



TAMPERE UNIVERSITY OF TECHNOLOGY

ALI KARAOGLU

A GENERIC APPROACH FOR DESIGNING MULTI-SENSOR 3D
VIDEO CAPTURE SYSTEMS

Master of Science Thesis

Examiners: Dr. Atanas Gotchev
M.Sc. Mihail Georgiev

Subject approved by departmental
council on 06 April 2011

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY

Master's Degree Programme in Information Technology

KARAOGLU, ALI: A Generic Approach for Designing Multi-Sensor 3D Video Capture Systems

Master of Science Thesis, 59 pages

September 2011

Major: Signal Processing

Examiner: Dr. Atanas Gotchev, M.Sc. Mihail Georgiev

Keywords: stereo, multi-view, depth, camera, calibration

The increased availability of 3D devices on the market raises the interest in 3D technologies. A lot of research is going on to advance the current 3D technology and integrate it into market products. Video gaming, displays, cameras, and transmission systems are some examples of such products. However, all of these products use different input 3D video formats. For instance, there are many types of 3D displays that work with different inputs like stereo video, video plus depth map or multi-view video. To provide input for these displays, the development of adequate 3D video capture systems is required. 3D video capture systems in general consist of multiple cameras. They also include tools for interfacing the cameras, synchronizing the captured video and compensating the physical disorientation effect of assembly process over hardware devices.

A drawback of the current approaches for data capture is that each of them is suitable only for a specific display type or specific application. Capture systems should be redesigned for each of them particularly. There is no single solution that can provide 3D streams for the multitude of 3D technologies.

The main objective of this thesis is to develop a generic solution that can be used with different camera array topologies. A system is developed to interface multiple cameras remotely and capture video from them synchronously in real-time data acquisition. Several computers are used to interface multiple cameras and a physical network is setup to build communication lines between the computers. A client-server software is implemented to interface the cameras remotely. The number of cameras is scalable with the flexibility of software and hardware of the system. This approach can handle integration of different video camera models. Moreover, the system supports integration of depth capture devices, which delivers depth information of the scene in real time. Calibration and rectification of the proposed multi-sensor camera array setup are supported.

PREFACE

This thesis was made for Department of Signal Processing in Tampere University of Technology.

I would like to thank my thesis supervisor Dr. Atanas Gotchev for all the good advices. Besides, Mihail Georgiev deserves special thanks for reviewing my thesis many times.

And of course, none of these could have happened without my parents, my elder sister Esin Guldogan and her family, my precious friends and their gratuitous love and faith in me. Thanks again.

Tampere, September 19, 2011

Ali Karaoglu
Opiskelijankatu 15 B 47
33720 Tampere
Tel: +358440816681

CONTENTS

1	Introduction.....	1
1.1	Definition of the Problem	3
1.2	Organization of the Thesis	3
2	Camera CaptuRe Model and Calibration	4
2.1	The Pinhole Camera Model	4
2.2	Optical Distortions	6
2.3	Image Rotation and Translation	8
2.4	Camera Calibration.....	11
3	Multi-Sensor Camera Array Capture Systems.....	13
3.1	Stereo Systems	13
3.1.1	Epipolar Geometry	14
3.1.2	Stereo Calibration and Rectification.....	16
3.1.3	Stereo Matching	18
3.1.4	Depth Estimation	19
3.2	Multiple Camera Capture Topologies	20
3.3	Implementation of Multi-Sensor Camera Capture Systems	22
4	Proposed Approach for Multi-Sensor Capture Systems.....	23
4.1	Mechanical and Hardware Setup.....	25
4.2	Software Modules.....	26
4.2.1	Server-Side Software Module	27
4.2.2	Client-Side Software Module.....	30
4.2.3	Calibration Software Module	31
4.3	Triggering and Synchronization.....	33
5	Results	35
5.1	Performance evaluation of the System.....	35
5.2	Applications of the System	36
5.3	Calibration Results	37
6	Conclusions.....	40
7	References	42
8	Appendix 1: Software Documentation	47
8.1	Server Side Software Module	47
8.1.1	Code Structure.....	49
8.2	Client Side Software Module	50
8.2.1	Code Structure.....	51
8.3	Calibration Software Module	51
8.3.1	Code Structure.....	53
9	APPENDIX 2: Tehcnical Manual.....	54
9.1	Camera Installation and Compilation	54
9.2	Data Structures and Protocols	56
9.3	Used Hardware.....	57
10	APPENDIX C: Calibration Results	58

ABBREVIATIONS AND NOTATIONS

2D	Two Dimensional
3D	Three Dimensional
4D	Four Dimensional
API	Programming Unit
BM	Block Matching
DTL	Direct Linear Transformation
<i>E</i>	Essential Matrix
<i>F</i>	Fundamental Matrix
FAST	A high-speed corner detection algorithm.
GC	Graph-Cuts
GFTT	Good Features to Track
HD	High Definition
HVS	Human Visual System
IVT	Integrating Visual Toolkit
LCD	Liquid Crystal Display
LDRI	Laser Dynamic Range Imager
<i>M</i>	Camera Matrix
MSER	Maximally Stable Extremal Regions
O	Optical Center
OpenCV	Open-source library for computer vision
P	Principle Point
<i>P</i>	Projection Matrix
PMD	Photonic Mixer Device
PNG	Portable Network Graphics
<i>R</i>	Rotation Vector
SGBM	Semi-Global Block Matching
SIFT	Scale-Invariant Feature Transform
SMB	Sub-Miniature Connector (Version B)
SURF	Speed Up Robust Feature
SVN	Software Versioning System
<i>T</i>	Translation Vector
ToF	Time-of-flight
USB	Universal Serial Bus
VXL	the Vision-Something-Libraries for Computer Vision

LIST OF FIGURES

Figure 2.1 The Pinhole camera model	4
Figure 2.2 Misalignment of the pinhole plane and sensor a)Center of the sensor is not on the same line with optical axis b)Pinhole plane is not parallel to the image plane	5
Figure 2.3 Tangential and Radial distortions	7
Figure 2.4 A two dimensional rotation with an angle θ in Euclidean Space	8
Figure 2.5 Three dimensional rotation angles θ , ψ , φ	10
Figure 2.6 Different types of calibration patterns: dots, chessboard, circles, 3D object with Tsai grid [3]	11
Figure 2.7 Chessboard calibration pattern with different orientations	11
Figure 2.8 Estimated camera positions (extrinsic parameters) with respect to the calibration pattern (Camera centered).....	12
Figure 3.1 Principals of binocular vision	13
Figure 3.2 Stereo camera system.....	14
Figure 3.3 a)Point Correspondence Geometry, b)Epipolar Geometry	15
Figure 3.4 Orientation differences in stereo setups. a)Ideal stereo camera setup, b)Optical centers are not aligned on z-axis, c)Optical centers are not aligned on y-axis, d)Optical centers coordinate system angle difference over y-axis (yaw), e)Optical centers coordinate system angle difference over z-axis(roll), f)Optical centers coordinate system angle difference over x-axis (pitch)	16
Figure 3.5 a)Randomly located cameras, b)Rectified cameras	17
Figure 3.6 Rectified image pairs.....	18
Figure 3.7 A depth map extraction of “rocksl” samples with semi-global block matching. From left to right respectively, left image, right image and obtained disparity map.....	19
Figure 3.8 The Stanford Multi-Camera Array.....	20
Figure 4.1 Proposed setup for multi-sensor camera array capturing systems	23
Figure 4.2 Examples of developed Multi-Sensor Camera Array System a)Stereo Camera setup, b)Camera Rig Array (Proposed setup), c)Scattered camera setup, d)Targeted scene, e)Capture process on computer	24
Figure 4.3 Framework of Multi-Sensor Camera Array System	26
Figure 4.4 Flow of Multi-Sensor Camera Array System.....	27
Figure 4.5 A visual representation of Class Relations of Multi-Sensor Camera Array System – Server Side	29
Figure 4.6 Task flow realization of threaded real-time capturing for two cameras	30
Figure 4.7 Class Relations of Multi-Sensor Camera Array System – Client Side	31
Figure 4.8 Hardware Triggering Solution	34
Figure 5.1 An example of color coded dense depth map with color bar	35
Figure 5.2 Image samples of captured videos from different scenes. From left top to right: a)Chess playing, b)Corridor ball playing, c)Indoor billiards, d)Bikes, e)Close-up presentation, f)Floor ball tracking	36

Figure 5.3 Captured calibration pattern images from left to right respectively: left camera image, center camera image, right camera image, PMD camera intensity image	37
Figure 5.4 Images from cameras from left to right respectively left, center, right and PMD ToF.....	38
Figure 5.5 Disparity maps before rectification process with SGBM algorithm.....	38
Figure 5.6 Left and center images after rectification process and estimated disparity map of them with SGMB algorithm	38
Figure 5.7 Center and right images after rectification process and map of them with SGMB algorithm	39
Figure 5.8 Center and PMD-depth images after rectification process.....	39
Figure 8.1 GUI of Server Side Software	47
Figure 8.2 GUI of PMD Camera Settings Dialog	48
Figure 8.3 GUI of PMD Camera Settings Dialog	48
Figure 8.4 GUI of Video/Image Save Dialog.....	49
Figure 8.5 GUI of PMD Video Player	49
Figure 8.6 GUI of Client Side Software	50
Figure 8.7 GUI of Start Capturing on Servers Dialog	51
Figure 8.8 GUI of System Calibration Software	52
Figure 8.9 GUI of Check Quality Functionality of Calibration Software	52
Figure 8.10 GUI of Depth Estimation Functionality of Calibration Software.....	53
Figure 8.11 GUI of Lens Un-distortion Functionality of Calibration Software.....	53
Figure 9.1 Data Format of Raw Binary PMD data file for intensity, amplitude and range	56

1 INTRODUCTION

Vision is considered as the most complex sense among the five senses. It is able to discriminate objects in the surrounding world by color, shape, 3D space location, etc. In particular, depth information is mainly retrieved by two slight different perspectives delivered by the two eyes. Though there are some visual cues relying on a single eye, such as perspective, shading, shadows, occluding contours, focus-defocus, texture, motion parallax, highlights, and the stereo cue or the stereopsis. The stereopsis is the strongest to facilitate depth perception for human beings.

A camera is a device which captures and saves images. Its name comes from the “camera obscura”, which means in Latin “dark chamber”. In a sense, this device is an imitation of the human eyes. The aperture works as pupil and sensor works as fovea. In addition to this, lenses are included to scale and focus on larger field of view. Current camera technology provides digital sensors for capturing world information by discrete areas of pixels. Digital images representing pixel arrays allow digital processing and further manipulations such as filtering, object tracking, highlighting, depth extraction, etc. During the assembly process of a camera, some misalignments might occur between the sensor, the plane where the pinhole lays, and the lens. These misalignments are parameterized in a matrix called *camera matrix*. The lenses that are used on the cameras cause distortion over the captured images. This distortion is modeled with *distortion parameters*. The camera matrix and distortion parameters are called *intrinsic parameters*.

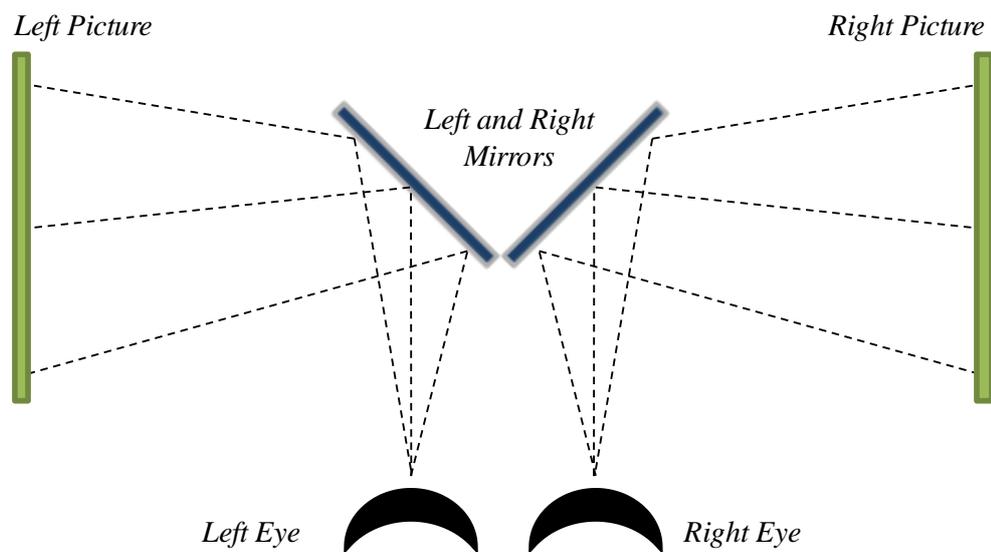


Figure 1.1 Stereoscope principle

Sir Charles Wheatstone discovered that an optical illusion of depth could be obtained from two planar images. He invented a device that projects planar stereo images to related eyes (Figure 1.1). When eyes are focused on the images, the brain processes the images as they are views of real world and perceives depth from them. This device is called stereoscope. With the invention of this device, it is proved that the main cue for depth perception is stereopsis. Therefore, a stereo camera setup is used to capture stereo images to use for this device. The setup consists of two horizontally aligned cameras as a simulation of human eyes. Since the cameras are capturing from different locations, an object from the real world is projected on different locations in the images. The difference between corresponding points in the two images is called *disparity* and can be separated to two components, *vertical disparity* and *horizontal disparity*. The calculation of horizontal disparity is done by matching corresponding points of the same object in stereo images. The search processes of the corresponding points is called *correspondence problem*. In an ideal stereo system, there is no vertical disparity and search process is simple. If there is a vertical disparity, the search process becomes more complex. In practice, it is not possible to produce ideally aligned stereo cameras. Some misalignment occurs between the cameras because of the assembling process. This misalignment is represented by some parameters called *external parameters*. These parameters include the location of the camera, which is represented by three dimensional coordinates, and the orientation of the camera, which is represented by three angles. A process called *calibration* is used to obtain these extrinsic parameters and also the intrinsic parameters. Various researchers studied calibration [1] , [2] , [3] , [4] , [5] . *Rectification* is then applied compensate for the estimated camera positions.

A multi-sensor camera array capture system for generating 3D data has some issues about capture process and post processing. First of all, the cameras have to be synchronized during capturing the scene. Otherwise, the images will not be captured simultaneously, which causes problems for the matching process of the corresponding points. To resolve the synchronization problem, software and hardware triggers are used. Secondly, the capture process is managed by software. This helps to store data and adjust the camera capturing parameters like brightness, gain, aperture, etc. If several cameras are used, only one computer is not sufficient to interface all cameras. Thus, multiple computers are used, and software implementation is done for camera interfacing over these computers. Depth estimation algorithms are not sufficient for obtaining good depth map in real-time. Depth capture devices are used for this purpose. Currently, two main methods are used in these devices. First one is Time-of-Flight [6] and the other one is structured-light 3D scanner [7] . These cameras obtain the depth information of the scene in real time. The data that is retrieved from the depth capture devices contains noise in both methods, and this noise can be eliminated with noise reduction algorithms. The integration of a depth camera to a video camera setup reduces the work for getting depth information. On the other hand, depth capturing devices has to be calibrated and rectified in order to use the data for 2D and 3D fusion.

The advancements in 3D capture devices is related with the 3D displays. These displays can be categorized into four general groups: stereoscopic, auto-stereoscopic, and volumetric and holographic displays. Each group uses different type of data such as stereo video or video plus depth map, or multi-video. Some displays generate multiple images from 2D plus depth map format, which is still being researched [8] , [9] , [10] , [11] . However, rendered images are not that satisfactory from the observers' point of view. If there are several views needed to be generated, more than 2D and a depth map are required. To obtain input images for these displays, a multi-view camera capture system is needed.

1.1 Definition of the Problem

In this thesis, a generic solution for continuous image capturing with multi-sensor camera arrays is targeted. This system allows to remotely interface multiple cameras, capture synchronized data and save it in real time. The scalable hardware and software are able to interface multiple cameras with several computers. The computers are connected each other via a network. In the system, each computer has server side software, which interfaces two cameras and communicates with the client side software.

Integration of depth cameras is accomplished to this generic solution as well. Thus, the server side software and client side software are also able to work with depth cameras. In addition to the capture software, an application is implemented to operate calibration and rectification process. This implementation is able to calibrate the multi-sensor cameras and a depth camera with the given calibration images. Moreover, it rectifies the images with the calibration results and provides rectified images for 3D video storage or transmission or depth extraction.

1.2 Organization of the Thesis

In Chapter 2, a general overview of the pinhole camera capture model is given. Furthermore, the distortions that are caused by the lenses, camera calibration and pose estimation are discussed. In Chapter 3, the calibration of stereo cameras, epipolar geometry of the stereo systems, rectification process and different multi-sensor camera array capture systems for 3D capturing are explained. Chapter 4 presents a generic approach for different multi-sensor camera array capture systems. Performance evaluation of the system and the calibration results are represented in Chapter 5. Conclusions and future work are given in Chapter 6.

2 CAMERA CAPTURE MODEL AND CALIBRATION

Euclidean Geometry is formed by distances, measurement and angles. In this geometry, if two lines lie on the same plane and they do not intersect, they are called parallel lines. We think this geometry represents the real world around us clearly. However, Euclidean Geometry is just a particular aspect of a more general one known as *Projective Geometry*. It originates from principles of perspective, where the parallel lines meet at infinity. Moreover, distance, size and angles are irrelevant in this non-metrical form of geometry. When the imaging of real world is considered, a change of coordinate system from real world to camera world occurs. The projective geometry explains how real world is observed and mapped on a camera sensor, which provides a capture model.

2.1 The Pinhole Camera Model

A *pinhole camera* is a light proof small box, which has a pinhole on one side and sensor on the opposite side. The pinhole directs light coming from the scene to the sensor as an inverted image (Figure 2.1). This image acquisition model is called *Pinhole Camera Model*. It maps the coordinates of a 3D point from the real world to its projected coordinates in an ideal pinhole camera. Current digital technology of camera sensors captures the projected light, and stores them by digital sensors. These projections of 3D world in 2D images are represented by pixels.

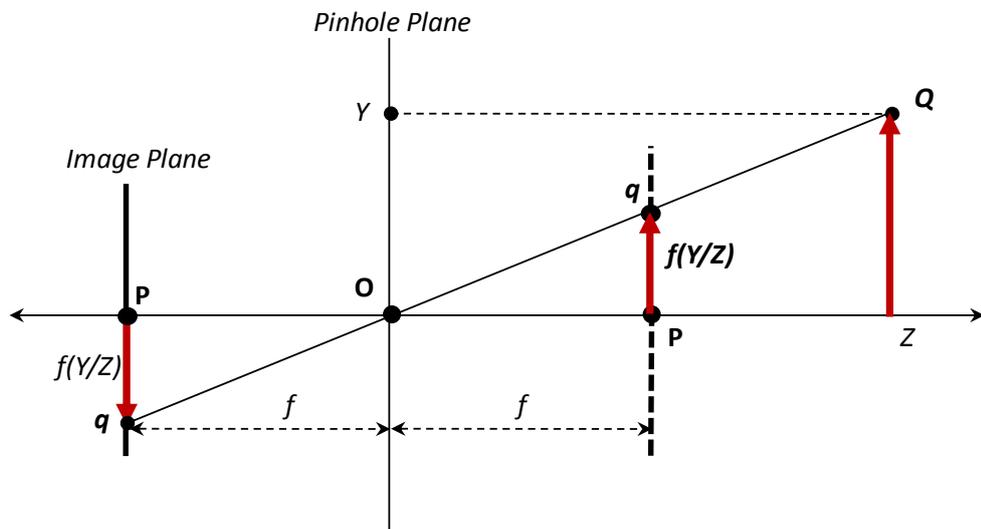


Figure 2.1 The Pinhole camera model

In the pinhole camera geometry, the pinhole projects the image on *Image Plane*. The center of this image plane is called *Principle Point P*. The optical center of the system **O** is the same point with the pinhole of the model, and it lies on a plane called *Pinhole Plane*. The distance between **O** and **P** is called focal length, and denoted by f . $\mathbf{Q} = (X, Y)$ is the vector of coordinates of an object in the real world and $\mathbf{q} = (u, v)$ is the vector of object coordinates on the image plane. The distance of the object from **P** is Z . With the rule of similar triangles, the relation between $(\mathbf{O}, \mathbf{P}, \mathbf{q})$ and $(\mathbf{O}, Z, \mathbf{Q})$ triangles is given by the formulas:

$$u = f \left(\frac{X}{Z} \right), \quad v = f \left(\frac{Y}{Z} \right). \quad (2.1)$$

In this case, the distance d is the same with the third coordinate of \mathbf{Q} . Here is the relation of \mathbf{Q} and \mathbf{q} :

$$\mathbf{q}(u, v) = \mathbf{Q} \left(f \frac{X}{Z}, f \frac{Y}{Z} \right). \quad (2.2)$$

Eq.2.2 explains how to map real world coordinates (X, Y, Z) to image coordinates (u, v) in pixels, and this mapping is called *Projective Transform*. The projective transform does not preserve size or angle but it preserves incidence (e.g. points on a line remain on the same line or two intersecting lines will intersect after the transformation) and cross-ratio.

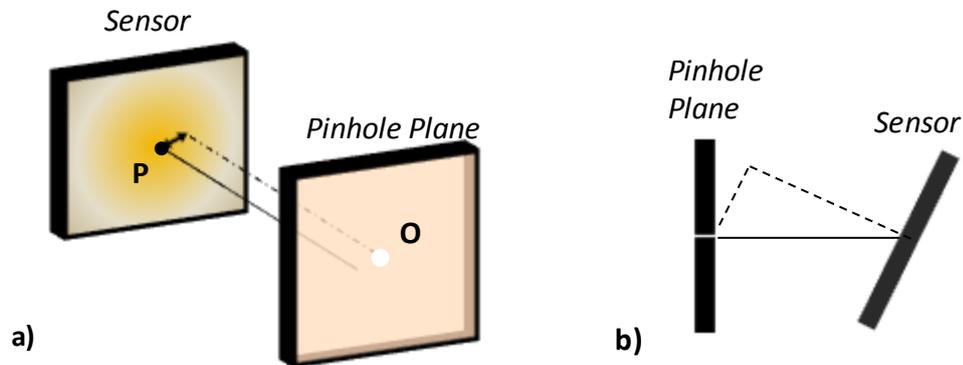


Figure 2.2 Misalignment of the pinhole plane and sensor a)Center of the sensor is not on the same line with optical axis b)Pinhole plane is not parallel to the image plane

The calculation in Eq. 2.2 is done with the assumption that the camera sensor and aperture orientation are ideal. The assembly process of the cameras causes some misalignments over the image plane and pinhole plane. When the image plane is not parallel to the pinhole plane, the focal length of the camera changes (Figure 2.2.b). Therefore, focal length is denoted by f_x and f_y . When the central point of the sensor is not aligned

with the optical center (Figure 2.2.a), the principal point (\mathbf{P}) is represented by two variables, c_x and c_y . Neither the actual focal length (f) nor the sizes of the imager elements (s_x, s_y) can be measured during the calibration process. Thus, only $f_x = s_x, f_y = f s_y$ are derived

$$u_s = f_x \left(\frac{X}{Z} \right) + c_x, \quad v_s = f_y \left(\frac{Y}{Z} \right) + c_y. \quad (2.3)$$

The misalignment coordinates are added to Eq.2.1 and we obtain Eq.2.3. This models the real world camera systems.

The basic idea behind projective transformation is to add additional points at infinity to the Euclidean space. The geometric transformation converts the additional points to normal points or other way around. Thus, while working with *projective transforms*, we add extra coordinates to the points and they become *homogenous coordinates*. These coordinates are related with a point in n^{th} dimension projective space and represented by on $(n + 1)$ dimensional vector, which means a 2D vector is described by a 3D vector or a point in 3D world is described with 4D vector. In homogenous coordinates, real pixel coordinates (u, v) are represented as $q = (x, y, w)$ and the relation between them is represented in Eq. 2.4.

$$(u, v) = (x/w, y/w). \quad (2.4)$$

In homogenous coordinates, real world coordinates are represented as $\mathbf{Q} = (X, Y, Z, 1)$. Using this property, the parameters in Eq. 2.3 are re-arranged into a 3-by-3 matrix and called *Camera Matrix* (\mathbf{M}).

$$q = \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \mathbf{M} [I | 0] \mathbf{Q}, \quad \text{where } \mathbf{M} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \quad (2.5)$$

By Eq.2.5, $w = Z$ will be found out. However, the result is not the 3D locations of the points but the homogenous representation of optical rays. Dividing the homogenous coordinates by w will invert the operation into image pixels back (Eq.2.3).

2.2 Optical Distortions

Lenses aid to gather more light from scene via focusing more light on a point, which allows brighter images. Current technology is not able to produce ideal lenses that can render the straight lines in real world remain as straight lines in the image plane. Therefore, lens causes *distortion* over the image. The process is illustrated in Figure 2.3.

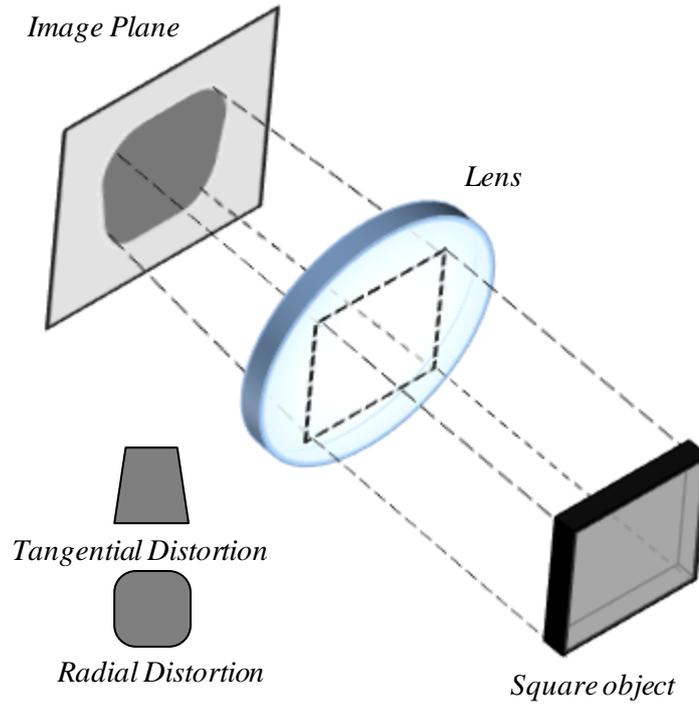


Figure 2.3 Tangential and Radial distortions

The distortions which are caused by current camera lenses are almost non-visible. But this does not change the fact that there is always a distortion even with expensive lenses. Most visible distortions are *radial distortions* caused by the spherical lenses. They can be classified into two groups: *barrel distortions* and *pincushion distortions*. In barrel distortions, the effect of magnification over the image increases when one gets further away from the optical center. Pincushion distortions are the opposite of barrel distortions.

Spherical lenses bend the light rays more when the light gets farther from the center of the lens. Therefore, distortion in the optical center is zero and increases to the edges similar to water drop rings. In mathematics, this property is modeled by the first few terms of Taylor series expansion around $r = 0$;

$$x_d = x + (1 + k_1 r + k_2 r^4 + k_3 r^6), \quad (2.6)$$

$$y_d = y + (1 + k_1 r + k_2 r^4 + k_3 r^6), \quad (2.7)$$

where (x, y) are the original location of distorted pixels and (x_d, y_d) are undistorted pixels in camera coordinates $r = \sqrt{x^2, y^2}$. The first two coefficients k_1, k_2 in Eq. 2.6 and Eq. 2.7 are the important parameters to remove the radial distortion. If the camera does have really high distortion, k_3 is used as well.

Another type of distortion occurs when the lenses cannot be perfectly aligned with the camera sensor. This assembly process error causes *tangential distortion* (Figure 2.3).

The distortion is visible as skewed geometry of the scene and characterized by two coefficients: (p_1, p_2)

$$x_d = x + [2p_1y + p_2(r^2 + 2x^2)] , \quad (2.8)$$

$$y_d = y + [2p_2x + p_1(r^2 + 2y^2)] , \quad (2.9)$$

where (x, y) are the original location of distorted pixels and (x_d, y_d) are undistorted pixels in camera coordinates by $r = \sqrt{x^2 + y^2}$.

In conclusion, there are five parameters which are used for un-distorting process, namely k_1, k_2, k_3, p_1, p_2 . In current lens technology, radial distortions are low and tangential distortion is almost zero. For that reason, the first two of these parameters are the most important for correcting the lens distortions.

2.3 Image Rotation and Translation

The change in the perspective of the 3D world results a change in the components of a 3D vector. The location change of the vector is called *translation*, represented by a 3D coordinate. These coordinates debate the differences from the previous ones. The orientation change of the vector is called *rotation* and it is represented by three rotation angles.

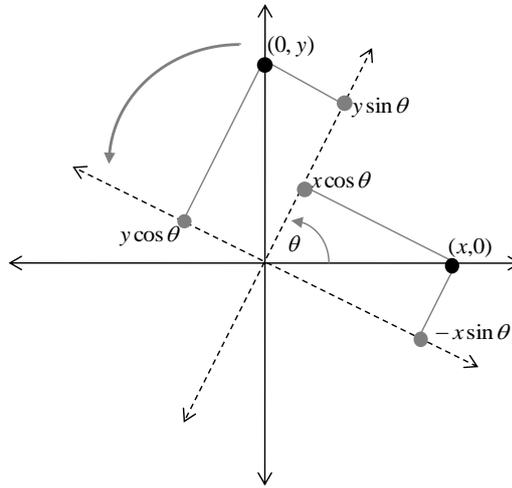


Figure 2.4 A two dimensional rotation with an angle θ in Euclidean Space

We assume that a camera is aligned with a vector where the optical center is at the position and the camera direction is the orientation of the vector. Thus, we can denote the orientation of the camera, which is defined by the *rotation matrix*, and the relative location of the optical center is defined by the *translation vector*. The rotation matrix consists of three rotation angles and the translation vector is defined by a 3D coordinate. These parameters are known as *extrinsic parameters* and to calculate them, a reference

point in the real world coordinate system is selected to be the *center of projection*. This center is the zero point of the system and all other parameters are obtained with respect to this point.

Rotation of an image is done in three dimensions with respect to the optical center. The rotations in axes x, y, z are represented by the angles θ, ψ, φ . Any rotation process can be reversed by the opposite sign angle. After a 2D rotation, the locations of the new coordinates are calculated by a matrix multiplication:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \quad (2.10)$$

To find the new locations of the coordinates in three dimensional space, a matrix multiplication is needed as well. Rotations in three dimension space are shown in Figure 2.5. These rotations are decomposed to three 2D rotations around a static axis. For instance, during the rotation around an axis, it stays steady and the other axes rotate (Figure 2.4). Three rotation matrixes for each of the axes in three dimensional space are calculated by the 2D rotation matrix multiplication (Eq. 2.10) with each rotation angle. Each of them is represented by a matrix:

$$\mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \quad (2.11)$$

$$\mathbf{R}_y(\psi) = \begin{bmatrix} \cos\psi & 0 & \sin\psi \\ 0 & 1 & 0 \\ -\sin\psi & 0 & \cos\psi \end{bmatrix}, \quad (2.12)$$

$$\mathbf{R}_z(\varphi) = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (2.13)$$

where $\mathbf{R}_x(\theta)$ represents the rotation matrix when rotation is done around x -axis, $\mathbf{R}_y(\psi)$ represents the rotation matrix when rotation is done around y -axis and $\mathbf{R}_z(\varphi)$ represents the rotation matrix when rotation is done around z -axis.

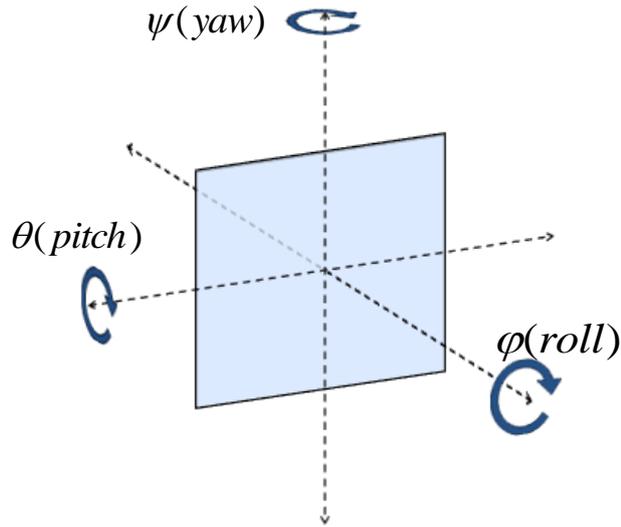


Figure 2.5 Three dimensional rotation angles θ , ψ , φ

The *Rotation Matrix* given in Eq. 2.14 is the product of all 2D rotation matrices. This matrix enables to rotate an image to every directions as illustrated in Figure 2.5.

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \mathbf{R} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \text{ where } \mathbf{R} = \mathbf{R}_x(\theta) \cdot \mathbf{R}_y(\psi) \cdot \mathbf{R}_z(\varphi). \quad (2.14)$$

\mathbf{R} is an orthonormal 3x3 rotation matrix[12] where the inverse process is possible with its transpose matrix.

$$\mathbf{R}\mathbf{R}^T = \mathbf{I}, \det \mathbf{R} = 1, \quad (2.14)$$

where \mathbf{I} is the identity matrix. The translation vector is used to express the position of the optical center. This vector represents the coordinate difference between the origin of the coordinate system and the optical center of the camera. This basic matrix subtraction operation is calculated by:

$$\mathbf{T} = \mathbf{O}_R - \mathbf{O}_c = [X, Y, Z]_R - [X, Y, Z]_c, \quad (2.15)$$

where \mathbf{O}_R is the origin of the reference principal point, \mathbf{O}_c is the camera principal point, and \mathbf{T} is a three dimensional vector.

$$\mathbf{P}_{3 \times 4} = \mathbf{M} [\mathbf{I} \mid 0] [\mathbf{R}_{3 \times 3} \mid \mathbf{T}_{3 \times 1}]_{3 \times 4} \quad (2.16)$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{P} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \quad (2.17)$$

The *Camera Projection Matrix* (\mathbf{P}) is calculated with the multiplication of the camera matrix, the rotation matrix and the translation vector. Thus, it includes intrinsic parameters except the distortion parameters and the extrinsic parameters. In 2D pixel coordinates, (u, v) is used and calculated by Eq. 2.4. World coordinate points are represented by (X, Y, Z) . In Eq. 2.16, homogenous coordinates of those points are used for projective mapping from world coordinates to pixel coordinates.

2.4 Camera Calibration

The camera projection matrix contains four parameters from the camera matrix: f_x, f_y, c_x, c_y , three parameters (angles) from the rotation matrix: θ, ψ, φ and three parameters from the translation vector: x, y, z . Thus, 10 parameters are needed to be calculated in total for the projection matrix. Moreover, the distortion parameters are used to eliminate the distortion over the image. *Camera calibration* is the process to find these intrinsic and extrinsic parameters.

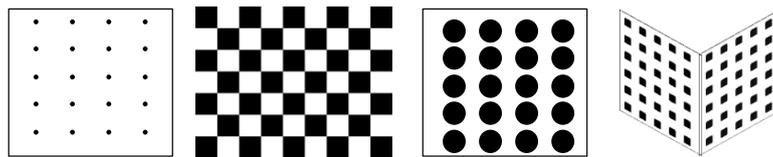


Figure 2.6 Different types of calibration patterns: dots, chessboard, circles, 3D object with Tsai grid [3]

In the calibration process, a planar surface is used to assume a center for the coordinate system of the target. Different types of planar surface calibration objects have been used for calibration. Those are given in Figure 2.6. Some calibration techniques use 3D objects covered with one of the calibration patterns. However, a flat planar chessboard object is easier to deal with. Zhang's technique uses a planar pattern from different orientations [18]. It does not matter which planar object is used as long as the metrics of the object is known. To provide enough information about real world coordinates, images with chessboard are taken from many orientations. This is illustrated in Figure 2.7.



Figure 2.7 Chessboard calibration pattern with different orientations

After capturing the calibration images, the corners or central points of the circles in the images are detected and calculated. Then straight lines are fitted to these detected edges or points to obtain the distortion parameters. The matches of real world coordinates and their projection in the images are found, and the pixel coordinates are converted to homogenous and focal length is calculated (Eq.2.5). Since the distances of the edges in the checkerboards are known, the physical location of the camera as given in Figure 2.8 can be calculated by triangulation (Figure 2.1).

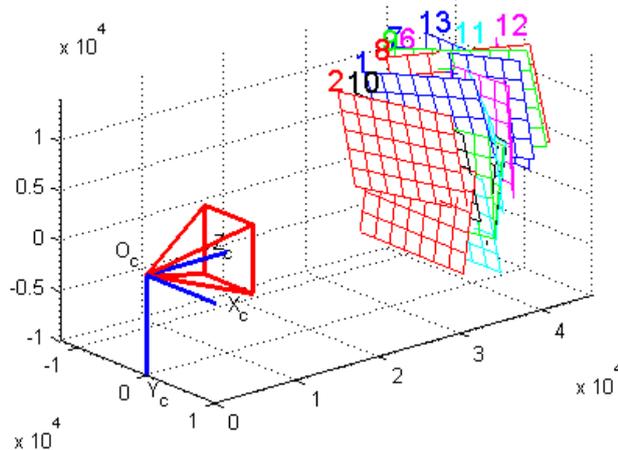


Figure 2.8 Estimated camera positions (extrinsic parameters) with respect to the calibration pattern (Camera centered)

Besides this basic method, there are more comprehensive methods to get the calibration parameters such as photogrammetric calibration [2], [3], [13], self-calibration [1], vanishing points for orthogonal directions [4], [14], calibration from pure rotation [15], direct linear transfer [16], and with implicit image correction [17]. The library that is used in this thesis is OpenCV¹, and it uses the Zhang's method [18] for calibration which is based on maximum-likelihood criterion. OpenCV uses a different method based on Brown [19] to obtain the distortion parameters.

¹ Available online <http://opencv.willowgarage.com/wiki>, retrieved on 18.03.2011

3 MULTI-SENSOR CAMERA ARRAY CAPTURE SYSTEMS

The main aim of this thesis is to overview different types of camera array topologies and to find a solution suitable for all of them. There are many camera array systems that are used for 3D capture such as stereo, aligned and non-aligned multi-sensor camera arrays and camera arrays plus depth camera. An overview of these systems is presented in this chapter.

3.1 Stereo Systems

When eyes capture an object, it appears in different places of the left and right eye (Figure 3.1). After capturing the images, they are converted into electro-chemical signals and transferred to the brain for further processing. The brain combines and extracts the differences of the images to get information about the third dimension of the scene.

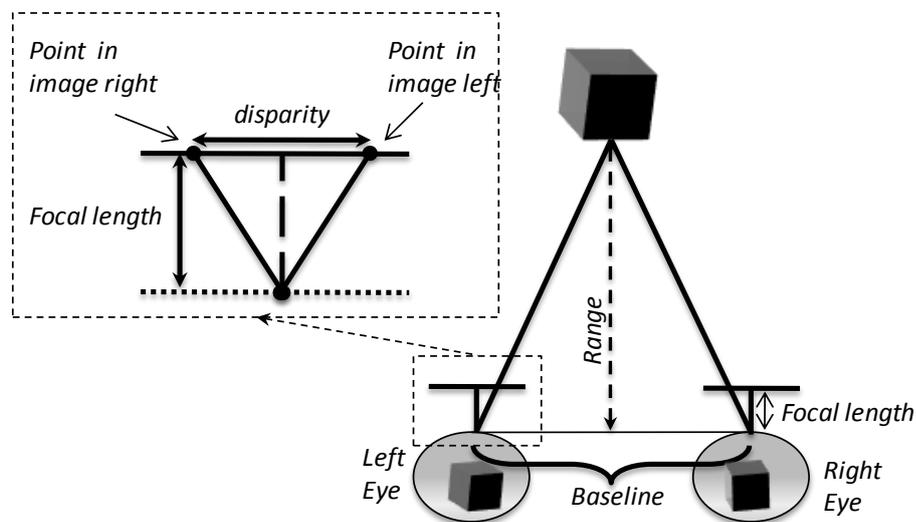


Figure 3.1 Principals of binocular vision

The corresponding points of an object are on different locations in the stereo images. The difference between these points is called *disparity*. It might be presented in 2 dimensions, therefore we distinguish between *vertical disparity* and *horizontal disparity*. The vertical disparity is zero in an ideal stereo setup, because the sensors are ideally aligned on the same plane. The horizontal disparity shows whether the image is close or far from the camera, and it is inversely proportional to the object distance. When the

object is close to the camera, the disparity value is large, and when the object is far away from the camera, the disparity value is small. The calculation of the horizontal disparity for each pixel or window provides the disparity map, which is helpful for estimating the depth of the scene. The distance between the centers of two sensors is called *baseline*.

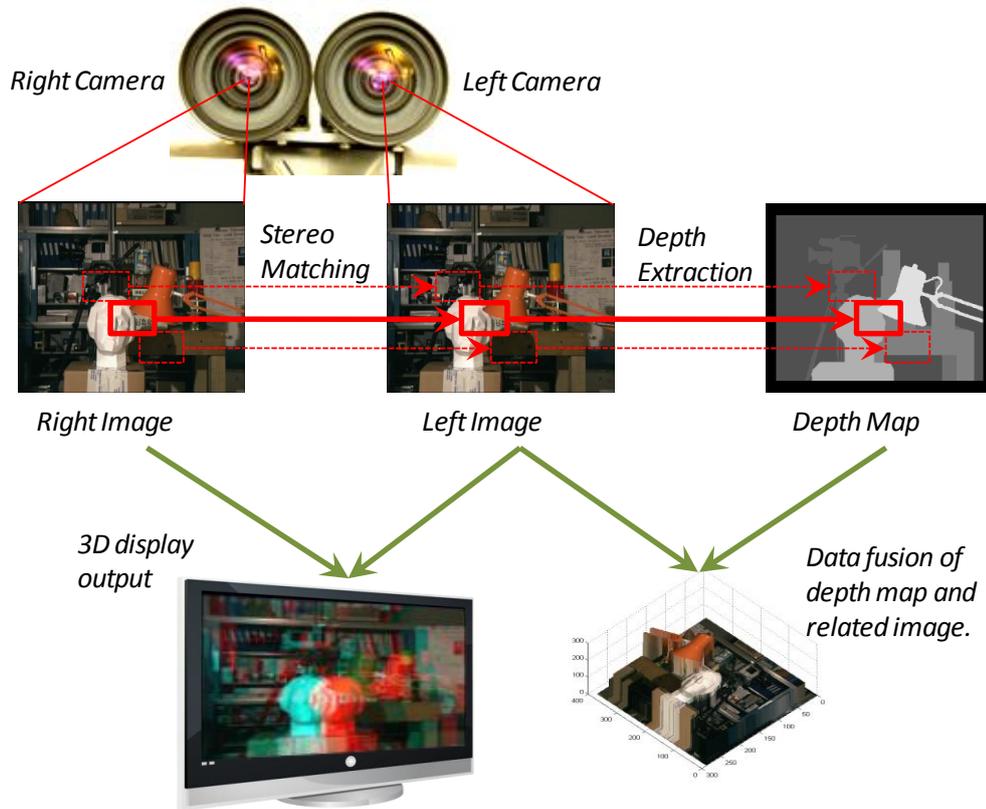


Figure 3.2 Stereo camera system

Current technology provides digital cameras, which uses digital sensors to capture images and pixels store the image data. To save the captured images, a hardware setup is used. Interfacing the cameras and visualizing the captured images are maintained by software. Furthermore, post-processing of the images, as calibration and depth map estimation, is done by software as well. After these steps, the video sequence is ready to be transmitted or used in multi-view stereoscopic displays. The process is illustrated in Figure 3.2.

3.1.1 Epipolar Geometry

The purpose of stereo systems is to retrieve depth information of a scene from stereo images. In this attempt, the relation between the images is important. Finding a projection of a real world point from one image in other image is a complex process. The solution of this issue provides the *disparity map*, which quantifies the difference between points in the two images corresponding to the same world point. This search for match-

ing points is called *correspondence problem*. To approximate the relation between the cameras, *epipolar geometry* is used, which represents the internal projective geometry between two images.

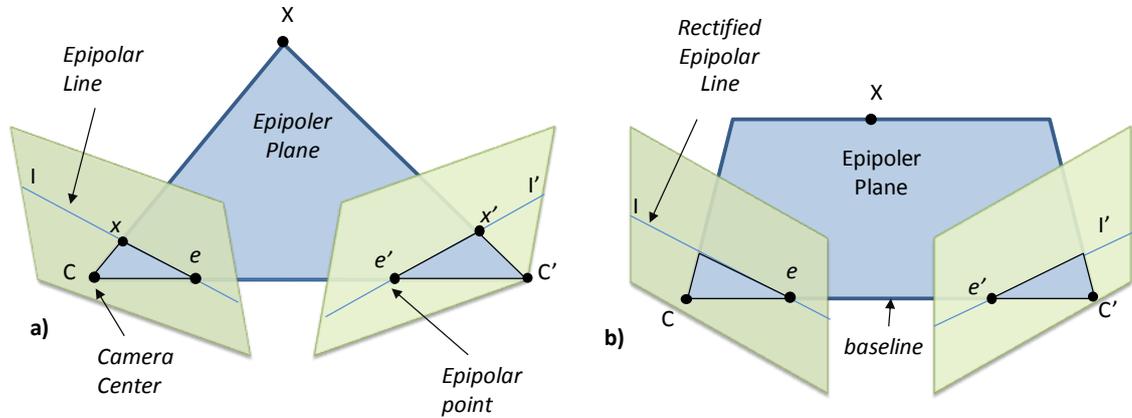


Figure 3.3 a) Point Correspondence Geometry, b) Epipolar Geometry

In stereo camera systems, an object in real world is projected to both camera sensors. Suppose that point X in 3D real world is projected on both images and represented by x and x' . Let's assume that the optical centers of the stereo cameras are C and C' . When we connect these optical centers and the 3D real world point (X), they define a plane called *epipolar plane* (Figure 3.3.a). The intersection of the baseline with the image planes; e and e' , are called *epipolar points*. The lines passing over e and x , or e' and x' are called *epipolar lines*. For each camera center, there is only one epipolar point, although there are numerous epipolar lines. This means all of these lines are passing through in only one point, which is the epipolar point.

For any point x' in one image, there is an epipolar line on the other image and epipolar lines on the one camera are in different place on the other camera. However, every x' points that are matching with x lies on the same epipolar line, which are all on the same epipolar plane. This mapping from points to lines is represented by a 3-by-3 matrix called *fundamental matrix*. Another form of this algebraic representation is called *essential matrix* [20]. The relation between these two matrices is:

$$\mathbf{E} = \mathbf{R}[\mathbf{T}]_x, \quad (3.1)$$

$$\mathbf{E} = \mathbf{M}'\mathbf{F}\mathbf{M}, \quad (3.2)$$

where \mathbf{E} is the essential matrix, \mathbf{F} is the fundamental matrix, \mathbf{R} is the rotation matrix, \mathbf{T} is the translation vector and \mathbf{M} is the camera matrix. The essential matrix is independent from the camera matrix, which means the camera has been calibrated in advance (Eq.3.1), (Eq.3.2). This property makes the essential matrix less complicated than the fundamental matrix, and includes both intrinsic and extrinsic parameters.

3.1.2 Stereo Calibration and Rectification

In camera calibration, the orientation and the location of the camera are obtained with respect to the reference point, which is mostly chosen from the one of the calibration chessboard corners. The number of the cameras in the system does not change the idea of the calibration. In stereo calibration, two orientations and locations of both cameras are obtained. However, selecting one of camera projection centers as the reference point simplifies the calibration process to find only one rotation matrix and one translation vector. The rotation matrix represents the difference of the orientation, and the translation vector represents the location difference between the two camera projection centers. In an ideal stereo setup, the rotation angles are zero and the translation vector is zero except for the horizontal difference, which is given by the baseline. The images captured with this setup are ready for showing on stereoscopic displays. However, cameras and stereo setup have some misalignments that occur from the assembling process (Figure 3.4). Therefore, the rotation angles never become zero and the translation vector never has a zero component.

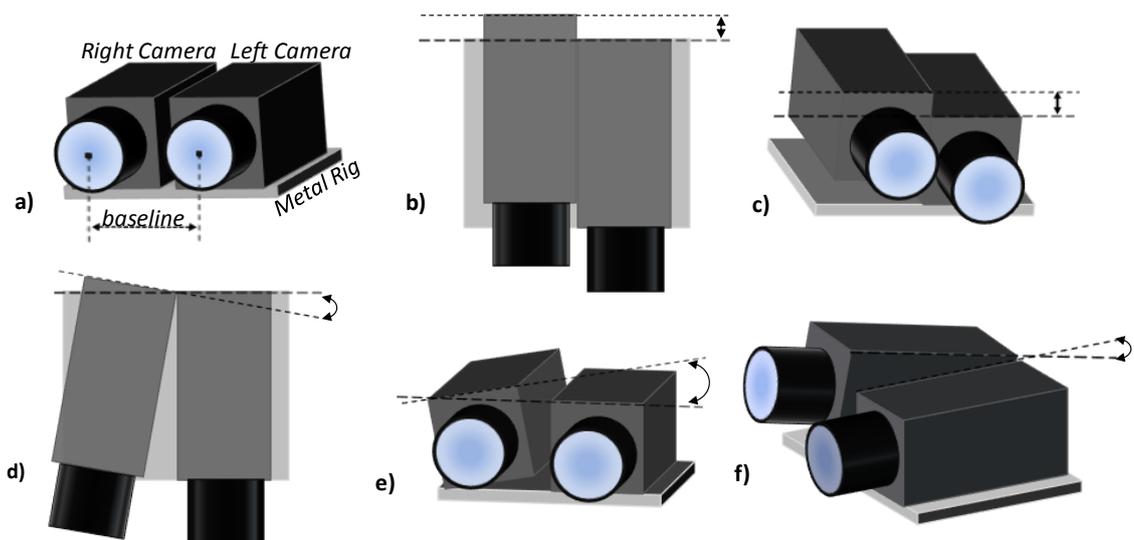


Figure 3.4 Orientation differences in stereo setups. a)Ideal stereo camera setup, b)Optical centers are not aligned on z-axis, c)Optical centers are not aligned on y-axis, d)Optical centers coordinate system angle difference over y-axis (yaw), e)Optical centers coordinate system angle difference over z-axis(roll), f)Optical centers coordinate system angle difference over x-axis (pitch)

In an ideal stereo system, the epipolar lines are parallel to each other. In this case, searching the correspondences of points is easy, because only one dimensional search is done. If the epipolar lines are not parallel, a relation between them is found to keep the easy searching process. This relation between the epipolar lines is given by the *fundamental matrix*. A projection by this matrix makes the epipolar lines become parallel and stereo images well aligned. This process is called *rectification*, and the rectified images

simulate the system as it is working ideally. The result of rectification is illustrated in Figure 3.5.

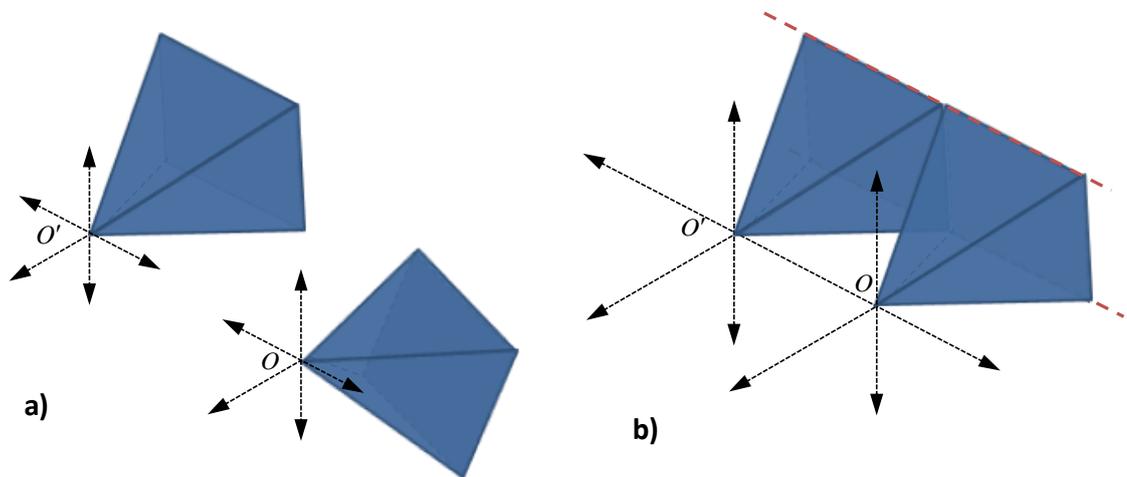


Figure 3.5 a) Randomly located cameras, b) Rectified cameras

Problems with epipolar geometry are solved by applying rectification process on the two images (Figure 3.3.b). Rectification methods can be classified into three groups: planar [2] , [21] , cylindrical [23] , [24] , and polar [25] , [26] . In planar rectification techniques, calibration is needed for un-distorting the images so that less-complex linear algorithms can be applied. On the other hand, non-linear techniques (cylindrical and polar) do not need to apply calibration in advance. However, these methods are more complex. In this thesis, OpenCV library is used for rectification, which contains the implementation of Hartley [4] (non-linear) and Bouguet [27] , [28] (linear) methods.

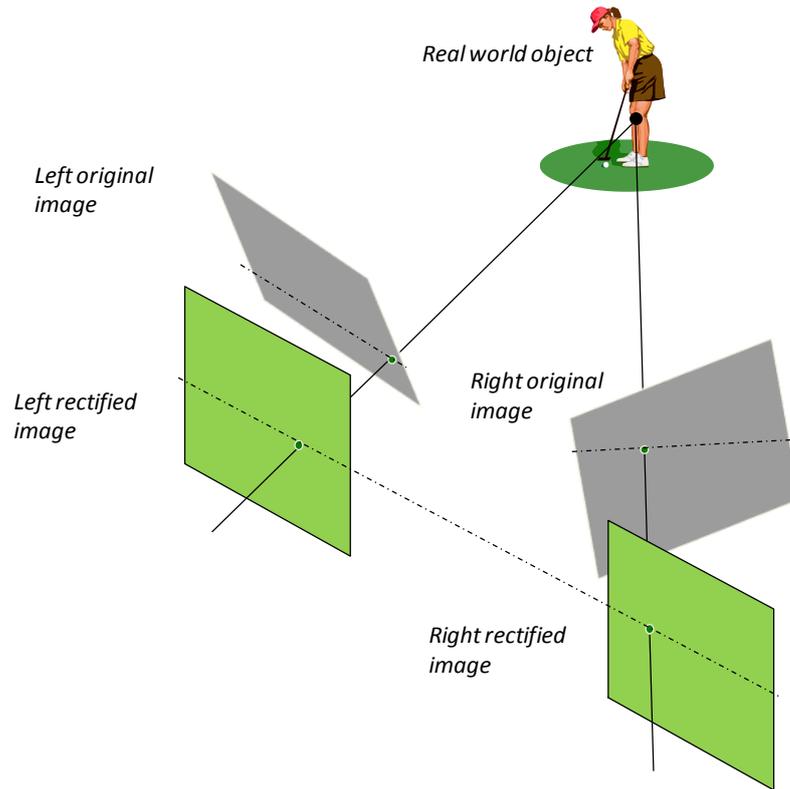


Figure 3.6 Rectified image pairs

3.1.3 Stereo Matching

As discussed before, calibration of the cameras and rectification of the images decrease the complexity of searching the corresponding points of an object in both images. In stereo images, matching the corresponding coordinates is called *stereo matching*. If the camera setup is not ideal, this search is in 2D, and after rectification it is one dimensional search, which makes stereo matching works faster and with higher accuracy (Figure 3.6).

The rectification is not applicable to stereo images when there is no information about the cameras. Thus, stereo matching becomes more complex. There are many stereo matching algorithms that can produce disparity maps, even with non-rectified stereo images. These methods can be classified into two main topics: Local Block Matching [29] and Global Matching [30], [31], [32], [33]. In local matching, the correlation between the neighborhood of a pixel in one image and in the other image is used. Global/Feature based methods use more edges, line segments, etc to find correspondence of them. A survey about taxonomy and stereo matching methods is given in [34]. As shown there, there is a trade-off between fast and accurate techniques where local techniques achieve high speed and global techniques achieve high-quality dense depth maps.



Figure 3.7 A depth map extraction of “rocks1” samples² with semi-global block matching. From left to right respectively, left image, right image and obtained disparity map

When the local structure is similar, block matching algorithms face difficulties to find the matching points [35]. Global matching algorithms are relatively insensitive to illumination changes and give better consequences if there are strong lines or edges. However, most of the global matching techniques are computationally complex and need many parameters to be tuned up. For that reason, other methods have been studied to find the optimum solution like adaptive-window based matching [35], [36], [37] or semi-global stereo matching [38]. In OpenCV library, Graph-Cuts [38] and Block Matching [40] algorithms are implemented already for stereo correspondence.

3.1.4 Depth Estimation

After the disparity map generated, basic triangulation methods over the binocular vision geometry (Figure 3.1) are applied to obtain depth of the pixels, which is calculated by the formula:

$$r = \frac{bf}{Nx}, \quad (3.3)$$

where r denotes the range of the object, b denotes the baseline between the cameras, f denotes the focal length of the camera, x denotes the pixel size of the camera sensor in millimeters, and N denotes the maximum disparity value.

The relation between the range and the disparity is inversely proportional to each other (Eq. 3.3). When the disparity equals 0, the range goes to infinity and vice versa. Moreover, disparity and baseline are directly proportional. Thus, shorter baseline causes shorter disparity and longer baseline causes wider disparity (Eq. 3.3). This information shows that wider baseline is better for further depth ranges. However, there is another trade-off between detecting a particular range and change in the disparity:

² Middlebury 2006 stereo datasets rocks1. Available online : <http://vision.middlebury.edu/stereo/data/scenes2006/FullSize/Rocks1/> , retrieved on 03.09.2011.

$$\Delta R = \frac{r^2}{bf} \Delta N , \quad (3.4)$$

where ΔN is the change in the disparity and ΔR is the change in the resolution of the range. With the smallest disparity increment ΔN , smallest achievable depth range resolution can be determined.

3.2 Multiple Camera Capture Topologies

Linearly arranged multi-camera systems are extended versions of stereo camera systems where additional cameras are added on the sides. It is like combination of many stereo camera pairs. Three and more aligned cameras are not different in case of calibration. In this situation, calibration is done for each camera. A reference camera is selected first, and in most cases this is the central or middle camera, and all other cameras pair with the central camera. Stereo calibrations of the cameras are done for each camera pair with respect to the reference camera, and this generates many different intrinsic and extrinsic parameters that depend on the number of cameras in the system. After this, rectification is done for all pairs in the setup. In this manner, more cameras bring more epipolar constraints and increase the computational complexity. Thus, rectification of multiple camera systems is not as easy as stereo systems [41] , [42] , [43] , [44] , [45] . On the other hand, more cameras provide more confident matches and generate more accurate depth maps.

In multiple camera arrays, there are different distances between cameras which mean multiple baselines in a setup. This ability of the system yields flexibility about determining the disparity and depth of an object.



Figure 3.8 The Stanford Multi-Camera Array

The purpose of multiple camera array system is to capture the scene from different viewpoints and use them in depth extraction or in multi-view stereoscopic displays. Having multiple views facilitate the generation of intermediate images by interpolation [6], [9], [10], [11]. An example of planar camera array setup is the Stanford Multi-Camera Array³ [46], [47], (Figure 3.8).

In dome arranged multiple camera systems, the cameras are scattered over a scene to capture it from various directions. The purpose of the system is mostly 3D reconstruction of the scene. There are many studies aimed to reconstruct objects in the scene [53], [54], [55], [56]. On the other hand, if the scene is static, instead of mounting many cameras, one camera capturing video of the scene is also used to reconstruct the scene [57], [58], [59], [60]. The biggest issue in these reconstruction methods is matching and as in stereo and multi array camera arrays. Improved feature extraction algorithms are used for point matching in 3D reconstruction algorithms such as SURF [61], SIFT [62], Harris Corners [63], FAST [64], MSER [65], GFTT [66] and etc. Doing calibration for each camera, getting the camera matrixes and applying un-distortion to the images the cameras improve the process of 3D reconstruction of a scene.

Depth camera is a device that can measure the depth of the scene and the coordinates of the world in real time. There are different depth cameras based on different principles. The most popular ones are Time-of-Flight (ToF) [5] and Structured-light 3D Scanner [7]. Time-of-Flight devices measures the travel time of a light beam to and from the object. The illumination part of such camera modulates an infrared light to the scene and the image sensor captures the light reflected from the objects. Within this time, as some object are closer to the camera, they reflect the light before those which are further away. Therefore, a time difference occurs between the received light. This duration of travel measures depth information at the camera sensor. In the structured light scanner, there is an illuminator also that emits infrared light. This emitted light is a continuous one. The infrared camera is located somewhere more than 15cm further away from the infrared light emitter, and it captures the infrared light from the scene. Since the light emitter and camera are on different location, the camera captures the shadows of the objects occurred by the infrared light emitter. The magnification calculation of these shadow amounts gives the depth information of the scene.

Stereo camera systems usually have problems with stereo matching on planar smooth surfaces. Stereo matching algorithms cannot give accurate information in this case, and it is complex to calculate it in real time (Section 3.1). Depth cameras are suitable for compensating this problem. Therefore, depth cameras are suitable for object tracking, pose estimation and scene reconstruction [48]. Depth cameras give accurate depth of the scene even though there are some errors. Calculating the depth with respect to the reflection of light is an issue when a surface is reflecting the light to somewhere else. The non-captured light rays cause errors in the depth map. Cons of PMD depth camera

³ Available online: <http://graphics.stanford.edu/projects/array> retrieved on 17.03.2011

and stereo systems are different, which means that they can be used together [49] . However, calibration and rectification should be done for 3D and 2D camera pairs [50] , [51] , [52] .

3.3 Implementation of Multi-Sensor Camera Capture Systems

Different camera systems show varying performance in extracting depth depending on the given scenes. Stereo and multi-cameras system exhibit the problems in finding the correspondences in non-textured surfaces. Depth range sensors have problems with reflecting or absorption of the light. There is no single solution for accurately finding the depth in all types of scenes. Using hybrid systems can compensate the drawbacks of their modules. For instance, using combined information from depth camera and a linearly aligned multi-sensor camera array should result in a better depth estimate.

Camera drivers are responsible for interfacing the cameras and for capturing images. Calibration, rectification and extracting the depth information by local or global stereo matching algorithms are implemented by using computer vision libraries. Examples include: OpenCV, Matlab Camera Calibration Toolbox⁴, Gandalf⁵, VXL⁶, and IVT⁷ .

Managing a multi-sensor camera array system should take into account some issues, such as distance between the cameras, number of cameras and their types, triggering and synchronization of the cameras and the system performance. For instance, using USB connected cameras limits the distance between cameras and the computer because of the USB restrictions. Furthermore, camera synchronization is very important. Images from the left and right cameras in a stereo pair should be taken at the same time so that the points in the pair will match. Using different cameras and long distance between the cameras have negative effect on triggering options as well. Using one computer solves the synchronization issue with a good software trigger, but interfacing many cameras with many computers require the use of hardware trigger.

⁴ J.-Y.Bouguet, Camera Calibration Toolbox for Matlab, available online http://www.vision.caltech.edu/bouguetj/calib_doc/ , retrieved on 18.03.2011

⁵ Available online <http://gandalf-library.sourceforge.net> , retrieved on 18.03.2011

⁶ Available online <http://vxl.sourceforge.net/> , retrieved on 18.03.2011

⁷ Available online <http://ivt.sourceforge.net> , retrieved on 18.03.2011

4 PROPOSED APPROACH FOR MULTI-SENSOR CAPTURE SYSTEMS

In this thesis, a generic approach for multi-sensor capture systems is developed. It allows interfacing several cameras and tackles issues related with distance or synchronization. A scalable array system is developed which can be used for indoors/outdoors scenarios. The system is illustrated in Figure 4.1. It can capture high-resolution image or video data using different camera array topologies such as

- Aligned camera arrays
- Non-aligned camera arrays
- Camera rig arrays using depth capture device

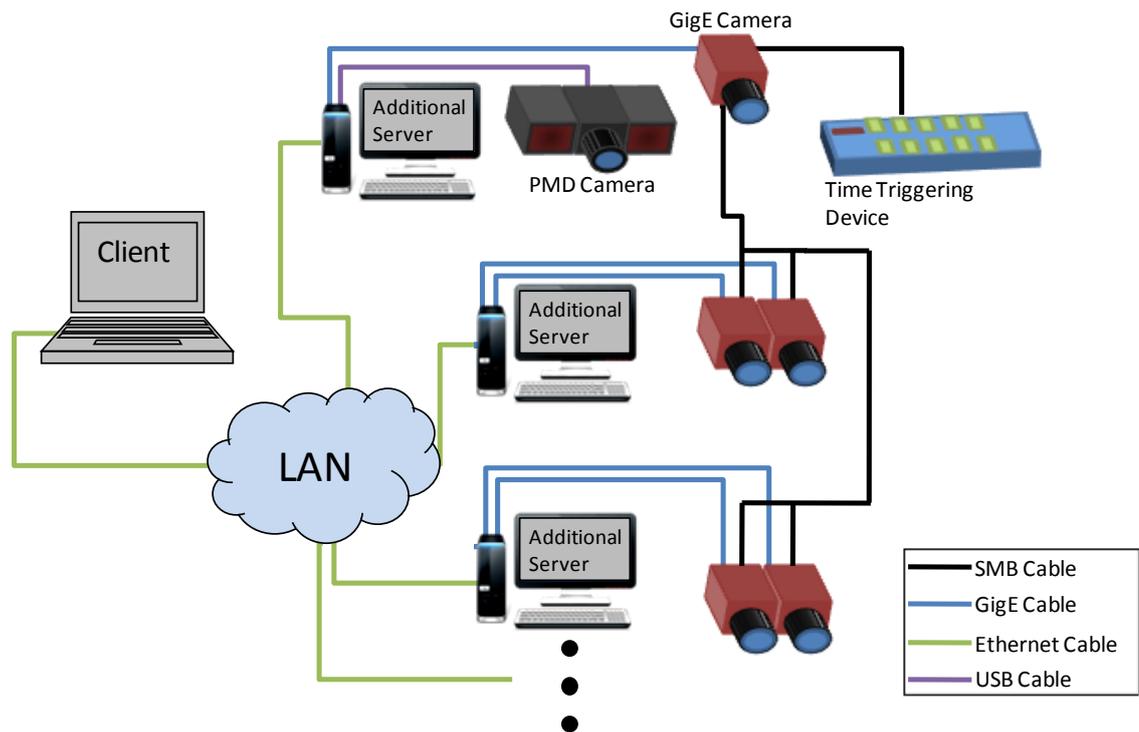


Figure 4.1 Proposed setup for multi-sensor camera array capturing systems

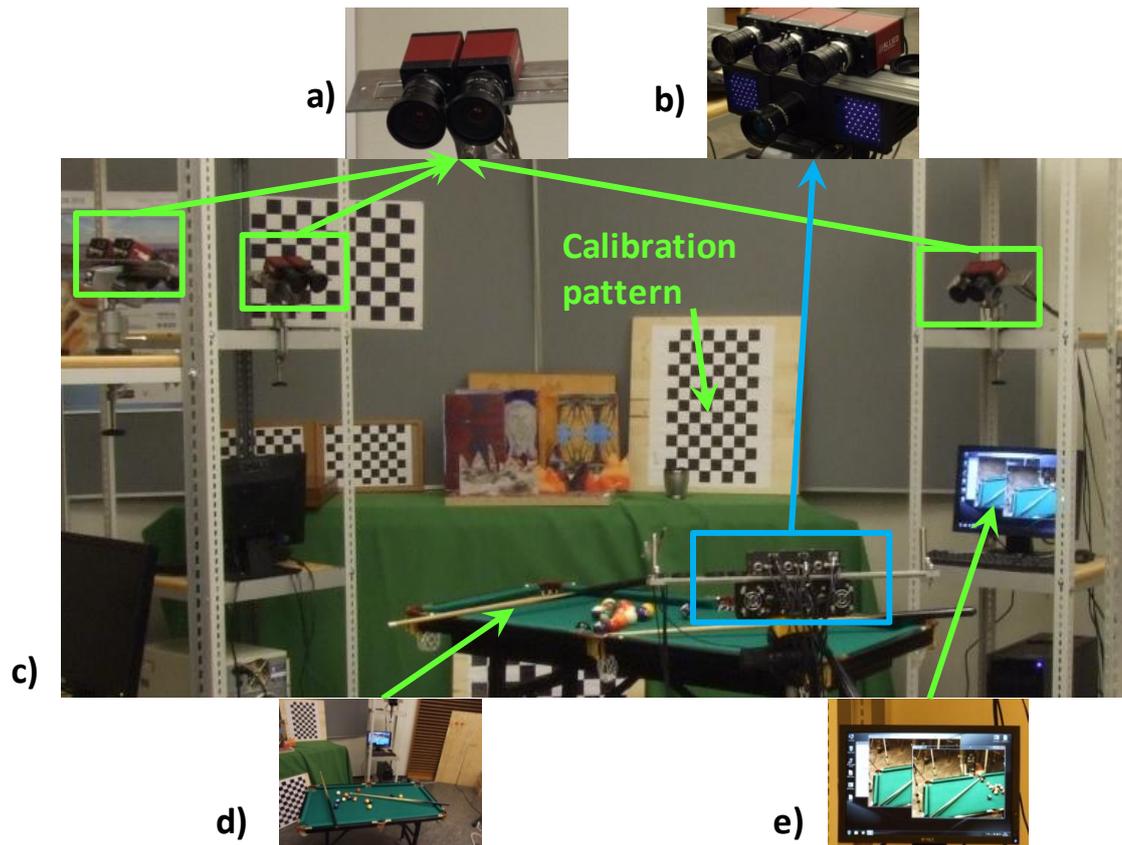


Figure 4.2 Examples of developed Multi-Sensor Camera Array System a) Stereo Camera setup, b) Camera Rig Array (Proposed setup), c) Scattered camera setup, d) Targeted scene, e) Capture process on computer

In the proposed system, there can be many cameras running simultaneously and the cameras can be located in different positions. To tackle issues related with interfacing many cameras and to keep the capturing speed high, multiple computers are used and connected with each other via local area network. A software module, called *Server-Side Software*, is implemented with multi-threading property to run on each computer to interface and to capture video from the cameras. Another software module called *Client-Side Software* is running only on one computer, which organizes the network and controls the computers. This software module includes multi-threading property as well. To synchronize the cameras, both hardware and software triggering are used.

Since there are many types of camera array topologies, a setup is proposed which can simulate many different topologies such as stereo, trinocular and video plus Depth camera systems. In this research setup, a *main* video camera is located in the middle, two *additional* cameras are located on the sides of the main camera and a *PMD* camera located under this setup. This camera array rig is illustrated in Figure 4.2.b.

The main camera is connected to a computer and the additional cameras are connected to another computer which provides high capture speed measured by frames per second, whereas the triggering via software is non-usable in this case. For that reason, hardware triggering is used instead. Hardware triggering eliminates problems with the distance

between cameras. The server computers communicate between each other through local area network.

The *Server-Side Software* runs on computers interfacing one or more cameras. The software is responsible for setting camera parameters and saving images or videos. Besides, it runs as a server, which gets commands and sends information to the client.

The *Client-Side Software* runs on one of the server computers or another computer, which is connected to the network (Figure 4.1). This computer gives commands to the server computers such as start, stop or save videos at the same time. While capturing and saving images or videos, hardware triggering is used to save synchronized frames in all server computers. Otherwise, moving objects in the scene will be in different position in different images. After capturing the scene, to extract depth information from image, calibration and rectification of the system is needed. This issue is handled as well for each stereo pairs in the system. For this purpose, a software module called *Calibration Software* is implemented.

The whole system has 3 main parts: mechanical, hardware, and software. Each of these has to be implemented with care so that the system can work properly and fast. The mechanical part contains the camera setups and rigs to hold the cameras aligned. Tripods and camera towers are included to this section as well. Cameras, computers, network devices, triggering devices and cables for connections are included by the hardware part. To manage the hardware, software is implemented. This software is implemented using open source libraries and hardware driver libraries.

4.1 Mechanical and Hardware Setup

A client/server hardware system is implemented over a local area network. There is a main server computer which is responsible for interfacing a video camera and a depth sensor. Other server computers are meant for interfacing two video cameras. At least one video camera should be connected to the main server. This camera is the central camera, which controls the synchronization of the system. It is responsible for evoking hardware triggering to synchronize the other cameras in the setup. The depth camera, which used in the setup does not have any hardware triggering port. For that reason soft-triggering is used to synchronize it. It should be connected to the main server to be synchronized with the central camera of the system. Additional computers have a role of camera capturing server where all camera devices are connected and server software is started. Each of these additional servers can interface up to two cameras.

Cameras interfaced by Gigabit Ethernet (GigE) technology have been used. This choice provides high-bandwidth for data transfer and ensures distances up to 100m. With these properties, cameras are more mobile than in other solutions like USB or Firewire (IEEE 1394) standards. Moreover, GigE vision standards allow us to use dif-

ferent GigE cameras. *Prosilica-GE1900C*⁸ cameras produced by Allied Vision Technologies have been selected. Such cameras provide images with HDTV format (1080p) resolution with 32 frames per second. The camera allows changing of the settings such as shutter speed, gain, white balance, etc. The computer that interfaces GigE cameras has to have GigE Ethernet. A PMD ToF camera is integrated to the setup, which enables real-time capturing of depth with a resolution of 204x204 pixels [67]. The camera is connected to the computer via USB2.0.

A metal camera rig is constructed to align mutually a horizontal array of the video cameras in fixed or sliding position. It is designed in such a way that the mounting plate of the rig can be fixed over the PMD camera body, and cameras are located over the rig. While mounting, PMD camera is considered always in the middle of the rig so that it can be aligned with the central camera. Main camera is located right over the PMD camera and the rest of the video cameras are located to the sides of central camera. This whole metal rig is mounted on top of a tripod. Other stereo setups are mounted on the camera towers (Figure 4.2.b).

4.2 Software Modules

The overall system includes three modules of software:

- Server-Side Software Module
- Client-Side Software Module
- Calibration-Side Software Module

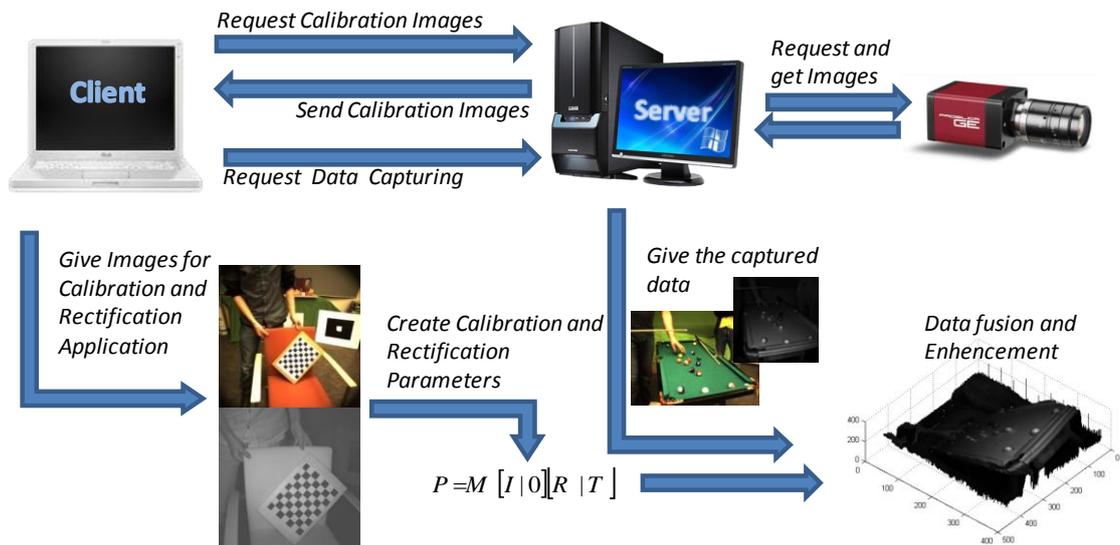


Figure 4.3 Framework of Multi-Sensor Camera Array System

⁸ Available online at: <http://www.alliedvisiontec.com/emea/products/cameras/gigabit-ethernet/prosilica-ge/ge1900.html>, retrieved on 29.08.2011

The main aim of the system is to capture images for 3D imaging purposes. The software modules are designed to tackle the connection problems and to avoid slow processing steps. After setting up the hardware and software on computers, the server-side software module is started on the server computers. The system starts with a request for calibration images from servers by the client. The server side software module get the request, captures images from connected cameras and sends them back to the client. The client gets the images, and checks the checkerboard corners for calibration. If the images are usable, they are saved to a hard drive for calibration. Otherwise, the client sends request for a new image. After getting enough images, the calibration software module starts to generate calibration parameters. These parameters are checked if they are acceptable or not, and they are used to fuse depth images and images after getting video samples from servers. Save video command is also given by the client to all servers at the same time (Figure 4.3).

4.2.1 Server-Side Software Module

This server-side software module is used for interfacing cameras, providing capturing data and running a server to get request from the client. It can maintain up to 3 *GigE* cameras and a PMD camera connected to the same capturing server. To be able to run this software, there has to be at least one connected GigE camera. The graphical user interface of this software is given in Figure 8.1. This software is designed to provide flexibility in capturing 3D data. Thus, the user is allowed to change almost every parameter of the cameras and the system.

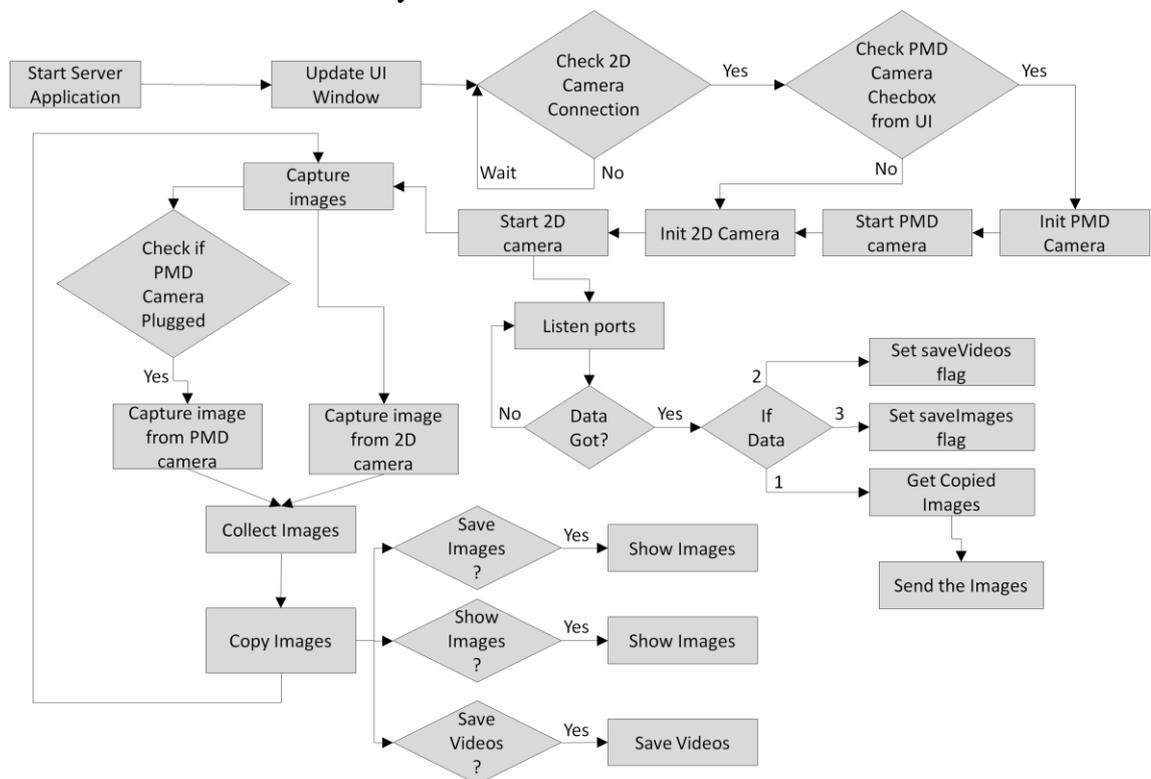


Figure 4.4 Flow of Multi-Sensor Camera Array System

4.2.1.1 Features of the Software

First of all, the software module allows for controlling connected cameras. When it starts, it shows the connected cameras on the display. If there is no camera, refreshing the list helps the user to find the cameras. There is no dynamic connection support for PMD ToF camera. Thus, it is not included to the list. However, the user can decide if the PMD ToF camera will be started or not by clicking on a checkbox in the GUI (Figure 8.1). The software module allows the user to start/stop the cameras and to see the streams. After each action is finished, a feedback is given on the screen as text to the user. The software module listens to ports, sends and receives data via network. The user needs to start this feature to serve to the client computer. This software can be used as a standalone if there is only one computer in the system. Figure 4.4 depicts the system.

Adjusting the camera parameters is important for capturing in different environments such as indoor or outdoor. Pixel format, gain, white balance, shutter speed are some of parameters that can be changed. GigE internet properties of the camera are available as well to optimize the speed of communication between the camera and the computer. The values of package size and byte stream for the frame size are important for optimum speed. In addition to this, PMD ToF camera parameters are adjustable as well, such as integration time and modulation frequencies are the basic variables of PMD ToF camera. The limits and usage of these variables are explained in the software tutorial [67]. The PMD ToF camera provides four different types of data, which are: intensity reflection, amplitude confidence, range and depth vertices. “Intensity” is the scene information based on radiating or reflecting light which is used to have basic gray level image from the scene. “Amplitude” is the magnitude of the change of the oscillating infrared lights reflected from the surface. “Range” gives the depth of the pixels and “depth vertices” give information about 3D world coordinates.

The software also handles saving image and video. The user is able to change parameters such as frames per second, time, and resolution of the video, before starting to save video files. While saving images from all cameras, intensity values of PMD camera and images from 2D cameras are saved as PNG (Portable Network Graphics) files. This is done for easy visualization of the data and for using calibration. Including the intensity values, all of the four data from PMD ToF camera are saved as raw binary file apart from the visualization file. This binary data is used for calibration and data fusion.

4.2.1.2 Design of the Software

The module is designed in layers. This facilitates further maintenance of the code. The layers include: user *interface*, *middle* and *driver* layer.

The main UI class is called *MainWindow*, and it is located in the UI layer, which operates all the user interface operation flows and interfaces the other GUI sub-classes. It is also the layer between the user and the middle layer of the software module. The other UI classes get input from the user, sends signals of the command through the *MainWindow* class to the middle layer. The Core of the software module is *CameraController*

class, which is in the middle layer. It interfaces the driver layer classes with higher level classes. It manages all the logical operations related with the cameras. Sub-classes are abstracted by *MainWindow* class through *CameraController* class. The class relations are depicted in Figure 4.5.

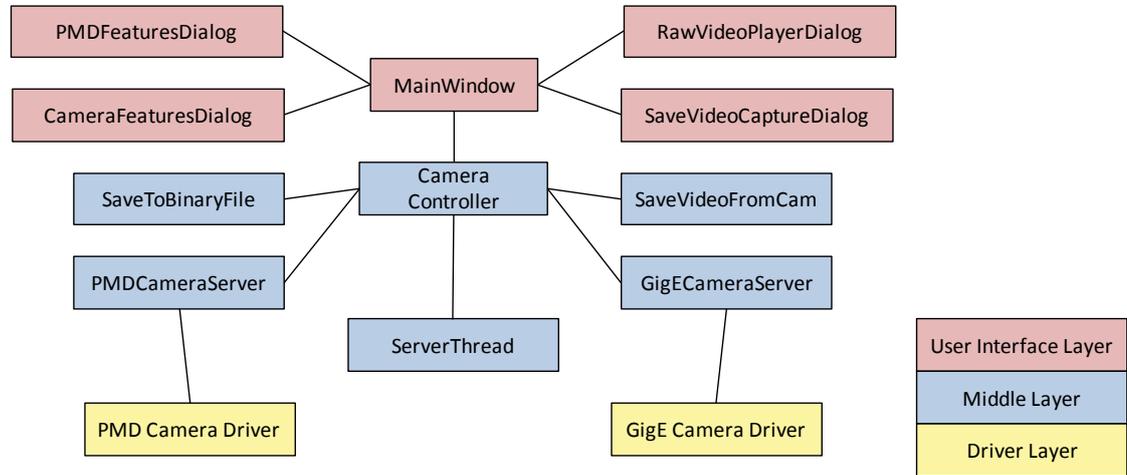


Figure 4.5 A visual representation of Class Relations of Multi-Sensor Camera Array System – Server Side

GigE Camera Server and PMD Camera Server classes are responsible for all operations, with the respected camera. Each of them can interface only one camera. There is only one PMD Camera Server object in the software.

4.2.1.3 Threading System of the Software

The *CameraController* class contains the main threading loop, and controls the main camera flow. This loop provides getting, setting and post-processing of captured data. There are some operations, which take place after getting the images from cameras, including; saving raw data of PMD camera, saving video of GigE Camera, receiving signals from the client, sending signals/data to the client and showing images.

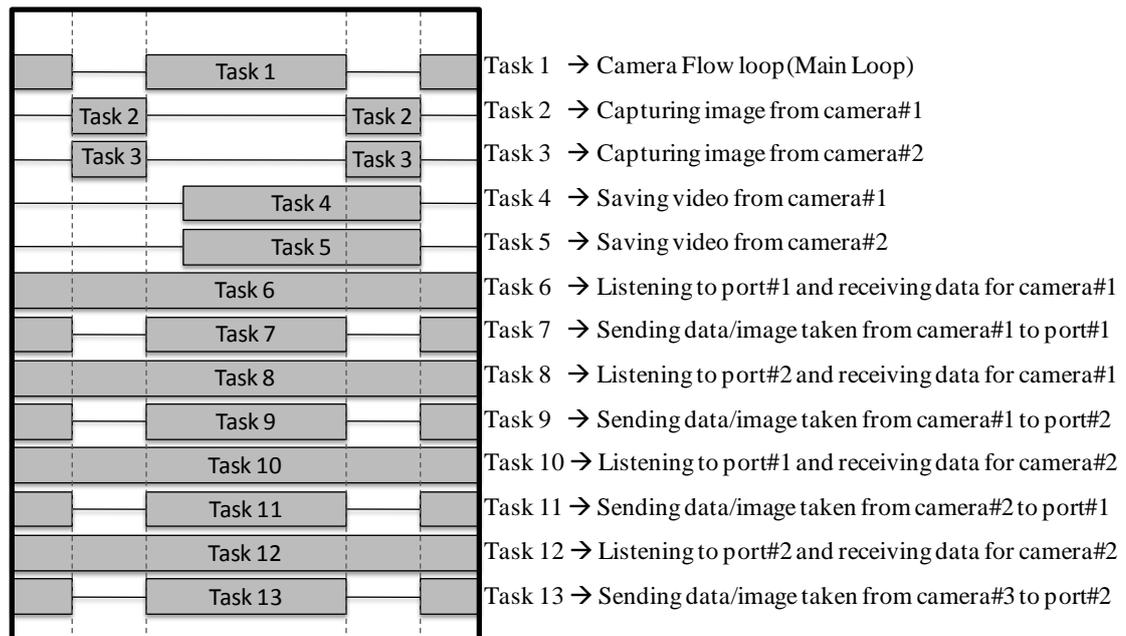


Figure 4.6 Task flow realization of threaded real-time capturing for two cameras

The main loop operates these heavy functions in the loop, and they slow down the speed. Therefore, each of these operations are set to a thread so that they will run without disturbing the main loop. This multiple threading avoids to have low frame rate. To keep the synchronization between the threads, semaphores are used.

While the camera flow is running, *SavetoBinaryFile*, *SaveVideoFromCam* and the *ServerThread* classes are allowed to work in background as other threading tasks. These threads are used in image capturing, video saving or data sending. With this property, the software is able to capture many images, save and send data at the same time. For instance, when it is needed to capture image from many cameras, the signal is sent to all at the same time with the help of threads. All of them work simultaneously, and the system waits for all of the cameras to finish their work. This works as a software trigger, which will be explained in Section 4.3. The speed of video saving and data sending are increased with this threading system as well. Using threads also provides maintaining the software easily even in very complex simultaneous capturing configurations. The task flow of threaded execution of the approach is depicted in Figure 4.6.

4.2.2 Client-Side Software Module

Client-side software module is used on the client computer to manage several server computers. It is used for getting calibration images from servers and starting to save videos on the servers. Graphical user interface of this software is depicted in Figure 8.6.

4.2.2.1 Properties of the Software

The client software interfaces up to seven server computers. First, it tests the connection with downloading sample images. If a server is connected and a camera is connected to

the server computer, the “OK” feedback is printed on the screen. If connection is failed, the user has to check the computer or camera connections. All of the server computers and cameras are checked in the same way. Another property of this software is that it provides an automation system that requests images in certain time period with a certain limit. The purpose of this automated system is to capture calibration images with only one user. First, the user sets a time period between the shots and the number of frames to be captured. Then, the user shows the chessboard from different orientations to the cameras. The server software captures calibration images, and checks them if they are suitable for calibration or not. The client receives the usable calibration images from the server application. When the client software receives certain number of good images, they are stored for calibration purpose. For capturing sample videos, the client software sends request for capturing videos from the cameras. The following variables can be sent to the servers by the client: frame per second, time and the frame size.

4.2.2.2 Design of the Software

A visual representation of class relations is given in Figure 4.7. A class is shown in different color depending on whether it belongs to drivers, UI dialogs or program flow classes. *MainWindow* class is the central class which makes the connection between logical operations and user interface operations.

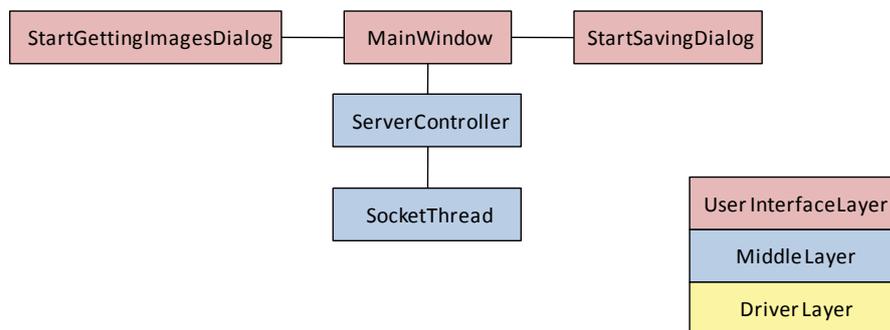


Figure 4.7 Class Relations of Multi-Sensor Camera Array System – Client Side

Each server is represented by a *servercontroller* class. The server objects has two socket threads, which are objects of the *socketthread* class. These threads are responsible for port listening, data sending and data receiving via network. The *ServerController* class is also used for evaluating calibration quality of the received images.

4.2.3 Calibration Software Module

This additional software module is implemented to apply calibration and rectification processes from the calibration images received by client software side. It is designed for calibrating up to 5 horizontally aligned sets of GigE cameras and one PMD camera lo-

cated below the central camera (Figure 4.2.b) and it provides one click calibration and rectification automated system.

Since we use odd number of cameras, and they are arranged linearly, there is always a central camera, which is the reference camera for the software. For stereo calibration of the system, the central camera forms different pairs with all other cameras in the setup. In our proposed setup, three aligned 2D cameras and one PMD ToF camera are located. Therefore, the pairs are left-center cameras, center-right cameras and center-PMD cameras. The outputs of the system are the calibration parameters of each pair-wise camera combination with the reference (central) camera.

4.2.3.1 Properties of the Software

The inputs for the software are the following: calibration image folders, number of chessboard corners on the checkerboard, rectification method and some calibration parameters. Image folders should contain the correct images provided by the client software which are tested before saving. On the other hand, names of the image files in different folders should have the same time tag, which is useful for getting the images in the same order. The number of the corners of chessboard should be entered correctly to get proper results from the software. The user interface of the software is shown in Figure 8.8.

4.2.3.2 Design of the Software

The functional property of this software is 2D/PMD joint-calibration process. To use already proposed calibration methods and functions, some pre-processing is applied to the images from the cameras to utilize this 2D/PMD joint-calibration. Images from 2D cameras are cropped to 1000x1000 and images from PMD camera are rescaled to the same resolution. To have the orientation of the right and left camera, images are rotated 90 degrees to left or to right. It does not matter which way they are rotated but if they are rotated to left, 2D camera will be the left camera. If they are rotated to the right, PMD camera will be the left camera. After these pre-processing steps, the calibration for PMD and 2D cameras can start. In the beginning of the calibration, the chessboard corners are detected to obtain known points from the real world. All of the points are stored from all of the calibration images. First, single camera calibration takes part for each camera. Camera matrices and distortion parameters are gathered from this function. Then the stereo calibration is done for both cameras where rotation matrix and translation vector are calculated. These parameters are used for the rectification algorithm proposed by Bouguet [27]. Another rectification method, which is proposed by Hartley [2] does not need any calibration parameters. However, the un-distortion process is done in advance. These rectification methods have been re-implemented in the OpenCV library. The tunable properties of these functions are available to the user as well. The results of the processing are printed to log files, where camera intrinsic and extrinsic parameters are available. Furthermore, there is a function which checks the

quality of calibration. It uses information from epipolar lines to get a subjective error evaluation from images.

Depth estimation from calibrated pairs is available in the functionality of this software as well. The used methods for depth estimation are block-matching and graph-cuts stereo matching algorithms that are provided by OpenCV. For faster solutions, block matching is better even though results are not as good as graph-cuts algorithm. Our calibration software provides users with controls to adjust the parameters of these two matching algorithms. Moreover, the software module gets camera matrices and distortion parameters as input for un-distorting images. This provides undistorted image samples for other research purposes.

4.3 Triggering and Synchronization

A multi-sensor camera array capturing system has to provide synchronized images from the connected cameras and other sensors. Thus, all of the cameras has to capture image at the same time, and triggering systems are used to signal to all of them. There are two triggering systems:

- Software triggering system
- Hardware triggering system

In this project, both of them are used for particular reasons. The PMD ToF camera has no input/output ports for hardware triggering. It only has free-run mode functionality, because of the specific nature of the capturing process. While it is capturing images continuously, an image is acquired from the camera, which is captured at that moment. It never stops capturing, and the image of that instant can be obtained by this way. Using this trick provides software synchronization between a PMD ToF camera and a 2D camera. In the software solution, two capturing functions of the cameras are given to a function and these capturing processes are run in different threads at the same time. When the images are obtained from both cameras, the function returns both results from the capturing functions. Synchronization between two different cameras is achieved by this method.

On the other side, a hardware trigger is used between the main camera and the other cameras. The idea behind this setup is to prevent losing the synchronization in capturing images between different server computers. Only one triggering server is used which is controlled by main camera. The main camera itself is triggered by a hardware signal as well, whereas all other cameras are triggered by a hardware signal produced from this camera. A machine vision timing controller provided by Gardasoft is used to trigger the camera server.

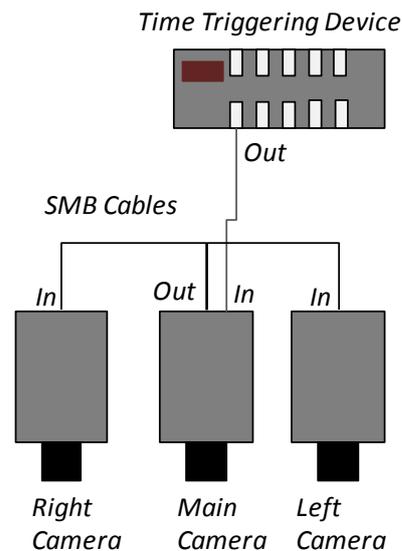


Figure 4.8 Hardware Triggering Solution

The output of the time triggering device is connected to the input of main camera, which captures images by the signal. The output socket of the main camera is connected to the input of the other cameras and a signal is sent to other devices via a ring cable. Other cameras receive the signal from the main camera and then start capturing (Figure 4.8). The triggered cameras are able to trigger other cameras as well. This type of system is called daisy chain. However, our system is slightly different from a daisy chain system, where there is no time triggering device.

The reason behind using the main camera instead of using time triggering device is to synchronize cameras, which are connected to different computers. For instance, the main camera gets the signal, and starts capturing with PMD ToF camera. At the same time, all other cameras start to capture as well. However, capturing frame rate of *Main Server* computer is lower than additional computers, because of the PMD ToF camera. During the capturing process, the time triggering device may give extra signals to the additional cameras so that the central camera may capture less number of images than the additional ones, and the frame synchronization between the servers might get lost. For that reason, using the main camera as a time triggering server gives more precise synchronization solution, which determines the frame rate of the whole system as well.

5 RESULTS

Multi-sensor camera capturing systems include many processing stages before displaying the captured content. The stages include: capture, calibration, rectification, stereo matching, depth extraction, encoding and transmission. In this thesis, the first three steps are studied to give a good overview of those issues.

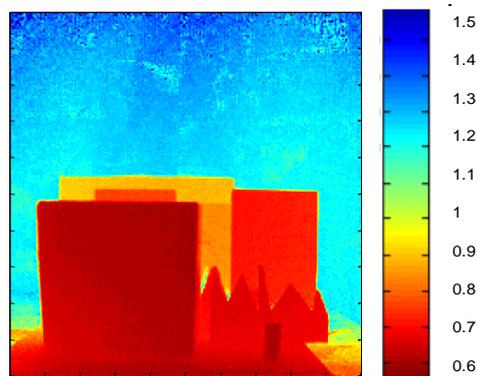


Figure 5.1 An example of color coded dense depth map with color bar

5.1 Performance evaluation of the System

Capturing videos from all cameras in the setup with a high frame rate is the first goal of the system. Using the proposed setup, the system can save video up to 15 frames per second, which can be increased by removing the PMD ToF camera from the main server. Even though the speed is not so high, the synchronization between the cameras is accurate. It is decided to create 2D/PMD camera real-life captured of different scenarios, which could be used for test data. Several life-scenes are captured on purpose. All captured data provides a calibration video set in the beginning of the captured sequence. For the calibration and rectification, around 25 chessboard images are taken by the cameras (Figure 5.3). The intensity images of the PMD ToF camera are used for the calibration process. These intensity images are different than the images from conventional cameras. In this case, joint calibration and rectification of these cameras become challenging.

The dense depth map that is taken from the PMD ToF camera is mapped in color mode. This type of mapping is used for better visualization to show the distance in the scene. A color bar which shows the distance and color relation of the dense depth map is depicted in Figure 5.1.

5.2 Applications of the System

Since the system proposed in this thesis is generic, it is able to give video and calibration samples from different scenes with different setups and with different properties. To have different comparing samples, many shooting sessions have been carried out including indoor and outdoor scenes. We provide examples of the following scenes: chess playing, billiards playing and corridor ball bouncing and bike scene (Figure 5.2). Furthermore, other samples were captured for an indoor floor ball tracking research. In this case, there was no need for depth camera. Therefore, the scene capturing is handled with only one camera.



Figure 5.2 Image samples of captured videos from different scenes. From left top to right: a)Chess playing, b)Corridor ball playing, c)Indoor billiards, d)Bikes, e)Close-up presentation, f)Floor ball tracking

The aim of this session was keeping the shape of the ball as round in the captured images because the ball is moving so fast that normal cameras are capturing it as ellipse. For that reason, the exposure time was decreased to 1500m and gain is increased to 45dB. With these settings, the images became dim but suitable for the object tracking. This session was held in the sports hall of TUT(Figure 5.2.f).

Besides these samples, there have been many sessions held indoors for getting samples as well. These samples were taken with the proposed system of this thesis which includes 3GigE and one PMD cameras. In these shooting sessions, the aim was to get samples for joint calibration and rectification of GigE and PMD cameras. The scenes referred to as reporter, wooden toys and chess playing.

5.3 Calibration Results

Apart from the technical details of the calibration, there are also practical issues that may cause error over the results. During the capturing process of calibration images, the checkerboard kept static over a table or some kind of flat area (Figure 5.3).



Figure 5.3 Captured calibration pattern images from left to right respectively: left camera image, center camera image, right camera image, PMD camera intensity image

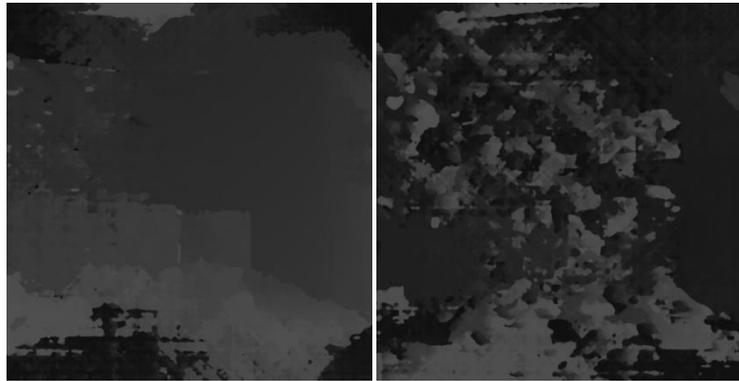
This avoids causing any synchronization issue between the camera and motion blur over the images, and corner detection over the calibration patterns is done better. To get more precise results from calibration, static checkerboards are needed. To eliminate motion blur, exposure time of the cameras can be decreased as well, because longer time means more motion in the image. Calibration of the cameras is done in three pairs in the proposed setup. These are: PMD ToF camera-central camera, PMD ToF camera-right camera, and central camera-right camera.

First of all, the single camera calibration is done for each camera. Before the calibration starts, PMD ToF intensity images are scaled up to 1000x1000 which is the resolution of the 2D cameras. This action is taken to compensate the image size difference between the 2D and PMD cameras and the calibration results are given with respect to this image size (Appendix 3).



Figure 5.4 Images from cameras from left to right respectively left, center, right and PMD ToF

Before doing any processing on the images (Figure 5.4), the disparity maps of the image pairs are taken with two algorithms to show why rectification is needed for stereo vision. The results of this process are depicted in Figure 5.5. If we compare these disparity map results with the depth map taken from the PMD camera, they are not close to each other and not satisfactory at all.



**Figure 5.5 Disparity maps before rectification process with SGBM algorithm
a) Left and center image disparity map, b) Center and right image disparity map**

Using implementation of the classical calibration approaches provided by Hartley and Bouguet has some limitations for the proposed system. These implementations are expecting the sensors to be of the same size.



Figure 5.6 Left and center images after rectification process and estimated disparity map of them with SGBM algorithm

Moreover, estimation of the location of checkerboard corners using PMD data is not accurate. This causes wide ambiguous range of obtained data for calibration and rectification. The expertise in stereo calibration among depth capturing devices and 2D cameras is not advanced. However, the proposed system is able to obtain good and accurate results.



Figure 5.7 Center and right images after rectification process and map of them with SGMB algorithm

Before the rectification of the left and center images have really bad results of stereo matching (Figure 5.6). However, after rectification, the stereo matching results are improved considerably. The rectification for center and right images improves the stereo matching results as well (Figure 5.7).

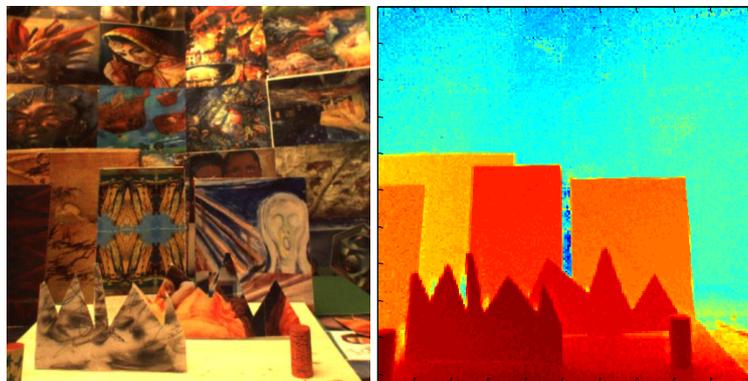


Figure 5.8 Center and PMD-depth images after rectification process

The rectified image results show that calibration and rectification processes worked accurately so the depth map of the stereo pairs is quite close to the depth map from PMD camera. A part from these stereo images, central and PMD camera joint calibration and image rectification is done as well as mentioned before (Section 5.1). With this process, the central camera obtains a depth map without any stereo correspondence process (Figure 5.8).

6 CONCLUSIONS

The thesis started with an overview of the pinhole camera capture model. Then, stereo camera systems were reviewed along with capture artifacts produced by the optical system, sensor device and mechanical misalignments. The calibration process was reviewed resulting in estimating the actual focal length, optical center and the distortion parameters. Extrinsic camera parameters, i.e. rotation matrix and translation vector, were explained as well. The thesis presented an overview about how a basic stereo camera system works along with various aspects of multi-sensor camera array capturing systems. Possible effects of stereo camera misalignments on depth estimation quality of stereo matching methods were discussed. Optimal configuration for optimal depth estimation was commented. Compensation of mechanical misalignments based on improved rectification, which has good metric evaluation of camera misalignments was provided. Moreover, an improved calibration process was introduced that increases the accuracy of the camera rectification applications.

A multi-sensor camera array capture system was implemented to acquire images for different purposes. The hardware and software of the system was designed flexibly so that the user can plug/unplug cameras and computers to change the setup. Besides, a PMD ToF camera, which delivers real-time depth measurements of the scene, is supported by the system. Multiple computers are used to interface many cameras and a network is set for communicating with the computers. For our research purposes a metal rig is constructed, which can accommodate three video cameras, and one PMD ToF camera is mounted under them. This rig is mounted on a tripod so that it is mobile but stable. Since there are many computers, software synchronization is not efficient for the system. Therefore, hardware triggering is used to maintain the synchronization between the cameras. However, PMD ToF do not have any input for hardware triggering, therefore a software trigger is used along with the hardware triggered video cameras.

A server/client software module is implemented. The server-side implementation includes camera interfacing, real-time data capturing, data saving and sending and software synchronization between video and PMD ToF cameras. Client side is able to send commands for capturing image/video to server computers and receive images from them. With this server/client software, calibration images can be obtained via network from multiple cameras.

Calibration of the proposed setup with three video cameras and a PMD ToF camera is implemented in another software module. The artifacts which occur during the capture of calibration images are mentioned, and how to minimize these artifacts for better calibration is discussed. Furthermore, a robust stereo-pair to stereo-rectified-pair applica-

tion is implemented for the proposed setup. The results of three 2D and a PMD ToF camera joint calibration and rectification are summarized.

Consequently, a multi-sensor camera array system is developed and studied from capturing to displaying images to have a generic solution for various types of 3D data needs. Several shooting sessions have been held with the system and acquired data are available for test purposes. The proposed camera array setup have been tested for obtaining sample images for calibration and rectification processes, and the results show an accurate performance of all modules. These results indicate that the system is capable of running multi-sensor camera arrays without problems and can give accurate results in terms of calibration and rectification.

The present work constitutes a first stage of a full end-to-end 3D video system, namely the multi-sensor camera array systems. This end-to-end system starts with calibration of the cameras, goes on with capturing images and rectifying them for further post processing. Such post processing might include depth extraction or data fusion to prepare 3D videos for showing on different types of displays. Some displays, e.g. auto-stereoscopic displays with 9-8 views require depth map along with the related image, some others require stereo images, e.g. polarized displays. The proposed system can provide specific formats for specific screens.

The proposed generic approach is suitable for building and using a multi-sensor camera arrays. Synchronization, calibration and rectification abilities of the system provide ready-made data. The system is flexible that the user is able to acquire several types of data with the proposed approach. The range of images varies from basic stereo (two cameras) to surround camera system dome like up to 14 cameras. The system can be used for different application, i.e. interpolating intermediate images from given stereo images plus depth map, using a depth camera for face tracking, structure from motion, 3D scene reconstruction with multiple cameras with depth camera, face and body detection and tracking, super-resolution depth map calculation and many others.

7 REFERENCES

- [1] Hartley, R. I. "Self-calibration from multiple views with a rotating camera" In *Proceedings of the third European conference on Computer vision*, vol. 1, 1994
- [2] Tsai, R. Y. "An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision", In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 364-374, Miami Beach, FL, 1986
- [3] Tsai, R. Y. "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*, No. RA-3(4), pp. 323-344, 1987
- [4] Luong, Q.T. and Faugeras, O. D. "Self-Calibration of a Moving Camera from Point Correspondences and Fundamental Matrices" *Int. J. Computer Vision* 22,vol. 3, pp. 261-289, 1997
- [5] Caprile, B. and Torre, V. "Using vanishing points for camera calibration" *International Journal of Computer Vision*, vol. 4, no. 2, pp. 127-139, 1990
- [6] Schuon, S., Theobalt, C., Davis, J. and Thrun, S. "High-quality scanning using time-of-flight depth superresolution" *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1–7, 2008
- [7] Guan, C., Hassebrook, L. G., and Lau, D. L. "Composite structured light pattern for three-dimensional video" *Opt. Express* 11, 406–417, 2003
- [8] Smolic, A., Müller, K., Dix, K., Merkle, P., Kauff, P., and Wiegand, T., "Intermediate View Interpolation based on Multi-View Video plus Depth for Advanced 3D Video Systems" In *Proc. IEEE International Conference on Image Processing (ICIP'08)*, pp. 2448-2451, San Diego, CA, USA, 2008
- [9] Chen, S. E. and Williams, L. "View interpolation for image synthesis" In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques (SIGGRAPH '93)*,1993
- [10] Park, Joon H. and Park, H. "Fast view interpolation of stereo images using image gradient and disparity triangulation." *ICIP*, 2003
- [11] Lim,H., Kim,S., Lee, Y. and Park, H. "A Simultaneous View Interpolation and Multiplexing Method using Stereo Image Pairs for Lenticular Display." *ICIP*, 2007
- [12] Moons, T., Van Gool, L. and Vergauwen, M. "3D Reconstruction from Multiple Images" *Foundations and Trends® in Computer Graphics and Vision*, vol. 4, no 4, pp. 287-404, 2009
- [13] Faugeras, O. D."Three-Dimensional Computer Vision: A Geometric Viewpoint" *MIT Press*, Cambridge, MA, USA, 1993

- [14] Liebowitz, D. and Zisserman, A. "Metric Rectification for Perspective Images of Planes" In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '98)*, 1998
- [15] Stein, G. P. "Accurate internal camera calibration using rotation, with analysis of sources of error" In *Proceedings of the Fifth International Conference on Computer Vision (ICCV '95)*, pp. 230-236, IEEE Computer Society, Washington, DC, USA, Cambridge, Massachusetts, June 1995
- [16] Abdel-Aziz, Y.I. and Karara, H.M. "Direct Linear Transformation from comparator coordinates into object space coordinates in close-range photogrammetry" In *Proceedings of the ASP Symposium on Close-Range Photogrammetry*, pp. 1-18, 1971
- [17] Heikkila, J. and Silven, O. "A Four-step Camera Calibration Procedure with Implicit Image Correction" In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition*, 1997
- [18] Zhang Z. "A Flexible New Technique for Camera Calibration" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330-1334, 2000
- [19] Brown, D. C. "Close-range camera calibration" *Photogrammetric Engineering* 37, pp. 855–866, 1971
- [20] Hartley, R. I. "In defense of the 8-point algorithm" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 6, pp. 580–593, 1997
- [21] Fusiello, A., Trucco, E. and Verri, A. "A compact algorithm for rectification of stereo pairs" *Machine Vision and Applications*, vol. 12, issue 1, pp. 16-22, July 2000
- [22] Hartley, R. I. and Zisserman, A. "Multiple View Geometry in Computer Vision", *Cambridge University Press*, Second Edition 2004
- [23] Oram, D., "Rectification for any epipolar geometry" *BMVC01*, session 7, Geometry & Structure, 2001
- [24] Roy, S., Meunier, J. and Cox, I. "Cylindrical rectification to minimize epipolar distortion" In *CVPR*, pp. 393–399. 1997
- [25] Pollefeys, M., Koch, R., Van Gool, L. "A simple and efficient rectification method for general motion" In *Proc. International Conference on Computer Vision*, pp. 496-501, 1999
- [26] Papadimitriou, D. V. and Dennis, T. J. "Epipolar line estimation and rectification for stereo image pairs" *IEEE Transactions on Image Processing*, vol. 5, no. 4, pp. 672–676, 1996
- [27] Bouguet, Jean-Yves "Visual methods for three-dimensional modeling" *Dissertation (Ph.D.)*, California Institute of Technology, 1999
- [28] Zhang, Z. "Determining the epipolar geometry and its uncertainty: a review." *International Journal of Computer Vision*, vol. 27, issue 2, pp. 161–195, 1998

- [29] Luo, A. and Burkhardt, H. "An intensity-based cooperative bidirectional stereo matching with simultaneous detection of discontinuities and occlusions" *In Proceedings of International Journal of Computer Vision*. pp 171-188, 1995
- [30] Tao, H., Sawhney, H. and Kumar, R. "A global matching framework for stereo computation" *In International Conference on Computer Vision*, vol. 1, 2001
- [31] Zhang, Y. and Kambhamettu, C. "Stereo matching with segmentation-based cooperation." *ECCV*, 2002
- [32] Bleyer, M. and Gelautz, M. "A layered stereo algorithm using image segmentation and global visibility constraints." *ICIP*, 2004
- [33] Haralick, R.M. and Shapiro, L.G. "Image matching, computer and robot vision" *1st ed.*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1992
- [34] Scharstein, D. and Szeliski, R. "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms." *International Journal of Computer Vision*, vol. 47, issue 1/2/3, pp. 7-42, 2002
- [35] Yang, R., Pollefeys, M., and Welch, G. "Dealing with Textureless Regions and Specular Highlight: A Progressive Space Carving Scheme Using a Novel Photo-consistency Measure" *In Proc. of the International Conference on Computer Vision*, pp. 576-584, 2003
- [36] Brockers, R. "Cooperative Stereo Matching with Color-Based Adaptive Local Support" *In Proceedings of CAIP*, 2009
- [37] Yoon, K. J. and Kweon, I. S. "Adaptive support-weight approach for correspondence search." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006
- [38] H. Hirschmuller, "Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information" *In Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2, pp. 807-814, 2005
- [39] Birchfield, S., Tomasi, C., "Multiway Cut for Stereo and Motion with Slanted Surfaces" *In Proc. Int'l Conf. Computer Vision*, pp. 489-495, 1999
- [40] K. Konolige, "Small vision system: Hardware and implementation" *In Proceedings of the International Symposium on Robotics Research*, pp. 111-116, Hayama, Japan, 1997
- [41] Li, Z. and Baker, H. and Kurillo, G. and Bajcsy, R., "Projective Epipolar Rectification for a Linear Multi-imager Array" *3DPVT10*, 2010
- [42] Kang, Y., Lee, C., Ho, Y., "An Efficient Rectification Algorithm for Multi-View Images in Parallel Camera Array" *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*, pp.61-64, 2008
- [43] Baker, P., Fermuller, C., Aloimonos, Y. and Pless, R. "A spherical eye from multiple cameras (makes better models of the world)" *Computer Vision and Pattern Recognition(CVPR)*, 2001
- [44] Chang, W. and Chen, C., "Pose Estimation for Multiple Camera Systems" *In Proceedings of the Pattern Recognition*, 2004

- [45] Frahm, J., Köser, K., and Koch, R. "Pose Estimation for Multi-camera Systems" *In Proceedings of DAGM-Symposium*, pp. 286-293, 2004
- [46] Wilburn, B. "High Performance Imaging Using Arrays Of Inexpensive Cameras" *PhD Thesis*, 2004
- [47] Joshi Neel S. "Color Calibration for Arrays of Inexpensive Image Sensors" *Master's with Distinction in Research Report*, 2004
- [48] Ringbeck, T., Hagebecker, B., "A 3D time of flight camera for object detection." *In Object 3D Measurement Techniques*, ETH Zürich, 2007
- [49] Koch, R., Schiller, I., Bartczak, B., Kellner, F. and Köser K. "MixIn3D: 3D Mixed Reality with ToF-Camera" *Dynamic 3D Imaging*, Springer, vol 5742, pp. 126-141, Berlin / Heidelberg, 2009
- [50] Fuchs, S. and Hirzinger, G. "Extrinsic and depth calibration of ToF-cameras" *Computer Vision and Pattern Recognition(CVPR)*, 2008
- [51] Fuchs S. and May, S. "Calibration and registration for precise surface reconstruction with TOF cameras." *In Proceedings of the ADGM Dyn3D Workshop*, Heidelberg, Germany, 2007.
- [52] Lindner, M. and Kolb, A. "Calibration of the intensity-related distance error of the PMD TOF-camera" *In Intelligent Robots and Computer Vision XXV: Algorithms, Techniques, and Active Vision*, 2007.
- [53] Igwe, P.C. and Knopf, G.K. "3D Object Reconstruction Using Geometric Computing" *Geometric Modeling and Imaging--New Trends*, pp.9-14, 16-18 Aug. 1993
- [54] Liu, J., Cao, L., Li, Z. and Tang, X., "Plane-Based Optimization for 3D Object Reconstruction from Single Line Drawings" *IEEE Trans. Pattern Anal. Mach. Intell.* vol. 30, issue 2, pp. 315-327, 2008
- [55] Esteban, Hernandez C. and Schmitt, F., "Multi-stereo 3D object reconstruction" *3D Data Processing Visualization and Transmission(3DPVT)*, pp. 159- 166, 2002
- [56] Lu, F., Ji, X., Dai, Q. and Er, G. "Multi-View Stereo Reconstruction with High Dynamic Range Texture" *ACCV*, pp. 412-425, Springer Berlin – Heidelberg, 2010
- [57] Dellaert, F., Seitz, S.M., Thorpe, C.E. and Thrun, S. "Structure from motion without correspondence" *Computer Vision and Pattern Recognition*, 2000 Proceedings. IEEE Conference on , vol.2, no., pp.557-564 vol.2. 2000
- [58] Zhang, J., Boutin, M. and Aliaga, D.G. "Robust Bundle Adjustment for Structure from Motion" *Image Processing, IEEE International Conference*, pp. 2185 – 2188, Atlanta, GA, 2006
- [59] Zhang, Q. and Pless, R. "Fusing video and sparse depth data in structure from motion" *ICIP*, 2004
- [60] Oliensis, J. "A new structure-from-motion ambiguity" *Computer Vision and Pattern Recognition. IEEE Computer Society Conference*, 1999
- [61] Bay, H., Ess A., Tuytelaars, T., Van Gool, L., "SURF: Speeded Up Robust Features" *Computer Vision and Image Understanding (CVIU)*, vol. 110, no. 3, pp. 346-359, 2008

- [62] Szeliski, R., "Image alignment and stitching" *Microsoft Research, A tutorial. Technical Report*, 2006
- [63] Harris, C. and Stephens M., "A combined corner and edge detection" *In proceedings of The Fourth Alvey Vision Conference*, pp. 147–151, 1988
- [64] Rosten, E. and Drummond, T. "Machine learning for high-speed corner detection" *ECCV*, 2006
- [65] Matas, J., Chum, O., Urba, M., and Pajdla, T., "Robust wide baseline stereo from maximally stable extremal regions" *In Proc. of British Machine Vision Conference*, pp. 384-396, 2002
- [66] Shi, J., and Tomasi, C., "Good Features to Track" *9th IEEE Conference on Computer Vision and Pattern Recognition*, 1994
- [67] PMDvision CamCube 2.0 Datasheet, No. 20090601, Siegen, Germany, PMDTechnologies, 2009

8 APPENDIX 1: SOFTWARE DOCUMENTATION

8.1 Server Side Software Module

On the left-upper corner side of the GUI of Server (Figure 9.1), the list of available connected cameras is given. If there is no available camera, the refresh button below the list will help for finding connected cameras in the network. Due to the restrictions of PMD SDK library of PMD ToF Cam Cube 2.0 camera, active connection manager is not working for that device. The check box above the camera connection list indicates, whether the user wants to force the start of PMD camera or not. The feedback of the camera status is printed in the text area.

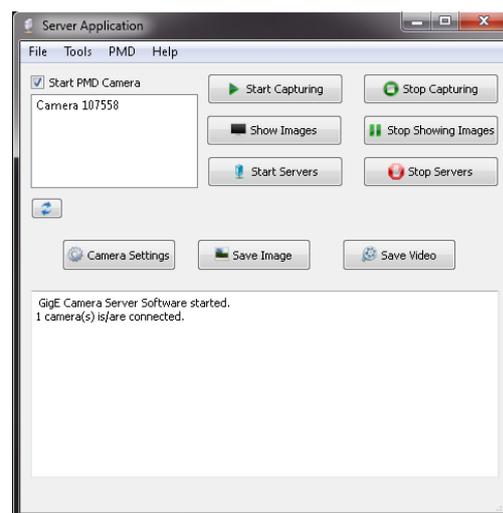


Figure 8.1 GUI of Server Side Software

Start Capturing button starts interfacing and streaming the cameras.

Stop Capturing button stops the streaming.

Show Images button shows the streaming images on the screen. They are not showed by default because of the display restrictions. When the PMD camera and a GigE camera are connected, an HD resolution video and four 204*204 resolution video are shown on the screen.

Stop Showing Images button stops showing the images so that user can close them and decrease the usage of the processor or graphics card.

Start Servers button streams images via network. When you click that button, it starts to listen to different ports for each camera.

Stop Servers button stops to listening the client ports.

PMD drop menu contains settings dialog of PMD parameters (Figure 9.2). This helps setting the following variables of the PMD camera: Integration time and Modulation Frequency.

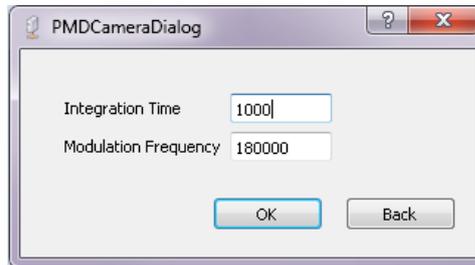


Figure 8.2 GUI of PMD Camera Settings Dialog

Camera Settings button opens a dialog that contains the properties of the 2D cameras (Figure 9.3). Some capturing properties can be changed here and also some network properties. Optimized package size and byte string per frame are selected for maximum speed and quality performance by default.

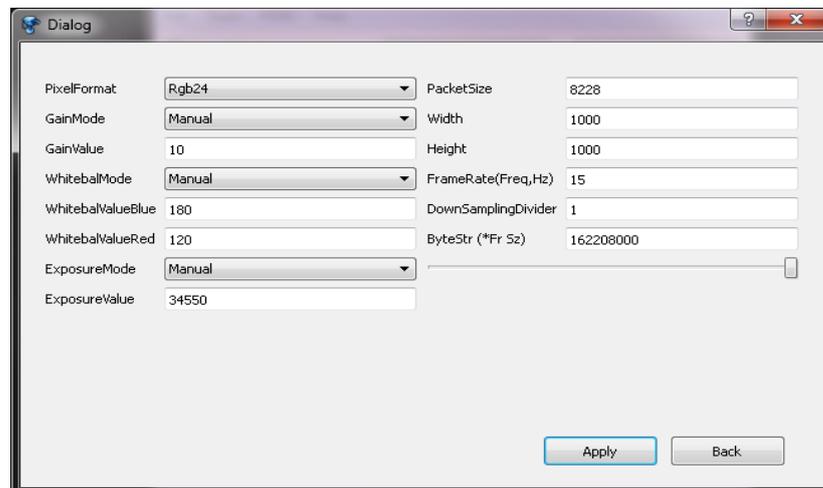


Figure 8.3 GUI of PMD Camera Settings Dialog

Save Image button provides Saving data properties dialog (Figure 9.4). Data is saved immediately after OK button is pressed. For each camera, captured image will be saved and put in separate folder with time stamp name.

Save Video button opens a dialog to set the properties of the video saver.

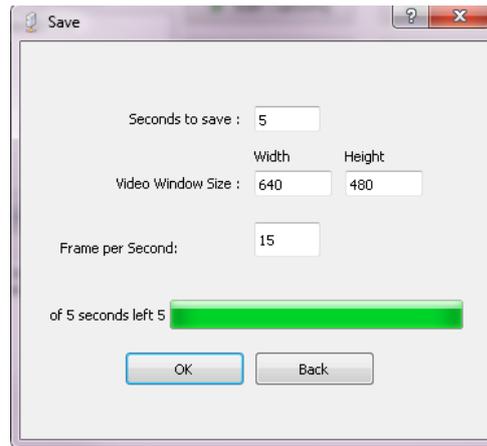


Figure 8.4 GUI of Video/Image Save Dialog

Tools drop menu contains a video player that visualizes the video of saved PMD camera (Figure 9.5). The player handles only saved raw binary floating-point data with the designed header.

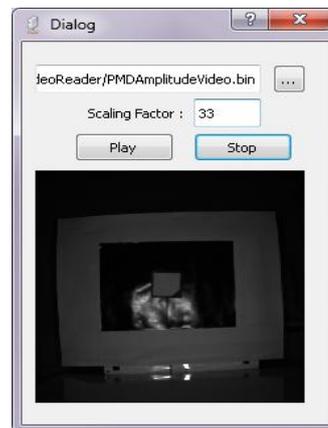


Figure 8.5 GUI of PMD Video Player

8.1.1 Code Structure

CameraServer.sln -solution file
main.cpp -main() program file

GUI files

mainwindow.cpp/h/ui	-controls all the functions in the main window.
camerafeaturesdialog.cpp/h/ui	-the dialog to get the features from user.
pmdcameradiolog.cpp/h/ui	-the pmd camera settings dialog to get the settings from user.
RawVideoPlayerDialog.cpp/h/ui	-for reading the data save from “savetobinaryfile” class
savevideocapturedialog.cpp/h/ui	-the dialog to get information about saving video from user.

Core files

cameracontroller.cpp/h	-controls the all cameras and the operations over the images
cameraserver.cpp/h	-the server class to listen ports. Each camera has one server.
camera.cpp/h	-camera class that interfaces the GigE cameras
serverthread.cpp/h	-the thread class for listening ports and managing the send/receive operations.
pmdcameraserver.cpp/h	-the camera server for PMD camera.
pmd_camera.cpp/h	-the camera class that works for PMD camera.
savetobinaryfile.cpp/h	-saves the data that has taken from PMD camera.
saveVideoFromCam.cpp/h	-saves the video taken from GigE cameras.
definitions.h	-includes definitions of parameters and structs.

8.2 Client Side Software Module

The list on the left side of GUI of *Client Software* is the list of servers to be connected(Figure 9.6). IP numbers of the servers should be entered to connect. Checkboxes on the left side of the server numbers show whether they are in the system or not. If the checkbox of a server is not set, no connection will be provided with that server.

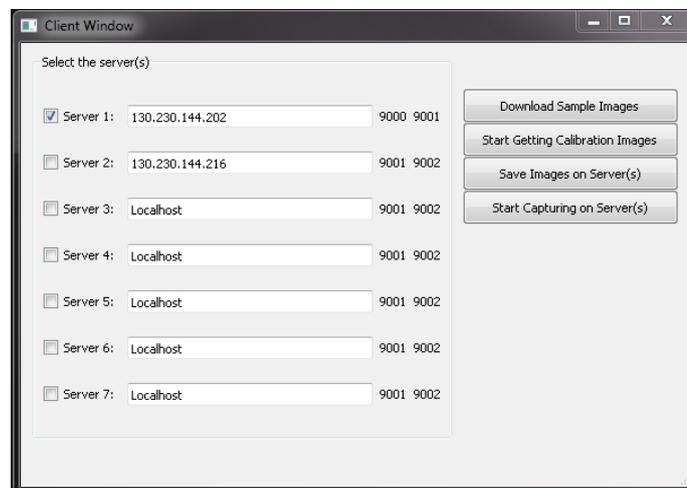


Figure 8.6 GUI of Client Side Software

Download Sample Images button sends signal to the servers to send images of that moment. These images are not used for any purpose, but to check the network and the s work properly.

Start Getting Calibration Images sends signals to the servers to start providing the automated calibration images.



Figure 8.7 GUI of Start Capturing on Servers Dialog

Save Images on Servers sends signals to the servers to save images on the server side.

Start Capturing on Servers opens a dialog, which provides submitting information to save video on server computers (Figure 9.7).

8.2.1 Code Structure

QTCPCClient.sln -solution file
main.cpp -main() program file

GUI files

mainwindow.cpp/h/ui -the main window class,GUI, controls the data send/receive operations
startgettingimagesdialog.cpp/h/ui -automation for getting the calibration images from servers.
startsavingdialog.cpp/h/ui -the dialog to start saving videos on servers.

Core files

servercontroller.cpp/h - server controller class. It controls each server connection.
socketthread.cpp/h -the thread class for listening ports and sending messages.
videocalibrator.cpp/h -to check the chessboard corners.

8.3 Calibration Software Module

This module gets the path of the folders that contains the images of each camera as the primary input(Figure 9.8). On the left side of the window, there are checkboxes that user can select cameras that are going to be calibrated with respect to the reference camera. In those calibration images, to find the chessboard corners, numbers of horizontal and vertical chessboard corners should be entered.

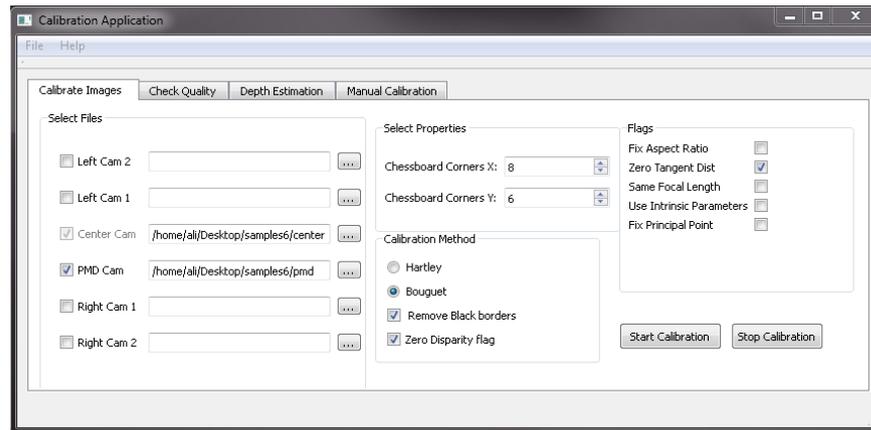


Figure 8.8 GUI of System Calibration Software

The stereo calibration parameters and rectification method is available for selection by user (Figure 9.9). *Start Calibration* button starts the process and gives results as images and log files. Besides these, there is an that checks the quality of the calibration process also. It gets the calibrated and rectified calibration images controls the chessboard points in the images. Scan-lined output images and epipolar matching errors are printed to the screen.

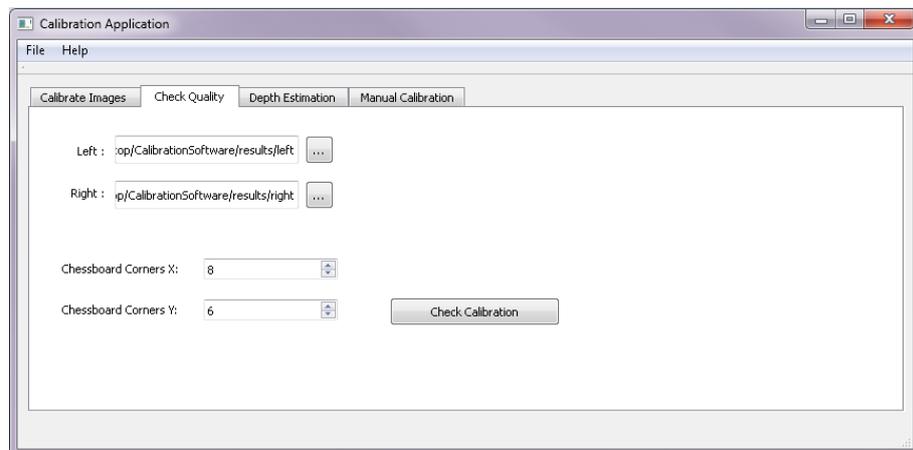


Figure 8.9 GUI of Check Quality Functionality of Calibration Software

A block matching algorithm of depth estimation is used to see how efficient the results are (Figure 9.10). For block matching algorithms, there are multiple inputs that should be filled. This part of the UI gets left and right images and the number of the corners in the checkerboard.

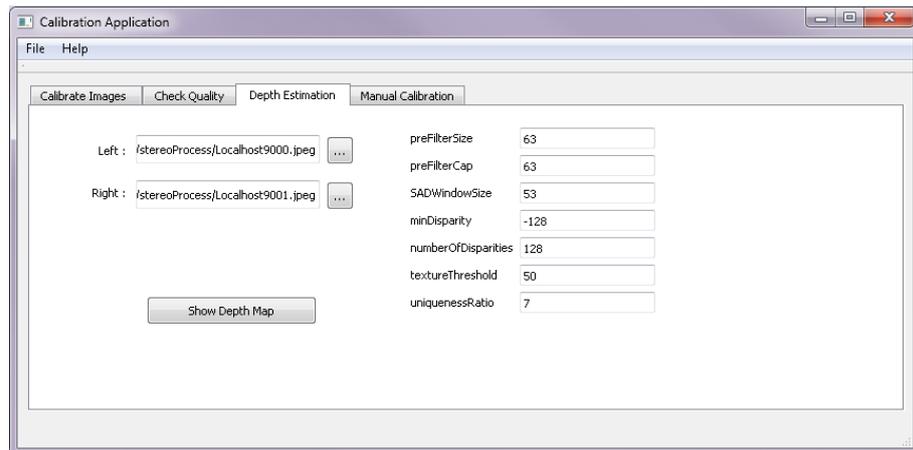


Figure 8.10 GUI of Depth Estimation Functionality of Calibration Software

Un-distortion process of the images is handled also in the software (Figure 9.11). Camera matrix, distortion parameters and the folder of the images are taken as input. Un-distorted images are saved to the same folder.

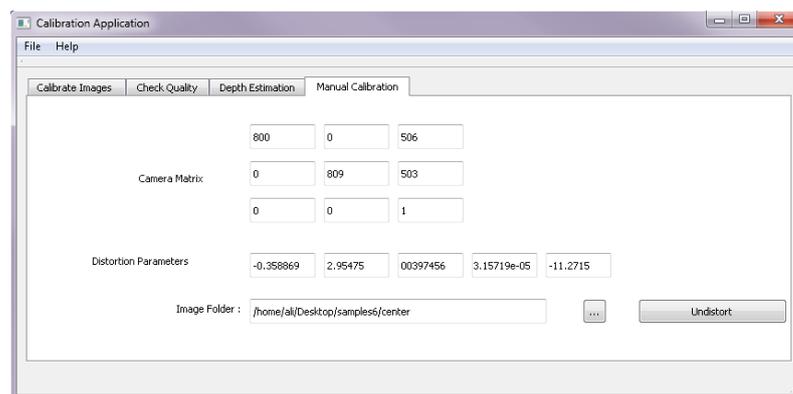


Figure 8.11 GUI of Lens Un-distortion Functionality of Calibration Software

8.3.1 Code Structure

main.cpp	-main() program file
mainwindow.cpp\h\ui	-controls the UI and has the functions for depth estimation
videocalibrator.cpp\h	-has all the functions to make calibration
checkcalibrationquality.cpp\h	-the class to check the calibration quality.
CalibrationSoftware.pro	-QtCreator pro file

9 APPENDIX 2: TECHNICAL MANUAL

The libraries for the camera application are developed in C/C++ programming languages. Thus, QT/C++ is selected for development environment, which includes different libraries such as GUI, Network, Core and etc. It provides an easy way of creating user interfaces, handling multiple simultaneous processing and creating communication over network. The cross-platform ability of QT allows developing platform independent applications. Furthermore, it supports OpenGL for possible displaying of 3D images from the captured data. Image processing operations are handled by OpenCV library version 2.1. It has the largest image processing library for image transformations, calibration functions and image data structures. Since the library for the PMD ToF camera has no UNIX support, the MS Windows operating system is selected and MS Visual Studio 2008 is used for compilation environment. However, a QT Visual Studio add-in⁹ should be installed to Visual Studio to run QT on it.

9.1 Camera Installation and Compilation

For the hardware setup, GigE Ethernet cards should be plugged in to the computers and libraries should be installed to use *Prosilica GE1900C* cameras. In this project Intel PRO/1000MT Dual Port Server Adapter¹⁰ is used as GigE Ethernet cards. For development of GigE camera from *Allied Vision Technologies*, a driver and its *Programming Interface* are available which is called *PvAPI*. Download¹¹ and install¹² this interface that supports all *GigE Vision* cameras from *Allied Vision Technologies*. However, the settings of the GigE network cards should be changed to interface the cameras as explained in the manual¹³. After this, to use PMD[tech] Cam Cube 2.0 camera, *PMDSDK2* library and plug-ins should be installed which comes in a CD with the camera package. The libraries and frameworks are installed before starting the implementation phase. Then, start MS Visual Studio 2008 and open the solution file of the selected project.

⁹ Available online <http://qt.nokia.com/downloads/visual-studio-add-in> , retrieved on 25.03.2011

¹⁰ Available online <http://www.intel.com/products/server/adapters/pro1000mt-dualport/pro1000mt-dualport-overview.htm> , retrieved on 25.03.2011

¹¹ Available online <http://www.alliedvisiontec.com/us/products/software/windows/avt-pvapi-sdk.html> , retrieved on 25.03.2011

¹² Available online http://www.alliedvisiontec.com/fileadmin/content/PDF/Software/Prosilica_software/Prosilica_software_s_tuff/AVT_PvAPI_Manual_V1.24.pdf , retrieved on 25.03.2011

¹³ Available online http://www.1stvision.com/cameras/AVT/dataman/GE1900_User_Manual.pdf , retrieved on 13.03.2011

The steps for making MS Visual Studio2008 ready for implementation are explained below.

Add in *ProjectProperties* following information :

- C/C++
 - General
 - Additional Include Folders
 - *OpenCV* include folder
 - *GigESDK* include folder
 - *PMDSKD2* include folder
- Linker
 - General
 - Additional Library Directories
 - *OpenCV* library folder
 - *GigESDK* library folder
 - *PMDSKD2* library folder
 - Input
 - Additional Dependencies
 - "qtmain.lib"
 - "QtCore4.lib"
 - "QtGui4.lib"
 - "QtNetwork4.lib"
 - "PvAPI.lib"
 - "cv210.lib"
 - "cvaux210.lib"
 - "cxcore210.lib"
 - "cxts210.lib"
 - "highgui210.lib"
 - "opencv_ffmpeg210.lib"
 - "pmdaccess2.lib"

In system Environment Variables, add these to the "PATH" variable with ";" separation;

- [*OpenCV* bin directory]
- [*QT* bin directory]
- [*GigESDK* bin directory]
- [*PMDSKD2* bin directory]

The PMD plugin files should be with the software code files.

- camcube.W32.pap
- camcubeproc.W32.ppp
- pmdfile.W32.pcp

After setting these properties, the project is ready for building in *Release* or *Debug* Mode.

9.2 Data Structures and Protocols

OpenCV matrix representation is used for storing our captured data (*cv::Mat*). OpenCV image formats are used for visualization purposes as well. Captured images are saved either as BMP or PNG and videos from *GigE* Cameras are saved into *AVI* container with *YUV420 Video Codec*.

PMD camera outputs 4 different type data:

1. Intensity Reflection,
2. Amplitude Confidence,
3. Depth Vertices
4. Range

That information could be not classified as image and therefore it is saved it for further purposes as binary raw floating-point data. This binary file has 16 bytes header to provide additional information about captured settings. The header includes:

1. Integration time
2. Modulation frequency
3. Width
4. Height
5. Number of Saved Frames

Header Information									
Integration Time - 4 bytes [floating point]		Modulation Frequency - 4 bytes [floating point]		Width - 4 bytes [floating point]		Height - 4 bytes [floating point]		#Frames - 4 bytes [floating point]	
PMD Data – 4 bytes per pixels [floating point]									

Figure 9.1 Data Format of Raw Binary PMD data file for intensity, amplitude and range

Header Information									
Integration Time - 4 bytes [floating point]		Modulation Frequency - 4 bytes [floating point]		Width - 4 bytes [floating point]		Height - 4 bytes [floating point]		#Frames - 4 bytes [floating point]	
PMD Data – 12 bytes per pixels [floating point]									
x	y	z	x	y	z	x			

Figure 9.2 Data Format of Raw Binary PMD data file for 3D depth vertices

The connection between client and servers is done via the HTTP protocol and socket programming using *QTNetwork* library.

The information that client sends contains;

- Option for start saving or send a calibration image
- Frame rate for video saving
- Image width and height for video saving
- Duration for video saving
- Send messages to servers
- Size of the data that includes the entire image.

- Image width, height, width step (*iplImage* property), depth and the colour channel.

9.3 Used Hardware

The proposed setup is explained in Section 4.1. The list of the hardware and mechanical stuff that is used in the proposed setup is depicted in Table 10.1

	Device Type	Number of Items
1	PMD Camera – PMDTek, CamCube 2.0	1
2	Prosilica GigE 1900C Cameras	3
3	Camera Tripod	1
4	Camera Mounting Plates	1
5	Dual Core PCs, NVidia GPU, Windows 7	2
6	GigE Dual Ethernet Cards – Intel 1000/PRO ET	2
7	Hardware Time Triggers	1
10	Ethernet Cables	6
11	USB Cables	1
12	SMB Cables(5m)	4

Table 9.1 List of used hardware in this research

10 APPENDIX C: CALIBRATION RESULTS

The calibration of the proposed setup (Figure 4.2.b) is accomplished in the thesis. The calibration results are used by the rectification process. These calibration results are given in the following tables.

<i>Right Camera Matrix</i>			<i>Central Camera Matrix</i>		
810.48345	0	520.67698	813.82546	0	514.68715
0	815.64131	478.66152	0	818.75448	504.19891
0	0	1	0	0	1

<i>Left Camera Matrix</i>			<i>PMD ToF Camera Matrix</i>		
814.92109	0	501.78569	1405.71064	0	509.43885
0	820.08656	496.18692	0	1413.80755	510.98379
0	0	1	0	0	1

Table 10.1 Camera matrices (Section 2.1)

<i>Camera</i>	k_1	k_2	k_3	p_1	p_2
<i>Right</i>	-0.1950	0.2146	-0.0024	0.0016	0
<i>Center</i>	-0.1917	0.2562	0.00064	0.00115	0
<i>Left</i>	-0.16582	0.16317	-0.00296	0.00251	0
<i>PMD ToF</i>	-0.41934	0.16742	0.00083	-0.00237	0

Table 10.2 Distortion parameters for each camera (Section 2.2)

<i>Rotation matrix of the right camera</i>			<i>Translation vector of the right camera</i>		
0.9997	-0.0067	-0.0018	5.1759	-0.0218	-0.0998
0.0067	0.9999	-0.0002			
0.0018	0.0002	0.9998			

Table 10.3 Central and right stereo camera calibration results (Section 2.3)

Rotation matrix of the central camera

0.9998	-0.0121	-0.0147
-0.0121	0.9999	0.0031
0.0148	-0.0029	0.9999

Translation vector of the right camera

5.3347681	-0.0423	0.0261
-----------	---------	--------

Table 10.4 Left and central stereo camera calibration results (Section 2.3)*Rotation matrix of the ToF camera*

0.9999	-0.0117	-0.0087
0.0117	0.9999	-0.0053
-0.0088	0.0052	0.9999

Translation vector of the ToF camera

0.0961	-6.1975	-1.6892
--------	---------	---------

Table 10.5 Central and PMD ToF stereo camera calibration results (Section 2.3)

The rectifications applied to the images with respect to these parameters are depicted in the Section 5.3.