

# **3D Points Recover From Stereo Video Sequences**

**Based on**

**OpenCV 2.1 Libraries**

A Thesis Presented to the Faculty of Mechanical Engineering and Robotics

AGH University of Science and Technology of Krakow

In Partial Fulfillment of the Requirements of the Degree of

Master of Science

in

Mechanical Engineering

by

Rosário D. V. Valente

July 22, 2011

# **3D Points Recover From Stereo Video Sequences**

**Based on**

**OpenCV 2.1 Libraries**

In Partial Fulfillment of the Requirements of the Degree of

Master of Science

in

Mechanical Engineering

by Ros rio D. V. Valente

Faculty of Mechanical Engineering and Robotics

AGH University of Science and Technology of Krakow

July 22, 2011

Under the guidance and approval of the committee, and approved by all its members, this thesis has been accepted in partial fulfillment of the requirements for the degree.

Approved:

---

Chairperson (Dr. Piotr Kohut)

---

Date

---

Committee Member (Dr. Wojciech Lisowski )

---

Date

# 1 Acknowledgements

It is a pleasure to thank those who made this thesis possible, supporting it directly and indirectly. First of all I would like to thank my supervisor, Dr. Piotr Kohut, for his guidance, advice and specially for his time during the numerous meetings and laboratory experiments that enabled me to achieve better results and a deeper understanding on the subject. Also, I would like to thank Dr. Wojciech Lisowski to whom I owe my deepest gratitude for his support and assistance since the start of my stage as erasmus student at AGH University and for his comprehension and kindness during all the process. I would like to thank my mum Gabriela and my sisters Mary and Fernanda for being always supportive even when I spend long time far from them. I would like to thank Paula Olim for her friendship, kindness and encouragement and to make me believe that it is always possible to go further. Also I would like to show my gratitude to Olinha for her support, joy and company and to all my friends who have been helping and supporting me on my goals and ambitions.

Lastly, but not the least, I would like to thank all the OpenCV community for providing a great support to all of those interested in learning OpenCV. Finally I would like also to tank all the authors who provided all the interesting research and literature on stereo vision.

## 2 Abstract

The purpose of this study was to implement a program in C++ using OpenCV image processing platform's algorithms and Microsoft Visual Studio 2008 development environment to perform cameras calibration and calibration parameters optimization, stereo rectification, stereo correspondence and recover sets of 3D points from a pair of synchronized video sequences obtained from a stereo configuration. The study utilized two pretest laboratory sessions and one intervention laboratory session. Measurements included setting different stereo configurations with two Phantom v9.1 high-speed cameras to: capture video sequences of a MELFA RV-2AJ robot executing a simple 3D path, and additionally capture video sequences of a planar calibration object, being moved by a person, to calibrate each stereo configuration. Significant improvements were made from pretest to intervention laboratory session on minimizing procedures errors and choosing the best camera capture settings. Cameras intrinsic and extrinsic parameters, stereo relations, and disparity-to-depth matrix were better estimated for the last measurements and the comparison between the obtained sets of 3D points (3D path) with the robot's 3D path proved to be similar.



# RECOVERING 3D POINTS FROM STEREO VIDEO SEQUENCES BASED ON OPEN CV 2.1 LIBRARIES

1 Acknowledgements.....	iii
2 Abstract.....	iv
3 Index of Tables.....	vii
4 Illustration Index.....	viii
5 Chapter One.....	1
5.1 Introduction.....	1
5.1.1 Statement of the Problem.....	2
5.1.2 Background and Need.....	3
5.1.3 Purpose of the Study.....	5
5.1.4 Research Questions.....	6
5.1.5 Significance of the field.....	6
5.1.6 Definitions.....	7
5.1.7 Limitations.....	9
6 Chapter Two.....	10
6.1 Review of the Literature.....	10
6.1.1 Introduction.....	10
6.1.2 Research Synthesis.....	10
6.1.3 Summary.....	36
7 Chapter Three.....	37
7.1 Methods.....	37
7.1.1 Introduction.....	37
7.1.2 Settings.....	38
7.1.3 Intervention and Instructional Materials.....	38
7.1.4 Measurements Instruments.....	39
7.1.5 Procedures.....	41
8 Chapter Four .....	63
8.1 Results.....	63
8.1.1 Introduction .....	63
8.1.2 Research Question N°1 – Results.....	68
8.1.3 Research Question N°2 – Results.....	68
8.1.4 Research Question N°3 – Results.....	81
8.1.5 Research Question N°4 – Results.....	82
8.1.6 Research Question N°5 – Results.....	84
9 Chapter Five.....	88
9.1 Discussion .....	88
9.1.1 Introduction.....	88
9.1.2 Discussion .....	88
9.1.3 Limitations.....	93
9.1.4 Recommendations for Future Research .....	94
9.1.5 Conclusions.....	94
10 References.....	96
11 Appendix A: Stereo Imaging.....	99
11.1 Stereo Imaging.....	100
11.1.1 Introduction.....	100
11.1.2 Working With a Single Camera.....	100
11.1.3 Calibration .....	103

11.1.4	Camera Calibration .....	106
11.1.5	Undistortion.....	108
11.2	Working With Two Cameras .....	111
11.2.1	Stereo Imaging .....	111
11.2.2	Stereo Calibration.....	114
11.2.3	Stereo Rectification.....	115
11.2.4	Stereo Correspondence.....	117
12	Appendix B: MELFA Basic IV Presentation.....	120
13	Appendix C: VideoStrobe & VideoFlood LEDs.....	141
13.1	VideoStrob and VideoFlood LEDs.....	142
13.2	LEDs Arrays Specifications .....	143
13.3	VideoFlood LED Light Comparison Table.....	144
14	Appendix D: Phantom v. 9.1 Data Sheet.....	145
15	Appendix E: MatLab M-Files Code.....	151
15.1	File 1: getNodeData.m.....	152
15.2	File 2: readDataFromXML.m.....	154
15.3	File 3: plot3DPath.m.....	155
15.4	File 4: readMelfaData.m.....	157
15.5	File 5: transformReferential.m.....	158
15.6	File 6: testTransformReferential.m.....	160
16	Appendix F: Motion.....	161
16.1	Motion .....	162
16.1.1	Introduction.....	162
16.1.2	Corners Identification .....	162
16.1.3	Corners Sub-pixel Accuracy .....	163
16.2	Optical Flow.....	164
16.2.1	Introduction.....	164
16.2.2	Sparse Tracking Techniques .....	165
16.2.3	Dense Tracking Techniques.....	169
17	Appendix G: Targets and MELFA Basic IV Program.....	171
18	Appendix H: StereoVisionProg.....	176
18.1	Introduction .....	177
18.2	Main menu's Option [0] – Compute Optical Flow.....	177
18.3	Main menu's Option [1] – Single video operations.....	180
18.4	Main menu's Option [2] – Stereo video operations.....	182
18.5	Main menu's Option [3] – Stereo calibration.....	186
18.6	Main menu's Option [4] – Compute 3D points.....	191
18.7	Main menu's Option [5] – Rotation matrix parametrization.....	196
18.8	Main menu's Option [6] – List current directory files .....	196

### 3 Index of Tables

Table 7.1: OpenCV Calibration Object's Characteristics .....	44
Table 7.2: Laboratory 03 – Stereo Configuration's Variables and Video Files.....	45
Table 7.3: List of MatLab Implemented Functions.....	62
Table 8.1: Video Sequences Collected During Laboratories Experiments .....	63
Table 8.2: Calibration Methods Study Process's Output Variables.....	64
Table 8.3: Stereo Calibration Process's Output Variables.....	65
Table 8.4: Rotation Parametrization Process's Output Variables.....	66
Table 8.5: Calibrated and Uncalibrated Rectification Process's Output Variables.....	66
Table 8.6: Recovering 3D Points Process's Output Variables.....	67
Table 8.7: L01 Set S03 Calibration Methods Study Using $f_x$ and $k_2$ Parameters.....	68
Table 8.8: L02 Set S03 Calibration Method's Study Using $f_x$ and $k_2$ Parameters.....	69
Table 8.9: L03 Set S04 Calibration Method's Study Using $f_x$ and $k_2$ Parameters.....	70
Table 8.10: Stereo Calibration Parameters Optimization Results (L01, L02, and L03).....	75
Table 8.11: Laboratory L01's Camera Calibration Parameters (Methods M1, M2, and M3).....	78
Table 8.12: Laboratory L02's Camera Calibration Parameters (Methods M1, M2, and M3).....	79
Table 8.13: Laboratory L03's Camera Calibration Parameters (Methods M1, M2, and M3).....	80
Table 8.14: Laboratory Stereo Configuration's Measurements.....	83
Table 8.15: Stereo Configuration's Relations .....	83
Table 9.1: Calibration Method's Study Summary.....	89
Table 13.1: Model 900405 3-by-4 Super Bright LEDs Array.....	144

## 4 Illustration Index

Figure 6.1: Relative errors vs. noise level ( $\alpha$ , $\beta$ ), Zhang (2000).....	16
Figure 6.2: Absolute errors vs. noise level ( $u$ , $v$ ), Zhang (2000).....	17
Figure 6.3: Relative error vs. number of planes ( $\alpha$ , $\beta$ ), Zhang (2000).....	17
Figure 6.4: Absolute error vs. number of planes ( $u_0$ , $v_0$ ), Zhang (2000).....	17
Figure 6.5: Relative error vs. angle with image plane ( $\alpha$ and $\beta$ ), Zhang (2000).....	18
Figure 6.6: Absolute error vs. angle with image plane ( $u_0$ , $v_0$ ), Zhang (2000). ....	18
Figure 6.7: Parameter result's variations with different sets of images, Zhang (2000).....	19
Figure 6.8: Calibration parameter's results with different image sets, Zhang (2000). ....	19
Figure 6.9: Stereo matching process, Stefano et al. (2002).....	29
Figure 6.10: Scores associated with point $R(x, y)$ , Stefano et al. (2002).....	30
Figure 6.11: SAD matching window, Stefano et al. (2002).....	31
Figure 6.12: Tsukuba image (left) and ground truth (right), Stefano et al. (2002).....	34
Figure 6.13: Disparity maps computed with the P.A (left) and with SVS 2.0 software (right), Stefano et al. (2002).....	35
Figure 6.14: Speed (fps) for P.A. and for the SVS 2.0 algorithm, Stefano et al. (2002).....	35
Figure 7.1: Phantom v.91 cameras arranged on a stereo configuration. ....	42
Figure 7.2: PCC1.2 software - cine settings.....	42
Figure 7.3: Targets used for points tracking purposes.....	43
Figure 7.4: PCC Software - save cine settings.....	46
Figure 7.5: StereoVisionProg: Main menu 's Option [ 2 ]......	46
Figure 7.6: StereoVisionProg: Main menu's Option[ 2 ] sub Option [ 3 ]......	47
Figure 7.7: Sequence of BMP images for calibration.....	47
Figure 7.8: StereoVisionProg: Main menu's Option [ 3 ] sub Option [ 1 ]......	47
Figure 7.9: StereoVisionProg: Main menu's Option[ 3 ] sub Option[ 2 ] .....	48
Figure 7.10: Reprojected (a) and projected (b) image points.....	49
Figure 7.11: Calibration parameters optimization process.....	50
Figure 7.12: StereoVisionProg: Main menu's Option [ 3 ] sub Option [ 4 ]......	50
Figure 7.13: StereoVisionProg: rectification method options. ....	50
Figure 7.14: Right camera rotation using Euler angles.....	53
Figure 7.15: Pseudo-code to compute quaternion from $R$ . ....	54
Figure 7.16: StereoVisionProg: Main menu's Option [ 4 ] sub Option [ 2 ]......	55
Figure 7.17: StereoVisionProg: Main menu's Option[ 4 ] sub Option[ 2 ] sub Option[ 2 ].....	55
Figure 7.18: Sparse stereo correspondence with Lucas-Kanade tracker. ....	56
Figure 7.19: Canonical stereo configuration (a), similarity of triangles(b).....	57
Figure 7.20: Camera-to-MELFA robot referential transformation.....	60
Figure 7.21: Referential transformation using point-line-plane method. ....	61
Figure 8.1: L01-S03Focal Length vs N° of Calibration Views (M1 and M2).....	71
Figure 8.2: L03-S04 Focal Length vs N° of Calibration Views (M1 and M2).....	72
Figure 8.3: Single calibration view from L02 set S03 (a) and L03 set S04 (b).....	72
Figure 8.4: L02-S03 Focal Length vs N° of Calibration Views (M1 and M2).....	73
Figure 8.5: L03-S03 Focal Length vs N° of Calibration Views (M1 and M2).....	73
Figure 8.6: Calibration parameters optimization (Method M3). ....	74
Figure 8.7: Calibration view's reprojection/projected image points.....	76
Figure 8.8: Stereo video capture without rectification.....	81
Figure 8.9: Stereo video capture after calibrated stereo rectification.....	82
Figure 8.10: Stereo matching using Block-Matching algorithm.....	84
Figure 8.11: MELFA RV-2AJ 3D Path (sampling time: 50ms).....	85

Figure 8.12: Figure 8.8: MELFA RV-2AJ 3D Path (sampling time: 10ms).....	85
Figure 8.13: MELFA robot's recovered 3D path(camera coordinates system).....	86
Figure 8.14: MELFA robot's recovered 3D path (robot coordinate system).....	86
Figure 8.15: MELFA robot's recovered 3D path (generic coordinate system).....	87
Figure 11.1: Pinhole camera model.....	100
Figure 11.2: Simplified pinhole camera model.....	101
Figure 11.3: Example of a 5-by-3 chessboard corner's detection. ....	103
Figure 11.4: Generic point defined on a planar calibration object. ....	104
Figure 11.5: Relation between a point in the planar object and the imager plane.....	105
Figure 11.6: Calibration object - chessboard 5-by-3.....	107
Figure 11.7: Building object point's vector of vector of 3D points.....	108
Figure 11.8: Stereo configuration geometry .....	111
Figure 11.9: Standard stereo configuration's epipolar geometry. ....	112
Figure 11.10: Video sequences after stereo rectification.....	117
Figure 13.1: Video Strobe - Flood Controller and 3-by-4 LED Array used during laboratory experiment 03. Adapted from VideoStrobe & VideoFlood LEDs by Visual Instrumentation Corporation 2009, Adapted with permission.....	142
Figure 13.2: LED Array specifications. Two 3-by-4 LED Array Model 900405 were used during laboratory experiment 03. Adapted from VideoStrobe & VideoFlood LEDs by Visual Instrumentation Corporation 2009, Adapted with permission.....	143
Figure 14.1: Phantom v. 9.1 Data Sheet p1/3. Adapted from Phantom v9.1 by Vision Research Inc, 2007. Adapted with permission. ....	146
Figure 14.2: Phantom v. 9.1 Data Sheet p2/3. Adapted from Phantom v9.1 by Vision Research Inc, 2007. Adapted with permission. ....	147
Figure 14.3: Record speed vs. Image Resolution p3/3. Adapted from Phantom v9.1 by Vision Research Inc, 2007. Adapted with permission. ....	148
Figure 14.4: Mechanical shutter p1/2. Adapted from V-Series Lens Shutter by Vision Research Inc, 2010. Adapted with permission. ....	149
Figure 14.5: Break-out-Box p2/2. Adapted from V-Series Lens Shutter by Vision Research Inc, 2010. Adapted with permission. ....	150
Figure 16.1: Sequence of frames where the optical flow is to be computed.....	165
Figure 16.2: Normal flow. ....	166
Figure 16.3: Aperture problem originated by a small aperture window.....	167
Figure 16.4: Result from applying the pyramid Lucas - Kanade technique.....	168
Figure 17.1: Dimensioning of the target1 and target 2.....	172
Figure 17.2: Dimensioning of the target 3. ....	173
Figure 17.3: Targets 1, 2, and 3.....	174
Figure 17.4: MELFA Basic IV simple 3D path program implementation. ....	175
Figure 18.1: StereoVisionProg: Main menu.....	177
Figure 18.2: Main menu's option 0 sub menu. ....	178
Figure 18.3: Current directory's list of avi files.....	178
Figure 18.4: TRACKING SETTINGS window's trackbars.....	178
Figure 18.5: Sparse optical flow tracking example.....	179
Figure 18.6: Dense optical flow using Horn - Schunck method.....	179
Figure 18.7: Main menu's option 1 sub menu.....	180
Figure 18.8: Capture controls' window.....	180
Figure 18.9: OpenCV video compression selection.....	181
Figure 18.10: Main menu's option 2 sub menu.....	182
Figure 18.11: Capturing from two A4Tech Web Cams.....	182
Figure 18.12: List image's names into a text file.....	184

Figure 18.13: Real-time stereo calibration. ....	185
Figure 18.14: Main menu's option 3 sub menu.....	186
Figure 18.15: Study optimal number of calibration views.....	187
Figure 18.16: Calibration process method M1 and M2.....	188
Figure 18.17: Calibration parameters optimization.....	189
Figure 18.18: Calibrated stereo rectification.....	190
Figure 18.19: Uncalibrated stereo rectification.....	190
Figure 18.20: Main menu's option 4 video input.....	191
Figure 18.21: Stereo matching modes.....	191
Figure 18.22: Dense stereo matching approach.....	192
Figure 18.23: Block – Matching control window.....	193
Figure 18.24: Image points tracking settings.....	194
Figure 18.25: Sparse stereo matching approach. ....	195
Figure 18.26: Rotation matrix parametrization.....	196

# 5 Chapter One

## 5.1 Introduction

The capability to perceive the three dimensional world where we live for us humans is a task that since the beginning seems easy and granted. It may take only two years for a baby be capable to experience the world through his senses such as looking, touching and even understanding that an object exist even when is hidden from the field of view (FOV). In the up following years he become capable to represent the three dimensional world with colours, objects, shapes and symbols resulting from the four stages of cognitive development that we humans pass through (<http://alleydog.com/psychology-topics.php>, Child Psychology, ¶ 4 , 5).

Because we humans are provided with such complex and efficient human visual system (HVS) that does all the computations for us it may create a misleading impression that attempting to effectively simulate and copy such functions is an easy task. For perceiving the information gathered from HVS the brain uses three main principals: stereo vision, motion parallax and the prior knowledge about the objects perspective appearance and their relation with the distance (May, Pervoelez, & Surmann, 2007), however, in machine visual systems the information is received from the sensing machine or a media storage and transformed into different layers of matrices numbers and in most cases without any previous knowledge about the surrounding variables (weather , lightning, reflections, occlusions, movements) that change the way images are captured and this is all the information available (Bradski, & Kaehler, 2008).

Computer Vision (CV) is the science challenged to study the transformation of that information in form of images or sequences of images into information that allows to implement and deal efficiently with all this three complex tasks for perceiving the 3D space by means of machine visual sensing. This science has been widely progressing through the years since the time it started capturing the interest of researchers to mimic the human intelligence and reproduce it into the robots intelligence (Szeliski, 2010). However, it was manly in the recent decades that the interest by the researchers and the high demands from the industry for new features affordable by computer applications has notoriously increased.

Computer vision have been developing in parallel with common areas such as computer science, optical systems and mathematical techniques that with time are becoming more and more able to decode the inverse vision problem. Computer vision became a very vast field of studies but one field in special has received major attention in particular – the capability of performing effective automatic reconstruction and analysis of the surrounding 3D environment and objects recognition in that space (Cyganek, & Siebert, 2009).

In addition to the fast development in computer vision, new ways of commercialization gives origin to performance-optimized software that runs in different platforms and that are mostly open source free for both academic and commercial use what makes easy to exchange experiences and documentation widely among research groups and therefore providing a base for a faster development. One example of this strategy is the case of Intel with the Open Source Computer Vision (OpenCV) that provides computer vision applications to increase the need for faster processors (Bradski et al. 2008).

OpenCV is a library of programming functions mainly directed for real time computer vision such motion tracking, stereo and multi-camera calibration and depth computation (<http://opencv.willowgarage.com/wiki/FullOpenCVWiki>, Introduction ¶ 1). This capabilities offered by OpenCV combined with the object-oriented and generic programming techniques

offered by C++ programming language is a suitable choice to implement large and reusable projects (Papademetris, 2006).

To recover the information lost on the process of projecting 3D world space to the 2D image space the implementation of a number of classes objects is an essential part for the recovering process. This process requires the successful implementation of a program capable to calibrate two cameras with equal properties, capture stereo-pair images of a scene and then compute the depth information within those two images and reproject it to 3D space in real time, by means of using OpenCV libraries and C++ object oriented programming (OOP) capabilities.

### **5.1.1 Statement of the Problem**

The challenges present in recovering the 3D information from 2D images based on processing stereo-pair images are related to three main areas: cameras calibration, images rectification, disparity maps and its reprojection to 3D space. To achieve the so desirable information in three-dimensional space from a sequence of stereo-pair images it is necessary to recall the basic principles that are crucial to its correct implementation.

Firstly to determine the coordinate transformation between the camera reference system in respect to an external coordinate system, it is necessary to know the variables that relates such relation. The direct measurement of those variables is difficult or even impossible and therefore a process of calibration is required to determine camera intrinsic and extrinsic parameters. The final performance of the machine vision system strongly depends on the accuracy of camera calibration (Niola, Rossi, Savino, & Strano, 2008). The information retained by the extrinsic parameters are used to correct the distortion induced by the hardware, this process is called undistortion.

Ideally cameras' arrangement would be such that the resulting images are row aligned and each point of one image will correspond to another point in the same row of a second image, however such canonical stereoscopic system is not possible physically and the images need to be remapped, this process is referred to as *rectification*.

The last challenge on the process of recovering the lost dimension requires the *correspondence* process. Simply stated, correspondence refers to the matches between two images captured from two different viewpoints looking at the same 3D world scene or object. It is one of the most active topics in computer vision due its complexity and important role in 3D object recognition and categorization, scene reconstruction and many other applications (Hsu, 2011).

#### **5.1.1.1 Correspondence.**

One of the major issues on estimating 3D structures based on stereo imagery is the correspondence problem defined as the capability to locate a pair of image pixels from two different images that represent the projection of the same point in 3D space. Given a point in one image, its correspondent point must lie on an line (epipolar line) in the other image. This constitutes a very important constraint called epipolar constraint: Each image point of a space point lies in the image plane only on the corresponding epipolar line (Cyganek et al., 2009). This constraint presents a second problem, in general cases the location of the epipolar lines are not known.

#### **5.1.1.2 Rectification.**

As stated previously the positions of the epipolar lines are not known for a standard stereoscopic camera configuration however for the canonical stereoscopic configuration those lines positions can be know using epipolar geometry and thus this transformations between the generic configuration to the desired canonical configuration constitutes the second problem of the research



area. In order to perform successfully this operation between both configurations previous knowledge is required to know how the cameras related the world coordinates with the image coordinates when capturing the images or sequence of images. This process is referred to as *camera calibration* (Shah, 1997).

#### **5.1.1.3 Camera Calibration.**

Camera calibration is the first problem to be solved and it plays an important role in the final results on the research area. Camera calibration is the process to estimate a set of parameters that describes the camera imaging process. Computing this set of parameters will allow to: link directly a point in the 3D world reference frame to its corrected image through the perspective projection matrix (Ma, Chen, & Moore, 2003) , map the camera coordinate system into the image coordinate system, and compute geometrical distortions that are originated by the imperfections and limitations of camera's physical parameters (Cyganek et al., 2009).

As noted, recovering the lost dimension to estimate 3D structures based on stereo images requires that three steps need to be successfully implemented. Initially the camera model, the distortions and perspective projection matrix are computed. This step is important once the accuracy in which the de camera parameters were computed will play an important role in the final depth results. In order to obtain two images with stereoscopic camera configuration and simplify the mathematical relations a process of rectification is performed on both images so they become as if the cameras where in the canonical configuration with the optical axis meeting at infinity. Finally in order to compute the depth images using the both rectified images a correspondence process is performed and the third dimension recovered from the image coordinates where was possible to match both image pixels (Wu, & Chen, 2007).

#### **5.1.2 Background and Need**

The earliest techniques for reproducing 3D information using stereo matching approach started in the area of cartography for automatic construction of topographic elevation maps from overlapping aerial images. This initial progress in the aerial imagery area played an important role in the progress and development of fully automated and efficient stereo matching algorithms in depth recovering field (Szeliski, 2010). However, since computer vision started out in the early 1970s many attempts to recover the three-dimensional structures happened in parallel with a wide area of studies as the case of stereo vision techniques and algorithms and cannot be seen as a separate issue. Szeliski (2010) provides a good and detailed synopsis of the main developments in computer vision over the last 30 years, as well a rich number of references used along the text for the researcher that wishes to go deeper in any particular subject.

In the context of stereo vision and depth recovery, in many cases, the overall performance of the machine vision system strongly depends on the accuracy of the camera calibration. Camera calibration is the process of determining camera geometric and optical characteristics and the 3D position and orientation of the camera frame relative to an external world frame (Heikkila et al., 1997). The epipolar geometry is implicitly connected with the pose and calibration of the stereo cameras, once this geometry is computed the epipolar line corresponding to a pixel in the left image can be used to constrain the search for a corresponding pixel in the right image.

Stereo vision besides being studied for long time it is still considered a mature technology. Recovering depth information requires great performance since pixel correspondence from left image to the right image need to be found what can be challenging if the images are from very

different viewpoints or contains noise, occlusions, homogeneous regions or unpredictable environment light conditions that makes it difficult, moreover depth based applications such as navigation for mobile robots requires high efficiency for a real-time response what makes it even more challenging (May et al., 2007).

Research problem: Recover of the 3D information lost in the process of projecting a 3D scene into an 2D image with OpenCV image processing platform.

#### **5.1.2.1 Camera Calibration.**

- Problem: Internal camera geometric and optical characteristics (intrinsic and extrinsic parameters) as well as the 3D position and orientation of the cameras frame relative to an external coordinate frame and relative to each other camera coordinates are unknown in a standard stereoscopic configuration.
- Solution: Algorithm for calibrating a camera with possibly variable intrinsic parameters and position, that copes with an arbitrary number of calibration planes and camera views. Calibration is achieved by projecting the planar calibration object into 2D image, each projection contributes with a system of homogeneous linear equations in the intrinsic parameters which are easily computed by solving the linear equations (Sturm, & Maybank, 1999).

#### **5.1.2.2 Rectification.**

- Problem: Canonical stereoscopic configurations are rare with a real stereo system since the two cameras almost never have coplanar and row-aligned imaging planes as desired for a more reliable and computationally efficient stereo correspondence.
- Solution: Reproject the image planes of the two cameras so that the epipolar line with the conjugate epipolar line become coincident with the horizontal scan-line reducing stereo matching from a 2D to a 1D search (Bradski et al., 2008).

#### **5.1.2.3 Correspondence.**

- Problem: Stereo analysis is the process of retrieving the depth information based on the 3D object/scene projection on two or more images. Finding corresponding pixels between both images or sequence of images constitutes the stereo analysis fundamental problem.
- Solution: Use a fast and effective area-based stereo matching algorithm that compares each small area with other area in a search window and then determines the extreme value of the correlation at each pixel resulting in a value that holds the disparity value between the left and right image patches at the best match that will result in a final disparity image (Konolige, 1997, Bradski et al., 2008).

The literature solutions previously described are strictly connected with their implementation in the OpenCV Image processing library what provides a better tool for a new researcher in this field. However due the vast number of research areas in the computer vision field and its fast progress in the last decades the need for standardization and definition of each individual research area within this field is needed for a better overall understanding. For a new researcher in this field of studies the literature still is one of the main obstacles for a fast learning curve. The link between the scientific and statistical approaches (vision analysis and formulation) and the engineering approach (algorithms implementation) constitutes the literature main gap (Szeliski, 2010). A better

explanations about the image programming languages available, their advantages and disadvantages as well as a methodology to analyse the efficiency and cost of the huge number of existing algorithms are also needed in the current literature.

### **5.1.3 Purpose of the Study**

#### **5.1.3.1 Purpose Statement.**

The purpose of this study was to utilize the OpenCV image processing platform together with Microsoft Visual Studio 2008 software to implement a program for recovering 3D information from video sequences captured with two Phantom v9.1 cameras arranged on a stereo configuration.

#### **5.1.3.2 Need/Rationale for the Study.**

In the recent decades the concept of Open Source has been increasing gradually captivating the interest of professionals and young researchers in the different areas. OpenCV image processing platform for computer vision is not exception and has been assisting to a great acceptance by the researchers community. Due the numerous functionalities it provides and the good documentation as well a vast community that can interact and provide fast answers, it was the image processing platform chosen to conduct the study. Stereo Vision is the computer vision area that during the last decades has gained special focus due its capacity to recover the depth information from two or more images. It is widely used in different applications such as surveillance, agriculture, mobile robotics, manufacturing and medical image analysis. This wide range of possible applications allied with the constant progress in innovative algorithms and increasing demand on the 3D computer graphics constitutes one of the main reasons to conduct the study using stereo vision approach.

#### **5.1.3.3 Description of the Study.**

In order to recover the depth from stereo video sequences, the researcher made use of OpenCV algorithms to implement a program with different stereo functionalities. This program studies the optimal number of calibration views needed, computes camera calibration using two methods and performs calibration optimization by excluding calibration views with higher error contributions. Calibrated and uncalibrated rectification methods were implemented in order to obtain the remapping maps and the disparity-to-depth matrix using different approaches. To recover the depth from stereo video sequences two methods were implemented: the first method used the stereo matching algorithms available from OpenCV libraries namely the Block Matching, Semi Global Block Matching, and Graph Cut algorithm to compute disparity images and then reproject it to 3D space, or the second method that made use of Lucas – Kanade Pyramid tracker code and mouse click event to track a set of points of interest over a stereo video sequence, compute its disparity and compute their corresponding 3D points related to the left or to the right camera coordinate system. Stereo video sequences were captured using two Phantom v9.1 high-speed cameras arranged on a standard stereo configuration. Different stereo configurations were used to record video from a calibration pattern and later from a MELFA RV-2AJ robot's end-effector movement. All the research output results were stored in xml file formats with proper nomenclature depending on which stereo operation was performed.

#### **5.1.3.4 Expected Outcomes.**

The expected outcomes of this case study are to develop programming skills using a C++ object oriented approach together with the newer and more efficient OpenCV C++ interface. By implementing a main program to deal with 3D points recovering it is expected to obtain a number

of ready-to-use classes capable to read a list of calibration views and AVI files, compute calibration parameters and stereo relations and optimize those results, undistort and remap stereo video captures from a standard stereo configuration and recover the depth of a captured scene (disparity image) or sparse set of points from the available stereo video sequences. Another goal of the study was to provide a more practical approach through laboratory experiments with two Phantom v9.1 cameras which it is expected to develop a better understanding how stereo configuration's settings, capture settings and environment settings influences the outputs results obtained from the captured video sequences for stereo calibration and 3D information recovering purposes.

#### **5.1.4 Research Questions**

1. *Which are the OpenCV main functions involved in the process of: stereo camera calibration, stereo image rectification, stereo matching and points reprojection into 3D space, and Lucas – Kanade Pyramid optical flow method. What are the inputs and outputs arguments of those functions.*
2. *How to compute camera calibration parameters using a planar calibration object known as chessboard and how to relate two cameras in a stereo configuration. How many calibration views are needed to perform the stereo calibration process and which calibration method (with and without initial guess to compute stereo relations) gives better results. How to optimize the stereo calibration process and improve the calibration parameters results.*
3. *Which are the differences between using calibrated and uncalibrated rectification methods and how to implement the image rectification process by means of using OpenCV functions.*
4. *How to parametrize the stereo relation's rotation matrix into Euler angles and quaternions and how to perform the transformation between this two rotation representations.*
5. *How to compute the disparity image and disparity of a sparse set of points given two rectified images captured from a stereo configuration previously calibrated. How to reproject a sparse set of points to the 3D space.*

#### **5.1.5 Significance of the field**

The contributions resulting from this study to the research literature were various. By using the OpenCV libraries' algorithms this case study provides a number of functionalities to work with video capture and video recording operation, capture and list stereo images for calibration purposes or perform simple tasks such as images colour space transformation or frame saving operations using different image formats. Related with the calibration process a number of functions were implemented within a class to allow single or stereo cameras calibration process using a text file with a list of calibration views to be loaded or alternatively it allows to perform the calibration process directly from AVI video files or real time video capture. Additionally was implemented a method that by excluding bad views used in the calibration process and recalling the calibration process again it allows to optimize substantially the final calibration parameters and reduce the reprojection errors. To perform the uncalibrated and calibrated stereo rectification after each stereo calibration process was implemented another class that can be easily reused in the future research. An additional class was built with functions that allows to capture video sequences from stereo

cameras or stereo AVI files and remap the images to compute the dense disparity image or the disparity for a sparse set of points and then proceed with the reprojection to 3D space. The case study will also provide conclusions and knowledge achieved during the laboratories experiments, as well the procedures' changes that allowed to improved substantially the calibration and 3D recovering results.

### 5.1.6 Definitions

- $f_x, f_y$  : Camera focal length in pixel units on XX and YY direction, respectively.
- $R$  : Stereo relation 3-by-3 rotation matrix that brings (rotates) the right camera to the left camera orientation.
- $Q$  : Disparity-to-depth transformation matrix used to reproject 2D image points to 3D world space.
- $D1(D2)$  : Left (right) camera distortion coefficients vector  $D=[K_1 K_2 p_1 p_2 K_3]^T$  where  $K_1, K_2$  and  $K_3$  (*only for wide-angle lenses*) are the radial distortion coefficients and  $(p_1, p_2)$  the tangential distortion coefficients.
- $L01, L02, L03$  and  $S01, S02, S03, S04$  : Nomenclature used to identify the different stereo video sets (S01, S02, S03, and, S04) recorded during the three laboratory sessions (L01, L02, and L03). If the nomenclature is used in the calibration context it refers to the video sequences captured for calibration purposes otherwise it refers to the video sequences captured for recovering 3D information purposes.
- $c_x, c_y$  : Principal point coordinates in pixel units.
- $map1_x, map1_y, (map2_x, map2_y)$  : Remapping maps for the left (right) camera's video capture that are used to perform the (undistortion+rectification) transformation.
- $T$  : Stereo relation translation vector  $T=[T_x T_y T_z]^T$  that brings (translates) the right camera to the left camera position.
- Camera calibration: Process of finding the camera intrinsic and extrinsic parameters such as focal length, principal point and lens distortion parameters.
- Camera Matrix: Projective transform matrix that relates the real world coordinates to the points on the image plane.
- Canonical Stereo Configuration: Stereo configuration where the cameras' imager are ideally coplanar and row aligned.
- CSR: Current Session Reference available from phantom camera control software.
- CV: Computer Vision.
- Disparity: The difference between two image points, representing the same 3D point in the world scene, within two stereo images.
- E: Essential matrix: Contains information about the rotation and translations that relates the two cameras on the stereo configuration.
- Epipolar geometry: The geometry relations between the 3D points and their projections into the 2D image planes that form a number of constraints between two stereo images' points.

- Epipolar line: Is the line formed by the epipolar plane's intersection with the camera's imager plane.
- Epipolar plane: Is the plane passing through an object point and the cameras' centres of projection.
- Euler angles: describe the rotations that moves a rigid body from one referential to another with different orientation by using only three parameters (yaw, pitch, roll).
- Extrinsic parameters: Are the parameters that define the camera's position and orientation (three rotation  $(\Phi, \Theta, \Psi)$  and three translation  $(T_x, T_y, T_z)$  parameters) with respect to a known 3D world reference frame.
- F: Fundamental matrix that relates the two cameras, on a stereo configuration, in pixels coordinates.
- Focal Plane: The plane in a camera, or other optical instrument in which a real image is in focus.
- HVS: Human Vision System.
- IDE: Integrated Development Environment.
- Intrinsic parameters: Are the parameters that define the camera's optical and geometric characteristics such as the focal length, the principal point coordinates and the radial and tangential lens distortions.
- Lens Distortion: Lens imperfections that introduces distortions on the image's pixel locations.
- Occlusions: Regions that are originated by disparity discontinuities.
- OOP: Object Oriented Programming.
- OpenCV: Open Source Computer Vision Programming Library.
- Optical flow: Is the velocity field in the image plane resulting from the motion of the objects being observed, the motion of the observer, or apparent motion which may be caused by changes in the image intensity between frames.
- PCC: Phantom Camera Control software.
- Planar Calibration Object: Object used to capture images for camera calibration purposes using OpenCV algorithms.
- Principal Axis: Line that passes through the lens curvature's center, also known as optical axis.
- Principal Plane: Plane that is perpendicular to the lens optical axis.
- Principal Point: Point that results from the intersection of the image plan and the optical axis.
- Quaternions: Mathematical notation for representing rotations and orientations of objects or frames in 3D space.
- SDK: Software Development Kit.
- Standard Stereo Configuration: The real stereo configuration where the cameras' imager are not ideally coplanar or row aligned as in the canonical configuration.

- Stereo Correspondence: The process of matching image points from two different images, captured from a stereo configuration, that represent the same object points on the 3D world space.
- Stereo Rectification: Relates the two cameras in the space by means of rotations and translations and the result are pair of images row-aligned and rectified.
- StereoVisionProg: Researcher-made program implemented to answer the proposed research questions.
- STL: Standard Template Library.
- UML: Unified Modeling Language.
- Undistortion: Process of computing undistorted images (corrected images) by mathematically removing radial and tangential distortions.

#### **5.1.7 Limitations**

The limitation of this research are mainly inherent to the data analysis and time limitations. The time available to make this research was exceptionally short for a complete and solid review of the extensive stereo imagery literature as well to learn the C++ programming language and to obtain a complete familiarization with the OpenCV algorithms. Due the lack of time, the initial purpose for implementing an interface able to interact with the Phantoms v9.1 cameras using the Phantom SDK was not implemented which constitutes also a limitation in this study.

In what concerns to the dense stereo matching methods the research design adopted did not included other stereo matching algorithms besides the ones available from OpenCV 2.1 libraries, the study also did not included any comparison related with speed and computational cost between those stereo matching algorithms, this limited the conclusions obtained from dense stereo matching and the possibilities to obtain better disparity images.

## 6 Chapter Two

### 6.1 *Review of the Literature*

#### 6.1.1 *Introduction*

The world we humans daily perceive is the 3D world, but the images captured from it are 2D, one dimension is lost in the capturing process. One important task in the Computer Vision field is to recover back the third dimension (Shah, 1997). There are several methodologies to recover the 3D information from 2D images, one of them widely studied in computer vision is the stereo imagery approach. In stereo imagery approach two images ( left and right ) are used to recover the depth information. The depth recovery relies on three main areas: the first consists on capturing a number of image pairs of a planar object that will be used to calibrate both cameras, this area is termed as calibration. The second area consist on correcting and remapping each pixel on both images in such a way that the images are suitable to apply matching algorithms, this area is termed as rectification. Finally after applying the calibration and rectification processes, respectively, a third step need to be performed to compare and compute the disparity maps, this area is termed as correspondence.

The literature review will address three areas related to the depth recovery using stereo imagery. The first section will address research related to the cameras calibration. The second section will focus on research studies about the images rectification. Finally the third section will discuss research related to stereo correspondence problem.

#### 6.1.2 *Research Synthesis*

##### 6.1.2.1 *Camera calibration review.*

- *Introduction*

One of the main goals of computer vision is to understand the visible world by inferring 3D properties from 2D images (Jiang, & Zhao, 2010). In the context of stereo imagery the first step that need to be performed in the process of recovering 3D information from 2D images is known by the term *calibration*. Camera calibration is the process of computing the internal camera geometric and optical characteristics and modelling the relationship between 2D images and 3D world.

A large number of calibration methods are presented in the literature. The literature suggest that this methods can be characterized in three main categories: traditional methods, self-calibration and the active-motion based methods. The former method, the one that will be reviewed, is performed by observing a calibration object whose exact geometry in 3D space is known with precision. This method provided by Zhang (2000) was particularly of the research interest once it provides similar methodology to the one implemented by OpenCV platform, as well a common ground for data comparison.

- *Purpose*

The purpose of the study is to provide an easy to use and flexible new technique to easily



calibrate a camera by observing a planar pattern shown at at least two different orientations. Either the camera or the planar pattern can be freely moved without the need to know the motion (Zhang, 2000).

- **Methods**

The calibration is performed by observing a planar calibration object whose geometry is known in 3D world with good precision. This methodology avoids the use of expensive calibration apparatus such the ones based on three coplanar planes or diffractive optical elements, instead it uses a pattern that can be easily printed on a laser printer and attached to a planar surface. Either the camera or the calibration object can be moved by hand to provide a rich set of pattern orientations.

*Calibration Procedure.* This section will provide the formulation to compute the camera calibration parameters. Firstly it presents the notation and the planar homography, then the analytical solution followed by a non linear optimization without and with lens distortion effects and finally the procedure summary.

### Notation

A 2D point is represented by  $m=[u,v]^T$  and a 3D point is represented by  $M=[X,Y,Z]^T$ ,  $\tilde{m}$  and  $\tilde{M}$  denote the augmented vector by adding 1 resulting in:  $\tilde{m}=[u,v,1]^T$ ,  $\tilde{M}=[X,Y,Z,1]^T$  respectively. The camera is modelled by the pinhole model. The relation between the 3D point  $M$  and its image projection  $m$  is:

$$s \tilde{m} = A [R t] \tilde{M} \quad (6.1.1)$$

With  $A = \begin{bmatrix} \alpha & c & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}$ .

The extrinsic parameters are the rotation and the translation  $(R, t)$  that relates the world coordinate system to the camera coordinate system.  $A$  is the camera intrinsic matrix and  $(u_0, v_0)$  are the coordinates of the principal point. The  $\alpha, \beta$  are the scale factors along  $u, v$  image axis, and  $c$  the skewness of the two image axes.

Along the article  $A^{-T}$  is used in place of  $(A^{-1})^T$ , or  $(A^T)^{-1}$ .

### Planar Homography

Without lose of generality, the model plane is assumed to be on  $Z=0$  world coordinate system. Denoting the  $i^{th}$  column of the rotation matrix  $R$  by  $r_i$ . From equation (6.1.1), is possible to obtain the following relation:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \begin{bmatrix} r_1 & r_2 & r_3 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = A \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Since  $Z=0$  for all the planes,  $M$  was redefined to denote a point on the model plane  $M=[X,Y]^T$  and consequently  $\tilde{M}=[X,Y,1]$ . A model point  $M$  and its image

$m$  is related by a homography  $H$  :

$$s \tilde{m} = H \tilde{M} \quad (6.1.2)$$

With  $H = A[r_1 r_2 t]$ . The 3-by-3 matrix  $H$  is defined up to a scale factor.

### Intrinsic Camera Parameters Constraints

Given an image of the model plane, the planar homography can be estimated. Denoting it by  $H = [h_1 h_2 h_3]$  and by substitution in equation (6.1.2) the following relation was obtained:  $[h_1 h_2 h_3] = \lambda A[r_1 r_2 t]$ , where  $\lambda = 1/s$ . Given an homography and using the knowledge that  $r_1$  and  $r_2$  are orthonormal vectors, two basic constraints on the intrinsic parameters are obtained :

$$h_1^T A^{-T} A^{-1} h_2 = 0 \quad (6.1.3)$$

$$h_1^T A^{-T} A^{-1} h_1 = h_2^T A^{-T} A^{-1} h_2 \quad (6.1.4)$$

Because a homography has 8 degrees of freedom and there are 6 extrinsic parameters (3 for rotation and 3 for translation), only 2 constants can be obtained on the intrinsic parameters.

### Solving Camera Calibration

Once presented the notation and planar homography, this section summarize the article methodology to solve the camera calibration problem. Firstly was presented the analytical solution followed by a non linear optimization technique and finally the consideration of radial distortion in the calibration process.

#### *Closed-form solution.*

Let

$$B = A^T A^{-1} \equiv \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix} \quad (6.1.5)$$

$$B = \begin{bmatrix} \frac{1}{\alpha^2} & -\frac{c}{\alpha^2 \beta} & \frac{c v_0 - u_0 \beta}{\alpha^2 \beta} \\ -\frac{c}{\alpha^2 \beta} & \frac{c^2}{\alpha^2 \beta^2} + \frac{1}{\beta^2} & -\frac{c(c v_0 - u_0 \beta)}{\alpha^2 \beta^2} - \frac{v_0}{\beta^2} \\ \frac{c v_0 - u_0 \beta}{\alpha^2 \beta} & -\frac{c(c v_0 - u_0 \beta)}{\alpha^2 \beta^2} - \frac{v_0}{\beta^2} & \frac{(c v_0 - u_0 \beta)^2}{\alpha^2 \beta^2} + \frac{v_0^2}{\beta^2} + 1 \end{bmatrix}$$

B is a symmetric matrix defined by 6D vector

$$b = [B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}]^T \quad (6.1.6)$$

Defining the  $i^{th}$  column vector of  $H$  as  $h_i = [h_{i1} h_{i2} h_{i3}]^T$  then a new relation is obtained:

$$h_i^T B h_j = v_{ij}^T b \quad (6.1.7)$$

With  $v_{ij} = [h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i2}h_{j2}, h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3}]^T$ . Therefore the two fundamental constraints (6.1.3) and (6.1.4), from a given homography, can be rewritten as 2 homogeneous equations in  $b$ :

$$\begin{bmatrix} v_{12}^T \\ (v_{11} - v_{22})^T \end{bmatrix} b = 0 \quad (6.1.8)$$

If  $n$  images of the model plane are observed, by stacking  $n$  such equations as (6.1.8) a new relation is obtained:

$$Vb = 0 \quad (6.1.9)$$

Where  $V$  is a  $2n - \text{by} - 6$  matrix. If  $n \geq 3$  a unique solution is obtained up to a scale factor.

If  $n=2$ , the skewness constraint can be imposed to be zero,  $c=0$ , i.e. an additional equation  $[0, 1, 0, 0, 0, 0]b = 0$  is added to the equation (6.1.9).

The solution to equation (6.1.9) is known as the eigenvector of  $V^T V$  associated with the smallest eigenvalue. Once  $b$  is estimated, the camera intrinsic matrix  $A$  and the extrinsic parameters can be computed. From equation (6.1.2) the rotations and translation can be easily obtained:

$r_1 = \lambda A^{-1} h_1$ ,  $r_2 = \lambda A^{-1} h_2$ ,  $r_3 = r_1 \times r_2$  and  $t = \lambda A^{-1} h_3$  with  $\lambda = 1 / \|A^{-1} h_{1 \text{ or } 2}\|$ . Additional computation needs to be performed in the returned matrix in order to solve the best rotation matrix  $R$ , i.e. the one that satisfy a rotation matrix requirements.

### Maximum likelihood estimation

Given  $n$  images of a model plane containing  $m$  points and assuming that the image points are affected by independent and identically distributed noise the maximum likelihood estimate can be obtained by minimizing an algebraic distance which is not physically meaningful. This is done by minimizing the following equation:

$$\sum_{i=1}^n \sum_{j=1}^m \|m_{ij} - \tilde{m}(A, R_i, t_i, M_j)\|^2 \quad (6.1.10)$$

Where  $\tilde{m}(A, R_i, t_i, M_j)$  is the projection of point  $M_j$  in image  $i$ , according to equation (6.1.2). The rotation  $R$  is parametrized by a vector  $r$  of 3 parameters which is parallel to the rotation axis and with magnitude equal to the rotation angle.  $R$  and  $r$  are related by the Rodrigues formula. The non linear minimization problem of equation (6.1.10) is solved with Levenberg - Marquardt algorithm.

### Considering Radial Distortion

The previous steps of this article did not take into account the lens distortion. This section will summarize the methodology used to estimate the camera intrinsic and extrinsic parameters considering the first two terms of radial distortion.

Let  $(u, v)$  be the ideal or distortion-free pixel image coordinates and  $(\check{u}, \check{v})$  the corresponding real observed image coordinates. Similarly the  $(x, y)$  and  $(\check{x}, \check{y})$  are the ideal or distortion-free and real or distorted normalized image coordinates, thus the real pixels are given by:

$$\begin{aligned}\check{x} &= x + x[K_1(x^2 + y^2) + K_2(x^2 + y^2)^2] \\ \check{y} &= y + y[K_1(x^2 + y^2) + K_2(x^2 + y^2)^2]\end{aligned}$$

where  $K_1$  and  $K_2$  are the coefficients of the radial distortion. The center of radial distortion is the same as the point formed by the intersection of the principal ray with the image plane (principal point). Given the relation  $\check{u} = u_0 + \alpha \check{x} + c \check{y}$  and  $\check{v} = v_0 + \beta \check{y}$ , the real image coordinates are given by the following equations:

$$\check{u} = u + (u - u_0)[K_1(x^2 + y^2) + K_2(x^2 + y^2)^2] \quad (6.1.11)$$

$$\check{v} = v + (v - v_0)[K_1(x^2 + y^2) + K_2(x^2 + y^2)^2] \quad (6.1.12)$$

The method to compute both distortion parameters assume that initially these parameters are small and thus ignored to compute the five intrinsic parameters. Then the method estimates  $K_1$  and  $K_2$  based on those five parameters. From (6.1.11) and (6.1.12) each point in the image is constrained by two equations:

$$\begin{bmatrix} (u - u_0)(x^2 + y^2) & (u - u_0)(x^2 + y^2)^2 \\ (v - v_0)(x^2 + y^2) & (v - v_0)(x^2 + y^2)^2 \end{bmatrix} \begin{bmatrix} K_1 \\ K_2 \end{bmatrix} = \begin{bmatrix} \check{u} - u \\ \check{v} - v \end{bmatrix}$$

Given  $m$  points in  $n$  images, all the equations are joined to obtain a system of  $2mn$  total equations. This system of equations can be presented as  $DK = d$  where  $K = [K_1, K_2]^T$ . The linear least-squares solution is obtained by:

$$K = (D^T D)^{-1} D^T d \quad (6.1.13)$$

After solving the system of equations and obtaining the values of  $K_1$  and  $K_2$  the solution for the previous five intrinsic parameters can be refined by solving equation (6.1.10) with the two new distortion parameters taken into account. Similarly to equation (6.1.10) the new set of parameters are estimated by minimizing the following functional:

$$\sum_{i=1}^n \sum_{j=1}^m \|m_{ij} - \tilde{m}(A, K_1, K_2, R_i, t_i, M_j)\|^2 \quad (6.1.14)$$

Where  $\tilde{m}(A, K_1, K_2, R_i, t_i, M_j)$  is the projection of point  $M_j$  in the distorted image  $i$ , according to equations (6.1.2), (6.1.11) and (6.1.12). The non linear minimization is solved in the same way as demonstrated previously for calibration neglecting lens distortion. The literature suggests that a second approach can be done to initially estimate the

$K_1$  and  $K_2$  values by simply setting them to zero.

### Procedure Summary

The researcher of this article recommend the following procedure:

1. Print a pattern and attach it to a planar surface.
  2. Take a set of images from different orientations by moving either the camera or the model plane.
  3. Identify the points of interest in the image.
  4. Estimate the five intrinsic and extrinsic parameters neglecting radial distortions.
  5. Estimate radial distortions coefficients by solving the linear least-squares equation (6.1.13).
  6. Recompute all parameters by minimizing equations (6.1.14).
- Variables and Data Analysis

This article presents two distinct analysis. In the first part the article presents the computer simulated analysis for the algorithm performance with respect to the noise level, number of planes and the orientation of the model plane while the second part uses real data to analyse the influence of the number of planes in the intrinsic parameters and the influence of including the distortion coefficients on the refinement of those parameters.

### Simulated Data Analysis

The camera matrix used (notation of equation (6.1.1) ) was:

$$A = \begin{bmatrix} 1250 & 1.09083 & 255 \\ 0 & 900 & 255 \\ 0 & 0 & 1 \end{bmatrix}.$$

The pattern has a size of 18cm x 25cm containing 10 x 14 corners points with an image resolution of 512 x 512.

*Algorithm performance varying the noise level.* Three planes are used with the following orientations and positions, respectively:

$$\begin{aligned} r_1 &= [20^\circ, 0, 0]^T, t_1 = [-9, -12.5, 500]^T \\ r_2 &= [0, 20^\circ, 0]^T, t_2 = [-9, -12.5, 510]^T \\ r_3 &= \frac{1}{\sqrt{5}} [-30^\circ, -30^\circ, -15^\circ]^T, t_3 = [-10.5, -12.5, 525]^T \end{aligned}$$

Gaussian noise with 0 mean and standard deviation is added to the projected image points. An 100 independent trials is performed by varying the noise level from 0.1 to 1.5 pixels. The relative error for  $\alpha$  and  $\beta$  are measured as well the absolute error for  $u_0$  and  $v_0$ .

*Algorithm performance varying the number of planes.* This section investigates the algorithm performance varying the number of images of the model plane. The number of images vary from 2 to 16. For the first three images the orientation and position of the model plane are the same as in the previous section and from the fourth image a rotation angle of  $30^\circ$  is applied to an arbitrary rotation axis. For each number, 100 trials of independent plane orientations and independent noise with mean 0 and standard deviation 0.5 pixels are conducted.

*Algorithm performance varying the orientation of the model plane.* This section study the influence of the model plane orientation with respect to the image plane. The plane is initially parallel to the image plane and a rotation axis is chosen arbitrarily and the plane is

then rotated around that axis by an angle  $\theta$  that varies from  $5^\circ$  to  $75^\circ$ . Gaussian noise with mean 0 and standard deviation 0.5 pixels is added to the projected image points. The process is repeated 100 times and the average error are computed.

### Real Data Analysis

In the practical part of the study the images were captured with a PULNiX CCD camera 6 mm lens with 640x480 resolution. The model plane used a pattern of 8x8 squares with a size of 17 x 17 cm. Five images from different orientations were taken.

The algorithm was applied to sets of different number (2,3,4,5, respectively) of images and the intrinsic camera parameters were measured first neglecting the lens distortion and secondly by using the maximum likelihood estimation(MLE) after including the radial distortion coefficients effects. The estimated standard deviation was computed for each intrinsic parameter and distortion coefficients as well the root mean square (RMS) for each set of images. A second approach was implemented in order to study the stability of the proposed algorithm by applying the algorithm to all set of images combinations. The mean and deviation were computed for each intrinsic parameter and distortion coefficients as well the RMS for each combination.

- *Results*

The results returned by this study are summarized in two categories: The results concerning to the simulated data and the results concerning to the real data.

The results obtained for the simulated data are firstly discussed and presented in the same order. All the figures here included were adapted from this article.

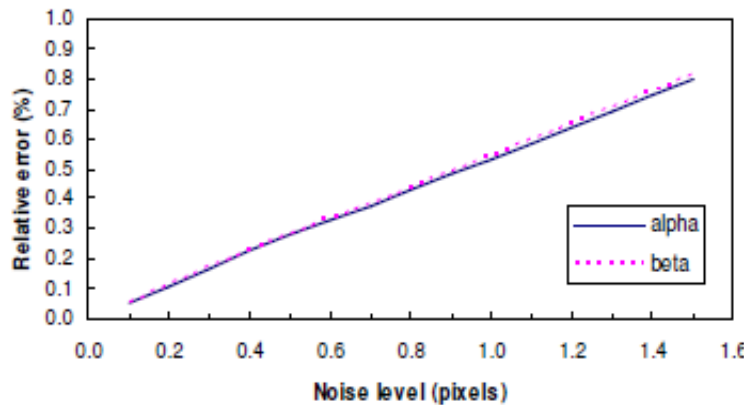


Figure 6.1: Relative errors vs. noise level ( $\alpha$ ,  $\beta$ ), Zhang (2000).

From Figure 6.1 and 6.2 is possible to conclude that both errors increase linearly with the noise level. Taking in account that in real cases the noise is normally lower than 0.5 ( $\sigma \leq 0.5$ ) the errors for  $\alpha$  and  $\beta$  for that level of noise are smaller than 0.3%. In the case of  $u_0$  and  $v_0$  the errors are less than 1.5 pixels however  $v_0$  presents a lower error that for the same noise value that is less than 1 pixel due the fact that the pattern has more corners along  $v$  direction than in  $u$  direction. To have similar comparison a square pattern should be used.

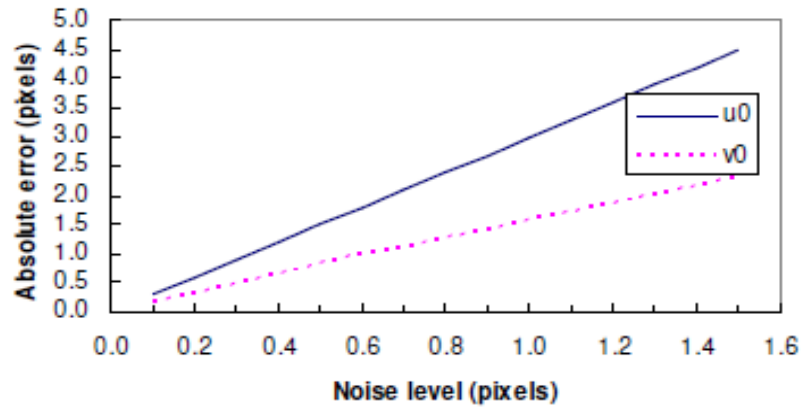


Figure 6.2: Absolute errors vs. noise level ( $u$ ,  $v$ ), Zhang (2000).

From Figure 6.3 and 6.4 the relative error of both scale factors ( $\alpha$  and  $\beta$ ) decrease significantly as the number of planes increase and tends to stabilize for a relative error around 0.25% for a number of images greater than 11. The principal point coordinates error curves present the same tendency with a vertical displacement between them due difference in the number of samples as already mentioned above in the noise results.

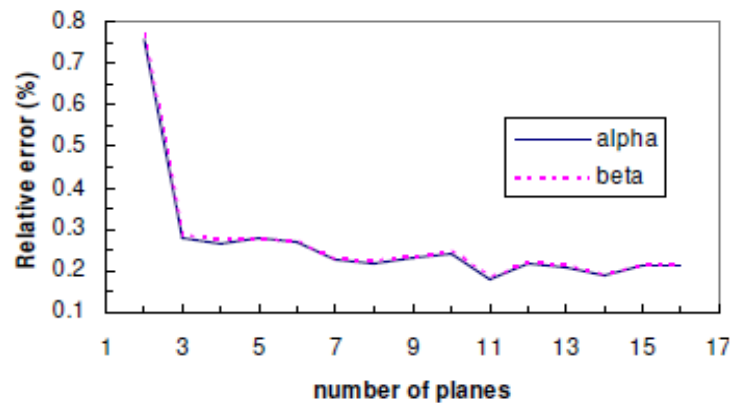


Figure 6.3: Relative error vs. number of planes ( $\alpha$ ,  $\beta$ ), Zhang (2000).

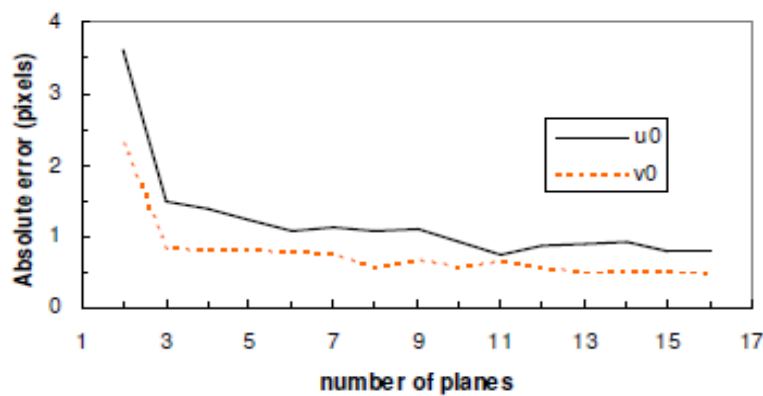


Figure 6.4: Absolute error vs. number of planes ( $u_0$ ,  $v_0$ ), Zhang (2000).

From Figure 6.5 and Figure 6.6 is possible to conclude that all parameters have minimum errors for angles within an interval of  $[40^\circ; 60^\circ]$ . The higher errors occur for small angles where the planes are almost parallel to each other and do not provide additional constraints on the camera intrinsic parameters (degenerate configurations). This occurrences, for some calibration algorithms, introduces numeric instability and can make the solution diverge to wrong results.

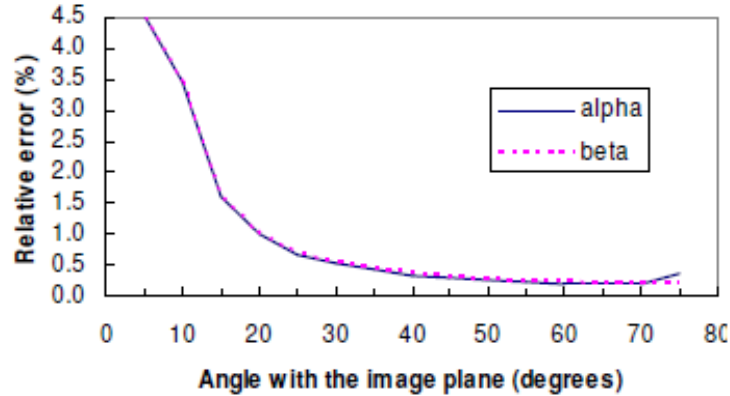


Figure 6.5: Relative error vs. angle with image plane ( $\alpha$  and  $\beta$ ), Zhang (2000).

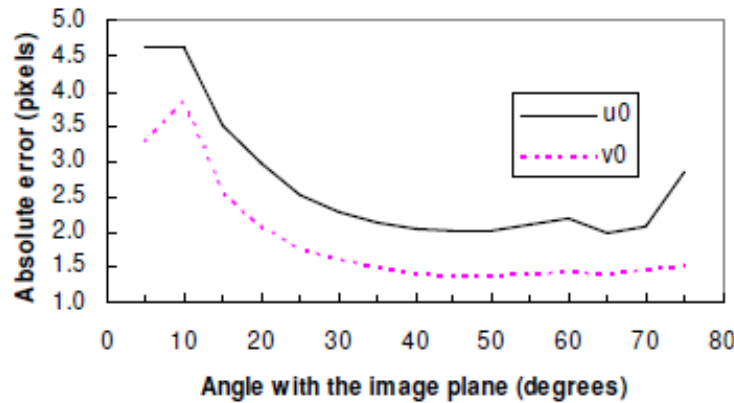


Figure 6.6: Absolute error vs. angle with image plane ( $u_0$ ,  $v_0$ ), Zhang (2000).

- Real data results

The algorithm results using real data are showed in Figure 6.7 and 6.8. Figure 6.7 shows the calibration parameters results using sets of 2, 3, 4 and 5 images. The “initial” column are the values obtained for the case were radial distortion was neglected and the “final” column are those parameters refined after estimate radial distortion coefficients

$K_1$  and  $K_2$ , the third column ( $\sigma$ ) is the estimated standard deviation. From this third column is possible to conclude that the parameter values do not present significant differences in each set and the standard deviation converges rapidly by only increasing the number of images from 2 to 5. Figure 6.8 shows the combinations of all sets of images. The mean and sample deviation are showed in the last columns. The higher deviation occurs for the scale factors ( $\alpha$  and  $\beta$ ) but still considerably small what shows that the algorithm proposed is stable. The aspect ratio ( $\alpha/\beta$ ) was also computed for each combination and its mean and sample deviation are 0.99995 and 0.00012, respectively. The value very close to 1 shows that the camera CCD used was square, i.e. the sizes ( $s_x$  and  $s_y$ ) of the individual imager element are equal.



nb	2 images			3 images			4 images			5 images		
	initial	final	$\sigma$	initial	final	$\sigma$	initial	final	$\sigma$	initial	final	$\sigma$
$\alpha$	825.59	830.47	4.74	917.65	830.80	2.06	876.62	831.81	1.56	877.16	832.50	1.41
$\beta$	825.26	830.24	4.85	920.53	830.69	2.10	876.22	831.82	1.55	876.80	832.53	1.38
$\gamma$	0	0	0	2.2956	0.1676	0.109	0.0658	0.2867	0.095	0.1752	0.2045	0.078
$u_0$	295.79	307.03	1.37	277.09	305.77	1.45	301.31	304.53	0.86	301.04	303.96	0.71
$v_0$	217.69	206.55	0.93	223.36	206.42	1.00	220.06	206.79	0.78	220.41	206.59	0.66
$k_1$	0.161	-0.227	0.006	0.128	-0.229	0.006	0.145	-0.229	0.005	0.136	-0.228	0.003
$k_2$	-1.955	0.194	0.032	-1.986	0.196	0.034	-2.089	0.195	0.028	-2.042	0.190	0.025
RMS	0.761	0.295		0.987	0.393		0.927	0.361		0.881	0.335	

Figure 6.7: Parameter result's variations with different sets of images, Zhang (2000).

quadruple	(1234)	(1235)	(1245)	(1345)	(2345)	mean	deviation
$\alpha$	831.81	832.09	837.53	829.69	833.14	832.85	2.90
$\beta$	831.82	832.10	837.53	829.91	833.11	832.90	2.84
$\gamma$	0.2867	0.1069	0.0611	0.1363	0.1096	0.1401	0.086
$u_0$	304.53	304.32	304.57	303.95	303.53	304.18	0.44
$v_0$	206.79	206.23	207.30	207.16	206.33	206.76	0.48
$k_1$	-0.229	-0.228	-0.230	-0.227	-0.229	-0.229	0.001
$k_2$	0.195	0.191	0.193	0.179	0.190	0.190	0.006
RMS	0.361	0.357	0.262	0.358	0.334	0.334	0.04

Figure 6.8: Calibration parameter's results with different image sets, Zhang (2000).

### Conclusions/Implications

The algorithm proposed in this article provides an easy and flexible method to calibrate a camera by capturing images either moving the camera or the pattern plane. From the simulated data few conclusions can be done: better results are obtained for a rich set of images, i.e. with distinct orientations for angles within an interval of  $[40^\circ; 60^\circ]$ , images that only differ in position (pure translation) do not contribute with any additional information, and for a number of image planes greater than 11 the parameters errors are approximately constant. The real data computation also allows to conclude that the algorithm do not require a big number of images, the algorithm converges rapidly.

The proposed technique is flexibly, reliable and do not requires large number of images neither very expensive or elaborated calibration objects making it easy to use. It present the methodology also used by the OpenCV calibration functions used along this study.

### Weakness/Limitations

This paper do not mentions explicitly any kind of limitations however some of its assumptions constitutes few limitations. The method assumes that the radial distortion function is mostly dominated by the first two terms however it is not partially true for wide angle or fish-eye lens types where the third coefficient has significant weight. The method did not establish an interval for the angle of rotation that gives the better results or studied the influence of higher angles (closer to  $90^\circ$ ) vs. corners detection precision. Another limitation of this paper is due the fact it does not provide results with different patterns varying the number of corners and their sizes and the influence that this changes may cause in the calibration parameter results. Finally the fact of using rectangular patterns ( $n_{\text{corners along } u} \neq n_{\text{corners along } v}$ ) did not allowed to compare directly the principal point coordinates for different noise levels (Figure 6.2).

### 6.1.2.2 *Stereo rectification review.*

- *Introduction*

In stereo vision algorithms image rectification plays an important role. Assuming that a pair of 2D images captured from, two different viewpoints, from a scene in a 3D world and with their epipolar geometry already computed. The two corresponding points between the pair of images must satisfy the epipolar constraint. For a point in one image its corresponding point must lie along an epipolar line in the second image. In standard stereo rigs, the epipolar lines are, in general, not aligned or parallel with the coordinate axis what constitutes a major drawback to compute dense and accurate correspondences. Stereo matching problem can be easily simplified if epipolar lines are horizontally coincide, this is achieved by applying 2D projective transforms, or homographies, to each image such that the search space is reduced to only one dimension. This process is known as image rectification and is possible by using epipolar geometry.

There exist different approaches to rectify images. They can be classified mainly according to the type of geometric transformations used:

- *Projective rectification.* Is a linear transformation in the projective space and uses 2D plane to plane homographies.
- *Non-linear rectification.* Uses general geometric transformation of images.

Further classification of image rectification approaches is based on the method used to determine the free parameters left by the rectification conditions (Matousek, 2007).

A large number of rectification methods have been proposed, they initially were based mostly on calibration parameters however due to the requirements of most recent vision systems the late research focused its attention on uncalibrated methods.

Traditional or calibration based rectification methods (Avache & Hansen, 1988), (Avache & Lustman, 1991) and more recently (Fusiello, et al, 2000) and Bouguet algorithm that is a simplification of the method first introduced by Tsai(1987) and Zhang (1999, 2000) all require the knowledge of camera parameters that, for some vision systems, constitutes one disadvantage. Contrarily to the rectification based on traditional methods, several methods have been developed to allow image rectification directly without the need of using camera parameters, however most of these methods requires that the fundamental matrix needs to be first estimated. Papadimitriou and Dennis (1996) proposed a method for partially aligned cameras. Robert et al(1997) developed a method that attempts to find the transformation that better preserves orthogonality around image centres. Loop and Zhang(1999) proposed decomposing each homography into a specialized projective transform, a similarity transform, followed by a shearing transform considering carefully the image distortion at each stage. Pollefeys et al. (1999) presented a simple algorithm for rectification which can deal with all possible two views stereo image geometries. Hartley (1999) proposed a method that computes the homographies making use of the differences minimization between matching points in both images.

- *Purpose*

The purpose of the study is to provide a rationale for the calibrated/uncalibrated image rectification methods, already implemented in OpenCV library, applied to this research. This section is presented in two subsections that summarize a methodology to compute the homographies by using the minimization of the differences between matching points, Hartley (1999), and the Bouguet calibrated stereo rectification methodology described by Bradski et al. (2008).

- *Methods*

### Hartley's Uncalibrated Stereo Rectification

#### Notation

If  $A$  is a square matrix then its matrix of cofactors is denoted by  $A^*$  and the following identities  $A^*A = AA^* = \det(A)I$  where  $I$  is the identity matrix. If  $A$  is an invertible matrix, then  $A^* \approx (A^T)^{-1}$ .

Given a vector  $t = (t_x, t_y, t_z)^T$  the skew-symmetric matrix is as follows:

$$[t]_x = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix} \quad (6.1.15)$$

For any non-zero vector  $t$ , matrix  $[t]_x$  has rank 2. Furthermore, the null-space of  $[t]_x$  is generated by the vector  $t$  that means  $t^T[t]_x t = 0$  and any other vector cancelled by  $[t]_x$  is a scalar multiple of  $t$ . For any vectors  $s$  and  $t$  the cross-product are  $s^T[t]_x s = s \times t$  and  $[t]_x s = t \times s$ . Also for any 3-by-3 matrix  $M$  and vector  $t$

$$M^*[t]_x = [Mt]_x M \quad (6.1.16)$$

Projective Geometry: Real projective  $n$ -space consists on the set of equivalent non-zero vectors real  $(n+1)$  vectors, where two vectors are considered equivalent if they differ by a constant factor. A vector representing a point in  $P^n$  is known as a homogeneous coordinate representation of the point. Real projective  $n$ -space contains Euclidean  $n$ -space as the set of all homogeneous vectors with coordinates different than zero, then a point in  $P^2$  is represented by a vector  $u = (u, v, w)^T$ , for  $w \neq 0$ , this represents a point in  $R^2$  expressed in Euclidean coordinates as  $(u/w, v/w)^T$ .

Lines in  $P^2$  are also represented in homogeneous coordinates, the line  $\lambda$  with coordinates  $(\lambda, u, v)^T$  is the line consisting of points connected by the equation  $\lambda u + uv + vw = 0$  i.e. a point  $u$  lies on a line  $\lambda$  if and only if  $\lambda^T u = 0$ . The line joining two points  $u_1$  and  $u_2$  is given by the cross product  $u_1 \times u_2$ .

The projective transformation from  $P^n$  to  $P^m$  is a map represented by a linear transformation of homogeneous coordinates. Projective transformations can be represented by matrices of dimensions  $(m+1) \times (n+1)$ .

If a given 3-by-3 non-singular matrix ( $A$ ) representing a projective transformation of  $P^2$  then  $A^*$  is the corresponding line map. This can be better understood if for example:  $u_1$  and  $u_2$  line on a line defined by  $\lambda$  then  $Au_1$  and  $Au_2$  line on the line  $A^*\lambda$ , using equation 6.1.16 this can be formulated as  $A^*(u_1 \times u_2) = (Au_1) \times (Au_2)$ .

Camera model notation. The camera is modelled by the pinhole model. In order to be coherent with the notation used in this paper, the pinhole model will be redefined according this paper. Appendices section provides a more detailed description of the pinhole model (see Appendix A: Stereo Imaging). The pinhole models a region in  $R^3$  seen by the camera into the image plane  $R^2$ . Points in world space are therefore denoted by homogeneous 4-elements vectors

$x=(x,y,z,t)^T$  or more commonly as  $(x,y,z,1)^T$  while points in the image plane are represented by 3-elements vectors  $u=(x,v,w)^T$ . The projection from world to image space is a projective transform represented by a 3-by-4 matrix  $P$  of rank 3, known as the camera matrix. This matrix transforms points from 3D space to 2D space according to the equation  $u=Px$ . The camera matrix  $P$  is defined up to an arbitrary scale factor and therefore has exactly 11 independent entries. Several important parameters are modelled: position and orientation of the camera, the principal point and the scale factors in two orthogonal directions (not necessarily parallel to the image plane axes). Assuming that the camera centre is not at the infinity and with Euclidean coordinates  $t_0(t_x, t_y, t_z)^T$ . The camera mapping is undefined at  $t$  in that  $P(t_x, t_y, t_z, 1)^T=0$ , if  $P$  is written as  $P=(M|v)$  then  $Mt+v=0$  and  $v=-Mt$ . The camera matrix can be written in the following form:  $P=(M|-Mt)$  where  $t$  is the camera centre and  $M$  is non-singular.

### • Epipolar Geometry.

Assuming that two images were taken from a common scene and  $u$  a point in the first image (left image). The place of all points in  $P^3$  that map to  $u$  consists of a straight line passing through the centre of the first image. This straight line seen from the second camera maps to a straight line in the second image (right image) known as a epipolar line. Any point  $u'$  in the second image matching point  $u$  must lie on this epipolar line. All the epipolar lines in the second image corresponding to points  $u$  in the first image meet in a point  $p'$  called the epipole.

This epipole  $p'$  is the point where the centre of projection of the first camera would be visible in the second image, the epipole  $p$  is similarly defined for the first image. Therefore a projective mapping exist such that points in the first image are mapped to epipolar lines in the second image. This mapping process is achieved by a 3-by-3 matrix  $F$  called the fundamental matrix according to  $u \Rightarrow Fu$ . If  $u_i \Leftrightarrow u'_i$  are a set of matching points, then the fact that  $u'_i$  lies on the epipolar line  $Fu_i$  means that:

$$u_i'^T Fu_i = 0 \quad (6.1.17)$$

Based on the equation 6.1.17 and a number of point matches, it is possible to determine the matrix  $F$  by solving the resulting system of linear equations. The resulting fundamental matrix  $F$  determines the epipoles in both images and provides the maps between points in one image and epipolar lines in the other image, therefore it encodes the complete geometry and correspondence of epipolar lines.

This fundamental matrix characteristics are summarized by the next proposition:

**Proposition 1.0.** Supposing that  $F$  is the fundamental matrix corresponding to an ordered pair of images  $(J, J')$  and  $p$  and  $p'$  are the epipoles.

1. Matrix  $F^T$  is the fundamental matrix corresponding to the ordered pair of images  $(J, J')$ .
2.  $F$  factors as a product  $F=[p']_x M = M^* [p]_x$  for some non-singular matrix  $M$ .
3. The epipole  $p$  is the unique point such that  $Fp=0$ . Similarly,  $p'$  is the unique point such that  $p'^T F=0$ .

A property of the fundamental matrix is its factorization into a product of non-singular and

skew-symmetric matrices, however, this factorization is not unique.

The next step to be done consist on determining the projective transformation from image  $J$  to image  $J'$  that take epipolar lines to the corresponding epipolar lines. This transformation is said to preserve epipolar lines and is summarized by the following proposition:

**Proposition 1.1.** Let  $F$  be a fundamental matrix and  $p$  and  $p'$  the two epipoles. If  $F$  factors as a product  $F = [p']_x M$  then:

1.  $Mp = p'$ .
2. If  $u$  is a point in the first image, then  $Mu$  lies on the corresponding epipolar line  $Fu$  in the second image.
3. If  $\lambda$  is a line in the first image, passing through the epipole  $p$ , then  $M*\lambda$  is the corresponding epipolar line in the other image.

Conversely, if  $M$  is any matrix satisfying condition 2, or 3, then  $F$  factors as a product  $F = [p']_x M$ .

As mentioned previously, the projective transformation preserve the epipolar lines, this fact is achieved by matrices  $M$  appearing in a factorization of  $F$ . Since the factorization of  $F$  is not unique, there exist a 3-parameter family of such transformation. For more details on the fundamental matrix Zhang(1996) provides a complete review of the techniques used for estimating the fundamental matrix and its uncertainty. The method presented in this paper are the basis in which OpenCV algorithms were implemented and therefore it provides a rationale for the research study.

#### • *Seek Homography $H$ .*

In this step, the paper provides the methodology to find a projective transformation  $H$  of an image mapping an epipole to a point at infinity. If epipolar lines are to be transformed to lines parallel with the  $x$  axis, then the epipole should be mapped to the 2D homogeneous point at infinity  $(1, 0, 0)^T$ . Homography has seven constraints, three are used to perform the mapping to infinity and four degrees of freedom are left to choose  $H$ , if an inappropriate  $H$  is chosen it may result in highly distorted images. To avoid this and obtain a final image close to the original image some restrictions were imposed on the choice of  $H$ .

One condition that allows to find a good result is to insist that the transformation  $H$  should act as a rigid transformation allowing only rigid rotation and translation in the neighbourhood of a selected point  $u_0$  of the image. An appropriate choice for  $u_0$  may be the centre of the image.

Supposing that  $u_0$  is the origin and the epipole  $p = (f, 0, 1)$  lies on the  $x$  axis. Considering the following transformation:

$$G = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/f & 0 & 1 \end{pmatrix} \quad (6.1.18)$$

This transformation (6.1.18) takes the epipole  $(f, 0, 1)^T$  to the point at infinity  $(f, 0, 0)^T$  as required. A point  $(u, v, 1)^T$  is mapped by  $G$  to the point  $(\tilde{u}, \tilde{v}, 1)^T = (u, v, 1 - u/f)^T$ . If

$|u/f| < 1$  then it can be write as follow:  $(\tilde{u}, \tilde{v}, 1)^T = (u, v, 1 - u/f)$  and  $(u, v, 1 - u/f) = (u(1 + u/f + \dots), v(1 + u/f + \dots), 1)^T$ .

The Jacobian is:

$$\frac{\delta(\tilde{u}, \tilde{v})}{\delta(u, v)} = \begin{pmatrix} 1 + 2u/f & 0 \\ v/f & 1 + u/f \end{pmatrix}$$

Plus higher order therms. If  $u=v=0$ , the previous expression is the identity matrix and  $G$  is said to be approximated at the origin by the identity mapping. Thus for an arbitrarily placed point of interest  $u_0$  and epipole  $p$ , the required mapping  $H$ , that performs the rigid transformation in the vicinity of  $u_0$ , is a product denoted by the expression  $H = GRT$ . Where  $T$  is the translation that takes the point  $u_0$  to the origin,  $R$  is a rotation about the origin taking the epipole  $p'$  to a point  $(f, 0, 1)^T$  on the  $x$  axis and  $G$  is the mapping just considered taking  $(f, 0, 1)^T$  to infinity.

- **Search Matching Homography.**

Considering two images  $J$  and  $J'$ , the next step is to resample these two images according to transformations  $H$  to be applied to  $J$  and  $H'$  to be applied to  $J'$  in such a way that an epipolar line in  $J$  is matched with its corresponding epipolar line in  $J'$ . In other words, if  $\lambda$  and  $\lambda'$  are any pair of corresponding epipolar lines in the two images, then  $H^* \lambda = H'^* \lambda'$ .  $H^*$  is the line map corresponding to the point map  $H$ . The pair of transformations that fulfil this conditions are termed as *matched pair* of transformations.

To choose a matched pair of transformations  $H'$  is firstly chosen and then seek a matching transformation  $H$  chosen so as to minimize the sum-of-squares distance formulated in the following equation:

$$\sum_i d(Hu_i, H'u_i')^2 \quad (6.1.19)$$

The next step to be performed is to find a transformation matching  $H'$ . This is done based on the following theorem:

**Theorem 1.0.** Let  $J$  and  $J'$  be images with fundamental matrix  $F = [p']_x M$ , and let  $H'$  be a projective transformation of  $J'$ . A projective transformation  $H$  of  $J$  matches  $H'$  if and only if  $H$  is formulated, for some vector  $a$ , as follows:

$$H = (I + H' p' a^T) H' M \quad (6.1.20)$$

The paper corroborates the theorem with the following demonstration:

If  $u$  is a point in  $J$ , then  $p \times u$  is the epipolar line in the first image, and  $Fu$  is the epipolar line in the second image. Transformations  $H$  and  $H'$  are a matching pair if and only if  $H^*(p \times u) = H'^* Fu$ . Once this relation must be verified for all  $u$  the equivalent can be written as  $H^*[p]_x = H'^* F = H'^*[p']_x M$  or applying equation (6.1.16) the next relation is obtained:

$$[Hp]_x H = [H' p']_x H' M \quad (6.1.21)$$

Recalling that the fundamental matrix  $F$  factorizes into a product of non-singular and skew-symmetric matrices and that this factorization is not unique a new proposition is given as follow:

**Proposition 1.2.** Let the 3-by-3 matrix  $F$  factor in two different ways as  $F = S_1 M_1 = S_2 M_2$  where each  $S_i$  is a non-zero skew-symmetric matrix and each  $M_i$  is non-singular. Then  $S_2 = S_1$ . Furthermore, if  $S_i = [p']_x$  then  $M_2 = (I + p' a^T) M_1$  for some vector  $a$ . Contrarily, if  $M_2 = (I + p' a^T) M_1$  then  $[p']_x M_1 = [p']_x M_2$ .

Equation (6.1.21) is a necessary and sufficient condition for  $H$  and  $H'$  to match. In view of the above proposition, this implies equation (6.1.20) as required.

To prove the opposite, if equation (6.1.20) holds, then:

$$\begin{aligned} Hp &= (I + H' p' a^T) H' Mp \\ &= (I + H' p' a^T) H' p' \\ &= (I + a^T H' p') H' p' \\ &\simeq H' p' \end{aligned}$$

Equation (6.1.20) together with the proposition 1.2 are sufficient for equation (6.1.21) to hold, and therefore  $H$  and  $H'$  are matching transformations.

The transformation  $H'$ , that takes the epipole  $p'$  to a point at infinity  $(1,0,0)^T$ , is the one with particular interest. In this case,  $I + H' p' a^T = I + (1,0,0)^T a^T$  is of the form:

$$A = \begin{pmatrix} a & b & c \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (6.1.22)$$

Which represents an affine transformation. A special case of the theorem previously presented states that:

A transform  $H$  of  $J$  matches  $H'$  if and only if  $H$  is of the form  $H = AH'M$  and  $A$  is an affine transformation of the form (6.1.22).

Once  $H'$  maps the epipole to infinity, this special case may be used to choose the best matching transformation  $H$  to minimize the disparity. The minimizing problem (6.1.19) is to find  $A$  of the form (6.1.22) such that

$$\sum_i d(A \tilde{u}_i, \tilde{u}_i')^2 \quad (6.1.23)$$

Is minimized, where  $\tilde{u}_i' = H' u_i'$  and  $\tilde{u}_i = H' M u_i$ . Once the transformation  $H'$  and  $M$  are known, and assuming that  $\tilde{u}_i$  and  $\tilde{u}_i'$  hold the vectors  $\tilde{u}_i = (\tilde{u}_i, \tilde{v}_i, 1)$  and

$\tilde{u}_i' = (\tilde{u}_i', \tilde{v}_i', 1)$  respectively, these vectors may be computed from the matched points  $u_i' \Leftrightarrow u_i$ . Then the minimization problem of equation (6.1.23) is rewritten in the form:  $\sum_i (a\tilde{u}_i + b\tilde{v}_i + c - \tilde{u}_i')^2 + (\tilde{v}_i - \tilde{v}_i')^2$ , since  $(\tilde{v}_i - \tilde{v}_i')^2$  is constant, the minimization problem is reduced to the form:

$$\sum_i (a\tilde{u}_i + b\tilde{v}_i + c - \tilde{u}_i')^2 \quad (6.1.24)$$

Equation (6.1.24) is a linear least-squares parameter minimization problem solved using linear techniques to find  $a, b$  and  $c$ . Then  $A$  is computed by substitution of  $a, b$  and  $c$  in 6.1.22 and  $H$  is obtained solving equation 6.1.20.

Summarizing,  $H'$  is the transform that sends the epipole  $p'$  to infinity and align the rows of two images. To align the rows the method uses the fact that aligning the rows minimizes the total distance between all matching points between the two images. Thus a good transformation  $H'$  minimizes the total disparity in  $u_i' \Leftrightarrow u_i$  matching points. The two transformations  $H'$  and  $H$  define the stereo rectification.

### Bouguet's Calibrated Stereo Rectification

Bouguet's image rectification method attempts to minimize the amount of change caused by reprojection in each of the two images while maximizing the matching area. To accomplish this goal and minimize the reprojection distortion, the given rotation matrix  $R$ , that rotates the right camera's image plane into left camera's image plane so that both cameras become coplanar aligned, is split in half between both cameras. The two resulting rotation matrices  $r_1$  and  $r_2$  for the left and right cameras, respectively, are then used to rotate each camera half a rotation so their principal rays become parallel to a vector that would result from combining their original principal rays.

At this point the cameras are coplanar aligned but the epipolar lines are not aligned with any image axis. To obtain images row aligned a new rotation matrix  $R_{rect}$ , that will take the left epipole  $e_1$  to infinity and align the epipolar lines horizontally, need to be computed.

Assuming that the principal point  $(c_x, c_y)$  as the left image's origin the direction of  $e_1$  is along the translation vector between the two cameras centres of projection:

$$e_1 = \frac{T}{\|T\|}$$

The next vector  $e_2$ , that need to be orthogonal to vector  $e_1$ , is computed by choosing a direction orthogonal to the principal ray. By using the cross product of  $e_1$  with the direction of the principal ray followed by a normalization the vector as the form:

$$e_2 = \frac{[-T_y \ T_x \ 0]^T}{\sqrt{t_x^2 + T_y^2}}$$

Knowing that the third vector is orthogonal to  $e_1$  and  $e_2$  it is computed using the cross



product:  $e_3 = e_1 \times e_2$  the new rotation matrix  $R_{rect}$  that rotates the left camera about the centre of projection so that the epipolar lines become horizontal and the epipoles at the infinity has the form:

$$R_{rect} = \begin{bmatrix} (e_1)^T \\ (e_2)^T \\ (e_3)^T \end{bmatrix}$$

In order to transform both images to row aligned images the following transformations need to be done:

$$\begin{aligned} R_l &= R_{rect} r_l \\ R_r &= R_{rect} r_r \end{aligned}$$

The projection matrices take a 3D point in homogeneous coordinates to 2D point in homogeneous coordinates with the following relationship  $P(X, Y, Z, 1)^T = (x, y, w)^T$ . Where  $P$  is the projection matrix ( $P_l$  or  $P_r$ ) that are obtained by the following relation:

$$\begin{aligned} P_l &= M_{rect_l} P_l' \\ P_r &= M_{rect_r} P_r' \end{aligned}$$

Where  $M_{rect_l}$  and  $M_{rect_r}$  are the rectified left and right camera matrices. The projection matrices ( $P_l$  or  $P_r$ ) presented in the matrix form are:

$$\begin{aligned} P_l &= \begin{bmatrix} f_{x_l} & \alpha_l & c_{x_l} \\ 0 & f_{y_l} & c_{y_l} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \\ P_r &= \begin{bmatrix} f_{x_r} & \alpha_r & c_{x_r} \\ 0 & f_{y_r} & c_{y_r} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{aligned}$$

This image rectification method from Bouguet here summarized transforms a general stereo configuration into the canonical stereo configuration. New image centres and bounds are then chosen for the rotated and row aligned images so their matching area is maximized.

### 6.1.2.3 Stereo correspondence review.

- *Introduction*

The recover of the lost dimension or depth estimation is widely used in vision systems for 3D object recognition and reconstruction, 3D remote applications and a large number of other applications. Initially those applications were reduced to sparse stereo correspondence or feature-based techniques due the computational resource limitations. The hardware available nowadays allow to overcome this limitations and most of the stereo matching algorithms currently focus on dense correspondence (Szeliski, 2010).

Stereo Correspondence is related to the matches between two images perceived from different viewpoint of an object in the 3D space. Depth information is obtained by triangulation of corresponding image points subjected to epipolar geometry transformations and with known stereoscopic camera parameters.

Stereo correspondence besides being one of the most active topics in computer vision it still remains a big challenge. In this field a large number of algorithms have been proposed and new ones are being introduced, however, the research in evaluating stereo matching methods has still its limitations. An approach to better understand those methods was done by Szeliski and Zabih(1999) where is presented an experimental comparison of several different stereo algorithms and their performance with real data. Once stereo correspondence is one of the most active subjects in computer vision and new matching techniques continue to be introduced a good way to follow the most recent algorithms is to check the Middlebury evaluation site at <http://vision.middlebury.edu/stereo/eval/>, as well a good number of references are provided at the end of the same site.

While initially the stereo correspondence algorithms were commonly classified in sparse and dense methods the late classification divide them in two groups: local methods and global methods.

- Global methods are used in the optimization process to determine disparity and occlusions. They perform some optimization or iterations steps after computing the disparity maps, many work on the basis of energy minimization with the objective to find a solution that minimizes the global cost function. They currently produce the best stereo matching results with accurate and dense disparity measurements, however, this is achieved with the main drawback of high computational cost that makes them unsuited to real-time stereo applications.
- Local methods compute the disparity maps based on local information of the neighbouring pixels.

In the later case, and the one with particular interest for this research, the recent development focused essentially on area-based matching algorithms that consist on measuring the correlation between pixels in both images taking in account a number of pixels in their vicinity defined by a fixed size window.

Due the diversity of research in this computer vision area only very general information was provided in this section. A better understanding on stereo correspondence classification as well the analyse of the most used and practical matching measures is provided by Cyganek and Siebert (2009). Szeliski (2010) also provide a good survey about stereo correspondence, containing references from the seminal to the earliest works on this area

In this section the researcher opted to review one of the stereo matching algorithms already implemented in the OpenCV library more precisely the block matching stereo algorithm similar to the one developed by Konolige(1997). Although the OpenCV algorithm implementation was based on Konolige paper this review was based in a similar paper proposed by Stefano et al. (2002). This later paper provides more recent and better approach as well a comparison with another existing area-based stereo matching algorithm.

- Purpose

The purpose of the study was to present an area-based stereo algorithm suitable to real time applications. The base of the algorithm relies on the uniqueness constraint and on a matching process that allows the rejection of previous matches when more accurate ones are found. It

provides experimental results obtained on stereo pairs as well a comparison with an already implemented fast area-based algorithm Stefano et al. (2002).

- Methods

*Matching Approach.* Assuming a binocular stereo pair in the canonical form or already subjected to rectification, as done in the previous section, with the epipolar lines lying on corresponding image scanlines and assuming that the left image is the reference, that disparity,  $d$ , belongs to the interval  $[0 \dots d_{max}]$  and that the left image is scanned from top to bottom and from left to right during the matching process.

The algorithm, starting from one point of left image,  $L(x - d_{max}, y)$ , searches for the best match by evaluating the similarity function,  $\epsilon$  (that represents the degree of similarity between two small regions of the stereo pair), within the interval  $[R(x - d_{max}, y) \dots R(x, y)]$ . This process is repeated for the successive points along the scanline  $L(x + i - d_{max}, y)$  and repeating the search for the best match within the interval  $[R(x + i - d_{max}, y) \dots R(x + 1, y)]$ , where  $i$  is the iteration for the successive points along the scanline. Figure (6.9) shows each point on the left scanline corresponding to intervals on the right image where are the potential matching points within a certain disparity range.

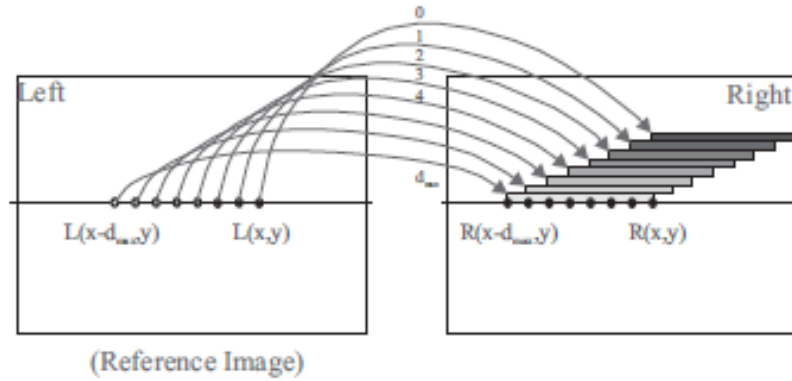


Figure 6.9: Stereo matching process, Stefano et al. (2002).

Assuming now that the best match found  $L(x + \beta - d_{max}, y)$  is  $R(x, y)$  with degree of similarity  $\epsilon(x + \beta - d_{max}, x, y)$  the notation  $L(x + \beta - d_{max}, y) \rightarrow R(x, y)$  is used to indicate the match from left to right has been established.

Area-based algorithms use photometric properties as principal criteria to perform the matching process, however, this criteria can be ambiguous due to different causes such as noise, lens distortion and occlusions. Besides this ambiguity can lead to wrong matches and result in inconsistencies within the matches already established that can be used to detect and discard them, i.e. reject previous matches.

Assuming that another point in the left image,  $L(x + \alpha - d_{max}, y)$  with  $\alpha \leq \beta$ , has been identified also as a possible good match for  $R(x, y)$  with error similarity  $\epsilon(x + \alpha - d_{max}, x, y)$ , two matches for the same point are available violating the uniqueness constraint. This fact can be conveniently used to detect wrong matches by assuming that at least one of the matches  $L(x + \beta - d_{max}, y) \rightarrow R(x, y)$ ,  $L(x + \alpha - d_{max}, y) \rightarrow R(x, y)$  is wrong and can be discarded to give place to match with better score. Therefore if a point being analysed,  $L(x + \beta - d_{max}, y)$ , has better score than another already matched i.e.  $\epsilon(x + \beta - d_{max}, x, y) \leq \epsilon(x + \alpha - d_{max}, x, y)$  the algorithm reject the previous match  $L(x + \alpha - d_{max}, y) \rightarrow R(x, y)$  and accept the new one

$L(x + \beta - d_{max}, y) \rightarrow R(x, y)$  correcting this way ambiguous matching errors.

Figure 6.10 shows how the algorithm can correct from previous misleading matches when new and better matches are found during the search. It presents the scores between the point  $R(x, y)$  of the right image and the points in the left image  $[L(x - d_{max}, y) \dots L(x, y)]$ , that are allowed to establish correspondence with  $R(x, y)$ , as a function of the disparity  $d \in [0, d_{max}]$ .

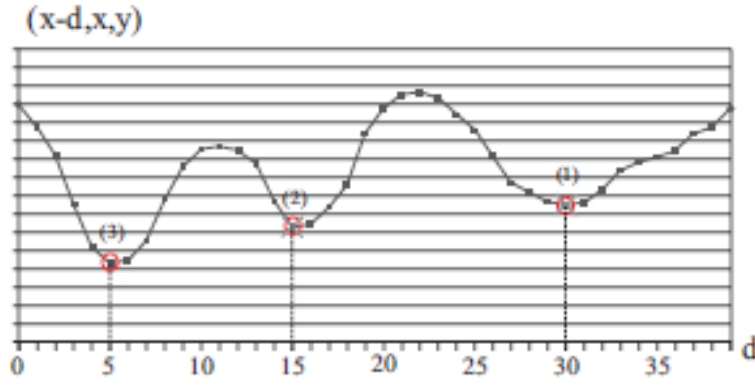


Figure 6.10: Scores associated with point  $R(x, y)$ , Stefano et al. (2002)

Recalling Figure 6.9 the arcs with smaller disparity values represent the similarity scores computed recently while higher disparity values represent the ones firstly computed. According to Figure 6.10 and assuming again two matches as follow:

$$\begin{aligned} \text{match}(1): L(x + \alpha - d_{max}, y) &\rightarrow R(x, y) \\ \text{match}(2): L(x + \beta - d_{max}, y) &\rightarrow R(x, y) \end{aligned}$$

The algorithm will discard the old match (1) and take the new match (2) since it has a better score. If another match ambiguity is found when analysing successive points in the left image  $\text{match}(3) L(x + \gamma - d_{max}, y) \rightarrow R(x, y)$ . Since match (3) has a better score than match (2),  $\varepsilon(x + \gamma - d_{max}, x, y) \leq \varepsilon(x + \beta - d_{max}, x, y)$ , match (2) is discarded and match(3) is set to current match.

#### Computational Optimization.

This subsection plays an important role in the whole algorithm once the computation of the sum of absolute differences (SAD) scores is the most expensive task in the direct matching process. It presents a summary of the optimization techniques to avoid redundant calculations and in order to obtain faster speeds an additional level of incremental calculations is also proposed.

Assuming that  $SAD(x, y, d)$  is the SAD score between a window of size  $(2n+1) \cdot (2n+1)$  centred at coordinates  $(x, y)$  in the left image and corresponding window centred at  $(x + d, y)$  in the right image:

$$SAD(x, y, d) = \sum_{i, j=-n}^n |L(x + j, y + i) - R(x + d + j, y + i)| \quad (6.1.25)$$

Using equation (6.1.25) the  $SAD(x, y+1, d)$  score can be obtained, as follows:

$$SAD(x, y+1, d) = SAD(x, y, d) + U(x, y+1, d) \quad (6.1.26)$$

Where  $U(x, y+1, d)$  is the difference between the SAD associated with the lowermost and uppermost rows of the matching window as shown in light grey points of Figure 6.11.

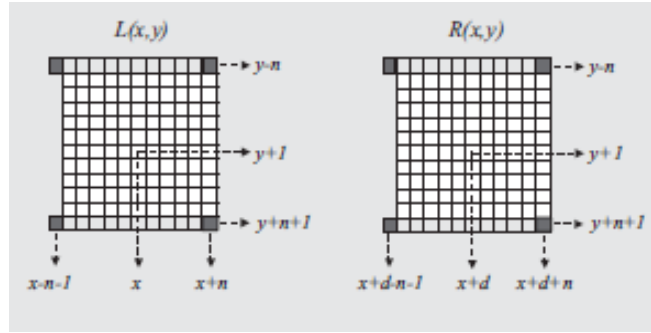


Figure 6.11: SAD matching window, Stefano et al. (2002)

The difference is formulated by the next equation:

$$U(x, y+1, d) = - \sum_{j=-n}^n |L(x+j, y-n) - R(x+d+j, y-n)| \\ + \sum_{j=-n}^n |L(x+j, y+n+1) - R(x+d+j, y+n+1)| \quad (6.1.27)$$

Moreover, to keep a low level of complexity and independent of the matching window size,  $U(x, y+1, d)$  can be computed from  $U(x-1, y+1, d)$  by only considering the attributes associated with the four points shown in dark grey in Figure 6.11. Thus only four operations are required to compute the SAD score at each new point. The contributions of those four points are formulated by the following equation:

$$U(x, y+1, d) = U(x-1, y+1, d) \\ + (|L(x+n, y+n+1) - R(x+d+n, y+n+1)| \\ - |L(x+n, y-n) - R(x+d+n, y-n)|) \\ - (|L(x-n-1, y+n+1) - R(x+d-n-1, y+n+1)| \\ - |L(x-n-1, y-n) - R(x+d-n-1, y-n)|) \quad (6.1.28)$$

Equations (6.1.26) and (6.1.28) use vertical recursion to obtain the SAD score and horizontal recursion to obtain the updating term,  $U$  and therefore it is necessary to store the SAD scores associated with the previous row as well the difference values,  $U$ , associated with the previous point.

An article of interest, concerning to the matching window size, is presented by Kanade & Okutomi (1991) where is proposed a stereo matching algorithm that studies the selection of the window size adaptively by evaluating the local variations of intensity and disparity and compute both disparity estimate and the uncertainty of the estimate.

Pre-processing.

The pre-processing step requires the computation of the mean and variance of both images. Choosing the left image to work with formulation and defining  $N^2 = (2n+1) \cdot (2n+1)$  the mean and variance are formulated, respectively, has follows:

$$\mu(x, y) = \frac{1}{N^2} \sum_{i, j=-n}^n L(x+j, y+i) = \frac{1}{N^2} S_1(x, y) \quad (6.1.29)$$

$$\begin{aligned} \sigma^2(x, y) &= \frac{1}{N^2} \sum_{i, j=-n}^n L^2(x+j, y+i) - \mu^2(x, y) \\ &= \frac{1}{N^2} S_2(x, y) - \mu^2(x, y) \end{aligned} \quad (6.1.30)$$

Equations (6.1.29) and (6.1.30) are obtained by scanning the image and summing all intensities. The paper presents the following set of equations to obtain this two values.

$$S_1(x, y+1) = S_1(x, y) + U_{S_1}(x, y+1) \quad (6.1.31)$$

$$U_{S_1}(x, y+1) = \sum_{j=-n}^n (L(x+j, y+n+1) - L(x+j, y-n)) \quad (6.1.32)$$

$$\begin{aligned} U_{S_1}(x, y+1) &= U_{S_1}(x-1, y+1) \\ &\quad + (L(x+n, y+n+1) - L(x+n, y-n)) \\ &\quad - (L(x-n-1, y+n+1) - L(x-n-1, y-n)) \end{aligned} \quad (6.1.33)$$

$$S_2(x, y+1) = S_2(x, y) + U_{S_2}(x, y+1) \quad (6.1.34)$$

$$U_{S_2}(x, y+1) = \sum_{j=-n}^n (L^2(x+j, y+n+1) - L^2(x+j, y-n)) \quad (6.1.35)$$

$$\begin{aligned} U_{S_2}(x, y+1) &= U_{S_2}(x-1, y+1) \\ &\quad + (L^2(x+n, y+n+1) - L^2(x+n, y-n)) \\ &\quad - (L^2(x-n-1, y+n+1) - L^2(x-n-1, y-n)) \end{aligned} \quad (6.1.36)$$

In order to achieve additional speed-up in both matching and pre-processing procedures a third level of incremental computation is introduced. Before presenting the formulation some observations need to be done.

As the matching step the pre-processing step make use of the four pixels at the corners of the correlation window (see Figure 6.11), this pixels contributes with two terms denoted by  $A$  and  $B$ , where  $A$  contains the two pixels on the left side and  $B$  the two pixels on the right side

of the correlation window. Denoting the array of  $B$  terms by  $T$ , each element can be referenced with the index  $\tilde{x} = x \bmod (2n+1)$ . All elements of  $T$  are visited for each time the correlation window is shifted by  $2n+1$  units. When the window is shifted by one pixel a new  $B$  is computed and  $A$  term is obtained from  $T(\tilde{x})$ . After  $A$  and  $B$  have been used the array is updated with the newest  $B$  term.

In order to implement the third level of incremental computation the mean equation(6.1.32) and variance equation(6.1.36) are respectively rewritten as follows:

$$U_{s_1}(x, y+1) = U_{s_1}(x-1, y+1) + (L(x+n, y+n+1) - L(x+n, y-n)) - T_1(\tilde{x}) \quad (6.1.37)$$

$$U_{s_2}(x, y+1) = U_{s_2}(x-1, y+1) + (L^2(x+n, y+n+1) - L^2(x+n, y-n)) - T_2(\tilde{x}) \quad (6.1.38)$$

Where :

$$T_1(\tilde{x}) = L(x-n-1, y+n+1) - L(x-n-1, y-n) \quad (6.1.39)$$

$$T_2(\tilde{x}) = L^2(x-n-1, y+n+1) - L^2(x-n-1, y-n) \quad (6.1.40)$$

With  $\tilde{x} = x \bmod (2n+1)$  for both equations.

Recalling equation(6.1.28) for the matching step, the third level of incremental computation is applied for each disparity  $d \in [0, d_{max}]$  and  $T$  array grows by one dimension. Thus equation(6.1.28) is rewritten as follows:

$$U(x, y+1, d) = U(x-1, y+1, d) + (|L(x+n, y+n+1) - R(x+d+n, y+n+1)| - |L(x+n, y-n) - R(x+d+n, y-n)|) - T(\tilde{x}, d) \quad (6.1.41)$$

$$T(\tilde{x}, d) = |L(x-n-1, y+n+1) - R(x+d-n-1, y+n+1)| - |L(x-n-1, y-n) - R(x+d-n-1, y-n)| \quad (6.1.42)$$

With  $\tilde{x} = x \bmod (2n+1)$ ,  $d \in [0, d_{max}]$ .

Therefore the aim to achieve further speed-up in the matching and pre-processing step is obtained by means of implementing this formulation.

- Variables and Data Analysis

The study presented in this paper was conducted using two algorithms, the proposed algorithm (P.A.) and an existing bidirectional matching algorithm (SVS) proposed by Konolige(1997). There were two independent variables, the image size and disparity range and one dependent variable that measures the speed of each area-based stereo matching algorithm.

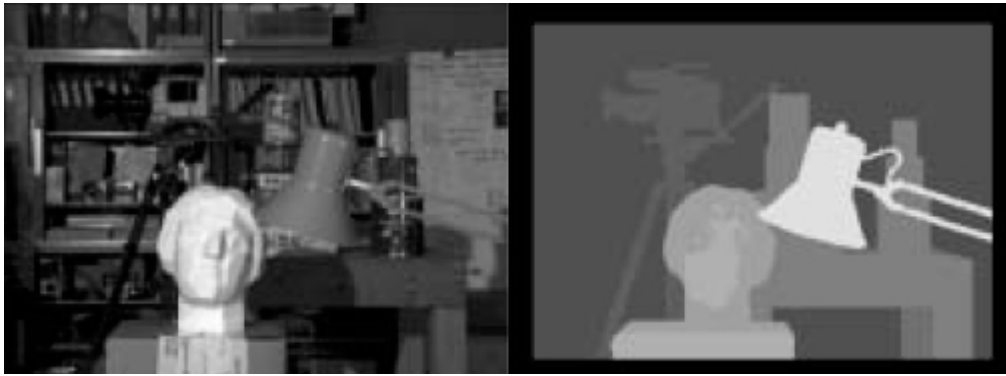


Figure 6.12: Tsukuba image (left) and ground truth (right), Stefano et al. (2002).

To compare each algorithm a stereo-pair of images from University of Tsukuba (Figure 6.12) were used. The measurements were obtained on an Intel Pentium III processor running at 800 MHz.

The analysis was performed by comparing the output of the proposed algorithm (PA) (see Figure 6.13) with the given ground truth image, additionally a table with measurements aimed at assessing the speed of the two algorithms using different image sizes and disparity ranges were provided (Figure 6.14).

- Results

The results returned from this study are mainly two: the disparity images computed from both algorithms (see Figure 6.13) and a table (see Figure 6.13) that shows the behaviour, in terms of speed, of both algorithms when varying the image size and disparity range.

Comparing the output returned by the PA with the Tsukuba ground truth image (Figure 6.12, left and right) the overall 3D structure was recovered even for the regions closer to the stereo rig such the lamp and the statue head and the algorithm was able to deal with the occlusions identified by red points.

However, due the *border-localisation* problem, i.e. when the correlation window is within an area with different depths, the matching process is affected by the uncertainty in the localization of the borders and the algorithm fails to perfectly localize objects as the case of the supporting arm of the lamp that disappeared. Moreover this problem prevents the algorithm from fitting the object's silhouette accurately as can be stated by simply observing both outputs in Figure 6.13.

The results obtained by the SVS 2.0 software on Tsukuba image pair are almost similar to the PA. The 3D structure was recovered correctly and the occlusions detected, however it presents a better performance identifying the silhouettes and the occlusions on the objects further from the stereo rig.



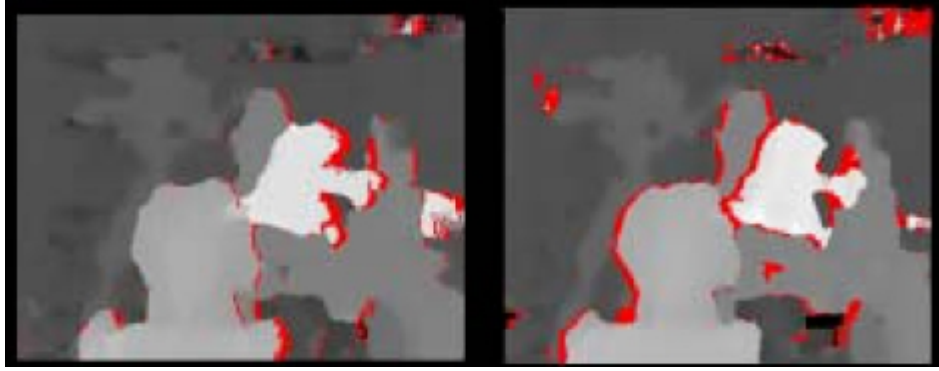


Figure 6.13: Disparity maps computed with the P.A (left) and with SVS 2.0 software (right), Stefano et al. (2002).

Another important output from this study is the speed comparison between the stereo matching algorithm proposed and the SVS 2.0 area-based algorithm. Figure 6.14 report the speed of both algorithms. The SVS algorithms has a better performance for smaller images and smaller disparity ranges however as the image size and disparity range increases the proposed algorithm is faster than SVS algorithm.

- **Conclusions/Implications**

The proposed algorithm, which relies only on a left-to-right matching process, presents a methodology to detect matching ambiguities via “colliding matches” i.e. matches that violate the uniqueness constraint discarding the ones with smaller similarity.

It provides a comparison with the well-known area-based algorithm SVS 2.0 based on bidirectional matching. In most cases the proposed algorithm obtain similar results to the ones obtained by SVS 2.0 with the exception of the errors caused by the border-localisation problems that are more evident in the PA.

The reported measurements (see Figure 6.14) shows that the proposed methodology, strongly based on different levels of incremental calculations, results in a stereo matching algorithm typically faster than SVS 2.0 for big images and large disparity ranges.

Algorithm (size)	$d = 16$	$d = 32$	$d = 48$	$d = 64$	$d = 80$
P.A. (320 × 240)	39.59 fps	31.25 fps	27.44 fps	25.94 fps	25.96 fps
SVS (320 × 240)	57.99 fps	33.68 fps	20.49 fps	15.31 fps	12.71 fps
P.A. (640 × 480)	8.94 fps	6.92 fps	5.77 fps	5.17 fps	4.78 fps
SVS (640 × 480)	11.99 fps	5.93 fps	4.07 fps	3.18 fps	2.54 fps
P.A. (800 × 600)	5.56 fps	4.28 fps	3.60 fps	3.18 fps	2.89 fps
SVS (800 × 600)	6.96 fps	3.65 fps	2.53 fps	1.94 fps	1.51 fps
P.A. (1024 × 768)	3.32 fps	2.56 fps	2.09 fps	1.86 fps	1.67 fps
SVS (1024 × 768)	3.79 fps	2.07 fps	1.45 fps	1.06 fps	0.78 fps

Figure 6.14: Speed (fps) for P.A. and for the SVS 2.0 algorithm, Stefano et al. (2002).

- **Weakness/Limitations**

There were several limitations and weakness in this study. Among the weakness was the fact the paper did not study the influence of the correlation window size in the border-localisation problem and its final result on improving or deteriorating the object silhouettes on the original image

reconstruction. Another important limitations in this study is related to the fact it does not provides a direct comparison between the solely unidirectional algorithm and the bidirectional algorithm. Would be interesting to know the behaviour of those algorithms in the same circumstances, i.e. the comparison of the proposed algorithm without incremental computation schemes with the SVS 2.0 algorithm. The paper also did not provide measurements to justify the incremental computation scheme, i.e. the stereo matching speed with and without incremental computation and in the later case quantify its influence on the algorithm performance.

### 6.1.3 Summary

To ensure that good calibration results are obtained for stereo cameras and the stereo configuration relations are correctly estimated it is necessary to have in consideration a group of factors that are not directly related with the calibration process by itself. The literature review related to camera calibration gives a flexible and reliable technique that do not requires very expensive or elaborated calibration object for camera calibration, however, the article seems to overlook the importance of other factors such as camera capture settings, lightning conditions, focusing, it also do not provide a comparison between results obtained with good set of calibration views and with sets in which the position and orientation variations are minimal (poor set of calibration views). Also the research study reviewed did not studied the cases where the calibration views were captured with lightning gradients or with the object being oriented close to perpendicular with the camera's imager which introduces higher reprojection error. Providing a method for excluding those calibration views can improve substantially the resulting calibration parameters.

Another area that has been studied, related to stereo correspondence, shows that the proposed area-based matching algorithms, which relies only on a left-to-right matching process, presents a methodology to detect stereo matching ambiguities by identifying the matches that violate the uniqueness constraint discarding the ones with smaller similarity. It also provided a speed comparison with other area-based algorithm SVS 2.0. Although this study showed to recover 3D structures efficiently with accurate image silhouettes, the study used a fixed window size and did not study the influence of the external stereo configuration variables such as the angle of convergence between cameras' optical axis, the distance between cameras, and the stereo cameras' calibration influence on the final results. Moreover the study do not provide a method that allows to work with single set of points instead of having to deal with all the image points.

More research with different camera recording settings and illumination conditions, different stereo configurations by changing the cameras orientation or the distances between cameras are needed to determine the best settings that outputs the best results. This current study will contribute to the existing research literature by implementing two different calibration methods and providing a approach to optimize the cameras' calibration parameters. Additionally it will study the influence of the stereo configuration relations on the 2D to 3D reprojection results and provide a method to perform the stereo correspondence only for a sparse set of 2D image points instead of working with all the image points. Furthermore this study will make use of the OpenCV dense stereo matching algorithms to implement an interface that allows to study the influence of different matching settings such as the matching window size, minimum and maximum number of disparities, on the final disparity image results. Additionally this study will contribute with a number of C++ classes that together with OpenCV libraries can be easily used for video capturing and recording operations, image space colour conversion, single and stereo cameras' calibration, uncalibrated and calibrated stereo rectification, and the stereo matching processes.

## 7 Chapter Three

### 7.1 Methods

#### 7.1.1 Introduction

Since decades the researchers have been trying to mimic the human behaviour in robots. One of the most, if not the most, difficult tasks in copying human capabilities is related to perceiving 3-D information. Sensing 3D space can be done by different ways, however, the techniques that are most common nowadays are based on CCD/CMOS cameras or laser-based scanners, the first case gained more attention due its potential to deal with dynamic vision analysis. Stereo vision is assisting to great research advances and presently is one of the most active fields in computer vision area. Recovering the depth information through stereo vision requires, in general lines, three main steps: camera calibration, image undistortion and rectification and stereo matching, this last step presents the most challenging task in the 2D to 3D transformation process.

The stereo correspondence performance and accuracy determines how good and at which cost the depth information is recovered from the correlation of both images.

The following research questions were addressed in this study:

1. *Which are the OpenCV main functions involved in the process of: stereo camera calibration, stereo image rectification, stereo matching and points reprojection into 3D space, and Lucas – Kanade Pyramid optical flow method. What are the inputs and outputs arguments of those functions.*
2. *How to compute camera calibration parameters using a planar calibration object known as chessboard and how to relate two cameras in a stereo configuration. How many calibration views are needed to perform the stereo calibration process and which calibration method (with and without initial guess to compute stereo relations) gives better results. How to optimize the stereo calibration process and improve the calibration parameters results.*
3. *Which are the differences between using calibrated and uncalibrated rectification methods and how to implement the image rectification process by means of using OpenCV functions.*
4. *How to parametrize the stereo relation's rotation matrix into Euler angles and quaternions and how to perform the transformation between this two rotation representations.*
5. *How to compute the disparity image and disparity of a sparse set of points given two rectified images captured from a stereo configuration previously calibrated. How to reproject a sparse set of points to the 3D space.*

This experimental case study used OpenCV v.2.1 image processing platform algorithms together with an C++ OOP approach to implement a main program that provided a set of functionalities

capable to deal with video capture and recording from a single or stereo camera's configuration, stereo cameras calibration, stereo rectification and stereo correspondence and recover 3D information from synchronized stereo video sequences.

A study of the OpenCV main functions related with *Motion Analysis*, *Object tracking*, *Camera Calibration* and *3D Reconstruction* was first conducted to provide the necessary knowledge and background to use those functions in this case study main program implementation. Additionally in parallel to this research were done two presentations: the first presentation addressed the next topics: **Camera Model**, **Projective Transform**, **Lens Distortions**, **Planar Homography**, **Camera Calibration** and **Stereo Calibration**. The second presentation, and broader one, addresses the following topics: **Calibration Parameters Optimization**, **Lukas-Kanade 2D Points Correspondence**, and **Rotation Matrix Parametrization**. Both presentation were included in the DVD attached to this thesis.

The second part of this case study consisted on laboratory experiments divided in three sessions. The first two sessions (pretest experiments) were used to get familiar with the Phantom v9.1 hardware, PCC software and collect sets of stereo video sequences for calibration purposes, with different stereo configurations and video settings, with the primary goal of determining the best video settings, illumination conditions and revealing experimental errors. The third session (intervention experiment) was used to record stereo video sequences of a 3D path executed by MELFA RV-2AJ robot with the main goal of providing a comparison method between the 3D points obtained with the main program implementation and the 3D path given by MELFA RV-2AJ.

For all the three experiments the data obtained with Phantom v9.1 cameras were saved to an external hard drive and processed with the main program functions accordingly the topics related to the research questions.

### 7.1.2 Settings

This study took place in AGH University of Science and Technology of Krakow, WIMIR-Faculty of Mechanical Engineering and Robotics, room 18. The resource laboratory is a classroom laboratory containing namely the Mitsubishi MELFA RV-2AJ robot, a desktop computer with MELFA programming language software installed and all the vision systems hardware used for this case study.

### 7.1.3 Intervention and Instructional Materials

The independent variables measured by this study consisted of laboratory experiments where two interventions were performed: (1) stereo video capture for calibration purposes and (2) stereo video capture of a Mitsubishi MELFA RV-2AJ robot end-effector executing a simple 3D path for 3D information recovering purposes.

The stereo video capture for calibration was intended to provide a set of images of a calibration object (independent variable) for camera calibration and stereo relations estimation. Stereo video capture of a Mitsubishi MELFA RV-2AJ robot end-effector movement was intended to capture the 3D end-effector path (independent variable) and targets, with known geometries, arranged in strategic places within the field of view (FOV) for later being used on the process of recovering 3D object points from 2D image points tracked over those video sequences. This two interventions were repeated one after another for each new stereo configurations.

The dependent variables measure by this study for the first type of intervention consisted of left and right camera intrinsic and extrinsic parameters( camera matrix and distortion coefficients), stereo relations( rotation, translation, essential, and fundamental matrices), reprojection error

resulting from calibration process, the remapping maps(undistortion+rectification), and the disparity-to-depth reprojection matrix resulting from rectification process. On the second type of intervention the dependent, and main variable measured by this study, consisted on the sequence of 3D points (or 3D path) recovered from the stereo video sequence, as defined by the research questions.

Two main types of instructional materials were used during the intervention to record stereo video files for calibration and for the 3D information recovering purposes. Instructional materials to program the Mitsubishi MELFA RV-2AJ robot controller using Melfa Basic IV robot programming language was provided by the chairperson of this thesis (Kohut, 2011) (see Appendix B: MELFA Basic IV Presentation). In addition was used the manual provided by Visual Instrumentation Corporation (2011) to determine the right lightning and VideoStrobe – FloodController settings (see Appendix C: VideoStrobe & VideoFlood LEDs).

#### **7.1.4 Measurements Instruments**

This study utilized different measurement instruments and tools that will be described by the order in which the were applied to conduct the research. Two main measurement instrument were used to collect data: The commercial software, and the research-made StereoVisionProg program.

##### **7.1.4.1 Phantom stereo configuration hardware.**

To collect video files during the intervention was used a stereo configuration composed by the following list of material mostly from Vision Research Inc. (2011): Two **Phantom v9.1** high speed cameras that are able to provide 14-bit image depth and 1000 frames per second at full resolution 1632 x 1200 pixel with 6 GB of internal memory. Both cameras had installed the **v-Series Lens Shutter** – used to automatically shade the sensor and calibrate the camera to a black reference (CSR operation), and on each shutter was mounted the **SIGMA 24-70, 1:2.8 EX DG** lens. Additionally were used two **Break-out-Box** that gave access to every available signal on the camera capture cable like the **Trigger, Strobe** and frame synchronization signal **F-Sync**. It was also used a **Manfrotto 454 Sliding Plate** with finger tip control mounted with the right camera that allowed fine positioning adjustments when changing the horizontal distance between the two cameras on the stereo configuration. Each camera was engaged to a **Manfrotto 405 Geared Head** that allowed fine orientation adjustments for all three axis. To assemble the final stereo configuration was used a 4 heads accessory arm **Manfrotto 131DDB**, and two **Manfrotto Variable Friction Arm** to hold the LEDs arrays, mounted on a geared tripod **Manfrotto 475B Pro Geared** to support both cameras and all the stereo configuration hardware. More detailed information related to the cameras and some of its accessories is provided on the appendices section (see Appendix D: Phantom v. 9.1 Data Sheet). All Manfrotto's brand material can be found on the following Web site: <http://www.manfrotto.com/category/0>.

##### **7.1.4.2 VideoFlood LED and videoStrobe – floodController.**

To improve the lightning conditions during the experiment were used two LED arrays **Model 900405 3-by-4 LED Array** and one strobe controller **Model 201090A Hi-g Controller** that allowed to select 7 different durations of 1 to 500 microseconds per flash or two continuous light intensities of 50% and 100% used for a period not longer than 10 minutes. The lightning material was provided by Visual Instrumentation Corporation (2011).

##### **7.1.4.3 Phantom camera control v1.2 software (PCC).**

PCC is a commercial software from Vision Research Inc. (2011) company. This software was

used during the three interventions to control two Phantom v9.1 cameras using Ethernet connections. The main image video sequences settings controlled with PCC were: **Bit Depth**, **Resolution**, **Sample Rate**, **Exposure Time**, current session reference (**CSR**) to obtain more precise compensation of the pixel errors for the current settings, and **Post Trigger** value. During each intervention PCC software was also used to save the captured video files from the cameras RAM to an external hard drive in their original file extension (cine format). PCC was also used after the interventions to save cine files into sequences of BMP images and uncompressed AVI video sequences.

#### **7.1.4.4 Mitsubishi MELFA RV-2AJ.**

The research used **Mitsubishi MELFA RV-2AJ** robot to implement and execute a 3D path that was recorded using different stereo configurations. Using the **MELFA Basic IV**, the Mitsubishi programming language for the robot controller, a simple closed path with three circular interpolation movements was programmed. The resulting trajectory was executed using 10 and 50 millisecond sample time and the resulting file with the time, joints angles and end-effector position and orientations was saved to an external hard drive. Mitsubishi Electric Industrial Robots are manufactured at a factory certified for ISO14001 (standards for environmental management systems) and ISO9001 (standards for quality assurance management systems).

#### **7.1.4.5 StereoVisionProg program researcher-made instrument.**

For this case study the researcher had implemented a main program named **StereoVisionProg** that was used to read lists of images and video files obtained for calibration and depth recovering proposes. StereoVisionProg was used to study the optimal number of calibration views, compute the calibration parameter and stereo relations, compute the stereo rectification maps, compute disparity image using all the left and right image points (dense matching) or a set of 2D left and right image points (sparse matching), and then compute 3D points. The program contains all the functionalities and measurement instruments used to answer the research question. All the output results obtained were saved to XML files in the current directory from where the program was executed.

StereoVisionProg program consisted on the implementation of a number of classes that used the optimized algorithms from **OpenCV v. 2.1**. **Microsoft Visual Studio 2008 (C++ 9.0)** integrated development environment (IDE) was used to develop, debug, compile and link those classes with the OpenCV v.2.1 libraries.

#### **7.1.4.6 GML Camera Calibration Toolbox v. 0.4**

The researcher utilized GML Camera Calibration Toolbox (Vezhnevets & Velizhev, 2005) to compare the cameras intrinsic and extrinsic parameters results obtained individually for both left and right cameras with the results obtained with the researcher-made StereoVisionProg program. GML program was used to load a set of images for calibration, input the chessboard properties (number of corners and square size) with **Set Object Size** and **Set Square Size** options, detect all corners with **Detection Method** setted to **Squares Method** and then was used **Native OpenCV** on the **Calibration Type** option to compute: **Camera Matrix**, **Distortion Coefficients**, and **Pixel Error**. All the results were saved into files in XML format.

#### **7.1.4.7 MATLAB v. 7.7.0.**

This case study utilized MATLAB technical computing language software to treat the output results obtained with StereoVisionProg program functionalities. The XML files generated by

StereoVisionProg software with the output results were loaded into MATLAB workspace using a number of researcher-made Function M-files used to read, treat and plot those results, namely: results obtained for the optimal number of calibration views study and the results obtained for the experimental 3D path. In addition one function was implemented to read and plot the real 3D path given by MELFA Basic IV software, providing a method that allowed to compare the MELFA 3D path with the path obtained using the research procedure. All the M-files programming code were provided on the appendices section (see Appendix E: MatLab M-Files Code).

#### **7.1.4.8 Other material.**

To perform all the operation and answer the questions proposed by this study were used initially two web cams A4Tech Evolution PK-710MJ series to test and debug StereoVisionProg during its implementation. For all the input/output data computation was used a laptop ASUS A3500E, Intel Pentium M processor 1.73 GHz, with 1.50 GB of RAM.

### **7.1.5 Procedures**

The research study procedure used to conduct the study was divided in two phases: the first phase consisted in presenting a detailed description of OpenCV v.2.1 main algorithms used in this research, and two laboratory experiments used for qualitative analysis, the second phase consisted on a third laboratory experiment session where data were collected for quantitative analysis.

#### **7.1.5.1 Baseline**

For the sake of organization and brevity of this chapter the explanation of the OpenCV v.2.1 main functions, and as well the theory under them, used on this research was presented in the appendices (see Appendix A: Stereo Imaging and Appendix F: Motion). This two appendices were to answer the first research question.

The data were collected through three laboratory experiments. The first and second experiments were done with the main goal of providing the researcher with a self-familiarization with phantom software and hardware involved on the experiment. The data collected with this two experiments were reviewed with the chairperson that pointed experimental errors, changes and improvements to take in account for the third laboratory. The procedures used to conduct the first two laboratories were partially identical to the procedures used on the third laboratory described in next section.

#### **7.1.5.2 Intervention**

The intervention was done in different steps: **Phantom Stereo Configuration Assembling, Camera Settings Configuration with PCC Software and MELFA 3D Path Programming, and Data Collection.**

##### **1. Phantom Stereo Configuration Assembling**

To assemble the stereo configuration was used a geared tripod with an arm mounted on its top, before being attached other elements the tripod was levelled using the tripod's build-in bubble level. A sliding plate was fastened to the base of one geared head and then attached to the horizontal arm's right side, the second geared head was directly mounted to the horizontal arm's left side (without sliding plate). Then to each Phantom v 9.1 camera was added a SIGMA 24-70 mm lens and on their base was fastened a quick release plate that allowed to fix them on the geared heads. To both cameras were then connected the Break-out-Box and the Gigabit Ethernet cables, then the synchronization cable was connected to both break-out-Box BNC connectors to allow frame synchronize between both cameras. A cable was also connected to both trigger connectors to trigger the video recording on both



cameras with an external hard trigger. To complete the stereo configuration both Gigabit Ethernet cables were connected to a laptop using the Ethernet entrance and a Gigabit PCMCIA network adapter. The next figure (see Figure 7.1) shows both Phantom v.91 cameras arranged on a stereo configuration.



Figure 7.1: Phantom v.91 cameras arranged on a stereo configuration.

## 2. Camera Settings Configuration with PCC Software and MELFA 3D Path Programming.

To establish the connection between the Phantom v 9.1 cameras and the PCC software the Phantom control unit IP addresses were first defined by setting the to network connections as follows: **TCP/IP** → **Properties** → and then the field **Use the following IP address** was checked and the first camera's network **IP address** was set to **100.100.100.1**, and the second camera's network **IP address** to **100.100.100.2**, both networks **Sub net mask** field were set to **255.255.0.0** and the option **Use the following DNS server addresses** was selected a left empty and then pressed **OK** to conclude the configuration.

After both cameras were successfully connected the video capture settings were defined. On the **Camera Settings** the image **Bit Depth** was set two 8 bits and the number of RAM partitions to 1. To capture the robot executing a 3D path were used the following settings: image **Resolution** 960 x 720 pixel, **Sample Rate** 700 pps and **Exposure Time** 900  $\mu$ s as shown in the next Figure 7.2.

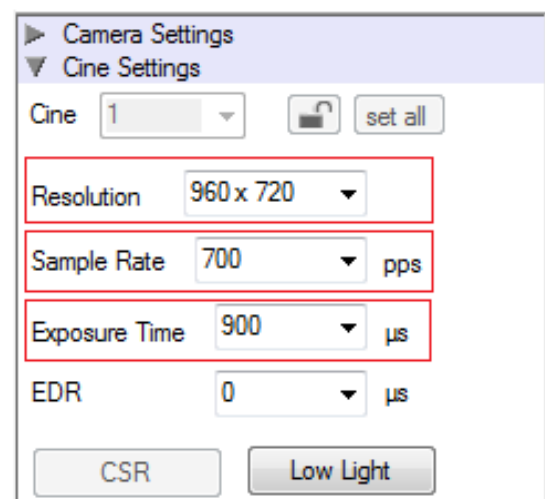


Figure 7.2: PCC1.2 software - cine settings.

To obtain synchronized video sequences, i.e. capture exactly the same object scene at the same time for the left and right camera, the right camera (**ID: 9644**) was defined as **Internal** (master clock source for two serial connected Phantom cameras) and the left camera (**ID**



9645) was defined as **External**. Few consideration were taken in account :

- Frame Delay (External) > Frame Delay (Internal) at least  $1\mu s$ .
- Exposure Time (External)  $\leq$  Exposure Time (Internal).
- Post Trigger Value (Internal) > Post Trigger Value (External) at least one frame.
- Use the external trigger (instead of the software-trigger) to guarantee that the cameras remained synchronized.

To capture video sequences for stereo calibration the sample rate were changed to 90 fps. This allowed to have more time to change the calibration object position and orientation and obtain better calibration results.

After the **Camera Settings**, **Cine Settings** and **Advanced Settings** were configured a simple closed 3D path was programmed with MELFA Basic IV software using circular interpolation movements command. To the robot end-effector was attached a printed target (**Target 2**, Figure 7.6) with known geometries to provide good features to track by the StereoVisionProg functionalities, also close to the robot's workspace were added static targets (**Target 1**, **Target 3**, and **Target 4**, Figure 7.6) to compare its geometries with the results obtained. The robot movement was programmed in such a way that the end-effector was kept visible for both cameras during all the robot's movement. MELFA robot's program and the targets dimensioning here mentioned were added in the appendices section (see Appendix G: Targets and MELFA Basic IV Program).

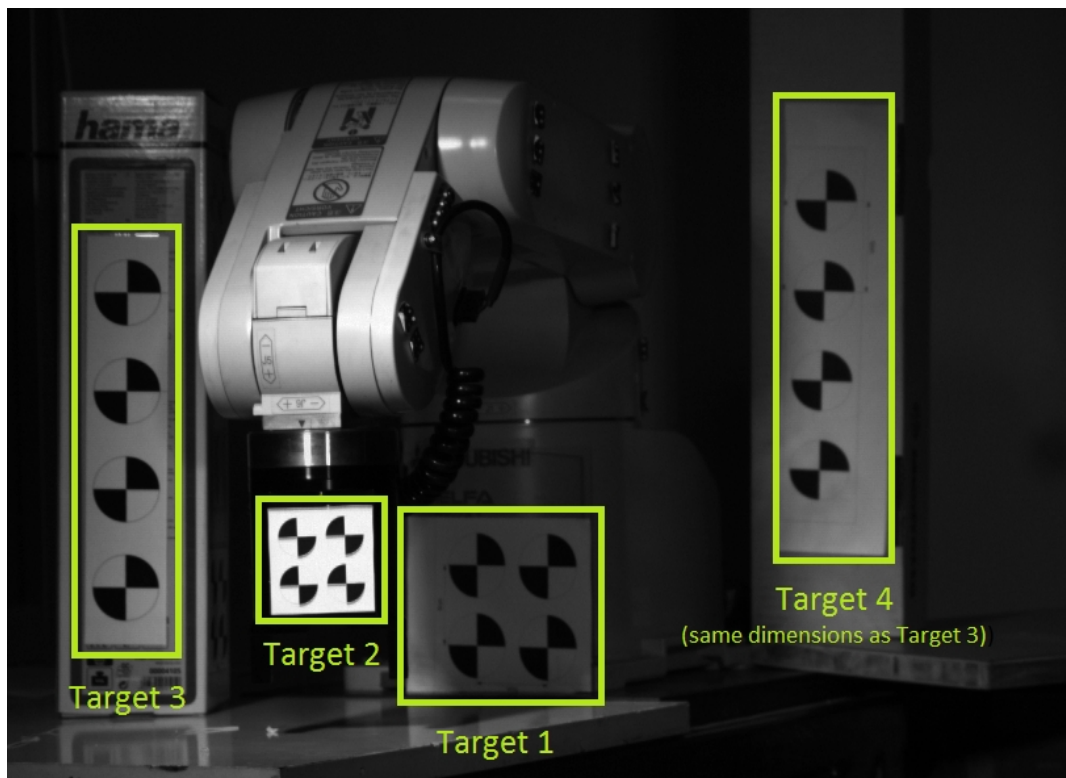


Figure 7.3: Targets used for points tracking purposes.

### 3. Data Collection

After concluding the procedures 1 and 2, two 3-by-4 LED Arrays, one attached to the stereo configuration horizontal arm and other attached to an external tripod, were connected to the

video strobe controller to provide good elimination conditions. The illumination formula from Visual Instrumentation Corporation was later used to verify if the illumination conditions were within the acceptable intervals ( see Appendix C: VideoStrobe & VideoFlood LEDs).

To capture video sequences for stereo calibration and 3D information recovering purposes four different stereo configuration were used. The procedure followed was the same for all the configurations, and is described as follows:

- Position the stereo configurations.
- Change horizontal distance between cameras.
- Change angles amplitude between both camera's optical axis (convergence angles).
- Run MELFA program and ensure that all robot movement is within both camera's FOV.
- Focus each camera lens assuming as focal-plane the initial end-effector position such that the target attached to it appeared acceptability sharp.
- Block the lenses with glue-tape to avoid unintentional defocusing.
- Apply CSR option to calibrate the image for the current cine settings parameters (process similar to black reference calibration adjustments ).
- Record one video sequence with the robot executing a path previously programmed.
- Record one video sequence of one person moving a calibration object covering as much as possible both camera's FOV.
- Save cine files to an external hard drive and perform the same procedure each time the stereo configuration is changed.

Due the fact that for some stereo configuration was necessary to use smaller calibration object to make easier the task of capturing the calibration object on both cameras' FOV at the same time four different calibration object were used. The list of those patterns and their properties can be seen on Table 7.1 as follows.

*Table 7.1*

*OpenCV Calibration Object's Characteristics*

Calibration Object		
Calibration Pattern	Square Size[mm]	Number of Corners [ nx x ny ]
A	14	[11 x 13]
B	30	[09 x 09]
C	40	[08 x 09]
D	20	[08 x 09]

Note. Number of Corners is equal to the number of squares along XX direction minus one (nx) – by - the number of corners along YY direction minus one (ny).

For each stereo configuration all the variables were noted and the cine files saved with convenient names. Table 7.2 lists the cine files and measurements obtained for each configuration during the third laboratory experiment.

*Table 7.2*

*Laboratory 03 – Stereo Configuration's Variables and Video Files*

Video Files (.cine)		Stereo Configuration Variables		
With Robot Movement	With Calibration Object	Distance Between Cameras $\pm 1$ [mm]	Distance From Target $1 \pm 1$ [mm]	Calibration Object
StereoS01L StereoS01R	StereoCalibS01L StereoCalibS01R	390 <sup>(*)</sup>	1930	C, A <sup>(#1)</sup>
StereoS02L StereoS02R	StereoCalibS02L StereoCalibS02R	390 + 90 <sup>(*)</sup>	1930	B, C
StereoS03L StereoS03R	StereoCalibS03L StereoCalibS03R	390 + 90 <sup>(*)</sup>	1930	B
StereoS04L StereoS04R	StereoCalibS04L StereoCalibS04R	390 + 115 <sup>(*)</sup>	2080	C, A <sup>(#1)</sup>

Note. All pair of video sequences were saved using “S[xx]” that indicates the stereo configuration set, and “L” and “R” suffix that indicates the video is related to the **L**eft or to the **R**ight stereo configuration's camera.

(\*) - The initial distance measured between cameras was obtained from the centre of each lens using a meter. The following distances were added by using the fine positioning sliding plate.

(#1) - Calibration object A is a special pattern used by Tema Camera Control Software. This pattern proved to be inappropriate to use with StereoVisionProg implementation.

### **7.1.5.3 Posttest.**

Each of the video files information collected during the intervention were then processed by the researcher-made StereoVisionProg program. The post-interventions main procedures used to conduct the research and answer the questions which this study proposed are described in this subsection.

To calibrate the cameras and compute the stereo relations for each stereo configuration to obtain the end-effector 3D path executed with MELFA RV-2AJ the next procedure was followed:

1. Using PCC software each left and right calibration cine file (set S02, S03 and S04) were saved into single image files using “**Windows BMP 8 images**”<sup>1</sup> format and cine range ( 0;

<sup>1</sup> The “OS/2 BMP” image format available from PCC software drop down list is a format supported by IBM OS/2 operative system and uses the same file extension as “Windows BMP” image format. Reading an OS/2 BMP file as

8130) as shown in Figure 7.4.

- Windows BMP 8 images format was used to preserve the same image quality and pixel bit depth with which the cine files were recorded (Camera Settings->Bit Depth = 8 ).
- Each left and right calibration cine file was saved into single images by adding the suffix "+4" to the file name prefix. For example, for the set S02 was done: StereoCalibS02L+4 (StereoCalibS02L0001.bmp, StereoCalibS02L0001.bmp, ..., StereoCalibS02L8131.bmp ).

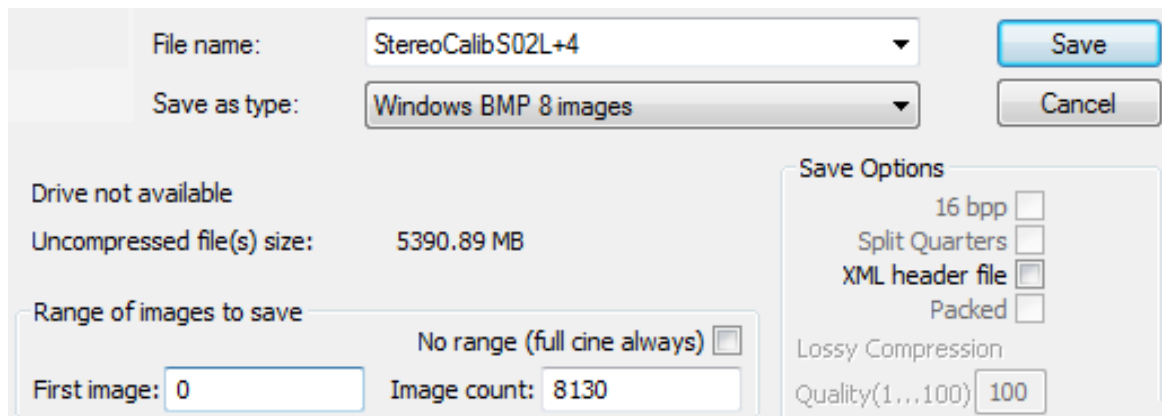


Figure 7.4: PCC Software - save cine settings.

2. To create a list of left and right calibration views was used StereoVisionProg program: **Main menu's Option [2]** sub **Option [3]** as shown in Figure 7.5 and Figure 7.6. Two types of calibration views lists were created, one was used to study the optimal number of calibration views, and a second one was used to calibrate each stereo configuration. The list were created with an image sample (increment between consecutive images) in such a way that approximately only 150 (for the first case) and 100 (for the second case) images from all cine range were used for calibration, avoiding similar calibration views that may cause divergence on calibration parameters and stereo relations estimation.

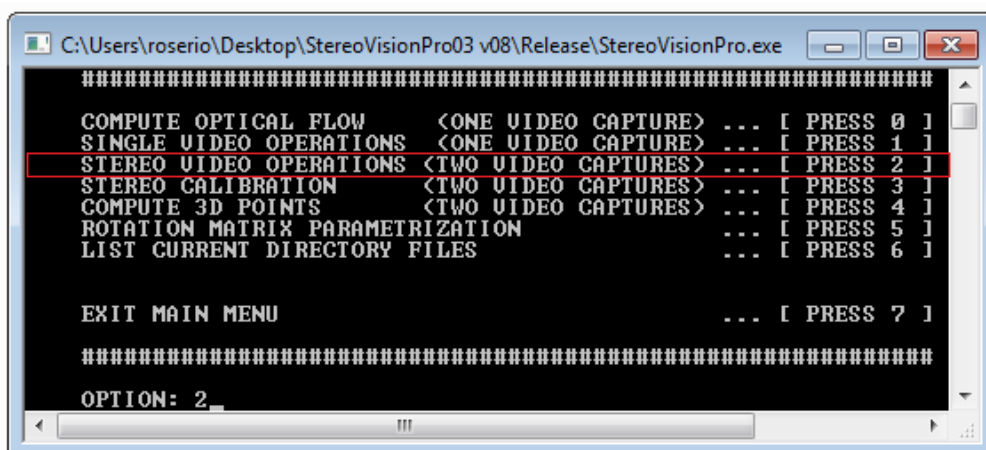


Figure 7.5: StereoVisionProg: Main menu 's Option [ 2 ].

if it was an Windows BMP file can produce unpredictable results therefore this format should not be used to save the sequence of images for calibration.

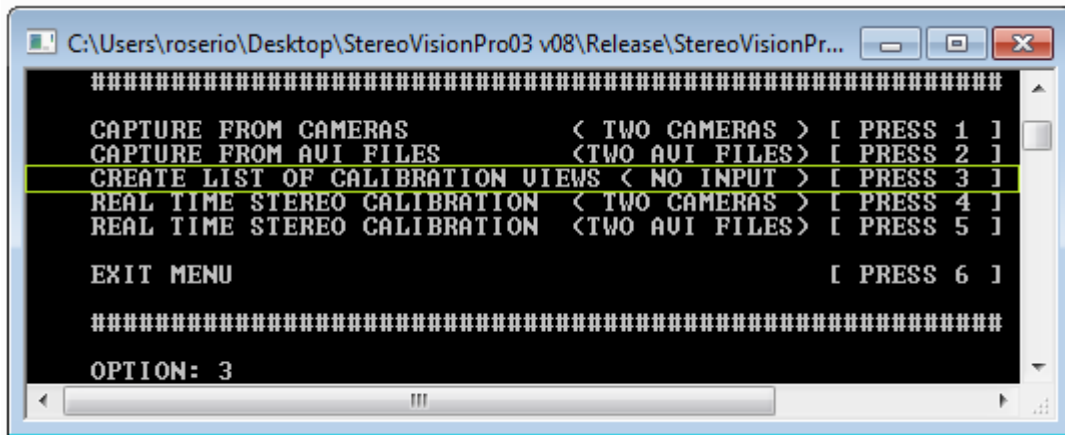


Figure 7.6: StereoVisionProg: Main menu's Option[ 2 ] sub Option [ 3 ].

After navigating through the options, the program will ask the user to input the calibration list arguments: 1-**file name**, 2-**left image sequence prefix**, 3-**right image sequence prefix**, 4-**starting sequence number**, 5-**ending sequence number**, and 6-**increment**. Following point 1 example such parameters were as follows:

- 1) Lab3CalibListS02 (arbitrary)
- 2) StereoCalibS02L
- 3) StereoCalibS02R
- 4) 1
- 5) 8131
- 6) 80

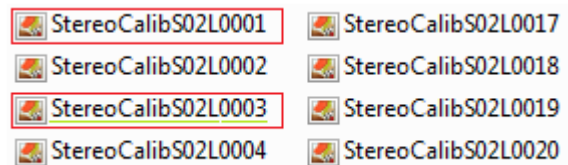


Figure 7.7: Sequence of BMP images for calibration.

The output file is a text file **Lab3CalibListS02.txt** with a list of left and right image names of synchronized calibration views as in Figure 7.7.

3. To study the optimal number of calibration views necessary to calibrate each stereo configuration was used one of the StereoVisionProg functionalities as follows: **Main menu's Option [ 3 ] sub Option [ 1 ]** as shown in Figure 7.8.

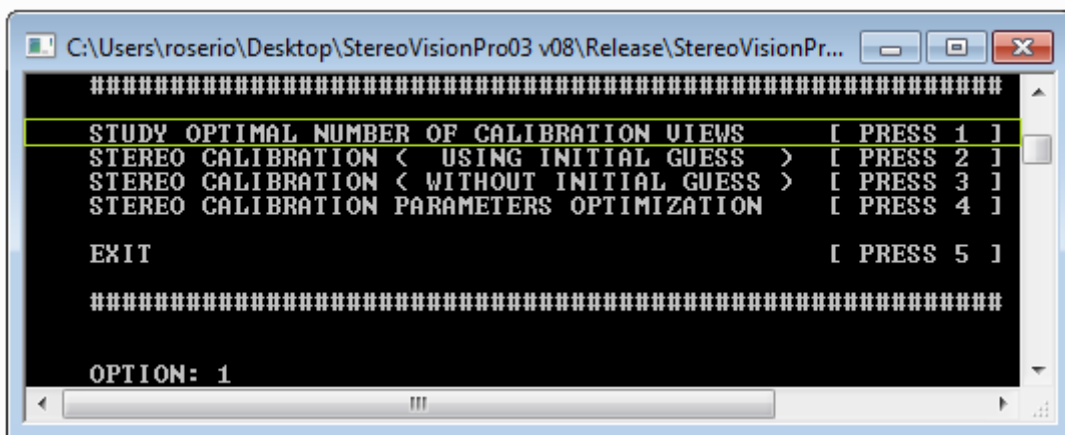


Figure 7.8: StereoVisionProg: Main menu's Option [ 3 ] sub Option [ 1 ].

This functionality required the calibration to be first performed using the text file containing a list of 150 left and right camera calibration views so the image points and object points were then loaded from the calibration process output file.

To study how the calibration parameters evolved with the number of calibration views 17 sets of images points and object points <sup>2</sup> [N02 N05 N10 N20 N30 N40 N50 N60 N70 N80 N90 N100 N110 N120 N130 N140 N150] were used individually to perform the calibration.

Two output files **Lab3CalibListS02\_StudyM1.xml** and **Lab3CalibListS02\_StudyM2.xml** were obtained for calibration method M1 and method M2 respectively. In the output files were stored the **intrinsic** and **extrinsic parameters**, **stereo relations**, and **reprojection errors** that resulted from each calibration sets. The variables were then loaded into MatLab, processed and presented graphically (see Appendix E: MatLab M-Files Code).

4. To compute calibration parameters and stereo relations was used StereoVisionProg : **Main menu's Option [ 3 ] sub Option [ 2 ]** (or **Option [ 3 ]** ) as shown by Figure 7.9. The program read all the text (.txt) files in the current directory and list them, after choosing the right calibration list (in this case **Lab3CalibListS02.txt** ) the program loaded the calibration images, computed the intrinsic calibration parameters (M1, M2, D1, D2) and the stereo relations (R, T, E and F) using two methods: method one (**M1**) estimates the intrinsic parameters of each camera individually with `cv::calibrateCamera( )` and then uses this parameters as input for `cv::stereoCalibrate( )` to estimate the stereo relations (R, T, E, F), method two (**M2**), on its turn, computes the intrinsic parameters and the stereo relations all at the same time, without initial guess. Method **M1** saves the results into an xml file named **Lab3CalibListS02\_CalibrationM1.xml** while on method **M2** the results output file is named **Lab3CalibListS02\_CalibrationM2.xml**.

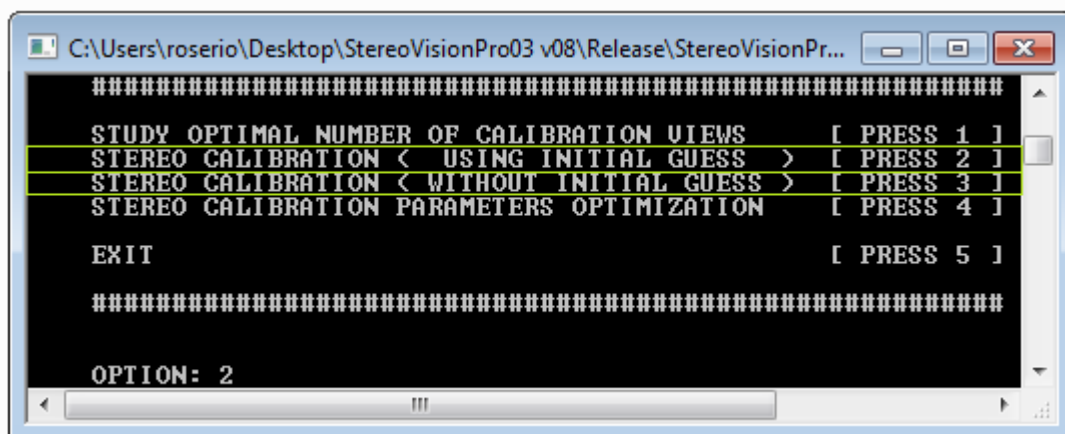


Figure 7.9: StereoVisionProg: Main menu's Option[ 3 ] sub Option[ 2 ] .

5. After computing the calibration parameters was performed the **calibration parameters optimization**. Based on the mean Euclidean distance between the reprojected and projected image points for each view it excludes the views with higher errors, i.e. higher mean Euclidean distances. The implementation of this functionality was formulated as follows:
  - First were computed the projected points for both cameras (Figure 7.19 (b)) using OpenCV function `cv::projectPoints( object points, camera parameters, projected points )` . This function uses the obtained camera intrinsic and extrinsic parameters, “as if” they were estimated correctly, to project the object points into image points.
  - Then the program computed the mean Euclidean distance between the reprojected image points (collected from the calibration views) and projected image points for each left and right calibration view as follows:

<sup>2</sup> Image points and object points were obtained using the calibration views and thus in this case they represented those sets of images.



$$Ed_i = \sqrt{(xi_R - xi_P)^2 + (yi_R - yi_P)^2} \quad (7.1.1)$$

Where  $Ed_i$  is the Euclidean distance between the reprojected (<sub>R</sub>) and the projected (<sub>P</sub>) point  $i$  of a calibration view.

$$MEd = \frac{1}{n} \sum_{i=1}^{n=nCorners} Ed_i \quad (7.1.2)$$

Where  $MEd$  is the mean Euclidean distance for each calibration view and  $nCorners$  is the number of chessboard corners. The resulting mean Euclidean distance for all the left and right calibration views (separately) is given by:

$$M = \frac{1}{n} \sum_{j=1}^{n=nViews} MEd_j \quad (7.1.3)$$

Where  $nViews$  is the number of left and right calibration views. Thus the standard deviation is given by the equation:

$$SD = \sqrt{\frac{1}{n} \sum_{i=1}^{n=nViews} (MEd_i - M)^2} \quad (7.1.4)$$

An interval  $[0; M + SD]$  was then used to filter the views with higher errors, i.e. views which  $MEd > M + SD$  were excluded<sup>3</sup>.

After the views with higher errors contributions being excluded the new calibration parameters were recomputed (using method M1 or M2 depending on which one was the last being used<sup>4</sup>) and stored into a new xml output file named **Lab3CalibListS02\_CalibrationM3.xml**.

The next figure (Figure 7.10) shows an example of image points reprojection and projection.

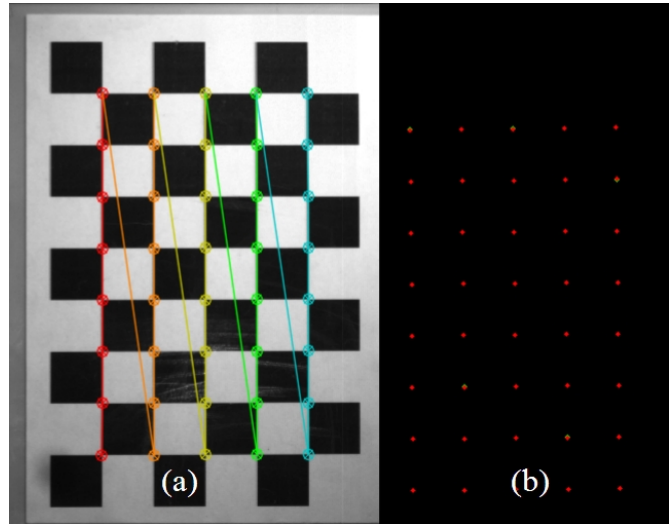


Figure 7.10: Reprojected (a) and projected (b) image points.

- 3 Besides stereo calibration being performed individually for each camera, to estimate the stereo relations with `cv::stereoCalibrate()` is necessary that both left and right image points correspond to exactly the same calibration object captured at the same instant, i.e. synchronized captures, and thus if the left (right) view is excluded its right (left) corresponding view is also excluded.
- 4 The most recent calibration operation (M1, M2 or M3) output file name is always overwritten into the node `<CalibrationParameters>...</CalibrationParameters>` of the main file “**StereoConfigurationOutput.xml**”. All subsequent operations (calibration study, calibration optimization, rectification, correspondence and rotation matrix parametrization) uses only the stereo calibration parameters and relations stored inside this output file.

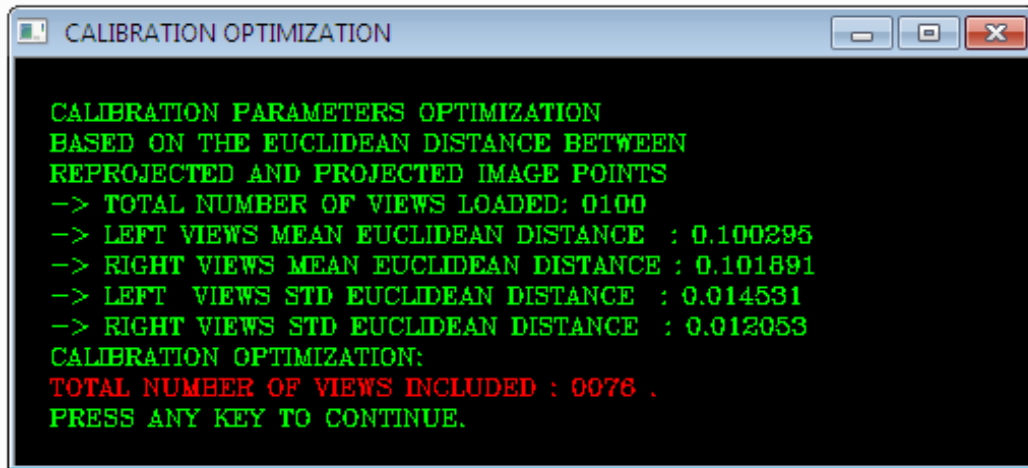


Figure 7.11: Calibration parameters optimization process.

Calibration optimization can be performed by choosing on StereoVisionProg: Main menu's Option [ 3 ] sub Option [ 4 ] as shown in Figure 7.12 .

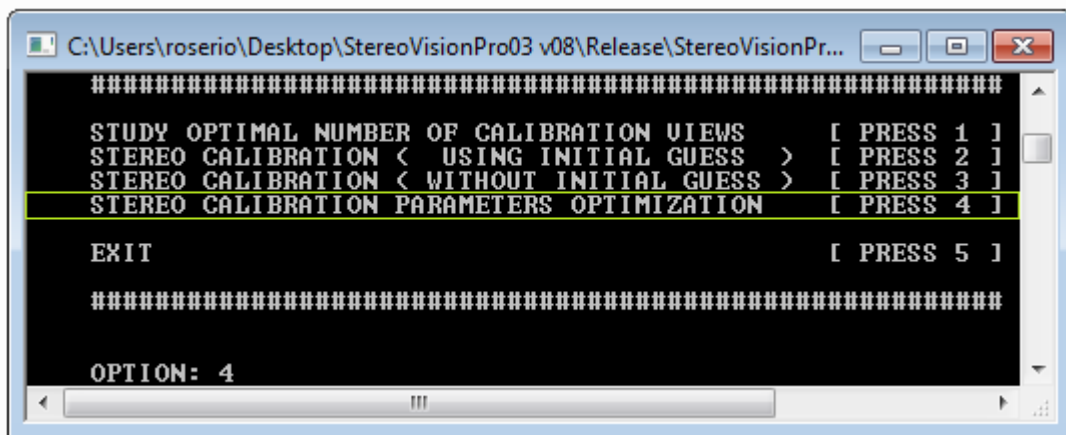


Figure 7.12: StereoVisionProg: Main menu's Option [ 3 ] sub Option [ 4 ].

After calibration was performed (using M1, M2 or M3 methods) the rectification process was called to compute undistortion+rectification maps and the disparity-to-depth matrix to use later on the process of recovering the 3D points from 2D tracked image points. Two rectification

methods were implemented as shown in Figure 7.13. On the first method the StereoVisionProg functionality starts by loading the calibration parameters, and the stereo

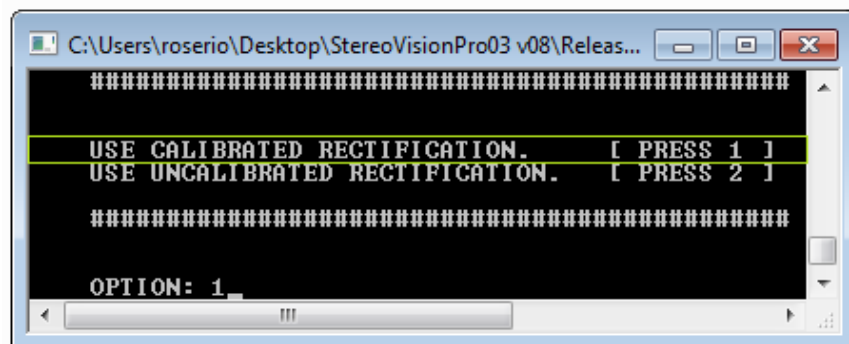


Figure 7.13: StereoVisionProg: rectification method options.

relations (R, T) from the last calibration operation's output file, and performs the Bouguet's calibrated stereo rectification directly.



For the second approach was implemented the Hartley's uncalibrated stereo rectification, if this option is selected the program loads the left and right image points ( **iPoints1** and **iPoints2**) generated from the last calibration operation, and proceeds as summarized in the next steps:

1. Compute fundamental matrix using left and right calibration view's image points.

```
F = cv::findFundamentalMatrix( iPoints1, iPoints2, ... );
```

2. Compute rectification homography matrices H1 and H2.

```
cv::stereoRectifyUncalibrated( iPoints1, iPoints2, F, imageSize, H1, H2, ...);
```

3. Pre-process homography matrices H1 and H2 to obtain the rectification transformations matrices R1 and R2 in object space.

$$R_1 = M_1^{-1} \times H_1 \times M_1 \text{ and } R_2 = M_2^{-1} \times H_2 \times M_2$$

4. Get the optimal new camera matrix to obtain the new principal point corrected.

```
nM1 = cv::getOptimalNewCameraMatrix( M1, D1, imageSize, 1, imageSize, 0 );
```

```
nM2 = cv::getOptimalNewCameraMatrix( M2, D2, imageSize, 1, imageSize, 0 );
```

5. Compute (undistortion + rectification) maps for both left and right images.

```
cv::initUndistortRectifyMap( M1, D1, ..., map1x, map1y );
```

```
cv::initUndistortRectifyMap( M1, D1, ..., map2x, map2y );
```

6. Build the disparity-to-depth transformation matrix.

$$Q = \begin{bmatrix} 1 & 0 & 0 & -cx \\ 0 & 1 & 0 & -cy \\ 0 & 0 & 0 & fx \\ 0 & 0 & -1/Tx & 0 \end{bmatrix}$$

The output file that results from rectification contains the remapping maps (undistortion + rectification) for both cameras (map1x, map1y, map2x, map2y) and the disparity-to-depth matrix (Q). The output results from stereo rectification process are saved into an xml file named **Lab3CalibListS02\_CalibRectification.xml** for calibrated rectification or **Lab3CalibListS02\_UncalibRectification.xml** if uncalibrated rectification method was selected.

6. The parametrization of a 3D rotation matrix into Euler Angles and Quaternions is frequently an indispensable operation in vision systems, computer graphics, robotics and kinematics in order to perform certain operations faster and avoid to have to deal with the rotation matrix. To perform the stereo relations rotation matrix ( R ) parametrization into Euler angles and quaternions were implemented functions based on the *Computing Euler Angles From a Rotation Matrix* (Slabaugh, year n.a.) online document.

The rotation matrix parametrization into Euler angles allows to describe the rotations that moves a rigid body from one referential to other with different orientation by using only three parameters – Euler angles: [  $\varphi$   $\theta$   $\psi$  ]. The approach formulation used to compute the Euler angles from a rotation matrix is as follows:

Define the 3-by-3 orthonormal rotation matrix to be parametrized as:

$$R_{3D} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \quad (7.1.5)$$

The rotation around XX axis -  $\psi(psi)$  is defined by:

$$R_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi) & \cos(\psi) \end{bmatrix} \quad (7.1.6)$$

Similarly, the rotation around YY axis -  $\theta(theta)$  is defined by:

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \quad (7.1.7)$$

The third rotation around ZZ axis -  $\phi(phi)$  is defined by:

$$R_z(\phi) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (7.1.8)$$

The Euler angles  $[\phi \theta \psi]$ , that represents the rotation first around XX axis, then around YY axis are later around ZZ axis, are presented in the reduced form by the next equation:

$$R = R_z(\phi) R_y(\theta) R_x(\psi) \quad (7.1.9)$$

The matrix product R from equation (7.1.9) is presented in the matrix form by:

$$R = \begin{bmatrix} \cos(\theta)\cos(\phi) & \sin(\psi)\sin(\theta)\cos(\phi) - \cos(\psi)\sin(\phi) & \cos(\psi)\sin(\theta)\cos(\phi) + \sin(\psi)\sin(\phi) \\ \cos(\theta)\sin(\phi) & \sin(\psi)\sin(\theta)\sin(\phi) + \cos(\psi)\cos(\phi) & \cos(\psi)\sin(\theta)\sin(\phi) - \sin(\psi)\cos(\phi) \\ -\sin(\theta) & \sin(\psi)\cos(\theta) & \cos(\psi)\cos(\theta) \end{bmatrix} \quad (7.1.10)$$

By combining the both  $R_{3D}$  and  $R$  matrices (eq. (7.1.5) and (7.1.10)) is possible to determine the Euler angles. Case  $\cos(\theta) \neq 0$  the Euler angles has two valid solutions:  $(\phi_1, \theta_1, \psi_1)$  and  $(\phi_2, \theta_2, \psi_2)$ . Case  $\cos(\theta) = 0$  the Euler angles have infinite number of solutions, this case is known by Gimbal Lock Problem – loss of one degree of freedom in the 3D space. In both cases the solutions are computed as follows:

$$Case(R_{31} \neq \pm 1) \Rightarrow (\cos(\theta) \neq 0); Solutions [(\phi_1, \theta_1, \psi_1) \text{ and } (\phi_2, \theta_2, \psi_2)] \dot{!}$$

$$\begin{bmatrix} \theta_1 = -\text{asin}(R_{31}) & \theta_2 = \pi - \theta_1 \\ \phi_1 = \text{atan2}\left(\frac{R_{21}}{\cos(\theta_1)}, \frac{R_{11}}{\cos(\theta_1)}\right) & \phi_2 = \text{atan2}\left(\frac{R_{21}}{\cos(\theta_2)}, \frac{R_{11}}{\cos(\theta_2)}\right) \\ \psi_1 = \text{atan2}\left(\frac{R_{32}}{\cos(\theta_1)}, \frac{R_{33}}{\cos(\theta_1)}\right) & \psi_2 = \text{atan2}\left(\frac{R_{32}}{\cos(\theta_2)}, \frac{R_{33}}{\cos(\theta_2)}\right) \end{bmatrix}$$

Case( $R_{31} = \pm 1$ )  $\Rightarrow (\theta = \frac{-\pi}{2} \vee \theta = \frac{\pi}{2})$ ; Infinite number of solutions can be used to transform a rigid body from its initial referential to a second referential with different orientation:  
 $(\phi_i = \text{any value}, \theta_i, \psi_i)$

$\phi_i = \text{any value}$ . Normally set to 0.

case( $R_{31} = -1$ )

$$\theta_i = \frac{\pi}{2}$$

$$\psi_i = \phi_i + \text{atan2}(R_{12}, R_{13})$$

case( $R_{31} = +1$ )

$$\theta_i = -\frac{\pi}{2}$$

$$\psi_i = -\phi_i + \text{atan2}(-R_{12}, -R_{13})$$

Figure 7.14 shows the Euler angles application to transform the right camera imager orientation into the left camera orientation. In OpenCV the rectification process is done by dividing the rotation matrix in two rotations and both left and right referential are rotated to a common plane.

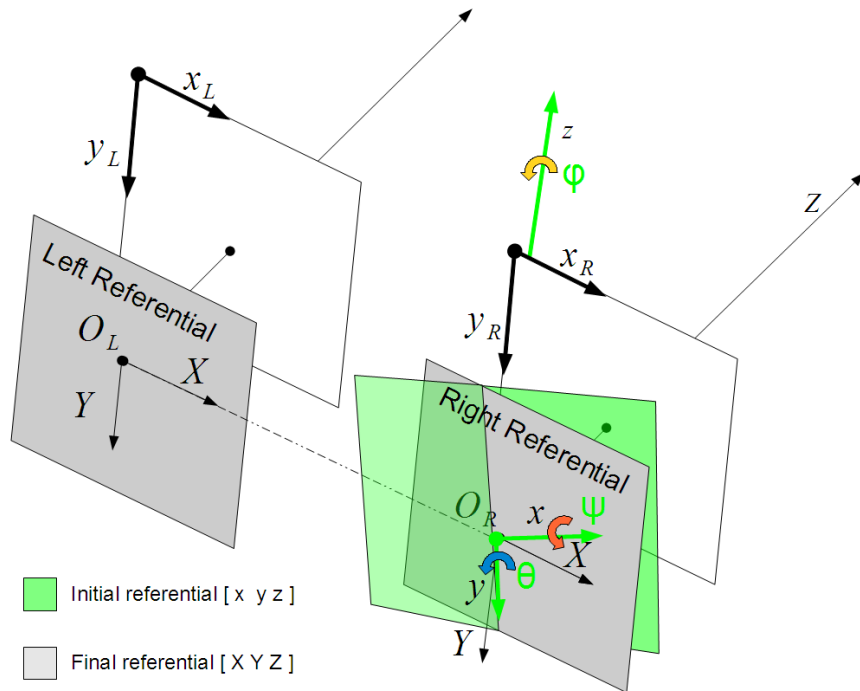


Figure 7.14: Right camera rotation using Euler angles.

- Quaternions are another form to represent a 3D rotation matrix. Quaternions represent orientations and rotations of objects in 3D dimensions. Assuming that  $q$  is the quaternion for the rotation that transforms a rigid body from one referential to a second referential with different orientation, the rotation matrix  $R(q)$ , or similarly  $R_{3D}$  as in eq. (7.1.5), corresponding to  $q$  is as follows:

$$R(q) = \begin{bmatrix} 1 - 2q_2^2 - 2q_3^2 & 2q_1q_2 + 2q_4q_3 & 2q_1q_3 - 2q_4q_2 \\ 2q_1q_2 - 2q_4q_3 & 1 - q_1^2 - q_3^2 & 2q_2q_3 + 2q_4q_1 \\ 2q_1q_3 + 2q_4q_2 & 2q_2q_3 - 2q_4q_1 & 1 - 2q_1^2 - q_2^2 \end{bmatrix} \quad (7.1.11)$$

Where  $(q_1, q_2, q_3, q_4)$  are the four quaternion's components  $q = [q_1 q_2 q_3 q_4]^T$ . In the next figure (Figure 7.15) is presented the pseudo-code based on the work *From Quaternion to Matrix and Back* (Waveren, 2005). The approach avoids numerical instability by selecting the trace (sum of the diagonal elements) with higher values.

```

if (  $R_{11} + R_{22} + R_{33} > 0.0$  )
    trace =  $R_{11} + R_{22} + R_{33} + 1$ 
     $s = \text{ReciprocalSqrt}(\text{trace}) * 0.5$ ;
     $q_1 = (R_{23} - R_{32}) * s$ 
     $q_2 = (R_{31} - R_{12}) * s$ 
     $q_3 = (R_{12} - R_{21}) * s$ 
     $q_4 = s * \text{trace}$ 
elseif (  $R_{11} > R_{22}$  and  $R_{11} > R_{33}$  )
    trace =  $R_{11} - R_{22} - R_{33} + 1$ 
     $s = \text{ReciprocalSqrt}(\text{trace}) * 0.5$ ;
     $q_1 = s * \text{trace}$ 
     $q_2 = (R_{12} - R_{21}) * s$ 
     $q_3 = (R_{31} - R_{13}) * s$ 
     $q_4 = (R_{23} - R_{32}) * s$ 
elseif (  $R_{22} > R_{33}$  )
    trace =  $-R_{11} + R_{22} - R_{33} + 1$ 
     $s = \text{ReciprocalSqrt}(\text{trace}) * 0.5$ ;
     $q_1 = (R_{12} - R_{21}) * s$ 
     $q_2 = s * \text{trace}$ 
     $q_3 = (R_{23} - R_{32}) * s$ 
     $q_4 = (R_{31} - R_{13}) * s$ 
else
    trace =  $-R_{11} - R_{22} + R_{33} + 1$ 
     $s = \text{ReciprocalSqrt}(\text{trace}) * 0.5$ ;
     $q_1 = (R_{31} - R_{13}) * s$ 
     $q_2 = (R_{23} - R_{32}) * s$ 
     $q_3 = s * \text{trace}$ 
     $q_4 = (R_{12} - R_{21}) * s$ 

```

Figure 7.15: Pseudo-code to compute quaternion from  $R$ .

7. In order to obtain 2D image points with the StereoVisionProg options from the sets of cine files with the robot movement video sequence (StereoS02L and StereoS02R, see Table 7.2), the cine files were converted to AVI files (**StereoS02L.avi** and **StereoS02R.avi**) format using PCC software with the next settings: **Frame Rate (fps) = 25** and **Video Compressor: none** to preserve the original bitmaps (or raster) images quality with which the cine files were recorded.
8. To obtain the MELFA RV-2AJ end-effector's 3D path from StereoS02L.avi and StereoS02R.avi files was proceeded as follows:

Using StereoVisionProg program was chosen: **Main menu's Option [ 4 ] sub Option [ 2 ]**, as shown in Figure 7.16, this option listed all AVI files inside the current directory from where StereoVisionProg was executed, from this list were selected **StereoS02L.avi** and

**StereoS02R.avi** while was ensured that the current stereo calibration parameters corresponded to the same stereo configuration used to obtain those video sequences (see footnote 4). After being selected the capture mode and the left and right video sequence input the program loaded the remapping maps and the disparity-to-depth matrix.

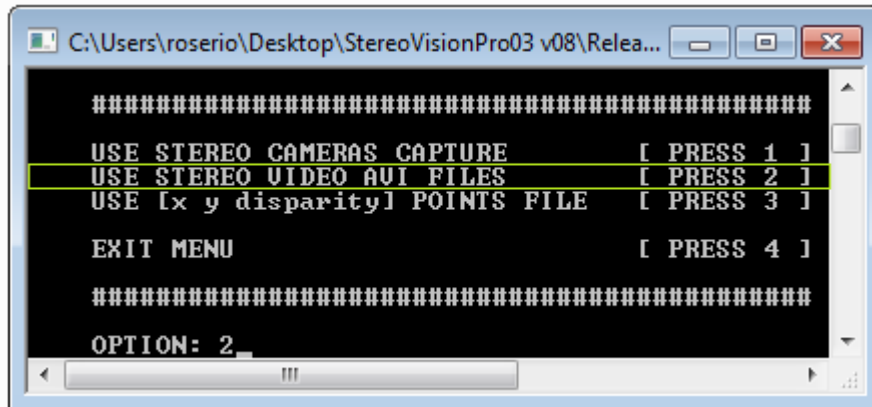


Figure 7.16: StereoVisionProg: Main menu's Option [ 4 ] sub Option [ 2 ].

A second sub menu was displayed to select the matching mode. For the stereo matching process were implemented different stereo matching approaches as shown in Figure 7.17. **Option [ 1 ] - COMPUTE DENSE IMAGE OF 3D POINTS** was implemented using the OpenCV stereo correspondence functions. This functionality allows to select one of the following algorithms: **Block-Matching**, **Semi Global Block-Matching**, and **Graph-Cut** algorithm, this last method is a non real-time stereo correspondence algorithm and its use with video sequences proved to be inefficient. For each method the program applied the remapping map to each new left and right video frame capture, computed the disparity image, and then reprojected it to 3D image space. The results obtained with this dense stereo correspondence algorithms were unsatisfactory and a new approach was implemented. The full implementation description of this option is described in more detail on the appendices section (see Appendix H: StereoVisionProg).

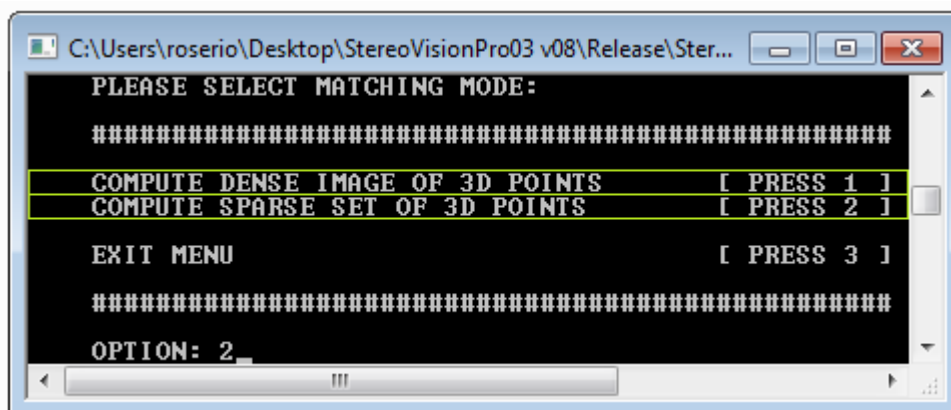


Figure 7.17: StereoVisionProg: Main menu's Option[ 4 ] sub Option[ 2 ] sub Option[ 2 ].

A second option **Option [ 2 ] - COMPUTE SPARSE SET OF 3D POINTS** was implemented on StereoVisionProg program. For this case study only a sparse set of points obtained from a stereo video sequence were needed to recover the MELFA RV-2AJ robot's end-effector path and then compare the results obtained with the robot's path.

By selecting StereoVisionProg: Main menu's Option [ 4 ] sub Option [ 2 ] sub Option [ 2 ] the program uses the Lucas-Kanade Sparse Optical Flow (see Appendix F: Motion) method to track points from the left to the right video capture, compute the disparities between left and right tracked points and then project those points to 3D space. The implementation methodology used by this approach is described as follows:

- The program captures the first left and right video frame from the video sequence and waits until a predefined number of points is added, over the left capture, with right mouse click event.
- Then the program starts capturing frames from both video sequences without interruption and the initial image points are tracked from the previous-left to current-left frame and then from the current-left to the current-right frame capture as shown in Figure 7.18.

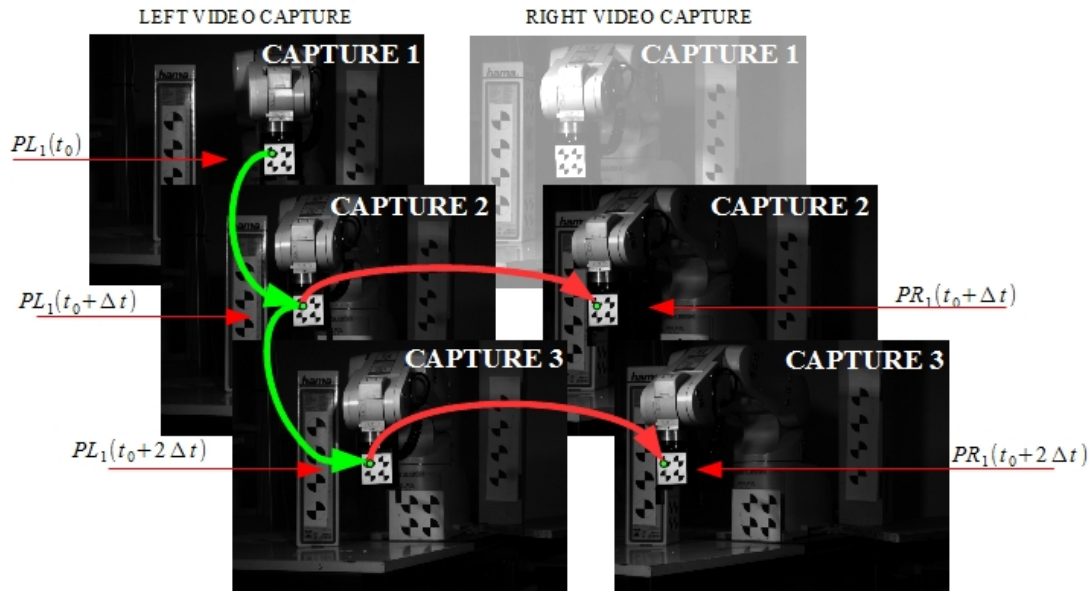


Figure 7.18: Sparse stereo correspondence with Lucas-Kanade tracker.

Points tracked successfully are saved and points that fail to be tracked at any instant are removed. At the end of the video capture two vectors of vectors with each point positions are available for the left and for the right image points. Using Figure 7.18 nomenclature, the resulting vectors are as follows:

$$\begin{aligned}
 \text{leftTrackedPoints} &= [\text{vect}PL_1 \quad \text{vect}PL_2 \quad \dots \quad \text{vect}PL_m] \\
 \text{rightTrackedPoints} &= [\text{vect}PR_1 \quad \text{vect}PR_2 \quad \dots \quad \text{vect}PR_m] \\
 \text{vect}PL_i &= [PL_i(t_0) \quad PL_i(t_0 + \Delta t) \quad PL_i(t_0 + 2\Delta t) \quad \dots \quad PL_i(t_0 + n\Delta t)] \\
 \text{vect}PR_i &= [PR_i(t_0) \quad PR_i(t_0 + \Delta t) \quad PR_i(t_0 + 2\Delta t) \quad \dots \quad PR_i(t_0 + n\Delta t)]
 \end{aligned}$$

Where  $m$  is the number of points successfully tracked over  $n$  stereo video captures.

The first step to obtain 3D points from 2D image points is to compute horizontal disparity

<sup>5</sup> vector as follows:

$$dP_i = [PR_i(1).x - PL_i(1).x, \dots, PR_i(n).x - PL_i(n).x]$$

The resulting 3D object points can be related to the left camera coordinate system or to the right camera coordinate system. Each case is possible by using the left or right image points coordinates, respectively, and the disparity vector to build a vector with homogeneous coordinates:

$$H_{Coord} P_i(\text{left}) = [PL_i(1).x \quad PL_i(1).y \quad dP_i(n) \quad 1, \dots, PL_i(n).x \quad PL_i(n).y \quad dP_i(n) \quad 1]$$

$$H_{Coord} P_i(\text{right}) = [PR_i(1).x \quad PR_i(1).y \quad dP_i(n) \quad 1, \dots, PR_i(n).x \quad PR_i(n).y \quad dP_i(n) \quad 1]$$

Having the homogeneous coordinates vector the program proceeds with the disparity to depth transformation. Knowing that the disparity-to-depth matrix is composed by the following parameters:

$$Q = \begin{bmatrix} 1 & 0 & 0 & -c_{xl} \\ 0 & 1 & 0 & -c_{yl} \\ 0 & 0 & 0 & f \\ 0 & 0 & \frac{1}{T_x} & \frac{(c_{xl} - c_{xr})}{T_x} \end{bmatrix} \Rightarrow Q = \begin{bmatrix} Q_{00} & Q_{01} & Q_{02} & Q_{03} \\ Q_{10} & Q_{11} & Q_{12} & Q_{13} \\ Q_{20} & Q_{21} & Q_{22} & Q_{23} \\ Q_{30} & Q_{31} & Q_{32} & Q_{33} \end{bmatrix}$$

The depth for a given point P (as in Figure 7.19) projected into a canonical stereo configuration is obtained using the similarity of triangles as follows:

$$\frac{Z}{T_x} = \frac{Z-f}{T_x-d} \Rightarrow Z = \frac{f \times T_x}{d} \quad (7.1.12)$$

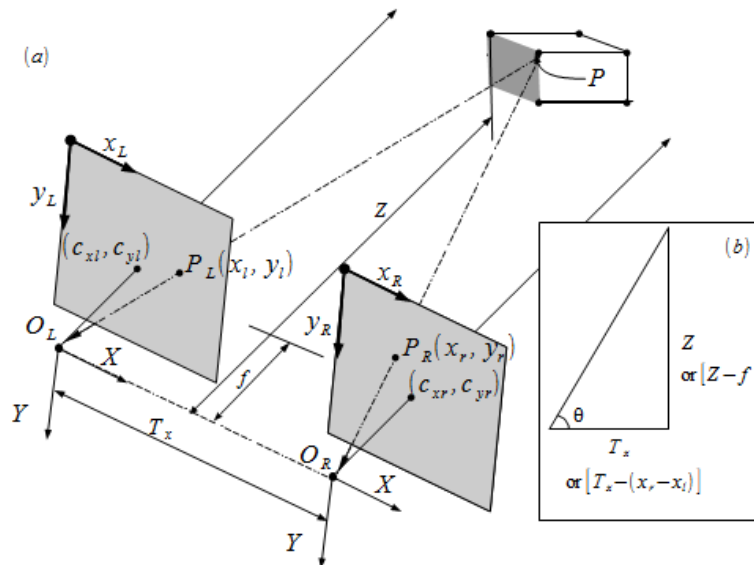


Figure 7.19: Canonical stereo configuration (a), similarity of triangles(b).

5 This step assumes that the stereo video captures are already undistorted and rectified such that the stereo configuration is close to a canonical stereo configuration i.e. the image planes are coplanar and row aligned.

Then the vector of 3D points of each tracked point was obtained using the following relation:

$$(Related\ to\ O_LXY)\ [X\ Y\ Z\ W]^T = Q[x_l\ y_l\ d\ 1]^T \quad (7.1.13)$$

$$(Related\ to\ O_RXY)\ [X\ Y\ Z\ W]^T = Q[x_r\ y_r\ d\ 1]^T \quad (7.1.14)$$

Generalizing equations (7.1.13 and (7.1.14 and recalling the disparity-to-depth matrix the coordinates of a 3D point are formulated as follows:

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = Q \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} X = x * Q_{00} + Q_{03} & \Rightarrow X = x - c_{xl} \\ Y = y * Q_{11} + Q_{13} & \Rightarrow Y = y - c_{yl} \\ Z = Q_{23} & \Rightarrow Z = f \\ W = d * Q_{32} + Q_{33} & \Rightarrow W = \frac{d}{T_x} \end{bmatrix}$$

The  $Q_{33}$  disparity-to-depth matrix element is null, i.e. the principal rays of each camera meet at infinity ( $c_{xl} - c_{xr} = 0$ ) and  $W$  depends only on the disparity and the horizontal distance between cameras. The last step to obtain  $[XYZ]$  points is done as follows:

$$[XYZ]^T = [X/W\ Y/W\ Z/W]^T \quad (7.1.15)$$

Or in the detailed form:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \frac{(x - c_{xl}) \times T_x}{d} \\ \frac{(y - c_{yl}) \times T_x}{d} \\ \frac{f \times T_x}{d} \end{bmatrix}$$

The same result is obtained for the depth as in equation (7.1.12). This methodology was the approach used to obtain the 3D path from the stereo video sequences. In practise, when StereoVisionProg: Main menu's Option [ 4 ] sub Option [ 2 ] sub Option [ 2 ] was chosen three windows were displayed: CAPTURING FROM AVI FILES, POINTS TRACKING CONTROLS and LEFT IMAGE. Before proceeding the tracking parameters were adjusted using the trackbars on POINTS TRACKING CONTROLS window.

- **Trackbar NPts:** sets the maximum number of image points to be tracked, the video capture will be waiting until the user selects this number of points over the (LEFT IMAGE window) left video frame capture.



- **Trackbars PyrLevel and WinSize:** sets the Lucas – Kanade tracker pyramid level and the search window size, this two values need to be adjusted on such a way that the left and right tracked image points represent the same 3D object points on a scene. As example, for the stereo set S02, values of PyrLevel = 7 and WinSize = 15 were used during the tracking operation.

By using the LEFT IMAGE window with the (first) left video frame capture displayed were added (using left mouse click event) the number of points setted by NPts trackbar. A target (Target 2) with four points was attached to the robot's end-effector during the experiment to provide good features to track and therefore NPts value was set to 4 and only Target2's four corners were selected.

After selecting the image points to be tracked over the first video frame capture the program starts by reading the video sequence and tracking the existing points. For each new left and right image capture the program stores the the new position of each left and its corresponding right image point. If the user clicks on the LEFT IMAGE window to make it active and press “C” key the program computes the horizontal disparity between the left and right image points positions, then it computes the 3D world coordinates, saves the 3D points and the [x y disparity] points into an output xml file. The output file name ending with “\_3DPOINTSC1.xml” has the 3D points related to the left camera coordinate system while the file name ending with “\_3DPOINTSC2.xml” has the 3D points related to the right camera coordinate system.

Considering that the list of calibration views used to calibrate the stereo configuration was Lab3CalibListS02.txt the resulting output file is: **Lab3CalibListS02\_3DPOINTSC1.xml** or **Lab3CalibListS02\_3DPOINTSC2.xml** depending on which camera was active <sup>6</sup>. To define which camera (coordinate system) is active the user should change the “**CameraOn**” trackbar value to 1 (left camera) or 2 (right camera) in POINTS TRACKING CONTROLS window.

To compare the obtained 3D set of points (3D path) with the real 3D path given by MELFA Basic IV software was necessary to transform the resulting 3D points (related to each camera coordinate system) to the robot referential as shown in the next figure (Figure 7.20).

---

6 All the results output files are named after the calibration list file name used to calibrate the current stereo configuration, i.e. If **Lab3CalibListS02.txt** calibration list was used to calibrate the current stereo configuration, all output files are named using “**Lab3CalibListS02**” as prefix, as in **Lab3CalibListS02\_CalibrationM1.xml**; **Lab3CalibListS02\_CalibrationM2.xml**; **Lab3CalibListS02\_Rectification.xml**; **Lab3CalibListS02\_Angles.xml**; **Lab3CalibListS02\_3DPOINTSC1.xml** and **Lab3CalibListS02\_3DPOINTSC2.xml**.

The coordinates system transformation was performed using the point-line-plane approach. To accomplish this task a set of points were tracked over a target (Target1) that was attached to the robot's base providing better features to track.

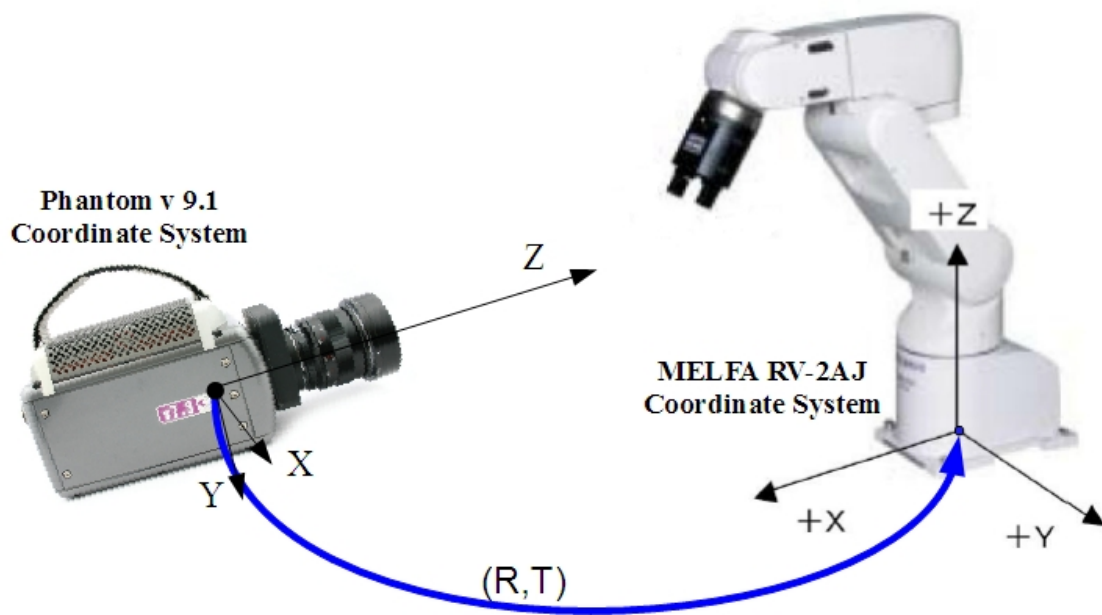


Figure 7.20: Camera-to-MELFA robot referential transformation.

Using the program on the same way as it was used previously to track points, a set of three points  $P_0$ ,  $P_1$  and  $P_{yz}$  as shown in Figure 7.21 were tracked and transformed to 3D space. Using the resulting set of 3D points were then computed the the rotation  $R$  and translation  $T$  that allows to transform the 3D path to MELFA robot's coordinate system.

The formulation to obtain  $R$  and  $T$  is described next:

$$Transform = R.T \quad (7.1.16)$$

$$\text{Where } [R] = \begin{bmatrix} R_{11} & R_{12} & R_{13} & 0 \\ R_{21} & R_{22} & R_{23} & 0 \\ R_{31} & R_{32} & R_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } T = \begin{bmatrix} 1 & 0 & 0 & P_{0.x} \\ 0 & 1 & 0 & P_{0.y} \\ 0 & 0 & 1 & P_{0.z} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

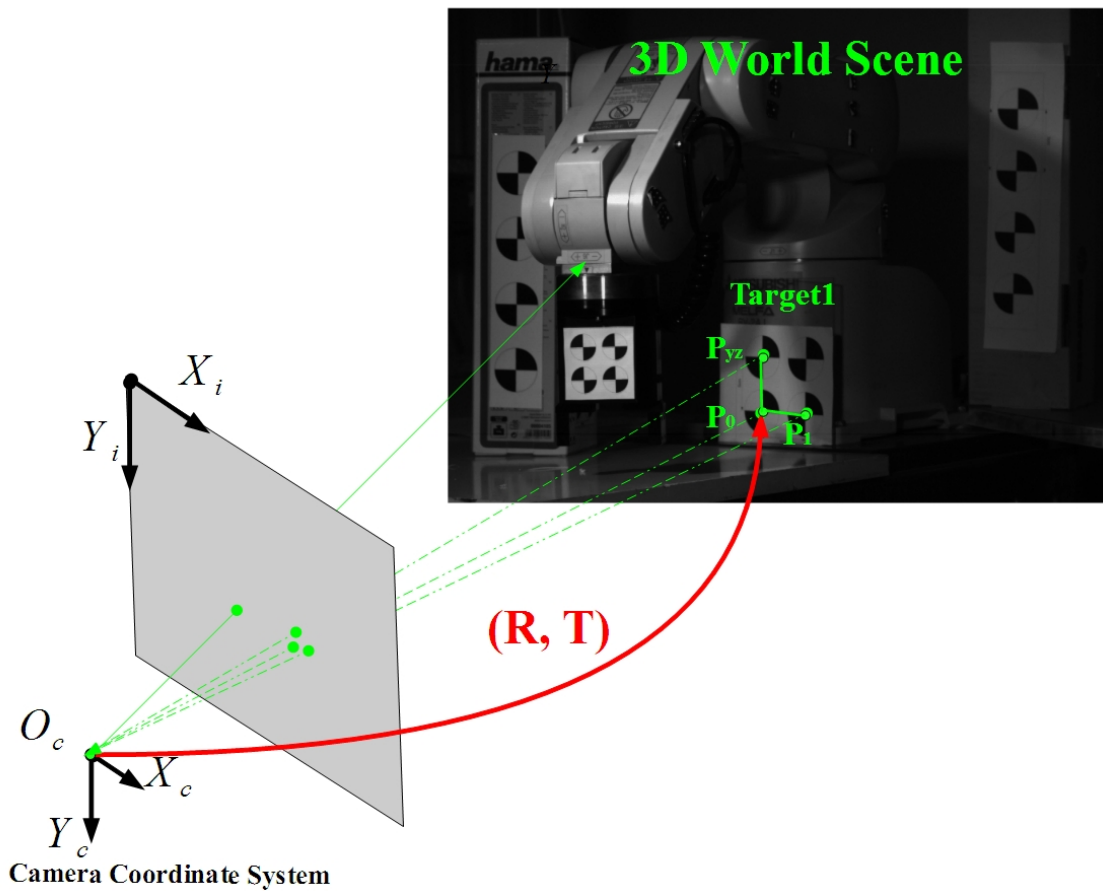


Figure 7.21: Referential transformation using point-line-plane method.

To build the rotation matrix the direction cosines for all three axes were obtained by the as follows:

$$YY_{axis} = (P_{1.x} - P_{0.x}, P_{1.y} - P_{0.y}, P_{1.z} - P_{0.z});$$

$$YZ = (P_{yz.x} - P_{0.x}, P_{yz.y} - P_{0.y}, P_{yz.z} - P_{0.z});$$

The XX and ZZ axes are computed using the cross product as follows:

$$XX_{axis} = YY_{axis} \cdot YZ_{axis}$$

$$ZZ_{axis} = XX_{axis} \cdot YY_{axis}$$

To obtain the rotation matrix R all three vector were transformed to unit vectors:

$$XX_{axis} = \frac{XX}{|XX|}, YY_{axis} = \frac{YY}{|YY|}, ZZ_{axis} = \frac{ZZ}{|ZZ|}$$

The final rotation and translation matrix are then build using the direction cosines and the the origin point P<sub>0</sub> as demonstrated next:

$$[R] = \begin{bmatrix} XX_x & XX_y & XX_z & 0 \\ YY_x & YY_y & YY_z & 0 \\ ZZ_x & ZZ_y & ZZ_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad [T] = \begin{bmatrix} 1 & 0 & 0 & P_{0x} \\ 0 & 1 & 0 & P_{0y} \\ 0 & 0 & 1 & P_{0z} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Recalling equation (7.1.16) the final transformation is rewrite as follows:

$$[R] = \begin{bmatrix} XX_x & XX_y & XX_z & XX_x \times P_{0x} + XX_y \times P_{0y} + XX_z \times P_{0z} \\ YY_x & YY_y & YY_z & YY_x \times P_{0x} + YY_y \times P_{0y} + YY_z \times P_{0z} \\ YY_x & YY_y & YY_z & YY_x \times P_{0x} + YY_y \times P_{0y} + YY_z \times P_{0z} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To corroborate the 3D path obtained with the research methodology approach and compare it with the 3D path given by MELFA Basic IV program a number of functions were implemented using MatLab software, the description of this functions is presented in Table 7.3 as follows:

Table 7.3

*List of MatLab Implemented Functions*

MatLab M - Files	
File Name	File Description
getDataNode.m	Loads a single variable's data from an OpenCV generated xml file.
readDataFromXML.m	Reads a group of variables' data from an OpenCV generated xml file.
plot3DPath.m	Plots the 3D path obtained with the StereoVisionProg research-made program.
readMelfaData.m	Loads and plots the robot's end-effector 3D path given by MELFA BASIC IV software.
transformReferential.m	Transform the initial 3D points coordinate system (camera's coordinate system) to robot's coordinate system.
testTransformReferential.m	Uses a generic 3D points to transform 3D points in camera coordinates to a generic coordinate system.
Instructions.m	Instructions to use each function and reproduce the results presented in this case study.

## 8 Chapter Four

### 8.1 Results

#### 8.1.1 Introduction

Recovering the depth information through stereo vision requires, in general lines, three main steps: camera calibration and stereo relations estimation, image undistortion and rectification, and stereo correspondence. The last step presents the most challenging task on the 2D to 3D transformation process. The stereo correspondence performance and accuracy determines how good and at which cost the depth information is recovered from the correlation of both images, however, the first process is also determinant to obtain good results once the parameter obtained from it are directly involved on the 2D points reprojection to 3D space.

This section starts by presenting a summary (see Table 8.1) of the data collected during the three laboratory experiments, the data collected from the first two experiments sessions were used to take conclusions mainly connected with calibration purposes and optimize the following experiment procedure. Data collected during the third laboratory sessions were used for calibration and 3D points recovering purposes.

Table 8.1

#### *Video Sequences Collected During Laboratories Experiments*

Lab – Stereo Configuration	Stereo Video Data Collected					
	For Stereo Cameras Calibration			For Recovering 3D Information		
	Image Size [pixel]	Sample Rate [fps]	Exposure $1 \times 10^{-6}$ [s]	Image Size [pixel]	Sample Rate [fps]	Exposure $1 \times 10^{-6}$ [s]
L01 - S01	1600-1200	100/50 <sup>(*1)</sup>	5000	-	-	-
L01 - S02	1152-1152	100/50 <sup>(*1)</sup>	3000	-	-	-
L01 - S03	960-720	100	4500	-	-	-
L02 – S01 <sup>(*2)</sup>	672-480	1000	900	672-480	1000	900
L02 – S02 <sup>(*2)</sup>	672-480	1000	900	672-480	1000	900
L02 – S03 <sup>(*2)</sup>	672-480	1000	900	672-480	1000	900
L02 – S04 <sup>(*2)</sup>	672-512	1000	900	672-512	1000	900
L03 – S01 <sup>(*3)</sup>	960-720	200/700 <sup>(*1)</sup>	900	960-720	700	900
L03 - S02	960-720	700/90 <sup>(*1)</sup>	900	960-720	700	900
L03 - S03	960-720	700/90 <sup>(*1)</sup>	900	960-720	700	900
L03 - S04	960-720	90	900	960-720	700	900

Note.

(\*1) - The left/right camera sample rate (fps) used to capture video sequences for stereo calibration were different for each stereo configuration's Phantom v9.1 camera.

(\*2) - Video data obtained with second laboratory experiment (L02) were influenced by experimental errors and only video sequences for stereo calibration were used for stereo calibration purposes.

(\*3) -The first set of video sequences for calibration used a special pattern for calibration purposes with Tema Software, this set was not included for the research purposes.

Next, are presented a list of the output files generated by StereoVisionProg researcher-made program with all dependent variables.

All the result output files obtained with StereoVisionProg were named using [FileName1] and [FileName2] as prefix. Both cases are explained on point 1 and 2 as follows:

1. [FileName1] is the name of the text file containing the list of 150 pairs of left/right calibration views used to build calibration sets with different number of calibration views (N02, N05, N10, N20, N30, N40, N50, N60, N70, N80, N90, N100, N110, N120, N130, N140, N150) and study the difference between calibration methods and how calibration parameters evolve.
2. [FileName2] is the name of the text file containing a list of 100 pairs of left/right calibration views used to perform stereo calibration.

For a better understanding how the results were obtained and stored by using StereoVisionProg researcher-made program options and OpenCV XML FileStorage each output files obtained from the research study and its list of variables are explained in more detail in the next pages.

- Files [FileName1]\_StudyM1.xml and [FileName1]\_StudyM2.xml were obtained from STUDY OPTIMAL NUMBER OF CALIBRATION VIEWS process using different sets of images and calibration method M1 and M2 respectively. The list of variables stored inside this two resulting files are listed in Table 8.2 as follows:

Table 8.2

*Calibration Methods Study Process's Output Variables*

[FileName1]_StudyM1.xml and [FileName1]_StudyM2.xml	
Variables Names	Variables Description
fx1(fx2)	Focal length along XX for the left(right) camera in pixel units.
fy1(fy2)	Focal length along YY for the left(right) camera in pixel units.
(cx1(cx2), cy1(cy2))	Left (right) camera principal point coordinates.
K11, k21, p11, p21, K31 (K12, k21, p12, p22, K33)	Left(right) camera distortion coefficients.
R, T, E, F	Stereo configuration relations .
E1(E2)	Left(and right) camera reprojection errors. This two parameters are only available for [FileName1]_StudyM1.xml
E3	Stereo reprojection errors.

Note. Each variable represent a vector of 17 elements with the values of one parameter (p) resulting from performing the calibration (using method M1 and M2, separately) for each individual set of images as follows: pVect = [ p(N02), p(N05), ..., p(N150)].

- File [FileName2]\_CalibrationM1.xml and [FileName2]\_CalibrationM2.xml were obtained from STEREO CALIBRATION (USING INITIAL GUESS) and STEREO CALIBRATION (WITHOUT INITIAL GUESS) processes, on both cases were used around (but not less than) 100 calibration views. File [FileName2]\_CalibrationM3.xml was obtained from STEREO CALIBRATION PARAMETERS OPTIMIZATION operation, this process used exactly 100 calibration views. The list of variables resulting from each process are the same and listed in the next table (see Table 8.3 ).

Table 8.3

*Stereo Calibration Process's Output Variable*

[FileName1]_CalibrationM1.xml, [FileName1]_CalibrationM2.xml, and [FileName1]_CalibrationM3.xml	
Variables Names	Variables Description
calibrationListUsed	Name of the text file used to read/input calibration views.
squareSize	Chessboard square size.
cornersAlongXX	Number of chessboard corners along XX direction.
cornersAlongYY	Number of chessboard corners along YY direction.
ImageHeight(Width)	Calibration view image height(width).
numberOfViewsUsed	Number of views used for the stereo calibration process.
M1(M2)	Left(right) camera matrix.
D1(D2)	Left(right) distortion coefficients vector.
R	Rotation matrix to transform right camera to the left camera orientation.
T	Translation vector to transform right camera to the left camera position.
E, F	Essential and Fundamental matrices.
objPoints	XML node containing sub-nodes with object points for each view.
imgPoints1	XML node containing sub-nodes with chessboard corners extracted from each left calibration view with sub-pixel accuracy.
imgPoints2	XML Node containing sub-nodes with chessboard corners extracted from each right calibration view with sub-pixel accuracy.

- File [FileName2]\_Angles.xml was obtained from ROTATION MATRIX PARAMETRIZATION using rotation matrix R obtained from stereo calibration. The output variables stored in this file are listed in the next table (see Table 8.4).

Table 8.4

*Rotation Parametrization Process's Output Variables*

[FileName2]_Angles.xml	
Variable Name	Variable Description
EulerAngles	Contains the Euler angles resulting from stereo rotation matrix parametrization.
Quaternion	Contains the quaternions [x, y, z, w] components resulting from stereo rotation matrix parametrization.
NormQuaternion	Contains the quaternions [x, y, z, w] components in the normalized form.

- Files [FileName2]\_CalibRectification.xml and [FileName2]\_UncalibRectification.xml were obtained from USE CALIBRATED RECTIFICATION and USE UNCALIBRATED RECTIFICATION process, respectively. The rectification process is called each time a calibration operation is performed. The list of variables is different for each case as described in the next table (see Table 8.5).

Table 8.5

*Calibrated and Uncalibrated Rectification Process's Output Variables*

[FileName2]_CalibRectification.xml and [FileName2]_UncalibRectification.xml	
Variable Name	Variable Description
map1x, map1y	Contains the remapping maps (undistortion + rectification) for the left camera.
map2x, map2y	Contains the remapping maps (undistortion + rectification) for the right camera.
Q	Contains the disparity-to-depth matrix used for recovering the depth information.
R1(R2)	Left(right) image rectification transformation (3-by-3 rotation matrices).
P1(P2)	Left(right) image 3-by-4 projection matrix in the new rectified coordinate system. (only for calibrated stereo



	rectification).
F	Fundamental matrix (only for uncalibrated stereo rectification).
H1(H2)	Left(right) image rectification homography matrices (only for uncalibrated stereo rectification).
nM1(nM2)	New left(right) camera matrix that can be used to define the region of interest in the new corrected image (only for uncalibrated stereo rectification).

- Files **[FileName2]\_3DPOINTSC1.xml** and **[FileName2]\_3DPOINTSC2.xml** are the research's most significant and interesting output files. This two output files were obtained from COMPUTE 3D POINTS operation – the process of recovering 3D point's sets (or 3D paths) using stereo video sequences. The first file contains information related to the left camera while the second file contains information related to the second camera. The list of variables saved into this two files is described in the following table (see Table 8.6).

Table 8.6

*Recovering 3D Points Process's Output Variables*

[FileName2]_3DPOINTSC1.xml and [FileName2]_3DPOINTSC2.xml	
Variables Names	Variables Description
nTrackedPts	Contains the number of image points tracked successfully over the left and right stereo video captures.
POINT[i]	Contains the image point (i) positions transformed to 3D points related to the left(right) camera coordinate system.
xydPOINTS[i]	Contains sub-nodes with all tracked positions (j) coordinates for each left(right) image point (i) and its disparity, i.e. vector<vector<Point3f>> and Point3f point(i,j)= (x,y,d).

The second section of this chapter presents the results that aim to answer the the research questions proposed initially by this case study. This section first presents the results from the study that allowed to determine the optimal number of calibration views and the calibration method to be used with the recorded video sequences, followed by the results obtained with calibration process for all three laboratories. Next are presented the results obtained from the stereo relation rotation matrix parametrization into Euler angles and quaternions obtained from calibration optimization process. In order to answer the fourth question the results from calibrated/uncalibrated stereo rectification are then presented. The last subsection present the research's most important results from stereo matching and 2D to 3D image points reprojection.

### 8.1.2 Research Question N°1 – Results

For the sake of brevity of this chapter the sections that answers to this research question were included at the end of this thesis on Appendix A and Appendix F (see Appendix A: Stereo Imaging and Appendix F: Motion for more details). The first appendix introduces the theory involved in the stereo vision process and presents a short description of the main OpenCV function used along the research for stereo cameras calibration, stereo rectification and stereo correspondence.

### 8.1.3 Research Question N°2 – Results

#### 8.1.3.1 Calibration Method.

To determine the better calibration method M1 (calibration parameters are first computed and then used as initial guess for the stereo relations estimation) or M2 ( calibration parameters and stereo relations are estimated all at the same time) were performed the calibration process using 17 groups of calibration views with both methods.

The next tables (Table 8.7, Table 8.8, and Table 8.9) list the results obtained for  $f_x$  and  $k_2$  parameters. This two parameters were chosen by the researcher because they are key parameters for the image undistortion and the 2D to 3D points reprojection process.

To reduce the amount of data to be presented was selected only the most representative stereo configuration set from each laboratory session. For the first laboratory (L01) was chosen the stereo configuration set S03. The results obtained are the following ones:

Table 8.7

*L01 Set S03 Calibration Method's Study Using  $f_x$  and  $k_2$  Parameters*

Nr.Views	Calibration Method M1				Calibration Method M2			
	CAM1		CAM2		CAM1		CAM2	
	$f_x$ [mm]	$k_2$ Coeff.	$f_x$ [mm]	$k_2$ Coeff.	$f_x$ [mm]	$k_2$ Coeff.	$f_x$ [mm]	$k_2$ Coeff.
N02	61.12	-3.4339	99.13	100.2520	61.91	-1.5658	99.06	70.6224
N05	66.11	3.5202	43.25	3.5672	62.27	4.6500	60.98	18.8661
N10	69.91	6.9794	52.06	5.6630	67.59	6.1852	66.15	25.3631
N20	68.77	7.3210	63.00	7.7685	68.65	2.3293	65.22	13.4387
N30	65.69	-3.8577	61.11	-0.7246	66.81	-0.3026	62.24	4.6413
N40	58.74	-2.0649	57.78	-0.8054	57.70	-1.5079	57.87	1.5450
N50	58.14	1.6506	57.62	5.9070	57.87	0.5893	57.03	18.0777
N60	56.93	5.2146	56.88	11.0362	56.01	10.9857	56.94	5.0639
N70	55.77	11.0179	56.20	32.1065	57.05	7.6388	56.69	44.4908
N80	56.53	-10.0158	56.61	-30.5942	56.87	40.7903	57.49	-37.9087
N90	56.71	-9.6395	56.63	-20.6211	57.38	-4.8749	57.40	-14.1531
N100	56.66	-9.2385	56.65	-20.3329	57.37	-4.2595	57.40	-13.8773
N110	56.87	-14.6354	56.88	-28.9517	57.44	-7.2798	57.68	-29.7505
N120	56.97	-14.9543	57.02	-50.7394	57.58	-12.6142	57.91	-38.8561
N130	56.78	-15.6795	56.95	-27.7510	57.05	-4.4990	57.90	-34.9140
N140	56.72	-8.7754	56.81	-9.9102	57.61	-1.1175	58.16	-3.0403
N150	56.78	-6.5623	56.82	-19.5146	57.50	4.4200	58.04	0.7956
M	59.72	-3.7149	58.91	-2.5673	59.69	2.3275	61.42	1.7885
STD	4.75	8.4582	11.13	33.2310	4.16	11.5008	10.12	29.4984

Note.

- M is the mean value of each parameters and STD is its standard deviation.

- N02 is the minimum number of views that it is possible to use for camera calibration using OpenCV .

For the second laboratory (L02) was chosen the stereo configuration set S03 results as follows:

Table 8.8

*L02 Set S03 Calibration Method's Study Using  $f_x$  and  $k_2$  Parameters*

Nr. Views	Calibration Method M1				Calibration Method M2			
	CAM1		CAM2		CAM1		CAM2	
	$f_x$ [mm]	$k_2$ Coeff.	$f_x$ [mm]	$k_2$ Coeff.	$f_x$ [mm]	$k_2$ Coeff.	$f_x$ [mm]	$k_2$ Coeff.
N02	25.84	0.5538	24.25	-30.5224	26.45	0.5756	24.27	-31.8050
N05	26.73	0.2319	24.60	-6.9374	26.70	0.2036	24.77	-5.5796
N10	25.67	0.2442	24.64	-9.2991	25.16	0.1777	24.82	-4.6070
N20	24.60	0.2314	24.53	-2.7038	24.58	0.3045	24.93	-2.4494
N30	23.89	0.4147	24.63	-4.8201	24.76	0.5095	24.51	-3.3021
N40	24.99	-8.6206	24.80	-7.6992	24.50	-5.4747	24.06	-5.1000
N50	24.29	-4.1924	24.74	-5.7922	24.79	-5.2780	24.27	-5.2664
N60	24.24	-3.1244	24.80	-5.1918	24.99	-4.3087	24.47	-7.0131
N70	23.36	-1.1179	24.79	-2.2444	25.14	-3.3803	24.61	-6.0149
N80	23.15	-0.8641	24.51	0.6870	24.99	-2.6928	24.50	-0.1555
N90	22.96	-0.7248	24.39	1.5106	24.84	-2.3837	24.45	1.9087
N100	22.98	-0.7830	24.34	1.8044	24.75	-2.4361	24.57	1.5000
N110	23.12	-1.0002	24.31	2.4924	24.70	-2.6474	24.62	1.0336
N120	23.31	-1.2584	24.29	2.8157	24.66	-2.8637	24.64	0.7374
N130	23.41	-1.4367	24.27	2.3825	24.15	-2.2003	24.32	1.0834
N140	23.42	-1.5261	24.27	1.5377	23.85	-2.0587	24.06	1.2202
N150	23.44	-1.5823	24.28	0.6795	23.86	-2.0846	24.07	0.7478
M	24.08	-1.4444	24.50	-3.6059	24.87	-2.1199	24.47	-3.7095
STD	1.13	2.2283	0.21	8.0202	0.75	1.9314	0.26	7.8928

Note.

- M is the mean value of each parameters and STD is its standard deviation.

- N02 is the minimum number of views that it is possible to use for camera calibration using OpenCV .

Comparing the mean (M) and standard deviation (STD) values in Table 8.7 and Table 8.8 was possible to conclude:

1. Method M2 has slightly better results than method M1 for  $f_x$  and  $k_2$  parameters on both cameras.
2. The radial distortion coefficient  $k_2$  estimation has higher STD values than the focal length estimation, also, for both tables  $k_2$  value presents higher STD values for the second camera (CAM2) than the first camera(CAM1).
3. Comparing STD values from L01 and L02 is possible to observe that better results were obtained for the second experiment. This difference between the L01 and L02 results were due significant improvements on the second laboratory experiment's procedure.

The next table (see Table 8.9) presents the results obtained from the stereo video sequences for calibration recorded during the third laboratory sessions (L03).

For the third laboratory session (L03) was chosen the stereo configuration video sequence set S04 as follows:

Table 8.9

*L03 Set S04 Calibration's Method's Study Using  $f_x$  and  $k_2$  Parameters*

Nr. Views	Calibration Method M1				Calibration Method M2			
	CAM1		CAM2		CAM1		CAM2	
	$f_x$ [mm]	$k_2$ Coeff.	$f_x$ [mm]	$k_2$ Coeff.	$f_x$ [mm]	$k_2$ Coeff.	$f_x$ [mm]	$k_2$ Coeff.
N02	27.01	0.2011	29.72	-0.8921	28.96	-0.0767	28.84	-0.8921
N05	27.66	0.0745	31.04	-0.8048	29.17	-0.0122	29.67	-0.7196
N10	28.01	0.1842	29.70	3.5332	28.32	0.3892	29.56	5.2769
N20	28.04	0.1956	29.05	3.2565	28.34	0.4867	28.63	4.7466
N30	28.05	0.1656	28.85	2.2236	28.20	0.3617	28.44	2.9455
N40	28.05	0.1670	28.68	1.2541	28.16	0.2800	28.40	2.0997
N50	28.06	0.1267	28.67	0.9682	28.13	0.1708	28.41	1.5279
N60	28.06	0.1477	28.60	0.7652	28.09	0.1336	28.42	1.1793
N70	28.06	0.1550	28.60	0.7727	28.08	0.1163	28.43	1.0771
N80	28.06	0.0656	28.55	0.8209	28.07	0.0265	28.43	0.9957
N90	28.06	0.0771	28.53	0.7681	28.07	0.0435	28.43	0.9068
N100	28.06	0.0810	28.53	0.7025	28.07	0.0572	28.43	0.8388
N110	28.06	0.0574	28.51	0.5621	28.07	0.0411	28.42	0.6795
N120	28.05	0.0481	28.50	0.4339	28.07	0.0429	28.41	0.5405
N130	28.05	0.0755	28.48	0.3983	28.07	0.0717	28.41	0.4509
N140	28.05	0.0774	28.47	0.4462	28.07	0.0733	28.41	0.4481
N150	28.16	0.3978	28.46	0.5543	28.10	0.3741	28.41	0.5232
M	27.97	0.14	28.88	0.93	28.24	0.15	28.60	1.33
STD	0.27	0.09	0.68	1.16	0.32	0.16	0.40	1.65

Note.

- M is the mean value of each parameters and STD is its standard deviation.

- N02 is the minimum number of views that it is possible to use for camera calibration using OpenCV.

From L03 results (see Table 8.9) is possible to conclude that  $f_x$  and  $k_2$  STD's values improved substantially, the following observation were taken:

1. The parameters obtained with calibration method M1 had smaller STD values than calibration method M2.
2. Calibration parameters values computed for CAM2 have higher errors comparatively with CAM1 as observed for L01 and L02 results. This error are more visible for the radial distortion coefficient  $k_2$ .

By comparing the results obtained with L01, L02, and L03 is possible to observe that the calibration results obtained from the L03's video sequences had significantly improved. This improvements were due few changes and considerations taken in account for L03 during the process of recording video sequences for calibration, as listed below:

- Lightning conditions were improved by using proper LED's arrays for high-speed framing.
- Both Phantom v.91 camera's lenses were correctly focused on a focal-plane from where the sequence of calibration views were captured.
- Contrary to L01 and L02, on L03 experiment was avoided the used of lens

maximum/minimum focal length range where the distortions are more visible.

- While on L01 and L02 were used high frame rates (100 and 1000 respectively) for L03 was used 90 fps to allow more recording time and capture calibration views with rich positions and orientations, i.e. object calibration images which the position and orientation varies considerably from frame capture to frame capture.
- Smaller image resolution (960x720) were used to capture video sequences with phantom v. 9.1 camera's and avoid the higher distortion that are more notable in the sensor corners.
- During L03 was used the CSR feature to compute the pixel offset for the current frame rate, resolution and exposure settings, giving a more precise compensation of the pixel errors and better calibration results.

Calibration methods M1 and M2 gave similar results, method M1 proved to give better results with the data collected during the third laboratory where the procedure errors were minimized and the video recording settings were properly chosen, also, taking in account computational costs, method M1 is less demanding and less time consuming. and therefore M1 was the method used in this case study to perform the stereo calibration process.

### 8.1.3.2 Optimal Number of Calibration Views.

The literature suggests that to perform camera calibration with OpenCV a number between 30 and 100 calibration views should be used, however this value depends highly on the calibration views quality such as sharpness, luminosity, rich sets of positions and orientations, and the area covered by the calibration object on the image.

After performing the calibration for the 17 different groups of calibration views was studied how the camera's focal length evolve to determine the optimal number of calibration views. Figure 8.1 shows camera's focal length results obtained for L01 set S03 calibration views as follows:

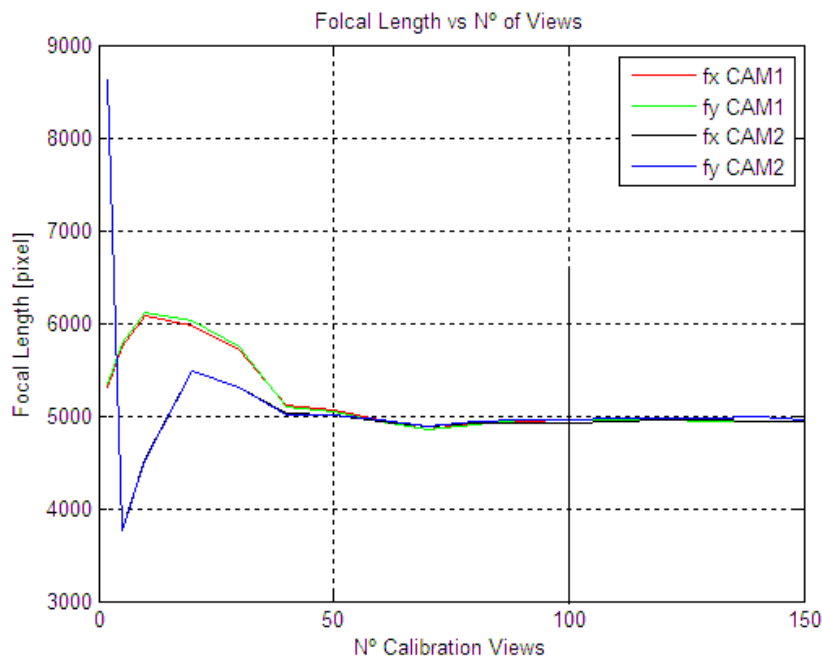


Figure 8.1: L01-S03 Focal Length vs N° of Calibration Views (M1 and M2).



Figure 8.2 shows camera's focal length results obtained for L03 set S04 calibration views as follows:

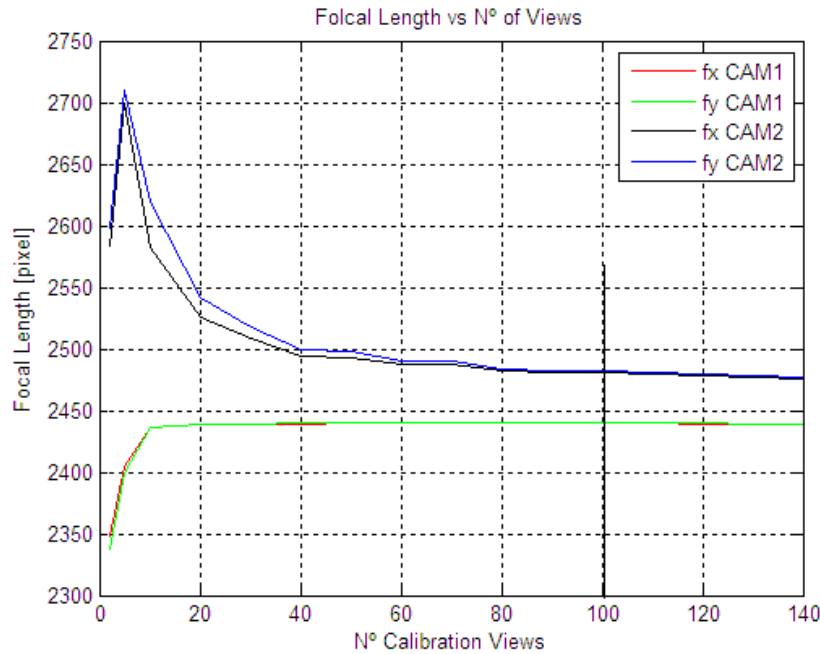


Figure 8.2: L03-S04 Focal Length vs N° of Calibration Views (M1 and M2).

Based on the L01 set S03 and L03 set S04's plots ( Figure 8.1 and Figure 8.2 ) was possible to take the following conclusions:

1. The focal length values were acceptably stable, i.e. with small variations for a number of 100 calibration views.
2. The focal length parameter estimation for the second camera (fx CAM2 and fy CAM2 plots) had in general higher fluctuations than the first camera (fx CAM1 and fy CAM1 plots). This difference between values are justified by the fact that after a thorough video analysis were detected few small dark regions on the image caused by dust particles in one phantom v9.1 camera's sensor.
3. Calibration parameters converge faster for calibration views with higher quality, i.e. image focus, illumination conditions and area percent occupied by the calibration object. Figure 8.3 shows two calibration views from L02 set S03 and L04 set S04 where the image quality differences between both views are evident.

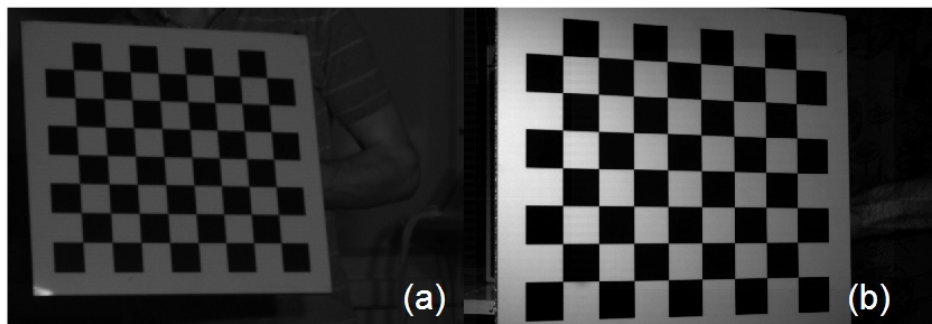


Figure 8.3: Single calibration view from L02 set S03 (a) and L03 set S04 (b).

From Figure 8.3 is possible to observe that L02 set S03 view's focus and illumination used to capture the video sequence set were poor, also, the view shows that the image area occupied by the calibration object is small when compared with the area occupied by L03 set S04 view (Figure 8.3 (b)). This analyse is corroborated with the results shown in Figure 8.4 where the camera's focal length values obtained for the same set (L02 set S03) do not converge for 100 views as in L01 set S03 and L03 set S04 cases, previously presented.

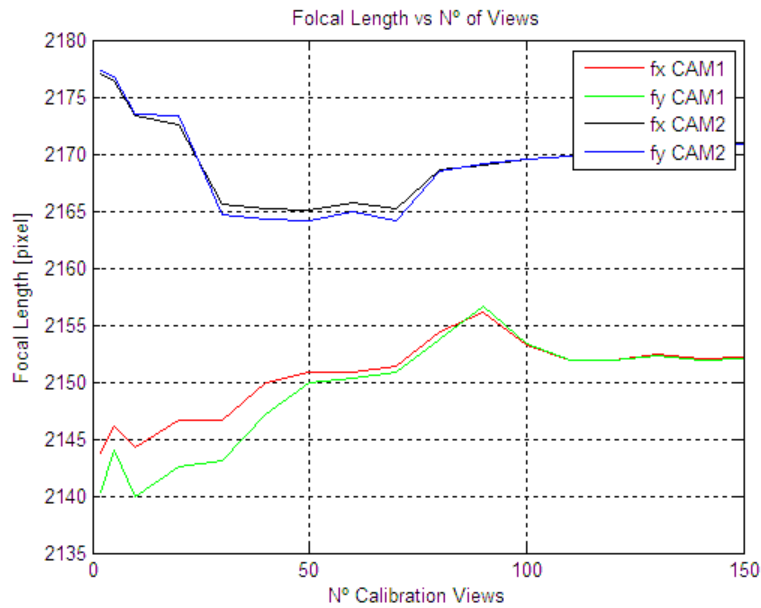


Figure 8.4: L02-S03 Focal Length vs N° of Calibration Views (M1 and M2).

Additionally to prove the object calibration's position and orientation influence on the calibration parameters estimation a list of calibration views from L03 set S04 video sequence was built intentionally with 150 consecutive images and the calibration process performed. Figure 8.5 shows the camera's focal length results obtained with this list of calibration views.

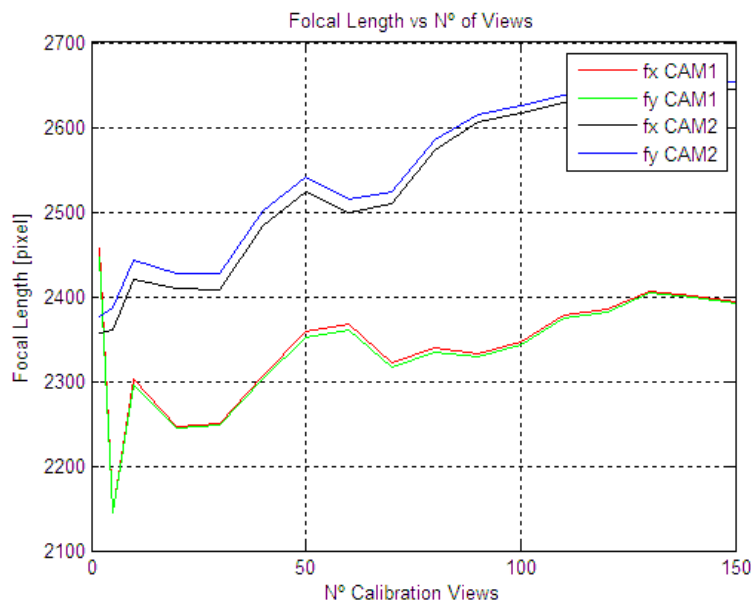


Figure 8.5: L03-S03 Focal Length vs N° of Calibration Views (M1 and M2).

From Figure 8.5 is possible to observe that calibration views with poor POSE, i.e. views with similar object's position and orientation, does not provide additional information to the equations involved on the calibration parameters and stereo relation estimation process. Adding similar views causes the algorithm to diverge from the solution, this is easily seen by comparing the plots in Figure 8.5 and in Figure 8.2.

To avoid bad calibration results the researcher used about 100 views from all the video sequence range, for example, the L03 set S04 stereo video sequences allowed to obtain 8131 calibration views thus for each 81 images (8131/100) only 1 image was considered, or in other words, only images multiple of 81 were considered for calibration purposes.

The number of calibration views necessary for the stereo calibration process is highly influenced by the calibration views quality such as image sharpness, how good the images were illuminated and the overall image area occupied by the calibration object. The variations of the calibration object's position and orientation are also important to obtain good results.

The optimal number of calibration views using stereo video sequences with satisfactory quality, such as L03 sets in which the video capturing settings and experiment procedure revealed to be the best from all three laboratories, was defined to 100 calibration views.

### 8.1.3.3 Calibration Parameters Optimization.

In order to improve the calibration parameters and stereo configuration relations a method was implemented to exclude the views with higher errors contributions. Figure 8.6 shows the implementation methodology using, without loss of generality, L03 set S04 calibration views as example.

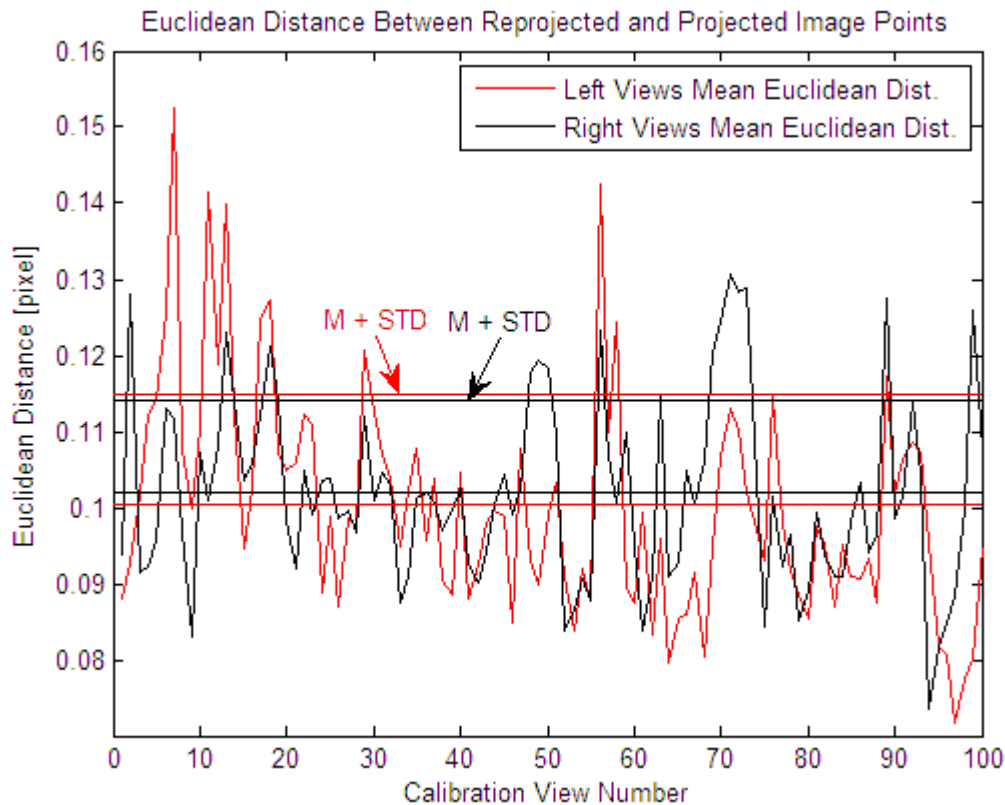


Figure 8.6: Calibration parameters optimization (Method M3).



The plots illustrated in Figure 8.6 shows the mean Euclidean distance between projected (built from calibration object points) and reprojected images points (retrieved from calibration views by detecting the chessboard corners) for each left and right view. The upper horizontal lines defined the acceptable interval  $[0; M+STD]$ , i.e. the mean of means plus the standard deviation, in which the calibration views were considered good for calibration purposes. Once the stereo configuration relations estimation required that the left and right images to be synchronized the smaller  $[0; M + STD]$  interval (in this case the right views interval) was used to filter both left and right calibration views, i.e. if one left(right) calibration view was excluded its right(left) corresponding view was also excluded to maintain the synchronization.

In the next table (Table 8.10) are presented the results obtained with calibration parameters optimization method (M3). The main value used to evaluate the improvements was the reprojection error (RE) given by OpenCV's calibration function ( `cv::calibrateCamera()` ) using the full set of calibration views (N° Views - Initial) and using the final number of views (views not excluded = N° Views (Final) ) .

Table 8.10

*Stereo Calibration Parameters Optimization Results (L01, L02, and L03)*

Lab-Stereo Config. Set	Calibration Views	Input/Output Calibration Parameters Optimization's Variables					
		Input Variables				Output Variables	
		N° Views (Initial)	Obj. Square Size [mm]	Corners [nx x ny]	RE(Initial) [pixel]	N° Views (Final)	RE (Final) [pixel]
L01 -S01	Left	100	30	40	38.88	74	26.67
	Right	100	30	40	37.63	74	25.30
L01 -S03	Left	100	40	56	2182.44	74	657.53
	Right	100	40	56	2085.77	74	618.58
L02 -S01	Left	100	40	56	34.88	87	16.78
	Right	100	40	56	44.99	87	20.71
L02 -S02	Left	100	40	56	197.59	91	128.83
	Right	100	40	56	106.16	91	85.56
L02 -S03	Left	100	40	56	63.53	73	18.95
	Right	100	40	56	78.49	73	25.87
L02 -S04	Left	100	10	64	185.28	81	89.29
	Right	100	10	64	190.57	81	98.10
L03 -S02	Left	100	30	40	55.39	76	36.38
	Right	100	30	40	54.27	76	35.50
L03 -S03	Left	100	30	40	55.15	86	38.73
	Right	100	30	40	60.08	86	42.54
L03 -S04	Left	100	40	56	33.71	81	19.35
	Right	100	40	56	1193.29	81	18.99

Note.

- The number of calibration views used to perform stereo calibration parameters optimization was exactly 100 views for all the sets.
- RE is the reprojection error (in pixel units) returned by OpenCV's single camera calibration function ( `cv::calibrateCamera()` ) for each left and right camera.

The reprojection error returned by the OpenCV single camera calibration function is obtained by computing the difference between the reprojected and project image points and then computes

the absolute norm of the resulting difference as formulated in eq. (8.1.1 and eq. (8.1.2.

The different between the projected and reprojected is given by the next formula:

$$distance(I) = projectedPoint(I) - reprojectedPoint(I) \quad (8.1.1)$$

And the final reprojection error is given by the absolute norm as follows:

$$reprojError = \sqrt{\sum_I distance(I)^2} \quad (8.1.2)$$

To better illustrate the reprojection errors a stereo configuration with two A4Tech cheap web cams were used to perform the stereo calibration with a 12-by-9 calibration pattern. In (Figure 8.7) the darker points(red) are the projected points and the brighter points (green) are the reprojected points that ideally would overlap each other meaning that the reprojection error was null.

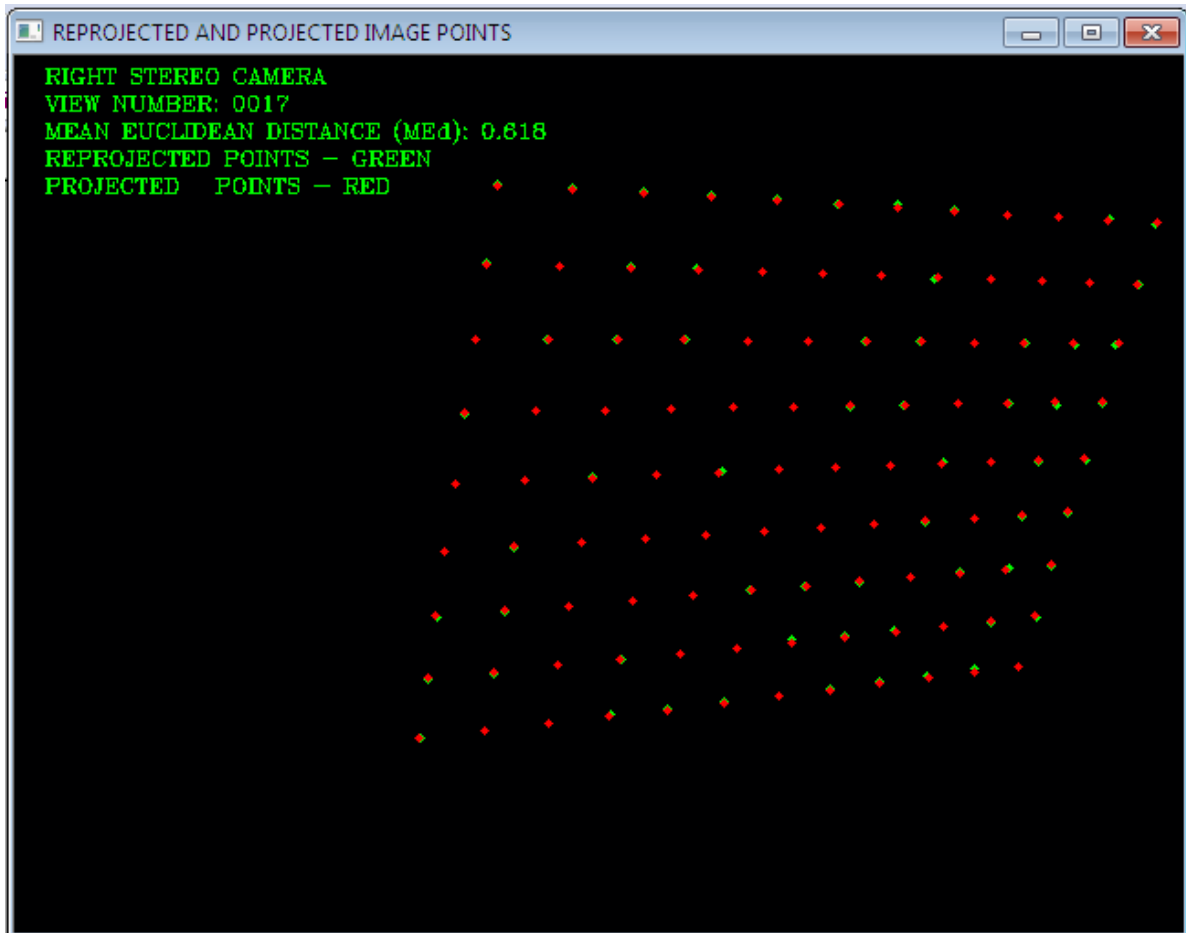


Figure 8.7: Calibration view's reprojection/projected image points.

By comparing both reprojection errors RE(Initial) and RE(Final) listed in Table 8.10 is possible to observe that the method implemented research improved significantly the calibration results, the reprojection errors obtained with M3 method for all the cases were minimized what allows to conclude that the calibration parameters were obtained with better precision. This approach allows to minimize the errors introduced by excluding calibration views with higher error and reduce the

number of images saving computational costs. Thus to compute the calibration parameters and stereo relations this case study used M1 together with M3 to ensure good results.

The next three tables ( Table 8.11, Table 8.12, and Table 8.13) present the calibration results obtained using all the data recorded for stereo calibration purposes during the three laboratory experiments. The calibration parameters were computed using all three calibration methods (M1, M2, and M3) , although, as mentioned before only the parameters resulting from M3 were used for the next stereo vision operations.

In general lines, the calibration method that allows to achieve better calibration results with less computational efforts is the calibration method M1. A number of 100 calibration views was defined as the optimal number of stereo views for calibration, however, was concluded that this number highly depends on the camera capturing settings and how the images were captured.

Additionally the calibration parameters optimization approach proved to be efficient on minimizing reprojection errors by excluding calibration views with higher errors contributions.

The next stereo vision operations used, as defined by the research findings, the output results obtained from calibration process M1 followed by M3 with 100 calibration views.

Table 8.11

*Laboratory L01's Camera Calibration Parameters (Methods M1, M2, and M3)*

Stereo Configuration	Calibration Method	Stereo Camera	Camera Intrinsic Parameters				Camera Extrinsic Parameters			
			Focal Length [mm]		Principal Point Coord. [pixel]		Radial Distortion Coeffs.		Tangential Distortion Coeffs.	
			fx	fy	cx	cy	k1	k2	p1	p2
S01	M1	CAM1	59.68	59.83	808.8753	600.0410	-0.0118	3.6606	-0.0097	-0.0130
		CAM2	60.38	60.55	788.6118	619.5446	-0.1979	7.4484	-0.0080	-0.0189
	M2	CAM1	59.10	59.18	848.6056	541.3404	0.1963	0.7019	-0.0117	-0.0004
		CAM2	59.86	59.69	996.1479	493.2986	0.2984	-0.1761	-0.0123	0.0226
	M3	CAM1	59.84	59.99	809.2514	593.2351	0.0136	3.4153	-0.0100	-0.0119
		CAM2	60.45	60.62	788.8335	615.2770	-0.1794	7.4277	-0.0078	-0.0181
S03	M1	CAM1	56.74	56.93	449.1972	479.7300	0.6711	-4.8720	0.0471	0.0020
		CAM2	56.90	57.20	469.0646	474.4410	0.9010	-12.7002	0.0558	-0.0029
	M2	CAM1	57.64	59.18	387.1714	780.6228	0.4407	4.4899	0.1034	-0.0105
		CAM2	58.30	60.75	455.1416	891.9835	0.5813	2.3352	0.1347	-0.0064
	M3	CAM1	55.85	56.24	430.1722	611.9272	0.5112	-3.2350	0.0507	-0.0035
		CAM2	56.31	56.79	459.8256	584.9412	0.8848	-12.6529	0.0629	-0.0032

Note. -For each stereo configuration (S02, S03, and S04) was performed the calibration process using all three methods M1, M2, and M3.

-The radial distortion parameter K3 is only used for wide-angle lenses, thus its value was set to zero for all the cases ( $k_3 = 0$ ).

-Focal length in [mm] units was obtained by multiplying focal length [pixel] by 0.0115 factor.

Table 8.12

*Laboratory L02's Camera Calibration Parameters (Methods M1, M2, and M3).*

Stereo Configuration	Calibration Method	Stereo Camera	Camera Intrinsic Parameters				Camera Extrinsic Parameters			
			Focal Length [mm]		Principal Point Coord. [pixel]		Radial Distortion Coeffs.		Tangential Distortion Coeffs.	
			fx	fy	cx	cy	k1	k2	p1	p2
S01	M1	CAM1	24.98	24.97	336.8382	261.2730	-0.0831	-1.7970	-0.0004	-0.0001
		CAM2	24.68	24.66	322.5725	255.9965	-0.0847	-0.8093	0.0016	-0.0010
	M2	CAM1	24.95	24.93	336.5431	260.2869	-0.0922	-1.0727	-0.0004	-0.0001
		CAM2	24.69	24.68	323.1908	253.6667	-0.0927	-0.6585	0.0013	-0.0013
	M3	CAM1	24.99	24.98	337.4393	268.4330	-0.1011	-0.9363	0.0007	0.0000
		CAM2	24.66	24.66	321.7637	249.9241	-0.1322	0.3174	0.0009	-0.0024
S02	M1	CAM1	23.37	23.32	288.8707	296.9317	0.0222	-1.5383	0.0018	-0.0008
		CAM2	24.32	24.35	318.2352	199.0930	-0.0455	-0.6280	-0.0093	-0.0008
	M2	CAM1	24.12	24.06	298.2936	275.6501	0.0703	-2.1384	0.0010	-0.0066
		CAM2	24.14	23.98	253.4934	213.6382	-0.0501	-1.4671	-0.0072	-0.0016
	M3	CAM1	23.85	23.83	309.1700	294.9672	0.0045	-1.3286	0.0023	-0.0014
		CAM2	24.28	24.32	316.8497	202.1186	-0.0423	-0.9497	-0.0087	-0.0010
S03	M1	CAM1	24.74	24.74	328.7818	246.9087	-0.0848	-0.3140	0.0010	0.0002
		CAM2	24.97	24.96	338.6817	266.0031	-0.0979	0.2569	0.0008	0.0005
	M2	CAM1	24.75	24.74	328.8060	247.2105	-0.0751	-0.4442	0.0013	0.0005
		CAM2	24.91	24.91	338.5166	265.5181	-0.1011	0.3365	0.0006	0.0005
	M3	CAM1	24.72	24.72	325.0836	249.0191	-0.0939	-0.1013	0.0011	-0.0004
		CAM2	24.95	24.95	337.8664	267.1120	-0.1001	0.2064	0.0008	0.0004
S04	M1	CAM1	37.04	37.09	356.8004	276.5897	-	-	-	-
		CAM2	35.94	35.73	273.7458	316.6878	-	-	-	-
	M2	CAM1	31.97	24.40	310.3517	274.5467	-	-	-	-
		CAM2	10.65	7.33	717.8691	318.3000	-	-	-	-
	M3	CAM1	37.23	37.31	370.7265	286.5862	-	-	-	-
		CAM2	35.80	35.62	263.6681	359.1597	-	-	-	-

Note. - For each stereo configuration (S02, S03, and S04) was performed the calibration process using all three methods M1, M2, and M3.

- The radial distortion parameter K3 is only used for wide-angle lenses, thus its value was set to zero for all the cases ( $k_3 = 0$ ).

- Focal length in [mm] units was obtained by multiplying focal length [pixel] by 0.0115 factor.

- A dash indicates that the solution for the extrinsic parameters did not converge and therefore were excluded from the table.

Table 8.13

Laboratory L03's Camera Calibration Parameters (Methods M1, M2, and M3).

Stereo Conf.	Calibration Method	Stereo Camera	Camera Intrinsic Parameters				Camera Extrinsic Parameters			
			Focal Length [mm]		Principal Point Coord. [pixel]		Radial Distortion Coeffs.		Tangential Distortion Coeffs.	
			fx	fy	cx	cy	k1	k2	p1	p2
S02	M1	CAM1	28.29	28.23	427.0194	407.6665	-0.0138	0.2953	0.0058	-0.0171
		CAM2	28.88	28.88	361.7643	398.8051	0.0719	-2.1130	0.0084	-0.0164
	M2	CAM1	28.22	28.17	433.5200	410.3129	0.0017	-0.0450	0.0060	-0.0157
		CAM2	28.49	28.50	373.6059	414.3428	0.0154	-0.5966	0.0084	-0.0167
	M3	CAM1	28.36	28.28	420.3281	401.6675	-0.0042	0.2235	0.0055	-0.0186
		CAM2	28.89	28.89	368.2153	388.8090	0.0735	-2.1017	0.0074	-0.0150
S03	M1	CAM1	28.15	28.15	450.7431	422.3490	-0.0611	0.2979	0.0065	-0.0100
		CAM2	28.49	28.48	435.1483	393.8991	-0.0654	0.5371	0.0059	-0.0108
	M2	CAM1	28.08	28.08	449.4412	428.3461	-0.0662	0.2814	0.0066	-0.0097
		CAM2	28.41	28.41	433.8292	406.7362	-0.0655	0.4295	0.0067	-0.0105
	M3	CAM1	28.14	28.14	443.2408	424.7560	-0.0496	0.1740	0.0070	-0.0114
		CAM2	28.50	28.48	418.5513	403.7597	-0.0524	0.3779	0.0071	-0.0134
S04	M1	CAM1	28.05	28.06	531.7591	377.4580	-0.0677	0.1012	0.0002	0.0004
		CAM2	28.51	28.52	499.2459	353.3265	-0.1146	0.9398	-0.0007	-0.0002
	M2	CAM1	28.08	28.08	532.1768	380.6691	-0.0679	0.0895	0.0007	0.0005
		CAM2	28.42	28.42	504.0630	355.3895	-0.1153	1.0618	-0.0005	-0.0003
	M3	CAM1	28.05	28.05	532.6571	378.2843	-0.0670	0.0783	0.0004	0.0005
		CAM2	28.38	28.37	523.9434	361.1249	-0.0600	0.0276	0.0006	0.0004

Note. -For each stereo configuration (S02, S03, and S04) was performed the calibration process using all three methods M1, M2, and M3.

-The radial distortion parameter K3 is only used for wide-angle lenses, thus its value was set to zero for all the cases ( $k_3 = 0$ ).

-Focal length in [mm] units was obtained by multiplying focal length [pixel] by 0.0115 factor.

### 8.1.4 Research Question N°3 – Results

The purpose of this research question was to compute the remapping maps and the disparity-to-depth matrix using two approaches: the uncalibrated stereo rectification based on Hartley's algorithm and the calibrated stereo rectification based on Bouguet's algorithm.

#### 8.1.4.1 Uncalibrated stereo rectification.

The uncalibrated method implementation revealed to be inefficient, the stereo video captures after being submitted to undistortion+remapping process, using the remapping maps obtained during uncalibrated rectification, lost all the 2D information needed to recover 3D data.

The rectification transformation matrices (rotation matrices)  $R_1$  and  $R_2$  obtained from the rectification homography matrices were the main error sources. To compute the fundamental matrix was considered that camera's distortion were significantly small and therefore were used the original image points without being submitted to undistortion, this may have introduced errors on the fundamental matrix estimation that in its turn was used to obtain  $H_1$  and  $H_2$ . To track the error source was verified the rotation matrices orthogonality for one case (L03 set S04), without loss of generality. The orthogonality results are presented next:

$$I = R_1 * R_1^T \Rightarrow I = 1.0e-003 \begin{bmatrix} 0.302934 & 0 & 0 \\ 0 & 0.418937 & 0 \\ 0 & 0 & 0.466325 \end{bmatrix}$$

$$I = R_2 * R_2^T \Rightarrow I = \begin{bmatrix} 1.109173 & 0 & 0 \\ 0 & 0.993053 & 0 \\ 0 & 0 & 0.897953 \end{bmatrix}$$

From the identity matrix (I) resulting from the orthogonality condition was possible to observe that while the right rotation matrix ( $R_2$ ) had small errors (I's main diagonal values close to 1) the left rotation matrix ( $R_1$ ) was not orthogonal as expected justifying the bad results obtained for the uncalibrated process.

#### 8.1.4.2 Calibrated stereo rectification.

The calibrated rectification method was the method that revealed to be very efficient, the next figures (Figure 8.8 and Figure 8.9) shows the stereo video capture without rectification and with stereo rectification, respectively.



Figure 8.8: Stereo video capture without rectification.

The drawback of performing the rectification is mainly connected with the fact that some parts of the image that may contain important information are transformed into outliers (grey area Figure 8.9) or are cropped as the images shifted towards each other.

From the next figure (Figure 8.9) was possible to observe that after both left and right stereo video captures were submitted to undistortion+rectification they became row aligned as expected, this allows also to conclude that the calibration/rectification maps were correctly estimated. To provide a comparison with the uncalibrated method the rotation matrix orthogonality was verified for R1 and R2 as follows:

$$I = R_1 * R_1^T \Rightarrow I = \begin{bmatrix} 0.977732 & -0.003397 & -0.018870 \\ -0.003397 & 0.996585 & 0.000016 \\ -0.018870 & 0.000016 & 0.981113 \end{bmatrix}$$

$$I = R_2 * R_2^T \Rightarrow I = \begin{bmatrix} 0.935669 & -0.002870 & -0.061460 \\ -0.002870 & 0.997084 & 0.000045 \\ -0.061460 & 0.000045 & 0.938494 \end{bmatrix}$$

From both I matrices was possible to conclude that, contrary to the uncalibrated methodology, R1 and R2 are orthogonal i.e. I matrices obtained were very close to identity matrices form as required for any three-dimensional rotation matrix.

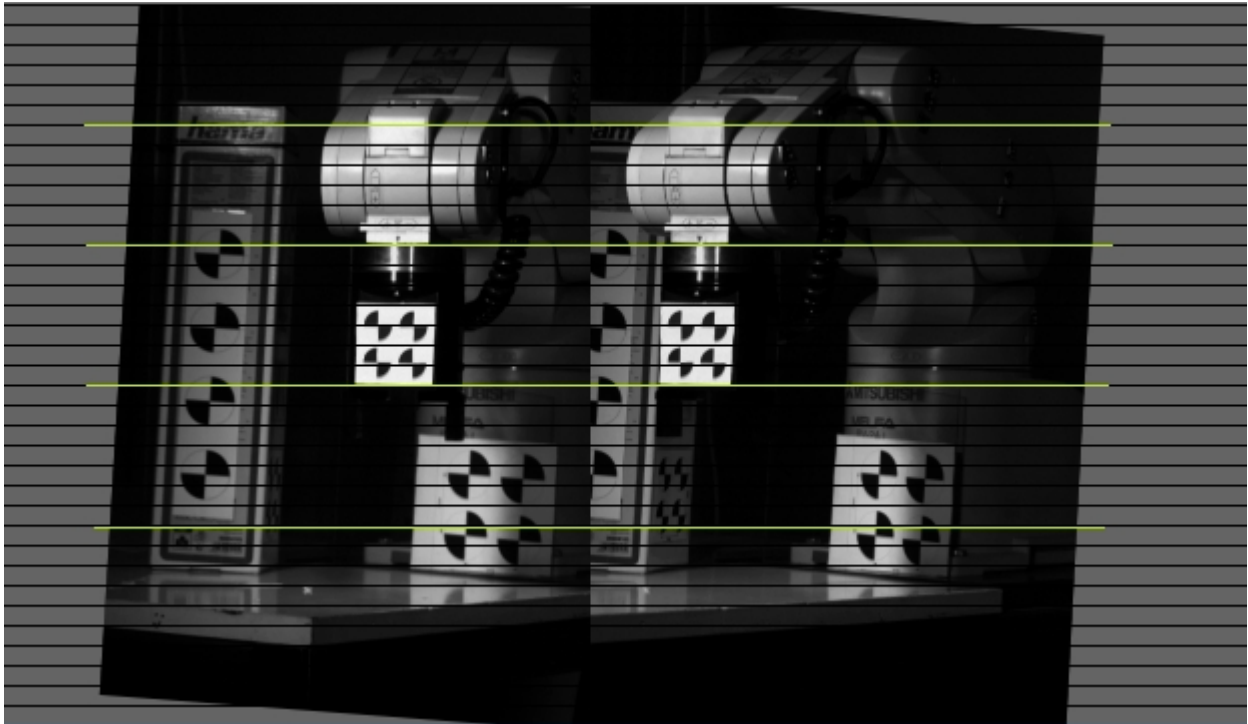


Figure 8.9: Stereo video capture after calibrated stereo rectification.

### 8.1.5 Research Question N°4 – Results

In order to answer this research question were used the output results obtained after stereo calibration parameters optimization (calibration method M3). The stereo configuration measurements taken during the three laboratory sessions and the stereo configuration relations obtained from calibration process, rotation matrix R that brings the right camera to the left camera's



orientation and the translation vector  $T$  that brings the right camera to the left camera's position, are summarized in the next tables (see Table 8.14 and Table 8.15).

Table 8.14

*Laboratory Stereo Configuration's Measurements*

Lab-Stereo Configuration	Distance Between Stereo Configuration's Cameras. [mm]	Distance Between Stereo Config. and MELFA robot.[mm]
L01-S01	(410)	3080.00
L01-S03	(410)	3080.00
L02-S01	(445)	1850.00
L02-S02	(445) – 50	2230.00
L02-S03	(445) + 50	2230.00
L02-S04	(445) + 30	2230.00
L03-S02	(390) + 90	1930.00
L03-S03	(390) + 90	1930.00
L03-S04	(390) + 115	2080.00

Note. The dimensions inside brackets were obtained by measuring the distance between each camera's lens center, this measure only gives an approximation once the real distance between camera sensors is not possible to obtain with direct measurements. The second distances were set by using the Manfrotto sliding plate.

Table 8.15

*Stereo Configuration's Relations*

Lab-Stereo Configuration	Euler Angles [degrees]			Quaternion Components				Translation [mm]		
	$\Phi$	$\Theta$	$\Psi$	q1	q2	q3	q4	Tx	Ty	Tz
L01-S01	-0.78	-6.05	1.24	-0.0105	0.0528	0.0062	0.9985	379.18	3.91	135.82
L01-S03	-0.95	-6.33	-0.11	0.0014	0.0552	0.0083	0.9984	372.67	5.96	95.84
L02-S01	-0.94	-12.68	0.17	-0.0006	0.1105	0.0080	0.9938	409.05	8.64	46.69
L02-S02	-0.58	-13.26	-1.91	0.0172	0.1154	0.0069	0.9931	327.19	8.90	92.87
L02-S03	0.61	11.73	-0.40	0.0040	-0.1022	-0.0057	0.9947	-456.94	-12.42	4.15
L02-S04	-5.64	44.76	-7.62	0.0427	-0.3825	0.0201	0.9227	-1049.69	-119.91	337.96
L03-S02	0.61	12.61	0.68	-0.0053	-0.1099	-0.0046	0.9939	-463.06	22.86	26.20
L03-S03	0.80	18.44	0.65	-0.0045	-0.1602	-0.0059	0.9871	-438.73	23.31	32.67
L03-S04	0.38	22.25	0.71	-0.0054	-0.1929	-0.0021	0.9812	-433.08	23.99	110.99

Note. Euler angles and Quaternion components were obtained by parametrizing the stereo configuration rotation matrix. The Euler angles represents the rotation matrix that brings the right camera to the left camera orientation. Translation represents the translation vector that bring the right camera to the left camera position.

From Table 8.14 and Table 8.15 was possible to make few observations.

- For some stereo configuration each left and right camera had different sample rate (fps) as in the case of L01-S01 set. The stereo configuration relations estimation proved to give different results when compared with the case where the cameras, using exactly the same stereo configuration, were synchronized as in L01-S03 set (see Table 8.1 for video files capture settings).
- From L02-S04 stereo video set is possible to observe that higher rotation angles between cameras introduced higher errors on the translation vector estimation.
- In the case of the sets were the distances between cameras were kept unchanged, as the case of L03-S02 and L03-S03 sets, was possible to observe that by only changing the cameras

orientations caused the translation variables ( $T_x$ ,  $T_y$ ,  $T_z$ ) to change.

### 8.1.6 Research Question N°5 – Results

In order to answer this research question two approaches were implemented. To compute the dense disparity image from a (undistorted + rectified) pair of stereo video captures were used the OpenCV stereo Block Matching, Semi Global Block Matching or the non real-time Graph-Cut Matching algorithm. The dense disparity results obtained using dense stereo matching methods did not provide good disparity results and the method proved to be unreliable, with the given stereo video sequences, to obtain the depth for all the image points. In the next figure (see Figure 8.10) is presented the disparity image obtained from stereo matching with L03-S02 stereo video capture.



Figure 8.10: Stereo matching using Block-Matching algorithm.

As shown in Figure 8.10 the disparity image obtained by using the stereo Block Matching algorithm with default settings available from OpenCV libraries results in a great number of outliers i.e. pixel disparity values which are outside an interval defined by [min disparity; max disparity].

Alternatively, using the sparse tracking techniques approach with pyramid Lucas-Kanade tracker code sparse sets of 2D points were projected to 3D space and then compared with the MELFA robot's 3D path.

The following figure (Figure 8.11) shows the 3D path programmed with MELFA RV-2AJ software with a sampling time equal to 50 ms.

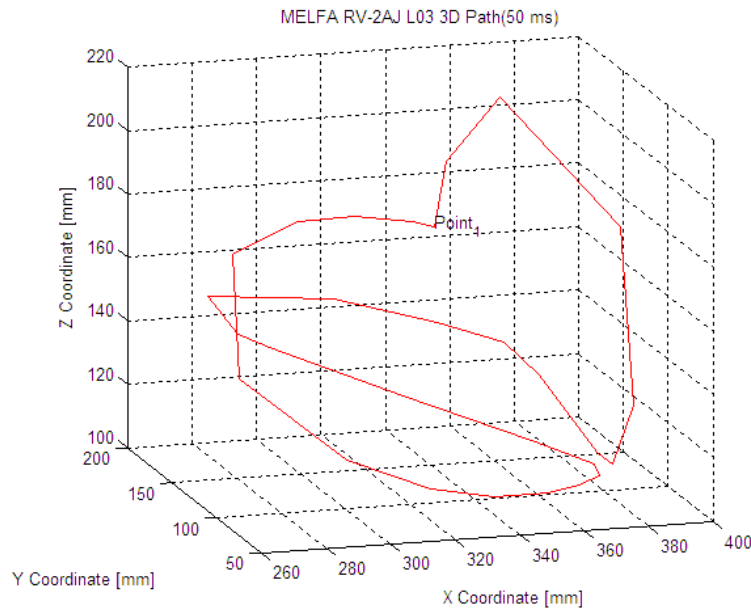


Figure 8.11: MELFA RV-2AJ 3D Path (sampling time: 50ms).

The next figure (Figure 8.12) contains the plot of the same 3D path obtained with a better sampling time of 10 ms. This path was the path taken in consideration for further analysis.

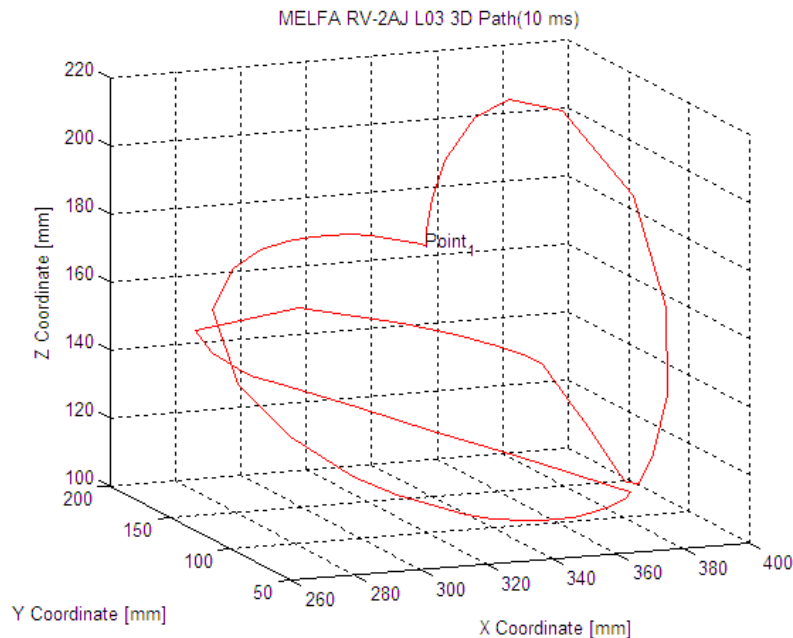


Figure 8.12: Figure 8.8: MELFA RV-2AJ 3D Path (sampling time: 10ms).

The next figure (Figure 8.13) shows the 3D path recovered using the sparse matching method implemented on this research. The two paths (green, red) are related to each camera coordinates system, the green path is the 3D path seen by the stereo configuration's left camera and the red is the 3D path related to the stereo configuration's right camera coordinate system.

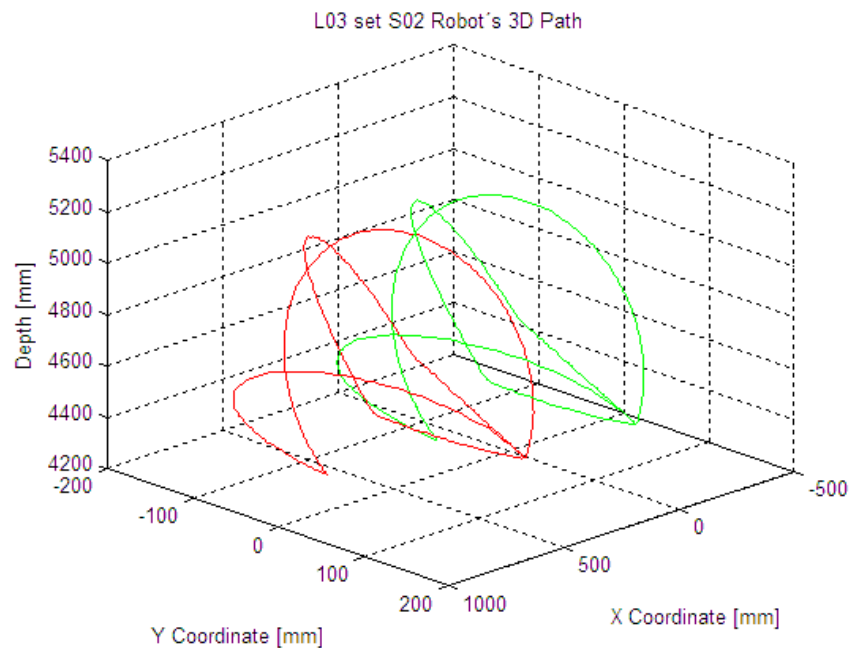


Figure 8.13: MELFA robot's recovered 3D path(camera coordinates system).

To obtain the same path as in the previous figure (Figure 8.13) on the MELFA robot's coordinate system all the points were transformed using the point-line-plane approach as presented on the methodology section.

The next figure (Figure 8.14) presents the resulting 3D path after being transformed with the rotation and translation that bring the camera coordinate system to the MELFA robot system as defined in the methods section (see Posttest. from Methods section for more details). Additionally, using the MELFA RV-2AJ robot's external dimensions, was performed a pure translations to relate the points into robot's coordinate system origin.

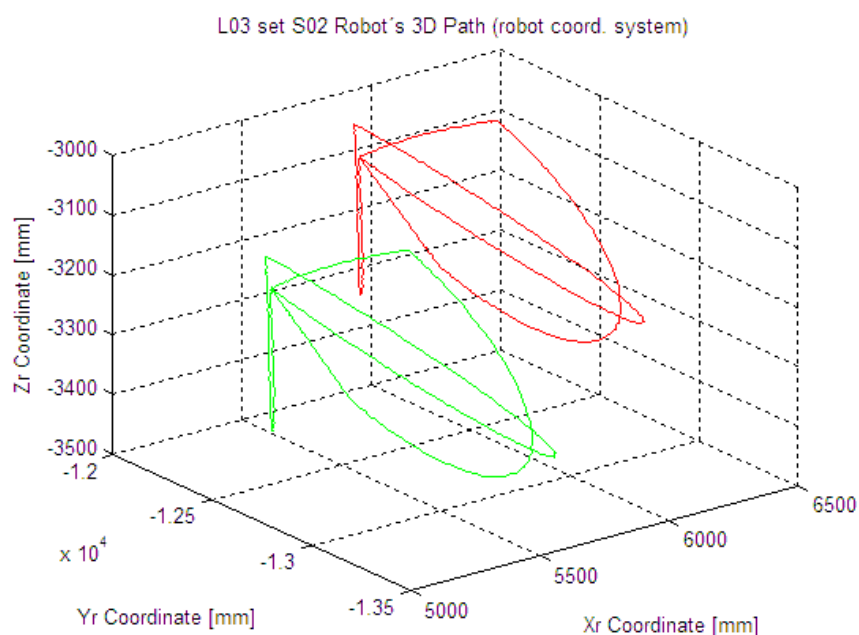


Figure 8.14: MELFA robot's recovered 3D path (robot coordinate system).

To test the point-line-plane coordinate system transformation were used a generic set of points defined in the 3D space (  $P_0=[100 \ 100 \ 1000]$  ,  $P_1=[200 \ 100 \ 1000]$  ,  $P_{yz}=[100 \ 300 \ 1100]$  ) to build the direction cosines matrix and translation vector. The next figure ( see Figure 8.15) is displayed the same 3D paths as in Figure 8.13 transformed into the generic coordinate system.

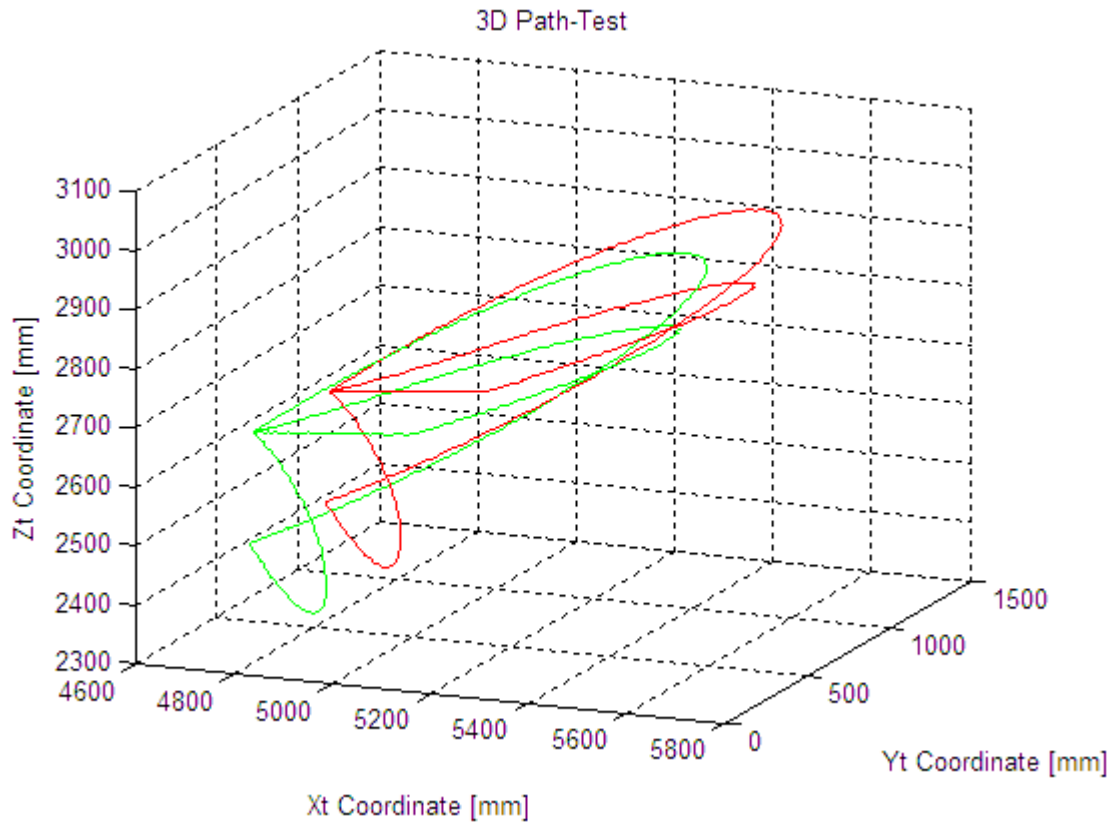


Figure 8.15: MELFA robot's recovered 3D path (generic coordinate system).

From the results obtained was possible to make different observations. The final 3D path shape obtained with the Lucas-Kanade sparse stereo matching approach presents similarities with the path obtained from MELFA software. However, the 3D points recovering process has errors resulting from the 2D to 3D reprojection process, the reprojection errors were amplified when the 3D points were transformed from camera's coordinate system to robot coordinate system due the fact that the computed 3D points were used in the point-line-plane implementation to obtain the transformation matrix (rotation and translation). This observation was easily verified by projecting the 3D path in camera's coordinates, Figure 8.13, to a generic coordinate system defined in the 3D space by three arbitrary points, the transforming test is shown in Figure 8.15.

## 9 Chapter Five

### 9.1 Discussion

#### 9.1.1 Introduction

To recover 3D information from stereo video sequences three main processes are required. The first and main process allows to compute the cameras calibration parameters and stereo configuration relations estimation, the results obtained from this operation determined how precise the following operation's results are obtained. The second process consists on implementing the (uncalibrated/calibrated) stereo rectification process to produce the (undistortion+rectification) maps necessary to correct image's radial and tangential distortion and transform the stereo captures so they became row-aligned and coplanar as if they were captured from a canonical stereo configuration. From rectification process is also obtained the disparity-to-depth transformation matrix, this matrix is the most important element on the process of transforming 2D image points to 3D space. To recover 3D information is necessary to perform the stereo matching operation to match the left image points with the right image points in such a way that both image points correspond to the same object's point in the scene.

This case study utilized the OpenCV image processing platform together with Microsoft Visual Studio 2008 software to implement a program for recovering 3D points from video sequences captured with two Phantom v9.1 high-speed cameras on a stereo configuration, it utilized two pretest laboratory sessions and one intervention laboratory session. Measurements included building different stereo configurations with two Phantom v9.1 high-speed cameras to capture video sequences of a MELFA RV-2AJ robot executing a simple 3D path, and additionally capture video sequences of a planar calibration object to calibrate each stereo configuration.

To perform the stereo cameras calibration and stereo relations estimation two methods were implemented to determine which allowed to obtained better results, additionally, a third option was implemented to optimize the calibration parameters results. To rectify the stereo video captures the calibrated and uncalibrated rectification methods were implemented. For the last operation, stereo matching process, were implemented two approaches, the first approach allows to compute the dense disparity image using OpenCV stereo matching algorithms, and for the second approach a sparse matching method was implemented. The sparse matching method made use of Lucas-Kanade Pyramid optical flow method to match sparse set of points and reproject them to 3D space.

#### 9.1.2 Discussion

##### 9.1.2.1 Research question nr. 1 – discussion.

Research question: *Which are the OpenCV main functions involved in the process of: stereo camera calibration, stereo image rectification, stereo matching and points reprojection into 3D space, and Lucas – Kanade Pyramid optical flow method. What are the inputs and outputs arguments of those functions.*

As mentioned in the previous chapters, the answer to this question constitutes by itself a review of the theory under the OpenCV main algorithms used to implement the researcher-made program implemented for this case study. The Appendices A and F presents the answer to this question( see Appendix A: Stereo Imaging and Appendix F: Motion).

The review was done based on the book from Bradski et al. (2008) to provide a better understanding and broader knowledge of the functions available from OpenCV.

The researcher started by using the old C programming language platform to implement the main program however after few programming attempts using the new OpenCV 2.1 C++ approach it was evident that the new OpenCV C++ platform had better performance, less memory leakage, specially when dealing with real-time video operations, and less programming once all the memory deallocation were performed automatically, furthermore the C++ STL's containers and iterators made easy to work with image and points storing operations allowing the researcher to focus more his attention on other important issues during the research.

### 9.1.2.2 Research question nr. 2 – discussion.

Research question: *How to compute camera calibration parameters using a planar calibration object known as chessboard and how to relate two cameras in a stereo configuration. How many calibration views are needed to perform the stereo calibration process and which calibration method (with and without initial guess to compute stereo relations) gives better results. How to optimize the stereo calibration process and improve the calibration parameters results.*

To answer to the proposed research questions were implemented different functionalities that allowed to perform the calibration process given different inputs: a text file with a list of calibration views, stereo avi files for calibration purposes, or real time video capture. To compute the calibration parameters were implemented two methods (M1 and M2) and additionally was implemented a method to improve the calibration results.

### Calibration method.

The calibration study based on two main calibration parameters  $f_x$  and  $k_2$  obtained by performing the stereo calibration by using calibration methods M1 and M2 with 17 distinct groups of calibration views was designed to determine which calibration method provided the best approach to compute the calibration parameters and stereo configuration relations. In the next table (see Table 9.1) is presented a summary of the standard deviation values obtained from calibration method study.

Table 9.1

*Calibration Method's Study Summary*

Lab – Stereo Configuration	Comparison Values	Calibration Method			
		M1		M2	
		CAM1	CAM2	CAM1	CAM2
L01-S03	STD( $f_x$ )	4.75	11.13	4.16	10.12
	STD( $k_2$ )	8.46	33.23	11.50	29.50
L02-S03	STD( $f_x$ )	1.13	0.21	0.75	0.26
	STD( $k_2$ )	2.23	8.02	1.93	7.89
L03-S04	STD( $f_x$ )	0.27	0.68	0.32	0.40
	STD( $k_2$ )	0.09	1.16	0.16	1.65

Note. STD is the standard deviation values for each  $f_x$ ,  $k_2$  parameter from Table 10.7, Table 10.8, and Table 10.9.

The standard deviation results obtained with calibration method M2 revealed to be slightly smaller comparatively with M1 for the first two calibration sets (L01-S03 and L02-S03). Better results were obtained for both cameras (CAM1 and CAM2) using method M1 with the data from the third laboratory (set L03-S04), this result may have been related to the fact that the data collected during the third laboratory were better for calibration purposes. In addition, this conclusion was consolidated with the literature that suggests method M1 is more robust than method M2, moreover M1 revealed to be less computationally demanding and time consuming, being the method used for the research. Also was possible to observe that parameters obtained from one of the cameras (CAM2 in general) had higher standard deviations than the other camera, this fact could have been a result of dust detected in one of the camera's sensor after carefully re-analysing the video captures.

From Table 9.1 is possible to verify that the reduction of standard deviations values for both calibration parameters were statistically significant from pretest (L01 and L02) to posttest laboratory sessions (L03), this significant gains could have been a result of different improvements done along the three laboratory sessions and other considerations taken in account for the third laboratory:

- Lightning conditions were improved by using proper LED's arrays for high-speed framing.
- Both Phantom v.91 camera's lens was correctly focused on a focal-plane from where the sequence of calibration views were captured.
- The lens's maximum and minimum focal length range, where the distortions are more visible, were avoided from being used.
- Low frame rates values (90fps) instead of higher values were used to allow more recording time and capture calibration views with rich positions and orientations, i.e. object calibration images which the position and orientation varies considerably from frame capture to frame capture.
- Smaller image resolution (960x720) were used to capture video sequences with Phantom v. 9.1 camera's avoiding the higher distortion that are more notable in the sensor corners (higher image resolutions).
- CSR PCC's software feature was used to compute the pixel offset for the current frame rate, resolution and exposure settings, giving a more precise compensation of the pixel errors and better calibration results.

#### **Optimal number of calibration views.**

The literature suggests that a number of calibration views between 30 and 100 should be used to calibrate a stereo configuration. To determine this value was performed the stereo camera calibration using different number of calibration views (02 to 150) and studied how the focal length parameter behaved. While for some stereo configurations the focal length values were acceptably stable for a number of 100 calibration views (see Figure 8.1 and Figure 8.2) for others stereo configurations (see Figure 8.4 and Figure 8.5) the same parameters did not converge this may have been related to factors such as the following ones:

- Calibration parameters converge faster for calibration views with higher quality, i.e. image focus, illumination conditions and area percent occupied by the calibration object in the calibration view. Figure 8.3 shows two calibration views from L02-S03 and L04-S04 where the image quality differences between both views are evident.
- Poor calibration information. i.e. calibration views with similar position and orientations do not add any additional information but instead it may cause the algorithms to diverge from the real solution as shown in Figure 8.5.



- Focal length values computed with calibration method M1 converged faster with less variations than the values obtained with calibration method M2.

The number of calibration views necessary for the stereo calibration process is highly influenced by the calibration views quality such as image sharpness, and how good the images were illuminated. The variations of the calibration object's position and orientation also are important to obtain good results.

Factors such as the area occupied by the calibration object in the image were difficult to control once the angle of convergence and the horizontal distance between the two stereo configuration's cameras made difficult the task of covering all the image resolution with the calibration object on both cameras at the same time to keep the synchronisation between cameras as required for the stereo configuration relations estimation.

### **Calibration parameters optimization.**

To improve the calibration parameters was implemented a method that based on the difference between the reprojected and project calibration image points allowed to optimize the calibration parameters results.

In Table 8.10 are presented the results obtained for the reprojection error (RE) before and after calibration optimization process (calibration method M3) for almost all the sets were obtained reprojection errors 30% smaller than initially verified, this improvements could have been a result of the methodology implemented that allowed to eliminate the views with higher error contributions and recompute the new parameters with only good calibration views.

Higher reprojection errors were verified for views in which the angle formed between the camera imager plane and the the calibration object plane were relatively big, i.e. angles higher than 60 degrees, however this value may only be used as a reference as this case study did not studied the influence of calibration object's orientations on the calibration parameters results.

Was verified that for calibration views with high luminosity gradients or defocused images, despite of the algorithms did not failed to find all the calibration object corners it failed to find the proper corners identifying wrong points as being corners locations. To avoid this cases the detected corners were drawn over the calibration views to help to visualize if the corners were correctly detected.

#### **9.1.2.3 Research question nr. 3 – discussion.**

Research question: *Which are the differences between using calibrated and uncalibrated rectification methods and how to implement the image rectification process by means of using OpenCV functions.*

To answer this research question were implemented stereo rectification methods to compute the remapping maps and the disparity-to-depth matrix using two approaches: the uncalibrated stereo rectification - Hartley's method based on the procedure described by Bradski et al. (2008) and the calibrated stereo rectification available from OpenCV algorithms based on Bouguet's algorithm.

While the results produced by the calibrated method (see Figure 8.9) were satisfactory the same results were not verified for the uncalibrated stereo rectification. The resulting images obtained after applying the remapping maps (undistortion+rectification) lost all its original information, this wrong outputs could have been a result of systematic errors by using directly the detected image points from calibration to compute the fundamental matrix used to obtain the rectification transformation matrices (rotation matrices) R1 and R2. In addition, to corroborate the results, both

rotations matrices were subjected to orthogonality test, from the identity matrix resulting from the orthogonality condition was possible to observe that while the identity matrix (using R2) had its diagonal values close to 1, the orthogonality condition was not verified for R1 matrix as expected justifying the bad results obtained for the uncalibrated process.

#### **9.1.2.4      *Research question nr. 4 – discussion.***

Research question: *How to parametrize the stereo relation's rotation matrix into Euler angles and quaternions and how to perform the transformation between this two rotation representations.*

To answer this research question were implemented two algorithms that allowed to transform a rotation matrix to Euler angles and quaternion. To study the rotation angles' influence on the right camera position estimation the stereo configuration translation vector T estimations were also included in the same table (see Table 8.15).

For some stereo configuration, as the case of L01-S01 and L01-S03 sets, that used exactly the same stereo configuration (position and orientation), different results were obtained. The difference between the stereo relations estimations may have been related to the fact that the video sequences for calibration recorded with different sample rates (asynchronous cameras' capture) introduces errors on the stereo relations estimation.

The stereo relation's rotation matrix parametrization into Euler angles helped to visualize which angles had higher contributions for the rotation. Rotation angles around YY axis (on camera axis) had higher values what could have been a result of setting the angles between cameras' optical axis in such a way that the cameras' FOV area overlapped was maximized.

The stereo configuration's translation vector had better estimations when smaller angles between cameras' optical axis were used to arrange the configuration, this could have been related to the fact that OpenCV stereo calibration algorithm requires the configuration to be as close to a canonical stereo configuration as possible.

In some cases were the distances between cameras were kept unchanged between new stereo configurations, as the case of L03-S02 and L03-S03 sets, was observed that by only changing the cameras' orientation caused the stereo configuration's translation variables ( $T_x$ ,  $T_y$ ,  $T_z$ ) to change, this may have been related to the fact that the cameras' rotation axis do not correspond to the same cameras' center of projection (or camera's coordinate system origin).

#### **9.1.2.5      *Research question nr. 5 – discussion.***

Research question: *How to compute the disparity image and disparity of a sparse set of points given two rectified images captured from a stereo configuration previously calibrated. How to reproject a sparse set of points to the 3D space.*

To answer this research question two approaches were initially implemented. By using the calibration parameters and the rectification maps given by the previous research questions implementations the video sequences were remapped and matched using the dense and sparse stereo matching methods.

Although good results were obtained from the rectification process when remapping the left and right video capture sequences the results obtained using one of the OpenCV stereo matching algorithms (Block Matching, Semi Global Block Matching, or Horn-Schunck) did not provide good disparity images, only some parts of the 3D scene were recovered and the stereo matching algorithms failed to fit the object's silhouettes accurately. The disparity images inaccuracy may have been related to the block matching settings used to perform the stereo-matching process and

to the high convergence angles values used between the two cameras' optical axis in the stereo configurations.

The sparse stereo matching approach using Lucas-Kanade Pyramid tracker to track sparse set of points and perform the stereo matching approach allowed to recover 3D points without the need of working with all image resolution. To corroborate the results were tracked points over an end-effector of a robot executing a known 3D path, the 3D path obtained by the stereo vision approach proved to be similar to the one executed by the end-effector, however, the direct comparison between distances obtained with the computed 3D points and the real object's distances showed that the 2D images points reprojection into 3D space had errors associated, this errors are minimal for the standard stereo configurations that were arranged closer to a canonical stereo configuration and increase for the stereo configurations with higher angles between cameras' optical axes. The reprojection errors may have been related with the wrong stereo configuration relations estimations namely the horizontal distance ( $T_x$ ) used in the disparity-to-depth reprojection. Also due the fact that the remapping process cropped parts of the image necessary to recover all 3D path the captures were only subjected to undistortion and this procedure could have been the major source of errors in the points disparity computation process.

The 3D points transformation from cameras' coordinates system into the MELFA robot's coordinate system using the point-line-plane approach increased the 3D points error, this could have been a result of using the computed 3D points, already affected with by reprojection errors, to build the transformation's direction cosine matrix and translation vector, this was easily proved by performing a transformation (rotation+translation) using three points selected arbitrarily in the 3D space.

### **9.1.3 Limitations**

Although the two pretest laboratory sessions helped to eliminate procedure errors and determine better capturing settings and video recording conditions for camera calibration and 3D information recovering processes there were several limitations to the study.

The first limitation was related with the great number of variables involved in the video capturing process such as luminosity, camera capture settings, video synchronisation, stereo cameras position and orientation, the distance between cameras and the distance between the stereo configuration and the scene being recorded. Having such amount of variables made difficult to determine the exact source of the stereo configuration relations estimation's errors and the disparity-to-depth reprojection's errors, moreover it made difficult to establish a direct comparison between different stereo video sequence sets either for calibration or for 3D points recovering process. Therefore, this limitation makes the results difficult to generalize to others stereo video sequences either for stereo calibration purposes or to depth recovering process.

The second limitations is related with the dense stereo matching method, during the research development the dense approach proved to be inefficient with the stereo video sequences recorded during the laboratory sessions. Furthermore the dense stereo matching was not tested with stereo video sequences from other sources to determine if the Block Matching (or Semi Global Block Matching) settings were the optimal settings used or, in the other hand, if the research's stereo video sequences were the main cause for the inaccurate results.

Other limitations were related with the intervention. Each laboratory session required a minimum of six hours to set the stereo configuration and capture an average of four stereo video sequences for stereo calibration and four video sequences for 3D information recovering purposes, moreover, the laboratory schedule was very filled by classes what reduced the number of laboratories that could have been used to improve the results obtained.

### **9.1.4 Recommendations for Future Research**

Based on the results of this case study, there are several recommendations for the future research. In the future research the number of independent variables need to be reduced by firstly determining the optimal illumination conditions and camera capturing settings such as focal plane, frame rate, exposure time, and camera synchronization settings. Also a standard procedure need to be well defined since the beginning to help to reproduce the same conditions between experiments, this will reduce the differences between stereo configurations and make easier to generalize the results obtained with different stereo configurations.

To test the stereo matching process with the obtained stereo video sequences others stereo matching algorithms, such as the new matching techniques available from Middlebury evaluation site at <http://vision.middlebury.edu/stereo/eval/>, should be included in the future implementations, this could help to determine if the inaccurate disparity images obtained with OpenCV matching algorithms were caused by selecting wrong matching settings or by the stereo video sequences used, moreover, those algorithms represent the newest advances in stereo matching techniques and could help to improve the speed and computational costs with real time operations. In what concerns to the stereo configuration, during this research the cameras' orientation angles and the distance between cameras were both simultaneously changed what made difficult to establish which had higher impact on the stereo rectification process, also was found that by simply changing the cameras rotations the distance between cameras' axis changed. Future research should set the cameras' orientation to constant and only change the distance between cameras on the stereo configuration, this will allow to take more general conclusions and determine more accurately which configuration produces the best results.

### **9.1.5 Conclusions**

Various conclusions can be made from this study. The first main conclusion is that calibration process plays an important role in the process of recovering 3D information from stereo images and it defines how good and how accurate the 3D information is obtained. The study showed that the quality of the results obtained from calibration process depends highly on the capturing conditions and video recording settings. Providing proper illumination for high-speed framing is crucial to obtain good results, also the study showed that the calibration results improved substantial from the first two laboratories to the third laboratory where the images were obtained with a better focusing during the procedure. The study shows that better calibration results were obtained for the calibration views where the calibration object covered almost all the image instead of a small area, also the study shows that when a set of calibration views with similar position and orientation are used the calibration algorithm diverge from the solution, better results were obtained for sets of images with different positions and orientation. Still related to cameras calibration the study determined that calibration method M1 provides a more efficient and robust method to calibrated the cameras and estimate the stereo relations.

The second conclusions is that by using the difference between the reprojected and projected image points was possible to establish a method that allowed to exclude calibration views with higher errors contributions and recompute the new calibration parameters reducing the reprojection errors by 30% in almost all of the cases.

Related with the stereo rectification was concluded that the uncalibrated stereo rectification do not provide good results and instead the calibrated stereo rectification method should always be chosen to obtain the correct undistortion and rectification maps, and the disparity-to-depth reprojection matrix as well.

The third main conclusions is related with the stereo matching process. The stereo matching step revealed to be one of the stereo vision's most challenging tasks to implement. The dense stereo

matching process using OpenCV matching algorithms did not provide accurate disparity images and further research must be done. Alternatively was implemented a second approach using Lucas-Kanade tracker that can efficiently match sparse set of points selected by the user between two stereo captures. In general better results were obtained for the stereo configuration arranged closer to a canonical stereo configuration, with smaller angles between stereo cameras' optical axis. It was also concluded that the disparity-to-depth reprojection depends highly on the disparity values and on the distance between cameras ( $T_x$ ) estimated by the stereo calibration process. Besides the calibration can be performed individually for each camera, to obtain correct stereo relations the stereo configuration cameras' synchronization need to be ensured. The 3D path obtained with the StereoVisionProg implementation proved to be similar to the real path given by MELFA robot's software and the approach with Lucas-Kanade Pyramid tracker proved to be reliable and viable to recover 3D information for a sparse set of points.

## 10 References

- Ayache, N. & Hansen, C. (1988). Rectification of images for binocular and trinocular stereovision. *Rapport de Recherche*, N°860, pp. 9-17.
- Ayache, N. & Lustman, F. (1991). Trinocular stereo vision for robotics. *IEEE Transactions on PAMI*, Volume 13, pp. 73-85.
- Bradski G. R., Kaehler, A. (2008). *Learning OpenCV: Computer Vision with the OpenCV Library*, 1st ed. Sebastopol, CA: O'Reilly Media Inc.
- Cyganek, B., & Siebert, J.P. (2009). *An introduction to 3D computer vision techniques and algorithms*. West Sussex, PO19 8SQ, United Kingdom: John Wiley & Sons Ltd.
- Fusiello, A., Trucco, E., & Verri, A. (2000). A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, Volume 12, pp. 16–22.
- Hartley, R. I. (1999). Theory and practice of projective rectification. *International Journal of Computer Vision*, Volume 35, pp. 115-127.
- Heikkila, J., & Silven, O. (1997). A four-step camera calibration procedure with implicit image correction. *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition*, pp. 1106 – 1112.
- Hsu, Gee-Sern J. (2011). In A. Bhatti (Ed.), *Advances in theory and applications of stereo vision* (pp. 129-151 ). Rijeka, Croatia: InTech
- Jain, R., & Kasturi, R., & Schunck, B. G. (1995). *Machine Vision*, New York, United States of America, McGraw-Hill Inc.
- Jiang, G., & Zhao, C. (2010). Camera calibration based on 2D-plane. *Proceedings of the Third International Symposium on Electronic Commerce and Security Workshops(ISECS '10)*, pp. 365-368.
- Kanade, T., & Okutomi, M. (1991). A Stereo matching algorithm with an adaptive window: theory and Experiment. *Proceeding of the 1991 IEEE International Conference on Robotics and Automation*, pp. 1088 – 1095.
- Kohut, P. (2011). MELFA Basic IV - Robot Programming Language. AGH University of science and Technology of Krakow, Faculty of Mechanical Engineering and Robotics.
- Konolige, K. (1997). Small vision system: Hardware and implementation. *Proceedings International Symposium on Robotics Research*, pp 111-116.
- Loop, C., & Zhang, Z. (1999). Computing rectifying homographies for stereo vision. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Volume 1, pp. 125–131.
- Ma, L., Chen, Y., & Moore, K. L. (2003). A New Analytical Radial Distortion Model for Camera Calibration. Retrieved February 28, 2011, from University of Trier Web site:

<http://www.informatik.uni-trier.de/~ley/db/journals/corr/corr0307.html#cs-CV-0307046>

Matousek, M. (2007, February 28). [PDF] (*PhD Thesis: Epipolar Rectification Minimising Image Loss*, Center for Machine Perception, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University). Retrieved from: <http://cmp.felk.cvut.cz/ftp/articles/matousek/Matousek-PhD-TR-2007-06.pdf>

May, S., Pervoez, K., & Surmann, H. (2007). In G. Obinata & A. Dutta (Eds.) *Vision systems: Applications* (pp.181-203). Vienna, Austria: I-Tech Education and Publishing.

Niola, V., Rossi, C., Savino, S. & Strano, S.(2008). In C. Rossi (Ed.), *Brain, Vision and AI* (pp. 211-243). Vienna, Austria: In-Tech.

OpenCV. (2011, Jun). OpenCVWiki: Welcome. Web site consulted July 2, 2011: <http://opencv.willowgarage.com/wiki/Welcome>.

Papademetris, X. (2006). An Introduction to Programming for Medical Image Analysis with the Visualization Toolkit, from Yale University BioImage Suite Web site: [www.bioimagesuite.org](http://www.bioimagesuite.org)

Papadimitriou, D. V., & Dennis, T. J. (1996). Epipolar line estimation and rectification for stereo image pairs. *IEEE Transactions on Image Processing*, Volume 5(4), pp. 672-676.

Pollefeys, M., Kock, R., & Gool, L. V. (1999). A Simple and efficient rectification method for general motion. *Proceedings of the 7<sup>th</sup> IEEE International Conference on Computer Vision, Volume 1*, pp. 496-501.

Robert, L., Zeller, C., Faugeras, O., & Hébert, M. (1997). Applications of non-metric vision to some visually-guided robotics tasks. *Rapport de Recherche*, No. 2584.

Shah, M. (1997). Fundamentals of computer vision. Retrieved January 25, 2011, from University of Central Florida Computer Science Department Web site: <http://www.cs.ucf.edu/courses/cap6411/book.pdf>

Slabaugh, G. G. Computing Euler angles from a rotation matrix. Retrieved May 17, 2011, from: <http://www.gregslabaugh.name/publications/euler.pdf>

Stefano, L. Di., Marchionni, M., Mattocia, S., & Neri, G. (2002). A fast area-based stereo matching algorithm. *International Conference on Vision Interface*, pp. 146-153.

Sturm, P. F., Maybank, S. J. (1999). On plane-based camera calibration: A general algorithm, singularities, applications. *1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Volume(1), pp.1432-1437.

Szeliski, R. (2010). *Computer vision: Algorithms and applications*. New York: Springer.

Szeliski, R., & Zabih, R. (1999). An experimental comparison of stereo algorithms. *IEEE Workshop on Vision Algorithms: Theory and Practice*, pp. 1-19.

Tsai, R. Y. (1987). A Versatile camera calibration technique for high accuracy 3D machine vision metrology using off -the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation*, Volume RA-3, pp. 323–344.

- Vezhnevets, V., & Velizhev, A. (2005). GML Camera Calibration Toolbox (Version 0.4) [Software]. MSU Graphics and Media Lab, Computer Vision Group. Available from web site: <http://graphics.cs.msu.ru/en/science/research/calibration/cpp> .
- Vision Research Inc. (2011). Vision Research, Phantom. Web site consulted July 2, 2011 : <http://www.visionresearch.com/home/> .
- Visual Instrumentation Corporation. (2011, Jan). Visual Instrumentation Corporation - Specializing in Precision Accessory Products for High Speed Video Imaging Cameras and Applications. Retrieved July 2, 2011, from: <http://www.visinst.com/index.html#Products>.
- Waveren, J. P. V. (2005). From Quaternion to Matrix and Back. Id Software, Inc. Retrieved May 17, 2011, from: <http://www.fd.cvut.cz/personal/voracsar/GeometriePG/PGR020/matrix2quaternions.pdf>
- Wu, H. P., & Chen, C. (2007). In R. Stolkin (Ed.), *Scene reconstruction, pose estimation and tracking*, (pp. 221-242). Vienna, Austria: I-Tech.
- Zhang, Z. (1996). Determining the epipolar geometry and its uncertainty: A review, *Rapport de recherche, N° 2927, 11-37*.
- Zhang, Z. (1999). Flexible camera calibration by viewing a plane from unknown orientations. *International Conference on Computer Vision*, pp. 666–673.
- Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 22(11)*, pp. 1330–1334.



# 11      **Appendix A: Stereo Imaging**

- Stereo imaging.
- Working with two cameras.

## 11.1 Stereo Imaging

### 11.1.1 Introduction

Any point in the space can be related by three coordinates ( $X, Y, Z$ ). An image is represented in 2-D plane thus only two coordinates ( $x, y$ ) are required to represent a point in the image. One dimension is lost in the projection process. One of the most important tasks of Computer Vision is to recover this lost dimension.

A common method for recover a lost dimension (depth) from images is to acquire a pair of images, at the same time, using two cameras displaced by a known distance from each other, where each pixel is a function of the corresponding point in the scene.

In this sections, in order to describe the stereo vision methodology used in OpenCV, the researched started by presenting the *camera model*, the basis on which relies the geometry projections of a particular scene through the lenses to the imager, followed by the *calibration*, the methods and routines that allows us to retrieve the intrinsic and extrinsic parameters and then discuss the *undistortion* that discuss how to use those parameters to correct the lenses distortions of a single camera.

The second part of this section will present the necessary steps for stereo imaging - *undistortion*, *rectification* and *correspondence*.

Undistortion has the task of computing undistorted images by mathematically removing radial and tangential distortions. Rectification in its turn relates the two cameras in the space by means of rotations and translations and the result are pair of images row-aligned and rectified. Finally the correspondence finds the same features in the left and right camera views and outputs a disparity map.

### 11.1.2 Working With a Single Camera

#### 11.1.2.1 Camera model.

The camera model is the model that describes geometrically how the light, reflected from an object, travels through the camera lens and then to the imager. One of the models used to present the geometric aspects of vision is the pinhole camera model. A pinhole is an imaginary plan with a very small hole in the centre that blocks all the rays except those passing through the tiny aperture as showed in Figure 11.1.

In a real pinhole camera a point taken from a scene is projected onto an imaging surface (*imager*), as a result the image on the image plane (or the *projective plane*) is always in focus and the resulting image size relative to scene located at a distance  $Z$  depends only on the focal length  $f$  of the camera.

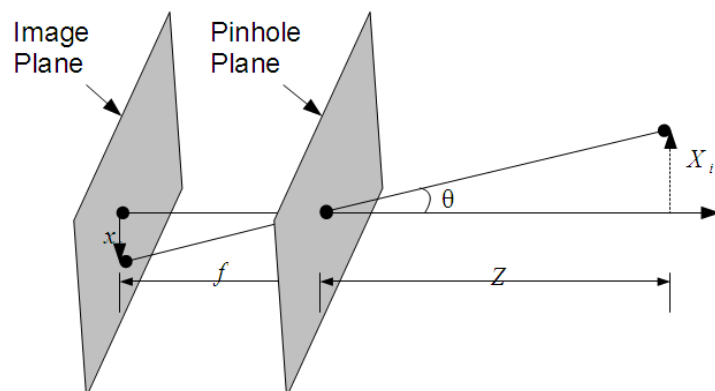


Figure 11.1: Pinhole camera model.

An object in the scene with the

length  $X_i$  is then projected into the object's imaging plane with a  $x_i$  length. From Figure 11.1 is possible to determine the following relation:

$$-\frac{x}{f} = \frac{X}{Z} \Leftrightarrow -x = f \frac{X}{Z} \quad (11.1.1)$$

In order to simplify the mathematical operations and make disappear the negative sign, the image plane and the pinhole plane are swapped, as showed in *Figure 11.2*, and the object's image is now right-side up.

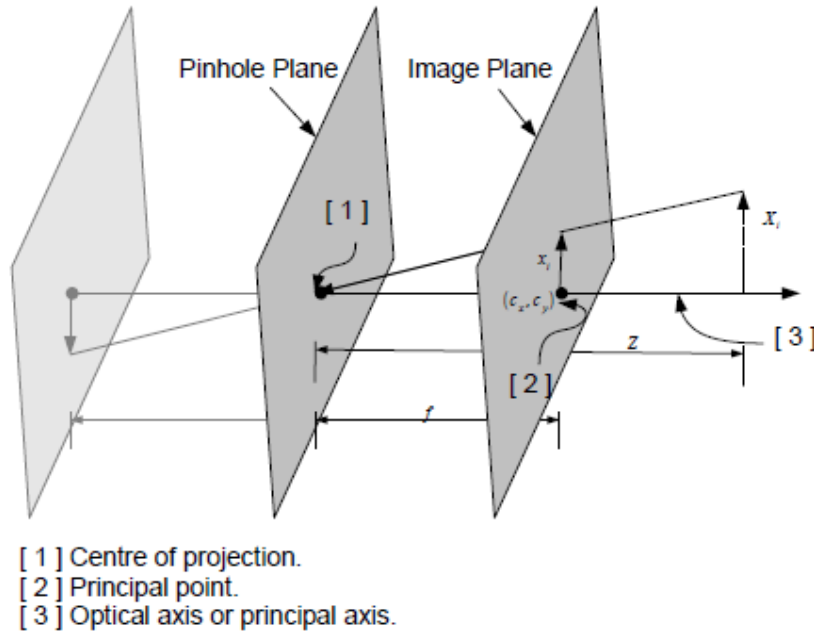


Figure 11.2: Simplified pinhole camera model.

The pinhole point becomes the *centre of projection* and the point that results from the intersection of the image plan and the *optical axis* is called *principal point*. In this rearrangement the rays travel from a specific point in the object towards the centre of projection and a more simplified relation is obtained :

$$\frac{x}{f} = \frac{X}{Z} \Leftrightarrow x = f \frac{X}{Z} \quad (11.1.2)$$

The previous equation (101) need to be reformulated to model possible displacements  $(c_x, c_y)$  of the imager centre coordinates from the *optical axis*. Thus a point in a scene  $Q(X, Y, Z)$  is projected in the projection plane at a pixel location  $(x_{imager}, y_{imager})$  given by the new equations:

$$x_{imager} = f_x \left( \frac{X}{Z} \right) + c_x, \quad y_{imager} = f_y \left( \frac{Y}{Z} \right) + c_y \quad (11.1.3)$$

The need to use two different focal length is due the fact that the individual pixel in the imager may be rectangular and not square as ideally expected. The new focal equations for the focal length are as follows:

$$f_x = F s_x, [pixels] = [mm] \frac{[pixel]}{[mm]} \quad (11.1.4)$$

$$f_y = F s_y, [pixels] = [mm] \frac{[pixel]}{[mm]} \quad (11.1.5)$$

The transform that relates the real world coordinates  $(X_i, Y_i, Z_i)$  to the points in the projection plane  $(x_i, y_i)$  is called a projective transform and is given by the following relation:

$$q_i = M Q_i \quad (11.1.6)$$

Where M, in this particular case, is the *camera matrix*.

$$\begin{bmatrix} x_i \\ y_i \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix}$$

#### 11.1.2.2 Lens distortion.

Another issue to be taken in account are the imperfections of the lenses that introduces distortions in the location of the pixels in the imager, thus new formulation need to be added to model to correct this distortions.

Two main distortions are present in the cameras: *radial distortions*, that results from the shape of the lenses, and *tangential distortions* that results from the assembly process of the camera.

Radial distortions happens near the edges of the imager resulting in a commonly known “barrel” or “fish-eye” effect. This distortion is zero at the centre of the imager and increases as the distance increases from the centre. To characterize radial distortions three terms are needed  $K_1, K_2$  and a third one  $K_3$  for lenses with high distortions such as fish-eye lenses. The relocation of a point in the imager will be computed according the equations as follows:

$$x_{corrected} = x_{imager} (1 + K_1 r^2 + K_2 r^4 + K_3 r^6) \quad (11.1.7)$$

$$y_{corrected} = y_{imager} (1 + K_1 r^2 + K_2 r^4 + K_3 r^6) \quad (11.1.8)$$

Tangential distortion is the distortions that happens due the imperfections resulting from the manufacturing process that prevents the lenses and the imager plane from being perfectly parallel. In order to correct such distortions two additional parameters are taken in account:

$$x_{corrected} = x_{imager} + [2p_1 y + p_2 (r^2 + 2x^2)] \quad (11.1.9)$$

$$y_{corrected} = y_{imager} + [p_1 (r^2 + 2y^2) + 2p_2 x] \quad (11.1.10)$$

Therefore five parameters are needed to model radial and tangential distortions. In OpenCV this parameters are normally grouped in a vector 5-by-1  $[K_1 \ K_2 \ p_1 \ p_2 \ K_3]^T$ . The third radial parameter  $K_3$  was introduced later in OpenCV and thus it appears in the fifth element of the *distortion vector*.

### 11.1.3 Calibration

The previous section introduced the camera model and how to determine mathematically the intrinsic and distortion properties of the camera. This section focus on how to use OpenCV to compute the intrinsic matrix and the distortion vector.

OpenCV provides helpful number of algorithms to compute the camera matrix and distortion parameters. The calibration is done by the routine `cv::calibrateCamera` and three additional routines: `cv::findChessboardCorners`, `cv::cornerSubPix` and `cv::drawChessboardCorners`.

This method requires that multiple views of a planar object are obtained by rotating and translating in different angles this same object. This object is a known structure with a predefined number of individual points.

#### 11.1.3.1 Find Chessboard Corners.

The calibration method uses a chessboard as a planar object, thus given an image of a chessboard the routine `cv::findChessboardCorners()` is used to locate a predefined number of corners of that chessboard as in Figure 11.3.

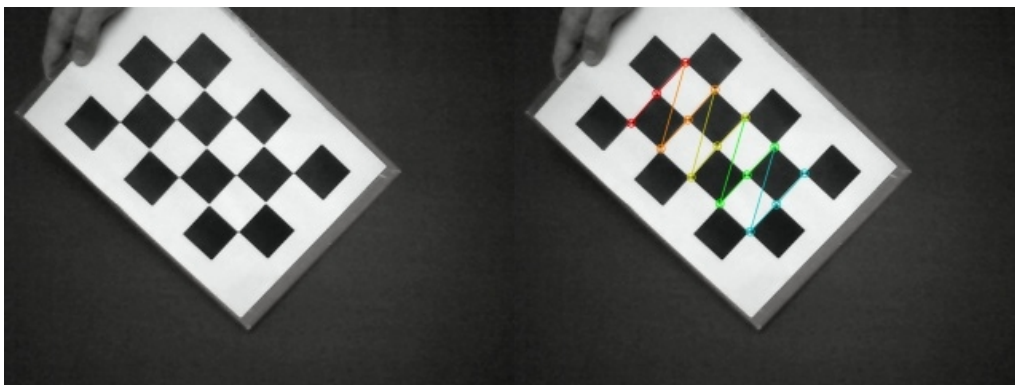


Figure 11.3: Example of a 5-by-3 chessboard corner's detection.

---

```
bool cv::findChessboardCorners( const Mat& image, Size patternSize, vector<Point2f>&
    corners, int flags = CV_CALIB_CB_ADAPTIVE_THRESH +
    CV_CALIB_CB_NORMALIZE_IMAGE );
```

---

`image` – Is the input image that contains the corners to be found, it must be an 8-bit grayscale or a color image.

`patternSize` – Indicates how many corners are in each row/column of the board – `cv::Size( nx, ny )`; This means that for example in a standard chessboard the correct `patternSize` values would be `cv::Size( 7, 7 )`.

`corners` – Is the output vector that will be used to store the corners locations. This vector must be preallocated and must be large enough to store all the corners on the board ( `nx * ny` ). They are

the locations of the corners in pixel coordinates.

flags – This argument can be used to implement one or more additional filtration steps to help find the corners on the chessboard. The filters can be combined using a boolean OR.

The available filter are:

- CV\_CALIB\_CB\_ADAPTIVE\_THRESH - cv::findChessboardCorners by default computes threshold in the image based on average brightness, but if this flag is set then an adaptive threshold will be used instead.
- CV\_CALIB\_CB\_NORMALIZE\_IMAGE – When set causes the image to be normalized via cv::equalizeHist before the thresholding being applied.
- CV\_CALIB\_CB\_FILTER\_QUADS - Once the image is thresholded, the algorithm attempts to locate the quadrangles and then a variety of additional constraints are applied to those quadrangles in order to reject the wrong ones.

The function cv::findChessboardCorners returns true if the function succeed finding all the corners ordered into rows and columns as expected, or false otherwise.

### 11.1.3.2 Find Corners Sub-pixel.

Once the routine to find the chessboard corners was applied to all images is necessary to compute the exact location of the corners to sub-pixel accuracy. This is done by the routine cv::cornerSubPix() described in Appendix F (see Appendix F for point sub pixel accuracy ).

In the presented calibration method a planar object was used. Before proceeding to the camera calibration first is showed what it is possible to do with a planar object by means of *planar homography* – mapping of points on a two dimensional planar surface to the imager.

### 11.1.3.3 Planar Homography.

Using homogeneous coordinates allow us to express Q and q points in the imager to which Q is mapped. The homography is expressed by the following relation:

$$\tilde{q} = sH\tilde{Q} \quad (11.1.11)$$

Where  $\tilde{q}$  and  $\tilde{Q}$  are defined as:  
 $\tilde{q} = [x \ y \ 1]^T$  and  $\tilde{Q} = [X \ Y \ Z \ 1]^T$

$H$  represents two parts: the physical transformation that locates the plane as we see and the and the projection that introduces the camera intrinsic matrix.

The parameter  $s$  is the scale value, i.e. the object will look the same even at different distances from the camera, thus the homography transform can only be defined up to an arbitrary scale value  $s$  that is that normally is set to 1.

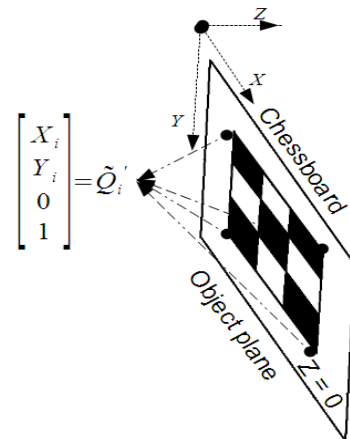


Figure 11.4: Generic point defined on a planar calibration object.

The physical transformation results from the rotation and translation that relates the chessboard to the image plane. The previous equation can be then rearranged into a new one:

$$\tilde{q} = sH \tilde{Q} \rightarrow \tilde{q} = sMW \tilde{Q} \quad (11.1.12)$$

Where W and M are :

$$W = \begin{bmatrix} R & t \end{bmatrix} \text{ and } M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (11.1.13)$$

The aim is to get the  $\tilde{Q}'$  (Figure 11.4 and Figure 11.5) a point defined only in a planar surface, the chessboard plane. Assuming that the object is defined in a plane so that  $Z = 0$  and decomposing the rotation matrix  $R$  in each rotation component  $R = \begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix}$  the homography matrix  $H$  that maps the chessboard points onto the imager is then described by the relation:

$$\tilde{q} = sH \tilde{Q}' \quad (11.1.14)$$

Eliminating the third rotation element, we obtain :

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = sM \begin{bmatrix} r_1 & r_2 & r_3 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = sM \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

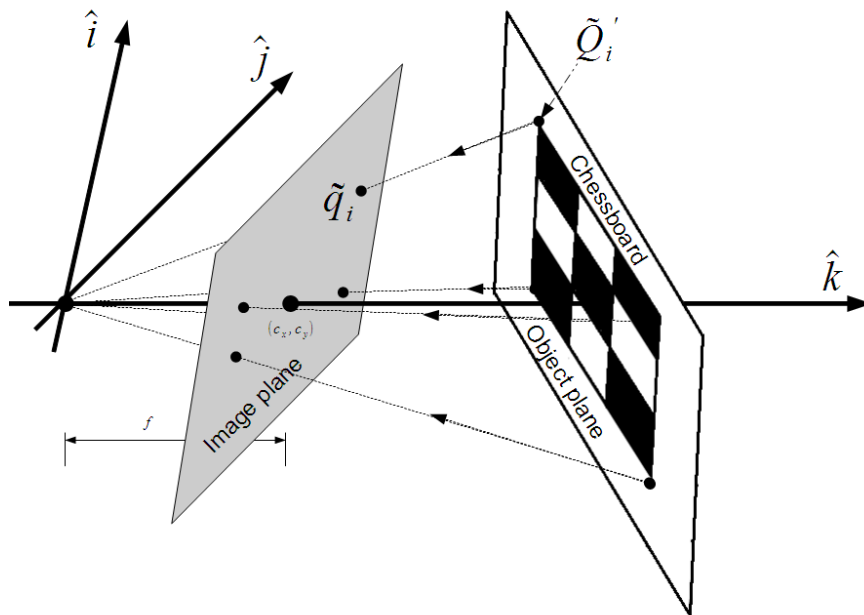


Figure 11.5: Relation between a point in the planar object and the imager plane.

From the previous equation (11.1.14) it is possible to conclude that for each planar object view there are six unknowns ( three angles for rotation and three offsets for translation ) for eight given equations that maps a square into a quadrilateral ( four corners times two coordinates (x, y) ). Thus given enough images it is possible to compute any number of unknowns.

Finally it is possible to answer the question – how the homography matrix  $H$  relates the points on a source object plane to the points on the imager plane?

The answer are the two next equations :

$$p_{\text{imager}} = H p_{\text{object plane}} \quad (11.1.15)$$

$$p_{\text{object plane}} = H^{-1} p_{\text{imager}} \quad (11.1.16)$$

Where  $p_{\text{imager}}$  and  $p_{\text{object plane}}$  are:

$$p_{\text{imager}} = \begin{bmatrix} x_{\text{imager}} \\ y_{\text{imager}} \\ 1 \end{bmatrix} \quad \text{and} \quad p_{\text{object plane}} = \begin{bmatrix} x_{\text{object}} \\ y_{\text{object}} \\ 1 \end{bmatrix}$$

The function that implements the presented homography formulation in OpenCV is `cv::findHomography()` which has the following parameters:

---

```
Mat cv::findHomography( const Mat& srcPoints, const Mat& dstPoints, Mat& status, int
                        method=0, double ransacReprojThreshold=0 );
```

---

**srcPoints and dstPoints** – Are matrices of CV\_32FC2 type or vectors containing Point2f elements as in vector < Point2f > srcPts. The first vector contains the object points  $(X_i, Y_i, 0)$  and the second the points in the target plane, that normally is the imager plane.

**method** – The method chosen to compute the homography matrix, it can be set to 0 to use all the points, CV\_RANSAC to used a more robust method or the Least-Median method by setting the method to CV\_LMEDS.

**ransacReprojThreshold** – Defines the threshold value used to accept or reject a maximum allowed reprojection error for the RANSAC method.

Computing multiple homographies from multiple planar object views is the method OpenCV obtains the intrinsic parameters.

#### 11.1.4 Camera Calibration

This subsection introduces how to compute the camera matrix and distortion parameters for one camera and how to use them to correct distortions in the raw image. Firstly is considered how many unknowns are related to each image and how this unknowns define the minimum number of images needed.

For each image there are four *intrinsic parameters*  $(f_x, f_y, c_x, c_y)$  and five *distortion parameters* : three radial  $(K_1, K_2, K_3)$  and two tangential parameters  $(p_1, p_2)$  . Intrinsic parameters are related to 3D space and distortion parameters to 2-D geometry, this two kinds of parameters are treated separately.



Six equations, from three corners points are enough to solve the five distortion parameters, thus only one view of a chessboard would be enough but for robustness more than one should be used. To compute the *extrinsic parameters* is necessary to know the three *rotation parameters* ( $\Psi, \Phi, \Theta$ ) and three translation parameters ( $T_x, T_y, T_z$ ) totaling ten intrinsic/extrinsic parameters that need to be solved for each view of the chessboard.

If  $N$  is the number of corners and  $K$  the number of images of the chessboard in different positions:

- $K$  images provides  $2NK$  constraints (  $NK$  constraints for each  $x, y$  coordinate ).
- Ignoring the five distortion parameters, we have 4 intrinsic parameters that are the same for all the images and  $6K$  extrinsic parameters
- To solve the unknowns requires that  $2NK \geq 6K + 4 \Leftrightarrow K(N-3) \geq 2$ . Having in mind that to compute the homography are needed at most four corners that result in eight parameters from  $(x, y)$  pairs, only four corners are taken in account no matters how many corners are in the chessboard. This implies that  $K \geq 2$  and  $N$  is at least 4 ( 3-by-3 chessboard ) are the minimum requirements to solve the calibration problem.
- For more precise results is recommended to use bigger number of images and a larger chessboard rotated at different angles in order to obtain a good set of views.

Each one of those parameters mentioned above, in the unknowns context, are used in different tasks :

- *camera intrinsic matrix* – Transform from 3-D coordinates to the imager 2-D coordinate. The inverse can be done but only to represent a line in 3-D space to which a 2-D imager point must correspond.
- *distortion coefficients* - Are used to correct the radial and tangential distortion in the raw image.
- *rotation and translation vectors* – Tells where the chessboards were found and their orientations.

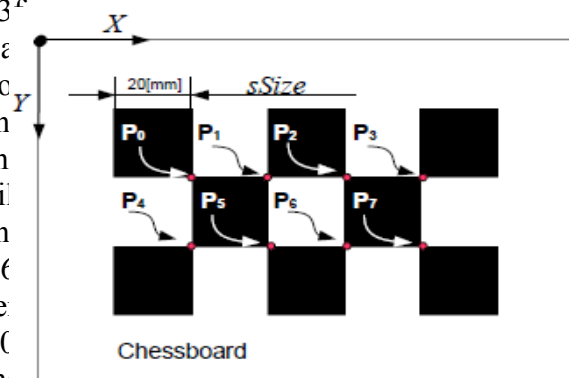
The routine that computes the camera intrinsics and distortion parameters in OpenCV is `cv::calibrateCamera()` and it is used internally in stereo calibration.

---

```
double cv::calibrateCamera( const vector<vector<Point3f>>& objectPoints, const
    vector<vector<Point2f>>& imagePoints, Size imageSize, Mat& cameraMatrix, Mat&
    distCoeffs, vector<Mat>& rvecs, vector<Mat>& tvecs, int flags=0 );
```

---

**objectPoints** – The input vector of `Point3f` containing  $n$  elements with the physical coordinates of each  $N$  corners on each one of the  $K$  chessboard images,  $n = N \times K$ . The way this points are defined are important once the manner of describing points in the object will define the physical units and the structure of the coordinate system used. As showed in Figure 11.6 if each square of a chessboard has 20mm size the camera and object coordinates are in mm/20. In the simplest case each square as unit one and the corners are represented by integers coordinates. Choosing a column to be all zero value it define that the location of the planar object relative to the



$nCorners = (n_x - 1) * (n_y - 1)$   
 $n_x$  = number of square along XX direction.  
 $n_y$  = number of square along YY direction.

Figure 11.6: Calibration object - chessboard 5-by-3.

camera will be along that direction i.e. 1<sup>st</sup> column – x direction, 2<sup>nd</sup> column – y direction and 3<sup>rd</sup> column – z direction. The procedure of building the object points for each calibration view's pattern (Figure 11.6) is showed in Figure 11.7.

**imagePoints** – The input vector containing a group of K vectors of Point2f elements that are no more than the group of N corners coordinates from each calibration view given by the `cv::findChessboardCorners` function.

**imageSize** – Defines the image size of the chessboards images from where the corners were extracted.

**cameraMatrix** and **distCoeffs** – Are the intrinsic camera parameters outputted by the routine, or optionally used as input matrices if `CV_CALIB_USE_INTRINSIC_GUESS` flag is used, affecting the computed results. The way these inputs are used depends on the flags parameter. The camera matrix is always 3-by-3 and the distortion coefficients a 5-by-1 matrix with the parameters in the next order:  $k_1, K_2, p_1, p_2, K_3$ .

**rvecs** – Output vector of rotations matrices computed for each calibration pattern view. Each matrix represents a group of 3 vectors axis in three-dimensional space on camera coordinate system around which the chessboard was rotated. It is possible to convert this vectors into 3-by-3 rotation matrices using the `cv::Rodrigues()` routine.

**tvecs** – Output vector of translations estimated for each calibration pattern view in the camera coordinate system. As stated in the first argument, the units of the camera coordinate system are exactly the same as the ones assumed for the chessboard, i.e. if the chessboard square size was defined in meters the translation will be in meters, if it was defined in inches the translation units will be in inches and so on.

**flags** – The last argument allows to control how the calibration will be performed. Different flags can be combined together with OR operator in order.

```
Point3f pointXYZ0;
vector< Point3f > objectPoints;
for( int i = 0; i < nCorners; i++ )
{
    pointXYZ0 = Point3f( i % nx * sSize, i / nx * sSize, 0.0f )
    objectPoints.push_back( pointXYZ0 );
}
```

*Figure 11.7: Building object point's vector of vector of 3D points.*

Once known the intrinsic and extrinsic parameters the next step is to make use of this parameters to correct the image distortions.

### 11.1.5 Undistortion

The routine that was previously discussed, `cv::calibrateCamera`, only provides a group of valuable parameters that need to be used in further code implementation.

One of the tasks to be done with the calibration parameters is to correct the distortion effects. OpenCV provides three main routines that allows to do exactly that: `cv::undistort()`, `cv::initUndistortRectifyMap()` and `cv::remap()`. The first function transforms the image to compensate radial and tangential lens distortion and it is a combination of the last two functions. This function

should be used only when the distortion parameters are changing however if the distortion maps are not changing the use of this routine is inefficient and time consuming specially when dealing with video sequences. The function has the following arguments:

---

```
void cv::undistort( const Mat& src, Mat& dst, const Mat& cameraMatrix, const Mat&
    distCoeffs, const Mat& newCameraMatrix=Mat() );
```

---

src – The input distorted image to be corrected.

dst – The output corrected image with the same size and type as the input image.

cameraMatrix – The input camera matrix that should have different parameters between different images remapping process.

distCoeffs – The input 5-by-1 vector of distortion coefficients for each input image.

newCameraMatrix – The input new camera matrix that by default is the same as cameraMatrix but it can be adapted to the new region of interest in the source image. This new camera matrix is obtained with `cv::getOptimalNewCameraMatrix()` function.

To compute the undistortion and rectification maps `cv::initUndistortRectifyMap()` function is used, it contains the following arguments:

---

```
void cv::initUndistortRectifyMap( const Mat& cameraMatrix, const Mat& distCoeffs, const
    Mat& R, const Mat& newCameraMatrix, Size size, int m1type, Mat& map1, Mat&
    map2 );
```

---

cameraMatrix – The input camera matrix that is constant for a sequence of images.

distCoeffs – The input 5-by-1 distortion coefficients vector.

R – The 3-by-3 optional rectification transformation matrix obtained with `cv::stereoRectify()`, if this matrix is empty the identity matrix is assumed i.e. no rotation will be assumed for any axis.

newCameraMatrix – The new camera matrix that for a single camera can be the same as the cameraMatrix or obtained with `cv::getOptimalNewCameraMatrix()`.

size – The size of the corrected image, it can be the same size as the original image or other if a different subregion of the image is used.

m1Type – The type of the first output map that can be CV\_32FC1 or CV\_16SC2.

map1 and map2 – The first and second output maps to be used by `cv::remap` function.

To apply the remapping maps to an image or sequence of images for video capture the `cv::remap` function need to be called. It has the following arguments:

---

```
void cv::remap( const Mat& src, Mat& dst, const Mat& map1, const Mat& map2, int
    interpolation, int borderMode = BORDER_CONSTANT, const Scalar& borderValue =
    Scalar() );
```

---

src – The current image to be subjected to the geometrical transformations.

dst – The output image with same size of map1 and the same type of src image.

map1 and map2 – The first and second map of x and y coordinates, respectively. If the maps are in the original floating-point format they can be converted to fixed-point representation for a faster computation, the most used conversion is from (CV\_32FC1, CV\_32FC1) to (CV\_16SC2, CV\_16UC1) and it can be done by using `cv::convertMaps()`.

interpolation – The interpolation method used in the remapping process.

borderMode - The pixel extrapolation method, by default is set to constant (BORDER\_CONSTANT).

borderValue - The value used to represent the pixel outliers if the borderMode is set to constant.

## 11.2 Working With Two Cameras

### 11.2.1 Stereo Imaging

There are several ways to recover 3D information from 2D images and one of them humans use frequently – the stereo imaging.

Previously was discussed in detail the camera model, how to model the lenses distortion and how to retrieve the necessary parameters to calibrate one single camera and subsequently correct the distorted images. The goal of this section is to construct 3D representations of the images, captured from two cameras relatively at the same time, using the basis achieved in the previous section to perform the stereo calibration, rectification and correspondence.

#### 11.2.1.1 Stereo Geometry.

The geometry of a stereo imaging is shown in the next figure (see Figure 11.8). To simplify, the model is composed by two identical cameras separated in the  $x$  direction from each other by a distance  $b$ . The image planes are ideally coplanar, which in reality doesn't happen specially when we want to have a bigger field of view. This topic will be discussed in more detail in rectification.

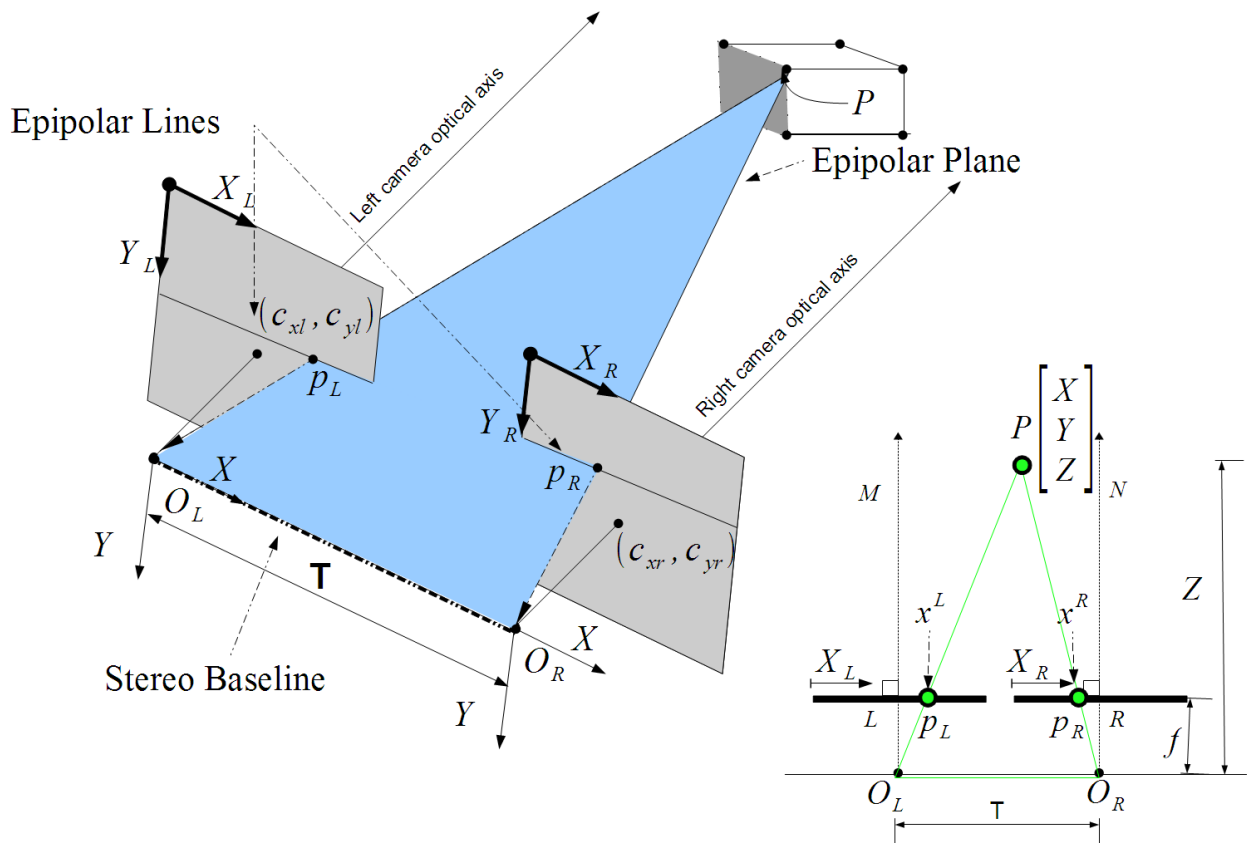


Figure 11.8: Stereo configuration geometry .

An object in the scene is viewed by the two cameras at different positions in the image plane, the displacement between this locations is called *disparity*. The plane passing through the object point and the centres of projection is called *epipolar plane* and the intersection of this plane with the imager plane defines the *epipolar line*. Also ideally, all the object points in one image will be in

the same row in the second image. In real cases there are some disparity vertically or horizontally, depending on the cameras arrangement, that need to be corrected.

Assuming that the origin of the coordinate system coincides with the left centre of projection, defining the disparity  $d$  as follows:

$$d = x^L - x^R \quad (11.2.1)$$

By similarity of the triangles is possible to obtain the following relationship:

$$\frac{T - (x^L - x^R)}{Z - f} = \frac{T}{Z} \Rightarrow Z = \frac{fT}{x^L - x^R} \quad (11.2.2)$$

### 11.2.1.2 Epipolar Geometry.

The previous assumption that the images planes are coplanar does not happen in a real situations. In some stereo configurations the cameras are oriented in such a way that their optical axis meet at a point in space in order to increase the field of view, this physical configuration showed in Figure 11.9 is known as *standard stereo configuration*.

In a standard stereo configuration the epipolar lines are no longer aligned with the image rows and thus the alignment need to be computed mathematically to obtain a row-aligned pair of images.

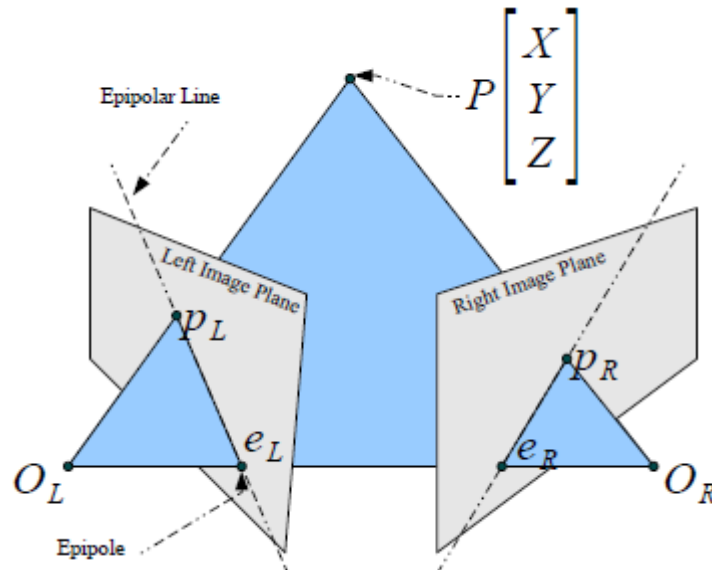


Figure 11.9: Standard stereo configuration's epipolar geometry.

To understand how the alignment is done mathematically the researcher starts by presenting the new nomenclature. When a point on the object scene  $P$  is projected into the right and left image plane,  $p_R$  and  $p_L$  respectively, that point can be anywhere along a single line formed by the point that is the projection of point  $P$  and the centre of projection ( $O_L$  and  $O_R$ ) on that camera. In Figure 11.9 the two lines are defined by the two segments  $\overline{p_R O_R}$  and  $\overline{p_L O_L}$ . The two segments correspond exactly to the epipolar lines when projected in the other imager plane i.e.  $\overline{p_R O_R}$  corresponds to  $\overline{p_L e_L}$  and  $\overline{p_L O_L}$  corresponds to  $\overline{p_R e_R}$ . Thus

given a projection of point in one image plane, its matching view in another image plane must lie along the corresponding epipolar line – this constraint is known as the *epipolar constraint* and it reduces the matching from 2D search to 1D search along the epipolar lines across the two imagers resulting in the rejection of bad points and less computational costs.

Additionally both cameras need to be related in physical space. The relation between the two cameras is achieved by computing the essential matrix  $E$  and the fundamental matrix  $F$ . The matrix  $E$  contains information about the rotation and translations that relates the two cameras and thus be able to relate a point in one image to a line in the other.

$F$  contains the same information as  $E$  and additionally the intrinsics parameters of both cameras. Because  $F$  takes in account the intrinsics parameters it relates the two cameras in pixels coordinates.

To obtain matrix  $F$  for a stereo configuration OpenCV provides the function `cv::findFundamentalMat()` that takes the following arguments:

---

```
Mat cv::findFundamentalMat( const Mat& points1, const Mat& points2, int method =
    FM_RANSAC, double param1=3., double param2 = 0.99 );
```

---

`points1` and `points2` – Are the  $n$ -by-2 or  $n$ -by-3 floating-point ( single or double precision ) arrays containing the  $n=N \times K$  2-D points that where collected from the ( left, right ) chessboard images.

`method` – This parameter defines the method to be used to compute the fundamental matrix. It can be one of the four following values:

- `CV_FM_7POINT`,  $n = 7$  – Uses only 7 points and impose that matrix  $F$  must be of rank 2 to fully constraint the matrix. This constraint is not absolutely unique and the routine may return three different matrices and, as mentioned, the fundamental matrix allocation need to be done for a 9-by-3 array.
- `CV_FM_8POINT`,  $n \geq 8$  – This method solves  $F$  as a linear system of equations, if more than 8 points were supplied then a linear least-square error is minimized across all points.
- `CV_FM_RANSAC`, `CV_FM_LMEDS`,  $n \geq 8$  – The previous two methods are very sensitive to outliers. This last two methods are more robust once they have the ability to recognize and remove those outliers. For both methods it is required two have much more points than the minimum.

`param1` and `param2` – This two parameters are related to the last methods RANSAC and LMedS. The first parameter defines the minimum distance in pixels from a point to the epipolar line from where a point is considered an outlier while the second parameter is the desired confidence that implicitly tells the routine how many times to iterate.

The function returns the fundamental matrix using one of the four available methods.

### 11.2.1.3 *Epipolar Lines.*

Having the fundamental matrix is possible to compute the epipolar lines. The routine that does this for us is `cv::computeCorrespondEpilines()`. Given a list of  $N$  points in one image it computes the correspondent epipolar lines for each point in the other image. Each epipolar line is described by three coefficients  $a$ ,  $b$  and  $c$  that define a line equation. It has the following arguments:

---

```
void cv::computeCorrespondEpilines( const Mat& points, int whichImage, const Mat& F,
    vector<Vec3f>& lines );
```

---

points – The input vector of image points `vector< Point2f >` or optionally it can be an N-by-1 or 1-by-N matrix of CV\_32FC2 type.

whichImage – This parameter is set to 1 or 2 to indicate which one of the images belongs the points.

F – The fundamental matrix that relates the points in both images. It can be obtained by using the function previously described `cv::findFundamentalMat( )` or `cv::stereoRectify( )`. Before using F matrix is wise to verify if it is not empty.

lines – The output vector that contains the coefficients of the line equation corresponding to each of the inputted image points.

### 11.2.2 Stereo Calibration

After introducing the stereo basis and methodology discussed up to this point this section will cover stereo calibration, stereo rectification and stereo correspondence. Stereo calibration has the task to compute the geometrical relationship between the two cameras in the space while stereo rectification corrects the individual images so they appear to be taken from two cameras with row-aligned coplanar image planes as ideally expected. Stereo correspondence is the last step to be performed , it matches the image points from an object in 3D space seen by two different camera over areas where the two camera views overlap.

#### 11.2.2.1 Stereo Cameras Calibration.

Stereo calibration is the process of computing the geometrical relationship between the two cameras in space. It depends on finding the single rotation matrix  $R$  and translation vector  $T$  , that relates the right camera to the left camera , obtained by the function `cv::stereoCalibrate( )`. This function operates similarly to `cv::calibrateCamera` except now it deals with two cameras and it can compute the camera matrices, distortion coefficients, essential matrix  $E$  and fundamental matrix  $F$ . The function `cv::stereoCalibrate( )` has the following parameters:

---

```
double cv::stereoCalibrate( const vector<vector<Point3f> >& objectPoints, const
    vector<vector<Point2f> >& imagePoints1, const vector<vector<Point2f> >&
    imagePoints2, Mat& cameraMatrix1, Mat& distCoeffs1, Mat& cameraMatrix2, Mat&
    distCoeffs2, Size imageSize, Mat& R, Mat& T, Mat& E, Mat& F, TermCriteria termCrit
    = TermCriteria(TermCriteria::COUNT+ TermCriteria::EPS, 30, 1e-6), int flags =
    CALIB_FIX_INTRINSIC );
```

---

objectPoints – The vector of vectors containing the physical coordinates of each of the  $N$  points on each of the  $K$  images of the 3D object such that  $n = N \times K$  where  $N$  is the number of points and  $K$  the number of images. When using chessboards as the 3D object the Z-coordinate of the points on the chessboard plane is usually set to 0 but any known 3D points may be used. This argument is crucial in the manner of describing the points on the object and to define the physical units and the structure of the coordinate system to be used from this point.

imagePoints1 and imagePoints2 – The vector of vectors containing the left and right pixel



coordinates, respectively, of all of the object reference points supplied in `objectPoints`, i.e. they contain the returned values for  $K$  calls to `cv::findChessboardCorners` for the left and right camera views.

`cameraMatrix1` and `cameraMatrix2` – The input/output 3-by-3 camera matrices for cameras 1 and 2, respectively.

`distCoeffs1` and `distCoeffs2` – The input/output 5-by-1 distortion matrices for cameras 1 and 2, respectively. These matrices are filled in such order that the two first parameters are the first two radial parameters followed by the two tangential parameters and the third radial parameter.

`imageSize` – The image size that is used only to initialize the intrinsic camera matrix.

`R` and `T` – The rotation and translation matrices, relating the right camera to the left camera – the routine main finality.

`E` and `F` – The output 3-by-3 essential and fundamental matrices.

`termCrit` – Sets the criteria to stop the internal optimization whether after a certain number of iteration or when the computed parameters change by less than a specific value. A typical value is `cv::TermCriteria( CV_TERMCRIT_ITER && CV_TERMCRIT_EPS, 100, 1e-5 )`.

`flags` – Define the way in which the intrinsic parameters are used. If set to `CV_CALIB_FIX_INTRINSIC`, then these matrices are used as input and are obtained by using the function `cv::calibrateCamera`. If `flags` is set to `CV_CALIB_USE_INTRINSIC_GUESS` these matrices are used as a starting point to optimize further the intrinsic and distortion parameters for each camera and will be set to the refined values on return from `cv::stereoCalibrate`. The first case is preferred once `cv::calibrateCamera` provides a more robust method to estimate the intrinsic and extrinsic parameters individually for each camera.

### 11.2.3 Stereo Rectification

The aim of stereo rectification is to reproject the left and right image of the two cameras so that they reside in the exact same plane, with image rows aligned into a frontal parallel configuration, or image column aligned in the case of vertical configuration.

#### 11.2.3.1 Uncalibrated stereo rectification - Hartley's algorithm.

This algorithm has the advantage of performing online stereo calibration by observing points in the scene however it does not have the notion of image scale - the feature points have the same 2D coordinates even though the 3D object positions (not orientation) differ.

The function `cv::stereoRectifyUncalibrated( )` compute the homographies used for rectification and has the following parameters.

---

```
bool cv::stereoRectifyUncalibrated( const Mat& points1, const Mat& points2, const Mat& F,
                                   Size imgSize, Mat& H1, Mat& H2, double threshold = 5 );
```

---

`points1` and `points2` – The function takes as input two 2-by- $K$  matrices with the corresponding points between the left and right images.

`F` – The fundamental matrix `F` obtained by using the `cv::findFundamentalMatrix`.

`imageSize` – Describes the width and height of the images that were used during calibration.

`H1` and `Hr` – The output rectification homography matrices for the first and for the second

images respectively.

threshold – If the distance from points to their correspondence epilines exceeds the threshold value, the corresponding point is considered outlier and eliminated.

After obtaining the homography matrices a last step need to be done to obtain the 3-by-3 rectification transformations  $R_1$  and  $R_2$  in object space. this is done by preprocessing the homographies using the next relations:

$$\begin{aligned} R_1 &= cameraMatrix_1^{-1} \times H_1 \times cameraMatrix_1 \\ R_2 &= cameraMatrix_2^{-1} \times H_2 \times cameraMatrix_2 \end{aligned} \quad (11.2.3)$$

### 11.2.3.2 Calibrated stereo rectification: Bouguet's algorithm.

This method minimize the amount of change reprojection produces for each of the two images while maximizing common viewing area. Applying the Bouguet's rectification method produces the ideal stereo configuration with a perfectly undistorted and row-aligned stereo images. The function that performs this task is `cv::stereoRectify()` and has the following arguments:

---

```
void cv::stereoRectify( const Mat& cameraMatrix1, const Mat& distCoeffs1, const Mat&
    cameraMatrix2, const Mat& distCoeffs2, Size imageSize, const Mat& R, const Mat& T,
    Mat& R1, Mat& R2, Mat& P1, Mat& P2, Mat& Q, int flags=CV_CALIB_ZERO_DISPARITY );
```

---

cameraMatrix1 and cameraMatrix2 – The input camera matrices of the left and right cameras returned by `cv::stereoCalibrate` function.

distCoeffs1 and distCoeffs2 – The input distortion parameters matrices of the left and right cameras returned by `cv::stereoCalibrate` algorithm.

imageSize – Is the size of the chessboard images used to perform the calibration.

R and T – The input rotation and translation matrices returned by `cv::stereoCalibrate` function.

R1 and R2 – Are the returned 3-by-3 row-aligned rectification rotations for the left and right image planes.

P1 and P2 – The output 3-by-4 left and right projection equations P1 and P2.

Q – The optional 4-by-4 reprojection matrix. When given a two dimensional homogeneous point and its associated disparity, to project the point into three dimensions.

Flags – The default value set the disparity at infinity (CV\_CALIB\_ZERO\_DISPARITY) with the principal point of each camera having the same pixel coordinates in the rectified image. Changing the default value means that the cameras are verging toward each other.

### 11.2.3.3 Rectification Maps.

Once the rectification rotations and the projection equations were computed is possible to pre-compute the left and right rectifications maps for the left and right camera views using two separate calls to `cv::initUndistortRectifyMap` function. The resulting maps are used by `cv::remap` function that is called once for each left and right camera's views. The arguments of this two

functions are described in the subsection **Working with a Single Camera**.

After computing the rectifications maps for the raw images `cv::remap` is used to take pixels from one place in the image and map them to another place given by `mapx` and `mapy`. In this way the feature points become horizontally aligned in the undistorted rectified images. In the next figure (Figure 11.10) is displayed two video captures after subjected to the rectification (undistortion+rectification) process, the horizontal lines were added to help to visualize the pixel row alignment obtained between the left and right capture.

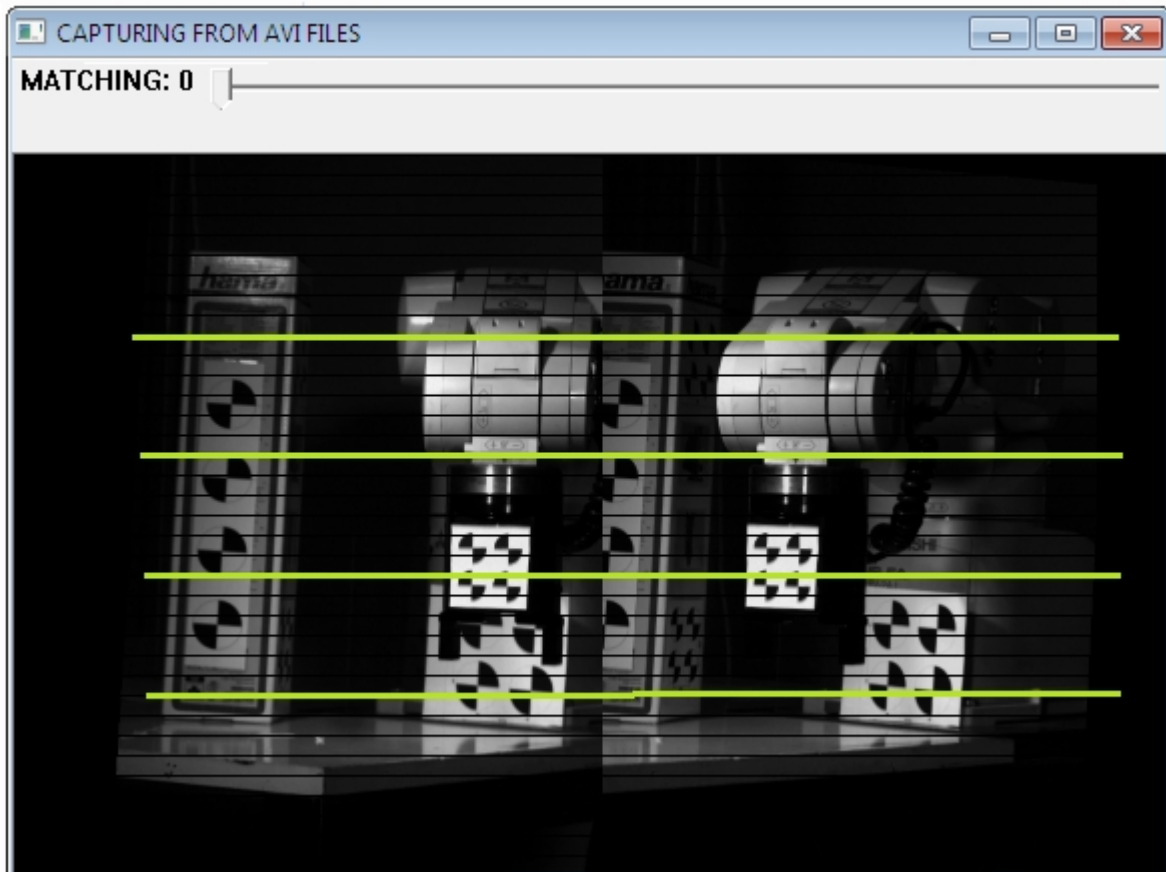


Figure 11.10: Video sequences after stereo rectification.

#### 11.2.4 Stereo Correspondence

Matching a 3D point in the two different camera views can be computed only over visual areas in which the views of the two cameras overlap. If rectification was done in both images the stereo-matching process is the next step to be performed.

OpenCV has different approaches to compute the disparity between two images captured from a stereo configuration. The block matching and semi-global block matching method are the ones that present better results with less computational costs.

The Semi Global Block-Matching and Block-Matching parameters and the internal buffers are kept in a data structure initialized by `cv::StereoSGBM()` (or similarly by `cv::StereoBM()`) as follows:

---

```
class cv::StereoSGBM{ ...; int presetFlag; int minDisparity; int numberOfDisparities; int
    SADWindowSize; int preFilterCap; int uniquenessRatio; int speckleWindowSize; int
    speckleRange; int disp12MaxDiff; bool fullDP; ... }
```

---

presetFlag – This preset parameter can be set to any of the following values:

CV\_STEREO\_BM\_BASIC (sets all parameters to their default values), CV\_STEREO\_BM\_FISH\_EYE (sets parameters for wide-angle lenses), CV\_STEREO\_BM\_NARROW (sets parameters for stereo cameras with narrow field of view).

minDisparity – The minimum acceptable disparity to be considered correct. By default this value is set to 0.

numberOfDisparities – The maximum disparity minus the minimum disparity. If set to non-zero value it overrides the default pre set value. It must be a number multiple of 16.

SADWindowSize – Is the size of the averaging window used to match pixel blocks. The block-matching window size must be equal or greater than 1 and odd number. If it is set to one, instead of blocks, the function matches single pixel. Larger values gives better robustness to noise, but origins blurry disparity maps. Values between 3 and 11 may give the better results.

preFilterCap – Is the value, ranging between 0 and 63, for the Tomasi cost function to limit the values to  $[-preFilterCap; preFilterCap]$  interval.

uniquenessRatio – Is the value that defines the percent by which the minimum cost function value should be considered better than a second good match value. Values within the range 5-15 are recommended.

speckleWindowSize – The integer value that defines the threshold disparity regions to be considered noise. To disable the speckle filter this value must be set to zero otherwise values within the range 50-200 are advised.

speckleRange – The maximum disparity variation within each region. If speckleWindowSize is a non-zero value this value must be positive and multiple of 16. Values of 16 and 32 are advised.

disp12MaxDiff – The maximum allowed difference between the first and second disparity image validation. To disable the validation the value should be set to a negative value.

fullDP – By default the function only considers 5 directions for the matching, if this value is set to true all the 8 directions are taken in account. Setting this value to true is not advisable for real time video processing due to its high computational costs.

After having initialized the block-matching group of variables the disparity maps can be computed using the operator `cv::StereoSGBM::operator( )` (or similarly `cv::StereoBM::operator( )`) as follows:

---

```
void cv::StereoSGBM::operator( )(const Mat& left, const Mat& right, Mat& disp);
```

---

left – The left camera's image 8-bit single channel or 3-channel image.

right – The right camera's image with the same size and same type as the left camera's image.

disp – The resulting single channel disparity image of 16-bit signed type with the same size as the left and right camera's image. To convert it to floating-point type(or other type) each disparity

value need to be divided by the number of disparities times 16 as follows:

$$\text{disp.convertTo}(\text{disp8UType}, CV\_8U, 256/nOfDisparities) \quad (11.2.4)$$

Both `cv::StereoSGBM::operator()` and `cv::StereoBM::operator()` functions operators takes undistorted rectified stereo images pairs and outputs a disparity map given its state structure. The block-Matching settings can be updated or readjusted by using trackbars values and imposing some validation for the state variables that requires specific input values.

## **12      Appendix B: MELFA Basic IV Presentation**

- Robot programming Language – MELFA.

## Robot Programming Language - MELFA

### MELFA BASIC IV

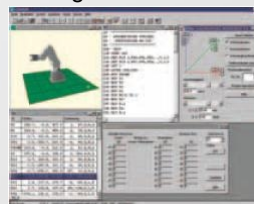
#### RV-2AJ



Mitsubishi robots use their own programming language for the robot controller, MELFA Basic IV.

The robot programming language MELFA BASIC IV is powerful yet easy to learn, ensuring that users can start producing their own powerful and efficient robot programs in a very short time.

Cosirob is programming environment for all Mitsubishi robots



Piotr Kohut, Ph.D

**R V - 2 A J**

Robot

2 kg

5-Axis

Vertical  
articulated

Series



## Main Main characteristics of characteristics of RV-2AJ



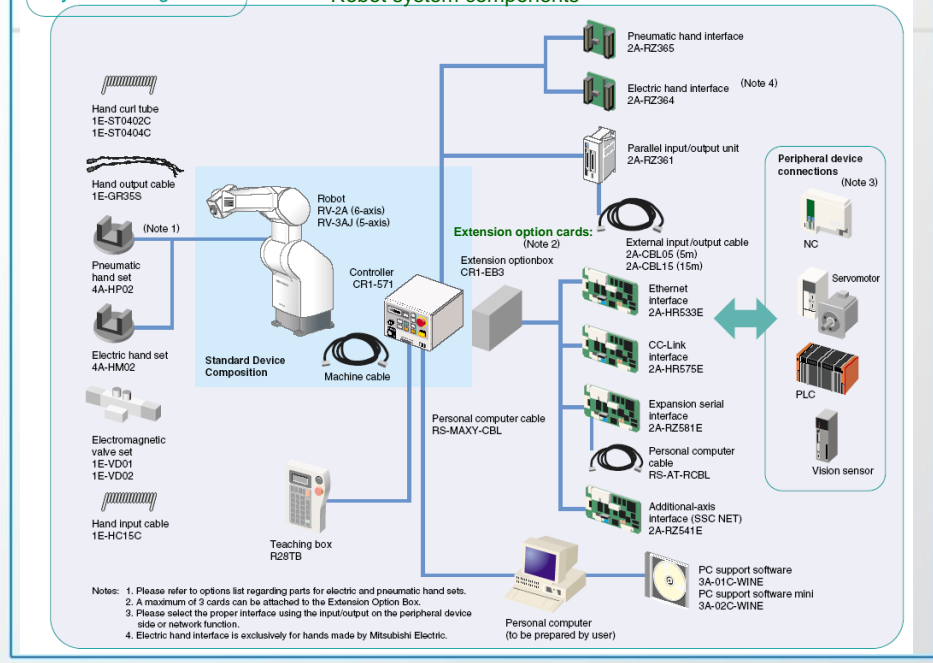
Manufacturer	Mitsubishi-Electric
No DOF	5
Robot weight	17 [kg]
range	410 [mm]
Repeatability	+/-0.02 [mm]
Max. load	2 [kg]
Max. speed	2100 [mm/s]
Max. No of Tasks	32
Max. No inputs / outputs	240 / 240
AC power	230 [V] / 50 [Hz]

It is ideal for the applications (testing, material handling, training...), it opens up a completely new range of possibilities with a speed of up to 2,100 mm/s, significantly improved repeatability of  $\pm 0.02$  mm and a handling payload up to 2 kg

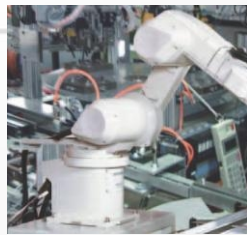
3

### System Configuration

### Robot system components

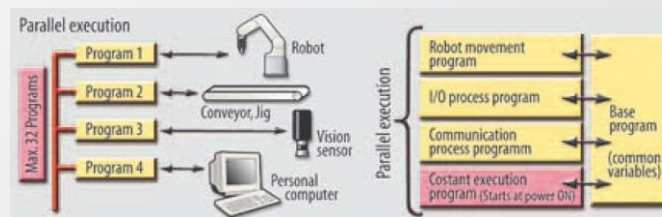






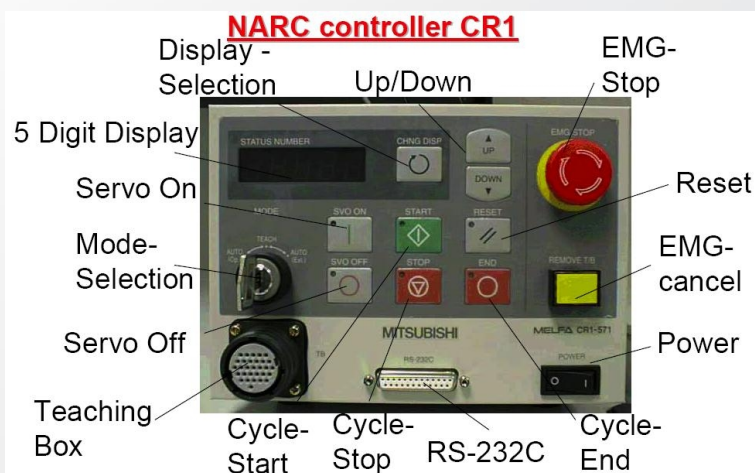
Controller type	CR1, CR2, CR3
Control mode	PTP and CP
Processor	64-bit RISC + DSP
Control functions	Axial, linear and 3-D circular interpolation; palletising functions, interrupt control and multitasking
Max. position points	88
Max. Anzahl der Positionspunkte	2,500 per program
Max. program lines	5,000 per program
Internal I/Os	CR1: 16 I/16 O, max. 240 I/240 O CR2/CR3: 32 I/32 O, max. 256 I/256 O
Safety functions	EMERGENCY OFF and door contact switch
Power supply	CR1/CR2: 207-253V AC, single-phase CR3: 400 V AC; three-phase
Max./normal power consumption	3.5kVA/0.9kVA
Outside dimensions (W/H/D in mm)	CR1: 212/166/290, CR2: 460/200/400, CR3: 450/975/380

Multitasking function for parallel execution of multiple tasks. A 64-bit RISC processor with DSP provides ample power for 3-D circular and linear interpolation, and for multitasking with up to 32 programs running in parallel.

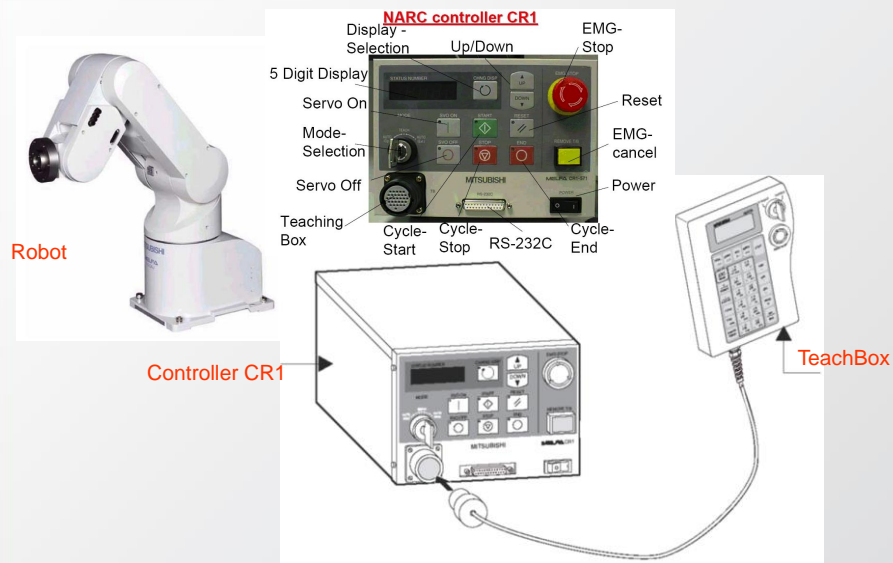


## Robot controller CR-1

The robot controller has a 64-bit RISC processor and can be programmed quickly and easily in MELFA BASIC IV.



## Controller CR-1 – Teaching Pendant



8

Robot controlling by means of  
Teach Box

Frame Coordinate Systems  
(Spaces in which the robot can be controlled )

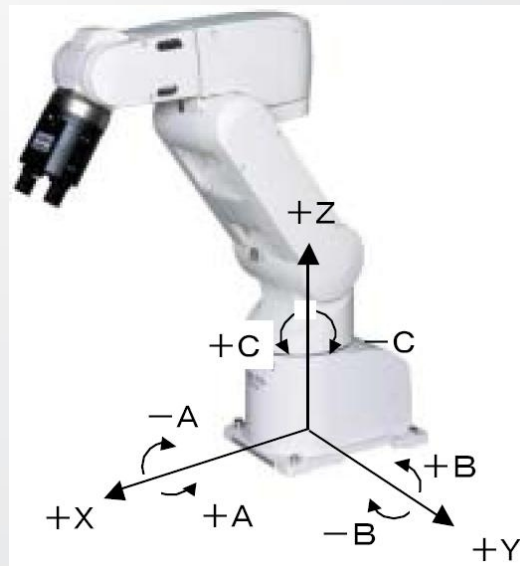
## Joint space – JOINT jog

Adjusts the coordinates of each axis independently in angle units



## Cartesian space – XYZ jog

Adjusts the axis coordinates along the direction of the robot coordinate system. The X, Y, and Z axis coordinates are adjusted in mm units. The A, B, and C axis coordinates are adjusted in angle units.

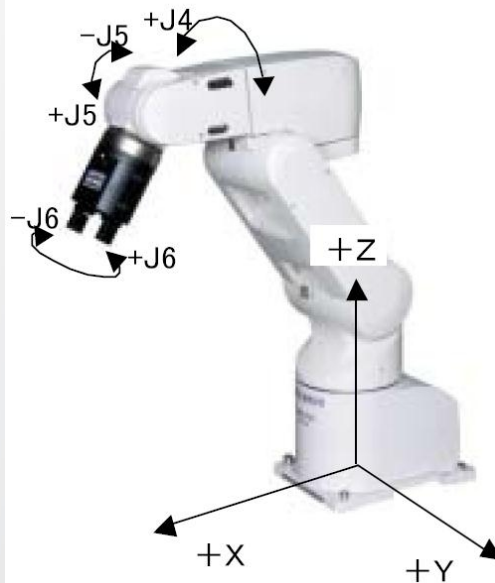


### Three axes XYZ jog

Adjusts the X, Y, and Z axis coordinates along the direction of the robot coordinate system in the same way as in XYZ jog feed.

The J4, J5, And J6 axes are adjusted independently in the same way as in JOINT jog feed.

The X,Y, And Z axis coordinates are adjusted in mm units.  
The J4, J5, and J6 axis coordinates are adjusted in angle units.

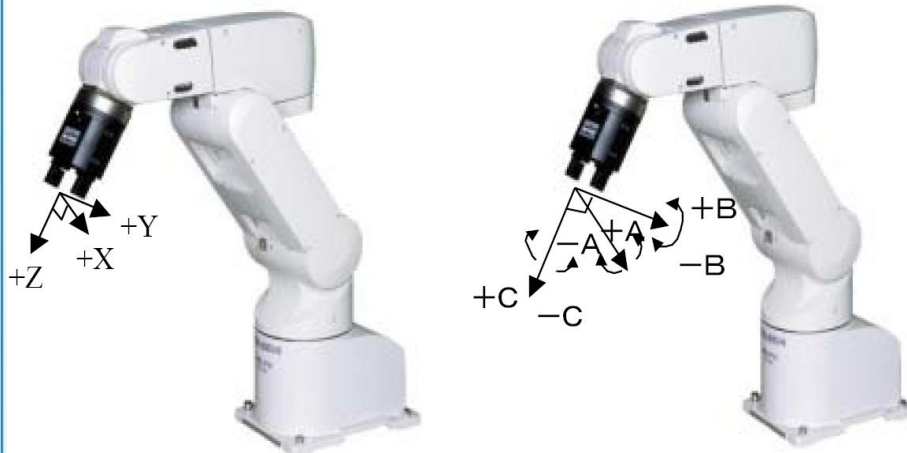


### TOOL jog

Adjusts the coordinates of each axes along the direction of the hand tip.

The X,Y, And Z axis coordinates are adjusted in mm units.

The A,B, and C axis coordinates are adjusted in angle units.

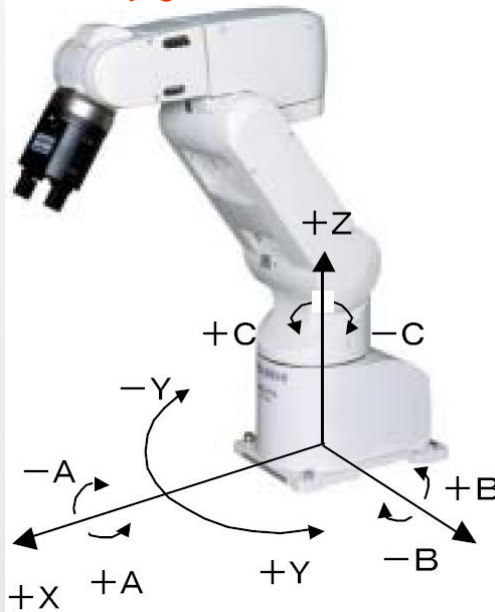


### CYLINDER jog

Adjusting the X-axis coordinate moves the hand in the radial direction away from the robot's origin. Adjusting the Y-axis coordinate rotates the arm around the J1 axis. Adjusting the Z-axis coordinate moves the hand in the Z direction of the robot coordinate system.

Adjusting coordinates of the A, B, and C axes moves the hand in the same way as in XYZ jog feed.

The X and Z axis coordinates are adjusted in mm units. The Y, A, B, and C axis coordinates are adjusted in angle units.



### Programming

- Programming by MELEA BASIC IV using COSIROP/COSIMIR

### Characters with special meanings (1)

#### Apostrophe (')

In a robot program comment lines are indicated by an apostrophe. The comment lines are transferred to the drive unit.

Example:

100 'Pick-up position



#### Asterisk (\*)

The asterisk defines jump marks in a robot program.

Example:

110 \*TABLE1



### Characters with special meanings (2)

#### Comma (,)

The comma serves as separator of several specified components of parameters.

Example:

100 P50 = (450, 100, 300, ...)



#### Dot (.)

For multiple data like positional data the dot serves as separator of the single components.

Example:

110 M10 = P10.X



### Characters with special meanings (3)

#### Space ( )

Between instructions and individual data, and after line numbers a space must be placed.

Example:

100 \_ MOV \_ P10



### Declaration of variables

Names for variables of the type position, joint, arithmetic, and character string begin with a certain character.

The rule is:

P = Positional variable

J = Joint variable

M = Arithmetic variable

C = Character string

Position P	Joint J	Arithmetic M	Character string C
P1	J100	M10	C30\$
P124	J100.J1	M99	C\$[M5+4]
P100.X	J10.J6	M[M6+3]	
P110.Z			
P[M5+3]			
P[M10].Z			

## Positions variables

Variables whose names begin with character **P** are considered position variables. If it is defined by the **DEFPOS** instruction, it is possible to specify a name beginning with a character other than **P**.

It is possible to reference individual coordinate data of position variables. In this case, add "." and the name of a coordinate axis, e.g. "X," after the variable name.

P1.X, P1.Y, P1.Z, P1.A, P1.B, P1.C, P1.L1, P1.L2

The unit of the angular coordinate axes A, B, and C is radians, U set the DEG function to convert it to degrees.

Example)

**P1=(110,-227,-148,45,180,0,0)**

M1=P1.X (Unit:mm)

M2=DEG(P1.A) (Unit:degree)

DEGPOS L10

MOVL10

### Positional

The syntax for position constants is as shown below.

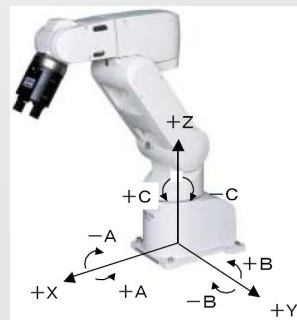
( 100, 100, 300, 180, 0, 180, 0, 0 ) ( 7, 0 )

- structure flag 2 (multi-rotation data)
- structure flag 1 (posture data)
- L2 axis (additional axis 2)
- L1 axis (additional axis 1)
- C axis
- B axis
- A axis
- Z axis
- Y axis
- X axis

Posture axes of the robot (degree)

Coordinate values of the hand tip (mm)

**P1 = ( 300, 100, 400, 180, 0, 180, 0, 0 ) ( 7, 0 )**





## Joints variables

A character string variable should start with **J**. If it is defined by the **DEFJNT** instruction, it is possible to specify a name beginning with a character other than **J**. It is possible to reference individual coordinate data of joint variables. In this case, add "." and the name of a coordinate axis, e.g. "J1," after the variable name.

JDATA.J1, JDATA.J2, JDATA.J3, JDATA.J4, JDATA.J5, JDATA.J6, JDATA.J7, JDATA.J8

The unit of the angular coordinate axes A, B, and C is radians. Use the DEG Function to convert it to degrees.

Example)

**JSTARAT=(10,30,90,0,90)**

JDATA=JSTARAT

DIMJ3(10)

M1=J1.J1 (Unit:radian)

M2=DEG(J1.J2)

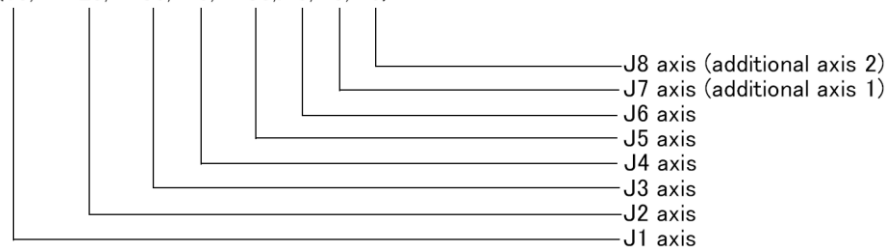
DEFJNTK10

MOVK10

### Joint

The syntax for the joint is the following:

(10, -20, 90, 0, 90, 0, 0, 0)



Example)

6axisrobot

J1=(0,10,80,10,90,0)

6axis+Additional axis

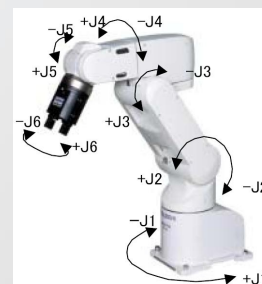
J1=(0,10,80,10,90,0,10,10)

5axisrobot

J1=(0,10,80,0,90)

5axis+Additional axis

J1=(0,10,80,0,90,0,10,10)



## Programming: MELFA BASIC IV

### SYNTAX

10 MOV P1 WTHM\_OUT(17)=1

↑    ↑    ↑    ↑  
1.   2.   3.   4.

1. -Line Nos. can be any integer from 1 to 32767. One line can have up to 127 characters
2. -Command statement
3. -Command parameter
4. -Appended statement (MOV, MVS, MVR, MVR2, MVR3, MVC)

24

## Motion Commands

## Joint interpolation movement(MOV)

### Joint interpolation movement (MOV)

The robot moves with joint axis unit interpolation to the designated position. (The robot interpolates with a joint axis unit, so the end path is irrelevant.)

**MOV**- The robot moves to the designated position with joint interpolation. An appended statement **WTH** or **WTHIF** can be designated.

**MOVP1** 'Moves to P1.

**MOVP1+P2** ' Moves to the position obtained by adding the P1 and P2 coordinate elements.

**MOVP1,-50** ' Moves from P1 to a position retracted 50mm in the hand direction.

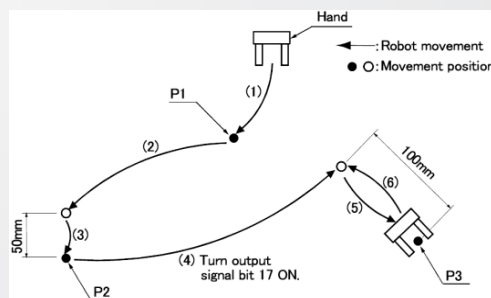
**MOVP1 WTH M\_OUT(17)=1.**

' Starts movement toward P1, and simultaneously turns output signal bit 17 ON.

**MOVP1 WTHIF M\_IN(20)=1,SKIP**

'If the input signal bit 20 turns ON during movement to P1, the movement to P1 is stopped, and the program proceeds to the next stop.

Program	Explanation
10 MOV P1	' (1) Moves to P1 with joint interpolation.
20 MOV P2, -50	' (2) Moves from P2 to a position retracted 50 mm in the hand direction
30 MOV P2	' (3) Moves to P2 with joint interpolation
40 MOV P3, -100 WTH M_OUT(17) = 1	' (4) Starts movement from P3 to a position retracted 100 mm in the hand direction and turns ON output signal bit (at the same time) .
50 MOV P3	' (5) Moves to P3
60 MOV P3 ,-100	' (6) Returns from P3 to a position retracted 100mm in the hand direction
70 END	' Ends the program



## Linear interpolation movement(MVS)

The end of the hand is moved with linear interpolation to the designated position.

**MVS** -The robot moves to the designated position with linear interpolation. An appended statement WTH or WTHIF can be designated.

**MVS** - The robot moves to the designated position with linear interpolation. An appended statement WTH or WTHIF can be designated.

Statement example

**MVS P1.** ' Moves to P1

**MVSP1+P2** ' Moves to the position obtained by adding the P1 and P2 coordinate elements.

**MVSP1,-50** ' Moves from P1 to a position retracted 50mm in the hand direction.

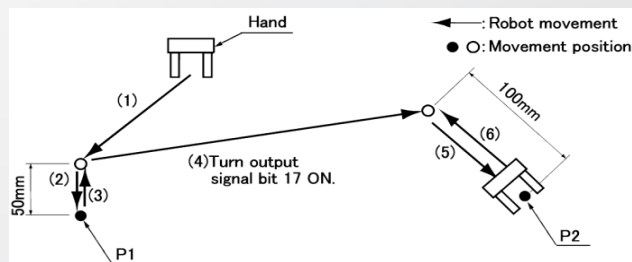
**MVSP1 WTH M\_OUT(17)=1**

' Starts movement toward P1, and simultaneously turns output signal bit 17 ON.

**MVSP1 WTHIF M\_IN(20)=1,SKIP**

' If the input signal bit 20 turns ON during movement to P1, the movement to P1 is stopped, and the program proceeds to the next stop.

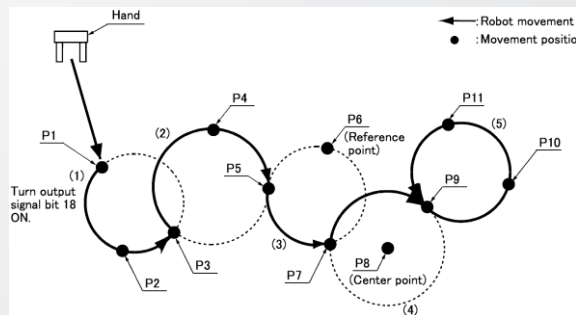
Program	Explanation
10 MVS P1, -50	' (1) Moves with linear interpolation from P1 to a position retracted 50mm in the hand direct.
20 MVS P1	' (2) Moves to P1 with linear interpolation
30 MVSP1,-50	' (3). Moves with linear interpolation from the current position (P1) to a position retracted 50 mm in the hand direction
40 MVS P2, -100 WTH M_OUT(17) = 1	' (4) Output signal bit 17 is turned on at the same time as the robot starts moving
50 MVS P2	' (5) Moves to P2 with linear interpolation
60 MVSP2,-100	' Moves with linear interpolation from the current position (P2) to a position retracted 100 mm in the hand direction
70 END	' Ends the program

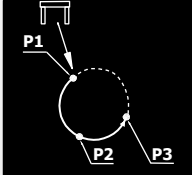
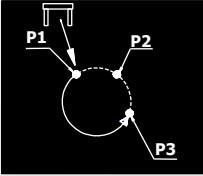


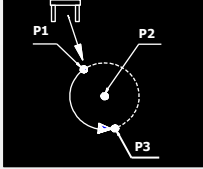
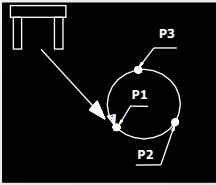
## Circular interpolation movement (MVR)

### Circular interpolation movement

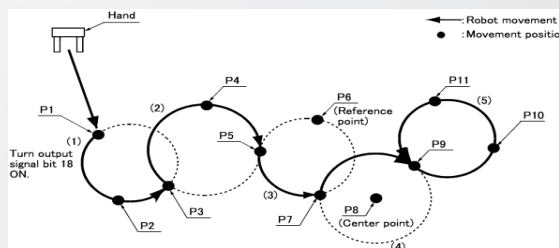
The robot moves along an arc designated with three points using 3D circular interpolation. If the current position is separated from the start point when starting circular movement, the robot will move to the start point with linear operation and then begin circular interpolation.



Command	Explanation
MVR	Designates the start point, transit point and end point, and moves the robot with circular interpolation in order of <b>the start point → transit point → end point</b> . An appended statement WTH or WTHIF can be designated
	<b>MVR P1,P2,P3</b> 
MVR2	Designates the start point, end point and reference point, and moves the robot with circular interpolation <b>from the start point → end point</b> without passing through the reference point. An appended statement WTH or WTHIF can be designated
	<b>MVR2 P1,P3,P2</b> 

<b>MVR3</b>	Designates the start point, end point and center point, and moves the robot with circular interpolation <b>from the start point to the end point</b> . The fan angle from the start point to the end point is 0 deg. < fan angle < 180 deg. An appended statement WTH or WTHIF can be designated
	<b>MVR3 P1, P3, P2</b> 
<b>MVC</b>	Designates the start point (end point), transit point 1 and transit point 2, and moves the robot with circular interpolation in order of <b>the start point → transit point 1 → transit point 2 → end point</b> . An appended statement WTH or WTHIF can be designated.
	<b>MVC P1,P2,P3</b> 

Program	Explanation
<b>10 MVR P1, P2, P3 WTH M_OUT(18) = 1</b>	' (1) Moves between P1 . P2 . P3 as an arc. The robot current position before movement is separated from the start point, so first the robot will move with linear operation to the start point. (P1) output signal bit 18 turns ON simultaneously with the start of circular movement.
<b>20 MVR P3,P4,P5</b>	' (2) Moves between P3 . P4 . P5 as an arc.
<b>30 MVR2 P5,P7,P6</b>	' (3) Moves as an arc over the circumference on which the start point (P5), reference point (P6) and end point (P7) in the direction that the reference point is not passed between the start point and end point.
<b>40 MVR3 P7,P9,P8</b>	' (4) Moves as an arc from the start point to the end point along the circumference on which the center point (P8), start point (P7) and end point (P9) are designated.
<b>50 MVC P9,P10,P11</b>	' (5) Moves between P9 . P10 . P11 . P9 as an arc. The robot current position before movement is separated from the start point, so first the robot will move with linear operation to the start point.(1 cycle operation)
<b>60 END.</b>	' Ends the program



## Motion types: Continuous movement (CNT)

### Continuous movement (CNT)

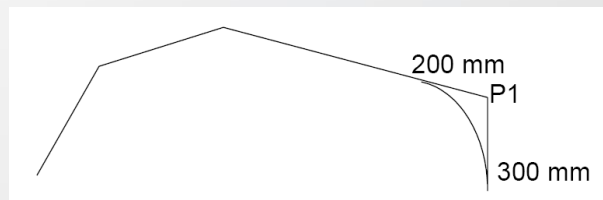
The robot continuously moves to multiple movement positions without stopping at each movement position. The start and end of the continuous movement are designated with the command statement. The speed can be changed even during continuous movement.

**CNT** denotes the start and end of the continuous movement.

**CNT1** denotes the start and end of the continuous movement.

**CNT 1, 200, 300**-Designates the start of the continuous movement, and designates that the start point neighbourhood distance is 200mm, and the end point neighbourhood distance is 300mm

**CNT0** denotes the end of the continuous movement.



```

10 CNT 0 ' invalidate continuous movement
20 MOV P1 ' axis interpolation to position 1
30 MOV P3 ' axis interpolation to position 3
40 CNT 1,200,300 ' validate continuous movement
50 MVS P5 ' linear interpolation to position 5
60 CNT 0 ' invalidate continuous movement
70 END ' program end

```

## Acceleration/deceleration time and speedcontrol

The percentage of the acceleration/deceleration in respect to the maximum acceleration/deceleration, and the movement speed can be designated.

<b>ACCEL</b>	Designates the acceleration during movement and the deceleration as a percentage (%) in respect to the maximum acceleration/deceleration speed. or disabled
<b>OVRD</b>	Designates the movement speed applied on the entire program as a percentage (%) in respect to the maximum speed
<b>JOVRD</b>	Designates the joint interpolation speed as a percentage (%) in respect to the maximum speed
<b>SPD</b>	Designate the linear and circular interpolation speed with the hand end speed (mm/s)
<b>OADL</b>	This instruction specifies whether the optimum acceleration/deceleration function should be enabled

<b>ACCEL.</b>	Sets both the acceleration and deceleration to 100%.
<b>ACCELL 60,80</b>	Sets the acceleration to 60% and the deceleration to 80
<b>OVRD 50</b>	Sets the joint interpolation, linear interpolation and circular interpolation to 50% of the maximum speed.
<b>JOVRD 70</b>	Set the joint interpolation operation to 70% of the maximum speed.
<b>SPD 30</b>	Sets the linear interpolation and circular interpolation speed to 30mm/s.
<b>OADL ON</b>	This instruction enables the optimum acceleration/deceleration function.

10 ACCEL 100,50      ' 100 means 100% = 0.2s acceleration;

' 50 means 200% = 0.4s deceleration

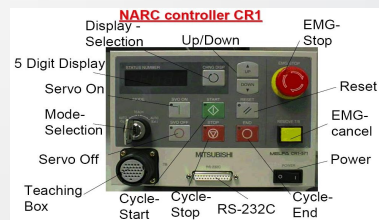
20 MOV P1 ' axis interpolation to position 1

30 MOV P2 ' axis interpolation to position 2

$$t = \frac{100\%}{A [\%]} \cdot 0,2s \quad \text{Where } A \text{ --ACCEL command's parameter}$$



## SPEED evaluation:



### For joint interpolated movement

**Movement speed during joint interpolation Controller (T/B) setting value<sup>x</sup>**  
**OVRD** command setting value<sup>x</sup> **JOVRD** command setting value.

### For linear interpolated movement

**Movement speed during linear and circular interpolation Controller (T/B) setting value<sup>x</sup>** **OVRD** command setting value<sup>x</sup> **SPD** command setting value.

## Hand control

Command	Explanation
<b>HOPENx</b>	Opens the designated hand x
<b>HCLOSEx</b>	Close the designated hand x
<b>DLY xxx</b>	Wait for the xxx seconds for the completion a previous task

Command	Explanation
<b>HOPEN 1</b>	Opens hand No1
<b>HOPEN 2</b>	Opens hand No1
<b>HCLOSE 1</b>	Closes hand No1

## Timer (DLY)

The program can be delayed by the designated time, and the output signal can be output with pulses at a designated time width

Syntax:

**DLY value** where value -time value (from 0.01s)

Ex

**100 DLY 1** - ' Waits for only 1second

**110 M\_OUT(4)=1 DLY 0.5** - 'Turns on output signal bit 10 for only 0.5 seconds.

**120 HOPEN 1**

**130 DLY 0.5** - - ' Waits for only 0.5 seconds

## Inputting and outputting external signals

This section explains the general methods for signal control when controlling the robot via an external device (e.g., PLC)

### Input signals

Signals can be retrieved from an external device, such as a programmable logic controller. The input signal is confirmed with a robot status variable (M\_IN(),

### Output signals

Signals can be output to an external device, such as a programmable logic controller. The signal is output with the robot status variable (M\_OUT(), etc.)

Command	Description
<b>M_IN(xx)</b>	returns the value of the input signal
<b>M_OUT(xx)</b>	writes or references external output signal
<b>WAIT</b>	Waits for the input signal to reach the designated state
<b>CLR</b>	Clears the general-purpose output signal according to the output signal reset pattern in the parameter

## **13      Appendix C: VideoStrobe & VideoFlood LEDs**

- VideoStrobe and VideoFlood LEDs.
- LEDs arrays specifications.
- VideoFlood LED light comparison table.

### 13.1 VideoStrob and VideoFlood LEDs


**Visual Instrumentation Corporation**  
 High-Speed Imaging Technology

## VideoStrobe & VideoFlood LEDs...

### With New Super Bright LED Technology



Model 900410 4 x 6 LED Array

VIC has continued its development of high intensity LED products. This data sheet introduces a new generation of **Super-Bright** LEDs with 2.5 times greater illumination than our previous models. This is achieved by retaining single die point-source technology for maximum efficiency!

In addition, a family of interchangeable collimators provide three beam-spreads of 8°, 21°, and 29°. This allows user flexibility by interchanging collimators to achieve a desired beam-spread. Light output is enhanced by these software optimized aspheric profile lenses with 85% collection efficiency of the total luminous flux produced by each LED. The very high illumination level of these LEDs rival tungsten-halogen lighting with virtually no heat in the beam.

#### Benefits Include:

- ✦ High Efficiency, 250 Lumens per LED
- ✦ Strobe rates to 10,000Hz @ 1μ sec.
- ✦ Long life, more than 25,000 hours
- ✦ Ruggedized for Hi-g applications
- ✦ Modeling light for camera set-up
- ✦ Hardwire or WiFi Remote Control
- ✦ Cool Illumination
- ✦ Very low EMI & RFI
- ✦ No UV radiation
- ✦ Low voltage



Model 900430 1 x 12 LED Array



Model 900415 Single LED Array



Model 900420 Triple LED Array



Model 201090A Hi-g Controller  
Operates on 28 VDC



Model 900405 3 x 4 LED Array

1110 West Avenue L-12, Unit 2, Lancaster, California, USA 93534-7039 Revised December 2009  
 Phone: (661) 945-7999 FAX: (661) 723-5667 E-mail: [visinst@earthlink.net](mailto:visinst@earthlink.net) Website: [www.visinst.com](http://www.visinst.com)  
Specifications subject to change without notice © 2010 Visual Instrumentation Corp. All Rights Reserved

Figure 13.1: Video Strobe - Flood Controller and 3-by-4 LED Array used during laboratory experiment 03. Adapted from VideoStrobe & VideoFlood LEDs by Visual Instrumentation Corporation 2009, Adapted with permission.

## 13.2 LEDs Arrays Specifications

### Specifications for all LED Models:

**LEDs:** Super Bright white LEDs @ 6000°K  
**Collimators:** 8°, 21° or 29° beam angle, interchangeable  
**Strobe Frequency:** Up to 10,000Hz, limited by pulse width  
**Flash Duration:** Adjustable, 1 to 500 micro-seconds  
**Video Sync:** Accepts a 3.3 to 5.0 Volt trigger pulse  
**Controller Power:** 100-240VAC 50/60Hz, also 12 & 28VDC  
 See Controllers for details...

#### Model 900410 4 x 6 LED Array

**Number of LEDs:** 24 Super Bright white LEDs  
**Luminous output:** 6,000 total lumens  
**Size:** 8.0" W x 5.5" H x 2.125" D (203 x 138 x 54mm)  
**Weight:** 2.8 pounds (1.36kg)  
**Mounting:** 1/4"-20 threaded holes, 6 locations; 2 top, 2 bottom and 1 on each side  
**Current Draw:** 4.0 amps

#### Model 900430 1 x 12 LED Array

**Number of LEDs:** 12 Super Bright white LEDs  
**Luminous output:** 3,000 total lumens  
**Size:** 14.75" W x 1.5" H x 2.25" D (37.47 x 3.81 x 5.59cm)  
**Weight:** 2.0 pounds (0.90kg)  
**Mounting:** Three 1/4"-20 and two 3/8" x 16 threaded holes  
**Current Draw:** 2.0 amps

#### Model 900405 3 x 4 LED Array

**Number of LEDs:** 12 Super Bright white LEDs  
**Luminous output:** 3,000 total lumens  
**Size:** 4.5" W x 3.0" H x 2.0" D (114.30 x 76.20 x 50.80mm)  
**Weight:** 15.4 ounces (380 grams)  
**Mounting:** Three 1/4"-20 threaded holes on bottom surface  
**Current Draw:** 2.0 amps

#### Model 900420 3 LED Circular Array

**Number of LEDs:** 3 Super Bright white LEDs  
**Luminous output:** 750 total lumens  
**Size:** 2.85" L x 2.25" Dia. (72.40 x 57.15mm)  
**Weight:** 8.5 ounces (241 grams) plus mounting base  
**Mounting:** Single 1/4"-20 threaded hole  
**Current Draw:** 1.0 amp

#### Model 900415 Single LED Module

**Number of LEDs:** 1 Super Bright white LED  
**Luminous output:** 250 lumens at 6000°K  
**Foot-Candles-LUX:** 7500 ft-candles at 12" with 8° collimator (80,700 lux)  
**Size:** 3.25" L x 1.63" Dia. x 2.11" H. (82.6 x 41.5mm x 53.9)  
**Weight:** 5.7 ounces (155.9 grams)  
**Mounting:** Single 1/4"-20 threaded hole  
**Current Draw:** 1.0 amp

### Description

Each Super Bright LED produces a luminous output of 250 lumens ( $\pm 10\%$ ) at a nominal drive current of 1.1 amp. Higher drive currents beyond 100% produce no significant increase in intensity and can lead to premature LED failure.

For some applications Red LEDs are better suited for monochrome cameras. Each LED produces 180 lumens or 225 lumens with a 25% increase in drive current.

A selection or binning process matches each Super Bright white LED for: **1)** brightness and **2)** color temperature. The result is a smooth light beam without halos, rings, hot spots or shadows. To further enhance LED performance a series drive circuit is utilized to reduce current consumption by up to 83% without compromising maximum LED luminous output. Note, the same LED arrays are used for both continuous and strobe light modes.

### Controllers

Two types of controllers are available, **VideoStrobe** and **VideoFlood** models. Strobe controllers provide 7-selectable durations of 1 to 500 microseconds per flash and two continuous or flood light intensities of 50% and 100%. A video camera's shutter pulse signal is required to sync the strobe with each image frame at flash rates up to 10,000 Hz. **VideoFlood** models have 4-selectable (flood only) intensity levels: 50%, 70%, 85%, and 100% with no provision for strobe output.

Both AC and DC powered HI-g controllers are available and accept: 100-240VAC power 50/60Hz or 28-volts DC. 12-volts DC powers the controller for up to 4 single LED modules. Options include various remote control capabilities including hard-wire and WIFI. All models are CE and RoHS compliant.

**VideoFlood** operation has a nominal 10-minute time limitation at 75%, 85% and 100% intensity levels. There is no time-limit in strobe or continuous flood modes at 50% intensity.

Customer selected strobe rates are available on request. Contact VIC for further details and see price schedules pages I-2A and I-2B for cost information.



Figure 13.2: LED Array specifications. Two 3-by-4 LED Array Model 900405 were used during laboratory experiment 03. Adapted from VideoStrobe & VideoFlood LEDs by Visual Instrumentation Corporation 2009, Adapted with permission.

### 13.3 VideoFlood LED Light Comparison Table

Table 13.1

*Model 900405 3-by-4 Super Bright LEDs Array*

Distance to Subject	Beam Spread		Illumination in Foot Candles & LUX	Drive Current to each LED
12 inches/30.50 cm	5.00" x 5.00"	2.70 cm	8000 – 86080	1.1 amp.
24 inches/60.96 cm	9.50" x 8.50"	24.13 cm	2255 – 27413	1.1 amp.
36 inches/91.44 cm	14.50" x 12.00"	35.56 cm	1150 – 12374	1.1 amp.
48 inches/121.90 cm	15.75" x 21.00"	3.54 cm	650 – 6995	1.1 amp.

Note. This table was adapted from *VideoFlood LED Light Comparison Table* by Visual Instrumentation Corporation, May 2011. Adapted with permission.

Table 13.1 presents the values of the model **900405, 3 x 4 array of 3-watt Super Bright LEDs** with 4 centre 29° lenses and 8 perimeter 21° lenses. This values are used for comparison with the values obtained with the illumination equation.

The formula suggested by Visual Instrumentation Corporation to calculate lightning in foot - candles for high – speed framing cameras is give by (13.3.1) as follows:

$$I = \frac{K \times A^2}{ISO \times ET} [foot\ candles] \quad (13.3.1)$$

Where :

- I is the illumination in foot - candles.
- K is a constant with value 25.
- A is the area to be illuminated in square feet.
- ISO is the image sensor speed.
- ET is the exposure time in decimal seconds (1 decimal second = 0.864 seconds ).

The formula to convert Foot-Candles (FC) to LUX is as follows:

$$Illumination_{(LUX\ units)} = 10.76 \times FC \quad (13.3.2)$$

While the formula to convert candelas to lumens is given by:

$$Illumination_{(Candela\ units)} = 4 \times Lumens \quad (13.3.3)$$

## **14      Appendix D: Phantom v. 9.1 Data Sheet**

- Phantom v. 9.1 Data Sheet.
- Phantom v. 9.1 Maximum Recording Speed vs. Image Size.
- V-Series Lens Shutter Data Sheet – Mechanical Shutter.
- V-Series Lens Shutter Data Sheet – Break-out-Box.





## Phantom v9.1

Provides 14-bit image depth, and 1,000 frames per second at a full resolution of 1,632 x 1,200 pixels

### HIGH RESOLUTION, HIGH SPEED, HIGH SENSITIVITY

With its CMOS sensor, the Phantom v9.1 offers 1,000, 14-bit, frames per second at a full resolution of 1,632 x 1,200 active pixels. Like its predecessor, the Phantom v9.0, the v9.1 preserves such feature as Gigabit Ethernet for camera control and file transfer, and the ability to segment a significantly larger DRAM image memory for multiple cine recording. In addition to these features, the v9.1 has added an HD-SDI interface, and the ability to continuously data stream 8-bit or 12-bit images.



- Full frame 4:3 aspect ratio CMOS sensor composed of 1,632 x 1,200 pixels
- 14-bit image depth (standard)
- 1,000 frames per second full resolution, up to 153,846 fps maximum
- “CAR” (Continuously Adjustable Resolution) in 96 x 8 pixel increments
- 2400 ISO/ASA monochrome, 600 ISO/ASA color sensitivity equivalency
- Global on-chip shuttering to 2 microseconds
- “EDR” Extreme Dynamic Range™ exposure control
- Auto Exposure control
- Up to 24 Gigabytes DRAM, 24 Gigabytes non-volatile flash memory (optional)
- IRIG-B timing capture, modulated or unmodulated, IRIG lock w/phase shift
- Continuous video output (NTSC, PAL, HD/SDI 720p, 1080p, 1080i, 1080psf)
- Optional continuous data streaming up to 500 fps (8-bits), 350 fps (12-bits)
- Automated multiple session recording for remote unmanned operation
- Gigabit Ethernet or RS232 control

Datasheet - Subject to Change

Revision: 10.8.2007

Figure 14.1: Phantom v. 9.1 Data Sheet p1/3. Adapted from Phantom v9.1 by Vision Research Inc, 2007. Adapted with permission.





## V9.1 Specifications

### FEATURES

*Auto Exposure*

*"EDR" Extreme Dynamic Range™*

*Continuous data streaming (optional)*

*Continuous recording*

*Pre-trigger recording*

*On chip global shuttering*

*Strobe sync*

*Segmented image memory*

*Continuous color HD-SDI video output*

*IRIG-B timing capture with phase shift*

*10/100/Gigabit Ethernet*

**Sensor:** 1,632 x 1,200 pixel SR-CMOS sensor.

**Image Bit Depth:** 14-bit (standard)

**Sensitivity:** 2400 ISO/ASA mono-chrome, 600 ISO/ASA color

**Frames per Second (FPS):** Full sensor; to 1,000 fps

**Allocated formats:** to 153,846 fps with "CAR" (Continuous Adjustable Resolution) feature

**Exposure Time:** Variable, independent of sample rate (fps), to 2 microseconds

**Trigger:** Continuously variable pre/post

**Imager Control:** 10/100/Gigabit Ethernet, or RS232 serial interface

**Preview and Focusing:** Via computer monitor or continuous video out

**Lens Mounts:** Nikon mount standard.

Many other lens mounts available, including C-mount

**INPUTS/OUTPUTS:** via integrated quick-release connector:

**Trigger:** Rising/falling TTL pulse w/filter, or switch closure

**Sync Image:** TTL pulse

**Event Marker:** TTL pulse or switch closure

**Ready Signal:** TTL pulse

**IRIG-B Timing:** IRIG-B code, modulated or unmodulated input, with IRIG-B output, lock, and variable phase shift

**Continuous Data Streaming:** Up to 500 fps (8-bits), 350 fps (12-bits)

**Strobe Sync:** TTL Pulse

**RS232**

**Network:** 10/100/Gigabit Ethernet

**Video out:** NTSC, PAL, and HD-SDI (720p, 1080p, 1080i, 1080psf at 24, 25, 59.9, and 60 fps)

**Power:** 24VDC/1.5 Amp

### MEMORY

**Standard:** 6 Gigabytes integral image memory records 3,192 images for 3.19 sec of continuous recording at 1,000 fps, full format (8-bits) or 1,824 images for 1.82 sec of continuous recording at 1,000 fps, full format (14-bits). Longer recording times for lower sample rates and allocated formats.

**Optional:** 12 Gigabytes integral image memory continuously records 6,427 images for 6.43 sec. (8-bits) or 3,762 images for 3.67 sec (14-bits) at 1,000 fps full frame, and 24 Gigabytes will record 12,899 images for 12.9 sec (8-bits) or 7,370 images for 7.37 sec (14-bits) at 1,000 fps full frame.

**Optional:** Non-Volatile Flash Memory, up to 24Gigabytes.

### ENVIRONMENTAL

**Ambient Temperature**

-14°F to +122°F (-10°C to +50°C)

**Maximum humidity:** 80%, non-condensing, at 5°C

### SOFTWARE

Phantom® operates in Windows XP Pro or Vista environments with familiar commands found in familiar places. Standard functions include:

**Acquisition:** Image capture, IRIG-B timing capture & standard time annotation. Field of view & focus. Sample rate & aspect ratio selection. Shutter speed. Histogram. Brightness, contrast, & gamma adjust. Trigger modes. Continuous record. Save & recall setups.

**Analytical playback:** Immediate playback of cine. Variable playback speed in forward or reverse, including freeze frame & endless loop. Random Go-to-Image. View single images at random from any cine. Tile/cascade multiple images on one screen. Timing data displayed with each image. Cine editor. Multi Cine Viewer.

**Measurements:** Linear or angular measurements. English and metric units. Generate Velocity, RPM, or 100 data points per measurement reports. Report files & images are compatible with Phantom, TEMA Starter Software or any spreadsheet software, and image analysis software such as TrackEye®, Image Express®, or Falcon®.

**Image processing:** Smooth, sharpen, pseudocolor, negative image, and edge detection. Brightness, contrast & gamma adjust. 3x3 and 5x5 filter matrix for custom image processing.

**File management:** Organize, save, compress and export cines, or single images. File formats are compatible with most word processing, desktop, publishing, and presentation software.

### DIMENSIONS

**Size:** 4.3 x 4.0 x 9.5 inch (HWD) (10.9 x 10.16 x 24.13 cm) (HWD)

**Weight:** 7 lbs (3.18kg)

**Power:** 24VDC/1.5 Amp

**Mounting:** 1/4-20 inch and four 10-32 threaded hole pattern in base and top

**Mounting Axis:** Any position

**Country of Origin:** The United States of America

### STANDARD ACCESSORIES

Phantom® software, Single user license\*

6 Gigabyte integral image memory

Ethernet, Sync output pulse, trigger, pretrigger, video out, and IRIG-B

110/220VAC -28VDC International Power Adapter, 12 foot (3.7 m) power cord

One year service contract included

### QUESTIONS?

For technical assistance, systems integration, custom options, or information on imaging techniques or training please call us toll free:

1.800.RESOLUTION

(US & Canada 1.800.737.6588)

For the most up-to-date information, specifications and options, please visit our website:

[www.visionresearch.com](http://www.visionresearch.com)

All specifications are subject to change. (Oct-07)

Figure 14.2: Phantom v. 9.1 Data Sheet p2/3. Adapted from Phantom v9.1 by Vision Research Inc, 2007. Adapted with permission.

## Phantom v9.1 Maximum Recording Speed vs. Image Size

The Phantom v9.1 camera system records up to 1,016 frames per second using the full 1632 x 1200 pixel CMOS imaging sensor array. The operator may specify other aspect ratios to increase recording speeds or extend recording times.

The chart below details some of the Phantom v9.1 aspect ratio choices available from a resolution and sample rates pull down menu. Using the CAR (Continuous Adjustable Resolution) feature, resolution/speed settings between these values are continuously adjustable in 96 x 8 pixel increments.

RESOLUTION	RATE
1632 x 1200	1,016
1632 x 960	1,268
1632 x 480	2,520
1632 x 240	4,975
960 x 960	1,972
960 x 480	3,906
960 x 240	7,648
480 x 480	6,420
480 x 240	12,422
480 x 120	23,391
480 x 64	36,603
96 x 96	51,948
96 x 48	81,632
96 x 32	100,000
96 x 16	129,032
96 x 8	153,846



All specifications are subject to change. (Oct-07)

Vision Research, Inc.  
T/+1 973-696-4500 F/+1 973-696-0560  
100 Dey Rd  
Wayne, NJ 07470 USA

Datasheet - Subject to Change

Revision: 10.8.2007

Figure 14.3: Record speed vs. Image Resolution p3/3. Adapted from Phantom v9.1 by Vision Research Inc, 2007. Adapted with permission.

**ViSiON**  
RESEARCH

An **AMETEK** Company

**PHANTOM**  
when it's too fast to see, and too important not to.

## DATA SHEET

### V-Series Lens Shutter

Hands-free CSRs

Remote CSRs

Access to all camera signaling



*Lens Shutter On Camera*

#### Key Benefits and Features:

##### WHEN IT'S TOO FAST TO SEE, AND TOO IMPORTANT NOT TO®

One of the most popular features on our newest Phantom cameras is an internal **mechanical shutter**. It is used to automatically shade the sensor during a Current Session Reference (CSR). This allows the camera operator to do a CSR without manually capping the lens.

Because of the overwhelming popularity of this feature, we've decided to bring it to our existing line of V-Series cameras – the v5.2, v7.3, v9.1 and v10.

To **take advantage of this feature** simply:

- Replace the existing Nikon lens mount with the new V-Series Lens Shutter accessory
- Update the firmware in the camera, if required
- Use our new Active Break-out-Box instead of the existing capture cable
- And, use Phantom software release 670, or later

Everything you need to upgrade your camera and software is included in the **V-Series Lens Shutter Kit**: VRI-EXTSHUTTER-F-MOUNT. Or, if you want Vision Research to do the camera upgrade for you, order VRI-EXTSHUTTER-F-MOUNT-UPG, and return the camera to us. To have the shutter added to a new camera order, specify the option VRI-EXTSHUTTER-F-MOUNT-OPT.

#### V-Series Lens Shutter Accessory:



*Lens Shutter Front*



*Lens Shutter Side*



*Lens Shutter Back*

Figure 14.4: Mechanical shutter p1/2. Adapted from V-Series Lens Shutter by Vision Research Inc, 2010. Adapted with permission.



when it's too fast to see, and too important not to.

## DATA SHEET

### V-Series Lens Shutter

#### How it Works:

The V-Series Lens Shutter simply replaces your existing F-mount on the camera. So, you continue to use your existing lenses. Exchange the existing capture cable with the included break-out-box, and connect the shutter control cable from the break-out-box to the shutter. Now, any time you do a CSR, the shutter will close, the black reference will be performed, and the shutter will reopen, ready for the shot.



Lens Shutter On Camera With Cable

You will want this feature for every existing V-Series camera! The **convenience** of not manually capping the lens during a CSR translates into higher productivity and better image quality. And, it enables doing a CSR in situations where it was previously impossible (with unattended cameras, for example).

The new **Break-out-Box** gives you access to every available signal on the camera's capture cable:

- IRIG-In
- IRIG-Out
- Video Out (75 ohm, composite)
- Trigger
- Event
- Strobe
- F-Sync / A-Trig
- Pre-Trigger / Memgate
- Ready
- Power-In (24V, both a 4-pin Amphenol and 3-pin XLR connector are available)
- Genlock (on some models)

(Pre-Trigger and Memgate still share a connector and must be selected in software.)



**VISION**  
**RESEARCH**  
An **AMETEK** Company

#### Technical Specifications:

Power: 20-36 VDC, 1.2 Watts idle, 2.4 Watts active

Open/Close Time: Less than 1 second

Self Test: On power up

Interface: RS-232, 38400 Baud

MTBF: 40,000/cycles

Shock: 70g sine wave in all three axes, no shutter position displacement when powered

Weight: 13 oz.

Connector: Fischer DG102Z054130

Operating Temperature: -5°C to +45°C

Size: 40 x 80 x 100 mm

#### Focused

Since 1950, Vision Research has been shooting, designing, and manufacturing high-speed cameras. Our single focus is to invent, build, and support the most advanced cameras possible.

100 Dey Road  
Wayne, NJ 07470 USA  
+1.973.696.4500  
phantom@visionresearch.com

[www.visionresearch.com](http://www.visionresearch.com)

All specifications are subject to change without notice. Rev August 2009

Figure 14.5: Break-out-Box p2/2. Adapted from V-Series Lens Shutter by Vision Research Inc, 2010. Adapted with permission.

## **15      Appendix E: MatLab M-Files Code**

- File 1: getNodeData.m.
- File 2: readDataFromXML.m.
- File 3: plot3DPath.m.
- File 4: readMelfaData.m.
- File 5: transformReferential.m.
- File 6: testTransformReferential.m



## 15.1 File 1: *getNodeData.m*

```

%% SEARCH FOR A VARIABLE AND READ ITS DATA FROM AN OPEN CV 2.1 XML FILE
% GIVEN AN XML FILE NAME GENERATED BY OPEN CV 2.1 FILE STORAGE AND THE
% NAME OF A VARIABLE TO BE FOUND (NODE NAME) PERFORM A SEARCH AND IF
% THE NODE EXIST RETRIEVE ALL THE DATA STORED IN ITS 'data' CHILD NODE.
%ARG 1: OPENCV GENERATED XML FILE NAME TO BE READ
%ARG 2: VARIABLE NAME TO BE LOADED

%USAGE EXAMPLE: READ CAMERA'S MATRIX FROM A CALIBRATION FILE
% calibParameters = readDataFromXML( 'CalibS04_CalibrationM1.xml', 'M1' );
%%
function [numbOfSubTags, dataVector] = getNodeData( fileName, tagName )

    %GET DOCUMENT OBJECT MODAL (DOM) NODE
    docNode = xmlread( fileName );

    %GET OPEN CV XML FILE ROOT NODE "<opencv_storage>"
    openCVStorage = docNode.getDocumentElement;

    %GET ALL "<opencv_storage>" SUB NODES
    varNodes = openCVStorage.getChildNodes;

    %GET NUMBER OF SUB NODES
    n = varNodes.getLength -1;
    output = '';

    % SEARCH IF tagName NODE EXIST WITHIN ALL SUB NODES
    varFound = 0;
    fprintf( '    SEARCHING FOR [%s] VAR \n', tagName );
    for i = 0 : n
        %GET EACH NODE ITEM
        varNodeI = varNodes.item(i);

        %IF NODE EXIST CHECK IF IT HAS A CHILD NODE "data"
        if ( varNodeI.getNodeName == tagName )
            fprintf( '    >>> VARIABLE [ %s ] FOUND.\n', tagName );
            varFound = 1;
            if ( varNodeI.hasChildNodes )
                node = varNodeI.getFirstChild;
                nChild = 1;
                dataFound = 0;
                while ~isempty(node)
                    %IF CHILD NODE == 'data' GET ITS CONTENT
                    if strcmpi(node.getNodeName, 'data' )
                        output = string(node.getTextContent);
                        fprintf( '    >>> [ %s ] NODE CHILD[ %i ] HAS DATA.\n', ...
                            tagName, nChild );
                        dataFound = 1;
                        break;
                    else
                        fprintf( '    >>> [ %s ] IS SINGLE VALUE NODE.\n', tagName );
                        %output = string(node.getTextContent)
                        node = node.getNextSibling;
                    end
                    nChild = nChild + 1;
                end
            end
            if ( dataFound == 0 )

```

```

        fprintf( '    >>> [ %s ] HAS NO DATA.\n', tagName );
    end
end
end

%SPLIT STRINGS AND CONVERT TO DOUBLE VALUES
cellC1 = regexp(output, ' ', 'split');
vec1 = str2double( cellC1 );
vec2 = [];
n1 = 0;
status = isnan( vec1 );
for i = 1 : size( status, 2 )
    if ( status(1,i) == 0 )
        n1 = n1 + 1;
        vec2(n1) = vec1( 1, i );
    end
end
end

if( varFound == 0 )
    fprintf( '    VARIABLE [ %s ] NOT FOUND.\n', tagName );
end

numbOfSubTags = n + 1;    %Number of variables listed in the XML file
dataVector     = vec2;    %Data within the subnode child node <data>
end

```

## 15.2 File 2: *readDataFromXML.m*

```

%% READ VARIABLES FROM AN XML FILE GENERATED BY OPEN CV 2.1 FILE STORAGE
% GIVEN AN OPEN CV 2.1 GENERATED XML FILE NAME AND A VECTOR WITH THE
% VAR NAMES TO BE READ FROM THE FILE IT LOADS THE DATA OF EACH VARIABLE.
% ARG 1: OPENCV GENERATED XML FILE WITH DATA TO BE READ
% ARG 2: VECTOR WITH THE VAR NAMES TO BE LOADED TO MATLAB. ALL THE
% VARIABLES NAMES NEED TO HAVE THE SAME LENGHT (BY PADDING WITH SPACES)
% OTHERWISE THE FUNCTION WILL RETURN AN ERROR.

%USAGE EXAMPLE: READ CALIBRATION PARAMETERS FROM A CALIBRATION FILE
% varNames = [ 'M1';'D1';'M2';'D2';'R '; 'T '; 'E '; 'F '];
% calibParameters=readDataFromXML('CalibS04_CalibrationM1.xml', varNames );
%%
function [ varsDataOut ] = readDataFromXML( xmlFileName, vecVarNames )

    %TRANSFORM VECTOR OF CHARS ARRAYS INTO STRING
    %ALL ELEMENTS NEED TO HAVE THE SAME LENGHT
    vars = cellstr( vecVarNames );

    %DEFINE CELL TO STORE EACH VAR NAME BY ROWS
    sz = size( vars, 1 );
    dataIn = cell( sz, 1 );
    for i = 1 : sz
        varToRead = vars{i,1};
        [ nVars, dataOut ] = getNodeData( xmlFileName, varToRead );
        dataIn{i,1} = dataOut;
    end

    %RETURN VAR DATA VECTORS OF VECTORS
    varsDataOut = dataIn;
end

```



### 15.3 File 3: *plot3DPath.m*

```

%% READS POINT/POINTS COORDINATES [X, Y, Z] AND PLOTS ITS 3D PATH
%TAKES A CELL OF ARRAY(S) OF DOUBLES CONTAINING ALL THE TRACKED
%POSITIONS OF ONE POINT OR MULTIPLE POINTS, SEPARATES ITS [Z, Y, Z]
%COORDINATES STORED SEQUENTIALLY [X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3, ...]
%AND PLOTS THE 3D PATH FOR ONE OR ALL POINTS PASSED AS CELL ARGUMENT.
%ARG 1: CELL WITH 3D POINTS TO BE TRANSFORMED AND PLOTTED
%ARG 2: PLOT TITLE
%ARG 3: XML FILE CONTAINING 3 3D POINTS TRACKED OVER TARGET1'S
%ARG 4: PLOT LINE TYPE
%ARG 5: TRANSFORM TYPE (0)-GENERIC REFERENTIAL, (1)-TRANSFORM COORDINATES
%TO THE REFERENTIAL BUILD WITH 3D POINTS TRACKED OVER TARGET1'S, (2)-KEEP
%POINTS RELATED TO THE ORIGINAL REFERENTIAL (CAMERA'S REFERENTIAL).
%%
function plot3DPath( cellOf3DPoints,graphTitle,xmlFile2,lineType,trans )

    %READ NUMBER OF POINTS INPUTED
    cellSize = size( cellOf3DPoints, 1 );
    X = [ ];
    Y = [ ];
    Z = [ ];

    %BUILD X, Y, Z COORDINATE VECTORS FOR ALL POINTS
    for i = 1 : cellSize
        pointI3D = cellOf3DPoints{i,1};
        X(i,:) = pointI3D(1:3:end);
        Y(i,:) = pointI3D(2:3:end);
        Z(i,:) = pointI3D(3:3:end);
    end

    %REDUCE THE NUMBER OF POINT SAMPLES AND PLOT THE RESULTING 3D PATH
    for i = 1 : cellSize
        x1 = X( i, 1 : 10 : end ); %Reduce the number of samples to n/10
        y1 = Y( i, 1 : 10 : end );
        z1 = Z( i, 1 : 10 : end );

        %TRANSFORM COORDINATES SYSTEM
        %trans == 0: Transform 3D points to a generic referential
        %trans == 1: Transform 3D points to robot's referential
        %trans == 2: Keep 3D points related to cameras' referential
        if ( trans == 0 )
            P0 = [ 100 100 1000];
            P1 = [ 200 100 1000];
            Pyz = [ 100 300 1100];
            [x1 y1 z1] = testTransformReferential( P0, P1, Pyz, x1, y1, z1 );
        elseif ( trans == 1 )
            %TRANSFORM COORDINATES REFERENTIAL
            [x1 y1 z1] = transformReferential( xmlFile2, x1, y1, z1 );
        end

        %PLOT 3D PATH
        plot3( x1, y1, z1, lineType );
        title ( graphTitle );
        if ( trans == 0 ) %Test transform
            xlabel('Xt Coordinate [mm]');
            ylabel('Yt Coordinate [mm]');
            zlabel('Zt Coordinate [mm]');
        end
    end

```

```
elseif( trans == 1 ) %Transform referential
    xlabel('Xr Coordinate [mm]');
    ylabel('Yr Coordinate [mm]');
    zlabel('Zr Coordinate [mm]');
elseif ( trans == 2 ) %Keep cameras referential
    xlabel('Xc Coordinate [mm]');
    ylabel('Yc Coordinate [mm]');
    zlabel('Zc Coordinate [mm]');
end
grid on
hold on;
end
end
```

## 15.4 File 4: readMelfaData.m

```

%% READ 3D PATH FROM MELFA BASIC IV OUTPUT FILE
%READ THE 3D PATH OUTPUT FILE GIVEN BY MELFA BASIC IV SOFTWARE.
%OBS: ALL THE COMAS NEED TO BE REPLACED BY A PERIOD AND THE VAR IN THE
%HEADER FILE SHOULD BE FORMATED AS IN THE NEXT LINE:
%Time | J1 | J2 | J3 | J4 | J5 | J6 | J7 | ... | A | B | C | L1 | L2
%ARG 1: MELFA BASIC OUTPUT FILE
%ARG 2: PLOT TITLE

%USAGE EXAMPLE:
%[varNames, varData]=readMelfaData('Lab3 Path11B 50ms.log','MELFA PATH');
%Time = varData{ 1, 1 }; %Get 'Time' column values
%J1 = varData{ 1, 2 }; %Get joint1 values
% ... = ...
%X = varData{ 1, 10 }; %Get end of arm X coordinates column
%Y = varData{ 1, 11 }; %Get end of arm Y coordinates column
%Z = varData{ 1, 12 }; %Get end of arm Z coordinates column
%%
function [ melfaHeader, melfaData ] = readMelfaData( melfaFileName, title1 )

    %OPEN MELFA 3D PATH FILE
    melfaDoc = fopen( melfaFileName );

    %READ MELFA HEADER FILE (NAMES OF THE VARIABLES)
    %Time | J1 | J2 | J3 | J4 | J5 | J6 | J7 | ... | A | B | C | L1 | L2
    melfaHeader1 = textscan( melfaDoc, '%s', 17, 'delimiter', '|' );

    %READ ALL DATA COLUMNS FROM MELFA OUTPUT FILE
    melfaData1 = textscan( melfaDoc, ...
        '%s %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f' );

    %PLOT MELFA 3D PATH
    x = melfaData1{1,10};
    y = melfaData1{1,11};
    z = melfaData1{1,12};
    plot3(x,y,z, '-r')
    grid on
    title( title1 );
    xlabel('X Coordinate [mm]');
    ylabel('Y Coordinate [mm]');
    zlabel('Z Coordinate [mm]');
    text( x(1),y(1),z(1), 'Point_1' );

    %RETURN VAR NAMES AND THEIR DATA(COLUMNS)
    melfaHeader = melfaHeader1;
    melfaData = melfaData1;

    %CLOSE INPUT FILE
    fclose( melfaDoc );

end

```

### 15.5 File 5: transformReferential.m

```

%% TRANSFORM POINTS COORDINATE SYSTEM
%GIVEN A FILE CONTAINING THREE 3D POINTS COORDINATES THAT FORM THE NEW
%COORDINATE SYSTEM IT COMPUTES THE DIRECTION COSINE MATRIX AND TRANSFORMS
%THE 3D POINTS VECTORS [X Y Z] INTO THE NEW REFERENTIAL.
%ARG 1: XML FILE CONTAINING 3 3D POINTS TRACKED OVER TARGET1'S
%ARG 2: 3D POINTS' COORDINATES TO BE TRANSFORMED AND THEN RETURNED
%%
function [ Xout, Yout, Zout ] = transformReferential( xmlFileIn, X, Y, Z )

    %LOAD THREE 3D POINTS TO USE IN THE POINT-LINE-PLANE IMPLEMENTATION
    [ n, Pt0 ] = getNodeData( xmlFileIn, 'POINT0' );
    [ n, Pt1 ] = getNodeData( xmlFileIn, 'POINT1' );
    [ n, Ptyz ] = getNodeData( xmlFileIn, 'POINT2' );
    points = [ Pt0; Pt1; Ptyz ];

    X1 = [];
    Y1 = [];
    Z1 = [ ];
    P0 = zeros(1,3);
    P1 = zeros(1,3);
    Ptyz = zeros(1,3);

    %IN OPENCV THE 3D POINTS COORDINATES ARE SAVED SEQUENTIALLY i.e.
    %P0 = [ x1 y1 z1 x2 y2 z2 ... xn yn zn] THUS FOR EACH POINT (i)THE
    %POINT'S COORDINATES NEED TO BE SEPARATED.
    cellSize = size( points, 1 )
    for i = 1 : cellSize
        pointI3D = points(i,:);
        X1(i,:) = pointI3D(1:3:end);
        Y1(i,:) = pointI3D(2:3:end);
        Z1(i,:) = pointI3D(3:3:end);
    end

    %COMPUTE THE MEAN OF EACH POINT'S COORDINATE TO REDUCE THE ERRORS THAT
    %WOULD BE MORE NOTABLE IF ONLY ONE VALUE FOR EACH COORDINATE WAS USED.
    if ( size(X1,2) > 1 )
        mx = zeros(1,3);
        my = zeros(1,3);
        mz = zeros(1,3);
        for i = 1: size(X1,1)
            Xi = X1(i,:);
            Yi = Y1(i,:);
            Zi = Z1(i,:);
            mx(i)= mean(Xi);
            my(i)= mean(Yi);
            mz(i)= mean(Zi);
        end

        P0(1,:) = [ mx(1,1) my(1,1) mz(1,1) ];
        P1(1,:) = [ mx(1,2) my(1,2) mz(1,2) ];
        Ptyz(1,:) = [ mx(1,3) my(1,3) mz(1,3) ];

        %BUILD DIRECTIONS COSINES USING POINT-LINE-PLANE
        YY = [ P1(1,1)-P0(1,1) P1(1,2)-P0(1,2) P1(1,3)-P0(1,3) ];
        YZ = [ Ptyz(1,1)-P0(1,1) Ptyz(1,2)-P0(1,2) Ptyz(1,3)-P0(1,3) ];
    end

```

```

XX = cross(YY,YZ);
ZZ = cross(XX,YY);
XX = XX/(sqrt(XX(1,1)^2+XX(1,2)^2+XX(1,3)^2));
YY = YY/(sqrt(YY(1,1)^2+YY(1,2)^2+YY(1,3)^2));
ZZ = ZZ/(sqrt(ZZ(1,1)^2+ZZ(1,2)^2+ZZ(1,3)^2));

%BUILD DIRECTION COSINE MATRIX AND TRANSLATION VECTOR
rot = eye(4);
trans = eye(4);
rot(1, 1:3) = XX;
rot(2, 1:3) = YY;
rot(3, 1:3) = ZZ;
trans(1,4) = P0(1);
trans(2,4) = P0(2);
trans(3,4) = P0(3);

%BUILD TRANSFORMATION MATRIX
A = rot * trans;
X0 = [];
Y0 = [];
Z0 = [];

%TRANSFORM POINTS FROM CAMERA COORDINATES SYSTEM TO
%THE COORDINATE SYSTEM DEFINED BY TRAGET 1'S POINTS
for il = 1 : size(X,2)
    pt = A * [ X(1,il); Y(1,il); Z(1,il); 1 ];
    X0(il) = pt(1,1);
    Y0(il) = pt(2,1);
    Z0(il) = pt(3,1);
end

%TRANSLATE THE COORDINATE SYSTEM ORIGIN TO MELFA'S COORDINATE SYSTEM
%ORIGIN WITH PURE TRANSLATION(SEE MELFA RV 2AJ'S BASE DIMENSIONING).
rot1 = eye(4);
trans1 = eye(4);
trans1(1,4) = -88;
trans1(2,4) = -15;
trans1(3,4) = -30;
A1 = rot1 * trans1
for i = 1 : size(X0,2)
    pt = A1 * [ X0(1,i); Y0(1,i); Z0(1,i); 1 ];
    X0(i) = pt(1,1);
    Y0(i) = pt(2,1);
    Z0(i) = pt(3,1);
end
Xout = X0;
Yout = Y0;
Zout = Z0;
else
    printf('NO POINTS TO COMPUTE (R,T) TRANSFORM ARE AVAILABLE')
    printf('PLEASE MAKE SURE %s IS NOT EMPTY', xmlFileIn );
    Xout = X;
    Yout = Y;
    Zout = Z;
end
end
end

```

## 15.6 File 6: testTransformReferential.m

```

%% TRANSFORM 3D POINTS [X,Y,Z] TO A GENERIC REFERENTIAL
%READS THREE GENERIC 3D POINTS (P0,P1,Pyz) AND TRANSFORMS THE GIVEN 3D
%POINTS COORDINATES [X,Y,Z] INTO A NEW REFERENTIAL GIVEN BY (P0,P1,Pyz).
%%
function [Xout,Yout,Zout]=testTransformReferential(P0,P1,Pyz,X,Y,Z)

    %P0 = [ x0 y0 z0 ];
    %P1 = [ x1 y1 z1 ];
    %P2 = [ x2 y2 z2 ];

    %BUILD DIRECTIONS COSINES VECTORS USING POINT-LINE-PLANE METHOD
    YY = [ P1(1,1)-P0(1,1) P1(1,2)-P0(1,2) P1(1,3)-P0(1,3)];
    YZ = [ Pyz(1,1)-P0(1,1) Pyz(1,2)-P0(1,2) Pyz(1,3)-P0(1,3)];
    XX = cross(YY,YZ);
    ZZ = cross(XX,YY);
    XX = XX/(sqrt(XX(1,1)^2+XX(1,2)^2+XX(1,3)^2));
    YY = YY/(sqrt(YY(1,1)^2+YY(1,2)^2+YY(1,3)^2));
    ZZ = ZZ/(sqrt(ZZ(1,1)^2+ZZ(1,2)^2+ZZ(1,3)^2));

    %BUILD DIRECTION COSINES MATRIX AND TRANSLATION VECTOR
    rot = eye(4);
    trans = eye(4);
    rot(1, 1:3) = XX;
    rot(2, 1:3) = YY;
    rot(3, 1:3) = ZZ;
    trans(1,4) = P0(1);
    trans(2,4) = P0(2);
    trans(3,4) = P0(3);

    %BUILD TRANSFORMATION MATRIX
    transf = rot * trans;
    X0 = [];
    Y0 = [];
    Z0 = [];

    %TRANSFORM POINTS FROM CAMERA COORDINATES SYSTEM TO
    %THE COORDINATE SYSTEM DEFINED BY GENERIC 3D POINTS.
    for i1 = 1 : size(X,2)
        pt = transf * [ X(1,i1); Y(1,i1); Z(1,i1); 1 ];
        X0(i1) = pt(1,1);
        Y0(i1) = pt(2,1);
        Z0(i1) = pt(3,1);
    end

    Xout = X0;
    Yout = Y0;
    Zout = Z0;
end

```

## 16 Appendix F: Motion

- Motion.
- Optical flow.

## 16.1 Motion

### 16.1.1 Introduction

For a computer vision system, the ability to deal with the moving and changing objects, changing illumination conditions and changing viewpoints is essential to perform several tasks. The input to a dynamic vision system analysis is a sequence of image frames taken in one of the three situations: The camera is moving and the object is stopped, the object is moving and the camera is stopped or both the camera and the objects are moving.

In a sequence of image frames each frame represents one different image of the scene at a particular instant that permits to understand the motion of a particular object through two processes: *identification* and *modeling*.

The dynamic vision systems main assumptions recalls that the changes in the scene are mainly due the motion of the object and camera simplifying other factors, such as illumination variation , temporal persistence – small movements and spatial coherence, soon discussed in more detail .

Dynamic vision system must detect the objects of interest, detect changes, determine the motion characteristics of the observer and the object, recover the structure of the object and later be able to recognize object in motion.

In this section the researcher started by introducing the formulation that makes possible to identify the unique points to track. This points need to be computed because some tracking methods are *feature based methods* and thus dependent on that features that must be previously computed.

The following part of this section focus in the two tracking techniques implemented in OpenCV for tracking the unidentified objects, the dense tracking techniques and the sparse tracking techniques. In the former case it includes the Horn-Schunck and the Block-Matching method while in the last case is presented one of the most used and efficient tracking techniques, the Lucas-Kanade pyramid method.

### 16.1.2 Corners Identification

The features to be identified in one frame should be unique and easily identified in the subsequent frames. A point is more likely to be unique when strong derivatives exist in orthogonal directions, this features are termed as *corners*. Besides edges presents strong derivatives the points within the same edge are not unique and therefore corners are not edges.

*“Mathematically, the corner definition provided by Harris, relies on the matrix of second order derivatives of the intensities that when applied to all points forms a new Hessian images” (Bradski et. al. 2008, p. 317).*

The Hessian matrix around a point is defined in two dimensions by:

$$H(p) = \begin{bmatrix} \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial x \partial y} \\ \frac{\partial^2 I}{\partial y \partial x} & \frac{\partial^2 I}{\partial y^2} \end{bmatrix}; \quad (16.1.1)$$



For the Harris corner is taken in account a small region around each pixel that defines where the autocorrelation matrix of the second derivative will operate. Such matrix is constructed as follows:

$$M(x, y) = \begin{bmatrix} \sum_{-k \leq i, j \leq K} W_{i,j} I_x^2(x+i, y+j) & \sum_{-k \leq i, j \leq K} W_{i,j} I_x(x+i, y+j) I_y(x+i, y+j) \\ \sum_{-k \leq i, j \leq K} W_{i,j} I_x(x+i, y+j) I_y(x+i, y+j) & \sum_{-k \leq i, j \leq K} W_{i,j} I_y^2(x+i, y+j) \end{bmatrix} \quad (16) \quad .1.2)$$

Where  $W_{i,j}$  is the weighting term that can be uniform or used to create a Gaussian weighting.

Thus if this autocorrelation matrix find two large eigenvalues it means that this particular point contain edges going in at least two different directions within the small region centred at that point. Shi and Tomasi later proposed a more satisfactory algorithm by concluding that good corners result as long as the smaller of the two eigenvalues are greater than a minimum threshold. This two approaches were implemented by `cv::goodFeaturesToTrack()` function as follows:

---

```
void cv::goodFeaturesToTrack( const Mat& image, vector<Point2f>& corners,
    int maxCorners, double qualityLevel, double minDistance, const Mat& mask=Mat(), int
    blockSize=3, bool useHarrisDetector=false, double k=0.04 );
```

---

image – The input image should be 8-bit or 32-bit single channel image.

corners – Output corners vector that will store the 32-bit 2-D points that were detected.

maxCorners – Indicates the maximum number of corners to return.

qualityLevel – Defines the coefficient that multiplied by the best corner quality measure defines the threshold value used to reject the corners which quality measure is less than this value. The values of this parameter should not exceed 1 ( typical value are 0.10 or 0.01 ).

minDistance - Defines a minimum Euclidean distance between corners to avoids that two returned points are within the indicated number of pixels.

mask - Is a Mat that defines a ROI where the corners are detected.

blockSize - As seen before the Harris's autocorrelation matrix of derivatives uses a small region where to operate in order to achieve better results. This argument defines that region

useHarrisDetector – If true the Harris corners are used in place of Shi-Tomasi method.

K – If useHarrisDetector was set to true then K parameter defines the weighting coefficient used in the autocorrelation matrix.

The final result of this routine is an array of pixel locations that are intended to be found in another image.

### 16.1.3 Corners Sub-pixel Accuracy

When working with images for extracting geometric measurements becomes necessary to have more resolution than the provided by `cv::goodFeaturesToTrack()` routine. Thus OpenCV provides an additional routine to compute the exact location of the corners to sub-pixel accuracy, i.e. integer pixel coordinates are computed to real-value coordinates. For example  $p_{pixel}(250, 12) \rightarrow p_{real\ value}(250.59, 12.45)$ .

The routine that computes the real-value pixel coordinates from a given array of integer

coordinates in OpenCV is `cv::cornerSubPix()` and has the following structure:

---

```
void cv::cornerSubPix( const Mat& image, vector<Point2f>& corners, Size winSize, Size
    zeroZone, cv::TermCriteria criteria );
```

---

`image` – Is the input 8-bit single-channel image.

`corners` – Input vector that contains the corners locations in pixels obtained from the previous routine `cv::goodFeaturesToTrack`. The refined pixel coordinates are to the same vector.

`win` - This argument specifies the size of window – `cv::Size( width, height)` from which sort of equations will be generated. This window is centred in the original integer corner location and extends outwards in each direction by the number of pixels specified in `win.width` and `win.height` defining the area to be considered in the system to constraint each of those equations.

`zeroZone` – Analogously to `win`, this argument defines a inner window that will not be considered to constrain the same equations. If no zero zone is desirable then this parameter should be set to `cv::Size( -1, -1 )`.

`criteria` – This argument defines the termination criteria to be reached once the real-value location is found. This criteria can be of type `CV_TERMCRIT_ITER` and of type `CV_TERMCRIT_EPS` or both. The later case will indicate the accuracy required for the sub-pixel values.

## 16.2 Optical Flow

### 16.2.1 Introduction

The motion of objects in 3-D space induces the 2-D motion in the image plan, this motion is what forms the optical flow which carries valuable information for analysing dynamic scenes. Optical flow or image flow is the distribution of velocity, relative to the observer, over the points of an image.

#### Definition

*“Image flow is the velocity field in the image plane due to the motion of the observer, the motion of objects in the scene, or apparent motion which is a change in the image intensity between frames that mimics object or observer motion.” (Jain et. al., 1995, p428)*

Assuming that image flow information is available there are several methods to deal with dynamic-scene analysis without any prior knowledge about the content in the frames. In this section the researcher had focused his work on the routines available in OpenCV libraries to explain the sparse optical flow methodology and dense optical flow methodology.

Dense optical flow method such as Horn-Schunck method associate the velocity to each pixel in the frame or in other methods the distance which the pixel has moved between the current and previous frame. Another dense optical method, that match windows around each pixel from one frame to the next frame, known as block matching is also implemented in OpenCV and soon will be described in more detail.

Due the high computational costs of dense optical flow methods sparse methods are an alternative however this methods are feature based methods that first selects some feature in the frames and then matches this features and computes the disparities between subsequent frames.

Next is presented the most used sparse tracking technique Lucas-Kanade also implemented with image pyramids for efficiently track fast motions. Finally are presented the Horn-Schunck and block-matching dense methods. The next figure Figure 16.1 is used as a simple sequence of two

frames where the main object moved slightly to the right.



Figure 16.1: Sequence of frames where the optical flow is to be computed.

## 16.2.2 Sparse Tracking Techniques

### 16.2.2.1 Lucas - Kanade (LK) Method – Sparse Optical Flow Method

LK method relies only on local information from small local windows surrounding each of the points of interest. To avoid that large motions points from being hidden outside the local windows and thus impossible to be tracked the LK method has a pyramidal implementation to allow large motions to be included by local windows. Tracks starts at lower detail towards the finer detail.

### 16.2.2.2 Lucas - Kanade Assumptions

Lucas - Kanade algorithm relies in three assumptions:

1. Brightness constancy – The brightness of a pixel does not change as it is tracked from frame to frame. Taking in account one dimension, the brightness consistency equation can be express as follows:

$$f(x, t) = I(x(t), t) = I(x(t + dt), t + dt); \quad (16.2.1)$$

Resulting in zero derivative:  $\frac{\partial f(x)}{\partial t} = 0$ ,

2. Spatial Coherence – Neighbours points in the the image frame belongs to the same surface , have similar motion and move for nearby points on the image plane of the next frame.
3. Small movements – The motion of a surface patch changes slowly in time - the object or the camera does not move substantially from frame to frame, This can be achieved with very small time increments between frames.

To understand the implications of the last assumption is first considered a single dimension and then generalize it for the two dimensions. Starting with brightness consistency equation (11.1.3)

and performing the next substitution  $f(x, t) \rightarrow I(x(t), t)$  and then applying the rules for partial differentiation, yields:

$$\frac{\partial I}{\partial x_t} \left( \frac{\partial x}{\partial t} \right) + \frac{\partial I}{\partial t_{x(t)}} \quad (16.2.2)$$

The first term is the spacial derivative  $I_x$ , the second term the velocity  $v$  and the last member the derivative over time  $I_t$ . Thus the optical flow velocity for one dimension is obtained as follows:

$$v = \frac{I_t}{I_x} \quad (16.2.3)$$

Adapting the one-dimensional solution to two dimensions and changing the nomenclature the velocity along  $xx$  axis is  $u$  and the velocity along  $y$  axis is  $v$ , then is possible to obtain:

$$I_x u + I_y v + I_t = 0 \quad (16.2.4)$$

This single equations has two unknowns for any given pixel making it impossible to obtain a unique solution for the two-dimensional motion at that pixel. Only the perpendicular motion to the line described by the flow equation – the normal flow, as shown in Figure 16.2, can be solved.

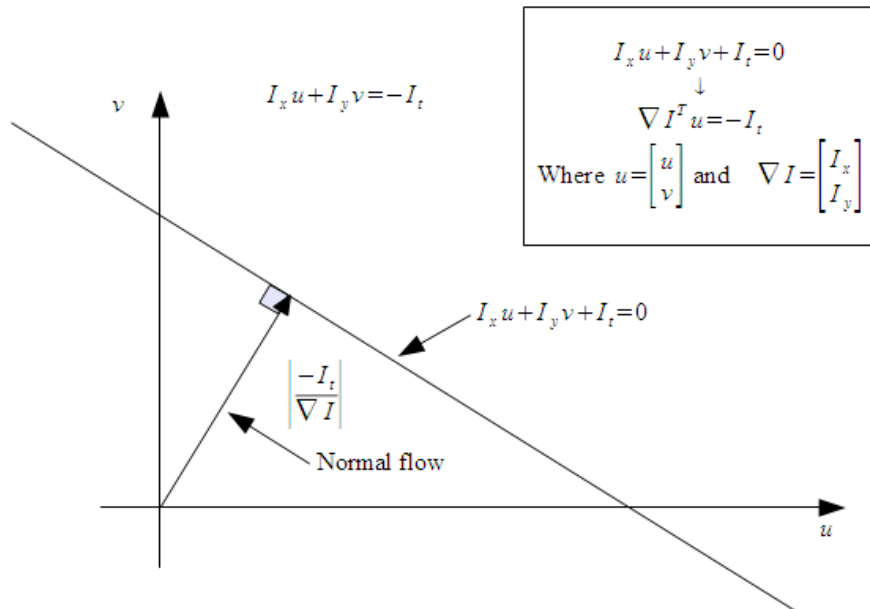


Figure 16.2: Normal flow.

Normal optical flow results from the aperture problem Figure 16.3 originated when motion is detected with small aperture or when there is only one edge instead of a corner. The edge alone is insufficient to determine exactly how the entire object is moving.

The second assumption is needed in order to solve the unconstrained equation. Under the assumption that pixels moves coherently the surrounding pixels can be used to build a system of equations. For example if a 3-by-3 window of brightness values is used around the current pixel to compute its motion is possible to obtain 9 equations in a single-channel image ( or 27 equations in a three channel image) as follows:

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_9) & I_y(p_9) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_9) \end{bmatrix}$$

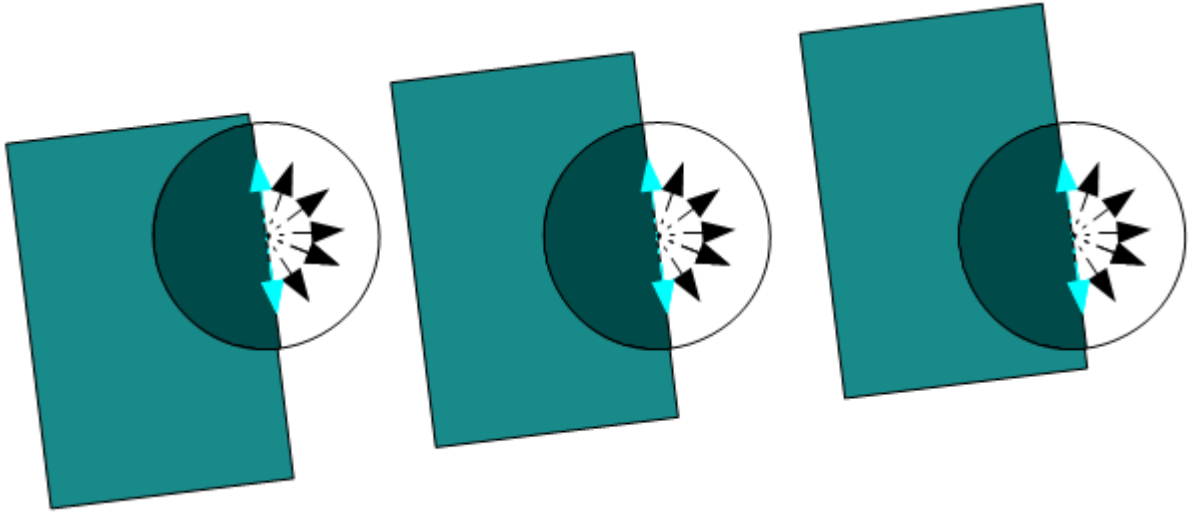


Figure 16.3: Aperture problem originated by a small aperture window.

The system of equations is now over constrained and can be solved. Using the least-squares minimizations, the equation,  $\min \|Ad - b\|^2$  is solved as follows:

$(A^T A)d = A^T b$  . From this relation is obtained the  $u$  and  $v$  motion velocity components:

$$\begin{bmatrix} u \\ v \end{bmatrix} = (A^T A)^{-1} A^T b \quad (16.2.5)$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}^{-1} \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

The system can be solved when  $(A^T A)$  is invertible and therefore when its rank is equal to 2, which occurs when the the image regions contains edges running in at least two directions.

$(A^T A)$  has the best properties (larger eigenvalues) when the tracking window is centred over a corner in the image, similarly to the previously mentioned Harris autocorrelation matrix.

The next figure Figure 16.4 shows the computed optical flow that occurs in the sequence of images illustrated in Figure 16.1. The routine that allow to implement this tracking technique is `cv::calcOpticalFlowLK()` routine.



Figure 16.4: Result from applying the pyramid Lucas - Kanade technique.

LK method requires large window to catch large motions, this requirement by itself may violate the spatial coherence assumption. Thus to minimize the motion assumptions violations the track happens first over large spatial scales using an image pyramid and then progress to the next layer down using the resulting motion estimates. This more efficient method is known as pyramid Lucas-Kanade method and has better performance than `cv::calcOpticalFlowLK()` function for most of the cases. The function that implements this method is `cv::calcOpticalFlowPyrLK()` and has the following parameters:

---

```
void cv::calcOpticalFlowPyrLK( const Mat& prevImg, const Mat& nextImg, const
    vector<Point2f>& prevPts, vector<Point2f>& nextPts, vector<uchar>& status,
    vector<float>& err, Size winSize=Size(15,15), int maxLevel=3, TermCriteria criteria
    = TermCriteria( TermCriteria::COUNT + TermCriteria::EPS, 30, 0.01 ), double
    derivLambda = 0.5, int flags = 0);
```

---

`prevImg` and `nextImg` – Are the initial and final images and both should be single-channel or 3-channel 8-bit images.

`prevPts` and `nextPts` – `prevPts` is the input vector that contains the points for which the motions is to be found, and `nextPts` is a similar vector into which the computed new locations of the points of `prevPts` are to be placed.

`status` – Output vector that contains the status of each feature. Each element has value 1 if `prevPts` element has been found or 0 otherwise.

`err` – Output vector that will contain the difference between patches around the `prevPts` and `nextPts`.

`winSize` - Defines the size of the window for computing the local coherence motion.

`maxLevel` – Defines the number of pyramid levels, if it is set to zero, then the pyramids are not used, if it set to 1 two levels are used, if set to 2, three levels are used and so on.

`criteria` – As seen before, it implements the termination criteria, it defines the number of iterations or constraint the movement of the search window by less than a number defined by `cv::TermCriteria::EPS`.

`derivLambda` – Defines the weight of image intensity and spatial derivatives used to estimate the optical flow. If `derivLambda = 0` only image intensity is taken in account else if `derivLambda =`

1 only spatial derivatives are used. For any value within [0; 1] interval the algorithm used both approaches with the corresponding proportions.

flags – Allows to set whether the algorithm uses the prevPts as initial estimation, by setting the flag to OPTFLOW\_USE\_INITIAL\_FLOW, or if the flag is set to zero then initially prevPts is set to nextPts.

### 16.2.3 Dense Tracking Techniques

The dense tracking techniques are less used due the high computational efforts they require and the need for very fast and efficient routines by systems that employ optical flow routines.

OpenCV contains two routines to implement the dense tracking techniques: Horn-Schunck method and Block-Matching method described next.

#### 16.2.3.1 Horn-Schunck Method

This method was one of the firsts to make use of the brightness constancy and implement the basic brightness constancy equations. The solution for those equations was a smoothness constraint on the velocities  $v_x$  and  $v_y$  that were derived by minimizing the Laplacian of the optical flow components as follows:

$$\frac{\partial}{\partial x} \frac{\partial v_x}{\partial x} - \frac{1}{\alpha} I_x (I_x v_x + I_y v_y + I_t) = 0 \quad (16.2.6)$$

$$\frac{\partial}{\partial y} \frac{\partial v_y}{\partial y} - \frac{1}{\alpha} I_y (I_x v_x + I_y v_y + I_t) = 0 \quad (16.2.7)$$

The coefficient  $\alpha$  is the weighting coefficient known as regularization constant. Larger values of  $\alpha$  lead to smoother vectors of motion flow and penalize regions in which the flow is changing in magnitude. The old C routine that implements this tracking method is `cvCalcOpticalFlowHS()` and has the following parameters:

---

```
void cvCalcOpticalFlowHS( const CvArr *imgA, const CvArr *imgB, int usePrevious, CvArr
    *velx, CvArr *vely, double lambda, CvTermCriteria criteria );
```

---

imgA and imgB – This first two parameters are pointer to the first and second images frames from a sequence of images. They must be 8-bit, single-channel images.

usePrevious – This parameter tells the routine to use the arrays velx and vely velocities computed from the previous frame as the initial starting point for the iterations process.

velx and vely – This parameters are pointers to 32-bit, floating-point, single-channel arrays that will be used to store the x and y velocities results.

lambda – This parameter is not the regularization constant seen previously but it has some relation. Lambda is the weight coefficient that arises when attempt to minimize both motion-brightness equation and the smoothness equations, it represents the relative weight to the errors in each equation as they are minimized.

criteria – Criteria to terminate the velocity calculations.

### 16.2.3.2 Block Matching Method

This method divides the image into small regions called blocks that are typically squares with a predefined size and are often overlapped in the image. The algorithm divide the previous and current images in such blocks and then compute their motion, thus the returned velocity are commonly of lower resolution than the input images due the fact the algorithm operates in the blocks level and not at the single pixel level.

The size of the resulting images is given by the following formulas:

$$W_{result} = \left\lfloor \frac{W_{prev} - W_{block} + W_{shiftsize}}{W_{shiftsize}} \right\rfloor \quad (16.2.8)$$

$$H_{result} = \left\lfloor \frac{H_{prev} - H_{block} + H_{shiftsize}}{H_{shiftsize}} \right\rfloor \quad (16.2.9)$$

The algorithm perform a search that starts from the original block position located in the previous frame and compares it with the new blocks for possible matches in the new frame. The comparison is a sum of absolute differences of all pixels within the block, in case of a good match the comparison for this block is stopped. This process is obtained with the old C routine `cvCalcOpticalFlowBM()` that has the following parameters:

---

```
alFlowBM( const CvArr *prev, const CvArr *curr, CvSize block_size, CvSize shift_size,
          CvSize max_range, int use_previous, CvArr *velx, CvArr *vely );
```

---

`prev` and `curr` – The first two parameters are pointers to the previous and current images that must be 8-bit, single-channel images.

`block_size` – Defines the size of the block to be used.

`shift_size` – This parameter defines the step size between blocks whether the blocks overlap, and if so, how much the block will overlap.

`max_range` – This parameter determines the size of the region around a given block where this block will search for matches in the subsequent frame.

`use_previous` – If set, it indicates that the values stored in the arrays `velx` and `vely` should be used as a starting points for the block searches.

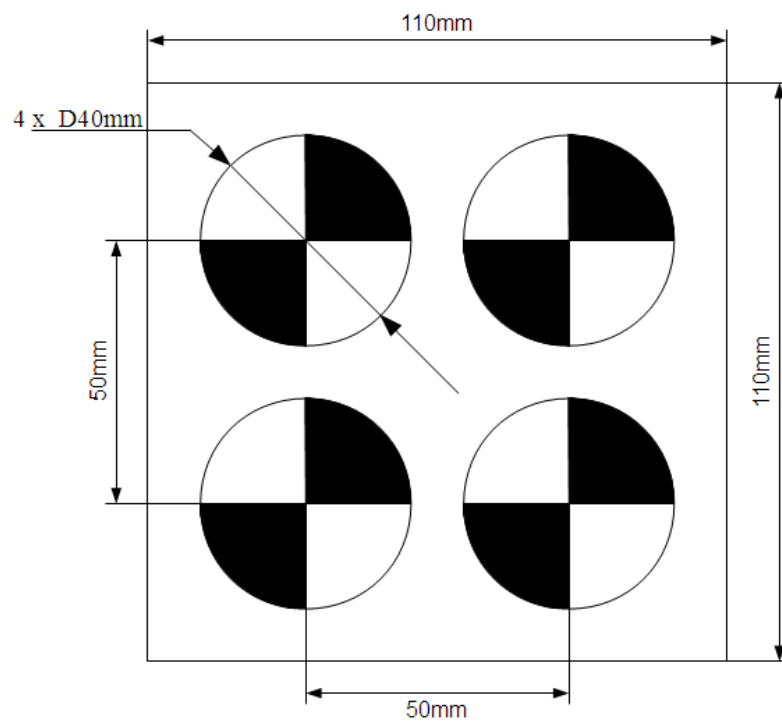
`velx` and `vely` – Are pointers to arrays of 32-bit, single-channel images that will store the computed motions for the individual blocks ( not for individual pixels once motion is computed block by block ).



## **17      Appendix G: Targets and MELFA Basic IV Program**

- Target 1, 2 and 3 dimensioning.
- Target 1,2 and 3 on real scale size.
- MELFA Basic IV 3D path program.

### Target 1 Dimensions



### Target 2 Dimensions

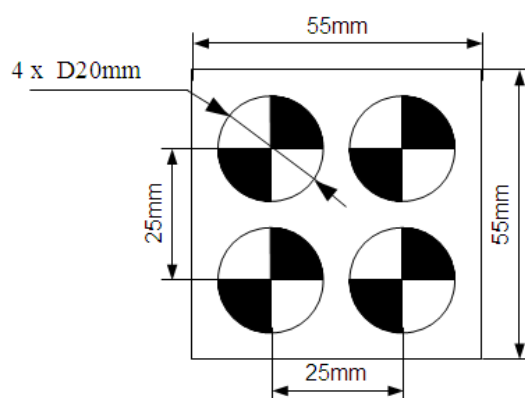


Figure 17.1: Dimensioning of the target1 and target 2.

## Target 3 Dimensions

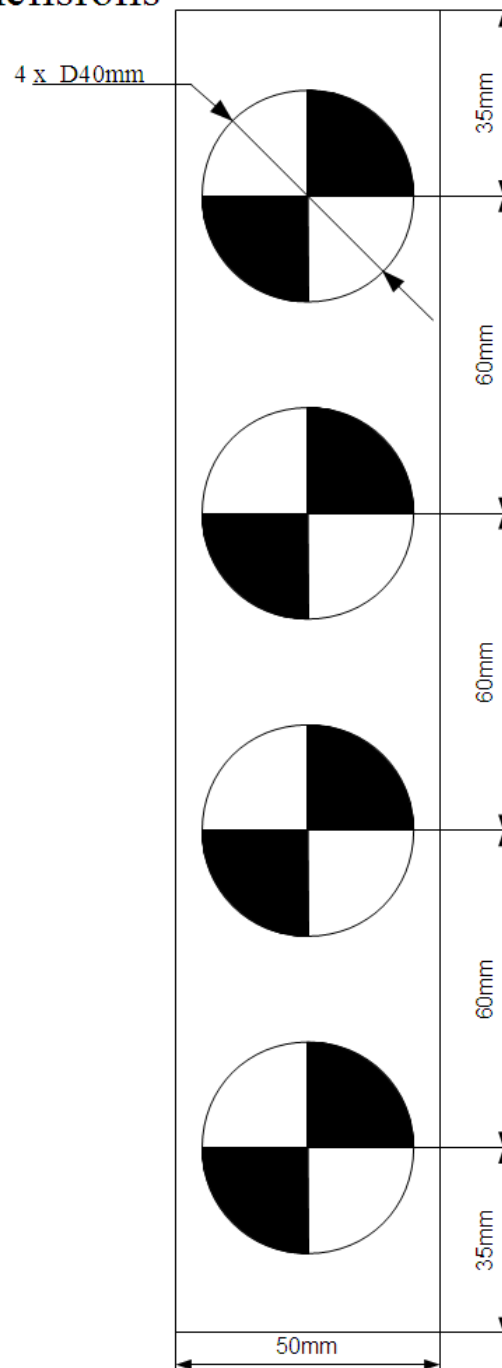


Figure 17.2: Dimensioning of the target 3.

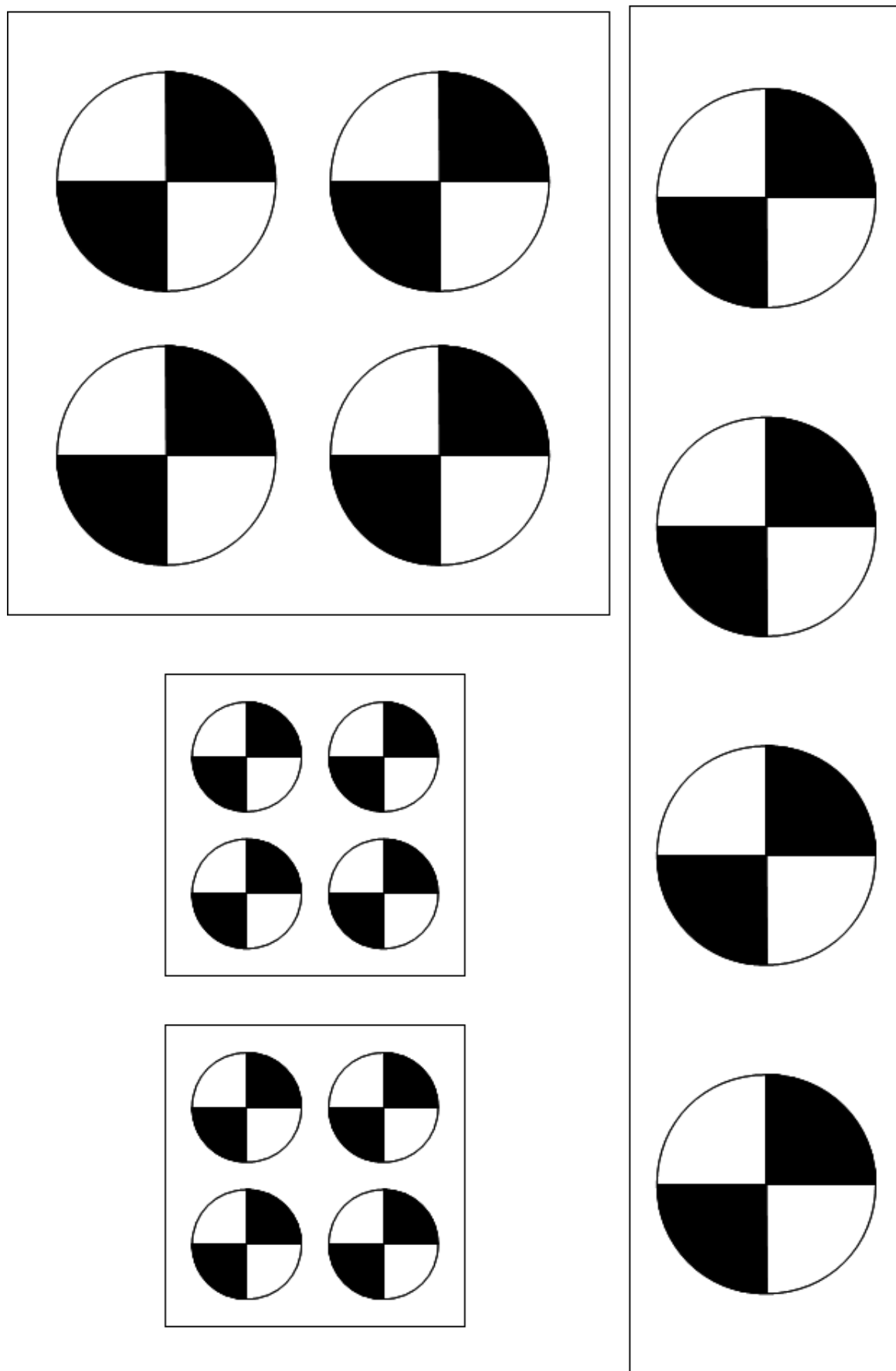


Figure 17.3: Targets 1, 2, and 3.

```
10 P1 = (351.84,187.36,165.93,164.9,180.51,0,0)
20 P2 = (366.74,119.42,217.99,159.89,180.51,0,0)
30 P3 = (378.44,87.62,109.8,149.89,180.51,0,0)
40 P4 = (348.1,80.6,151.7,149.89,180.51,0,0)
50 P5 = (270.86,144.24,160.06,149.9,180.51,0,0)
60 P6 = (301.6,196.13,130.03,149.9,180.51,0,0)
70 SERVO ON
80 OVRD 50
90 DLY 2
100 MVR P1, P2, P3
110 MVR P4, P5, P6
120 MVR P3, P5, P1
130 SERVO OFF
140 END
```

*Figure 17.4: MELFA Basic IV simple 3D path program implementation.*

## 18      **Appendix H: StereoVisionProg**

- Main menu's Option [0] – Compute optical flow.
- Main menu's Option [1] – Single video operations.
- Main menu's Option [2] – Stereo video operations.
- Main menu's Option [3] – Stereo calibration.
- Main menu's Option [4] – Compute 3D points.
- Main menu's Option [5] – Rotation matrix parametrization.
- Main menu's Option [6] – List current directory files.

## 18.1 Introduction

StereoVisionProg program is the name of the researcher-made program implemented with OpenCV v.2.1 libraries and Microsoft Visual Studio 2008 IDE software to provide the functionalities and methods needed along this research.

Due the program extension this section was necessary to provide a better description of each option and how to proceed in each case. It provides additional detail by using activity diagrams and snapshots of each menu and sub-menu to describe how the code was implemented. This appendix stands together with Methods chapter three (see Chapter Three). All the activity diagrams were created using a free *unified modeling language* (UML) tool from *Visual Paradigm for UML*.

For a better understanding, flexibility and code reuse, the main program was implemented with different classes. Each one of the classes has a main role in the process of loading images, performing cameras calibration and calibration optimization, stereo relations estimation, stereo rectification, and stereo correspondence. The main program was implemented to receive user inputs from a command line, all the results obtained are saved into individual XML files with proper nomenclature depending on which process was performed. The main program's menu is shown in the next figure (see Figure 18.1).

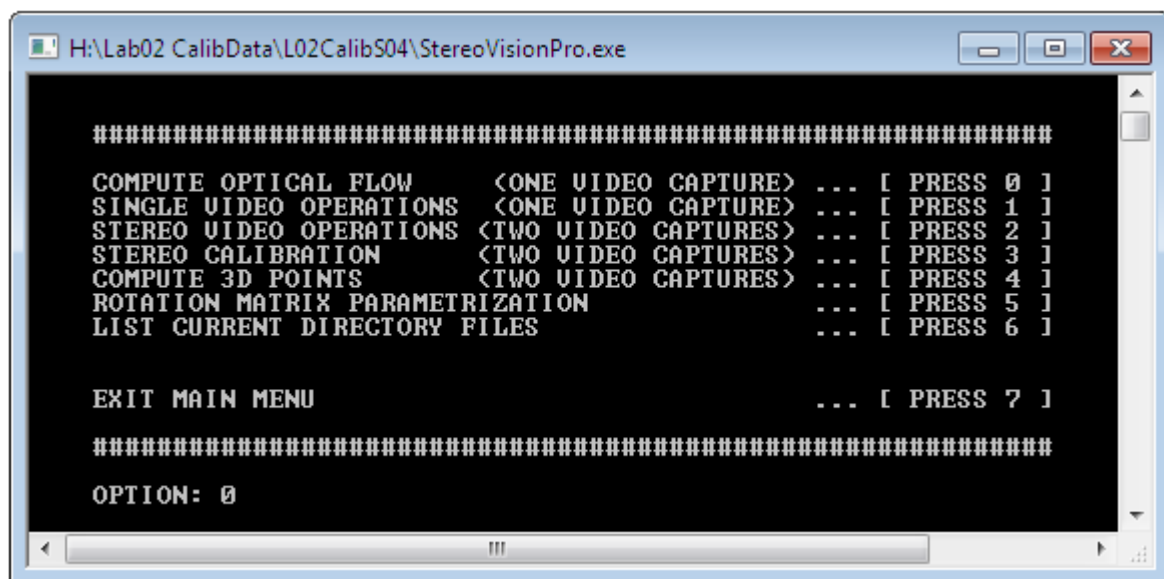


Figure 18.1: StereoVisionProg: Main menu.

## 18.2 Main menu's Option [0] – Compute Optical Flow

The first option available from StereoVisionProg's main menu was implemented to provide different functionalities for sparse and dense optical flow methods. By selecting this option a sub menu is displayed as shown in the next figure (see Figure 18.2). This sub menu allows to select the capture mode from camera or an AVI file.

Case **option [1]** is selected the user is asked to input the camera **INDEX**, this index is the ID number that identifies each cameras connected to the computer. On the other hand if **option [2]** is selected the program lists all AVI files in the directory from where the program was started and asks the user to select one option than corresponds to one of the file listed, as shown in the next figure ( Figure 18.3).

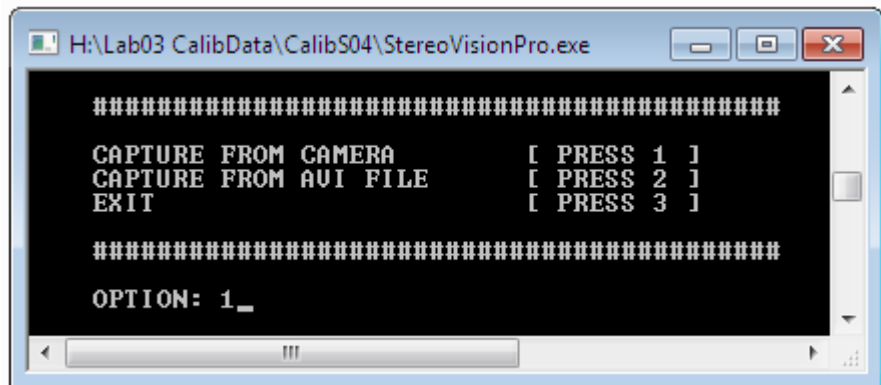


Figure 18.2: Main menu's option 0 sub menu.

After having selected the camera index/video file the program starts capturing the video sequences frames and displaying them in the **CAMERA CAPTURE** window. A second window named **TRACKING SETTINGS** is then displayed for control purposes as shown in the next figure (see Figure 18.4). This widow was created to allow fine adjustments for each parameter individually and also to allow the selection between sparse and dense optical flow method.

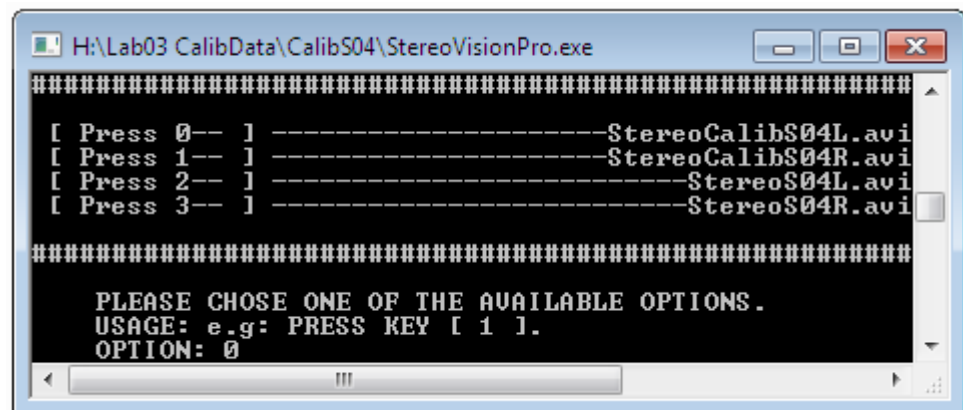


Figure 18.3: Current directory's list of avi files.

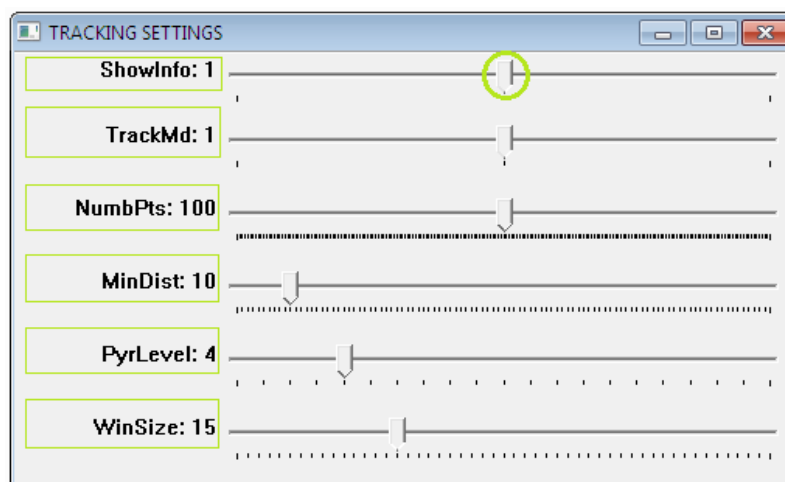


Figure 18.4: TRACKING SETTINGS window's trackbars.



Each trackbar functionality is described next:

- **InfoShow:** Displays settings information (1) or when its value is set to (2) it displays the information related with the sparse set of points currently being tracked with Lucas-Kanade Pyramid tracker code.
- **TrackMd:** Sets the optical flow method. If value (1) is selected - uses sparse optical flow method (**Lucas - kanade Pyramid**) to track set of points added by the user by left button mouse clicking over the capture window; If value (2) is selected - uses dense optical flow method to compute the velocity of each image's pixel, this approach used the Horn - Schunck optical flow algorithm; If value (0) is selected – no optical flow method is performed over the current video capture.
- **NumbPts:** Set the maximum number of points to be selected for sparse optical flow purposes.
- **MinDist:** Set the minimum Euclidean distance allowed between points (points selected by the user by left button mouse clicking event) in pixel units.
- **PyrLevel:** Sets the pyramid level to be used by Lucas-Kanade Pyramid tracker code.
- **WinSize:** Set the search window size to be used during the sparse optical flow tracking process.

For both cases (sparse and dense optical flow) was implemented a function that if the user presses “S” key the list with the current tracked points are saved into an XML file named **TrackedPoints.xml** if **TrackMd** value (1) is set or it saves the current dense image of pixel's velocities to an **DenseOFVelocity[xxx].bmp** file if **TrackMd** value is set to (2) (dense optical flow). Pressing “Esc” key closes the video capture and do not perform any saving operation. For **TrackMd (1)** mode if the right mouse click event is detected all current points being tracked are erased without the need of restarting the video capture.

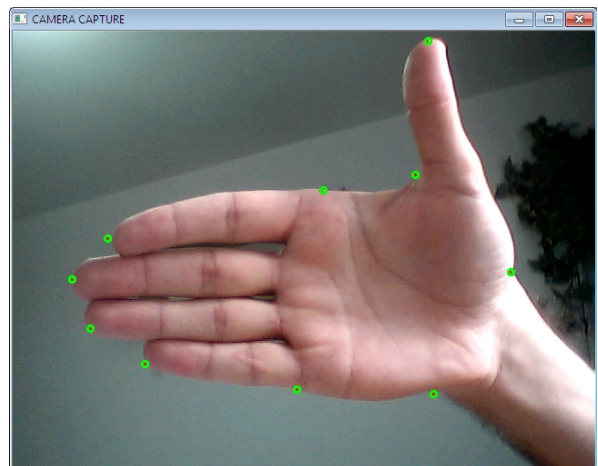


Figure 18.5: Sparse optical flow tracking example.

The right image (see Figure 18.5) shows an example of 10 points being tracked using the sparse optical flow methodology.

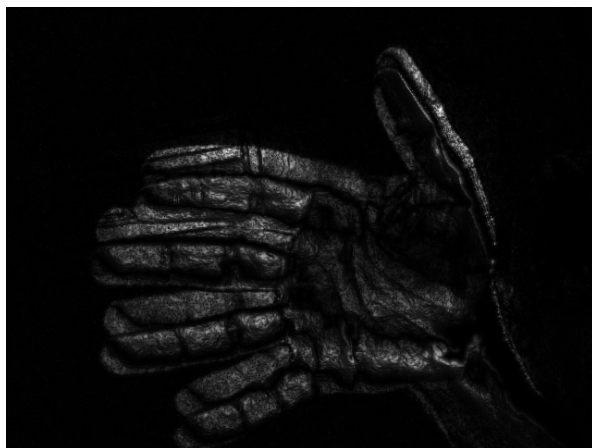


Figure 18.6: Dense optical flow using Horn - Schunck method.

In the left figure (see Figure 18.6) is shown the velocities image from a moving hand obtained using the dense Horn – Schunck optical flow method. The velocity for each image pixel was obtained by summing the vertical and horizontal velocity component of the optical flow maps. All the image pixel's velocity were normalize by the maximum pixel's velocity value.

### 18.3 Main menu's Option [1] – Single video operations

The second StereoVisionProg's option implemented for this case study is related to single camera operation such as read and write video sequences captured from a single camera connected to the computer or from a single video file imputed in AVI format. It allows to perform different operations and save the results into different files depending on the type of operation being performed. The next figure (see Figure 18.7) presents all the option available after being selected the main menu's option [1].

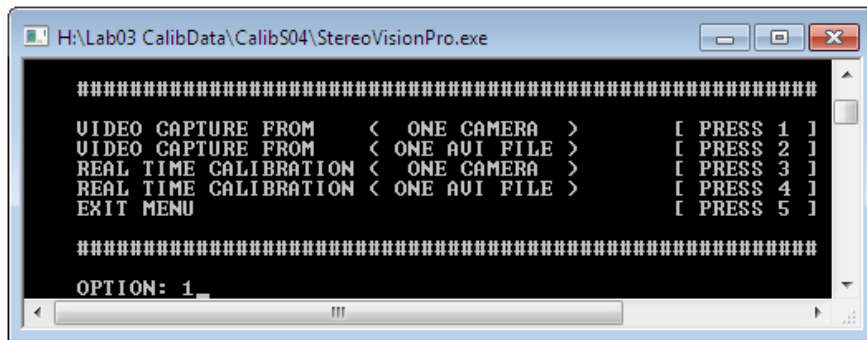


Figure 18.7: Main menu's option 1 sub menu.

Option [1] and [2] allows to capture video sequences from a camera connected to a computer or from an AVI video file inside the current directory the input for both cases were already described in the previous section. After choosing the video input successfully a control window named **CAPTURE CONTROLS** shown in the right figure (see Figure 18.8) is displayed. This window has as main purpose to provide controls to change the image capture resolution, colour space, and image format to be used if any saving operation is called. The description of each trackbar is as follows:

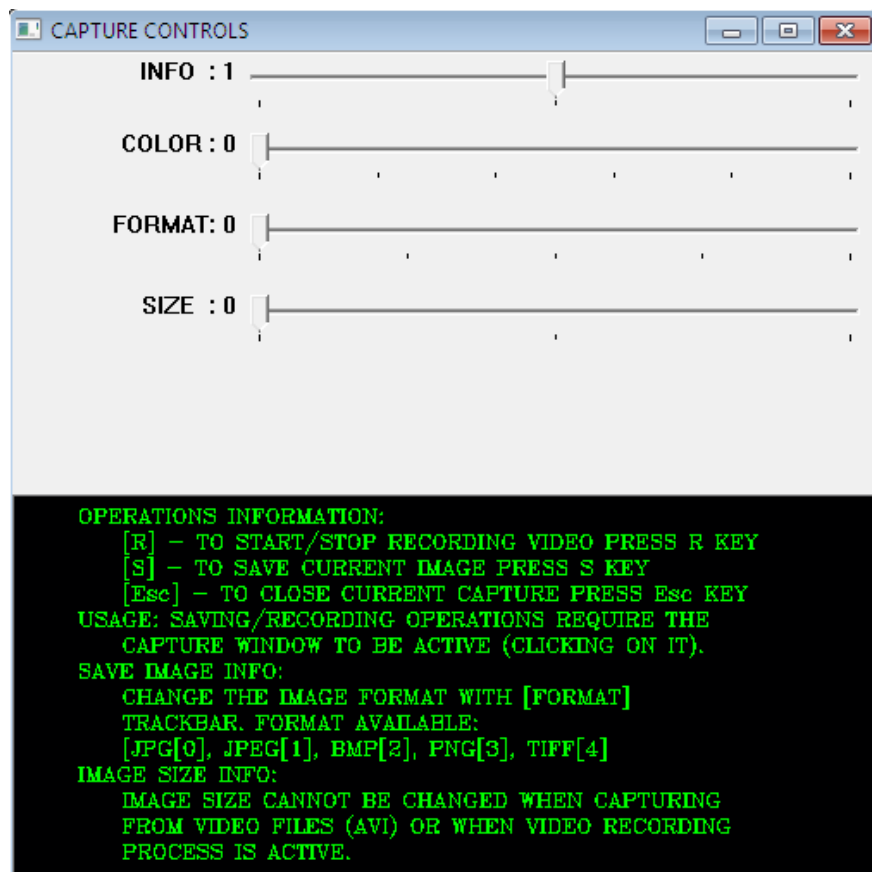


Figure 18.8: Capture controls' window.

- **INFO:** if its value equals (0) do not display any kind of information; If value equals (1) – display operations information as shown in Figure 18.8, and if value equals (2) – displays the current frame capture information such as image resolution, image depth, number of channels, current image format and colour space.

- **COLOR:** This trackbar is used to set the colour space to which the current video capture is being transformed. The values available are [(0), (1), (2), (3), (4), (5)] corresponding to the next color space [ RGB GRAY HSV HSL Lab Luv] respectively.
- **FORMAT:** This trackbar allows to set the image with which the current frame capture will be saved if any saving operation is performed. The values available are: [(0), (1), (2), (3), (4)] that correspond to [JPG JPEG BMP PNG TIFF] image formats, respectively.
- **SIZE:** This last trackbar allows to change the video capture image resolution if the video capture is not from an AVI video file or no video recording operation is on going.

During the video capture is possible to record video files. Pressing the “R” key will start or stop the video recording process. In the case of being the first video recording operation the program asks to introduce the **VIDEO NAME** and then prompts a window as shown in the the right figure (see Figure 18.9) to select the video compression type. To finish the video recording the user needs to press the “R” key again. Case the video capture closes unexpectedly before the user concluded the recording process, the program finishes the video writing process automatically.

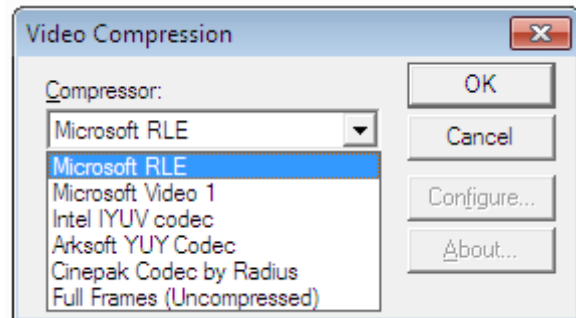


Figure 18.9: OpenCV video compression selection.

To save the current frame capture the “S” key need to be pressed, the image will be saved using the following default name **Image[XXX].[format]** where **XXX** is a sequence number 000-999 and **format** is the current format defined by **FORMAT** trackbar value.

If “Esc” key is pressed the current video capture operation is closed and the sub menu is display again. Selecting option (3) or (4) from the sub menu (already shown in Figure 18.7) it is possible to perform single camera calibration from a real time video capture or from an AVI file containing video sequences with a chessboard pattern being moved (to obtain rich sets of different chessboard position and orientations). Independently of the input type (camera or AVI file) the program asks the user to input the number of corners along XX direction (**nx**) and along YY direction (**ny**) and the square size (**squareSize**) in user defined units<sup>7</sup>.

If all corners were found over the current frame capture and the user presses “S” key the current chessboard corners coordinates are stored into a vector of 2D points, however its corresponding frame is not saved. After the user have added the required number of views, by pressing “Esc” the program will close the video capture and proceed with calibration procedure.

Having completed the calibration process all the resulting parameters are stored into an XML file named **SingleCameraCalibration.xml** by default. The **CALIBRATION INFORMATION** window displayed during the capture process contains information such as: all corners found (true/false), number of views currently added, chessboard properties, and operation's information updated during the video sequence capture process.

<sup>7</sup> The units are defined by the user (m, dm, cm, mm, inches, etc), all the next operation's results depending on the calibration parameters are computed using those units defined on the calibration process.

### 18.4 Main menu's Option [2] – Stereo video operations

This option allows to perform different operations using two video captures obtained from a stereo configuration. The next figure (see Figure 18.10) shows the sub-menu presented after selecting Main menu's Option [2].

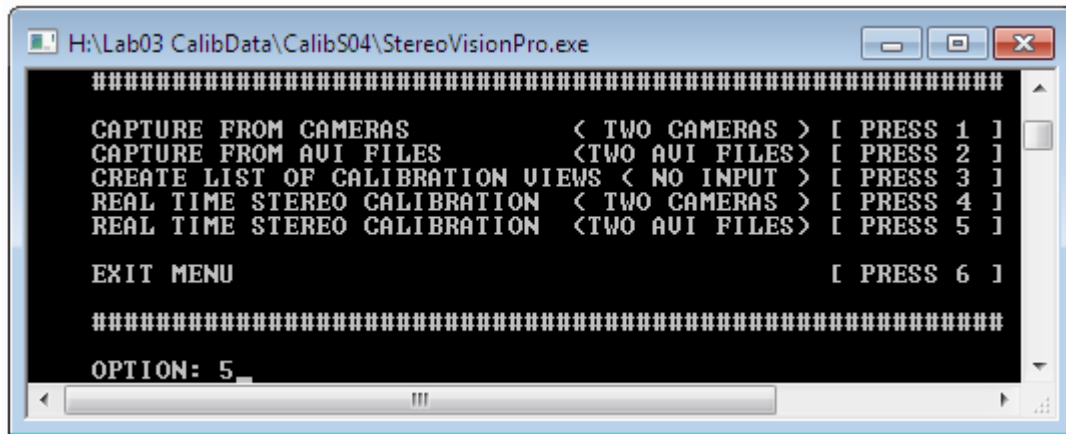


Figure 18.10: Main menu's option 2 sub menu.

Five options are available after selecting main menu's option 2. The first two options **Option [1]** and **Option [2]** are related with video sequences capture from two cameras or AVI files. This two options are similar to the video capture options presented in the previous section, the first requires that two camera indexes or in the second case two AVI files to be introduced. After the program starts capturing both video frames and displaying them in one single window (see Figure 18.1) named **CAPTURING FROM CAMERAS** or **CAPTURING FROM AVI FILES** it presents a window named **CONTROLS WINDOW** with exactly the same trackbars as the ones described in the previous section (see Main menu's Option [1] – Single video operations).

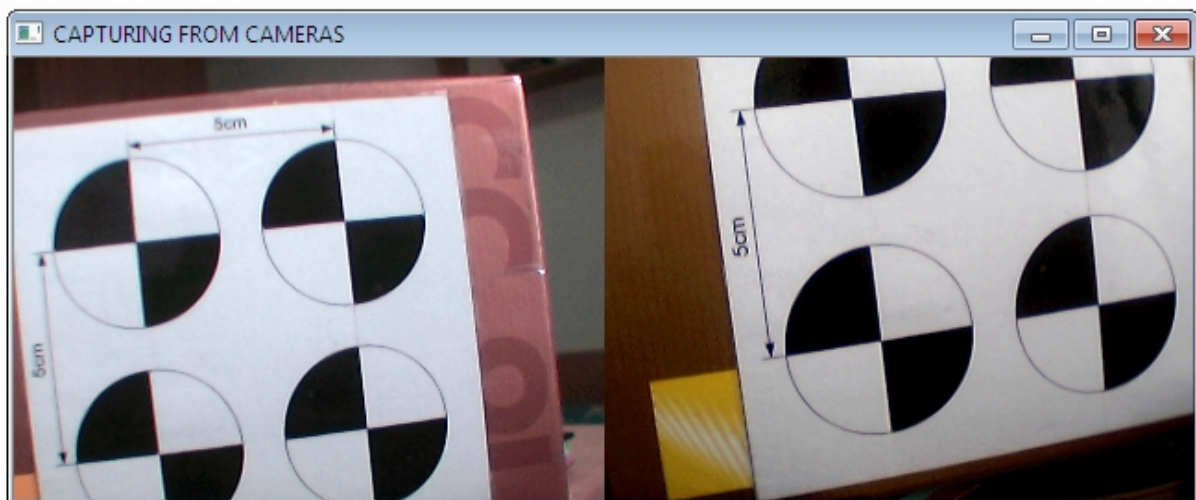


Figure 18.11: Capturing from two A4Tech Web Cams

- Pressing “S” key while capturing video the user is asked to introduce the text **FILE NAME** where the list of left and right image's names are to be saved and then it asks to input the image **FILE NAME PREFIX** to be used to create the sequential left and right image's names, i.e. if the image prefix is [ImgTest] by pressing “S” key the program generates **ImgTestL[XXX].[format]** and **ImgTestR[XXX].[format]** where **XXX** is a sequence number

between 000-999. A second text file is created [FILE NAME]Sync.txt containing the time elapsed between the left and right frame capture for each time “S” key is pressed, this time values can be used to evaluate the synchronisation between cameras captures. Despite the program stores the current images when “S” key is pressed it only save the images when the video capture finishes or the user ends it. This way the program avoids delays on video capture and extra processing to perform saving operations while capturing video frames.

- Pressing “Esc” key the program closes the current capture and, in case any pair of images were stored (by pressing “S” key), it generates both text files and saves the set of images stored in memory to the current directory.
- Pressing “R” key the program asks the user to input a video **FILE NAME PREFIX** to be used to create both left and right video file name, for example if the file name prefix introduced is **VideoTest01** the resulting video output files will be: **VideoTest02\_CAM1N000.avi** and **VideoTest02\_CAM2N000.avi**.

**Option [3]** was implemented to provide a functionality that allows to build a list of left and right views for calibration purposes. This option is useful when the user wants to use always exactly the same set of images for a number of operations such as stereo cameras calibration or retrieve a specific number of images from a big group of images. After selecting this option the programs asks the user to input the text **FILE NAME** where the image names are to be listed, **IMAGE PREFIX** for the left and for right images, the starting and the ending image sequence **NUMBER** and the step to be used to retrieved only few images within all range.

For example to create a list with a set of 100 left and right images from two sequences of images containing 8000 images named **StereoCalibS01L0000.bmp ... StereoCalibS01L8000.bmp** and **StereoCalibS01R0000.bmp ... StereoCalibS01R8000.bmp** the inputs would be as follows:

- **Text file name:** ListOfViews.txt (arbitrary name).
- **Left image prefix:** StereoCalibS01L.
- **Right image prefix:** StereoCalibS01R.
- **Starting number:** 0.
- **Ending number:** 8000.
- **Step number:** 80 (8000/100).

The next figure (see Figure 18.12) presents Option [3] activity diagram, this diagram is similar for the process implemented under the first two options from this sub menu, the only difference resides on the fact that when saving images with Option[1] and Option [2] the program stores the image's names into vectors for each time “S” key is pressed and the listing process only occurs when the video capture ends or the user ended the video capture voluntarily.



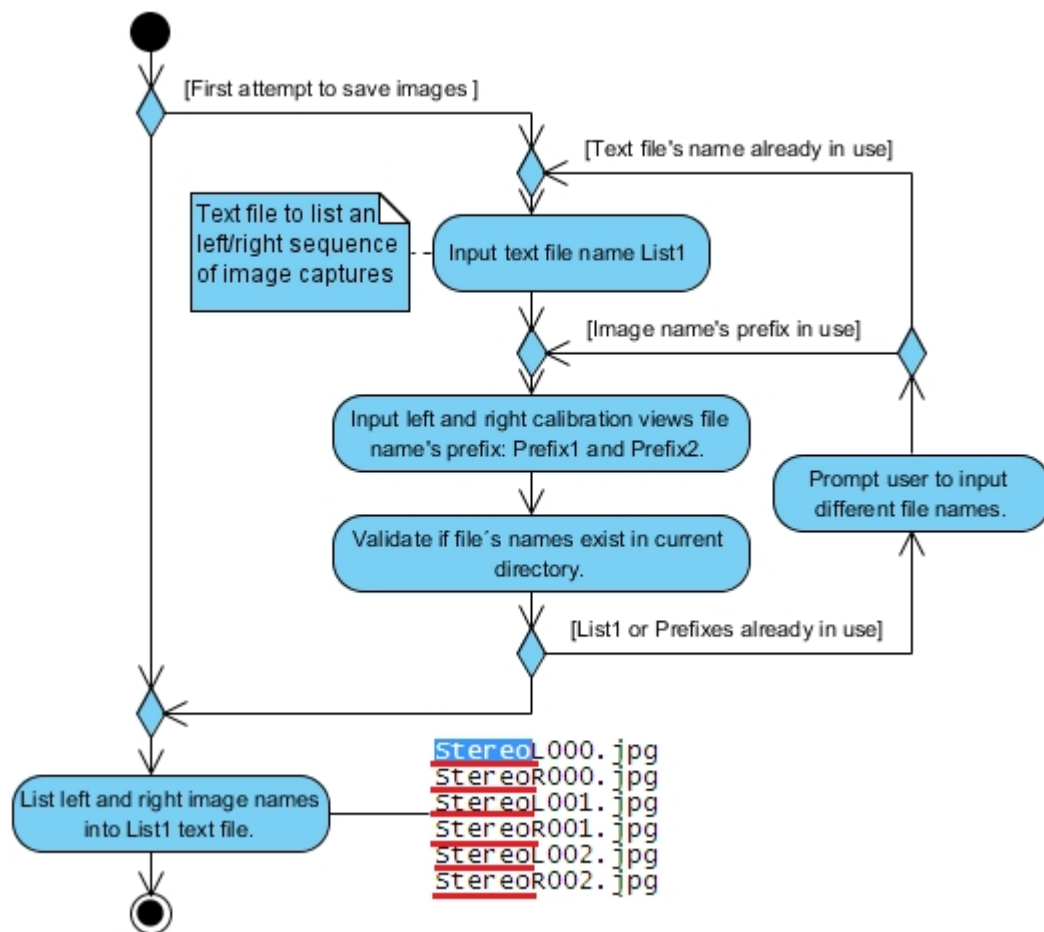


Figure 18.12: List image's names into a text file.

**Option [4]** and **Option [5]** were implemented to allow stereo cameras calibration by using real-time video capture or AVI files containing video for calibration purposes. By selecting one of this option the program asks the user to input the number of corners along XX direction (**nx**) and along YY direction (**ny**), and the square size (**squareSize**) in user defined units.

After starting capturing video from both inputs, the trackbars: **INFO**, **COLOR**, **FORMAT**, and **METHOD** from **CONTROLS WINDOW** allows to change the video properties and the calibration method in the last case. If trackbar **METHOD**'s value equals (1) – the program performs stereo calibration by first computing the intrinsic and extrinsic parameters and just then estimates the stereo relations using those parameters, if **METHOD**'s value equals (2) – the program performs stereo calibration by computing the intrinsic and extrinsic camera parameters and the stereo relations all at the same time. As normally, after completing the stereo calibration process, the program proceeds with the stereo rectification process using the last computed stereo calibration parameters. In the next figure (see Figure 18.13) is shown the activity diagram of the process implemented for Option [4] and Option[5]. The stereo calibration is described in deep detail in the next section ( see section Main menu's Option [3] – Stereo calibration).

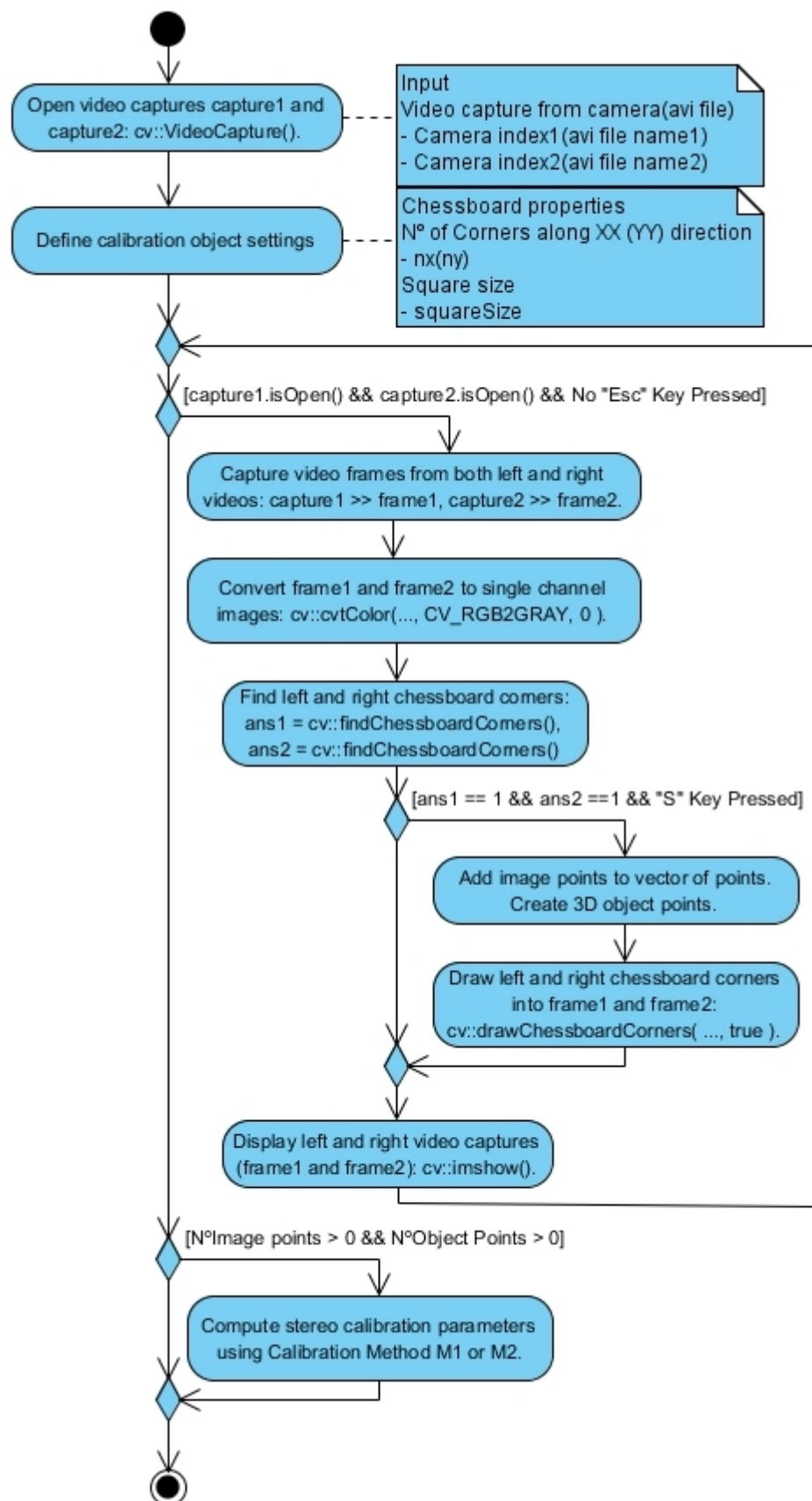


Figure 18.13: Real-time stereo calibration.

### 18.5 Main menu's Option [3] – Stereo calibration

By selecting Main menu's Option [3] another sub menu with four options are presented. **Option [2]** and **Option[3]** allow to perform stereo camera's calibration and stereo configuration's relations estimation. Additionally two other important functionalities were implemented with **Option [1]** and **Option[4]**, in the first case it makes possible to study how the calibration parameters evolve and in the second case it allows to optimize the stereo calibration parameters. The next figure (see Figure 18.14) shows the options available after selecting **Main menu's Option[3]**.

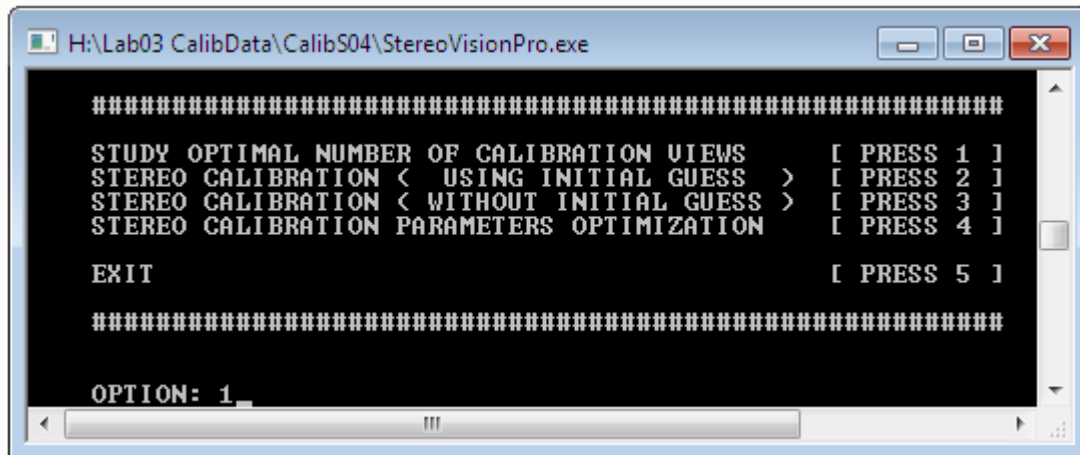


Figure 18.14: Main menu's option 3 sub menu.

**Option[1]** allows to study how calibration parameters evolve using sets with different number of calibration views. This option was used to define the optimal number of calibration views used during this research. The implementation uses the last calibration output file from where the image points for calibration are retrieved to build calibration views sets with 02, 05, 10, 20, 30, ... , 150 (or other range defined by the user) images. In the next figure (see Figure 18.15) is shown the activity diagram that better describes **Option[1]** implementation.

**Option[2]** allows to compute the stereo cameras calibration and stereo relations using calibration method **M1**. This method reads a text file with a list of calibration views and finds the chessboard corners for each left and right view, after completing the scan in all views it computes the calibration parameters for the left and right camera individually and just then it uses those obtained parameters to estimate the stereo configuration relation  $R$ ,  $T$ ,  $E$ ,  $F$ . This method is less time consuming and less computationally demanding than the calibration method **M2** implemented on **Option[3]**.

**Option[3]** allows to compute the stereo cameras calibration parameters and stereo configuration relations using calibration method **M2**. Similarly to method **M1**, this method reads a list with left and right calibration views and searches for the chessboard corners over each view. After building the image and object points vectors it computes the calibration parameters for both cameras and the stereo configuration relations all at the same time. This method revealed to be less robust and computationally more demanding comparatively with method **M1**.



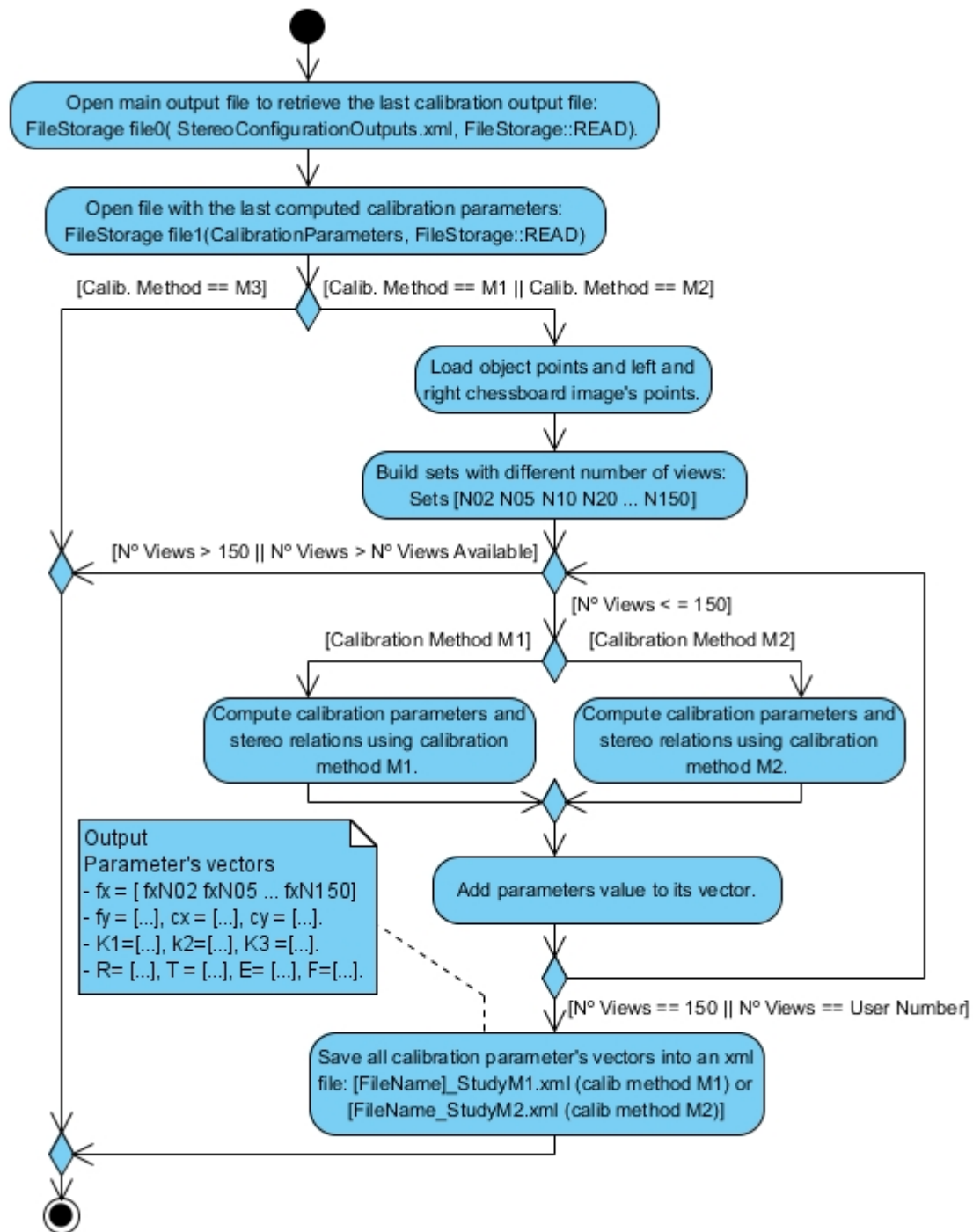


Figure 18.15: Study optimal number of calibration views.

For the optimal number of calibration views study both calibration methods M1 and M2 were used to allow to take conclusions about the robustness of each method and how the parameters evolve on each case. The next figure (see Figure 18.16) shows the activity diagram that better describes the two implemented calibration processes.

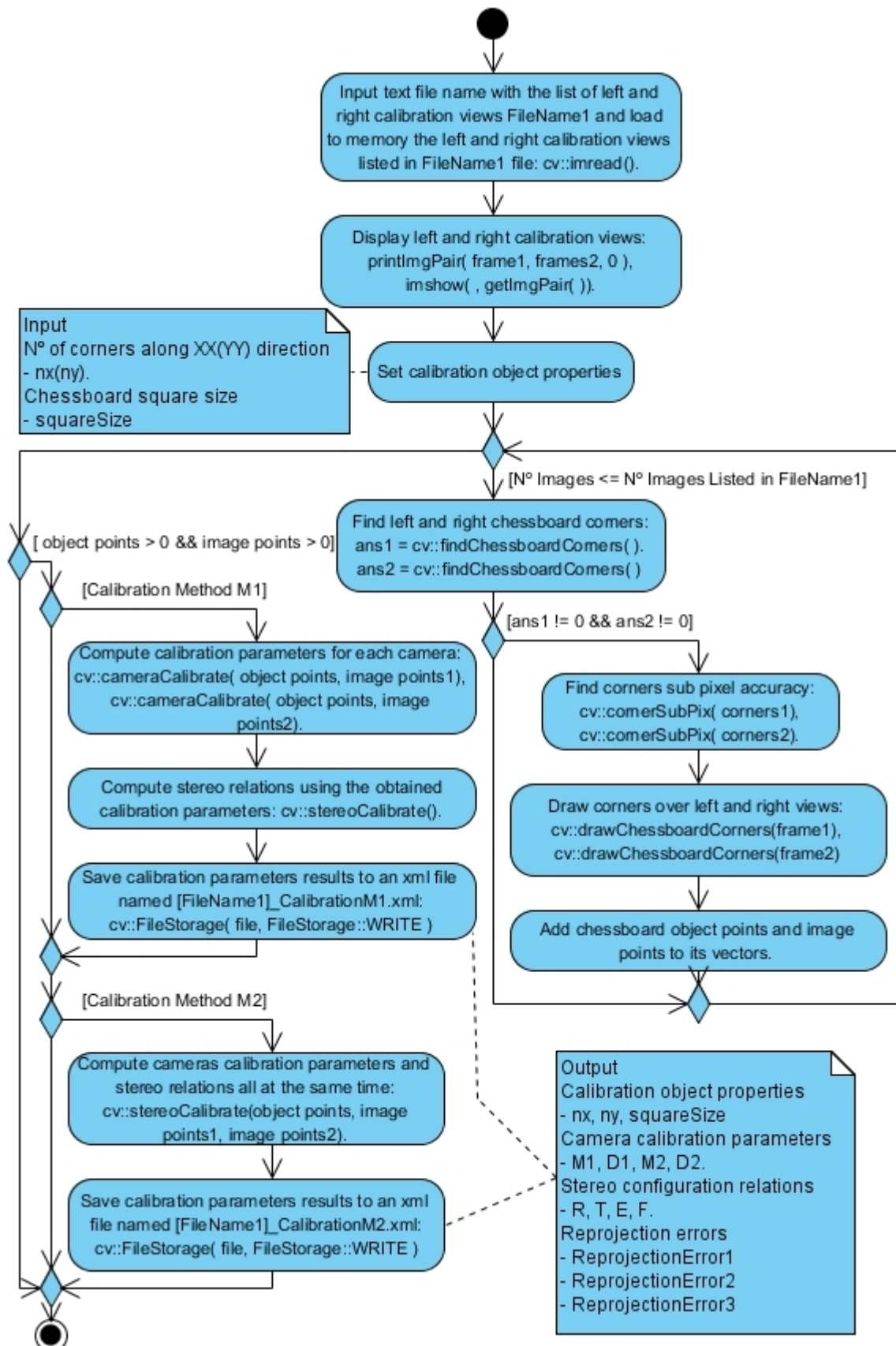


Figure 18.16: Calibration process method M1 and M2.

**Option[4]** was the last functionality to be implemented in order to improve the output results obtained from calibration process. This option allows to optimize the calibration parameters by excluding the views with higher errors contributions, the program starts by reading the calibration parameters and the object and image points resulting from the last calibration process, it asks the user to input the number of views to be filtered, and then projects the object points and computes the mean Euclidean distance between the reprojected and projected image points for each view allowing to determine which views have higher errors (higher Euclidean distances). The activity diagram that explain this procedure graphically is presented in the next figure (see Figure 18.17).

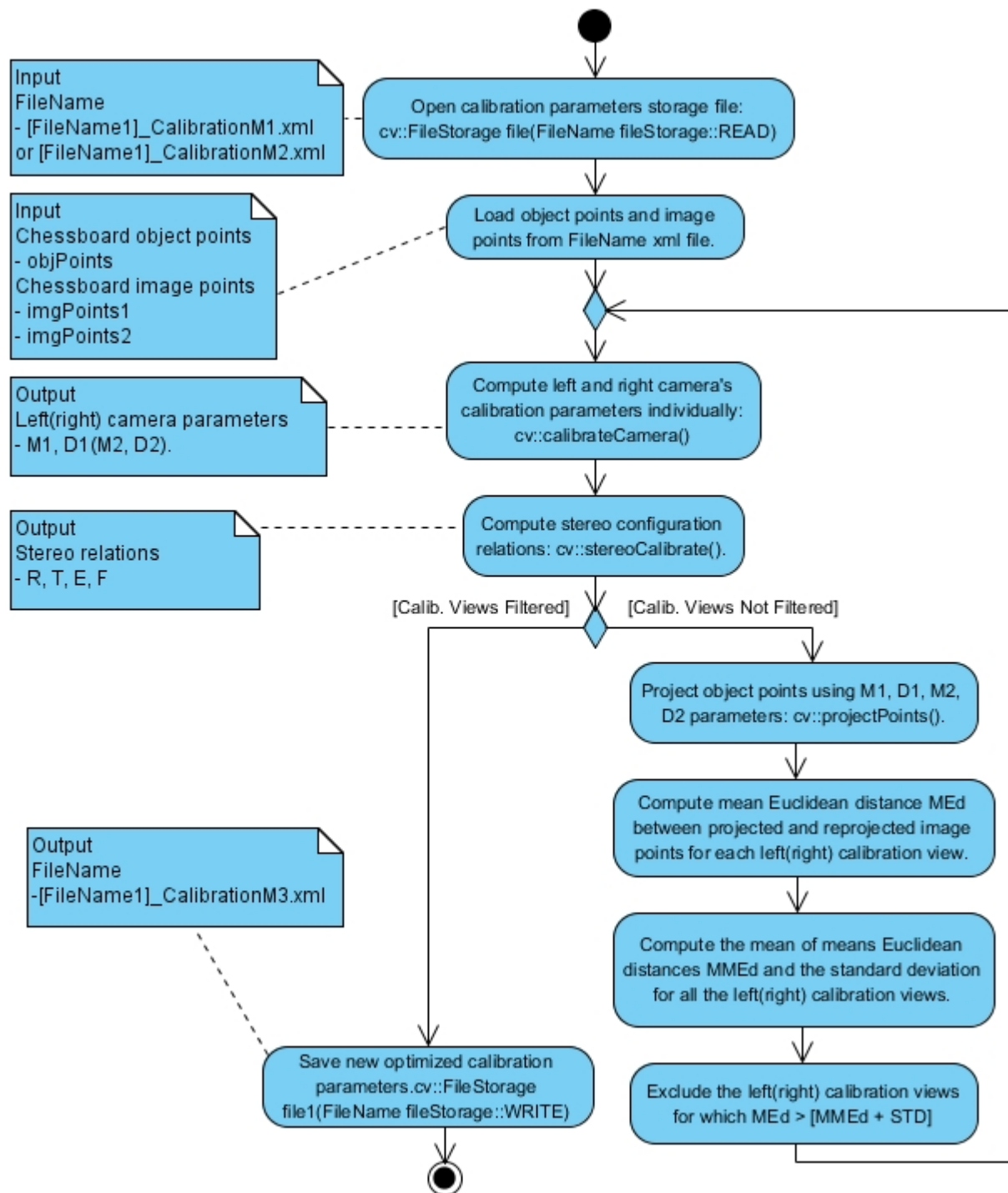


Figure 18.17: Calibration parameters optimization.

After performing any of the calibration processes available from Option[2], Option[3], and Option[4] the user is asked to select one of the following stereo rectification method: (1) **Calibrated Stereo Rectification** and (2) **Uncalibrated Stereo Rectification**. The main goal of performing the rectification is to compute the (undistortion+rectification) maps and the disparity-to-depth matrix. The outputs resulting from the former method are saved into an output XML file named **[FileName1]\_CalibRectification.xml** while the later method generates an output XML file named **[FileName1]\_UncalibRectification.xml**. In the next two figures (see Figure 18.18 and Figure 18.19) is shown the activity diagrams that explains in more detail the implementation for both calibrated and uncalibrated rectification processes.

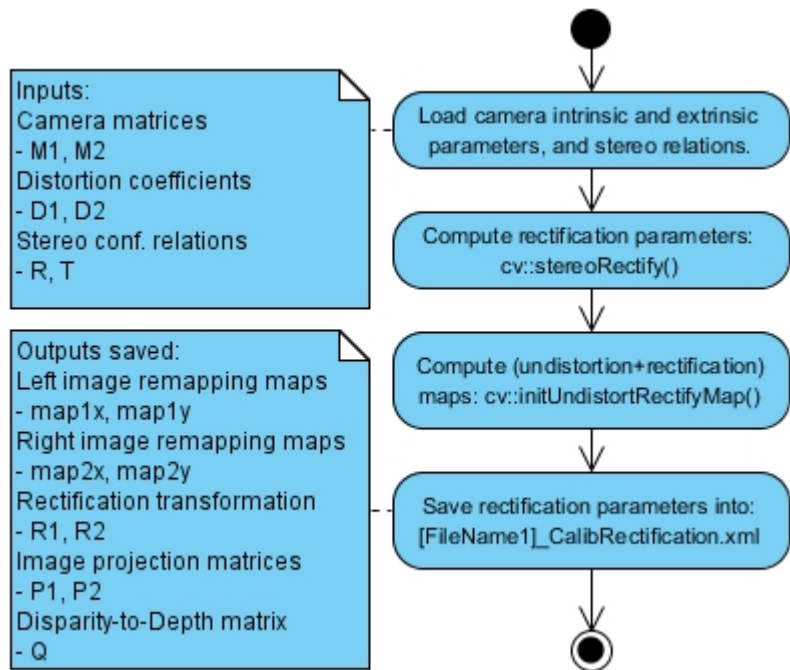


Figure 18.18: Calibrated stereo rectification.

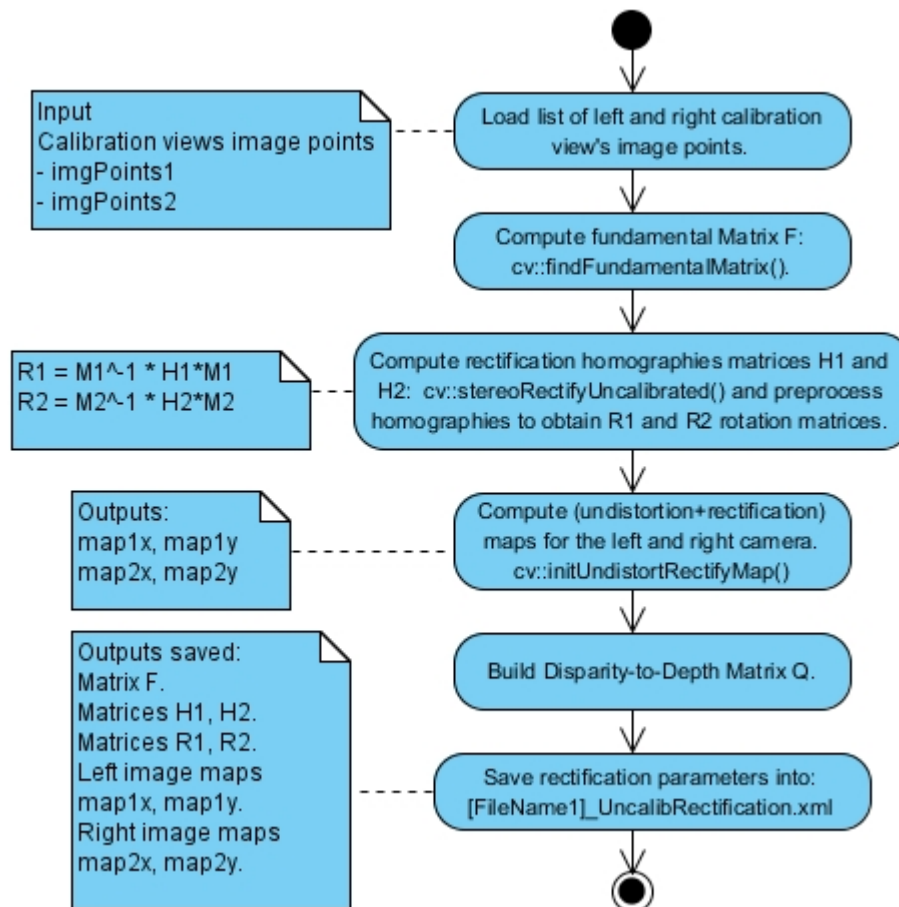


Figure 18.19: Uncalibrated stereo rectification.

### 18.6 Main menu's Option [4] – Compute 3D points

This Main menu option presents the last step needed to recover 3D information from stereo video captures. Two approaches were implemented to deal with dense and sparse stereo matching, the former computes the disparity map using all the left and right image while the second only performs stereo matching for a sparse set of points.

By selecting **Main menu's Option [4]** the program prompts a sub menu (see Figure 18.20) to select the input mode. In the case of option [1] is selected the user is asked to input left camera index and right camera index to capture from two specific cameras connected to the computer, however, instead of option [1] option [2] was selected the user is asked to select the video AVI file names.

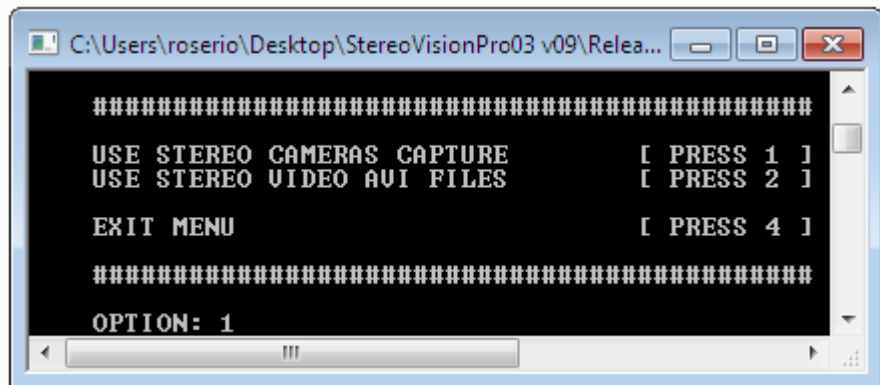


Figure 18.20: Main menu's option 4 video input.

On both cases the user needs to make sure that the stereo configuration used to obtain the video captures is the same as the one that was lastly calibrated and rectified, in other words, the program always loads the remapping parameters from the main xml output **StereoConfigurationOutputs.xml** file's node named **<RemappingMaps>**.

After selecting the stereo video capture mode the program presents another sub menu with two options, as shown in the next figure (see Figure 18.21).

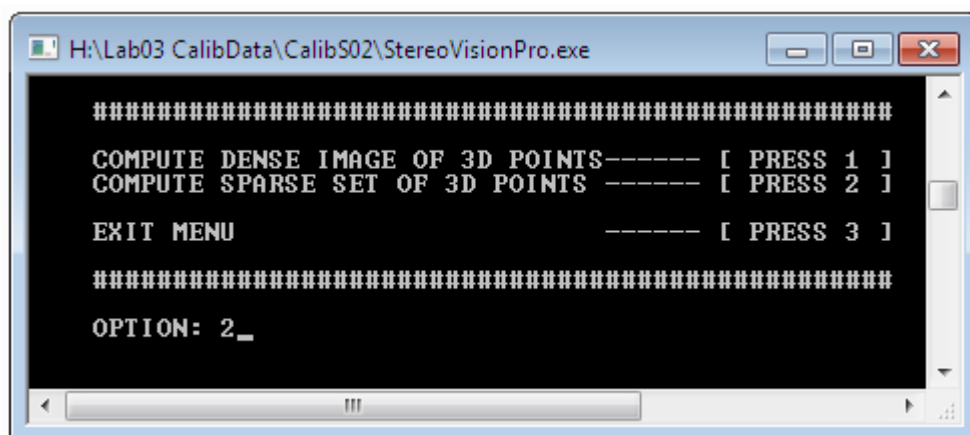


Figure 18.21: Stereo matching modes.

**Option [1]** allows to compute dense disparity image by using the stereo matching algorithms available from OpenCV libraries. Depending on the algorithm an additional control window named **MATCHING CONTROLS** allows to change the block match state settings by using the trackbars on this window. The next figure (see Figure 18.22) shows the activity diagram that better describes the dense stereo matching approach implementation.



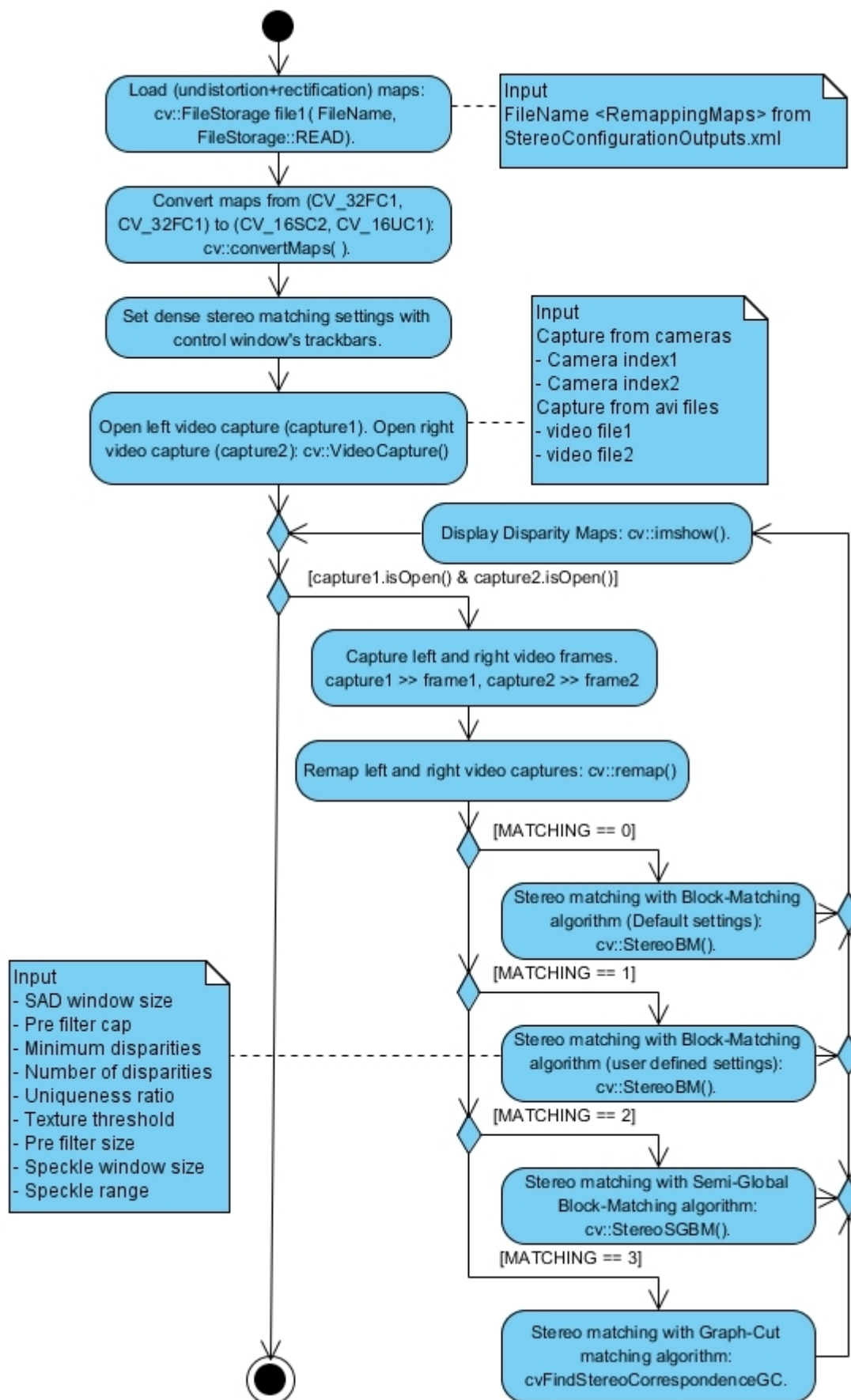


Figure 18.22: Dense stereo matching approach.

With the **MATCHING CONTROLS** trackbars (see Figure 18.23) is possible to adjust the Block-Matching state settings in real time and update the state settings for each new pair of video frames captures. The matching method is selected over the window named **CAPTURING FROM AVI FILES** or (**CAPTURING FROM CAMERAS**) were the left and right rectified capture are being displayed.

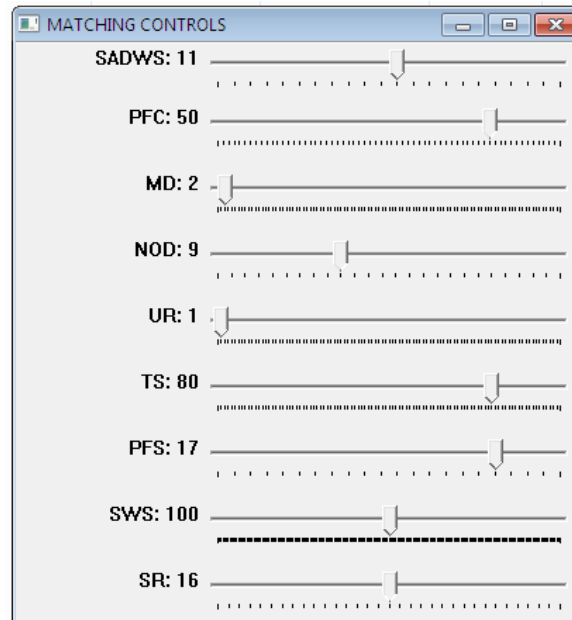


Figure 18.23: Block – Matching control window.

The **MATCHING** trackbar values available are [0 1 2 3] corresponding to [Block-Matching, Block-Matching (with user defined settings), Semi Global Block-Matching, and Graph-Cut] methods, respectively. Some of the most important block matching settings are adjusted using the trackbars, from **MATCHING CONTROLS** window, described next:

- **SADWS:** Sets the sum of absolute differences window size for block matching algorithm.
- **PFC:** Sets truncation value for the prefiltered image pixels.
- **MD:** Set the minimum number of disparities.
- **NOD:** Sets the maximum number of disparities minus the minimum number of disparities.
- **UR:** The margin in percent by which the best computed cost function value should overtake the second best value to consider the pixel match correct.
- **TS:** Sets the texture threshold.
- **PFS:** Sets the truncation interval values for the prefiltered image pixels.
- **SWS:** Sets speckle window size.
- **SR:** Sets maximum disparity variation within each connected component.

**Option [2]** allows to perform the stereo matching process for a sparse set of points. This approach makes use of the sparse Lucas-Kanade Pyramid optical flow method to track sparse set of points, selected by the user, over the left image and then look for those points in the right image. By selecting this option the program captures the first frame from each video sequence and waits for the user to introduce a number of points by left mouse clicking over the left image window named **LEFT IMAGE**. The optical flow settings are changed by using **POINTS TRACKING CONTROLS** window's trackbars as displayed in the next figure (see Figure 18.24).

When all points are introduced the program starts by capturing the next video sequences and the points tracking between left image (previous capture) to the left image (current capture) and then to the right image (current capture) is done simultaneously. The points that are tracked successfully are stored into two vectors (left and right image points) and the points that failed to be tracked correctly at any time are excluded as well its previous tracked positions.

Using the left and right image points (matched points) the program then computes the points disparities and, using the disparity-to-depth matrix, it reprojects the 2D points to 3D world space.

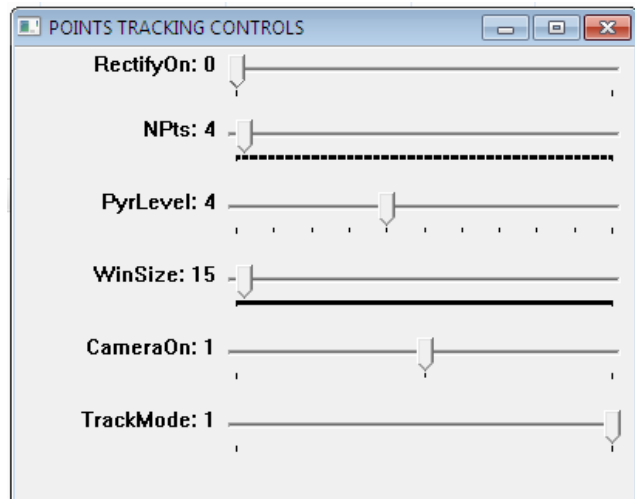


Figure 18.24: Image points tracking settings.

The POINTS TRACKING CONTROLS window trackbars' description is given as follows:

- **RectifyOn:** If the value is set to 0 the tracking procedure only corrects the images distortions (undistortion). In the case the value is set to 1 the tracking procedure remaps the image (undistortion+rectification).
- **NPts:** Sets the maximum number of points to be tracked. After this number of points being added no more points are allowed to be tracked. This setting is used to tell the program until when it should wait in stand-by to allow the user to add points.
- **PyrLevel:** Defines the number of pyramid levels to be used by the Lukas – Kanade Pyramid algorithm.
- **WinSize:** Defines the size of the search windows of each pyramid level.
- **CameraOn:** Defines which camera is active, this tells the program to which camera referential the new 3D space points should be related. Value 1 defines the left camera referential, and value 2 defines the right camera referential.
- **TrackMode:** If the trackbar value is 1 the program captures the first left and right video frame and waits for the user to introduce the number of points defined by NPts. If the value is set to 0 the program starts capturing and displaying the video sequences and at any time the user is allowed to add points to be tracked.

The two operations that are available while capturing the video sequences are described next:

- **“Esc” Key:** By pressing “Esc” key the program finish the video capture and if any points was tracked successfully until the present moment the program computes the image points disparities and reproject the 2D points to 3D space, all the 3D points are related to both left and right camera coordinate system. The resulting points  $[x, y, \text{disparity}]$  and  $[X, Y, Z]$  vectors are then saved into two XML files named  $[\text{FileName}]_{3DPOINTSC1.xml}$  and  $[\text{FileName}]_{3DPOINTSC2.xml}$  for the left and right camera respectively.
- **“C” Key:** By pressing “C” key while capturing video sequences, if any points is currently being tracked, the program reproject the 2D image points to 3D space and saves the results into one of the files (depending on which camera is active) as described in the previous bullet.



In the next figure (see Figure 18.25) is presented the activity diagram that better describes the stereo matching approach by using Lucas-Kanade Pyramid tracker algorithm.

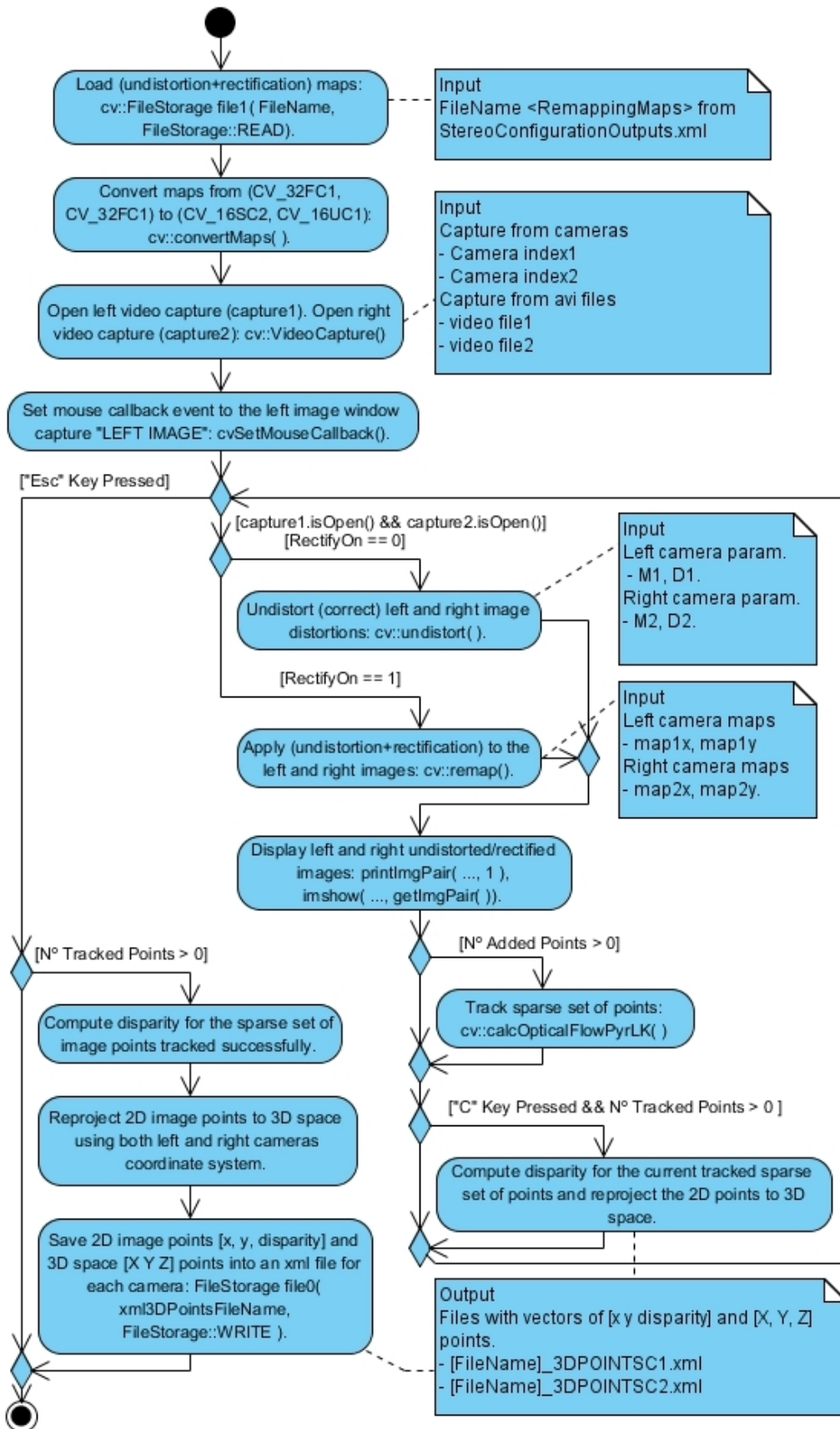


Figure 18.25: Sparse stereo matching approach.

### ***18.7 Main menu's Option [5] – Rotation matrix parametrization***

This Main menu option allows the stereo configuration rotation matrix parametrization into Euler angles and quaternion. The program opens the calibration output file obtained from the last calibration process listed inside the main XML output file - **StereoConfigurationOutputs.xml** (the file name stored inside the <CalibrationParameters> tags).

By selecting Main menu's **Option [5]** the program loads the stereo configuration rotation matrix and presents a sub menu as the one illustrated in the next figure (see Figure 18.26).

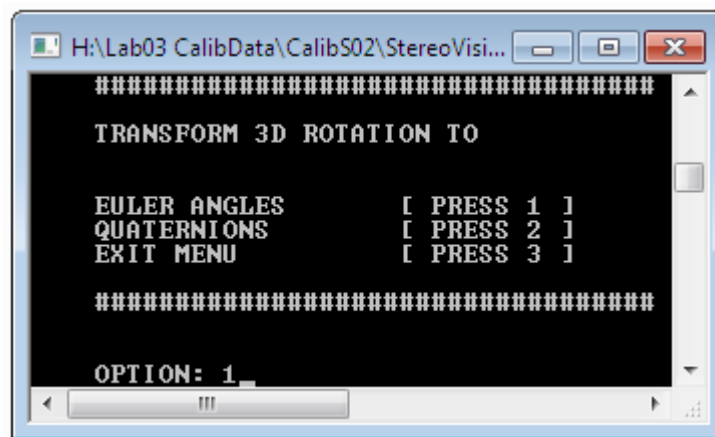


Figure 18.26: Rotation matrix parametrization.

Selecting **Option [1]** transforms the 3D orthonormal rotation matrix, that brings the right stereo camera to the left camera's orientation, into Euler angles. Two solutions are computed if no Gimbal Lock problem is found otherwise only one solution is computed.

Selecting **Option [2]** transforms the 3D orthonormal rotation matrix into quaternions and then all the four quaternion components are normalized.

Selecting **Option [3]** exits the sub menu and if any transformation was performed it saves the resulting parameters (Euler angles or quaternion, or both) into an XML output file named [FileName]\_Angles.xml. This functionalities were used during the research to study the axes rotation angles and allow to take further conclusions for the final 3D points recovering results.

### ***18.8 Main menu's Option [6] – List current directory files***

By selecting Main menu's **Option [6]** the program lists all the files in the current directory. This class was implemented to allow certain type of files to be listed and presented for input purposes such as the text files for calibration purposes and AVI video file inputs. It was also used to check if certain files names already exist in the current directory and avoid to overwrite them when performing images saving and video recording operations.