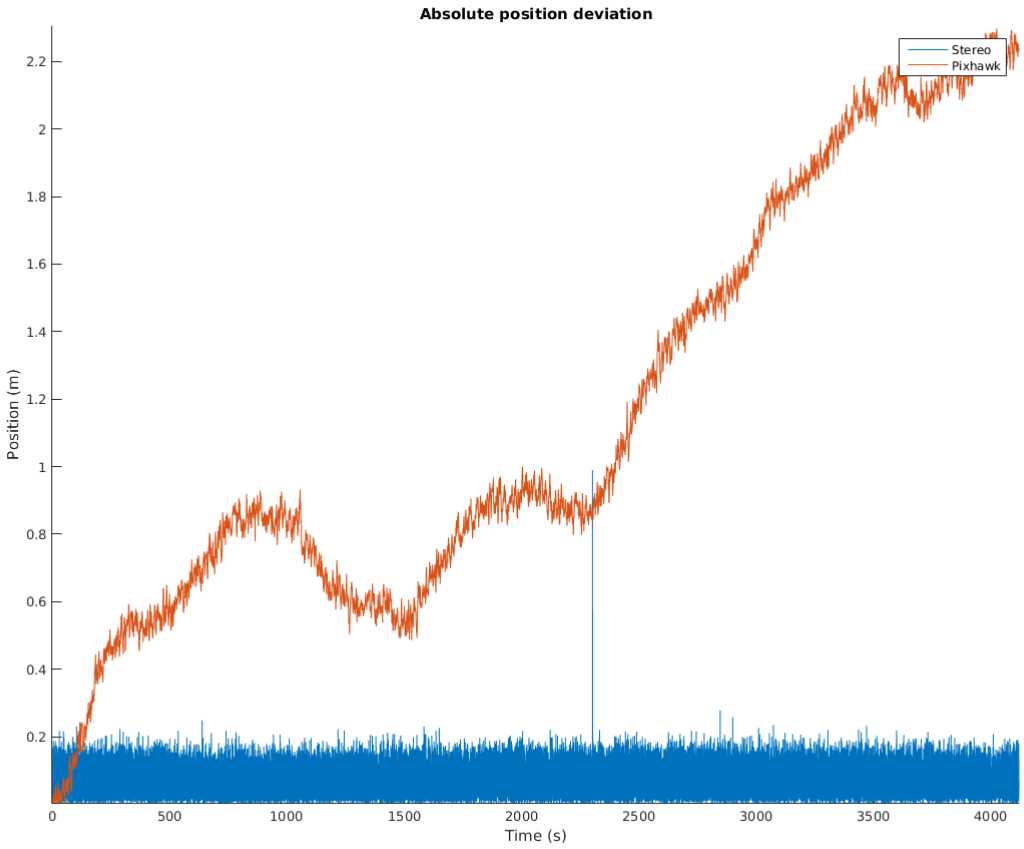


Figure 7.2: Position deviation for static setup.

It is evident the growing position deviation computed based on measures from the inertial system, caused by the cumulative error over the time reaching some meters after one hour.

As expected, this issue is not present on stereo visual system measurements and position computation. By the other hand, the vision detection is not very consistent over time which lead to higher differences between consecutive measurements than the inertial ones, as seen on figure 7.8.

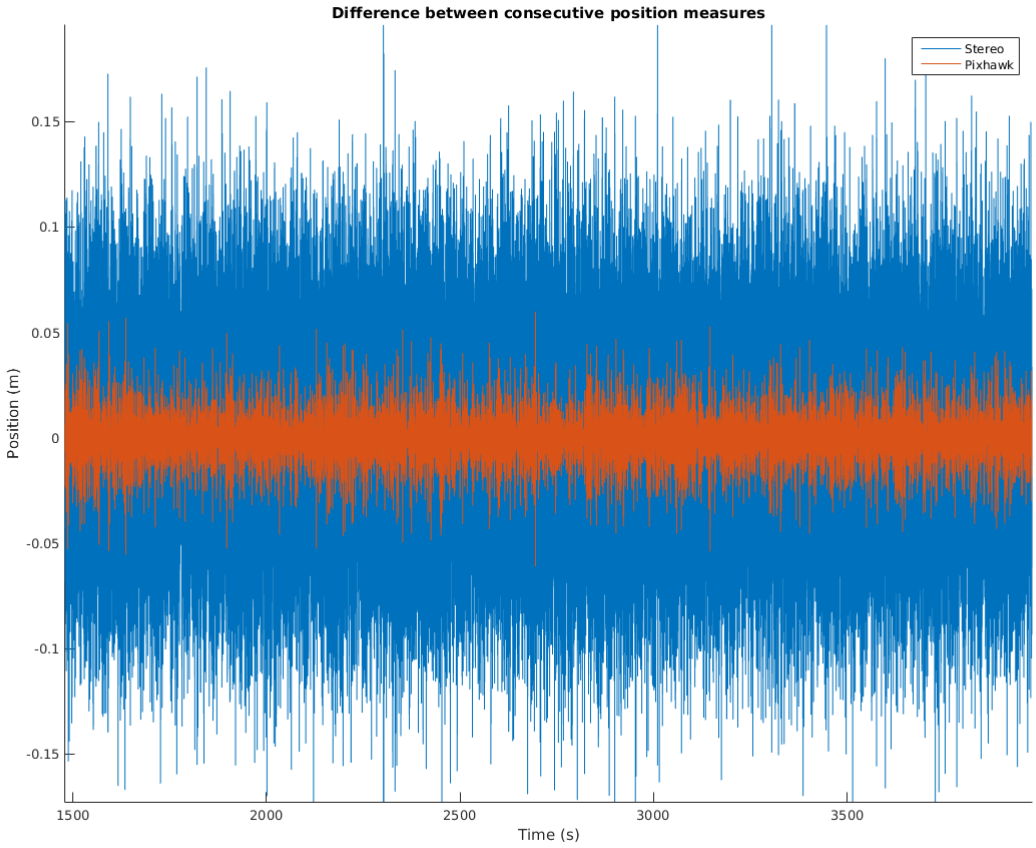


Figure 7.3: Difference between consecutive measurements.

The higher difference on the stereo system measurements is caused by the misdetection of the perfect ball's centroid of each ball which contributes negatively to the homography computation and consequently in the position and attitude computation. This difference could be easily minimized by exhaustive feature and circle detectors to

find the perfect centroid of each ball (which can not be performed in real-time) in order to achieve more precise results.

7.2 Pose computation

Although visual systems may have some advantages, they also have a huge problem which may limit its use for real-time control, the target occlusion or misdetection. This systems are fully dependent on the correct detection of the target.

The baseline distance will dictate the vision system accuracy and consistency. For small baselines, this is, for a small distance between cameras, the accuracy of the measures is highly compromised because the signal-to-noise ratio is small too [108]. Small baselines also present similar views, hence better results on feature matching and low occlusion cases but, in the other hand, are limited to low range applications since its depth accuracy depends directly on the triangulation angle between images.

The occlusion problem is highly felt in this case, since there are several balls on the designed target and it can overlap or partially occlude some balls when its roll or pitch angles are closer to -90 or 90 degrees where all the balls would be aligned face to the cameras or at least, one of them for bigger baselines, leading to perspective loss. This could be countered by introducing more cameras or view-points around the "action scene" where the target should be visible at least on two cameras.

7.3 Rotations

Since we are facing relatively small baselines (26cm to 163cm maximum), for the reasons presented above, roll and pitch rotations study would be hard to compare with the ground-truth system due to partial target occlusions on stereo vision system. Thus, to study the precision of the stereo system measurements, there will be studied yaw rotations only and simple translations along the z axis (from cameras to target).

This time, for a baseline between cameras of 163 centimetres, the target was placed at 3 meters and a rotation of approximately 180 degrees about yaw was performed, followed by a stop of approx. 13 seconds and then a new rotation of -180 degrees back to the original orientation.

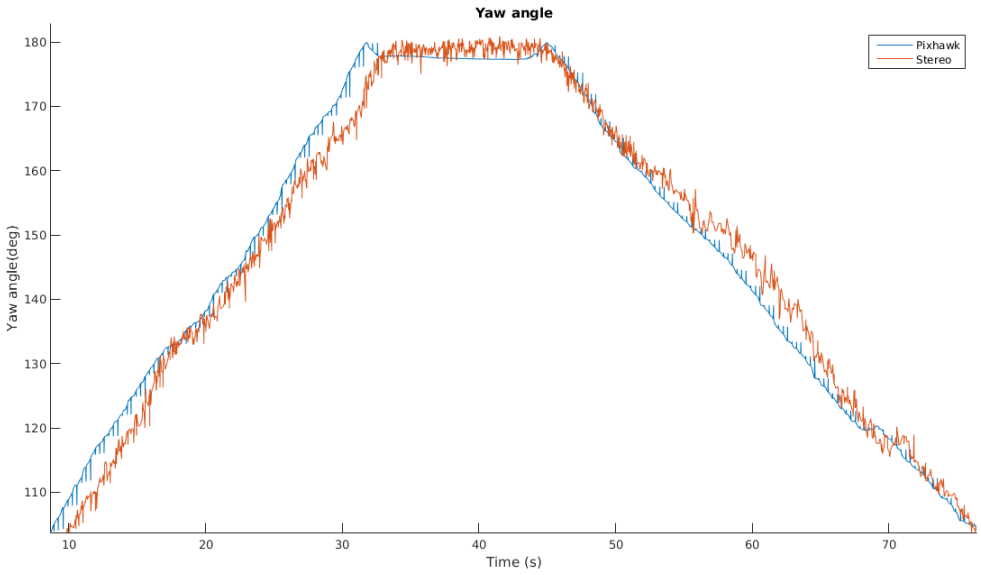


Figure 7.4: Yaw rotation.

The measures from the stereo system present a difference up to 2 degrees which is almost insignificant in view of the fact that this system accommodates several detection errors. Several measures were taken in the stationary moment of the manoeuvre to compare the accuracy of the stereo system face to the ground-truth given by the pose of pixhawk.

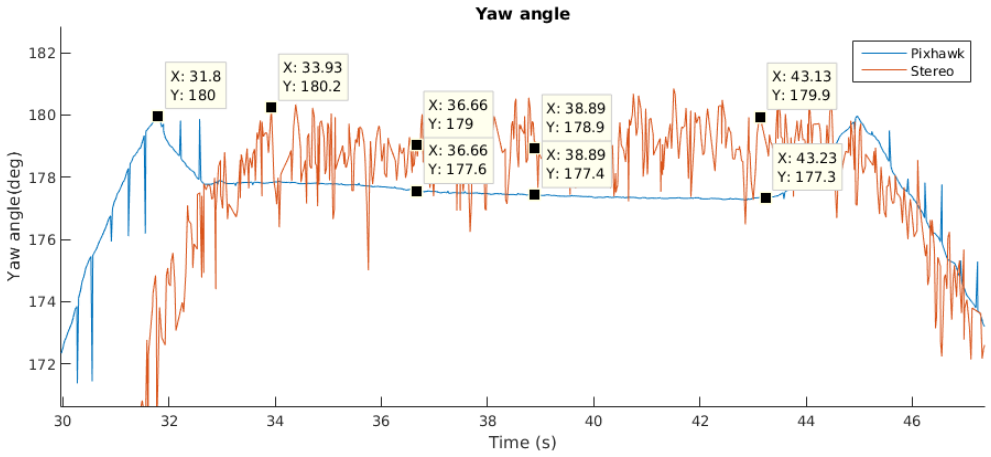


Figure 7.5: Measurements difference for the same timestamps.

The stereo system presents delayed measures by approximately two seconds face to the inertial system timestamps. This error is probably caused by unsynchronized times between the marker pixhawk (running on a Wi-Fi raspberry) and the host computer pose publisher where the stereo toolbox was running.

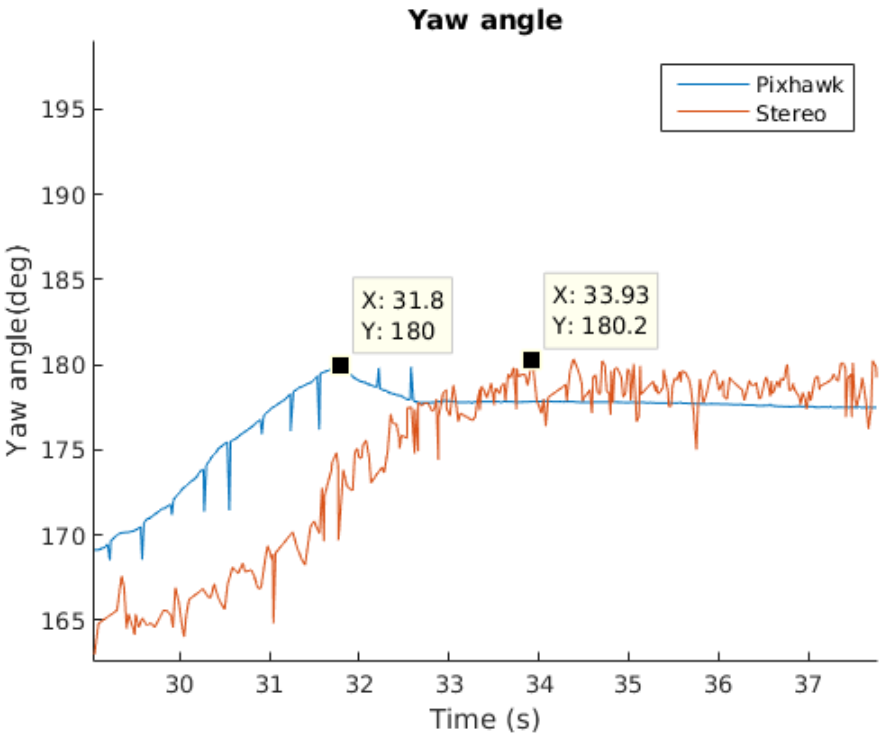


Figure 7.6: Measurements delay.

For real-time purposes, where control tasks could be employed, the delay must be minimized. This application does not perform images or ROIs resizing to avoid quality measure loss, leading to different processing times (hence pose publishing), growing proportionally to the occupied area by the target on the image. This problem could be easily countered by resizing the target ROIs always to the same size, where the processing times should have lower and similar processing times.

7.4 Translation

To characterize the position accuracy and error, a different setup was mounted. A simple movement along the z axis from the stereo system (coincident with the pixhawk)

was performed, approaching the target to the cameras by 30cm approximately. Due to manual positioning and movement of the target, the translation readings may differ from the desired ones. The pixhawk position also include some measurement error. Therefore, more exhaustive tests must be performed later with even more reliable motion readings.

This test was performed at 4 meters far from the cameras in 120 seconds with a baseline of approximately 163cm.

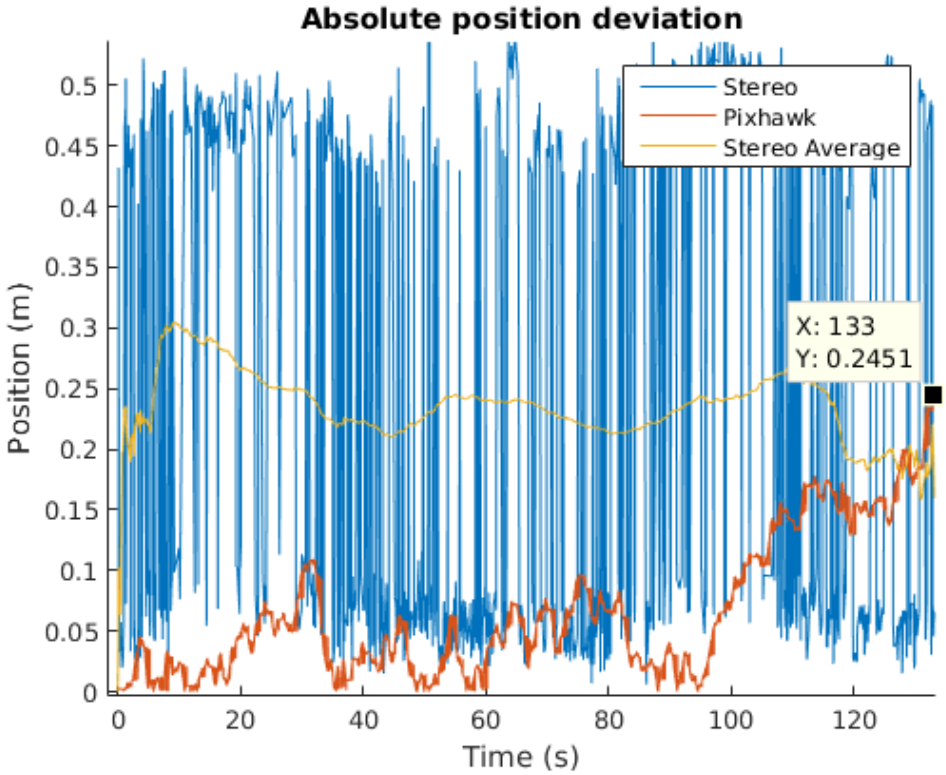


Figure 7.7: Translation measurements.

The stereo system does not present accurate measurements for a small translations like 30cm since its depth error is considerable due to the small signal-to-noise ratio characteristic of relatively small baselines and the misdetection of the target balls is highly felt.

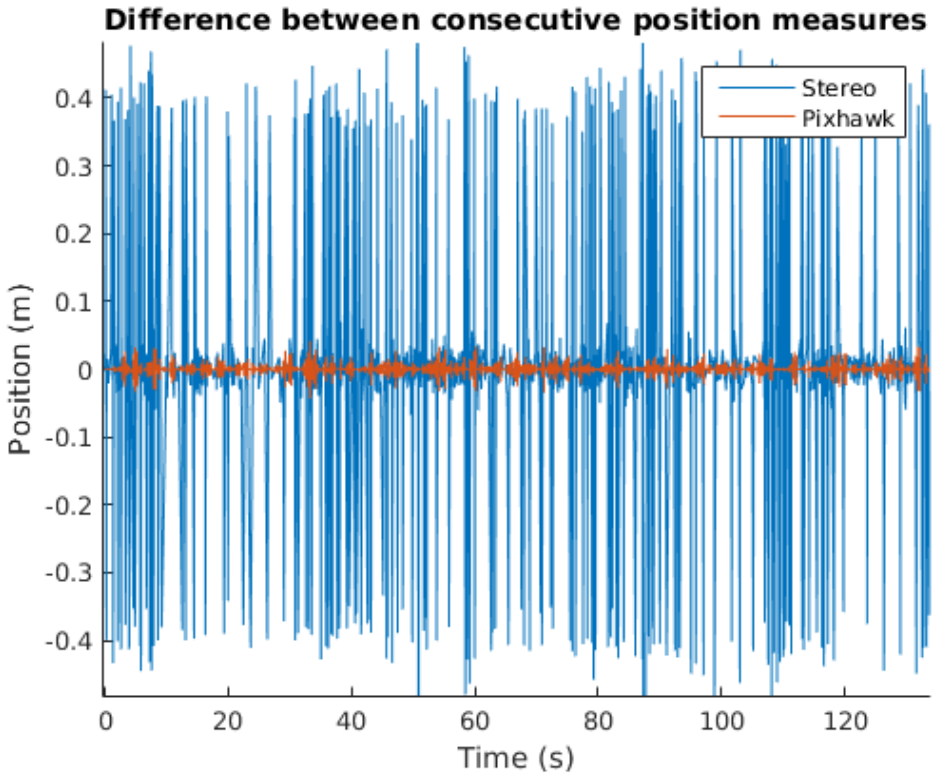


Figure 7.8: Translation measurements.

On the other hand, the pixhawk presents relatively accurate measurements for a 30cm travel, with a final position error of approximately 5cm.

Chapter 8

Conclusions and Future Work

This work aims to give the user a set of needed tools for visual tracking of active targets and pose computation on robotics. Two calibration toolboxes were successfully developed in order to fully calibrate a stereo system along with a pose computation tool for stereo vision systems.

It is possible to say that simpler, faster and even more complete intrinsic and extrinsic calibration toolboxes could be done, yet with similar accuracy to the already existing ones. Future tests must be done to quantify its certainty and accuracy. Some outputs as errors and the camera or the pattern centric view could be available after some tests. Due to a friendly interface and a careful code organization, the toolbox was designed to be easy-to-use and modular which allows future upgrades and the addition of more algorithms and features. It is successfully working with ROS-compliant applications with wide image/video streaming acquisition compatibility.

The calibration of stereo systems is also possible and its precision depends directly on the correct and accurate detection of the target features. Further algorithms could be used to minimize processing costs spent upon information filtering, feature extraction, target detection and occlusion recovery.

It was also proven that the designed target can fulfil the calibration requirements and attitude and position computation under real-time constraints. The attitude and position of the target can be computed with some accuracy, depending on the cameras baseline and target-cameras distance and can be almost instantaneously retrieved/published to console and ROS topics. Although the delay between the measurements and the "reality" is under a few seconds, it may limit its use for control applications. This delay does not exist on post-processing applications where the "timestamp change is static" between the subscribed and the published data.

A accurate 3D cloudpoint was successfully computed and published (type sensor_msgs/PointCloud2), keeping the relations between target balls up to a scalar factor for a correct homography computation with a absolute difference between ball's coordinates within some centimetres. Better results could be achieved if the target balls were smaller and brighter and there were applied complementary techniques such as machine learning to clever and faster target detections, information filters to smooth possible misdetections, path prediction in order to speed up ROIs extraction, adaptive foreground learning factor to better model the background and occlusion redundancy, leading to faster matching algorithms and attitude recovery.

Future updates should also include camera configuration tools in order to adjust camera's parameters such as exposure e brightness, saturation, frame rate and white balance to better environment fitting/detections. More studies about the behaviour of active coloured markers should be done using different color spaces, edge detectors, background extraction and feature detectors.

To unify the processing times to all the cameras resolution and target distances, a resize ratio should be studied or a adaptive resize ratio to ensure similar processing times for the many situations and allow its use on control applications. A post-processing mode should also be directly available with stronger and more accurate detectors.

Taking advantage of the modular architecture of this application, there would be added more cameras or view-points to reduce occlusions and improve attitude and position computation.

Bibliography

[1] Robotic Operative System http://wiki.ros.org/ROS/Introduction

[2] Camera Calibration https://www.mathworks.com/help/vision/camera-calibration.html

[3] Zhang, Z. "A Flexible New Technique for Camera Calibration". IEEE Transactions on Pattern Analysis and Machine Intelligence.Vol. 22, No. 11, 2000, pp. 1330–1334.

[4] Heikkila, J, and O. Silven. "A Four-step Camera Calibration Procedure with Implicit Image Correction." IEEE International Conference on Computer Vision and Pattern Recognition. 1997.

[5] Bouguet, J. Y. "Camera Calibration Toolbox for Matlab." Computational Vision at the California Institute of Technology.

[6] Bradski, G., and A. Kaehler. Learning OpenCV: Computer Vision with the OpenCV Library. Sebastopol, CA: O'Reilly, 2008.

[7] Single Camera Calibration https://www.mathworks.com/help/vision/ug/single-camera-calibrator-app.html

[8] Stereo Camera Calibration https://www.mathworks.com/help/vision/ug/stereo-camera-calibrator-app.html

[9] Single and Stereo Camera Calibration. http://www.vision.caltech.edu/bouguetj/calib_doc/ http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/ref.html

[10] Flexible Camera Calibration by Viewing a Plane from Unknown Orientations - Zhang, ICCV99

[11] A Four-step Camera Calibration Procedure with Implicit Image Correction - Heikkilä and Silven, CVPR97

[12] A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses - R. Y. Tsai ,IEEE J. Robotics Automat.

[13] On Plane-Based Camera Calibration: A General Algorithm, Singularities, Applications - Sturm and Maybank, CVPR99

[14] The Development of Camera Calibration Methods and Models - T.A. Clarke and J.G. Fryer, Photogrammetric Record, 16(91): 51-66, April 1998

[15] Close-Range Camera Calibration - D.C. Brown, Photogrammetric Engineering, pages 855-866, Vol. 37, No. 8, 1971.

[16] Lens Distortion for Close-Range Photogrammetry - J.G. Fryer and D.C. Brown, Photogrammetric Engineering and Remote Sensing Vol. 52(1) pp 51-58, 1986.

[17] Decentering Distortion of Lenses - D.C. Brown, Photometric Engineering, pages 444-462, Vol. 32, No. 3, 1966.

[18] Decentering lens systems - A. Conrady, Monthly notices of the Royal Astronomical Society, Vol. 79, pages 384-390, 1919.

[19] ROS Stereo Camera Calibration - Vincent Rabaud, James Bowman, Patrick Mihelich http://wiki.ros.org/camera_calibration/Tutorials/StereoCalibration

[20] MAtlab Intrinsics Camera Calibration - David Scaramuzza, Martinelli A., Siegwart R., Rufli https://sites.google.com/site/scarabotix/ocamcalib-toolbox

[21] Calibration tool of Jean-Yves Bouget. http://www.vision.caltech.edu/bouguetj/calib_doc/

[22] ETH Zurich ROS Stereo Camera Calibration Paul Furgale, Jérôme Maye, Jörn Rehder, Thomas Schneider, Luc Oth https://github.com/ethz-asl/kalibr

[23] ETH Zurich ROS Stereo Camera Calibration J. Maye, P. Furgale, R. Siegwart (2013). Self-supervised Calibration for Robotic Systems, In Proc. of the IEEE Intelligent Vehicles Symposium (IVS) https://github.com/ethz-asl/kalibr/wiki/multiple-camera-calibration

[24] Paul Furgale, Joern Rehder, Roland Siegwart (2013). Unified Temporal and Spatial Calibration for Multi-Sensor Systems. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan.

[25] Paul Furgale, T D Barfoot, G Sibley (2012). Continuous-Time Batch Estimation Using Temporal Basis Functions. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 2088–2095, St. Paul, MN.

[26] J. Maye, P. Furgale, R. Siegwart (2013). Self-supervised Calibration for Robotic Systems, In Proc. of the IEEE Intelligent Vehicles Symposium (IVS)

[27] L. Oth, P. Furgale, L. Kneip, R. Siegwart (2013). Rolling Shutter Camera Calibration, In Proc. of the IEEE Computer Vision and Pattern Recognition (CVPR)

[28] http://graphics.cs.msu.ru/en/node/909 - GML_cameracalibration - Vladimir Vezhnevets from Lomonosov Moscow State University, Alexander Velizhev from Lomonosov Moscow State University, Anton Yakubenko from Lomonosov Moscow State University, Nikita Chetverikov

[29] ROS Stereo Camera Calibration - Alex Goins, Lucas Walter http://wiki.ros.org/industrial_extrinsic_cal/Tutorials/Intrinsic%20Camera%20Calibration http://rosindustrial.org/news/2016/1/29/new-intrinsic-calibration-procedure

[30] Camera Intrinsics Calibration - MRPT (Mobile Robot Programming Toolkit) http://www.mrpt.org/list-of-mrpt-apps/application_camera-calib/

[31] Camera Calibration Tools - Danail Stoyanov 2011 http://www0.cs.ucl.ac.uk/staff/Dan.Stoyanov/calib/main.html

[32] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3264465/

[33] http://www.whatdigitalcamera.com/technology_guides/cmos-image-sensor-what-is-it-and-how-does-it-work-62586

[34] http://www.red.com/learn/red-101/color-monochrome-camera-sensors

[35] https://www.mathworks.com/help/vision/ug/camera-calibration.html?s_tid=gn_loc_drop

[36] https://www.comp.nus.edu.sg/ cs4243/lecture/camera.pdf

[37] http://www.cs.toronto.edu/ jepson/csc2503/readings/Camera.pdf

[38] https://courses.engr.illinois.edu/cs543/sp2011/lectures/Lecture%2002%20-%20Projective%20Geometry%20and%20Camera%20Models%20-%20Vision_Spring2011.pdf

[39] http://ksimek.github.io/2013/08/13/intrinsic/

[40] http://physbam.stanford.edu/cs448x/oldraParameters.html

[41] http://cilab.knu.ac.kr/English/research/Color/Interpolation.htm

[42] http://www.epixea.com/research/multi-view-coding-thesisse8.html

[43] Faria, André Ferreira. Calibração automática para sistema de localização Multi-câmara. Diss. Instituto Politécnico do Porto. Instituto Superior de Engenharia do Porto., 2014.

[44] https://www.youtube.com/watch?v=zc8b2Jo7mno

[45] http://learning.eng.cam.ac.uk/pub/Public/Turner/Teaching/intro-lecture-12.pdf

[46] Yi Ma, Stefano Soatto, Jana Kosecka, and S. Shankar Sastry. An Invitation to 3-D Vision: From Images to Geometric Models. SpringerVerlag, 2003

[47] http://www.cs.toronto.edu/ jepson/csc2503/tutorials/homography.pdf

[48] Zhengyou Zhang. A flexible new technique for camera calibration. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 22(11):1330–1334, 2000.

[49] http://docs.opencv.org/trunk/d1/dc5/tutorial_background_subtraction.html

[50] http://docs.opencv.org/3.1.0/db/d5c/tutorial_py_bg_subtraction.html

[51] KaewTraKulPong, P., and R. Bowden. "An Improved Adaptive Background Mixture Model for Real time Tracking with Shadow Detection,‖ in proceedings of 2nd European Workshop on Advanced Video Based Surveillance Systems, Sept 2001." Kingston, UK, September (2001).

[52] Godbehere, Andrew B., Akihiro Matsukawa, and Ken Goldberg. "Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation." American Control Conference (ACC), 2012. IEEE, 2012.

[53] http://docs.opencv.org/3.1.0/de/d25/imgproc_color_conversions.html

[54] Ford, Adrian, and Alan Roberts. "Colour space conversions." Westminster University, London 1998 (1998): 1-31.

[55] https://www.codeproject.com/KB/recipes/619039/SIFT.JPG

[56] http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_features_harris/py_features_harris.htm corners

[57] Rosten, Edward, and Tom Drummond. "Machine learning for high-speed corner detection." Computer Vision–ECCV 2006 (2006): 430-443.

[58] http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html#sift-intro

[59] Lowe, David G. "Distinctive image features from scale-invariant keypoints." International journal of computer vision 60.2 (2004): 91-110.

[60] http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html#surf

[61] Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features." Computer vision-ECCV 2006 (2006): 404-417.

[62] http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_brief/py_brief.html#brief

[63] Calonder, Michael, et al. "Brief: Binary robust independent elementary features." Computer Vision-ECCV 2010 (2010): 778-792.

[64] http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_orb/py_orb.html#exercises

[65] P. L. Rosin. Measuring corner properties. Computer Vision and Image Understanding, 73(2):291 - 307, 1999.

[66] Rublee, Ethan, et al. "ORB: An efficient alternative to SIFT or SURF." Computer Vision (ICCV), 2011 IEEE international conference on. IEEE, 2011.

[67] Hough, Paul VC. Method and means for recognizing complex patterns. No. US 3069654. 1962.

[68] Duda, Richard O., and Peter E. Hart. "Use of the Hough transformation to detect lines and curves in pictures." Communications of the ACM 15.1 (1972): 11-15.

[69] http://homepages.inf.ed.ac.uk/rbf/BOOKS/BANDB/LIB/bandb4_3.pdf

[70] Ballard, Dana H. "Generalizing the Hough transform to detect arbitrary shapes." Pattern recognition 13.2 (1981): 111-122.

[71] Zheng, Yang, Ka-Chun Chan, and Charlie CL Wang. "Pedalvatar: An IMU-based real-time body motion capture system using foot rooted kinematic model." Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on. IEEE, 2014.

[72] https://www.lp-research.com/real-time-human-motion-tracking/

[73] https://www.xsens.com/products/mvn-biomech/

[74] https://upload.wikimedia.org/wikipedia/commons/f/fe/KinectSensor.png

[75] https://www.xsens.com/wp-content/uploads/2014/11/MVN-BIOMECH-Awinda.jpg

[76] Chung, Jaeyong, Jin Ryong Kim, and Kwanghyun Shim. "Vision based motion tracking system for interactive entertainment applications." TENCON 2005 2005 IEEE Region 10. IEEE, 2005.

[77] https://optitrack.com/public/images/prime-13-perspective-1200w.png

[78] http://mocapsolutions.com/wp-content/uploads/2012/11/14mm-SBXV-angles.jpg

[79] http://i.vimeocdn.com/video/504113270_1280x720.jpg

[80] https://i.ytimg.com/vi/Kms0zEXpYN4/maxresdefault.jpg

[81] http://www.simi.com/en/products/hardware-and-recording-systems.html

[82] http://holoneer.com/

[83] http://metamotion.com/gypsy/Animazoo.html

[84] http://metamotion.com/images/Gypsy-Gyro-front.jpg

[85] https://www.ascension-tech.com/products/product-history/

[86] http://www.5dt.com/products/images/3rdp_ascen_motionstarW02.jpg

[87] http://www.red.com/learn/red-101/color-monochrome-camera-sensors

[88] https://upload.wikimedia.org/wikipedia/commons/d/d1/Samsung_Galaxy_S_camera_sensor.jpg

[89] https://courses.engr.illinois.edu/cs543/sp2011/lectures/Lecture%2002%20-%20Projective%20Geometry%20and%20Camera%20Models%20-%20Vision_Spring2011.pdf

[90] https://i.pinimg.com/736x/44/16/ed/4416eda8acaea3e8121beb6899276b84–perspective-drawing-lessons-linear-perspective-lesson.jpg

[91] https://qph.ec.quoracdn.net/main-qimg-18c12bb9cb0be29839ec55ba8d18fd2e

[92] https://www.mathworks.com/help/vision/ref/cameracalibrator_tangentialdistortion.png

[93] http://www.learnopencv.com/homography-examples-using-opencv-python-c/

[94] https://upload.wikimedia.org/wikipedia/commons/thumb/0/08/Manhattan_distance.svg/283px-Manhattan_distance.svg.png

[95] http://www.maths.manchester.ac.uk/ pas/code/notes/part2.pdf

[96] http://rockinrobotchallenge.eu/rockin_d2.1.7.pdf

[97] Fujii, Keisuke. Ëxtended kalman filter.R̈eference Manual (2013).

[98] Feng, Lin, and Qi Fangchao. "Research on the Hardware Structure Characteristics and EKF Filtering Algorithm of the Autopilot PIXHAWK." Instrumentation & Measurement, Computer, Communication and Control (IMCCC), 2016 Sixth International Conference on. IEEE, 2016.

[99] https://store.3dr.com/t/pixhawk

[100] https://upload.wikimedia.org/wikipedia/commons/2/2d/Intersection_over_Union_-_object_detection_bounding_boxes.jpg

[101] http://www.chrobotics.com/wp-content/uploads/2012/11/Inertial-Frame.png

[102] https://www.autodesk.com/products/motionbuilder/overview

[103] https://www.mathworks.com/help/vision/ug/camera-calibration.html?requestedDomain=www.mathworks.com

[104] https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html?high

[105] https://docs.opencv.org/3.1.0/canny1.jpg

[106] MA, Yi, et al. An invitation to 3-d vision: from images to geometric models. Springer Science & Business Media, 2012.

[107] https://www.mathworks.com/help/vision/ref/triangulatemultiview.html?searchHighlight=reprojection%

[108] VIDAL, René; OLIENSIS, John. Structure from planar motions with small baselines. In: European conference on computer vision. Springer, Berlin, Heidelberg, 2002. p. 383-398.