POLITECNICO DI TORINO

Master's Degree in Electronic engineering

Master's Degree Thesis

Extrinsic Stereo Camera Calibration using Human Pose Recognition

Supervisors

Prof. Stefano DI CARLO

Prof. Alessandro SAVINO

PHD Francesco ROSSI

Riccardo BUSSOLINO

Candidate

Federico VALENTINI

July 2021

Summary

The extrinsic camera calibration is a process to define a multi camera system relative positioning. This is a very time-expensive procedure that must be repeated every time one or more element of the system is barely moved from its original position. The calibration process requires the preparation of a test pattern (eg: chessboard). The user must capture simultaneously a large quantity of pictures from every camera where the test pattern is clearly visible and the common field of view of every camera must be covered as much as possible to obtain the best result possible. It's evident that if this process is performed by a non-expert user can result in troublesome situations, that can lead to failure or non-precise calibration. This work goal is to deepen the extrinsic calibration operation to discover a more intuitive procedure to obtain an automatic approach that is fast, accurate and easy to use for a non-expert user. The case analyzed is the one where the calibration is made using the human body as a test pattern instead of a reference object. In this way everyone is able to perform the calibration process by just walking in the common field of view of the multi-camera setup for a few minutes.

Acknowledgements

Many people are to thank due to their patience and support both moral and technical.
I would like to thank all the Astar group that always supported me and even if my main skills belong to another branch of engineering they believed in me more than I could have imagined. A special thanks goes to Francesco Rossi that has covered almost any kind of roles during this work, ranging from technical support to moral encouragement.
I would like to thank Prof. Stefano Di Carlo and Prof. Alessandro Savino for their support and their time.
Last but not least important, I would like to thank my family, my friends and my partner. They are part of my everyday life, they had to deal with me in the worst and in the best period of these years, so they deserve a big special thank.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction and basic concept

This work is based on the aim of Astar[1] group to make the stereo-camera calibration simpler and cheaper, not only in terms of money but also in terms of time.
Astar is a startup based in Turin that works in the sport intelligence field. The aim of this society is to translate the data obtainable from images or videos to accurate informations and make them more accessible to the final customer. Another business that Astar is involved into is the usage of video techniques to study the bio-mechanics of the movements of a human body recorded by one or more cameras.

[1] Astar SRL website

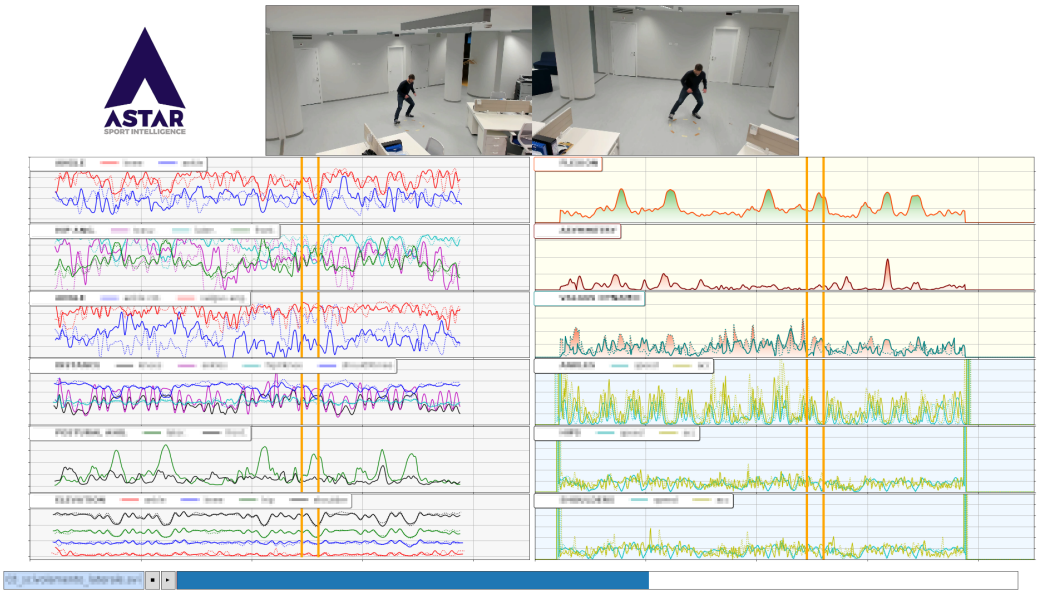


Figure 1.1: Example of an Astar project

To better understand the results of Astar projects, in the figure 1.1 one of their algorithms is analyzing and tracking the bio-mechanical structure of the human body captured by the two cameras. In this picture the multiple graphs reported under the camera views describe different body parameters versus time; in this way the analysis moment by moment of every human body parameter is intuitive and immediate. The indexes of the graphs have been deliberately blurred because they contains confidential informations that can't be disclosed.
Beginning with the premise that the space resolution is considered as the precision in terms of pixel of the recognition of a point in the space captured in an image and the time resolution as the amount of time that elapses between a frame and the next one. The study of the 3D human pose is required to obtain the result descripted before and there are two approaches, that differ for the space-time resolution:

- qualitative: the human pose is recognized without too many details, the three-dimensional mapping in the space is approximated and not very accurate (eg: it's recognizable if the human is standing up or sitting on a chair, but not the precise geometry of the body); in this case the spacial precision and the time resolution are not very high

- quantitative: this is the approach that the Astar group tries to pursue, it implies having the maximum space resolution; this can be obtained using high performance hardware (cameras able to capture videos with an high framerate) and a perfectly calibrated system to triangulate in the most accurate way the scene using every available point of view

Astar aims to use the highest space-time resolution possible to obtain the most accurate model of the movements of the body. This work has mainly two field applications:

- prevent injuries by analyzing the structure and the geometries of the human body during movements to find the critical points where the person must improve
- compare the movements before and after an injury to understand if there are any after-effects or if the cause of the injury has been corrected, since a lot of time the same injury can occur to the same person multiple times

Nowadays the gold-standard technique includes wearable sensors, mounted on a suit that the person must wear to allow the system to track his movements. Sometimes GPS sensor are used but the most common technique is to use infrared markers that can be tracked only if englightened by an infrared camera, so it's necessary to setup this system in a very technical environment. This system is very expensive and can be modified only by a skilled technician. An example of this technology is offered by Xsens, a company based in Netherland that offers different solution for human motion measurements for various applications, ranging from ergonomics to animation.

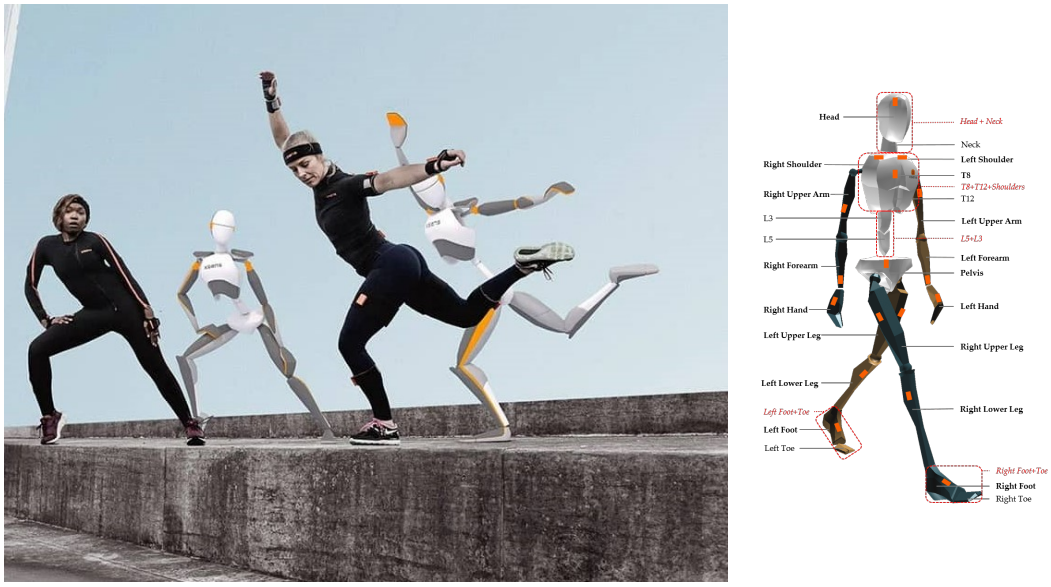


Figure 1.2: Xsense results using wearable sensors (left) and sensors position on the human body (right)

In the figure above 1.2 there is an example of how an human body (with wearable markers on it) can be replicated by the software to analyze in detail each movement

performed. On the left leg of one person the marker is clearly visible (the orange rectangular object on the middle of the thigh) and there are a lot of them spread out on the whole body.
Astar aims to replicate the same result both in small and wide fields using some cameras that can be mounted on the spot by a non-skilled person.
The idea is to obtain a markerless system, but to compete with this reference technique the maximum space-time resolution is required. The concept is based on a multi-camera setup, where each camera is already intrinsically calibrated, that is delivered to the final customer along with a software system that is able to obtain the extrinsic calibration of the system without having any computer vision knowledge. To obtain the extrinsic informations the algorithm requires that the final user records a video where a human body is moving in front of the camera for a couple of minutes. The recorded video is analyzed and all the necessary informations are automatically obtained and the system is ready to be used. This conception expects to use the human body recognition to calibrate the system that must be able to recognize the human body geometries, so the final application exploit the same technology that, in turn, is used to calibrate the system itself.
The pose recognition technology is expanding itself year by year, it ranges from the ergonomy field to the security one and Astar wants to bring its contribution by integrating it inside the sport technology. This concept tries to simplify the bio-mechanical study of the human body without using complex and expensive infrastructures and skilled technicians, just with some cameras and a software that is able to calibrate them as accurately as possible.
During this thesis all the ways to obtain this calibration are analyzed to understand the benefits and the defects of each one, to obtain at the end a result that is as close as possible to the gold-standard one.
This thesis work has been organized in five different chapters that will explain every steps performed during the research of the best method to achieve the results desired.
In the present chapter of this thesis the work environment and the concept of the idea at the base of all this work are described. During this chapter there is the explanation of the goals and the fields where this work wants to bring significant improvements. The focus is mainly on the easing and the speedup of this procedure (automatizing it as much as possible) without the need of an expert technician without losing the calibration accuracy.

In the second chapter of this thesis there is the introduction to camera calibration, its parameters and the description of how a single or dual camera setup is made; here it's also present the basic knowledge of the technique, called Stereo vision, needed to follow the next steps.

In the third chapter there is the description of how the intrinsic camera calibration has been performed; during this step the aim is to obtain all the data related to the imperfections of the two cameras individually that are mandatory in the next steps. This procedure is mandatory since without the intrisic calibration the distorsions and all the other non-ideality of the camera devices can't be corrected.

The fourth chapter consists of the overview of the extrinsic calibration of a dual camera system using an object with the reference pattern; in this procedure two cameras have been set up in an appropriate room with enough space to allow the work to proceed flawlessly and a software able to understand the relative positioning of the two cameras has been implemented. This step is used to have a gold standard to be used as a reference to compare the obtained result with the final automated method.

The fifth chapter provides the basic knowledge on how to obtain the final automated approach and all the components and techniques used to achieve this result. Inside this chapter there is also the description of the automated easy-to-use method, the results obtained nd the comparison of the result with the gold standard previously obtained.

The sixth chapter provides conclusions on this automated approach.

Chapter 2

Camera Calibration

The camera calibration is defined as the process used to obtain the parameters related to lenses and sensor of a camera. Cameras used in a lot of computer vision field must be calibrated before the usage to be sure that the distortion of the image due to the device imperfections are filtered out before the elaboration of the image. If the camera is used to measure object or to detect a distance it must be calibrated otherwise the distortions and the manufacturing imperfections could affect the quality of the final results.
Therefore are clear the reasons that make the calibration mandatory when the accuracy level required it very high. This procedure aims to obtain the direct correspondence between the 3D points in the real world captured and the 2D points contained in the image taken.

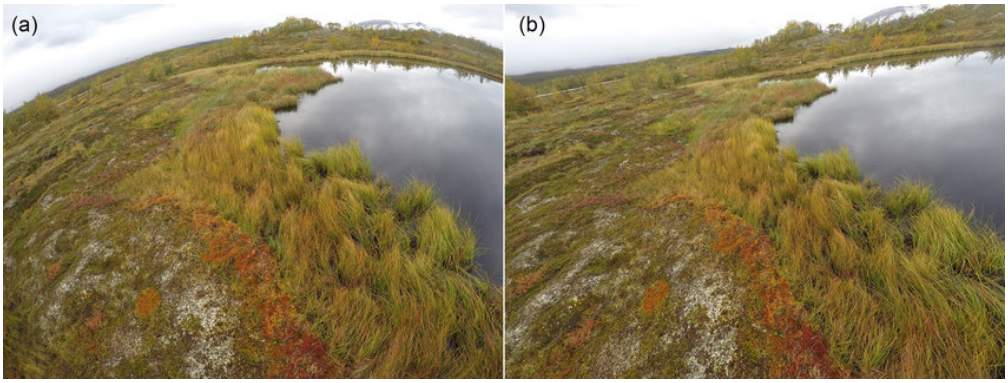


Figure 2.1: Raw image (a) vs undistorted (b) image

In the figure 2.1 there is a clear example of how much the distortions can affect the correspondence between the real world and the image; taking as example the horizon, it's evident that in the image on the left is rounded while in the other image is perfectly straigth as it is perceivable in real life.

2.1 Pinhole camera model

To describe how the camera works the pinhole camera model is used. This model describes the device as a camera that can see the real world through a very small hole. According to this the light rays are able to reach the camera only through this aperture, so the rays project the image upside down after passing through the hole.

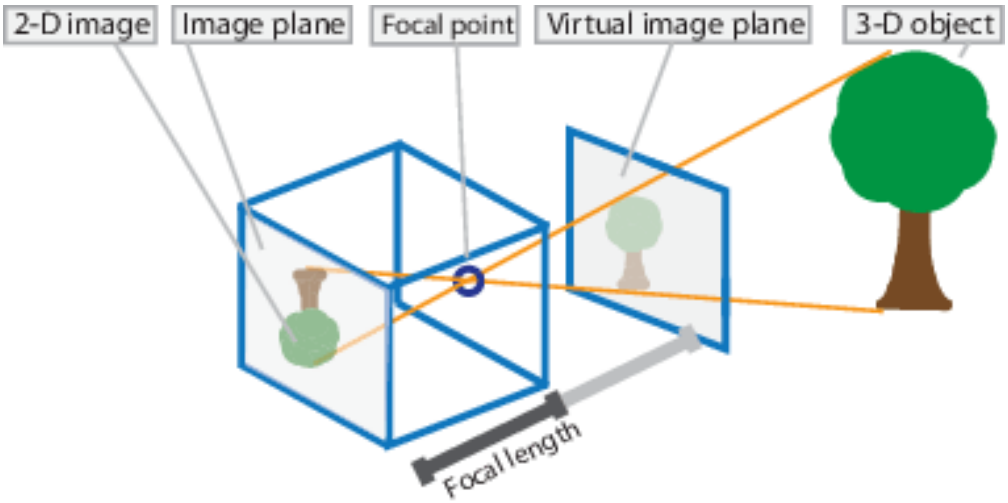


Figure 2.2: Pinhole camera model

In the figure 2.4 there is a graphical representation of how the pinhole camera model works and the labeling of every components involved that will be analyzed in the next steps to better understand the analysis of the camera calibration (according to [1].

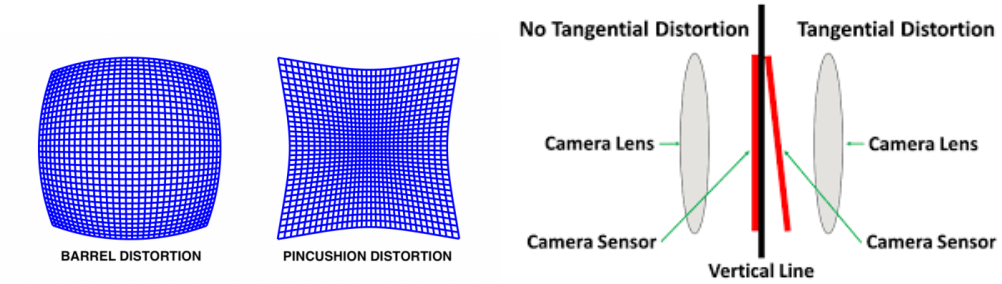
The algorithm used in the camera calibration process is able to transform the 3D coordinates of the camera to 2D coordinates of an image by finding:

- **Focal length**, the distance from the center of the sensor to the plane where the image is projected
- **Optical point**, also called optical center, that identify the point where all light rays converge to form the image, in the case of the pinhole camera model, these coordinates identify the center of the pinhole.

The pinhole camera model can introduce deviations from rectilinear projection on the image. This phenomena is called distortion and this can occur in two different ways:

- **Radial distortion**, this alteration causes the straight lines to appear slightly rounded

- **Tangential distortion**, this effect is caused by the imperfection of the alignment between the lens and the plane where the image is projected, this leads to feel nearer or further some parts of the image compared to how they really are



(a) Radial distortion **(b)** Tangential distortion

Figure 2.3: Description of the two types of distortion

In the figure 2.3 there is a graphical explanation of the two different types of distortion. The algorithm is able to detect these distortions and find the correct parameters to rectify every image taken with that specific camera.

2.2 Intrisic matrix

The intrinsic matrix defines how it is possible to transform 3D camera coordinates to 2D homogeneous image coordinates. From the pinhole camera model are retrieved the parameters needed to define the intrisic camera matrix.

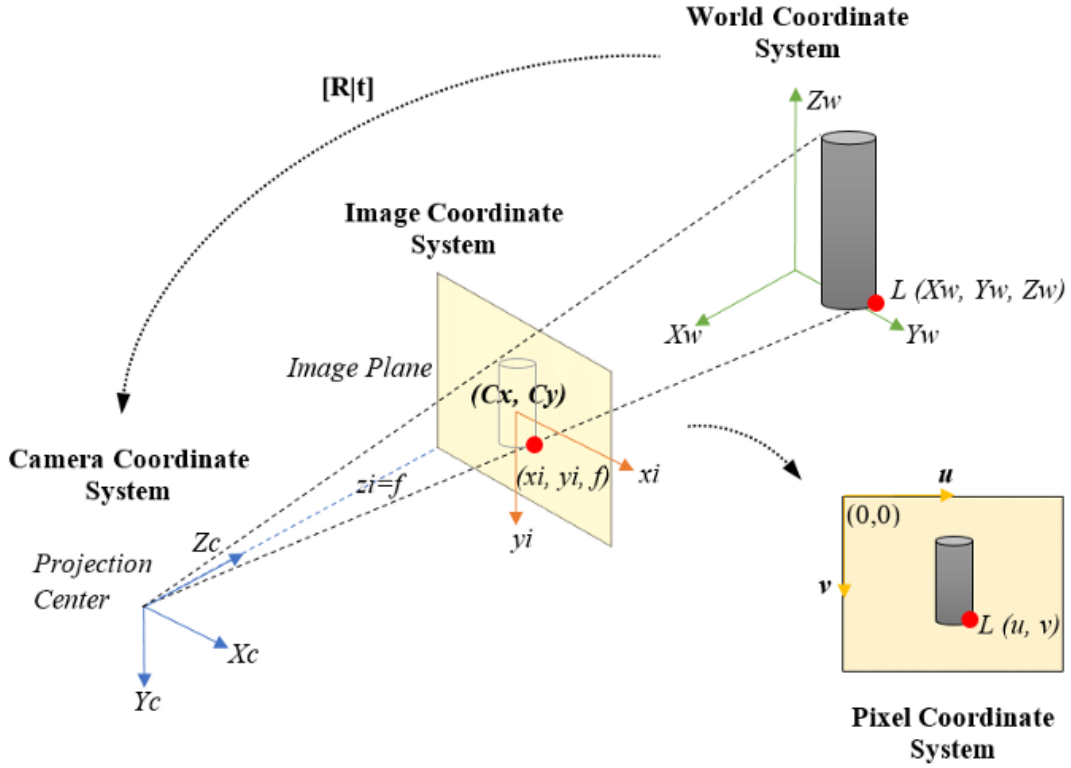


Figure 2.4: Pinhole camera model

In the figure 2.4 there is the simplified model of how the real world is projected in the pixel coordinate system. The following statements are done: $O_{world}(X_w, Y_w, Z_w)$ are the real world coordinates, $O_{cam}(X_c, Y_c, Z_c)$ the camera-centered coordinates and $O_{img}(u, v)$ the pixel coordinates on the imaging plane. To convert real world coordinates in the camera centered ones it is necessary to perform a rotation and a translation defined by two different matrix called respectively R (3x3 matrix) and t (a vector made of 3 values, one per dimension).

To obtain the entire conversion from 3D world to 2D image (according to [2]) two steps are needed:

- Conversion of real world coordinates to the camera coordinates

- Conversion of camera coordinates to the image coordinates

These steps are performed using the following relations:

$$O_{cam} = \Big[R|t\Big] * O_{world} \tag{2.1}$$

$$O_{img} = K * O_{cam} \tag{2.2}$$

Where the symbol | means concatenation so the $R|t$ coefficient is a 3x4 matrix, while K is the intrinsic camera matrix, that is composed in the following way:

$$Intrinsic\ camera\ matrix = \begin{bmatrix} f_x & 0 & c_x \\ s & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

where f_x and f_y represent focal length in pixels and c_x and c_y are the optical point's coordinates. The focal length coordinates can be obtained using the following relations:

$$f_x = F/p_x \qquad\qquad f_y = F/p_y \tag{2.4}$$

In the equation above p_x and p_y are the size of a pixel in the two directions and F is the focal length of the camera, both this values are expressed in millimeters.

$$O_{img} = P * O_{world} = K * \Big[R|t\Big] * O_{world} \tag{2.5}$$

As results of these two conversions, we obtain a projection matrix P:

$$P = K * [R|t] \tag{2.6}$$

The **S** parameter in position 2,1 of the intrisic matrix (5.6) indicates the pixel skew value. This parameter describes how the shape of the pixels of the sensor is done, this can assume the parallelogram shape instead of the square one. This phenomen is explained in the following figure 2.5, it is clear how the pixels in the grid are not perfectly squared. A light ray activate a pixel in a specific position based on the coordinates of the light ray on the image plane, but if the pixel grid is skewed the pixel activated is not the expected one.

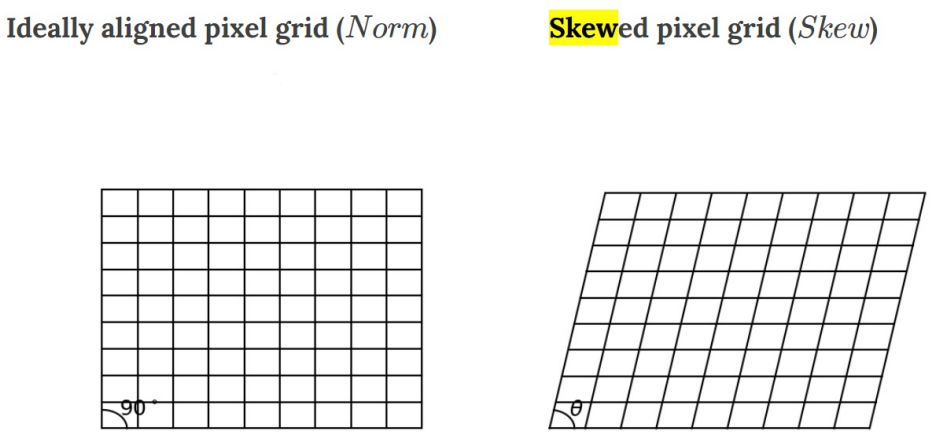


Figure 2.5: Pixel grid skew phenomenon

This distortion can be evaluated according to the following equation:

$$S = f_x * \tan(a) \tag{2.7}$$

where the f_x parameter is the focal length on the x axis and the angle a is defined according to the figure below 2.6 as the angle on the x axis between the ideal vertical direction and the slanted side of the pixel.

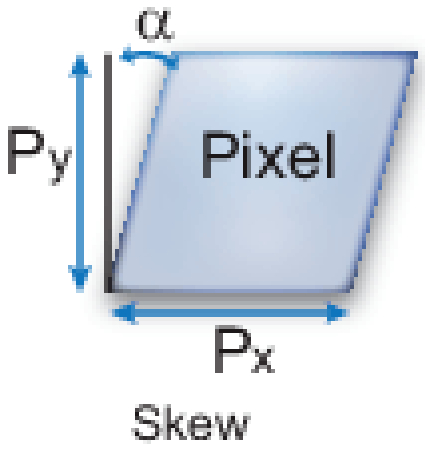


Figure 2.6: Pixel skew parameters

Nowadays in most of the cases the pixel skew is trascurable or is absent due to high quality manufacturing systems, but in some cases (eg: a picture of a picture) the skew must be taken in account.

The projection matrix (5.6) contains the intrinsic camera matrix and the rotation/translation information, but since multiplication of matrices is possible only if

the number of columns of the first matrix corresponds to the number of rows of the other matrix, it is necessary to add a value at the end of the O_{world} vector to allow the multiplication. The world coordinate must be expressed as:

$$O_{world} = [X_w, Y_w, Z_w, 1] \tag{2.8}$$

the last value is the extra value, called homogeneous coordinate or projective coordinate, to allow the multiplication with the other matrices.

This matrix is unique for every single image acquisition setup (considering lenses, sensors and all the parts that compose the acquisition device), since it's also related to the manufacturing tolerances.

Below in figure 2.7 there is an example of the effect of the correction of an image, is noticeable how the image is more compressed on the edges, since these areas are where the camera displays more distortions. During the acquisition procedure a lot of images were taken when the chessboard was placed on the perimeter of the field of view of the camera to better characterize the behaviour of the device in those areas. By zooming on the chessboard it's easy to notice how the lines in the undistorted image has been straightened.

(a) Before **(b)** After

Figure 2.7: Example of undistortion of an image

2.3 Stereo camera and extrinsic matrix

When an image is taken the 3D world is converted to a 2D image, in this way the information about the third dimension (depth) is lost.
As the human body does with the eyes, the same thing can be done with cameras, a stereo camera setup can be used to be able to estimate depth. This technique is called stereo vision ([3]).

In the figure below 2.8 there is a graphical representation of how the human eyes work compared to the stereo camera system, in both case the same scene is seen from two different point of view centered in different positions.

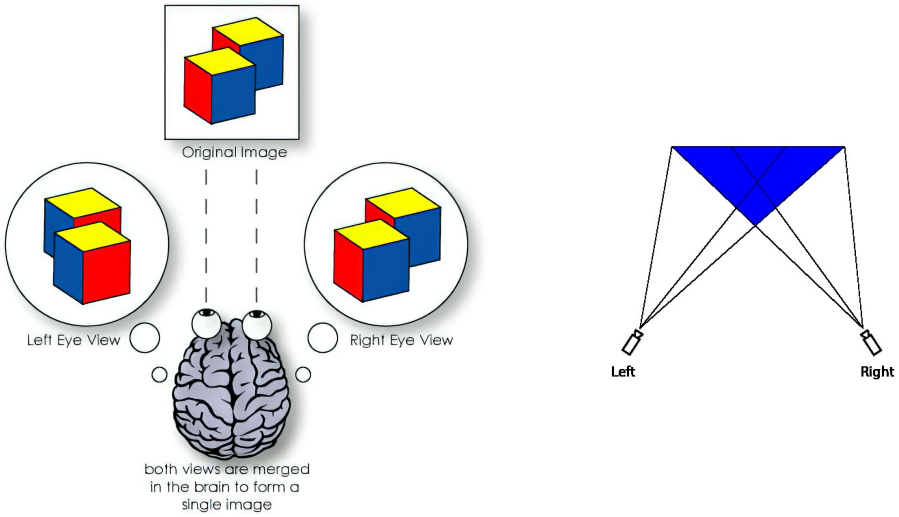


Figure 2.8: Stereo vision examples

The basic idea of this setup is to be able to triangulate items in 3D real world coordinate (according to [4]). A single camera with the sensor in the coordinate **O** can associate only one pixel to a generic point **X**, every other point on the line **OX** can't be seen from this camera.
If two different cameras are considered, the second one, centered in **O'**, is able to associate different pixels to various points on the line **OX**. These points are connected with a line **l'** that corresponds to the point **x** on the other camera, this line is called **epiline** (for example the purple line contained in the image 2.9).

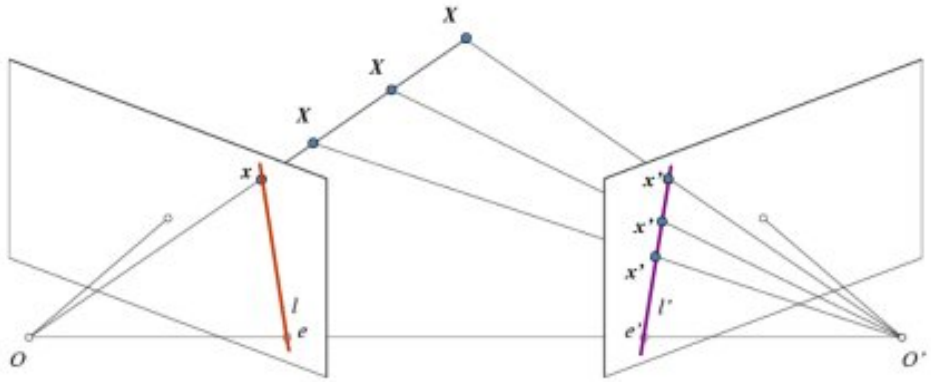


Figure 2.9: Stereo camera setup example

In the example above (figure 2.9) is clear how the different **X** points are compressed in a single pixel for the left camera due to the same x and y coordinates on the imaging plane, while they are all visible for the right camera, this is due to different positioning of the devices.

Figure 2.10: Epilines

Every epiline converges to a point called **epipole** that is the position of the other camera in the imaging plane. In the example above the epipole is outside of the image since the two cameras can't see each other. The line that connects the two centers of the devices (**O** and **O'**) is called **baseline**, so the epipoles are the intersection of the baseline with the imaging plane. The plane that contains the point **X** and the baseline is called **epipolar plane**. Based on the epipolar geometry statements done before it is stateable that the research of the correspondence of the **X** point on the camera centered in **O** with the one centered in **O'** is limited to a single 2D research on the epipolar line. With this kind of system an image's pixel of the left camera is associated to an epiline on the right camera and viceversa. In this way the matching process of the points in two different images is way faster instead of considering the entire amount of pixels in the image since a two-dimensional problem is converted into an one-dimensional one. To be able to match points between images of two different cameras we need to know the relative positioning of these devices. Rotation and translation vectors of the position of a camera in relation to the other defines the parameters needed to obtain the extrinsic matrix of a stereo camera setup.

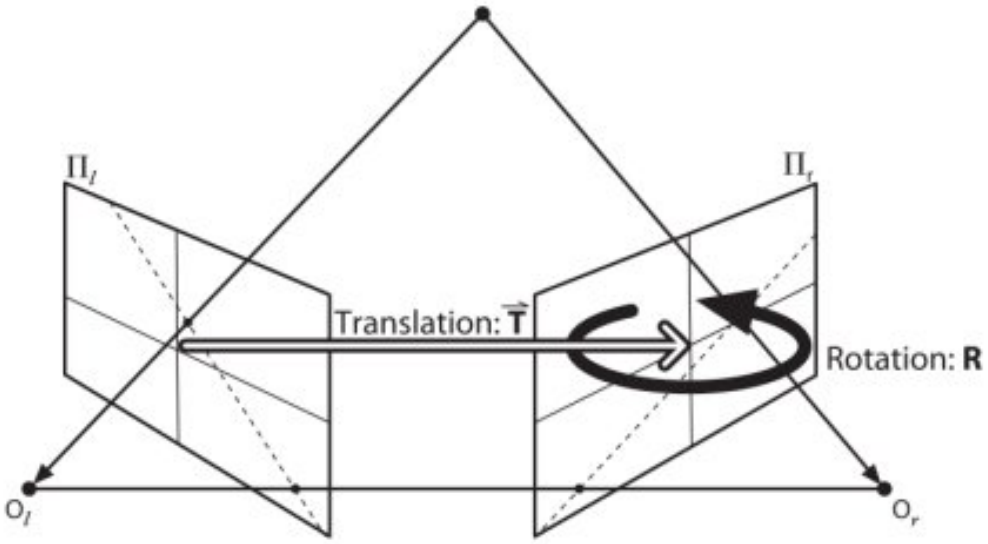


Figure 2.11: Extrinsic parameters meaning

The **essential matrix** describes the relative positioning of the two cameras in global coordinates.
The **fundamental matrix**, that is the final result of the stereo camera setup calibration, describes the entire epipolar geometry of the setup, substantially it's the same thing of the essential matrix but using pixels as unit of measure (according to [5]). Inside this 3x3 matrix there are also the intrinsic matrix information to be able to relate both cameras using pixel measurements.

Chapter 3

Intrinsic calibration

The final target of this thesis is to work with a stereo camera system, so the two cameras need to have all the parameters to characterize the intrinsic calibration of both cameras. These informations are mandatory otherwise the extrinsic calibration of the system isn't accurate due to distortions and manufacturing imperfections. In this work two simple webcams with low resolution sensor (0,7 Megapixel) have been used.

Figure 3.1: Camera model

In figure 3.1 there is a picture of the webcam used in this work, both cameras were identical (same manufacturer and same model) but as explained before an intrinsic matrix for each is required due to manufacturing imperfections. It only has an USB interface to connect it to the computer and a bracket to clip it to the monitor edge.

3.1 Software and programming language

The calibration process is based on a *Python* [6] algorithm that uses both *OpenCV* [7] and *NumPy* [8] libraries. *NumPy* is widely used to manage complex mathematical operations, to perform special functions on single or multi-dimensional arrays and to manage and store the data retrieved from the various calibration processes. *OpenCV* library is used to perform the core of the calibration process, this library is a very powerful computer vision library including various functions related to calibration processes and different tools that help during the development. Also *Matlab toolboxes* [9] [10] have been used during this thesis, this software provides two simple and easy-to-use app designed to help and improve speed of the calibration process (both intrinsic and extrinsic). The results obtained using *Matlab camera calibration toolbox* have been used to do a comparison with the results obtained during this work. This comparison is useful to verify the correctness of the obtained results.

3.2 Calibration method

To retrieve the intrinsic parameters of a camera there are different approaches. The most common method is to use a chessboard, the column and the row of the chessboard are for sure perfectly straight, so the accuracy of the algorithm is improved since the detection procedure of the corners is simplified by the high contrast of the different square of the chessboard (according to [11]).
The corner detection routine is based on the Harris Corner Detection, that is a frequently used technique in computer vision field; this approach consists in the detection of edges and corners of different shapes in an image. This algorithms is able to detect a corner as a junction of two edges since when the two edges join the brightness and other parameters in that zone change suddenly.

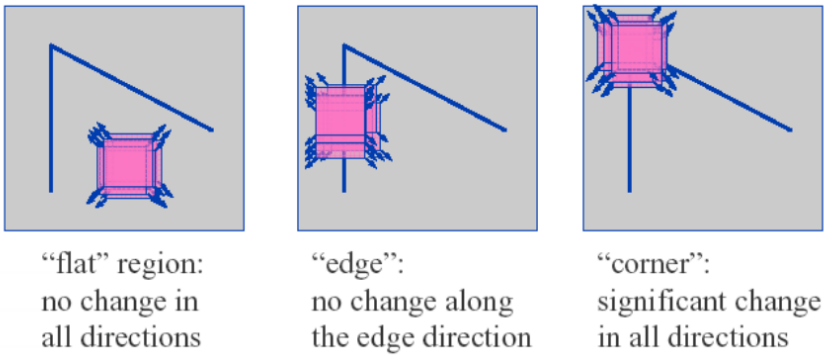


Figure 3.2: Harris Corner Detection Fundamentals

To better explain the working principles of this approach (according to [12]) in the figure above 3.2 is clear how the algorithm is able to distinguish:

- flat regions (absence of differences in the neighbouring areas)
- edges (no changes along the edge direction)
- corners (evident disparities in every direction)

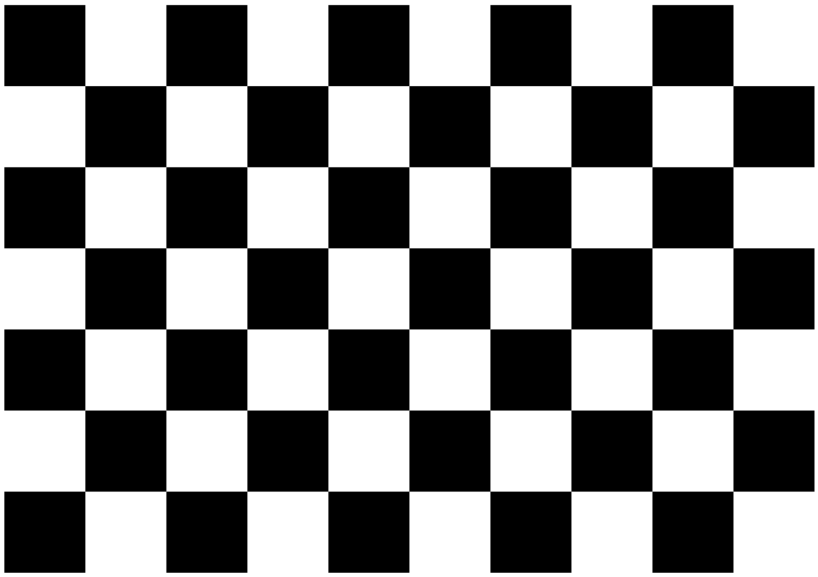


Figure 3.3: Reference chessboard pattern

To improve the accuracy of the chessboard's corners detection some tricks has been used (according to [13]), each of them slightly improved the result. The chessboard shall be such that its orientation is univocally recognizable. To achieve this the chessboard used must have an even number of column (as in figure 3.3, to obtain the last column with less black squares than the first one.
By being able to do so while taking pictures the chessboard can be freely rotated and moved in the space to improve the coverage of the field of view of the camera resulting in an improved consistency of the calibration without affecting its quality. The calibration must be performed at the same distance than the one required by the final application of the device, so in this case the distance between the camera and the chessboard in every image ranges from 50 centimeters to a few meters. This is a very remarkable trick because the calibration is slightly affected by the distance of the chessboard from the device; so if the procedure is done using a dataset that simulate the real working conditions, the parameters obtained are

more accurate and reliable. Since the camera used has less than one megapixel sensor the pattern of the chessboard was not too fine otherwise the corner detection wouldn't be very precise.
While taking pictures the light was diffused without any strong source of light pointing directly to the chessboard otherwise the black squares can reflect the light and affect the detection of the corners.

At the end of the acquisition process every image has been checked individually to remove every picture that has ambiguity, imperfections, reflections or blurred areas.

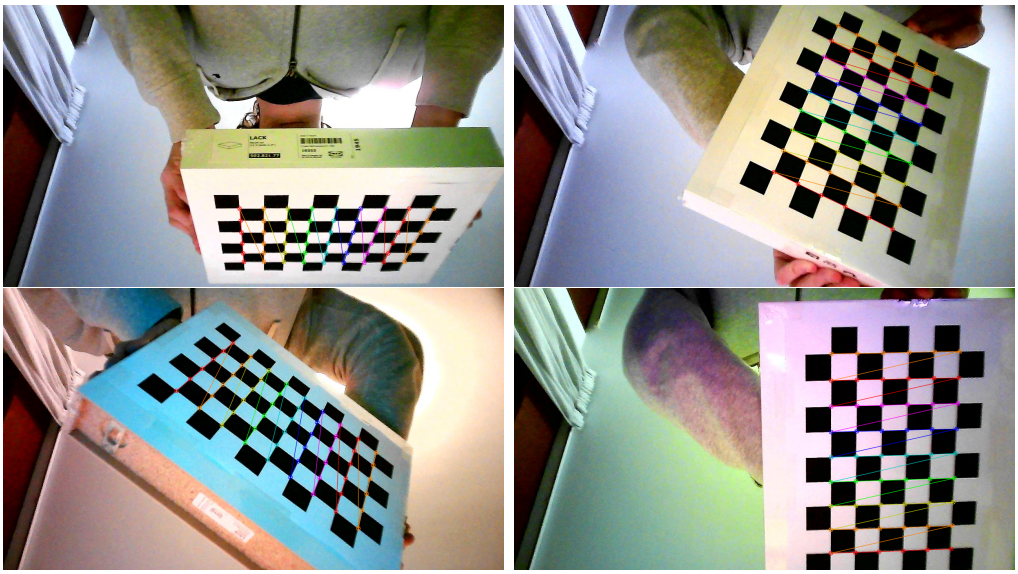


Figure 3.4: Example of four chessboard corners detection procedure

In the figure 3.4 it's noticeable how the corner detection algorithm puts a marker on every corner of the chessboard and that the first row of the chessboard is always identified with the red line. This phenomenon confirms that the algorithm is correctly recognizing the chessboard orientation independently from its rotation.

The acquisition process of the samples has been done using a wooden plank, that is a stiff material with a flat surface. An A4 sheet of paper on which the chessboard of the figure 3.3 has been printed was glued on one face of the wooden board. A simple algorithm that takes one picture every 3 seconds has been implemented in python to perform in an automatic way the acquisition. Per intrinsic calibration has been taken 500 pictures, in every of them the chessboard was moved and rotated sequentially, so the difference from one picture to the following was not too big. Therefore the script also managed to remix the images and to randomize their order to avoid the possibility that the bias-phenomenon could affect the calibration process. Every image was stored inside a folder and numerated sequentially to fill the dataset that is needed in the core of the calibration process.

3.3 Algorithm steps

The script to obtain the intrisic matrix is made of different functions (mainly implemented using the *OpenCV* library). First of all the initial parameters are loaded:

- Accuracy level, definition of the accuracy level or the number of reiteration required before ending a reiterative process
- Chessboard layout, number of rows and columns of the chessboard used
- Dataset path, the directory where the dataset containing the images taken is

Then the algorithm proceeds by loading the first picture, then it go on with chessboard corner detection. If every chessboard corner (in this case 9 rows and 6 columns) in an image is detected the coordinates of the positions are saved (a graphical representation of the corner detection is available in figure 3.4) and passed to another function that thoroughly scan around the coordinate of the corner's pixel to refine its position in order to improve the calibration accuracy. When every image has been scanned, the script find every parameter needed to obtain the intrinsic matrix as previously descripted in the Chapter 1. When this process is completed, using *NumPY*, the intrinsic matrix and the distortion informations are saved into a *.npz* file, that will be loaded later to calibrate the stereo camera setup. When the calibration process is done, the script offers a preview of one image undistorted to check if the calibration process is correct and at the end it calculates the final reprojection error. This parameter allows to understand the quality level of the calibration process.

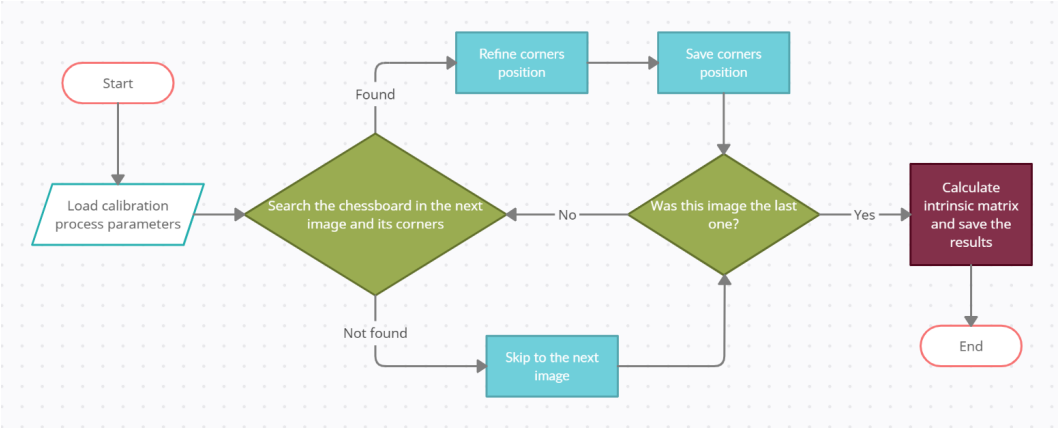


Figure 3.5: Algorithm flow

In the figure 3.5 is graphically described the path followed by the algorithm to perform the corner detection of the images contained in the dataset and to obtain the parameters needed (according to [13]).

In parallel the same calibration is done using *Matlab* and its *Camera Calibrator App* to obtain a gold standard to compare the results obtained through this python script.

3.4 Results obtained and correctness check

To measure the accuracy of the obtained results a measurement of the average reprojection error has been done. The reprojection error is defined as the distance between the detected keypoint and the real world coordinate of that point of the same picture expressed in pixels. This value must be kept as low as possible to obtain the best calibration parameters. The python script recalculates the average reprojection error everytime an image is processed, in this way it is possible to analyze the number of samples vs. mean error characteristic. To highlight how the bias effect could affect measurements only during this test has been used 4 different datasets made of 500 pictures each, with the chessboard positioned as following:

1. Near enough to cover at least 25% of the field of view of the camera, called near-field dataset
2. A few meters from the acquisition device, called mid-field dataset
3. Very far from the camera position, about 6 meters, called far-field dataset
4. The dataset containing one third of each of the other datasets in a non-sequential order, called mixed-field

Figure 3.6: Example for near/mid/far-field dataset

In the example in figure 3.6 it is clearly visible how in the three datasets there is a massive difference in how much field of view the chessboard covers. The mixed field dataset aims to remove the possible bias effect introduced by this constant distance of the chessboard pattern from the camera. Due to bias effect and also to the coherence with the distance from the camera and the reference object of the final application, the mixed-field dataset will be the one used to obtain the camera calibration parameters. For these reasons this set of parameters will be used later to proceed with the camera calibration procedure.

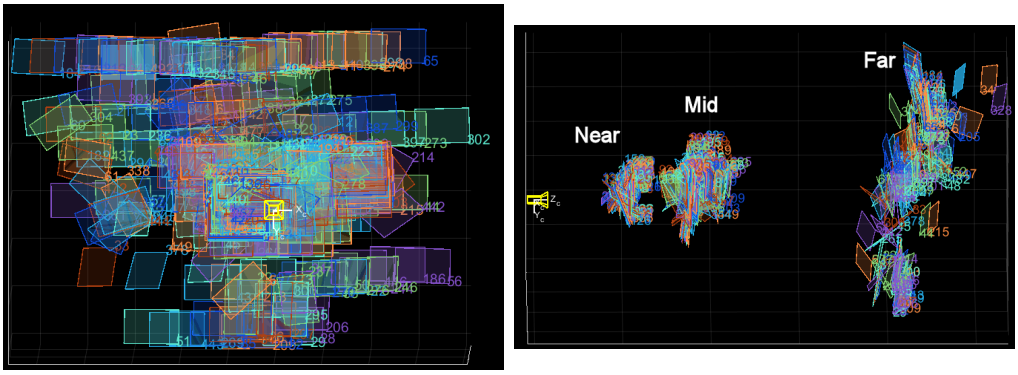


Figure 3.7: Two views of the 3D representation of the chessboard positioning in the dataset

In the figure above 3.7 there is a 3D reconstruction on two sides of how the chessboard was oriented and positioned in the images contained in the mixed-field dataset. These views are made on the XZ plane (sideview) and XY plane (rearview), the yellow cone represents the camera and every coloured square is an image containing the chessboard pattern. It's clear how it is composed of images where the distance of the chessboard from the camera ranges. Using this dataset is possible to obtain a reliable mean projection error, since it is as much coherent as possible with the field application of this work during the next steps.

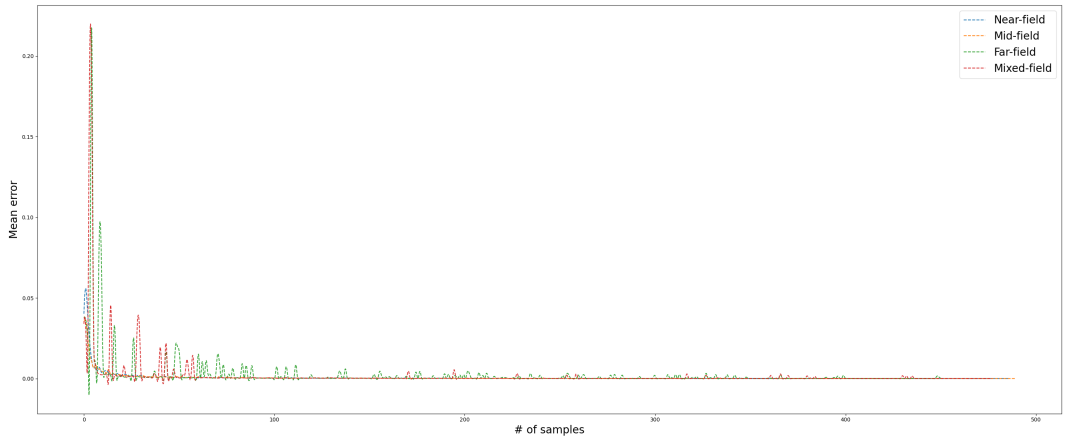


Figure 3.8: Mean reprojection error for each dataset

In figure 3.8 it's noticeable that the far-field dataset suffers more than the other one since the chessboard is further away than in the other datasets, in this case also the low resolution camera affects this measurement. Since the interest is focused on the mixed-field dataset, it is clear how it becomes stable starting from 60 samples, so that probably would be the minimum number of images required to calibrate the camera with this method.

By zooming on the range starting from 50 to 200 samples (as in figure 3.9) it's evident that the mixed-field dataset reached a stable value below 10^{-3}.

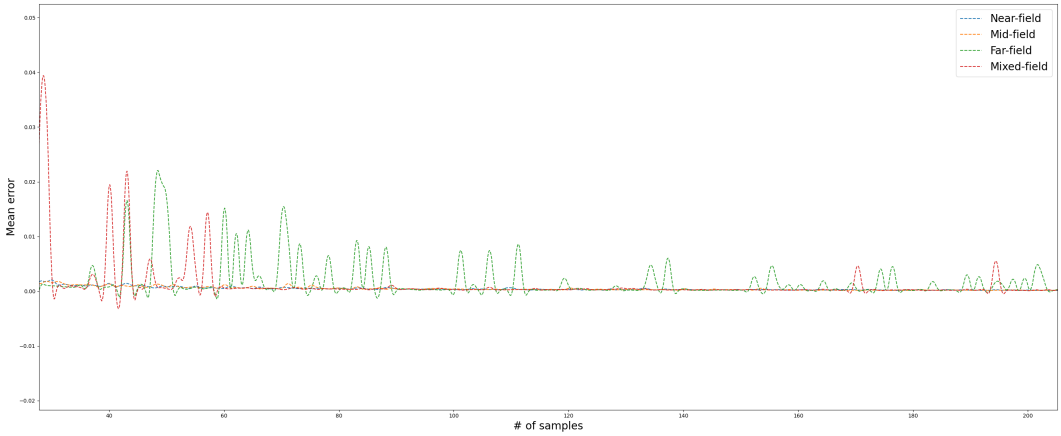


Figure 3.9: Zoom of the mean reprojection error characteristics

The intrinsic matrices and all the other results obtained using two different methods (*Matlab* and *python* script) are very similar. The RMS (root mean square, also called quadratic mean, is the square root of the mean square) reprojection errors obtained is listed below:

- *Matlab* $\rightarrow \pm 2 * 10^{-3}$
- *Python* script $\rightarrow \pm 10^{-3}$

In figure 3.10 there are two different images representing the output of the undistortion of the same image done with the two different methods and it's clear how they look almost identical.

Figure 3.10: Matlab (left) vs python (right) undistortion on an example image

In the figure below 3.11 there is the result of the subtraction between the two previous image in figure 3.10. The green marks stand for the areas where there are differences between the two images. A small difference is always present since the *Matlab* image is extracted manually with the windows snipping tool (this introduces

an error) and the two rectification algorithms resize the image slightly differently, so the two image has different size (in pixels).

Figure 3.11: Result of the difference between the two images highlighted in green

In the picture the green areas have been enlarged to be easily identified in the photo.

It is assumable that the two methods are identical so the *Python* script implemented can be considered as valid as the *Matlab* one after this comparison. A further check of the correctness of the results obtained can be done by comparing the rotation and translation informations obtained during the calibration process with the 3D world.

In the figure 3.7 there is the 3D plot of the room where the images were taken. To be sure that the rotation and translation informations were accurated, this room has been measured and there is an error that is surely below the manual obtainable measurements accuracy (around 1 millimeter).

After all these considerations, it's assumable that the correctness of the results obtained using the *python* script is verified, so these data can be used with confidence for the next steps of the calibration procedure.

Chapter 4

Stereo calibration

When the setup is made of two (or more) cameras the point of view of the system is not unique as in a single camera setup, but is composed by the two different sensors that are part of the stereo-setup. The stereo vision is one of the core topic of the computer vision field, it consists in the extrapolation of information of the 3D world from 2D images (in this case a couple of images, one for each camera). This technique aims to obtain three-dimensional data from bi-dimensional images by exploit the two different vantage points. The setup of a stereo vision consists of two cameras aligned on one axis (horizontal or vertical one). This layout recalls the human eyes positioning, where the two eyes can be compared with the cameras.

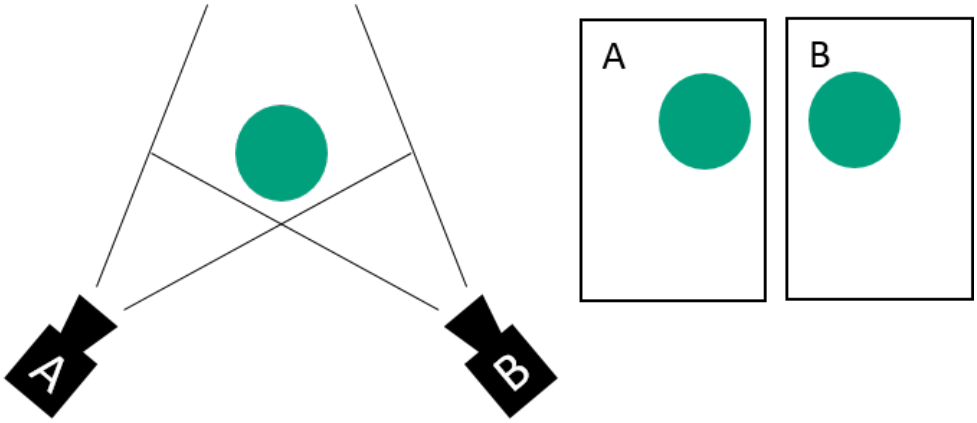


Figure 4.1: Stereo vision

In figure 4.1 there is an example of a stereo camera setup similar to the one that has been used during this work. The two cameras are placed on the same height but at different positions. In the same figure it is noticeable how the same object can be seen from both cameras only if it's in a certain area, the one covered

by both cameras. These devices partially share their field of view (exactly as the human body eyes), when a picture is taken of an object inside the common field of view (FOV), the images will report the object in two different image coordinates.

4.1 Setup of the system

As in the intrinsic calibration procedure, also this one must be performed in similar condition to the application field ones, so the setup has been installed in a large room.

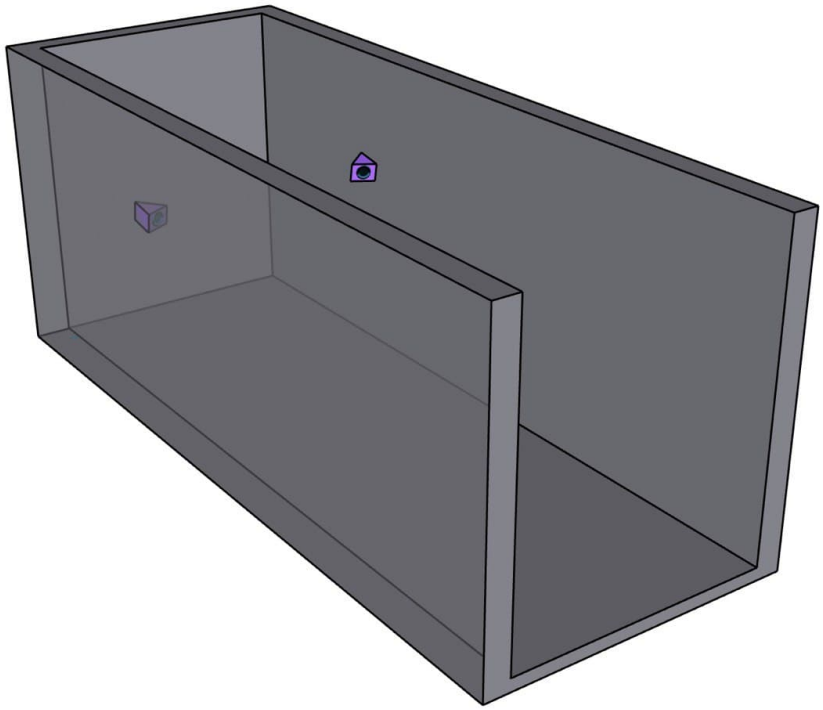


Figure 4.2: 3D View of the room

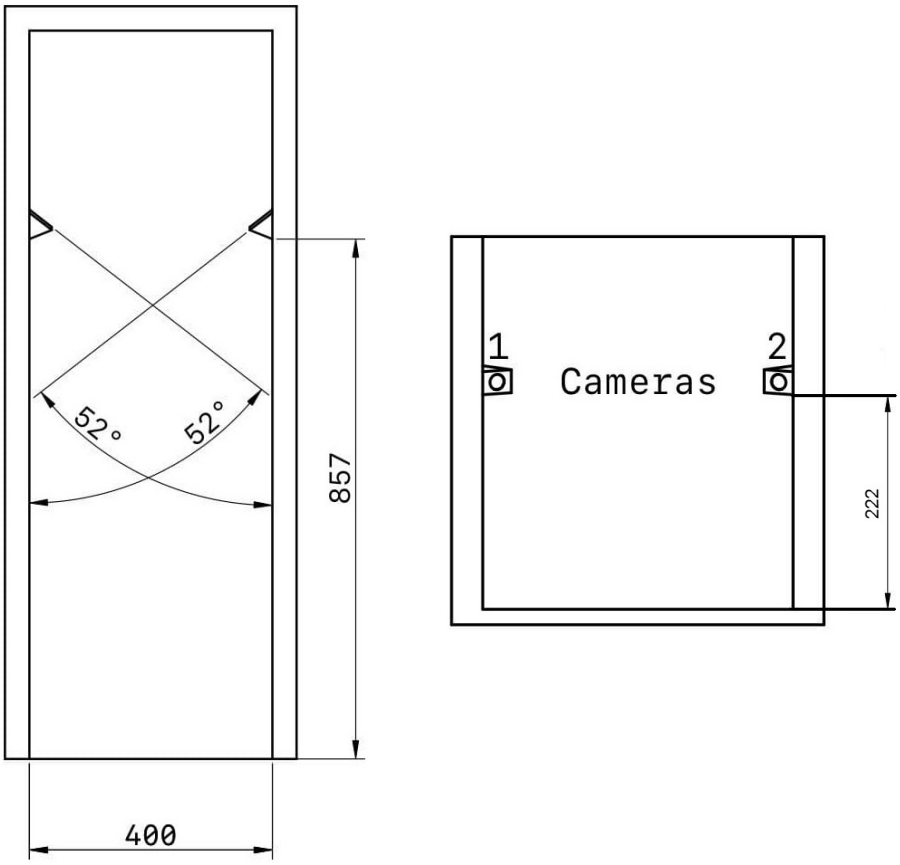


Figure 4.3: Respectively top view and front view

In this case the two cameras were set up as in figure 4.2 and 4.3. The cameras were mounted on a 3D printed bracket with an horizontal tilt of 8° each to increase the common field of view. The aim was to obtain an angle of incidence of the two lines in the center of the field of view of 60°, in order to maximize the common field of view. After all the adjustments and the mounting procedure the incidence angle between the two center of the FOV of the two cameras is approximatively 58°.
In the above figure 4.4 there is the FOV of each cameras highlighted to better understand how the setup was mounted and which area was covered by the two cameras. Between the two cameras, outside the field of view of each cameras, there was a laptop connected through usb to both cameras simultaneously to perform the data acquisition. The fixing and the stability of the cameras are very important since when the system is calibrated it must be still in that position otherwise a new calibration is mandatory.

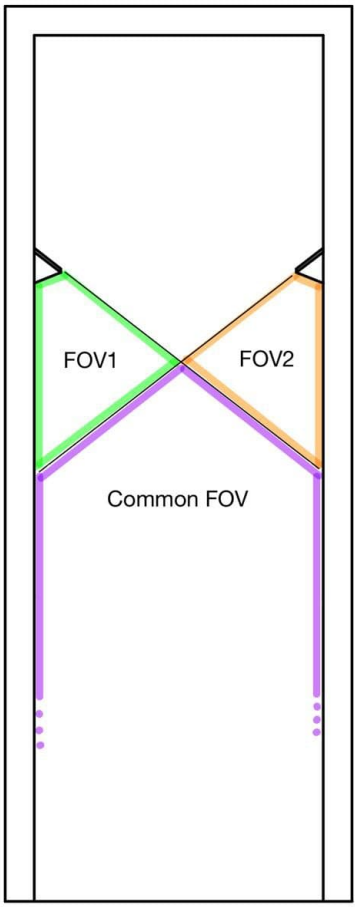


Figure 4.4: Cameras field of view

Cameras was bolted using two screws for each and also the cable has been fixed with a cable tie to the bracket, to prevent that the connection and disconnection movements of the USB connector (and its cable) can move the cameras.

4.2 Images acquisition

After the setup installation, it has been connected to a computer to acquire the dataset. In a *python* script has been implemented the required algorithm to obtain the pictures from both cameras. To cover the maximum height of the common field of view, the chessboard has been raised and held in position manually by a person, so the chessboard wasn't perfectly steady. To solve this problem and obtain the maximum accuracy the images are taken with a script so they are taken simultaneously, so any little differences between the images of the two cameras

during the chessboard exposure is filtered out, in addition this method is also a lot faster than the manual acquisition. The script also manages to rename and place in the correct directory the images to identify which camera shoots each image. The same trick used in the intrinsic calibration procedure to remove the bias-effect has been applied also in this case, so the sequence of the images has been randomized by the script.

Figure 4.5: Points of view for the same scene of the two cameras

In the figure above 4.5 there is an example of the same scene captured by the cameras of the setup used during this work.
During this calibration, instead of using an A4 sheet of paper as for intrinsic calibration, the chessboard has been splitted in two parts. These has been printed on an A3 sheets that have been joined together to form a sheet with the same width of an A3 but with doubled length. On the edge of this sheet there is a white margin to help the algorithm to locate correctly the chessboard position and to avoid wrong detections. As for intrinsic calibration the chessboard is mounted on a wood plate to ensure its flatness and to avoid any distortions due to the non-planarity of the chessboard.

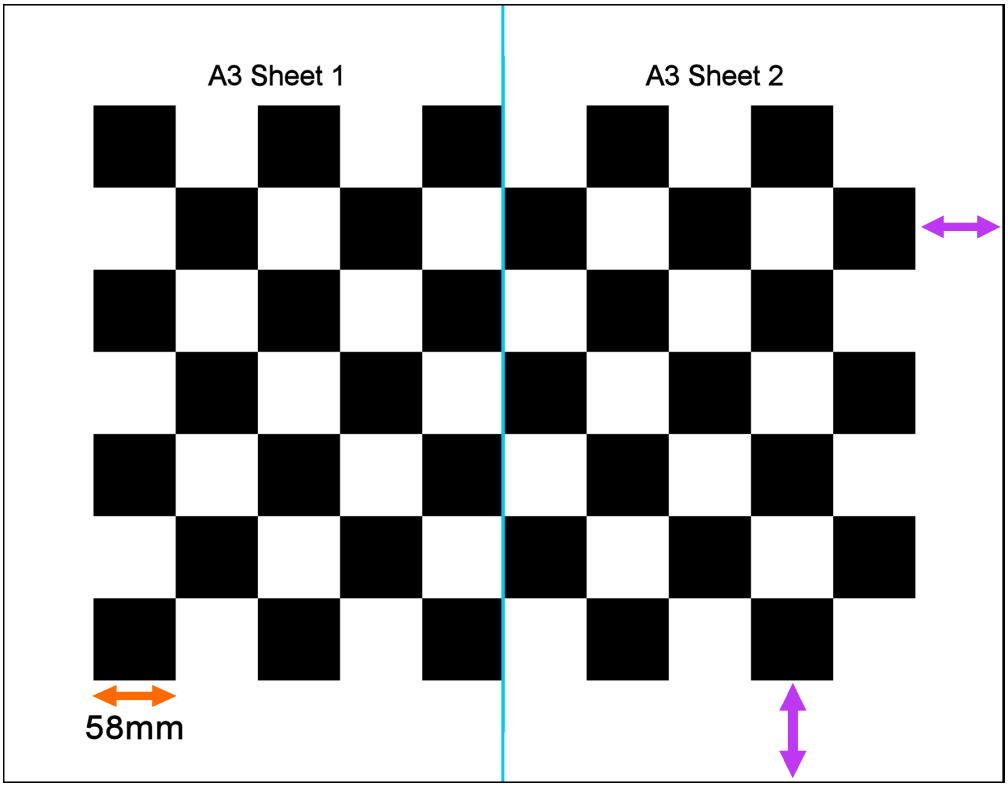


Figure 4.6: Chessboard setup used for the extrinsic calibration

With this enlargement of the chessboard the corner detection algorithm can easily detect the corners of the chessboard, because the squares are bigger, in this particular case (as in figure 4.6) the squares were of 58mm each and the margin (marked with the purple arrows) is at least 50mm. The light blue line that crosses the whole image is where the two sheets have been joint.
The acquisition of the images has been done by placing the wood plate on a chair and on the ground and by holding it at 2 meters of height, in each case the chessboard has been tilted in different angles.

Figure 4.7: Three examples of the chessboard position during this procedure

4.3 Fundamental matrix calculation

The fundamental matrix is able to relate the position of one camera with the other one in a stereo-camera setup. Using one camera as the reference point, it is able to describe the position of the other one. Independently from the method chosen to obtain the fundamental matrix, the relations that composes the fundamental matrix are the same. Assuming a stereo camera system that has in the common field of view one point with 3D coordinates P_w, it is assumable that:

$$P_l = R_l P_w + T_l \qquad \text{and} \qquad P_r = R_r P_w + T_r \tag{4.1}$$

where:

- P_l are the P_w coordinates seen from the left camera
- P_r are the P_w coordinates seen from the right camera
- R_l and R_r define the rotation in the 3D space between the left and right cameras (respectively) and the reference object
- T_l and T_r describe the distance between the device and the calibration item

The R and T parameters describe which is the relation between the cameras and the object that is used to calibrate the cameras.

Choosing the left camera as the reference one, below there is the mathematical procedure ([14]) to obtain the R and T parameters starting from the left camera relations and including it in the right camera ones:

$$R_l^T (P_l - T_l) = P_w \tag{4.2}$$

This result is inserted inside the right camera equations.

$$\begin{gathered} P_r = R_r R_l^T (P_l - T_l) + T_r \\ P_r = R P_l - R R^T (R T_l - T_r) \\ P_r = R P_l - R(T_l - R^T T_r) \\ \text{considering} \qquad R = R_r R_l^T \qquad \text{and} \qquad T = T_l - R^T T_r \\ P_r = R(P_l - T) \end{gathered} \tag{4.3}$$

The last relation describes P_r in function of P_l, so it correlates the view of the left camera with the right one.

4.4 Methods analysis

The algorithm in this case is made using the same libraries that were already used for intrinsic camera calibration. To obtain the extrinsic parameters of the stereo setup there are mainly three different methods proposed by *OpenCV* ([15]):

1. Using a 3D well-known object it is possible to obtain a set of extrinsic parameters of one camera related to the object analyzed, to perform this method the required function is *solvePNP*()

2. Performing the individual camera calibration or obtain the data from a previous calibration (the intrinsic calibration performed before) and use the sets of data obtained from this procedure to calculate a single rotation and translation relation between two cameras, this method can be implemented using the function *stereocalibrate*()

3. Obtaining the fundamental matrix from couples of images where seven or more points can be matched, to do this the core function is *findfundamentalmat*()

Each of these options has been considered to understand which one can better perform in this exact field of application.
Applying the first method (solvePNP()) to the case analyzed during this work is not recommended. As the matter of fact the aim of this thesis is to obtain a unique relation (the fundamental matrix describing both translation and rotation parameters) of the two cameras in the 3D space between them and not a set of data that describes the spatial relation between the camera and the calibration object. In addition, since this method requires to perfectly know the dimensions of the reference calibration object, when at the end of the thesis this method must be applied to the human body, this can't be done because every human body differs from the others.
The second method analyzed performs internally the calibratecamera() function to obtain the intrinsic parameters. This is not recommended (as also reported in the *OpenCV*'s documentation) due to the high dimensionality of the data space that can lead the function to obtain a solution way different from the right one. *OpenCV* library provides different flags for this function, one of them allows the user to insert manually the intrinsic parameters (CALIB_FIX_INTRINSIC), it's highly recommended to singularly calibrate each camera (as done in the previous chapter) and then insert these parameters inside the stereocalibrate() function using the specific flag. This function uses an iterative approach to obtain the minimum reprojection error for both cameras. Also in this case (as for solvePNP) it's required to know the dimensions of the reference object to obtain the calibration.
The last method analyzed avoid the computation of the intrinsic parameters of the camera and insert them directly inside the fundamental matrix. This can be

obtained by matching seven (or more) points in an images pair using the following method:

- 7 Point Algorithm, this approach is based on matching 7 points on the couple of images
- 8 Point Algorithm, works as the previous but matches 8 or more points
- Ransac, this method use the random sample consensus algorithm
- LMedS, that use the least median of squares algorithm

To obtain the extrinsic matrix using the chessboard as reference pattern the stereocalibrate() approach will be used ([16]), since it's the most accurate and it can't be considered as a gold-standard for the comparison during the next steps. Of course the stereocalibrate approach won't be used in the next chapters, because it implies to know the dimensions of the reference object and the aim of this work is to use the human body as reference pattern that haven't any fixed dimension.

4.5 Algorithm steps

As for the intrinsic calibration of each camera, a *python* script has been implemented to perform the entire calibration process after the data acquisition procedure. At the beginning the script loads the image dataset, the intrinsic parameters of both cameras and the calibration parameters (number of iterations and other parameters related to the desired accuracy of the procedure). When everything is correctly loaded, one pair of images at a time is loaded. The two images represent the same scene seen from the two different cameras, one at a time they are scanned. The algorithm checks if the chessboard is present inside one of the images, if this is confirmed then the chessboard research can start in the other image, if also in this one the presence of the reference object is verified, the algorithm starts to compute the corners coordinates (in the same way as the intrinsic calibration). When the keypoints positions are saved in the specific variable, the stereocalibrate functions is evoked to perform the operations required to obtain the fundamental matrix and other parameters related to the extrinsic calibration (eg: rotation and translation parameters). This function obtains as input the intrinsic parameters of both cameras and the detected corners positions. At the end the obtained values are stored through NumPY in an .npz file.

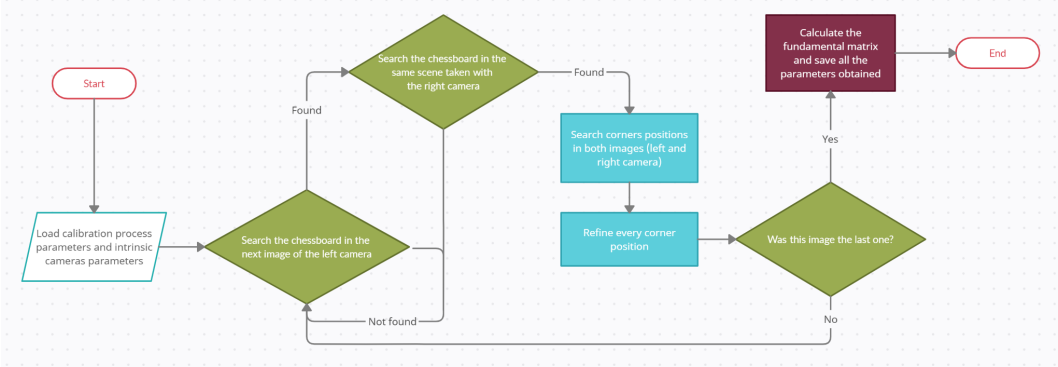


Figure 4.8: Description of the algorithm steps to obtain the fundamental matrix

In the figure 4.8 there is the explaination of the method used by *python* algorithm to obtain the fundamental matrix step by step ([17]). It is noticeable that if the chessboard recognition procedure fails on only one image of the pair both are discarded.

4.6 Results analysis

When the calibration procedure is done using both method (*Python* and *Matlab*), the results are analyzed to compare them directly and their rms error.

$8.4004 * 10^{-7}$	$1.0895 * 10^{-5}$	-0.0034
$1.0599 * 10^{-5}$	$-1.4042 * 10^{-6}$	-0.0519
$3.4611 * 10^{-4}$	0.0423	1.0175

Table 4.1: Extrinsic matrix obtained with the Matlab camera toolbox

$8.7408 * 10^{-7}$	$1.1513 * 10^{-5}$	-0.0035
$1.1480 * 10^{-5}$	$1.6224 * 10^{-6}$	0.0557
$3.5181 * 10^{-4}$	0.0458	1.0000

Table 4.2: Extrinsic matrix obtained with the python algorithm

As it is noticeable in the tables 4.1 and 4.2, that represents the fundamental matrices obtained respectively with *Matlab* and the *Python* approach, the results achieved are very similar and the differences between the two different parameters are very small. This tolerance is supposed to be present due to the implementation of the algorithm (different programming language for the two method, *Matlab* use *C++* version of *OpenCV* while the script implemented in this thesis uses *python*). The values obtained can be considered consistent since to double check the results a smaller dataset has been acquired in different conditions and the results were

always very close between them.
Considering the two sets of calibration parameters obtained with different methods as equivalent, it is required to check if the accuracy of the results is high enough to be able to obtain an high quality calibration. To fulfill this quality requirement the mean error in pixels is analyzed.

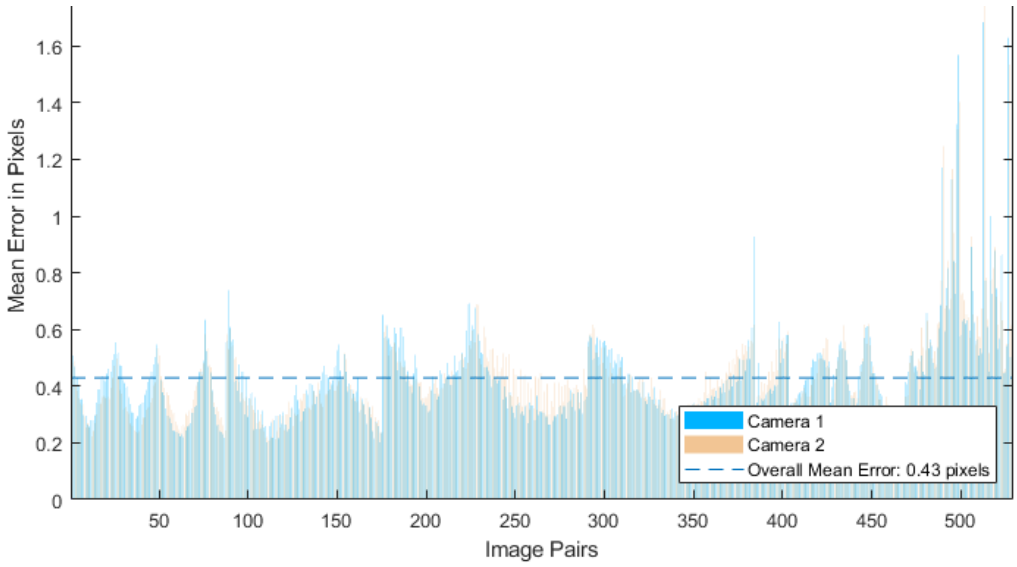


Figure 4.9: Mean error measure in pixel (Matlab)

In the figure 4.9 is reported the mean error vs number of samples obtained in this procedure by the *Matlab* script. As it is noticeable in the picture, the mean error is lower than an half pixel.

$$e_{pixel} \approx 0.43 \tag{4.4}$$

Also in the python script the mean error is calculated to be able to compare this procedure also in terms of pixel accuracy with the *Matlab* one. The algorithm implemented for this thesis work obtained:

$$e_{pixel} = 0.5548993081075833 \approx 0.55 \tag{4.5}$$

In this case the mean error is slightly higher than the *Matlab* one (reported in equation 5.5) but anyway it's acceptable since this parameter is evaluating that the calibration error is slightly higher than an half pixel.

The last check of the correctness of the results obtained is done using the inverse procedure (as done for the intrinsic calibration). Using the obtained parameters the pose of both cameras is estimated in three dimensions, in this way the room where the setup has been built is reconstructed using the camera calibration parameters. If the reconstruction is coherent with the real world setup, this is another check of the integrity of the results.

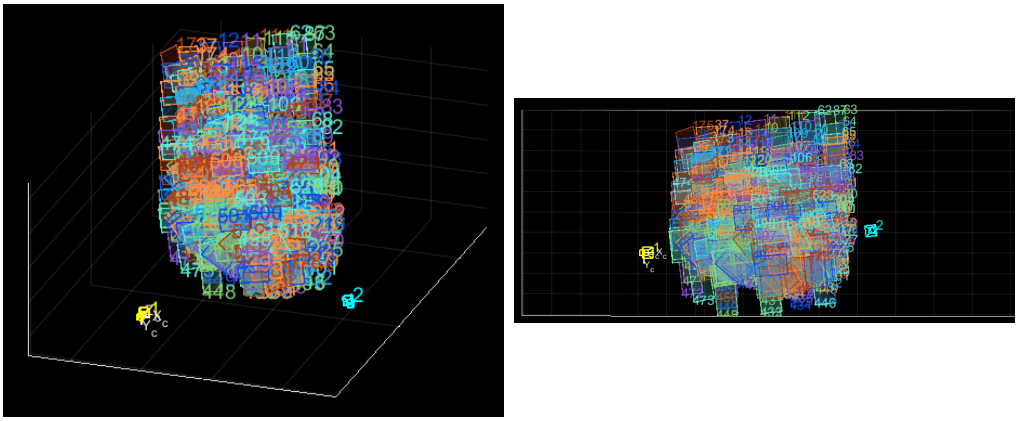


Figure 4.10: 3D Reconstruction of the camera setup vs chessboard positions

In the figure 4.10 the two icons with the shape of a cone are the cameras; they are represented in yellow and light blue.
It's noticeable how the cameras and the chessboard position in every image of the dataset are distributed in the room. As explained before, the two cameras have been placed at (almost) the same height and their field of view is tilted inward. This is coherent with how the setup was configured in the real world (described in the second paragraph of this chapter). After these considerations, it's assumable that the result obtained with the python algorithm is as accurate as the *Matlab* one. Since *Matlab* is the gold-standard reference, the procedure implemented in this work to find the extrinsic parameters is on the same level as the reference one.

Chapter 5

Human pose estimation

Human pose estimation is a technology based on the computer vision that is able to estimate the configuration of a human body pose captured in an image. This technology has been studied for more than 20 years and its relevance is due to the number of field that can obtain advantages from it. In this case this sector of the computer vision is important because it is the basis of the marker-less MoCap (motion capture). Still nowadays this sector brings along with it a lot of problems because:

1. Every human body is different from another
2. The body can appear in every orientation
3. Some limbs can be occluded in the image by an object or because of the orientation of the body itself
4. Loss of 3D information caused by the analysis of a 2D image

These are only a few problems that must be considered when is required to have to deal with the human pose estimation. During the human pose estimation process ([18]) there are multiple types of inputs possible; they are the following:

- RGB (red-green-blue), the images in this case are standard images that are obtainable with every camera (this input type is more convenient in terms of mobility and costs)
- Infra-red, where every pixel of the image captured depends on the quantity of infrared light reflected back to the camera from the object, this method requires very expensive devices and setup

Figure 5.1: Infrared view of an human body with multiple IR markers

- TOF (time of flight), this input type has pixels that depend on the distance between the camera and the object represented; instead of normal image this type of input returns a depth-map

Figure 5.2: RGB (left) and depth-map (right) of the same scene

During this work only RGB inputs are used since the final aim is to obtain an affordable and easy to setup system. The detected body can be modeled in the following ways:

- Skeleton, the body is described through different keypoints that usually correspond to joints (eg: wrists, elbows, knees), this model can be applied in a lot of field both 2D and 3D because it is very flexible (the number and the type of joints can be modified very easily)
- Volume, this model represents the human body as a 3D structure, typically used in 3D scanning applications
- Contour, in this case the human body is detected as a countour, this method obtains the perimeter of the body that defines its silhouette

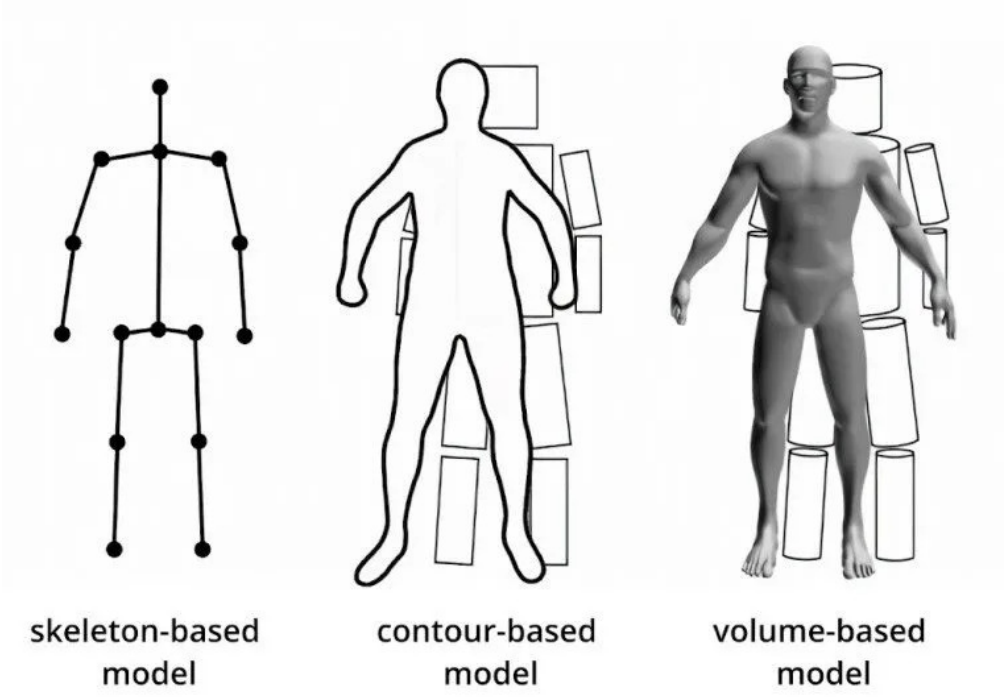


Figure 5.3: Different modelizations of the human body

The most common model used for 3D human pose recognition is the skeleton one, it consists in the detection of the keypoints on the human body and their representation to obtain the skeleton.

5.1 Previous similar attempts

Some papers deal with the subject of the multi-camera calibration process using the pedestrian recognition, but most of these face this process using the human detection and the description of the body position using a bounding box. In this

process the algorithm track an human body and encapsulates it in a box to describe its position.

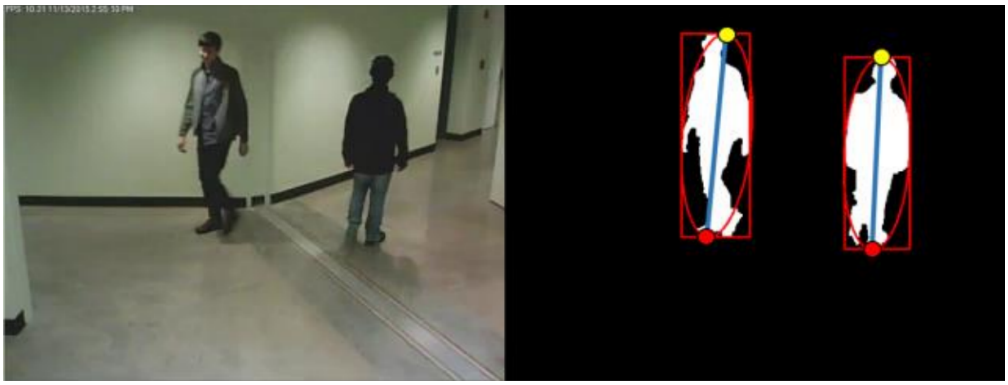


Figure 5.4: Example of human bodies detection with their bounding boxes

In the figure above 5.4 there is an example of how the bounding box approach works, this image is taken from the paper available at [19]. Every human body detected inside an image is surrounded by a box to describe its position. In most of the previous papers available, the skeleton was described with two points, the top of the head and the feet. These two points are very difficult to be defined with enough precision since they aren't perfectly defined. This approach is less accurate due to the keypoints analyzed and also slower to reach the results; this last disadvantage is caused by the higher number of frame necessary to obtain enough keypoints for the calibration. Using the human pose recognition it is possible to obtain at least about 8 keypoints per frame while with this bounding boxes approach every image contains only 2 keypoints. More information about these techniques are available at these papers: [20], [19], [21]. The pose recognition as currently implemented was impossible to consider years ago due to the too high computational complexity; since now the hardware available is capable to handle this computational load, the usage of this new approach can lead to some useful advantages.

5.2 Datasets

To estimate the pose of a human body there are different dataset that can be used, one of the most common is called *COCO* ([22]) and it's also public, for these reasons this is the one that is mainly used inside this thesis work. *COCO* is defined as a large-sale object detection, segmentation, and captioning dataset; this is called dataset because it contains 80 categories of pictures and 250000 images of different scenes containing a lot of different recognized object (person, food, pet, boat, phone, clock, bike, ...). This collection of images is one of the widest, because of this large amount of samples it's able to find inside an input picture a lot of different items

already catalogued in the dataset. In the picture 5.5 there are two images extracted from the *COCO* dataset, as can be seen different objects are described inside the same image, their perimeters are defined and they are also described using short sentences. The top image contains a female tennis player, the racket and tennis ball are described as well as the human body of the player.

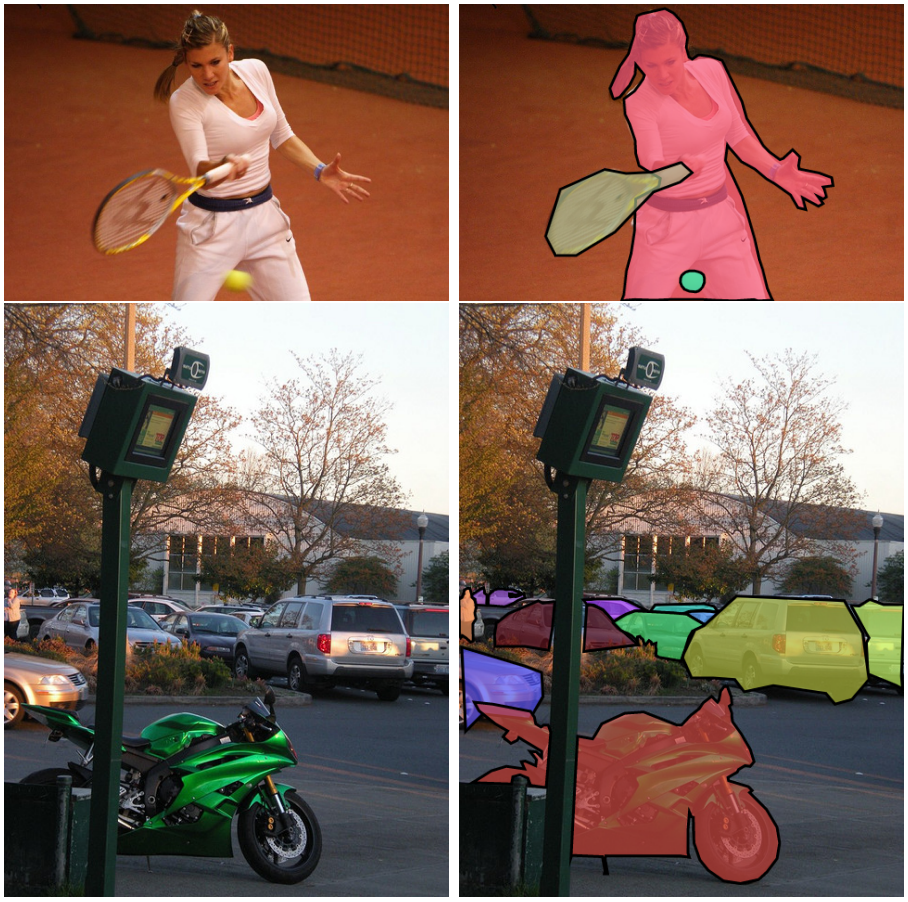


Figure 5.5: Two examples of images contained in the *COCO* dataset: raw images (left) vs image with object segmentation (right)

The couple of images on the bottom of the figure 5.5 contains a sport motorbike and some cars. These objects have their perimeter sketched onto, in this way the dataset is able to define what is a bike and what is a car. The *COCO* dataset has a lot of images, thanks to them it is able to detect an object in other images since all the images inside the dataset that contains that specific object explain to the algorithm how that object is represented with pixels in every orientation possible. All the images contained inside the *COCO* dataset are collected from Flickr, that is a website where people can share their photos with other users. For this thesis work mainly the *COCO* dataset is used to obtain the skeleton of a person, describing it

through keypoints.

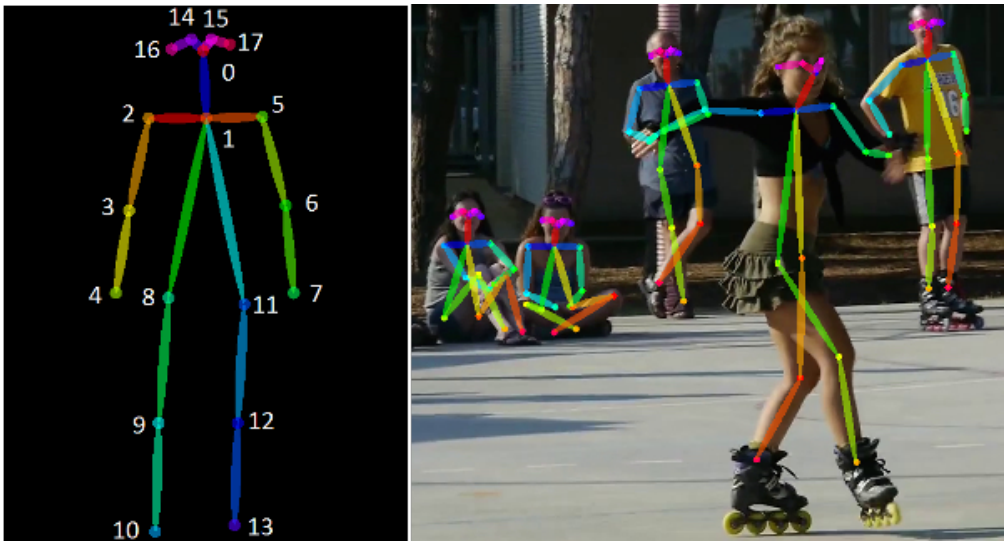


Figure 5.6: Example of skeleton-based modelization of multiple human bodies using *COCO* dataset

Also other datasets have been used to verify the reliability of the results obtained and eventually to improve them (eg.: MPII).

5.3 Pose estimation process

The pose estimation in this work is done exploiting the **deep learning**, this technology is a branch of the machine learning technologies. This approach has been widely used in a lot of field ranging from computer vision to speech recognition, is based on an artificial neural networks that is able to learn how to describe something starting from the feature detection and a fair amount of raw data, this learning process is called "**feature learning**" (more information about this approach are available at [23]).

In this work has been used a **model** of a trained neural network. This is generated by letting the neural network running through the whole dataset and to train itself. It analyzes the pictures to find keypoints and items similar to the ones contained in other examples (already analyzed) and labels it. When the model is generated and trained, it is possible to apply it to find the recognized object (in the case of this work, the skeleton keypoints) in the images sent as input in the neural network. During this work the model used is retrieved from an open source project ([24]) that offers a wide range of already trained neural network on different datasets.

5.4 Data acquisition method

After defining how the data can be processed, the same room used in the chapter above along with the same camera setup has been used to acquire the data required to perform the calibration process. Both cameras captured a video of the room while an human body was moving randomly inside the common field of view. During the recordings the person roam freely inside the room while also moving arms and legs in different ways. Since in the extrinsic calibration procedure with the chessboard it's assumable that starting from 150 samples the calibration results are accurate. In 150 samples, the number of keypoints is the following:

$$N_{points} = n_{row} * n_{column} * n_{samples} \tag{5.1}$$

where n_{row} and n_{column} stand for the number of row and column of the reference chessboard while the last parameter describe the number of samples taken in account. In the specific case of 150 samples, since the chessboard used during this work has the dimension of 9x6, the number of keypoints detected is 8100. During the human pose recognition process some keypoints won't be considered (eg: ears and eyes) since they are less precise than others, because they identify smaller features of the body; it's assumable that in every picture the algorithm will detect at least 6 keypoints on average, because sometimes it fails to detect a feature and sometimes, due to the position of the body, not every keypoints is visible from both cameras. In light of these considerations the minimum length, in seconds, of the video, considering a 30 frame per second recording, is the following:

$$n_k = n_{average\ keypoints} * \frac{Framerate}{3} * Duration \tag{5.2}$$

where the framerate is divided by 3 to take in account only one frame every three, for the reasons explained above, and n_k represents the total number of keypoints that are needed must be higher than the one considered with the gold-standard method. For these reasons the duration of the videos has been set at least at 3 minutes. During the acquisition process the light of the room is very important since it heavily affects the accuracy of the detection. After checking the time-synchronization of these two videos, a pyhton script has been implemented to be able to extract the frames from the videos. Another important parameter is the blur of the person inside the images; it is evident that if the human body is more clearly defined, the keypoints detection algorithm is able to extract more precise keypoints. In case of a blurred images the algorithm can also completely miss a keypoint. The frame extraction procedure has been done to be able to work on the videos as a sequence of images. One frame every three has been used since from one frame to the next one the differences were barely noticeable.

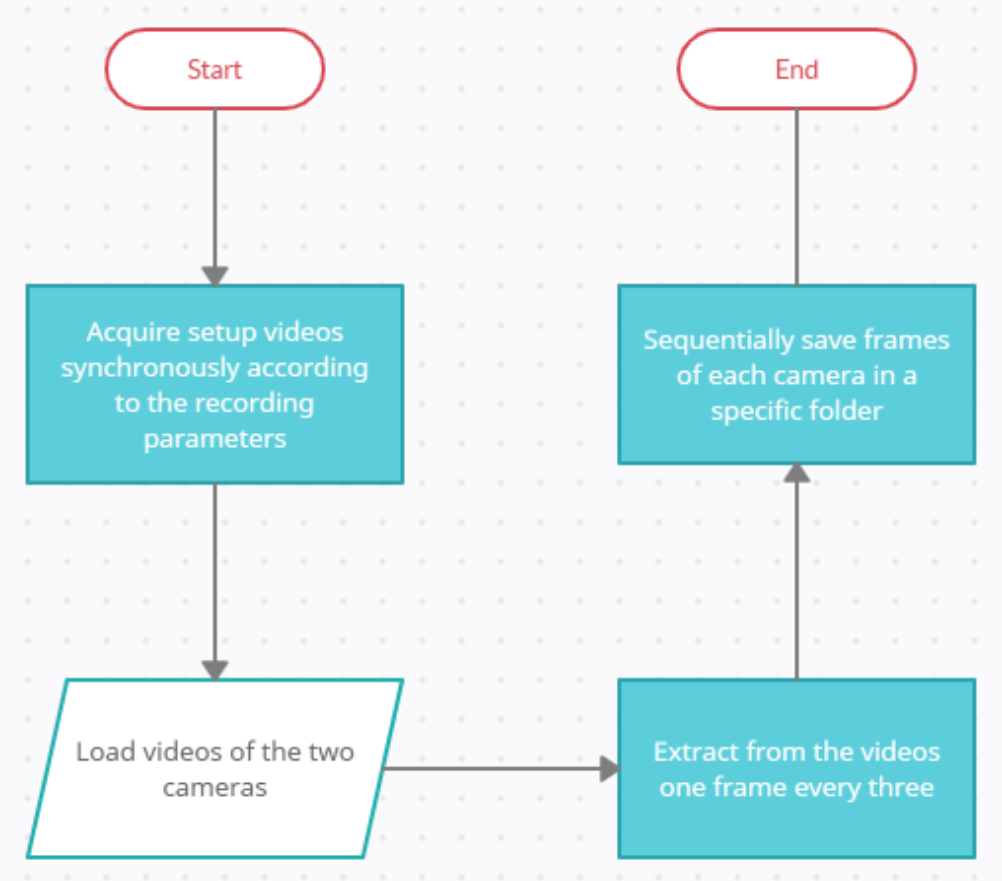


Figure 5.7: Description of the steps to obtain the required data

In the figure above 5.7 there is a description of the flow of the algorithm to retrieve the necessary data to perform the calibration.

5.5 Algorithm steps

When the frames have been extracted, before finding the coordinates of the skeleton keypoints, it is necessary to undistort every image points exploiting the OpenCV functions to apply undistortion method using the intrinsic camera informations obtained in the previous chapters.

When this process is completed the neural network starts the analysis of the images to find the skeleton keypoints based on the dataset used. During this work COCO dataset has been mainly used but to obtain a comparison between datasets also the MPII dataset has been used. In the implemented Python algorithm some keypoints were not considered since they were the most prone to error, like ears, eyes and all the other smallest features of the human body that are more difficult

to detect with enough accuracy.

Figure 5.8: Example of the keypoints of the COCO dataset used for the human body

In the image above 5.8 there is a sample image with the keypoints of the skeleton drawed onto it. It is clear how only the main skeleton structure has been used to increase the final accuracy of the detection and to avoid possible false detection of smaller features.

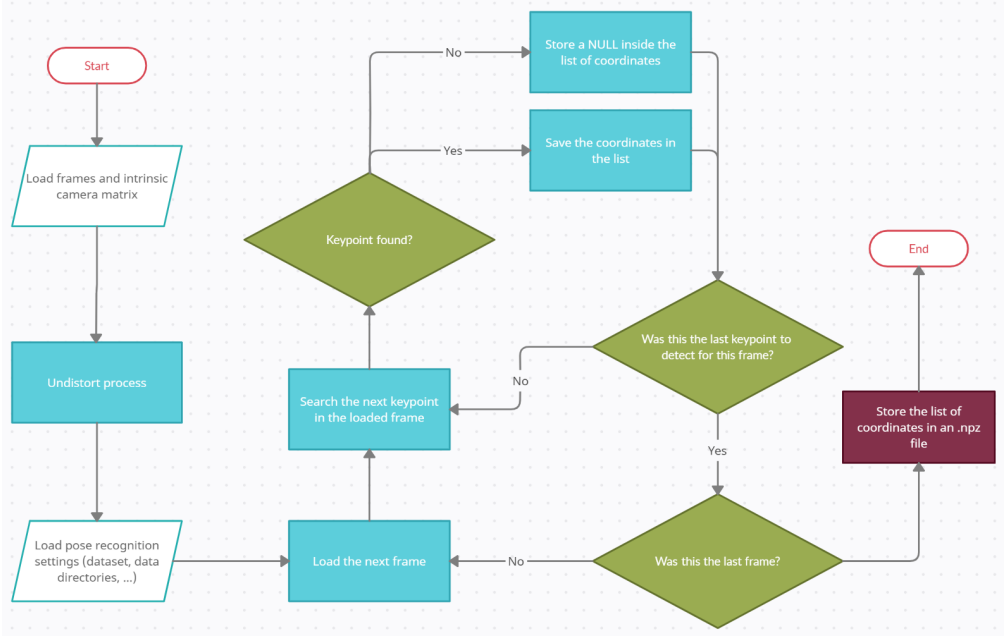


Figure 5.9: Algorithm flow

In the figure above 5.9 there is a graphical representation of the path that the algorithm follows to obtain the list of coordinates of the detected keypoints. This process must be repeated for both cameras of the stereo system.

After the analysis of the frames extracted from the videos recorded by the stereo camera setup, the python algorithm has a built-in function to remove wrong detections. A function compares the coordinates of a specific keypoint (eg.: left shoulder) in an image and in the next one; if the distance between the same keypoint from one frame to the next one is higher than a user-specified threshold, this pair of coordinates is removed. So that specific keypoint in the second frame is neglected for both fields of view. This is done to lower as much as possible the chance that some points in the surrounding environment are misread by the algorithm as skeleton keypoints. Applying this filter to the list of coordinates obtained using the COCO dataset reduced the number of wrong detections of at least 80

At the end of this correction process, another function takes care of the skeleton keypoints detected only from one point of view and not from the other one, with a setup that has convergent fields of view during a movement one leg often overlap the other one for one camera.

Figure 5.10: Different point of view with some keypoints hidden for one camera

In the figure above (5.10) there is an example where one leg is covering the other one only for one of the two cameras, due to this the left ankle is not detected for one of the field of view. In this case the algorithm during the pose recognition saves a NULL value for all the keypoints that cannot be detected. These points are removed by this function in both set of points (left and right camera).

When this process is completed the coordinates of the detected keypoints are correctly ordered and saved in a list.

To obtain the extrinsic matrix of the stereo camera setup used during this work there are different ways, as analyzed in the chapters before. In this case the function that has been used is findfundamentalmat() because it is able to obtain the fundamental matrix without using a rigid object with fixed coordinates (as in

the case of stereocalibrate()). When using this function there are different method to match points inside the image pair. Currently OpenCV is implemented in such a way that the 8-point hasn't outlier rejection (removal of abnormal measurements with an unexpected high difference with the other data), so this method is not used. Based on the RMS error that every method generates, the final choice was the RANSAC method, that if it is set with the correct threshold and confidence leads to low RMS values (detailed comparison at [25]). The side effect of the RANSAC method is that (according to OpenCV documentation) it needs about 12-13 keypoints to perform in the best possible way.

Using the list of keypoints coordinates obtained before with this OpenCV function, the fundamental matrix is retrieved and saved to be used later to verify the correctness of the results.

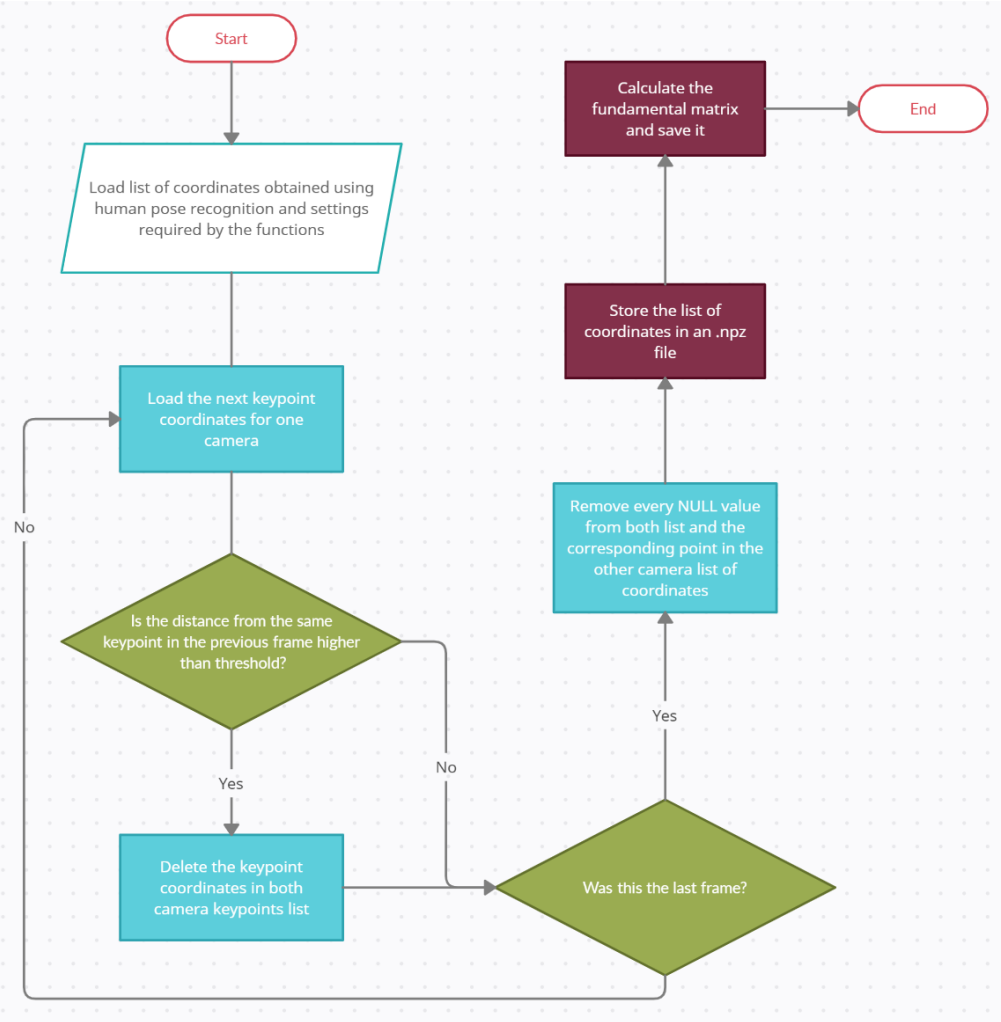


Figure 5.11: Algorithm flow

In the figure 5.11 there is the description of the path followed by the algorithm to correct and arrange the list of coordinates obtained during the human pose recognition process.

5.6 Results analysis

During a preliminary results analysis, it was clear that the MPII dataset in this case performed worse than the COCO. While analyzing the skeleton recognition on various frames there were a lot of wrong detections with MPII, at least double more than the COCO dataset. This is probably due to the fact that the test context of this work (environment, one human per image, light conditions, ...) was more similar to the informations contained in the COCO instead of MPII. Due to this reason the MPII comparison has been started, but it has been halted after the pose recognition process.

Figure 5.12: Example of false detection; COCO (left) and MPII (right)

In the figure above 5.12 there is an example where in the same frame with the COCO dataset the skeleton is correctly recognized, while the MPII wrongly detect a lot of points. This lead to a deep alteration of the fundamental matrix while using MPII.

When at the end of the flow of the algorithm the fundamental matrix is obtained, it is possible to compare and understand how this calibration method performs compared to the gold-standard one.

Mainly two different methods are take into account to demonstrate both graphically and mathematically the reliability and the correctness of the results obtained.

The comparison is done directly with the Matlab results because it is the considered as a gold-standard technique to obtain the fundamental matrix.

5.7 Mathematical demonstration

To ensure the correctness and to measure the accuracy of the results obtained compared to the ideal situation and to the gold-standard technology, it is necessary to exploit a mathematical relation that describes how much precise is the fundamental matrix. In other words the aim of this step is to evaluate how much precise is the correspondence of points in a pair of images. Using the epipolar geometry and the fundamental matrix properties described before, considering a point X in the 3D real world space that is captured by a stereo camera system. One camera will detect the point X in the point x, while the other one in x'. Ideally the mathematical relation that exists between these points is the following:

$$x'^{T} F x = 0 \tag{5.3}$$

Considering x'^{T} as the transpose of the x' coordinate and F as the fundamental matrix, this equation describes how the perfect fundamental matrix leads this equation to be equal to zero. The output value is measured in $pixel^2$ and in the real cases this value is always non-zero but it is considered acceptable as long as its value is about 1.

This equation can be solved for both the fundamental matrix obtained with the gold-standard method and the one obtained with the human pose recognition approach. Comparing the results it is possible to evaluate their accuracy.

In the case of the fundamental matrix obtained using the chessboard approach, that is considered as the gold-standard during this work, the mean error measure in pixel is:

$$e_{pixel} \approx 0.55 \tag{5.4}$$

While in the case of the human pose recognition approach this value is:

$$e_{pixel} = 1.4436963 \approx 1.44 \tag{5.5}$$

As expected the error for the skeleton detection approach is higher since the different keypoints cannot be univocally associated to one specific pixel. For example a shoulder occupies a large amount of pixels, so it is identified as centered as possible in this space of pixels, but it can't be as precise as a chessboard corner that is very sharp and clear, due to the shape and the contrast of colors.

5.8 Graphical demonstration

To graphically demonstrate the correctness of the results obtained the stereo rectification is exploited. This technique is able to simplify the calculation of disparities of a specific pixel in both images. Considering a stereo camera system,

this technique acquires the image containing the same scene from both cameras and transforms them in such a way that they look like they have been taken on the same horizontal plane. So the result of this procedure is to have both images that are misaligned only in the horizontal axis (more informations about this technique are available at [26]). In this way a 2D problem is converted in a 1D problem, that is much easier to solve.

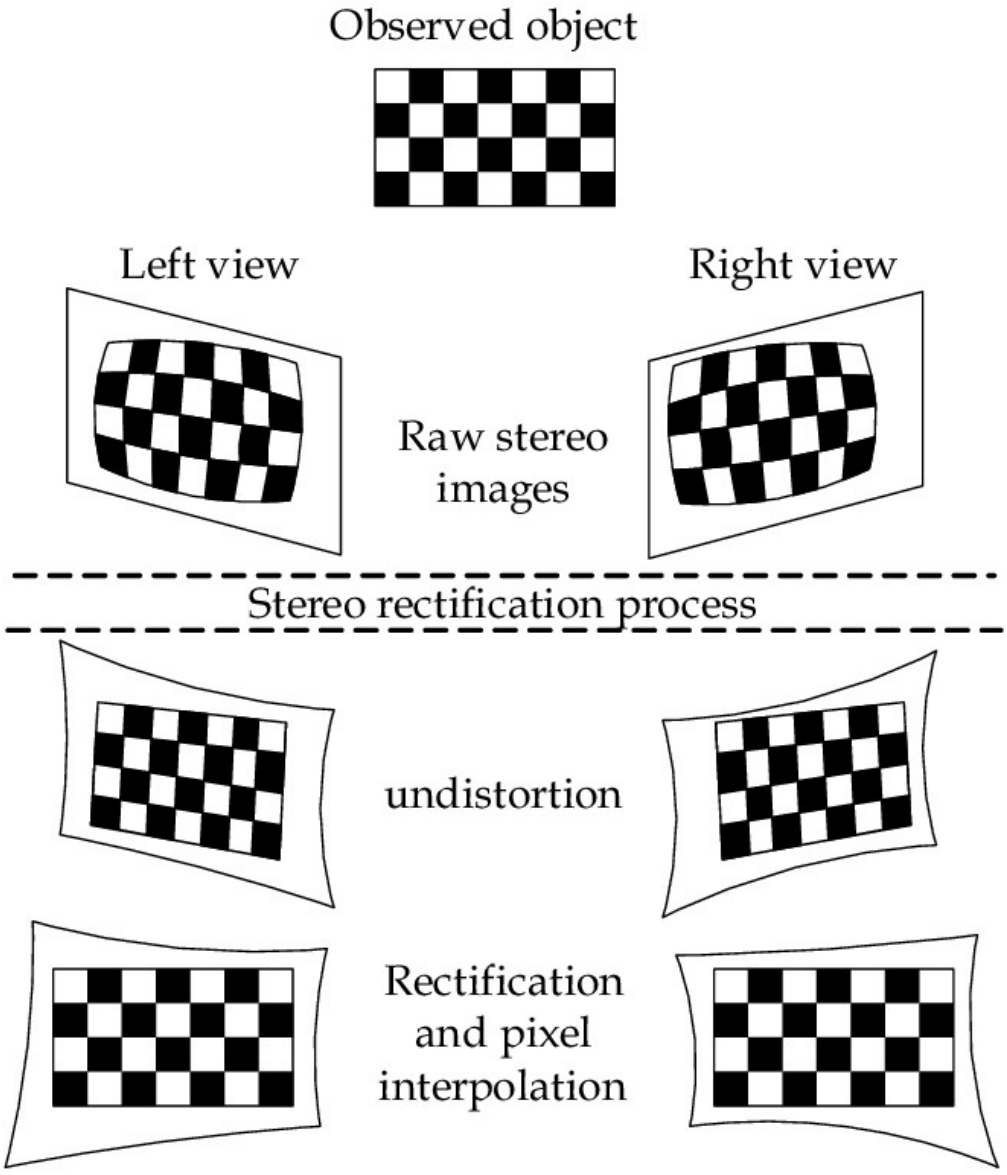


Figure 5.13: Flow of the stereo rectification process

In the figure above 5.13 is available the flow of how this technique start from the

raw images and obtain the final results described before. The OpenCV library has been used to implement this procedure using Python. Considering a pair of images, the homography matrix describes the mapping of a pixel on a plane contained in the image to the other one; so using this matrix is possible to relate a field of view of one camera to the other one.

Figure 5.14: Homography application on a pair of example images

In the figure 5.14 there is an example of a pair of raw images (on the left) that contains a plain (the cover of the book) with a specific point A (marked in red in both images). The points $A'(x_1, y_1)$ and $A''(x_2, y_2)$ (respectively in the first image and in the second one of the same pair on the left) are related through an homography.

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = H * \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \tag{5.6}$$

In the equation above the H parameter is the homography matrix (a 3x3 matrix) that describes the alignment of the two planes in the images, as shown in the right pair of images in figure 5.14 (more information about this technique are available at [27]).

During this results validation process, starting from two raw images (one per camera) already used in the calibration process with the chessboard, the findchessboardcorners() function has been used to obtain the coordinates of 45 keypoints (one chessboard with 9 rows and 6 columns) contained in both images. Using the StereoRectifyUncalibrated() function of the OpenCV library it is possible to obtain the homography matrix H for the pair of images. The warpPerspective() function modifies the perspective of the images and transforms them using the information obtained by the homography matrix and the fundamental matrix. At the end of this process a new pair of images is generated, these are ideally perfectly aligned on the horizontal plane.

This procedure has also been repeated using the Matlab Stereo Camera Calibrator App to be able to compare the results obtained.

Figure 5.15: Rectification of a pair of images using the Matlab App

In the figure above 5.15 is clear how the two images are perfectly aligned. To ease the graphical detection of the alignment there are some horizontal lines; during their path those lines cross the chessboard ideally in the same pixels in both images. By zooming on the chessboard it is clear every specific lines touch the chessboard in the same points on both images, without any visible error.

Using the python implementation described above, it is possible to perform the same rectification on a pair of images using the fundamental matrix obtained with the human pose recognition approach.

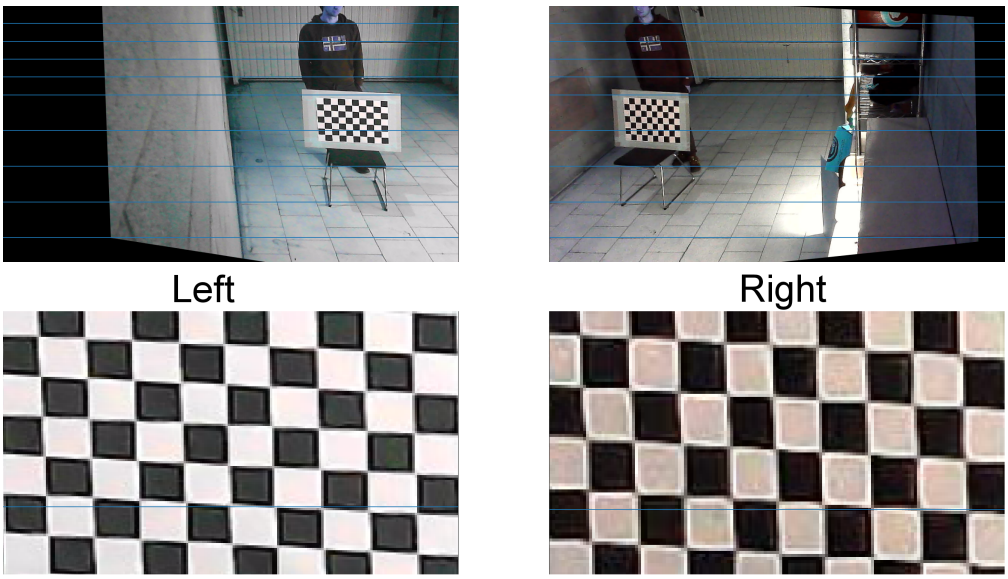


Figure 5.16: Stereo rectification of an image pair using the Python implementation and the fundamental matrix obtained with human pose recognition

The figure 5.16 shows on top a pair of images rectified; in the lower part of the image there is the chessboard zoomed to highlight that the blue line crosses the chessboard not exactly in the same places in both images, but in the case of the right one the line appears to be slightly lower than in the left one compared to the chessboard.

The accuracy obtained using this method is surely lower than the gold-standard one, but it is still high enough to be acceptable, especially for the field application of this work that doesn't require a sub-pixel precision.

5.9 Time-related considerations

The aim of this work, as said at the beginning, is also related to lower the time needed to obtain the results. The comparison between the two methods to obtain the fundamental matrix can be also done time-wise. The chessboard calibration process, on this side, has a lot of disadvantages, one of the most important contribution of time spent with this approach is related to the acquisition process.

	Setup and Acquisition	Elaboration and Validation
Chessboard	120 min	60 min
Human pose	5 min	60 min

Table 5.1: Table of the time spent per step using different approaches

In the table above 5.1 there are estimations of how much time each step needs. Every procedure has been performed using an high performance consumer level CPU (Intel Core i9 9900K). It's noticeable how the chessboard approach requires a lot of time during the acquisition process. The acquisition of some hundreds of pictures from both cameras, where the chessboard is moved in each of them into a different position, is a very time-consuming step. On the other hand the human pose recognition only requires to record an human body in the common field of view for some minutes. This last procedure of course is faster and also a lot easier, because it can be performed without any specific knowledge in camera calibration, while in the case of the chessboard approach there are a lot of tips that can drastically improve the correctness of the result. For both approaches the elaboration time is similar, so in this case there aren't noticeable differences. In addition to this it must be highlighted that the human pose approach can be simplified using different models and datasets to lower the computational complexity of the elaboration; in this way this approach can potentially get very close to the real time processing.

Chapter 6

Conclusions and future perspectives

During this work an alternative approach to calibrate a stereo camera system is proposed. This idea came from the desire to lower costs of the hardware and to get rid of qualified personnel. As described in the previous chapters, the results obtained are good and even if the accuracy is lower than the gold-standard technology, it is high enough for the application field that this method must face. This approach lays the groundwork to a new simpler, faster and cheaper way to obtain the required informations to calibrate a stereo camera system, instead of using slow calibration procedures (eg: the chessboard one) associated with very expensive hardware (wearable sensors technology). All the implementation is made using Python language, but can easily be translated to C language to increase performances, because OpenCV supports both of these languages. The entire extrinsic calibration process of the stereo camera system using an high performance consumer level processor takes about one hour, starting from the frame extraction, through the human pose recognition in every frame, to the achievement of the fundamental matrix. This numbers can be shorten if an HPC (high performance computing) system is used or if the dataset and the model used are simpler than the ones used during this work. This dramatically increases the versatility of this approach. During the entire work every algorithm step has been implemented starting from a blank project exploiting different libraries; to obtain these results more than one thousand of lines of code were written without the use of any intermediate toolbox. For confidentiality reasons the entire algorithm cannot be published in this paper, but in every chapter the flow of the algorithm implementation was described.

Bibliography

[1] Mathworks. URL: https://it.mathworks.com/help/vision/ug/camera-calibration.html (cit. on p. 7).

[2] Aerin Kim. *Camera Calibration: Camera Geometry and The Pinhole Model*. URL: https://towardsdatascience.com/camera-calibration-fda5beb373c3 (cit. on p. 9).

[3] Stefano Mattoccia. *Stereo Vision: Algorithms and Applications*. URL: http://vision.deis.unibo.it/~smatt/Seminars/StereoVision.pdf (cit. on p. 12).

[4] Kenji Hata and Silvio Savarese. *Epipolar Geometry*. URL: https://web.stanford.edu/class/cs231a/course_notes/03-epipolar-geometry.pdf (cit. on p. 13).

[5] Dennis Kant. *Extrinsic calibration and aperture angle*. URL: https://www.eecs.tu-berlin.de/fileadmin/fg144/Courses/06WS/scanning/Dennis/Extrinsic%20calibration.htm (cit. on p. 15).

[6] Python. URL: https://python.org/ (cit. on p. 17).

[7] OpenCV. URL: https://opencv.org/ (cit. on p. 17).

[8] NumPY. URL: https://numpy.org/ (cit. on p. 17).

[9] Mathworks. URL: https://it.mathworks.com/help/vision/ref/cameracalibrator-app.html (cit. on p. 17).

[10] Mathworks. URL: https://it.mathworks.com/help/vision/ug/stereo-camera-calibrator-app.html (cit. on p. 17).

[11] Zhengyou Zhang. *A Flexible New Technique for Camera Calibration*. URL: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr98-71.pdf (cit. on p. 17).

[12] Robert Collins. *Harris Corner Detector*. URL: http://www.cse.psu.edu/~rtc12/CSE486/lecture06.pdf (cit. on p. 18).

[13] OpenCV. URL: https://docs.opencv.org/master/dc/dbb/tutorial_py_calibration.html (cit. on pp. 18, 20).

[14] Nabil Madali. *Introduction To Epipolar Geometry*. URL: https://towardsdatascience.com/introduction-to-epipolar-geometry-1bbe6e505b81 (cit. on p. 31).

[15] OpenCV. URL: https://docs.opencv.org/master/ (cit. on p. 32).

[16] Brigham Young University Robotic vision lab. *Stereo Calibration Rectification*. URL: http://ece631web.groups.et.byu.net/Lectures/ECEn631%2014%20-%20Calibration%20and%20Rectification.pdf (cit. on p. 33).

[17] Mohan Chand Hanumara. *Calibrating and Creating Point Cloud from a Stereo Camera Setup Using OpenCV*. URL: https://www.researchgate.net/publication/308325524_Calibrating_and_Creating_Point_Cloud_from_a_Stereo_Camera_Setup_Using_OpenCV (cit. on p. 34).

[18] Disney Research Leonid Sigal. *Human pose estimation*. URL: https://www.cs.ubc.ca/~lsigal/Publications/SigalEncyclopediaCVdraft.pdf (cit. on p. 37).

[19] Zheng Tang - Yen-Shuo Lin - Kuan-Hui Lee - Jenq-Neng Hwang - Jen-Hui Chuang - Zhijun Fang. *Camera Self-Calibration from Tracking of Moving Persons*. URL: https://projet.liris.cnrs.fr/imagine/pub/proceedings/ICPR-2016/media/files/0243.pdf (cit. on p. 40).

[20] Anh Minh Truong - Wilfried Philips - Nikos Deligiannis - Lusine Abrahamyan - Junzhi Guan. *Automatic Multi-Camera Extrinsic Parameter Calibration Based on Pedestrian Torsors*. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6891296/ (cit. on p. 40).

[21] Junzhi Guan - Francis Deboeverie - Maarten Slembrouck - Dirk Van Haerenborgh - Dimitri Van Cauwelaert - Peter Veelaert - Wilfried Philips. *Extrinsic Calibration of Camera Networks Based on Pedestrians*. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4883345/ (cit. on p. 40).

[22] COCODataset. URL: https://cocodataset.org/#home (cit. on p. 40).

[23] Bagavathi Sivakumar Pb Kanishka Nithin.Da. *Generic Feature Learning in Computer Vision*. URL: https://www.researchgate.net/publication/283184746_Generic_Feature_Learning_in_Computer_Vision (cit. on p. 42).

[24] Caffe2. URL: https://caffe2.ai/docs/tutorial-models-and-datasets.html (cit. on p. 42).

[25] Zhaowei Li and David R. Selviah. *Comparison of Image Alignment Algorithms*. URL: https://www.ee.ucl.ac.uk/lcs/previous/LCS2011/LCS1115.pdf (cit. on p. 47).

[26] Andrea Fusiello. *Tutorial on Rectification of Stereo Images*. URL: https://www.researchgate.net/publication/2841773_Tutorial_on_Rectification_of_Stereo_Images (cit. on p. 50).

[27] Columbia University - Department of Computer Science. *Homography, Transforms, Mosaics*. URL: http://www.cs.columbia.edu/~allen/F17/NOTES/homography_pka.pdf (cit. on p. 51).

While researching all the informations needed the topics contained in the bibliography sometimes has been further explored to better understand it. There were several more papers that has been used to learn informations about covered topics that were left out for the sake of brevity. Only the main ones have been inserted in this bibliography.