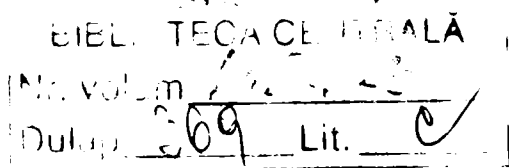


*Eng. LIVIU TOMA Jr.*

CONTRIBUTIONS ON STEREO VISION 3D  
ACCURATE MEASUREMENTS

**PhD Thesis**

SCIENTIFIC COORDINATOR:  
*Prof. Dr. Eng. ALIMPIE IGNEA*



TIMIȘOARA

2005

# Acknowledgements

I would like to thank first my scientific coordinator Prof. Dr. Alimpie Ignea, who helped me a lot during all the development period of my PhD thesis. There were some critical moments in my work and with his help I succeeded to go further and finally, to finish my PhD thesis.

I also would like to thank Prof. Dr. Werner Neddermeyer, who was my supervisor at the University of Applied Sciences of Gelsenkirchen. Here, I made most of my research work and I realized all the practical parts of my PhD thesis. Many thanks, to Prof. Dr. Wolfgang Winkler, for his support that I received for the image processing part. I would like to thank Prof. Dr. Michael Schnell for helping me to realize a practical application that used my stereo sensor.

I also would like to thank all my colleagues, from the Informatics Department at University of Applied Sciences of Gelsenkirchen, for helping me in different phases during my PhD work. I want to mention here Dr. Angela Lilienthal, who helped me a lot with the mathematical part of my work. Many thanks, to my colleague Mr. Fangwu Shu, who wrote most of the Visual C++ programs necessary to test the system.

I would like to thank also my parents for their support in all this period.

I would like to thank a lot my wife for understanding me in all this time when I had to work and let her alone. Without her understanding and support I could not have finished my thesis.

Liviu Toma Jr.

# Contents

<b>Chapter 1. Introduction</b> .....	1
1.1. Important items in Computer Vision .....	1
1.2. Scene Reconstruction .....	1
1.3. Accurate Visual Metrology .....	2
1.4. Thesis Outline .....	2
<b>Chapter 2. Projective Geometry</b> .....	4
2.1. Introduction in the geometry of the projective 2-space .....	4
2.1.1. Points and lines in $\mathbf{P}^2$ - duality .....	5
2.1.2. The line at infinity and absolute points .....	5
2.2. Introduction in the geometry of the projective 3-space .....	7
2.2.1. Points and planes in $\mathbf{P}^3$ - duality .....	7
2.2.2. The plane at infinity and the absolute conic .....	8
<b>Chapter 3. Camera: Models and Calibration Methods</b> .....	10
3.1. Camera Models .....	10
3.1.1 Distortion free models .....	10
3.1.2. Models including distortion .....	16
3.2. Calibration Methods .....	19
3.2.1. Off-line calibration methods .....	19
3.2.2. On-line calibration methods .....	20
3.3. Analysis of Camera Off-line Calibration Methods .....	21
3.3.1. General presentation of a camera calibration procedure .....	21
3.3.2. Lenz calibration method .....	27
3.3.3. Tsai calibration method .....	33
3.3.4. Contributions at the simulation and analysis of the errors .....	41
<b>Chapter 4. Stereo Sensor: Improvements and Contributions</b> .....	45
4.1. Mathematical Models .....	45
4.1.1. Description of the stereo sensor .....	45
4.1.2. Non-parallel configuration .....	46
4.1.3. Parallel configuration .....	47
4.1.4. Description of the camera model .....	47

4.2. Contributions at the Calibration Procedure .....	48
4.2.1. Description of the calibration device .....	48
4.2.2. Description of the calibration procedure .....	49
<b>Chapter 5. Image Processing and Shape Recognition .....</b>	<b>59</b>
5.1. Theoretical Introduction in Image Processing .....	59
5.1.1. Image enhancement techniques .....	59
5.1.2. Edge detection techniques .....	64
5.1.3. Methods in grey level segmentation .....	69
5.2. New Image Processing for Stereo Vision .....	76
5.2.1. Marks selection .....	76
5.2.2. Mathematical approach .....	78
5.2.3. Sub-pixel approach .....	83
<b>Chapter 6. Analysis of the 3D Measurements .....</b>	<b>93</b>
6.1. Contributions at the Development of Accurate Measurement Procedures .....	93
6.2. Non-parallel Configuration .....	98
6.2.1. Analysis of the errors .....	98
6.2.2. New method to eliminate the systematic errors .....	100
6.3. Parallel configuration .....	104
<b>Chapter 7. Industrial Applications .....</b>	<b>113</b>
7.1. Fixed Sensor Configuration .....	113
7.2. Mobile Sensor Configuration .....	120
<b>Chapter 8. Conclusions and Contributions .....</b>	<b>126</b>
<b>References .....</b>	<b>131</b>
<b>Annexes</b>	
Annex A – Logical Algorithm for Lenz Calibration Method .....	141
Annex B – C Program for Lenz Calibration Method .....	148
Annex C – Logical Algorithm for Tsai Calibration Method .....	153
Annex D – C Program for Tsai Calibration Method .....	160
Annex E – MATLAB Programs for making the analysis of the measurement errors ....	165

# List of Figures

Fig. 2.1. A model of the projective 2-space .....	6
Fig. 3.1. Pinhole camera geometry .....	10
Fig. 3.2. Mapping from 3D to 2D .....	11
Fig. 3.3. Image and camera coordinate systems .....	12
Fig. 3.4. The transformation between the camera frame and the world frame .....	13
Fig. 3.5. Lens radial distortion .....	16
Fig. 3.6. Details of the lens distortion .....	17
Fig. 3.7. The transformation from the 3D scene to the 2D image .....	21
Fig. 3.8. The correspondence between the center of the chip image and the center of the computer image .....	24
Fig. 3.9. The structure of chip area and pixels area .....	24
Fig. 3.10. The calibration board .....	29
Fig. 3.11. The calibration board in three different $z$ positions .....	35
Fig. 3.12. The simulation of the camera measurement .....	42
Fig. 3.13. The analysis of the results in the presence of 3D errors of the calibration points .....	43
Fig. 3.14. The analysis of the results in the presence of pixel errors of the identified calibration points .....	44
Fig. 4.1. Stereo configuration for big distance .....	45
Fig. 4.2. Stereo sensor in non-parallel configuration .....	46
Fig. 4.3. Non-parallel configuration – coordinates frames .....	46
Fig. 4.4. Stereo sensor in parallel configuration .....	47
Fig. 4.5. The calibration device .....	49
Fig. 4.6. Results of the calibration procedure .....	58
Fig. 5.1. Effect of the special average to an ideal edge .....	62
Fig. 5.2. Example of step edge .....	64
Fig. 5.3. Block scheme for an edge detector using a 2D gradient .....	65
Fig. 5.4. Linear interpolation of individual pixel thresholds .....	75
Fig. 5.5. Marks used to be identified in image processing .....	76
Fig. 5.6. Model to be recognized .....	77
Fig. 5.7. Example of histogram .....	78
Fig. 5.8. Details of a square region from the calibration plate .....	79
Fig. 5.9. Details of a region from the calibration plate with a black point .....	80
Fig. 5.10. Details at the cells level on the chip of a CCD camera .....	83
Fig. 5.11. Correspondence between Cartesian and Polar coordinates .....	85
Fig. 5.12. Details for an exploration line at sub-pixel level .....	85
Fig. 5.13. Sub-pixel resolution .....	86
Fig. 5.14. Graphical representation for $G(D)$ .....	87
Fig. 5.15. Graphical representation of the functions $f_1$ and $f_2$ .....	90
Fig. 5.16. Functions $f_1$ and $f_2$ - theoretical situation .....	90
Fig. 6.1. Measured 3D coordinates .....	97
Fig. 6.2. Real 3D coordinates .....	97
Fig. 6.3. The distribution of the errors for the coordinate $x$ .....	98
Fig. 6.4. The distribution of the errors for the coordinate $y$ .....	99
Fig. 6.5. The distribution of the errors for the coordinate $z$ .....	99
Fig. 6.6. The distribution of the errors for the coordinate $x$ after correction .....	101
Fig. 6.7. The distribution of the errors for the coordinate $y$ after correction .....	102

Fig. 6.8. The distribution of the errors for the coordinate $z$ after correction .....	102
Fig. 6.9. The distribution of the errors for the coordinate $x$ .....	104
Fig. 6.10. The distribution of the errors for the coordinate $y$ .....	105
Fig. 6.11. The distribution of the errors for the coordinate $z$ .....	105
Fig. 6.12. The distribution of the errors for the position vector .....	106
Fig. 6.13. The distribution of the errors for the coordinate $x$ .....	106
Fig. 6.14. The distribution of the errors for the coordinate $y$ .....	107
Fig. 6.15. The distribution of the errors for the coordinate $z$ .....	107
Fig. 6.16. The distribution of the errors for the position vector .....	108
Fig. 6.17. The distribution of the errors for the coordinate $x$ .....	108
Fig. 6.18. The distribution of the errors for the coordinate $y$ .....	109
Fig. 6.19. The distribution of the errors for the coordinate $z$ .....	109
Fig. 6.20. The distribution of the errors for the position vector .....	110
Fig. 6.21. The distribution of the errors for the coordinate $x$ .....	110
Fig. 6.22. The distribution of the errors for the coordinate $y$ .....	111
Fig. 6.23. The distribution of the errors for the coordinate $z$ .....	111
Fig. 6.24. The distribution of the errors for the position vector .....	112
Fig. 7.1. Definition of Camber .....	114
Fig. 7.2. Definition of Toe .....	114
Fig. 7.3. Stereo sensor and light projector .....	115
Fig. 7.4. Structured light projected on the tire surface .....	115
Fig. 7.5. Description of the measurement system .....	116
Fig. 7.6. Calibration of the measurement system .....	117
Fig. 7.7. Explanations for the measurement procedure .....	118
Fig. 7.8. Distribution of the errors for Camber .....	119
Fig. 7.9. Distribution of the errors for Toe .....	119
Fig. 7.10. The object frame definition .....	120
Fig. 7.11. Introducing the object – C program .....	121
Fig. 7.12. The deviation of the actual object with respect to the reference object .....	122
Fig. 7.13. Errors for the reference position of the object .....	122
Fig. 7.14. Measured relative position of the object frame .....	123
Fig. 7.15. The description of the test application .....	123
Fig. 7.16. Visual point and application point .....	124
Fig. 7.17. The taught position for one application point .....	124

## Introduction

### 1.1 Important items in Computer Vision

Having 2D representations of an environment the goal is to obtain information, which could be used for different purposes, from these available representations. One can divide this information in two categories. First category includes 3D details of the represented environment. In the literature, it is called scene reconstruction [Arm96]. The second category includes the motion of the camera, which generates that sequence of 2D representations. This is the most important requirement to realize realistic insertion of an artificial object in a video sequence [HZ03].

The subject of these PhD theses belongs to the first category. The goal of this chapter is to define precisely where in the great field of scene reconstruction is placed this work.

### 1.2 Scene Reconstruction

At a simple thinking, one can say that from a single image it is not possible to obtain scene reconstruction. But, using techniques of projective geometry it is possible in many cases to realize it. There are special techniques that involve the analysis of features such as parallel lines and vanishing points to determine the affine structure of a scene, for example by determining the line at infinity for observed planes in the image. Knowledge or assumptions about angles observed in the scene, most particularly orthogonal lines or planes can be used to upgrade the affine reconstruction to Euclidean, [HZ03].

The general case is to reconstruct scenes from several images. There are of course a lot of mathematical developments, which deals with the cases when there are two, respectively three images available for reconstruction. The basic tool in the reconstruction of point sets from two views is the fundamental matrix. This is  $3 \times 3$  matrix of rank 2. This matrix relates the coordinates of the corresponding points in the two images. For three views the role of the fundamental matrix is taken by the trifocal tensor. This is a  $3 \times 3 \times 3$  array of numbers that relate the coordinates of corresponding points and lines in the three images.

For any real scene there is a set of points representing the structure in that scene, and the position of those points can be measured in a Euclidean world coordinate frame. Any other representation is related to the original set of points by a transformation of their homogeneous coordinates. There are four level of representations: projective, affine, similarity, and Euclidean [Fau95]. Generally, similarity and Euclidean are grouped together as metric. The level in which a point is represented depends only on the transformation required to map the point to its real coordinates. Metric transformations are sub-groups of affine transformations and both are sub-groups of projective transformations [SK79]. For each level of representation there are different proprieties, which are invariant. Invariant propriety means that measurements of this propriety give the same value in the original level and in any other transformed frame at the same level [HZ03].

### 1.3 Accurate Visual Metrology

Not all the applications require a metric reconstruction. For example, only a projective reconstruction is needed for object recognition, and only an affine reconstruction is needed for path planning, grasping and fixation point tracking [Arm96]. If we want that the reconstruction to be the same as the original we need to make it at the Euclidean level, references [Cri99], and [Men01]. The goal of this PhD thesis is to analyze the problem of using stereo cameras to realize accurate 3D measurements. The 3D measured coordinates must reflect the real position of the measured point in the environment where the measurements are done. So, it is obviously clear that a Euclidean reconstruction is needed. To compute a Euclidean reconstruction requires the camera calibration to be known.

The camera calibration problem represents a very important research field. Starting with the research of Brown, reference [Bro71] a lot of scientists studied this subject having the goal to obtain a camera model as close as possible to the real camera, and to compute the parameters defined in their models, references [BC97], [Cum02], [DG97], [HA99], [LD99], [MC99], [Ste97], [TVDF89], [WM94] and [WMC03]. We also have treated detailed the calibration problem and brought contributions, as one will see in chapter 3 and 4.

### 1.4 Thesis Outline

This thesis is structured in 8 chapters. **Chapter 1** makes a brief introduction in the field of computer vision and tries to place the subject treated in my PhD thesis at the right place in this field. In **chapter 2** are presented basic knowledge about projective geometry,



which is the most important tool used to solve the scene reconstruction problem. **Chapter 3** is an overview of the existing camera models and calibration methods. In the last part of this chapter are presented my own contributions at the simulation and the analysis of the errors produced by measuring with a calibrated camera. In **chapter 4** are presented the two types of the stereo sensors, which were built for testing the developed algorithm. Then my improvements and contributions to the calibration procedure are described. **Chapter 5** presents in the first half a theoretical introduction in image processing and in the second half my algorithm developed in order to identify a certain point at sub-pixel level. A detailed analysis of the measurements done with the stereo sensors, described in chapter 4, is presented in **chapter 6**. Here are also presented my contributions in order to eliminate the systematic errors, which appear for the non-parallel configuration of the stereo sensor. **Chapter 7** describes some possible industrial applications for both fixed and mobile configurations of the stereo sensor. Also, a test application will be presented together with the obtained results. Finally, in **chapter 8** are presented the most important conclusions of this work, the main contributions in the field and some ideas for the future work.

## Projective Geometry

The Projective geometry is an extension of the Euclidean geometry and offers the possibility to solve different computer vision problems, which are mostly impossible to be solved using Euclidean geometry [Fau93]. The duality principle and the possibility of treating points situated at infinity as normal points are the most powerful tools of this geometry. In the following parts of this chapter are presented basic knowledge concerning the geometry of the projective 2-space and 3-space, which are mostly used in scene reconstruction.

A good support to learn the necessary knowledge of Projective geometry can be obtained by studying the references [Sp64], [SK79], and [Vra62]. Also, the most important tools of the Projective geometry used in computer vision are treated in the references [Fau93], [FL01], and [HZ03].

### 2.1 Introduction in the geometry of the projective 2-space

The projective 2-space corresponds to the Euclidean space  $\mathbf{R}^2$ . The projective 2-space is denoted with  $\mathbf{P}^2$ . One can represent an element of this space using a tri-dimensional vector  $(x_1, x_2, x_3)^T$ . Its corresponding element  $(X, Y)^T$  in  $\mathbf{R}^2$ , is computed using the following relations:

$$X = \frac{x_1}{x_3}, \quad (2.1)$$

$$Y = \frac{x_2}{x_3}. \quad (2.2)$$

From these two relations is clear that also the element  $(kx_1, kx_2, kx_3)^T$  of  $\mathbf{P}^2$ , corresponds to the same element  $(X, Y)^T$  of  $\mathbf{R}^2$  for any non-zero  $k$ . In fact, two such vectors related by an overall scaling are considered as being equivalent. An equivalence class of vectors under this equivalence relationship is known as a *homogeneous vector*. Any particular vector  $(x_1, x_2, x_3)^T$  is the representative of the equivalence class. The set of equivalence classes of vectors in  $\mathbf{R}^3 - (0, 0, 0)^T$  forms the projective 2-space  $\mathbf{P}^2$ .

### 2.1.1 Points and lines in $\mathbf{P}^2$ – duality

A line in a plane is represented by the following equation:

$$aX + bY + c = 0. \quad (2.3)$$

Different values for the components  $a$ ,  $b$ , and  $c$  defines different lines of that plane. This way, one can define a line by the vector  $(a, b, c)^T$ , which is an element of  $\mathbf{P}^2$ . So, an element of  $\mathbf{P}^2$  can be interpreted as a line or as a point. This represents the duality principle between lines and points in the projective 2-space.

In the following part we will define some basic operations between points and lines using also the duality principle. A point  $\mathbf{x} = (x_1, x_2, x_3)^T$  belongs to a line  $\mathbf{l} = (a, b, c)^T$  if it satisfies the following equation:

$$\mathbf{l}^T \mathbf{x} = 0. \quad (2.4)$$

The intersection between two lines  $\mathbf{l}_1$  and  $\mathbf{l}_2$  represents a point  $\mathbf{x}$ . The equation (2.5) gives the connection between these elements, as follows:

$$\mathbf{x} = \mathbf{l}_1 \times \mathbf{l}_2. \quad (2.5)$$

Using the duality, one can define a line  $\mathbf{l}$  passing through the points  $P_1$  and  $P_2$  as being the cross product between the homogeneous coordinates of these two points. So, one can write the following relation:

$$\mathbf{l} = \mathbf{x}_1 \times \mathbf{x}_2, \quad (2.6)$$

where  $\mathbf{x}_1$  and  $\mathbf{x}_2$  are the respectively the homogeneous vectors of the points  $P_1$  and  $P_2$ .

### 2.1.2 The line at infinity and absolute points

A special category of points belonging to  $\mathbf{P}^2$  are that ones which have the last component equal to 0. These points have no correspondent in  $\mathbf{R}^2$ . They are called *ideal points* or *points at infinity*. This set of ideal points may be written  $(x_1, x_2, 0)^T$ , where a particular

point is specified by the ratio  $x_1 : x_2$ . One can see that all these points lie on a single line, the line at infinity. We denote this line  $\mathbf{l}_\infty = (0, 0, 1)^T$ . One can easily verify the following:

$$(0 \ 0 \ 1)(x_1 \ x_2 \ 0)^T = 0. \quad (2.7)$$

Using the relation (2.5) one finds that a line  $\mathbf{l} = (a, b, c)^T$  intersects the line at infinity in the ideal point  $(b, -a, 0)$ . A line  $\mathbf{l}_1 = (a, b, c_1)$  parallel to  $\mathbf{l}$  intersects the line at infinity in the same point as line  $\mathbf{l}$ . So, an important conclusion is that in the projective 2-space, two lines always intersect. If the lines are parallel their intersection point is an ideal point situated on the line at infinity.

Referring now to the line  $\mathbf{l} = (a, b, c)^T$ , using inhomogeneous notation, the vector  $(b, -a)$  is a vector tangent to the line, and orthogonal to the line normal  $(a, b)$  and so, represents the line's direction. If the line's direction varies, also the ideal point  $(b, -a, 0)$  varies over the line at infinity. For this reason one can consider the line at infinity as the set of directions of lines in the plane.

In figure 2.1, one can see a model of the projective 2-space. Points and lines of  $\mathbf{P}^2$  are represented by rays and planes, respectively, through the origin in  $\mathbf{R}^3$ . Rays lying in the  $x_1x_2$ -plane are representing ideal points. The  $x_1x_2$ -plane represents the line at infinity [HZ03].

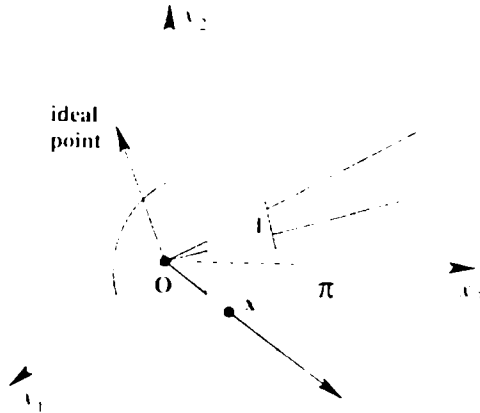


Fig. 2.1. A model of the projective 2-space.

Between the ideal points there are two points, which have special properties. These points are called *circular points* or *absolute points*. Their canonical coordinates are:

$$\mathbf{I} = \begin{pmatrix} 1 \\ i \\ 0 \end{pmatrix}, \quad \mathbf{J} = \begin{pmatrix} 1 \\ -i \\ 0 \end{pmatrix}. \quad (2.8)$$

## 2.2 Introduction in the geometry of the projective 3-space

The projective 3-space corresponds to the Euclidean space  $\mathbf{R}^3$ . This space will be denoted with  $\mathbf{P}^3$ . If we consider an element of  $\mathbf{P}^3$  as being  $(x_1, x_2, x_3, x_4)$  then one can compute its corresponding element  $(X, Y, Z)$  in  $\mathbf{R}^3$  as follows:

$$X = \frac{x_1}{x_4}, \quad (2.9)$$

$$Y = \frac{x_2}{x_4}, \quad (2.10)$$

$$Z = \frac{x_3}{x_4}. \quad (2.11)$$

The same as in the case of the projective 2-space also, for the projective 3-space its elements are homogenous vectors.

### 2.2.1 Points and planes in $\mathbf{P}^3$ – duality

A plane in the projective 3-space can be written as follows:

$$\pi_1 X + \pi_2 Y + \pi_3 Z + \pi_4 = 0, \quad (2.12)$$

where  $(X, Y, Z)$  are the inhomogeneous coordinates of any point which lies on this plane. Using homogeneous coordinates the relation (2.12) will take the following form:

$$\pi_1 x_1 + \pi_2 x_2 + \pi_3 x_3 + \pi_4 x_4 = 0 \quad (2.13)$$

This last relation can be also written as follows:

$$\boldsymbol{\pi}^T \mathbf{x} = 0, \quad (2.14)$$

where  $\boldsymbol{\pi} = (\pi_1, \pi_2, \pi_3, \pi_4)$  represents a plane and  $\mathbf{x} = (x_1, x_2, x_3, x_4)^T$  represents a point. This way, an element of the projective 3-space can be interpreted as a plane or as a point. So, the

duality between lines and points from the projective 2-space exists also in the projective 3-space, but between planes and points.

A line in  $\mathbf{P}^3$  is defined by the join of two points or the intersection of two planes. Lines have 4 degrees of freedom in the projective 3-space [HZ03].

### 2.2.2 The plane at infinity and the absolute conic

As it was presented in the case of the projective 2-space the line at infinity and the absolute points as powerful tools of this space their equivalent in the projective 3-space are the plane at infinity and the absolute conic.

We consider a point  $\mathbf{x} = (x_1, x_2, x_3, x_4)^T$  of the projective 3-space. A special category of points is represented by that ones which have  $x_4 = 0$ . These points have no correspondent in  $\mathbf{R}^3$ . They are called ideal points or points at infinity. One can easy verify the next relation:

$$(0 \ 0 \ 0 \ 1)(x_1 \ x_2 \ x_3 \ 0)^T = 0, \quad (2.15)$$

where  $\pi_\infty = (0, 0, 0, 1)^T$  represents the plane at infinity. So, the conclusion is that all the ideal points lie on the plan at infinity.

The plane at infinity contains all the directions  $\mathbf{D} = (x_1, x_2, x_3, 0)^T$ , and enables the identification of affine proprieties such as parallelism. This way, we have the following:

- two planes are parallel if, and only if, their line of intersection is on the  $\pi_\infty$ ;
- a line is parallel to another line, or to a plane, if their point of intersection is on  $\pi_\infty$ .

In order to define the absolute conic we have first to explain what is a conic. A conic is a curve described by a second-degree equation in the plane. In Euclidean geometry conics are divided in three main types: hyperbola, ellipse, and parabola. Using inhomogeneous coordinates the equation of a conic is:

$$aX^2 + bXY + cY^2 + dX + ey + f = 0. \quad (2.16)$$

With homogeneous coordinates the equation (2.16) becomes as follows:

$$ax_1^2 + bx_1x_2 + cx_2^2 + dx_1x_3 + ex_2x_3 + fx_3^2 = 0. \quad (2.17)$$

Using matrices and vectors the equation (2.17) can be also written as follows:

$$\mathbf{x}^T \mathbf{C} \mathbf{x} = 0, \quad (2.18)$$

where

$$\mathbf{C} = \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix}, \quad (2.19)$$

represents the conic coefficient matrix.

The *absolute conic*,  $\Omega_\infty$ , is a conic on the plane of infinity  $\pi_\infty$ . The points belonging to the absolute conic satisfy the next two equations:

$$x_1^2 + x_2^2 + x_3^2 = 0, \quad (2.20)$$

$$x_4 = 0. \quad (2.21)$$

For the directions on  $\pi_\infty$ , the defining equation (2.20) can be written as follows:

$$(x_1 \ x_2 \ x_3) \mathbf{I} (x_1 \ x_2 \ x_3)^T = 0. \quad (2.22)$$

This means that absolute conic corresponds to a conic with a coefficient matrix  $\mathbf{C} = \mathbf{I}$ . So, it is a conic of purely imaginary points on the plane of infinity [HZ03].

This basic knowledge about Projective geometry will be used mostly in sub-chapter 3.1 in order to define the camera models. Also, all the on-line camera calibration methods are making use of this geometry.

## Camera: Models and Calibration Methods

### 3.1 Camera Models

A camera is a mapping between the 3D world and a 2D image. A camera model is a matrix with particular proprieties that represents the camera mapping. Due to the lens proprieties we have to divide the camera models in two categories: ideal models, which are distortion free models and real models, which include the influence of the lens distortion. Both categories will be presented in the following parts of this chapter.

#### 3.1.1 Distortion free models

The simplest camera model is the *pinhole model*, as one can see in figure 3.1.

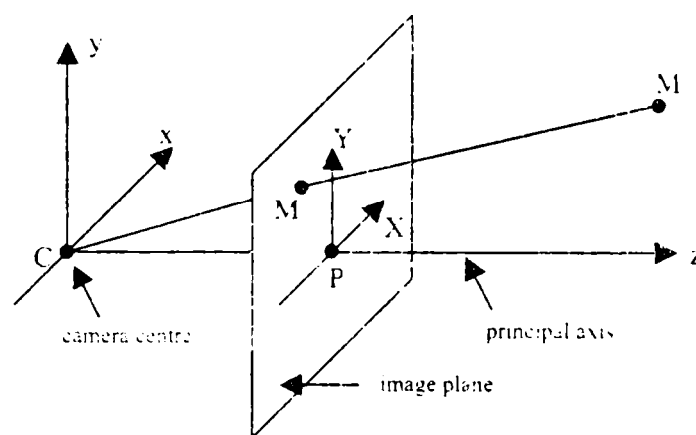


Fig. 3.1. Pinhole camera geometry.

We define a coordinate system with the origin in point C. This coordinate system is called the *camera coordinate frame*. The point C represents the centre of projection called also *optical center* or *camera center*. The axis  $x$  and  $y$  of the camera coordinate frame are defining a plane parallel with the image plane. This plane, called also *focal plane*, is described mathematically by the next equation:

$$z = f. \quad (3.1)$$



where  $f$  is the *focal length*. The line from the camera center perpendicular to the image plane is called the *principal axis* or *principal ray* of the camera. The point P where the principal ray meets the image plane is called *principal point* or *image center*.

A point M having the coordinates  $(x, y, z)^T$  in space is mapped to the point on the image plane where a line joining the point x to the center of the projection meets the image plane. As one can see in figure 3.2, from similar triangles one can simply compute that the point M is mapped to the point  $(fx/z, fy/z)^T$ , on the image plane.

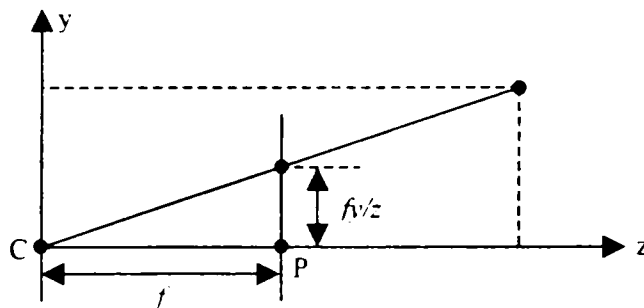


Fig. 3.2. Mapping from 3D to 2D.

Using homogeneous coordinates the mapping of the world point M to the image point can be written mathematically as follows:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} fx \\ fy \\ z \end{pmatrix} = \begin{bmatrix} f & & 0 \\ & f & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}. \quad (3.2)$$

The matrix from the relation (3.2) can be also written as  $diag(f, f, 1)[\mathbf{I} \mid \mathbf{0}]$  where  $diag(f, f, 1)$  is a diagonal matrix and  $[\mathbf{I} \mid \mathbf{0}]$  represents a matrix divided up into a 3 x 3 block (the identity matrix) and a column vector, in this case the zero vector. We introduce now the notation  $\mathbf{x}$  for the world point represented by the homogeneous 4-vector  $(x, y, z, 1)^T$ , the notation  $\mathbf{X}$  for the image point represented by the homogeneous 3-vector  $(fx, fy, z)^T$ , and the notation  $\mathbf{P}$  for the 3 x 4 homogeneous *camera projection matrix*. This way, the relation (3.2) can be written compactly as follows:

$$\mathbf{X} = \mathbf{P}\mathbf{x}, \quad (3.3)$$

where

$$\mathbf{P} = \text{diag}(f \quad f \quad 1) [\mathbf{I} | \mathbf{0}]. \quad (3.4)$$

The relation (3.4) defines the camera projection matrix for the pinhole model of central projection. This model assumed that the origin of the coordinates in the image plane is situated at the principal point P. If they are not the same, as one can see in figure 3.3, then the mapping will be done according to the next relation:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} fx + zP_x \\ fy + zP_y \\ z \\ 1 \end{pmatrix} = \begin{bmatrix} f & P_x & 0 \\ & f & P_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}. \quad (3.5)$$

where  $(P_x, P_y)$  are the coordinates of the principal point with respect to the image coordinates.

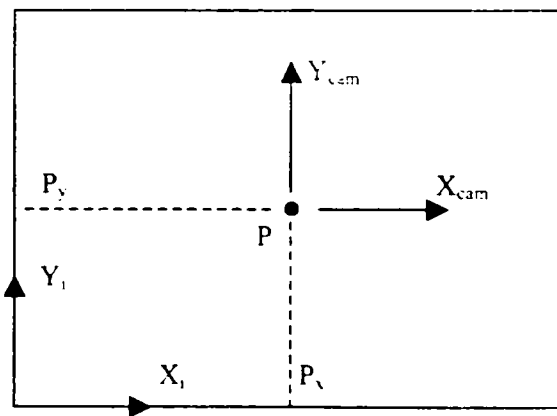


Fig. 3.3. Image and camera coordinate systems.

The equation (3.5) can be also written in the following form:

$$\mathbf{X} = \mathbf{K} [\mathbf{I} | \mathbf{0}] \mathbf{x}_{cam}, \quad (3.6)$$

where  $\mathbf{x}_{cam}$  is the coordinates vector of the point X relative to the camera frame, and  $\mathbf{K}$  represents the *camera calibration matrix* having the following form:

$$\mathbf{K} = \begin{bmatrix} f & P_x \\ & f & P_y \\ & & 1 \end{bmatrix}. \quad (3.7)$$

Generally the points in the space are not given with respect to the camera frame, but with respect to a different Euclidean frame, known as the *world coordinate frame*. Having a point  $X$  we will denote, as before, with  $x$  its homogeneous coordinates and with  $x^E$  its inhomogeneous coordinates. Between the coordinates of the point  $X$  with respect to the camera frame and its coordinates with respect to the world frame we have the following relation:

$$\mathbf{x}_{cam}^E = \mathbf{R}(\mathbf{x}_w^E - \mathbf{c}^E). \quad (3.8)$$

As one can see in figure 3.4,  $\mathbf{R}$  represents the rotation from the camera frame to the world frame and  $\mathbf{c}^E$  the inhomogeneous coordinates of the camera center point with respect to the world frame.

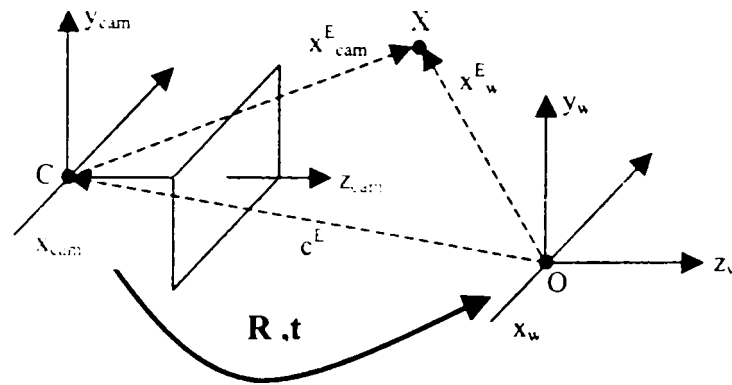


Fig. 3.4. The transformation between the camera frame and the world frame.

Using homogeneous coordinates the relation (3.8) can be written as follows:

$$\mathbf{x}_{cam} = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\mathbf{c}^E \\ 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\mathbf{c}^E \\ 0 & 1 \end{bmatrix} \mathbf{x}_w. \quad (3.9)$$

The general mapping given by a pinhole camera is obtain by replacing the  $\mathbf{x}_{cam}$  from the relation (3.6) with the relation (3.9):

$$\mathbf{X} = \mathbf{KR} \begin{bmatrix} \mathbf{I} & -\mathbf{c}^E \end{bmatrix} \mathbf{x}_w. \quad (3.10)$$

This way, the camera projection matrix  $\mathbf{P}$  of a pinhole camera described by the following relation:

$$\mathbf{P} = \mathbf{K}\mathbf{R}\left[\mathbf{I} \mid -\mathbf{c}^t\right], \quad (3.11)$$

has 9 degrees of freedom: 3 for  $\mathbf{K}$  (the elements  $f, P_x, P_y$ ), 3 for  $\mathbf{R}$ , and 3 for  $\mathbf{c}^E$ . The parameters contained in  $\mathbf{K}$  are called the *internal* camera parameters, or the *internal orientation* of the camera. The parameters of  $\mathbf{R}$  and  $\mathbf{c}^E$ , which give the position and the orientation of the camera frame relative to a world frame, are called the *external* camera parameter or the *exterior orientation* [HZ03].

In most of the cases is not useful to make the camera center explicit, but to use the transformation from the camera coordinates to the world coordinates as follows:

$$\mathbf{x}_{cam} = \mathbf{R}\mathbf{x} + \mathbf{t}, \quad (3.12)$$

where

$$\mathbf{t} = -\mathbf{R}\mathbf{c}^E. \quad (3.13)$$

This way, the camera projection matrix will become as follows:

$$\mathbf{P} = \mathbf{K}\left[\mathbf{R} \mid \mathbf{t}\right]. \quad (3.14)$$

The pinhole model assumes that the image coordinates are Euclidean coordinates having equal scales in both directions. In the case of a CCD camera can be possible that the number of the pixels per unit in both directions to be different. If we denote with  $s_x$  the number of the pixels per unit in  $X$  direction and with  $s_y$  the number of the pixels in  $Y$  direction then the general form of the calibration matrix of a CCD camera will be obtain by multiplying the calibration matrix of the pinhole model with an extra factor  $diag(s_x, s_y, 1)$ . This way, we will obtain as follows:

$$\mathbf{K} = \begin{bmatrix} s_x & & \\ & s_y & \\ & & 1 \end{bmatrix} \begin{bmatrix} f & P_x \\ f & P_y \\ & 1 \end{bmatrix} = \begin{bmatrix} s_x f & s_x P_x \\ & s_y f & s_y P_y \\ & & 1 \end{bmatrix}. \quad (3.15)$$

With the notations:

$$p_x = s_x f, \quad (3.16)$$

$$p_y = s_y f, \quad (3.17)$$

$$C_x = s_x P_x, \quad (3.18)$$

$$C_y = s_y P_y, \quad (3.19)$$

the relation (3.15) becomes as follows:

$$\mathbf{K} = \begin{bmatrix} p_x & & C_x \\ & p_y & C_y \\ & & 1 \end{bmatrix}. \quad (3.20)$$

The conclusion is that a CCD camera has 10 degrees of freedom, one more than the pinhole camera because the calibration matrix  $\mathbf{K}$  is defined by four parameters.

A more complete model has the following form of the calibration matrix:

$$\mathbf{K} = \begin{bmatrix} p_x & s & C_x \\ & p_y & C_y \\ & & 1 \end{bmatrix}, \quad (3.21)$$

where  $s$  is called the *skew parameter*. A camera whose calibration matrix has the form described by the relation (3.21) is called a *finite projective camera* [HZ03]. Another form to describe the calibration matrix for a finite projective camera is the following:

$$\mathbf{K} = \begin{bmatrix} p_x & -p_x \cot \theta & C_x \\ & p_y / \sin \theta & C_y \\ & & 1 \end{bmatrix}, \quad (3.22)$$

where  $\theta$  represents the angle between the axis of the image plane [Arm96]. Usually, this angle is 90 degrees and then the calibration matrix will have the form described by the equation (3.20).

A finite projective camera has 11 degrees of freedom. This is the same number of degrees of freedom as a  $3 \times 4$  matrix, defined up to an arbitrary scale. The conclusion is that the set of camera matrices of finite projective cameras is identical with the set of homogeneous  $3 \times 4$  matrices for which the left hand  $3 \times 3$  sub-matrix is non-singular [HZ03].

The last step in this hierarchy of projective cameras is to define the *general projective camera*. This is represented by an arbitrary homogeneous  $3 \times 4$  matrix of rank 3. The rank 3 requirement arises because if the rank is less than 3 then the range of the mapping matrix will be a line or a point and not the whole plane; in other words not a 2D image [HZ03].

### 3.1.2 Models including distortion

All the camera models presented before can be applied when the focal lens is big and the lens has a high quality. If we use normal lenses with a small focal lens then the distortion will have a big influence to the obtained image. There are two ways to eliminate the effect of the lens distortion. One way is to make a correction of the image and to obtain an undistorted image, which can be then used for the models defined in the sub-chapter 3.1.1. This can be realized using some constraints, for example the constraint that a line must be always straight [DF01]. Another way is to find a mathematical function, which relates the distorted coordinates to the undistorted coordinates and then to replace them in the equations obtained from the distortion free models [WCH92].

There are three types of the distortion, which can influence an image: radial distortion, decentering distortion and prism distortion [WCH92]. From these types the most important influence is given by the radial distortion. In the following parts of this chapter it will be presented a mathematical model, which describes this type of distortion.

As one can see in figure 3.5 there are two types of radial distortion: barrel distortion and pincushion distortion [Lan58], [Dod82]. Barrel distortion corresponds to a negative displacement of the image points and pincushion distortion to a positive displacement of the image points.

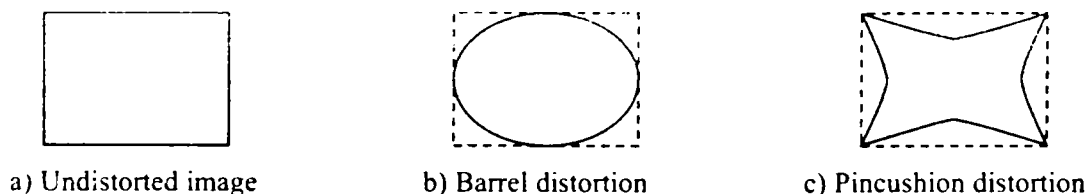


Fig. 3.5. Lens radial distortion.

For the cameras we used in our experiments a distortion of type b) was detected. We take two points as in figure 3.6.  $P_u$  is the ideal point, undistorted and  $P_d$  is the real point, distorted. The coordinates of these points are  $X_u, Y_u$  respectively  $X_d, Y_d$ .

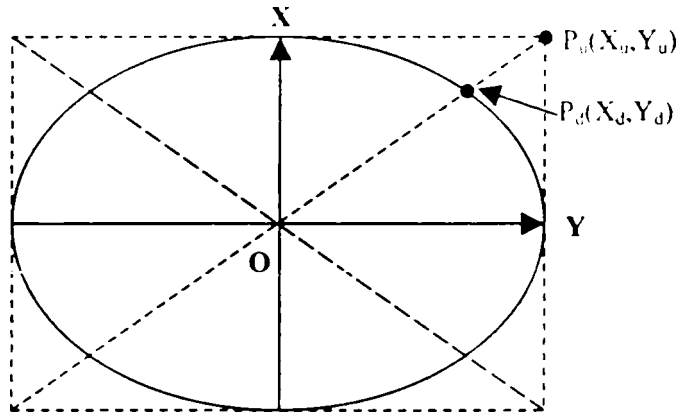


Fig. 3.6. Details of the lens distortion.

We will approximate the distortion with the following relations:

$$X_u = X_d + X_d \cdot f(R_d), \quad (3.23)$$

$$Y_u = Y_d + Y_d \cdot f(R_d), \quad (3.24)$$

where  $R_d$  is defined by the next relation:

$$R_d = \left( (X_d)^2 + (Y_d)^2 \right)^{\frac{1}{2}}. \quad (3.25)$$

According to the literature there are many ways to approximate the function  $f$ . We decided to use the following form for the function  $f$ :

$$f(R_d) = k_p \cdot R_d^2, \quad (3.26)$$

where  $k_p$  is the coefficient of the radial distortion and has a positive value.

With this last relation we can replace  $f$  in the relations (3.23) and (3.24) and we obtain the next two relations:

$$X_u = X_d \cdot (1 + k_p \cdot R_d^2), \quad (3.27)$$

$$Y_u = Y_d \cdot (1 + k_p \cdot R_d^2). \quad (3.28)$$

Knowing the fact that  $k_p \cdot R_d^2$  is very small we can write according to Taylor approximation the following relation:

$$\frac{1}{1 - k_p \cdot R_d^2} = 1 + k_p \cdot R_d^2, \quad (3.29)$$

then the relations (3.27) and (3.28) will become:

$$X_u = X_d \cdot \frac{1}{1 - k_p \cdot R_d^2}, \quad (3.30)$$

$$Y_u = Y_d \cdot \frac{1}{1 - k_p \cdot R_d^2}. \quad (3.31)$$

When we introduced the coefficient of the radial distortion we considered its value positive. We will make the following notation:

$$k_p = -k, \quad (3.32)$$

where  $k$  takes negative values. With this notation the relation (3.30) and (3.31) the next two relations, which will be used in our future calculation:

$$X_u = X_d \cdot \frac{1}{1 + k \cdot R_d^2}, \quad (3.33)$$

$$Y_u = Y_d \cdot \frac{1}{1 + k \cdot R_d^2}. \quad (3.34)$$



## 3.2 Calibration Methods

Considering one of the camera models defined in the sub-chapter 3.1 the goal is to compute the components of the camera projective matrix, or with other words, to compute camera internal parameters and the camera external parameters. The calibration methods can be divided in two categories.

The first category includes the traditional calibration methods, based on images from a special calibration object with a known structure. These methods are called also *off-line calibration methods*. The big advantage of these methods is the high accuracy of the obtained camera parameters. The disadvantages are that they cannot be applied when the camera parameters are changing during normal operation, as zooming, or when we try to reconstruct a scene from a pre-recorded image sequence [Arm96]. In the sub-chapter 3.2.1 it will be presented an overview of the off-line calibration methods.

The second category includes the *on-line calibration methods*. They were introduced first by Faugeras and his collaborators [FLM92], [MF92]. They introduced the idea that a camera can be calibrated using only point matches between images, without requiring the knowledge of the scene. They called this method *camera self-calibration method*. This allows the possibility to reconstruct a scene from pre-recorded images and to compute the camera parameters during the normal vision tasks, references [AP95], [CDR99], [CT90], [HA97], [LL96] and [Stu92]. In the sub-chapter 3.2.2 it will be presented an overview of the on-line calibration methods.

### 3.2.1 Off-line calibration methods

A very good survey of the traditional calibration methods is presented in the reference [ASB00]. According to this reference they are five different off-line calibration methods. The other calibration methods combine these five or are similar to them. The calibration method depends on the camera model used to simulate the behavior of the camera.

1. The first method is the method of Hall, [HTMS82]. He considers a linear model (without distortion) and computes directly the elements of the projection matrix  $\mathbf{P}$ . Considering that the 3D position of a set of  $n$  ( $n > 6$ ) calibration points and their 2D projection in the image are known one can obtain the elements of projection matrix using least squares technique, [PTVF92].

2. The second method is the method of Faugeras – Toscani, [FT86]. Also this method considers a linear model, but the method computes directly the elements of the calibration

matrix  $\mathbf{K}$  (the internal camera parameters), the elements of the rotation matrix  $\mathbf{R}$ , and the translation vector  $\mathbf{t}$  (the external parameters). They use some special notations and mathematical operators to solve the obtained system of linear equations [FT86].

3. The third method is the method of Faugeras – Toscani with radial distortion [FT87]. This method uses a non-linear model considering also the influence of the radial distortion. This way, the equation system turns into a non-linear system so the least-square techniques, used before, have to be combined with an iterative algorithm to solve this system.

4. The fourth method is the method of Tsai [Tsa86], [Tsa87], and [LT88]. He used also of a non-linear model including the radial distortion. The method uses a two-stage technique. In the sub-chapter 3.3.2 we will present in details this calibration method.

5. The fifth calibration method is the method of Weng [WCH92]. He improves the model of Faugeras – Toscani, by including three types of lens distortion. This method uses a two-stage technique, the same as the method of Tsai.

### 3.2.2 On-line calibration methods

One can divide the on-line calibration methods from the beginning in two categories: self-calibration of the cameras, which have unchanged internal parameters ( $K=ct.$ ) and self-calibration of the cameras having fixed internal parameters.

Interesting calibration methods belonging to the first categories are presented in the references [AHH99], [AHR01], [HM03] and [Stu97]. They are dealing with the problem of calibrating zooming cameras.

Different approaches of the self-calibration methods belonging to the second category are very well summarized in the reference [Arm96]. He divides these methods in another two sub-categories: approaches for self-calibration for a monocular camera and approaches for self-calibration for a stereo head.

Concerning the monocular camera, important calibration algorithms were developed by Faugeras et al., references [FLM92], [LF96], [LF97] and Hartley, references [Har92], [Har94], [Har97]. Their methods are based on identifying the image of the absolute conic, which is equivalent with finding the camera calibration.

Concerning the stereo head, important calibration algorithms were developed by Zissermann et al., reference [ZBR95]. He shows that is possible to obtain constraints to the plane at infinity and the image of the absolute conic from the vector decomposition of the projective transformation [Arm96]. Other interesting algorithms belonging to the same category were developed by and Zhuang et al., references [Zhu95], [ZRXW93].

### 3.3 Analysis of Camera Off-line Calibration Methods

#### 3.3.1 General presentation of a camera calibration procedure

In this sub-chapter we will present two of the most important camera off-line calibration methods. Before starting to present them it is necessary to make a short description of a general calibration procedure.

There are two important steps, which must be presented. The first one is referring to the transformation from the 3D scene to a 2D chip image and the second one is referring to the transformation from the chip image to the computer image.

##### A. The transformation from a 3D scene to a 2D chip image

In the following part it will be analyzed the relations between the 3D coordinates of a real point  $P(x, y, z)$  and the 2D coordinates of his correspondent point  $P_d(X_d, Y_d)$  on the chip.

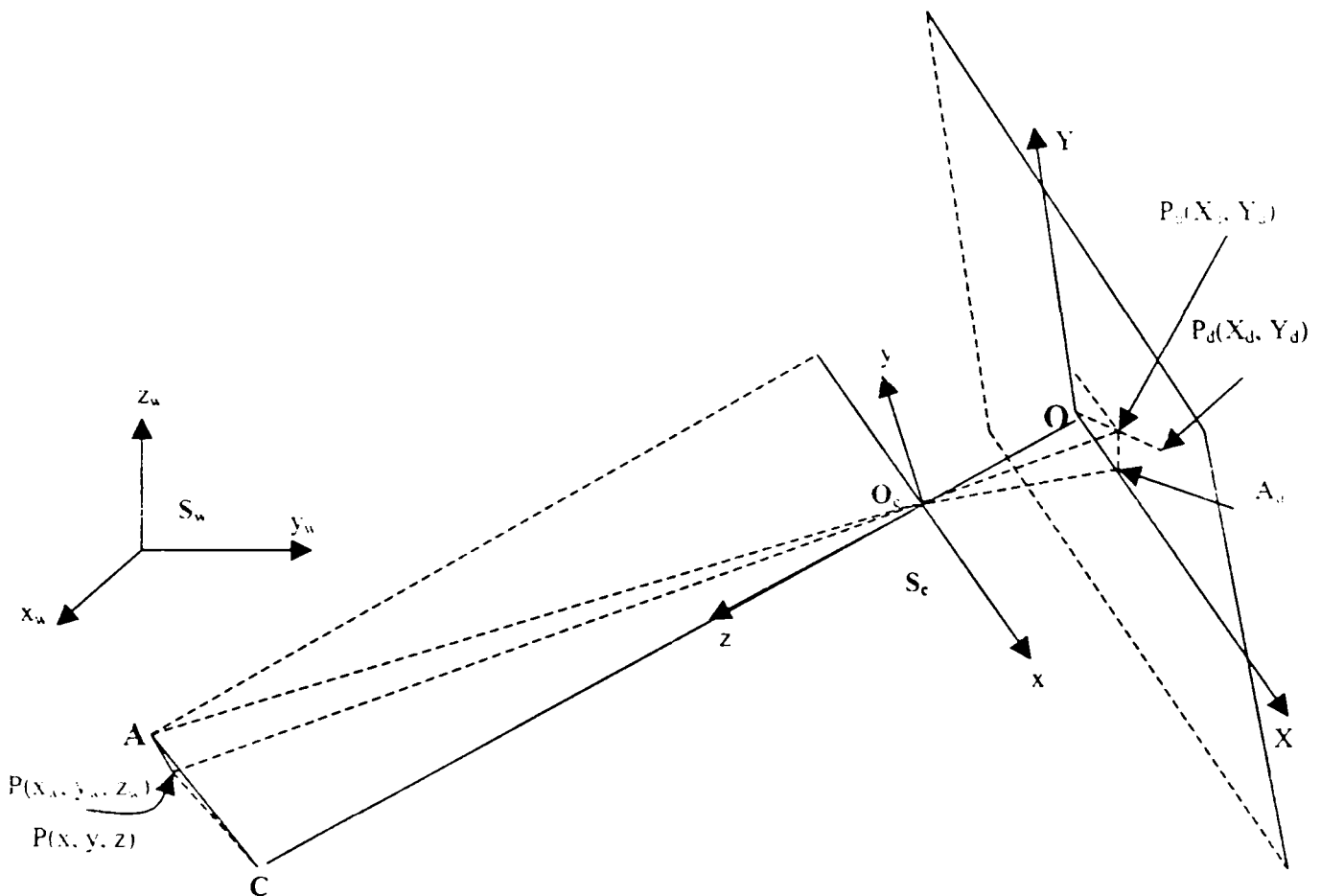


Fig.3.7. The transformation from the 3D scene to the 2D image.

The notations used in figure 3.7 have the following meaning:

- $S_w$  represents the world frame or the reference frame;
- $S_c$  represents the camera frame;

- P represents a point whose coordinates are:  $(x_w, y_w, z_w)$ , with respect to the world frame, and  $(x, y, z)$ , with respect to the camera frame;

-  $P_u(X_u, Y_u)$  is the image point of P if a perfect pinhole camera model is used;

-  $P_d(X_d, Y_d)$  is the actual image point which differs from  $P_u(X_u, Y_u)$  due to lens distortion;

Going further, it will be established first the mathematical relations between 3D coordinates  $(x, y, z)$  and the 2D coordinates  $(X_u, Y_u)$ . Using the geometry we obtain, as follows:

$$\Delta ACO_c \approx \Delta A_uOO_c \Rightarrow \frac{AC}{A_uO} = \frac{CO_c}{OO_c} \Rightarrow \frac{-x}{X_u} = \frac{z}{f} \Rightarrow X_u = -f \frac{x}{z}, \quad (3.35)$$

$$\left. \begin{aligned} \Delta PCO_c \approx \Delta P_uOO_c &\Rightarrow \frac{PO_c}{P_uO_c} = \frac{CO_c}{OO_c} \\ \Delta APO_c \approx \Delta A_uP_uO_c &\Rightarrow \frac{AP}{A_uP_u} = \frac{PO_c}{P_uO_c} \end{aligned} \right\} \Rightarrow \frac{AP}{A_uP_u} = \frac{CO_c}{OO_c} \Rightarrow \frac{-y}{Y_u} = \frac{z}{f}$$

$$\Rightarrow Y_u = -f \frac{y}{z}, \quad (3.36)$$

where  $OO_c$  represents the focal length, and his value was noted by  $f$ .

The relations between the distorted coordinates  $(X_d, Y_d)$  and the undistorted coordinates  $(X_u, Y_u)$  are given by the relations (3.33), and (3.34). From these two relations one can obtain the following:

$$\frac{X_u}{Y_u} = \frac{X_d}{Y_d}. \quad (3.37)$$

Using now the relations (1.1), (1.2) and (1.6) we obtain:

$$\frac{X_d}{Y_d} = \frac{x}{y}. \quad (3.38)$$

In this moment we have the relations (3.35), (3.36) and (3.38), which represent the connection between the 3D scene and his correspondent 2D image. The next step is to find some relations between the world coordinates of the point P, which are  $(x_w, y_w, z_w)$  and their correspondents  $(X_d, Y_d)$ , from the chip.

Between the world frame  $S_w$  and the camera frame  $S_c$  is possible to write the next relation:

$$S_w = S_c T_{S_c}^{S_w}, \quad (3.39)$$

where  $T_{S_c}^{S_w}$  represents the transformation from the camera system to the world system. From the mathematical point of view  $T_{S_c}^{S_w}$  is a matrix, which has the following form:

$$T_{S_c}^{S_w} = \begin{bmatrix} r_1 & r_2 & r_3 & t_x \\ r_4 & r_5 & r_6 & t_y \\ r_7 & r_8 & r_9 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ where } \mathbf{R} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \text{ represents the rotation, and}$$

$\mathbf{T} = [t_x \quad t_y \quad t_z]^T$  the translation between these two systems.

Going further, one can write the next relation:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}, \quad (3.40)$$

where  $(x, y, z)$  are the coordinates of the point P with respect to the camera frame and  $(x_w, y_w, z_w)$  are the coordinates of the same point P, but with respect to the world frame.

From the relation (3.40) one can simply write the next three relations:

$$x = x_w r_1 + y_w r_2 + z_w r_3 + t_x, \quad (3.41)$$

$$y = x_w r_4 + y_w r_5 + z_w r_6 + t_y, \quad (3.42)$$

$$z = x_w r_7 + y_w r_8 + z_w r_9 + t_z. \quad (3.43)$$

Using the relations (3.38), (3.41) and (3.42) we will obtain the relation between the 3D coordinates of a point and its 2D correspondent coordinates in the chip image:

$$\frac{X_d}{Y_d} = \frac{x_w r_1 + y_w r_2 + z_w r_3 + t_x}{x_w r_4 + y_w r_5 + z_w r_6 + t_y}. \quad (3.44)$$

B. The transformation from the 2D chip image to the 2D computer image

In the following part we will find the relations between the coordinates  $(X_d, Y_d)$  of a point P, from the chip image and the coordinates  $(X_p, Y_p)$  of its correspondent point, on the computer image.

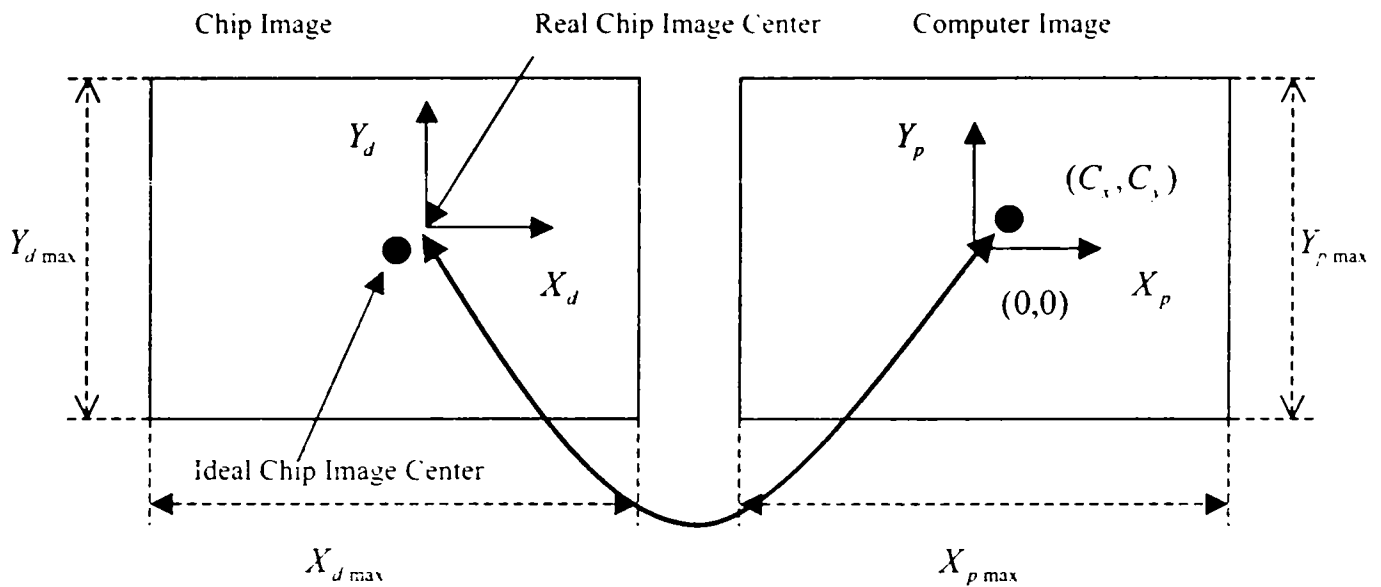


Fig. 3.8. The correspondence between the center of the chip image and the center of the computer image.

Using the information from the figure 3.8. it is possible to write the next two relations:

$$X_p = coefx \cdot X_d + C_x, \quad (3.45)$$

$$Y_p = coefy \cdot Y_d + C_y, \quad (3.46)$$

where  $(C_x, C_y)$  represent the coordinates of the real center image from the chip and  $coefx$  and  $coefy$  are two scale factors, which will be presented detailed in the next part.

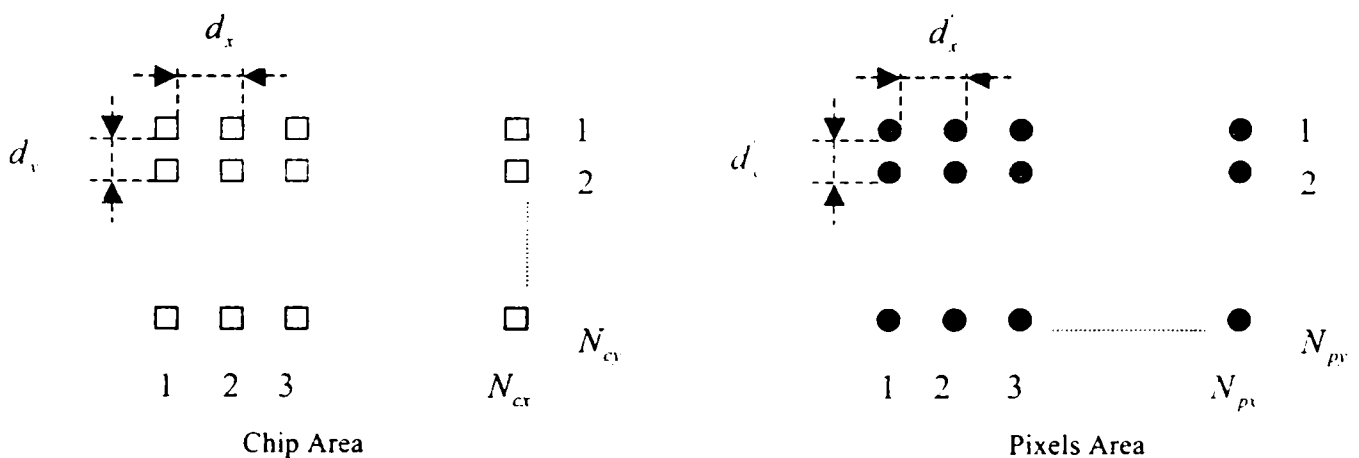


Fig. 3.9. The structure of chip area and pixels area.

In figure 3.8,  $X_{dmax}$ ,  $Y_{dmax}$  are the chip dimensions and  $X_{pmax}$ ,  $Y_{pmax}$  represent the total number of pixels in  $x$ , respectively  $y$ , direction. So, we can write the next relations:

$$X_{dmax} = N_{cx} \cdot d_x, \quad (3.47)$$

$$Y_{dmax} = N_{cy} \cdot d_y, \quad (3.48)$$

$$X_{pmax} = N_{px}, \quad (3.49)$$

$$Y_{pmax} = N_{py}, \quad (3.50)$$

where, as one can see in figure 3.9, we have made the following notations:

-  $N_{cx}$ ,  $N_{cy}$  represent the total number of the sensor elements in  $x$ , respectively  $y$ , direction;

-  $d_x$ ,  $d_y$  represent the center to center distances between adjacent sensor elements in  $x$ , respectively  $y$ , direction;

-  $N_{px}$ ,  $N_{py}$  represent the total number of the pixels in  $x$ , respectively  $y$  direction.

Using the relations (3.45) and (3.46) we obtain, as follows:

$$X_{pmax} = coefx \cdot X_{dmax}, \quad (3.51)$$

$$Y_{pmax} = coefy \cdot Y_{dmax}. \quad (3.52)$$

From the relations (3.47), (3.48), (3.49), (3.50), (3.51), (3.52) and the fact that the number of the lines from the chip is equal with the number of the lines from the computer image:

$$N_{cy} = N_{py}, \quad (3.53)$$

one can compute the values for the scale factors as follows:

$$coefx = \frac{1}{d_x} \frac{N_{px}}{N_{cx}}, \quad (3.54)$$

$$coefy = \frac{1}{d_y}. \quad (3.55)$$

Due to a variety of factors, such as slight hard-were timing mismatch between image acquisition hard-were and camera scanning hard-were, or the imprecision of the timing of TV scanning itself the scale factor in  $x$  direction will be different from that, computed with relation (3.54). We must correct this factor with a parameter named the uncertainty of the scale factor,  $u_{sx}$ , which can be different from 1 with maximum five-percent, reference [Tsa87]. We will obtain the next relation:

$$coefx' = u_{sx} \frac{1}{d_x} \frac{N_{px}}{N_{cx}}. \quad (3.56)$$

*Numerical example for the scale factors!*

For a Panasonic WV-CD50 camera we have the following characteristics:

- sensor area is  $8.5mm \times 6.4mm$  :

-  $d_x = 17\mu m$ ,  $d_y = 11\mu m$  ;

-  $N_{cx} = 500$ ,  $N_{cy} = 582$  .

The standard used for image acquisition is standard CCIR, which means that for the computer image we have:

-  $N_{px} = 752$ ,  $N_{py} = 582$  (standard CCIR).

Using the relations (3.54) and (3.55) and the dates presented just before one can obtain the next values:

-  $coefx = 88.471 \text{ pixel} / mm$   $coefx^{-1} = 0.01303 \text{ mm} / \text{pixel}$  :

-  $coefy = 90.91 \text{ pixel} / mm$   $coefy^{-1} = 0.011 \text{ mm} / \text{pixel}$  .

As it was said in the beginning of the sub-chapter 3.3, two methods for camera calibration will be presented here. The first is called Lenz calibration method, reference [Len87] and the second is called Tsai calibration method, reference [Tsa87]. Before starting to describe these methods we will present the camera parameters, which will be calibrated.

For both methods we must compute the position and the orientation of the camera, so the rotation matrix  $\mathbf{R}$  and the translation vector  $\mathbf{t}$  will be determined.

The focal length is a parameter for both methods. Both methods don't considering the offset of the image center, and the image center is chosen in the middle of the computer image ( $C_x = 0$ , and  $C_y = 0$ ). Also the coefficient of radial distortion is considered, as being zero ( $k = 0$ ).



We will compute the scale factors for the first method using the relations (3.54) and (3.55):

$$S_y = coefy. \quad (3.57)$$

$$S_x = coefx. \quad (3.58)$$

But, in the second method we will use also, as a parameter, the uncertainty of the  $x$  scale factor, according to the reference [Tsa87]. This parameter is denoted with  $u_{sx}$ , so the  $x$  scale factor will be computed with the next relation:

$$S'_x = coefx' = u_{sx} \cdot coefx = u_{sx} S_x. \quad (3.59)$$

The  $y$  scale factor is computed in the same way, as in the first method.

We made this choice of the camera's parameters according to the conclusions from reference [Jim93], which are the following:

- the offset of the image center has little effect on the determination of the position and orientation of a coordinate frame;
- the lens distortion will not dramatically change the position and orientation of a coordinate frame;
- the scale factor has a great effect on the position of the coordinate frame, and on the accuracy of the measurements;
- the offset of the image center is more sensitive than the lens distortion on the determination of the position and orientation of a coordinate frame.

### 3.3.2 Lenz calibration method

As we said at the end of sub-chapter 3.3.1, the camera parameters, which will be calibrated in the Lenz method, according to the reference [Len87], are:

- focal length ( $f$ ), as internal parameter;
- the rotation matrix ( $\mathbf{R}$ ), and the translation vector ( $\mathbf{t}$ ), as external parameters.

We know the coordinates  $(x_{wi}, y_{wi}, z_{wi})$ , in millimeters, for a point  $P_i$  in the world frame, and the coordinates  $(X_{pi}, Y_{pi})$ , in pixels, for its correspondent on the computer image. We also, know some specifications for our camera (the model used is CV-M50), and using

this information we can simply compute the scale factors with the relations (3.54) and (3.55). We will obtain the next values:

$$S_x = coef_x = 116.589 \text{ mm / pixel} . \quad (3.60)$$

$$S_y = coef_y = 120.248 \text{ mm / pixel} . \quad (3.61)$$

We need in our next operations the values for  $S_x^{-1}$ , and  $S_y^{-1}$ . In fact,  $S_x$ , and  $S_y$  are the scale factors from the chip image to the computer image, and  $S_x^{-1}$ , and  $S_y^{-1}$  are the scale factors from the computer image to the chip image. These values are:

$$S_x^{-1} = coef_x^{-1} = 0.008577 \text{ pixel / mm} , \quad (3.62)$$

$$S_y^{-1} = coef_y^{-1} = 0.008316 \text{ pixel / mm} . \quad (3.63)$$

The specifications for the camera model CV-M50, which gives the possibility to compute the scale factors, are the next ones:

-scanning area:  $6.45\text{mm} \times 4.84\text{mm}$  :

-CCIR standard:  $752(H) \times 582(V)$  .

Going further, this method is divided in six steps, which will be presented in the following parts.

### Step 1

Using the relations (3.45), (3.46), (3.62), and (3.63) one can find the values for  $X_{di}$ , and  $Y_{di}$ , as follows:

$$X_{di} = S_x^{-1} X_{pi} , \quad (3.64)$$

$$Y_{di} = S_y^{-1} Y_{pi} . \quad (3.65)$$

where  $X_{di}$ , and  $Y_{di}$  are the distorted coordinates of the point  $P_i$  from the chip image.

## Step 2

For this method we use a calibration board, like in figure 3.10, which contains all the calibration points  $P_i$ . The calibration board is a plate with marked points all of them being in the same plane. The world coordinate frame  $S_w$  is chosen, as one can see in the figure 3.10.

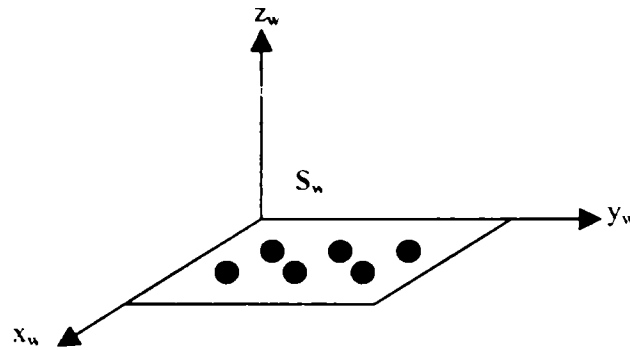


Fig. 3.10. The calibration board.

Any point  $P_i$ , from the calibration board, will have the coordinates  $(x_{wi}, y_{wi}, 0)$  and in this case the relation (1.13) becomes:

$$\frac{X_{di}}{Y_{di}} = \frac{x_{wi}r_1 + y_{wi}r_2 + t_1}{x_{wi}r_4 + y_{wi}r_5 + t_1}. \quad (3.66)$$

From the relation (2.7) one can obtain an equation, as follows:

$$Y_{di}x_{wi} \begin{pmatrix} r_1 \\ t_1 \end{pmatrix} + Y_{di}y_{wi} \begin{pmatrix} r_2 \\ t_1 \end{pmatrix} + Y_{di} \begin{pmatrix} t_3 \\ t_1 \end{pmatrix} - X_{di}x_{wi} \begin{pmatrix} r_4 \\ t_1 \end{pmatrix} - X_{di}y_{wi} \begin{pmatrix} r_5 \\ t_1 \end{pmatrix} = X_{di}. \quad (3.67)$$

For  $N$  points  $P_i$ , it will be obtained an over determined system with  $N$  equations, each of them having the same nature with the one noted (3.67). Using the matrices one can write the next relation:

$$\mathbf{C} [a_1 \ a_2 \ a_3 \ a_4 \ a_5]^T = [X_{d1} \ X_{d2} \ \dots \ X_{dN}]^T, \quad (3.68)$$

where

$$\mathbf{C} = [Y_{di}x_{wi} \ Y_{di}y_{wi} \ Y_{di} \ -X_{di}x_{wi} \ -X_{di}y_{wi}]_{i=1, \dots, N}. \quad (3.69)$$

and

$$a_1 = \frac{r_1}{t_y}, \quad a_2 = \frac{r_2}{t_y}, \quad a_3 = \frac{t_x}{t_y}, \quad a_4 = \frac{r_4}{t_y}, \quad a_5 = \frac{r_5}{t_y}. \quad (3.70)$$

The solution for this system is, according to the reference [PTVF92], the following:

$$[a_1 \ a_2 \ a_3 \ a_4 \ a_5]^T = \mathbf{C}^{-1} [X_{di} \ X_{di} \ \dots \ X_{di}]^T, \quad (3.71)$$

where  $\mathbf{C}^{-1}$  is the inverse of the matrix  $\mathbf{C}$ .

We have the next relation between the elements of the rotation matrix  $\mathbf{R}$ :

$$\left[ (r_1 + r_5)^2 + (r_2 - r_4)^2 \right]^{\frac{1}{2}} + \left[ (r_1 - r_5)^2 + (r_2 + r_4)^2 \right]^{\frac{1}{2}} = 2. \quad (3.72)$$

From the relations (3.70), and (3.71) one can find the value of the translation  $t_y$ , as follows:

$$t_y = \frac{2}{\left[ (a_1 + a_5)^2 + (a_2 - a_4)^2 \right]^{\frac{1}{2}} + \left[ (a_1 - a_5)^2 + (a_2 + a_4)^2 \right]^{\frac{1}{2}}}. \quad (3.73)$$

Knowing now the value for  $t_y$ , from (3.73), the values for  $a_1$ ,  $a_2$ ,  $a_3$ ,  $a_4$  and  $a_5$  from (3.71), and using the relation (3.70) it is possible to compute the values for  $r_1$ ,  $r_2$ ,  $t_x$ ,  $r_4$  and  $r_5$ , as follows:

$$r_1 = a_1 t_y, \quad r_2 = a_2 t_y, \quad t_x = a_3 t_y, \quad r_4 = a_4 t_y, \quad r_5 = a_5 t_y. \quad (3.74)$$

### Step 3

Using the proprieties of the elements of the rotation matrix  $\mathbf{R}$  we can find the values for  $r_7$ , and  $r_8$ :

$$r_7 = (1 - r_1^2 - r_4^2), \quad (3.75)$$

$$r_8 = -(1 - r_2^2 - r_5^2) \text{sign}(r_1 r_2 + r_4 r_5). \quad (3.76)$$

#### Step 4

In this step we will compute the focal length  $f$ , and the translation  $t_z$  from an over determined system of equations. Using the fact that the coefficient of the radial distortion was chosen zero the relations (3.33), and (3.34) become, as follows:

$$X_{ui} = X_{di}, \quad (3.77)$$

$$Y_{ui} = Y_{di}, \quad (3.78)$$

where  $(X_{ui}, Y_{ui})$  are the undistorted coordinates of the point  $P_i$  from the CCD-chip. With these last, two relations. (3.35), and (3.36) become, as follows:

$$X_{di} = -f \frac{x_i}{z_i}, \quad (3.79)$$

$$Y_{di} = -f \frac{y_i}{z_i}, \quad (3.80)$$

where  $(x_i, y_i, z_i)$  are the coordinates of the point  $P_i$  with respect to the camera system. In this moment it is possible to compute the values for  $x_i$ , and  $y_i$  using the relations (3.41), and (3.42), because the values for all the parameters which define these two relations, are known:

$$x_i = x_{wi}r_1 + y_{wi}r_2 + t_x, \quad (3.81)$$

$$y_i = x_{wi}r_4 + y_{wi}r_5 + t_y. \quad (3.82)$$

For  $z_i$  we use the relation (3.43), which in this conditions becomes:

$$z_i = x_{wi}r_7 + y_{wi}r_8 + t_z. \quad (3.83)$$

Using the last five relations one can obtain a system with two equations, where  $f$  and  $t_z$  are the unknowns. These equations are:

$$-x_i f - X_{di} t_z = x_{wi} r_7 X_{di} + y_{wi} r_8 X_{di}, \quad (3.84)$$

$$-y_i f - Y_{di} t_z = x_{wi} r_7 Y_{di} + y_{wi} r_8 Y_{di}. \quad (3.85)$$

If there are taken  $N$  points  $P_i$ , it is possible to obtain  $2N$  equations having the same form like (3.84), and (3.85). Using the matrices one can write this system of equations, as follows:

$$\begin{bmatrix} -x_1 & -X_{d1} \\ \cdot \\ -x_N & -X_{dN} \\ -y_1 & -Y_{d1} \\ \cdot \\ -y_N & -Y_{dN} \end{bmatrix} \begin{bmatrix} f \\ t_z \end{bmatrix} = \begin{bmatrix} x_{w1}r_7X_{d1} + y_{w1}r_8X_{d1} \\ \cdot \\ x_{wN}r_7X_{dN} + y_{wN}r_8X_{dN} \\ x_{w1}r_7Y_{d1} + y_{w1}r_8Y_{d1} \\ \cdot \\ x_{wN}r_7Y_{dN} + y_{wN}r_8Y_{dN} \end{bmatrix}. \quad (3.86)$$

The solution for this over determined system is, according to the reference [PTVF92], as follows:

$$\begin{bmatrix} f \\ t_z \end{bmatrix} = \begin{bmatrix} -x_1 & -X_{d1} \\ \cdot \\ -x_N & -X_{dN} \\ -y_1 & -Y_{d1} \\ \cdot \\ -y_N & -Y_{dN} \end{bmatrix}^{-1} \begin{bmatrix} x_{w1}r_7X_{d1} + y_{w1}r_8X_{d1} \\ \cdot \\ x_{wN}r_7X_{dN} + y_{wN}r_8X_{dN} \\ x_{w1}r_7Y_{d1} + y_{w1}r_8Y_{d1} \\ \cdot \\ x_{wN}r_7Y_{dN} + y_{wN}r_8Y_{dN} \end{bmatrix}. \quad (3.87)$$

### Step 5

In this step we will establish the right sign for all of the parameters determined before. For that we must do the next operations:

$$\{r_1, r_2, r_4, r_5, t_x, t_y\} = \frac{\{r_1, r_2, r_4, r_5, t_x, t_y\}}{\text{sign}\left(\frac{b}{t_z}\right)}, \quad (3.88)$$

$$\{r_7, r_8, t_z\} = \frac{\{r_7, r_8, t_z\}}{\text{sign}(t_z)}, \quad (3.89)$$

$$\{b\} = \frac{\{b\}}{\text{sign}(b)}. \quad (3.90)$$

## Step 6

We will compute here the last three parameters, which are:  $r_3$ ,  $r_6$ , and  $r_9$ . We will make use of the propriety that any two columns of the rotation matrix  $\mathbf{R}$  must be orthogonal. It means that one can write the next relation:

$$r_3i + r_6j + r_9k = \begin{vmatrix} i & j & k \\ r_1 & r_4 & r_7 \\ r_2 & r_5 & r_8 \end{vmatrix}. \quad (3.91)$$

From the relation (3.91) it is possible to compute the values for our last unknown parameters:

$$r_3 = r_4r_8 - r_5r_7, \quad (3.92)$$

$$r_6 = r_2r_7 - r_1r_8, \quad (3.93)$$

$$r_9 = r_1r_5 - r_2r_4. \quad (3.94)$$

So, after these six steps for all the camera parameters we have a mathematical expression. The next step was to make a logical algorithm, which would contain all these expressions in such an order that can be later implemented in a C program. This logical algorithm is presented in Annex A. The corresponding C program is presented in Annex B.

### 3.3.3 Tsai calibration method

The camera's parameters, which will be calibrated in Tsai method, according to the reference [Tsa87] are:

- focal length ( $f$ ), and the  $x$  scale factor by considering the uncertainty of the  $x$  scale factor ( $u_x$ ), as a camera internal parameter. reference [Tsa87]:

- the rotation matrix ( $\mathbf{R}$ ), and the translation vector ( $\mathbf{t}$ ), as external parameters.

As in the first method, we know the coordinates  $(x_{wi}, y_{wi}, z_{wi})$ , in millimeters, for a point  $P_i$  in the world frame, and the coordinates  $(X_{pi}, Y_{pi})$ , in pixels, for its correspondent on the computer image. Going further, this method is divided in five steps, which will be presented in following parts.

## Step 1

For any point  $P_i$  having the coordinates  $(X_{pi}, Y_{pi})$ , on the computer image, and the coordinates  $(X_{di}, Y_{di})$ , on the chip image, using the relation (3.45), (3.46), (3.57), (3.58), and (3.59), with the observation that in this case instead of  $coefx$  we have  $coefx'$ , one can obtain, as follows:

$$X_{pi} = coefx' \cdot X_{di} = u_{sx} S_x X_{di} , \quad (3.95)$$

$$Y_{pi} = coefy' \cdot Y_{di} = S_y X_{di} , \quad (3.96)$$

where  $S_x$  and  $S_y$  are computed in the same way as the ones computed in the sub-chapter 3.3.2, with relations (3.60), and (3.61). Obvious, because we use the same camera in both methods of calibration, the values for  $S_x$  and  $S_y$  will be the same in both cases. So, we have:

$$S_x = coefx = 116.589 \text{ mm / pixel} , \quad (3.97)$$

$$S_y = coefy = 120.248 \text{ mm / pixel} . \quad (3.98)$$

From the relation (3.95), and (3.96) one can obtain:

$$X_{di}' = S_x^{-1} X_{pi} , \quad (3.99)$$

$$Y_{di}' = S_y^{-1} Y_{pi} , \quad (3.100)$$

where

$$X_{di}' = u_{sx} X_{di} . \quad (3.101)$$

From (3.101) we can simply obtain, as follows:

$$X_{di} = u_{sx}^{-1} X_{di}' . \quad (3.102)$$



## Step 2

For this method of calibration we need a non-coplanar set of points. One can obtain this set of points using the same calibration board, as in the first method, but it will be moved to different heights in  $z$  direction, as it is shown in the figure 3.11:

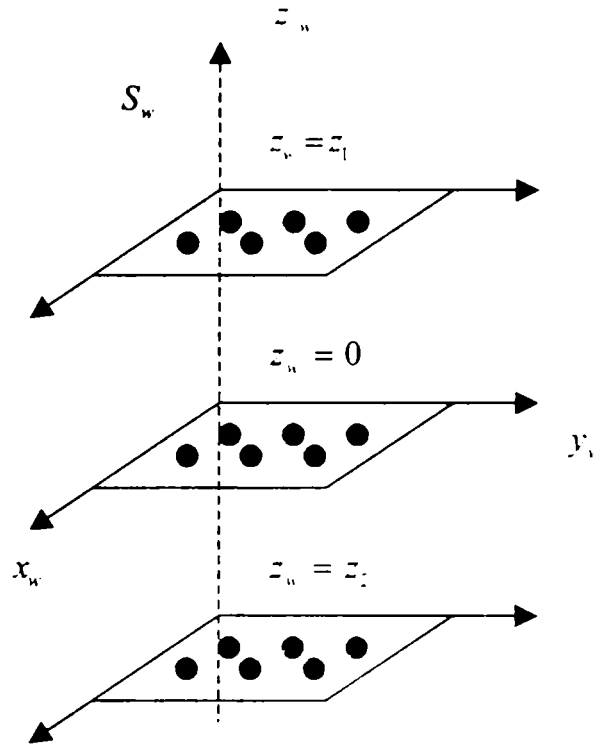


Fig. 3.11. The calibration board in three different  $z$  positions.

In this case for any calibration point  $P_i$ , which has the coordinates  $(x_{wi}, y_{wi}, z_{wi})$  with respect to the world frame  $S_w$ , the relation (3.44) becomes:

$$\frac{X_{di}}{Y_{di}} = \frac{x_{wi}r_1 + y_{wi}r_2 + z_{wi}r_3 + t_x}{x_{wi}r_4 + y_{wi}r_5 + z_{wi}r_6 + t_y} \quad (3.103)$$

Using the relations (3.102), and (3.103) we obtain the following relation:

$$\begin{aligned} & Y_{di}x_{wi}t_y^{-1}u_{sx}r_1 + Y_{di}y_{wi}t_y^{-1}u_{sx}r_2 + Y_{di}z_{wi}t_y^{-1}u_{sx}r_3 + \\ & + Y_{di}t_y^{-1}u_{sx}t_x - X'_{di}x_{wi}t_y^{-1}r_4 - X'_{di}y_{wi}t_y^{-1}r_5 - X'_{di}z_{wi}t_y^{-1}r_6 = X'_{di} \end{aligned} \quad (3.104)$$

One can make the next notations:

$$t_v^{-1} u_{\alpha} r_1 = a_1, \quad (3.105)$$

$$t_v^{-1} u_{\alpha} r_2 = a_2, \quad (3.106)$$

$$t_v^{-1} u_{\alpha} r_3 = a_3, \quad (3.107)$$

$$t_y^{-1} u_{sx} t_x = a_4, \quad (3.108)$$

$$t_v^{-1} r_4 = a_5, \quad (3.109)$$

$$t_v^{-1} r_5 = a_6, \quad (3.110)$$

$$t_y^{-1} r_6 = a_7. \quad (3.111)$$

With these notations the relation (3.104) becomes, as follows:

$$Y_{di} x_{wi} a_1 + Y_{di} y_{wi} a_2 + Y_{di} z_{wi} a_3 + Y_{di} a_4 - X'_{di} x_{wi} a_5 - X'_{di} y_{wi} a_6 - X'_{di} z_{wi} a_7 = X'_{di}. \quad (3.112)$$

For  $N$  points  $P_i$ , one can obtain an over determined system with  $N$  equations having the same nature with the one noted (3.112). Using the matrices one can write the next relation:

$$\mathbf{C} [a_1 \ a_2 \ a_3 \ a_4 \ a_5 \ a_6 \ a_7]^T = [X'_{d1} \ X'_{d2} \ \dots \ X'_{dN}]^T, \quad (3.113)$$

where

$$\mathbf{C} = [Y_{di} x_{wi} \ Y_{di} y_{wi} \ Y_{di} z_{wi} \ Y_{di} \ -X'_{di} x_{wi} \ -X'_{di} y_{wi} \ -X'_{di} z_{wi}]_{i=1, \dots, N}. \quad (3.114)$$

The solution for this system is, according to the reference [PTVF92], the following:

$$[a_1 \ a_2 \ a_3 \ a_4 \ a_5 \ a_6 \ a_7]^T = \mathbf{C}^{-1} [X'_{d1} \ X'_{d2} \ \dots \ X'_{dN}]^T, \quad (3.115)$$

where  $\mathbf{C}^{-1}$  is the inverse of the matrix  $\mathbf{C}$ .

### Step 3

In this step the goal is to compute  $r_1, r_2, r_3, r_4, r_5, r_6, t_x$ , and  $t_y$  knowing the values for  $a_1, a_2, a_3, a_4, a_5, a_6$ , and  $a_7$  which were computed in the step 2.

In the next derivation it is used the fact that the norm of any rows or columns of the rotation matrix  $\mathbf{R}$  is equal with one. So, one can write the next two relations:

$$r_1^2 + r_2^2 + r_3^2 = 1, \quad (3.116)$$

$$r_4^2 + r_5^2 + r_6^2 = 1. \quad (3.117)$$

From the relations (3.109), (3.110), (3.111), and (3.117) we obtain:

$$|t_y| = (a_5^2 + a_6^2 + a_7^2)^{-\frac{1}{2}}. \quad (3.118)$$

We don't know the sign for  $t_y$ . This problem will be solved a little later.

From the relations (3.105), (3.106), ..., (3.111) one can obtain:

$$u_{xy} = (a_1^2 + a_2^2 + a_3^2)^{\frac{1}{2}} (a_5^2 + a_6^2 + a_7^2)^{-\frac{1}{2}}. \quad (3.119)$$

Now it's time to find the right sign for  $t_y$ . The procedure is, as follows:

- a) we take a point  $P_i(x_{wi}, y_{wi}, z_{wi})$  whose computer image coordinates  $(X_{pi}, Y_{pi})$  are far away from the image center:
- b) we chose +1 as being the sign of  $t_y$ ;
- c) we compute  $r_1, r_2, r_3, r_4, r_5, r_6$ , and  $t_x$  from the relations (3.105), (3.106), ..., (3.111). We will obtain the following:

$$r_1 = a_1 u_{xx}^{-1} t_x, \quad (3.120)$$

$$r_2 = a_2 u_{yy}^{-1} t_x, \quad (3.121)$$

$$r_3 = a_3 u_{xz}^{-1} t_x, \quad (3.122)$$

$$t_x = a_4 u_{xx}^{-1} t_y, \quad (3.123)$$

$$r_4 = a_5 t_y, \quad (3.124)$$

$$r_5 = a_6 t_y, \quad (3.125)$$

$$r_6 = a_7 t_y; \quad (3.126)$$

d) using the relations (3.42), and (3.43) it is possible to compute  $x_i$ , and  $y_i$  as follows:

$$x_i = x_{wi}r_1 + y_{wi}r_2 + z_{wi}r_3 + t_x, \quad (3.127)$$

$$y_i = x_{wi}r_4 + y_{wi}r_5 + z_{wi}r_6 + t_y; \quad (3.128)$$

e) IF (( $x_i$  and  $X_{pi}$  have not the same sign) and ( $y_i$  and  $Y_{pi}$  have not the same sign))  
then  $t_x = |t_x|$

f) ELSE  $t_y = -|t_y|$ ;

*Note!*

If the sign for  $t_y$  is equal to  $-1$  we must also change the signs for  $r_1, r_2, r_3, r_4, r_5, r_6$ , and  $t_x$  in the relations (3.120), (3.121), ..., (3.126). If the sign for  $t_y$  is equal to  $+1$  the values computed for  $r_1, r_2, r_3, r_4, r_5, r_6$ , and  $t_x$  remain unchanged.

#### Step 4

In this step we will compute the values for  $r_7, r_8$ , and  $r_9$ . The procedure is the same as the one used in step 6 from Lenz method. We will use the fact that the cross product between the first row and the second row of the rotation matrix  $\mathbf{R}$  must be equal to the third row:

$$r_7i + r_8j + r_9k = \begin{vmatrix} i & j & k \\ r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \end{vmatrix}. \quad (3.129)$$

The values for  $r_7, r_8$ , and  $r_9$  are:

$$r_7 = r_2r_6 - r_3r_5, \quad (3.130)$$

$$r_8 = r_3r_4 - r_1r_6, \quad (3.131)$$

$$r_9 = r_1r_5 - r_2r_4. \quad (3.132)$$

## Step 5

Here, we will compute the focal length  $f$ , and the translation  $t_z$ . This step is almost the same with the step 4 from Lenz method. The single difference is that here  $z_{wi}$  is not all the time equal with zero. We remind that the lens distortion was not considered. In this case for any point  $P_i(x_{wi}, y_{wi}, z_{wi})$  which has the coordinates  $(X_{pi}, Y_{pi})$  on the computer image, and the coordinates  $(X_{di}, Y_{di})$  on the chip image, the relations (3.33), and (3.34) become:

$$X_{ui} = X_{di}, \quad (3.133)$$

$$Y_{ui} = Y_{di}. \quad (3.134)$$

With these two last relations, (3.35), and (3.36) will be written, as follows:

$$X_{di} = -f \frac{x_i}{z_i}, \quad (3.135)$$

$$Y_{di} = -f \frac{y_i}{z_i}, \quad (3.136)$$

where  $(x_i, y_i, z_i)$  are the coordinates of the point  $P_i$  with respect to the camera system. In this moment it is possible to compute the values for  $x_i$ , and  $y_i$  using the relations (3.41), and (3.42), because the values for all the parameters which define these two relations are known:

$$x_i = x_{wi}r_1 + y_{wi}r_2 + z_{wi}r_3 + t_x, \quad (3.137)$$

$$y_i = x_{wi}r_4 + y_{wi}r_5 + z_{wi}r_6 + t_y. \quad (1.138)$$

For  $z_i$  we use the relation (3.43), which in this conditions becomes:

$$z_i = x_{wi}r_7 + y_{wi}r_8 + z_{wi}r_9 + t_z. \quad (3.139)$$

Using the last five relations one can obtain a system with two equations, where  $f$  and  $t_z$  are the unknowns. These equations are:

$$-x_i f - X_{di} t_z = x_{wi} r_7 X_{di} + y_{wi} r_8 X_{di} + z_{wi} r_9 X_{di}, \quad (3.140)$$

$$-y_i f - Y_{di} t_z = x_{wi} r_7 Y_{di} + y_{wi} r_8 Y_{di} + z_{wi} r_9 Y_{di}. \quad (3.141)$$

If  $N$  points  $P_i$  are taken, it is possible to write  $2N$  equations having the same type like (3.84), and (3.85). Using the matrices one can write the system of equations obtain for  $N$  points  $P_i$ , as follows:

$$\begin{bmatrix} -x_1 & -X_{d1} \\ \cdot & \cdot \\ -x_N & -X_{dN} \\ -y_1 & -Y_{d1} \\ \cdot & \cdot \\ -y_N & -Y_{dN} \end{bmatrix} \begin{bmatrix} f \\ t_z \end{bmatrix} = \begin{bmatrix} x_{w1}r_7X_{d1} + y_{w1}r_8X_{d1} + z_{w1}r_9X_{d1} \\ \cdot \\ x_{wN}r_7X_{dN} + y_{wN}r_8X_{dN} + z_{wN}r_9X_{dN} \\ x_{w1}r_7Y_{d1} + y_{w1}r_8Y_{d1} + z_{w1}r_9Y_{d1} \\ \cdot \\ x_{wN}r_7Y_{dN} + y_{wN}r_8Y_{dN} + z_{wN}r_9Y_{dN} \end{bmatrix}. \quad (3.142)$$

The solution for this over determined system is, according to the reference [PTVF92], as follows:

$$\begin{bmatrix} f \\ t_z \end{bmatrix} = \begin{bmatrix} -x_1 & -X_{d1} \\ \cdot & \cdot \\ -x_N & -X_{dN} \\ -y_1 & -Y_{d1} \\ \cdot & \cdot \\ -y_N & -Y_{dN} \end{bmatrix}^{-1} \begin{bmatrix} x_{w1}r_7X_{d1} + y_{w1}r_8X_{d1} + z_{w1}r_9X_{d1} \\ \cdot \\ x_{wN}r_7X_{dN} + y_{wN}r_8X_{dN} + z_{wN}r_9X_{dN} \\ x_{w1}r_7Y_{d1} + y_{w1}r_8Y_{d1} + z_{w1}r_9Y_{d1} \\ \cdot \\ x_{wN}r_7Y_{dN} + y_{wN}r_8Y_{dN} + z_{wN}r_9Y_{dN} \end{bmatrix}. \quad (3.143)$$

So, after these five steps for all the camera parameters we have a mathematical expression. The next step is to make a logical algorithm, which will contain all these expressions in such an order that can be later implemented in a C program. This logical algorithm is presented in Annex C. The corresponding C program is presented in Annex D.

The big difference between Tsai method, and Lenz method is that in Tsai method the  $x$  scale factor is very precisely known by considering the uncertainty of  $x$  scale factor, as a parameter, which is computed in this method, reference [Tsa87].

Another difference is that in the second method we need a non-coplanar set of calibration points, which could be a big disadvantage in the practical situation where a camera calibration is needed.

### 3.3.4 Contributions at the simulation and analysis of the errors

After the calibration process is made for the same camera using, both methods presented in 3.3.2 and 3.3.3, one set of camera's parameters is obtained, for each method. Normally, if there aren't any errors in the calibration process the two sets of parameters must have the same values. This is true, as we will see later in figure 3.13 and figure 3.14. When the camera is calibrated we need the values for all the 3D coordinates of the calibration points and the values for the 2D coordinates of their correspondent points in the computer image. The values for the scale factors, for the coordinates of the image center and for the coefficient of the lens distortion are computed as it was explained before in this chapter.

In the practical part we analyzed the effect of three error types. The first type is referring to the errors that can appear on the 3D coordinates of the calibration points. We will consider these errors less than 2 mm. The second type is referring to the errors that can appear on the pixel coordinates of the calibration points from the image after the image processing is finished. These errors will be taken less than 2 pixels. The last error type, which is consider, is the error of the  $x$  scale factor, whose real value can be different to the computed value from the camera's specifications. This error will be considered less than 0.5%.

Going further, for the same set of errors we make the calibration process, using both methods and we obtain two different sets of camera's parameters. This procedure will be repeated for 50 different sets of errors. The next problem is to establish, which set of parameters is better. For that we need to simulate a camera measurement, made with the parameters obtained after each calibration process.

In the following part an explanation of our procedure to simulate the camera measurements will be presented. We chose a point, whose real position in the world system is known. With one set of camera's parameters we must simulate a camera measurement and find the position of this point in the world system. The difference between the real position of the point and his measured position will show which of our two calibration methods is better. We will name this difference the error of the position vector of a point. The next step is to find a mathematical relation in order to be able to compute this error.

In figure 3.12,  $S_w$  is the real world system, and  $S_c$  is the real camera system. If there are no errors in the calibration process we will obtain the same position and orientation for the camera system, as in the reality. If the errors exist in the calibration process we will obtain a new position and orientation of the camera system with respect two the reference system. Mathematically, we can consider the new camera system the same, as the real one, and we

will have then, another world system, a computed world system, noted  $S_{wc}$ , as one can see in figure 3.12.

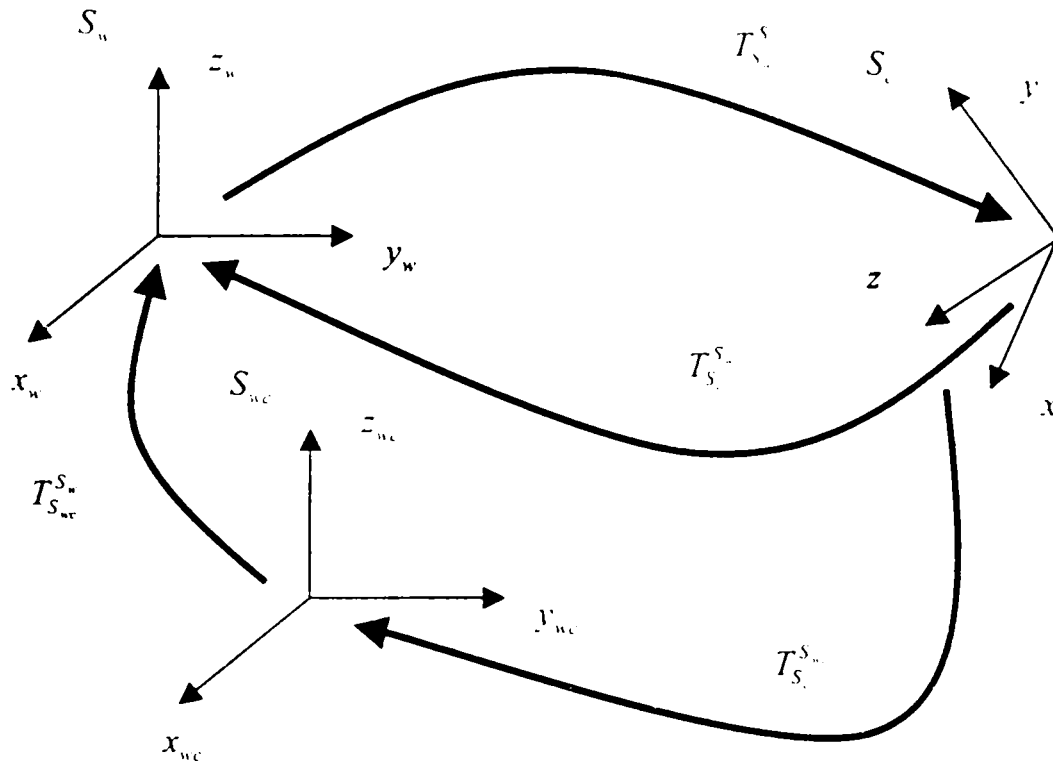


Fig. 3.12. The simulation of the camera measurement.

We will write now some mathematical relations. We take from the calibration points a point, which has the coordinates  $(x_w, y_w, z_w)$  in the real world system. We measure the position of this point and we obtain the same coordinates, but in the computed world system. We note with  $(x_{wc}, y_{wc}, z_{wc})$  the coordinates of this point in the world system. So, when we measure the position of this point we will obtain instead of  $(x_w, y_w, z_w)$ , the coordinates  $(x_{wc}, y_{wc}, z_{wc})$ .

Now, we can define the error of the position vector as follows:

$$v = \left( (x_{wc} - x_w)^2 + (y_{wc} - y_w)^2 + (z_{wc} - z_w)^2 \right)^{\frac{1}{2}}. \quad (3.144)$$

Between the coordinates  $(x_w, y_w, z_w)$ , and the coordinates  $(x_{wc}, y_{wc}, z_{wc})$  it is possible to write the next relation:

$$\begin{bmatrix} x_{wc} \\ y_{wc} \\ z_{wc} \end{bmatrix} = T_{S_w}^{S_{wc}} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}, \quad (3.145)$$



where  $T_{S_n}^{S_n}$ , is the transformation from the computed world system to the real world system.

The results of our tests are presented in figure 3.13, respectively 3.14. Both graphics show the final error of the position vector for different sets of errors. The graphic from the figure 3.13 was obtain in the situation when the calibration process was made without pixel errors and the 3D errors of the calibration points were situated between 0 mm and 2 mm.

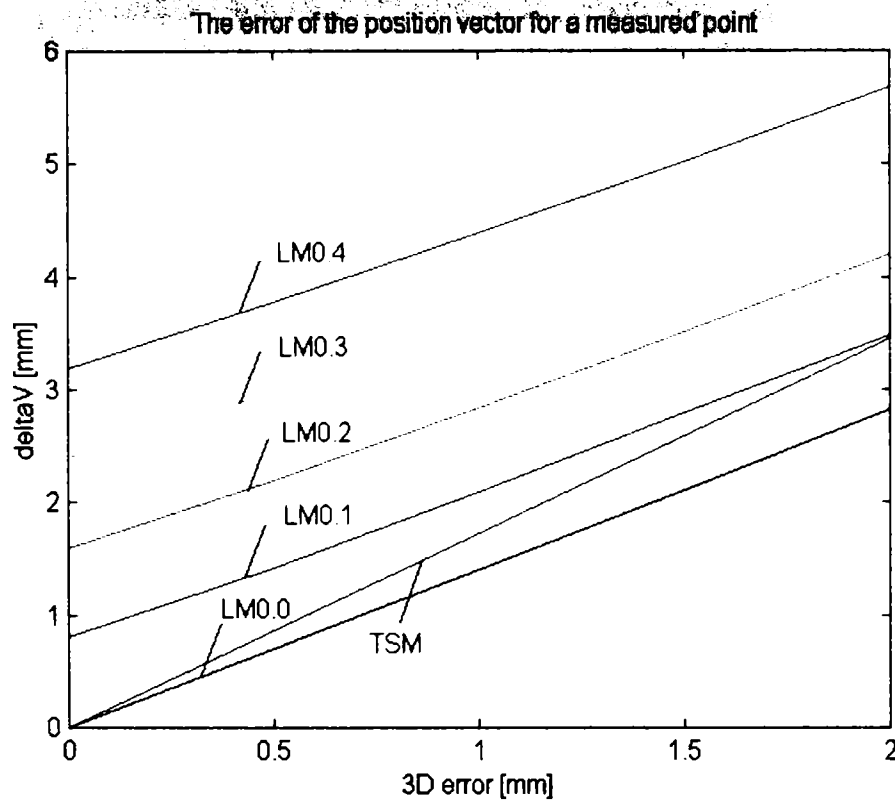


Fig. 3.13. The analysis of the results in the presence of 3D errors of the calibration points.

The graphic from the figure 3.14 was obtain in the situation when the calibration process was made without 3D errors and the pixel errors of the identified calibration points were situated between 0 pixels and 2 pixels. In the references [Tom00], and [TIN00] one can find a complex analysis also for all the possible combinations between the 3D errors and the pixel errors.

For the error of the  $x$  scale factor we considered five values: 0%, 0.1%, 0.2%, 0.3%, and 0.4%. Tsai calibration method computes the uncertainty of the  $x$  scale factor, which will completely eliminate the influence of this error. So, for this calibration method the error of the  $x$  scale factor has no effect on the error of the position vector of a measured point. One can see in both graphics the line, having the label TSM, which represents the error of the position vector when the camera calibration is made using the second method. For that there is only one line in each graphic for this method. When Lenz Method is used the error of the  $x$  scale factor has a great effect on the error of the position vector of a measured point, as one can see

in both graphics. The line, which has the label LM0.0, represents the error of the position vector when the calibration is made with a 0.0% error of the  $x$  scale factor. For the line with the label LM0.1 the error is 0.1%, for the line with the label LM0.2 the error is 0.2%, for the line with the label LM0.3 the error is 0.3% and for the line with the label LM0.4 the error is 0.4%.

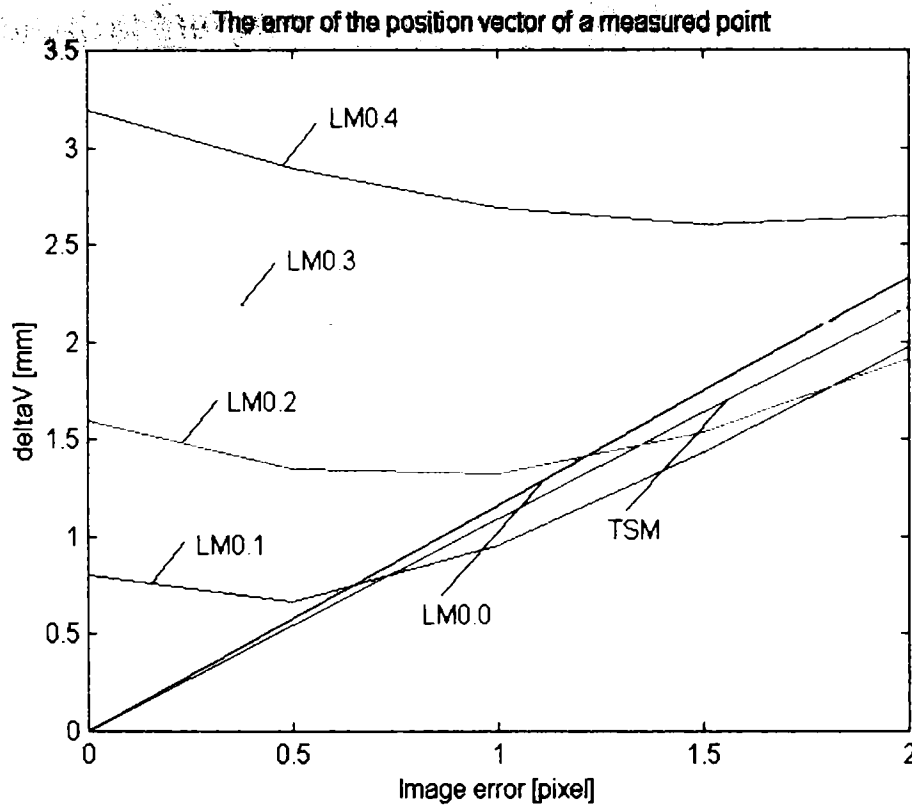


Fig. 3.14. The analysis of the results in the presence of pixel errors of the identified calibration points.

Having these explanations is obvious to see that the error of the  $x$  scale factor has a great influence on the accuracy of the measurements when the first calibration method is used. An error of only 0.4% of this scale factor will produce errors around 3 mm for the position vector when there aren't any other errors. If the 3D errors are added the error of the position vector increases to almost 6 mm. It's interesting to see that the pixels errors has practically no effect when the error of the  $x$  scale factor is greater than 0.2%.

The conclusion of our analysis is that Tsai method for camera calibration is better than Lenz method because using Tsai method one can compute an more exact value for the  $x$  scale factor. The errors introduced if an approximate value for this parameter is used, are becoming insignificant, for Tsai calibration method, because the approximate value of the scale factor will be corrected with a computed factor, called by Tsai uncertainty of the  $x$  scale factor.

## Stereo Sensor: Improvements and Contributions

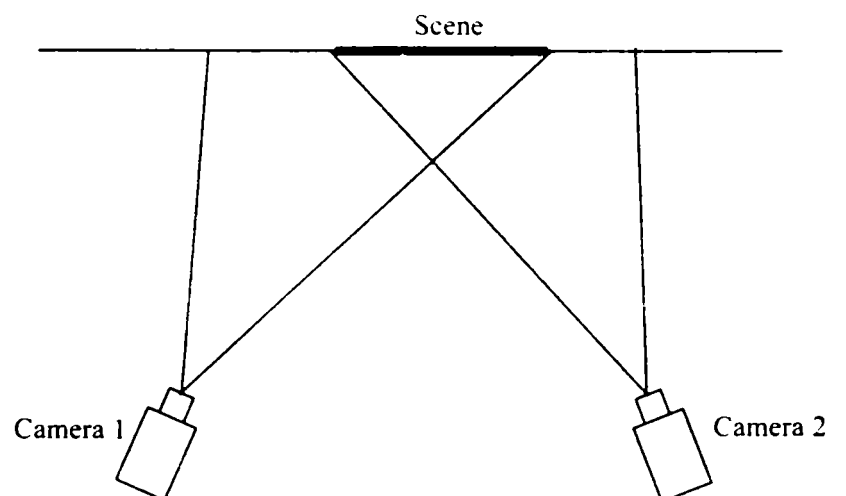
### 4.1 Mathematical Models

In this sub-chapter it is presented first the description of the stereo sensor. Then, two possible configurations of the stereo sensor are presented. The description of the camera model used by us ends this sub-chapter.

#### 4.1.1 Description of the stereo sensor

The stereo principle is well known and generally, means to look to the same scene with two cameras. Using the information given by the pictures made with these cameras it is possible to rebuild the scene without knowing any information about it. The dimensions of the scene and the distance to the scene are the main causes that influence the construction of the stereo sensor.

If we have a scene situated at a big distance relative to the cameras then in order to obtain optimal 3D information it is recommended also to place the cameras at a big distance between them as one can see in figure 4.1. In this situation we have another constraint namely to orient the cameras in such a way that both cameras will see the same scene. So, a parallel configuration is not possible in this case.



*Fig. 4.1. Stereo configuration for big distance.*

If the scene is situated at small distance then both non-parallel and parallel configurations are possible, as we will see in 4.1.2 and 4.1.3.

### 4.1.2 Non-parallel configuration

As one can see in figure 4.2, the stereo sensor is built from two cameras mounted in a metallic box. This box has two functions. One function is to realize a good fixation between the cameras themselves. The other function is to offer the possibility to mount the stereo sensor in both calibration and application environments.



Fig. 4.2. Stereo sensor in non-parallel configuration.

The distance between the cameras measured from the optical center of the camera left to the optical center of the camera right is about 25 cm. The angle between the cameras is about 45 degrees. This angle is defined by the two principal axes of the cameras. The measurement space for a fixed stereo sensor is situated at a distance between 200 mm and 300 mm from the stereo sensor. The visual field of the stereo sensor has the dimensions 200 x 200 mm at a distance of 200 mm.

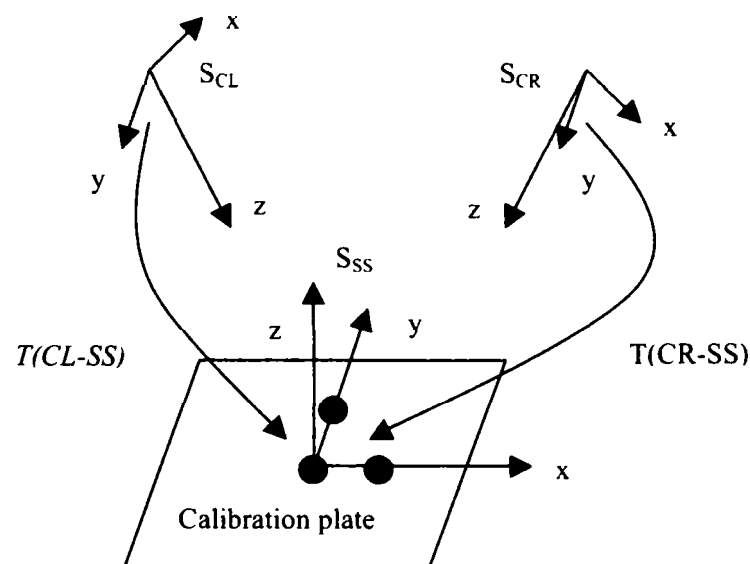


Fig. 4.3. Non-parallel configuration – coordinates frames.

In figure 4.3 one can see the three coordinate frames, which are defined for a stereo sensor. There are two camera frames, denoted  $S_{CL}$  for camera left and  $S_{CR}$  for camera right. There is also another frame defined for the stereo sensor. We will call it *stereo sensor frame* and denote it  $S_{SS}$ . We have also represented in figure 4.3 the transformations from both camera frames to the stereo sensor frame. We will use them in the sub-chapter 4.2.

### 4.1.3 Parallel configuration

In figure 4.4 one can see the stereo sensor in the parallel configuration. The cameras are mounted as near as possible one to the other having parallel directions.

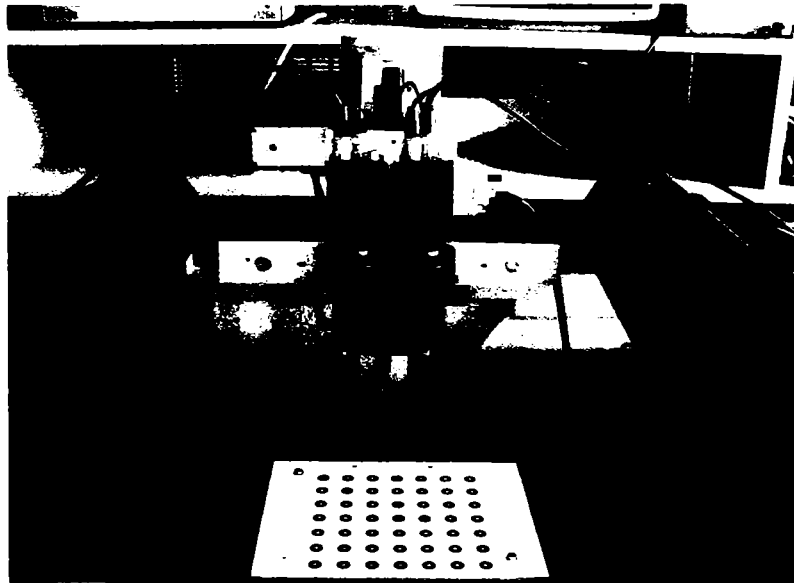


Fig. 4.4. Stereo sensor in parallel configuration.

Of course the coordinates frames are the same as for the non-parallel configuration, the only difference is that the  $z$  directions of all the three frames are parallel.

### 4.1.4 Description of the camera model

To define a camera model means to find a set of parameters, which simulate as good as possible the behavior of a real camera. Generally, the camera parameters are divided in two categories: extrinsic parameters and intrinsic parameters [Fau93].

About the camera extrinsic parameters the situation is clear there are six parameters. We denote them  $t_x, t_y, t_z, \alpha, \beta, \gamma$ . The first three give the position and the last three the orientation of the camera frame with respect to a reference frame or a world frame. In our case we called this reference frame the stereo sensor frame. The position of the stereo sensor frame is in the middle of the calibration plate and the orientation is as one can see in figure

4.3. The axes  $x$  and  $y$  are in the same plane with the calibration plate and the axis  $z$  is orthogonal to this plane.

Concerning the camera intrinsic parameters the situation is a little bit more complicated. The simplest model is the pinhole model, as it was presented in chapter 3.1.1. This model is a distortion-free model and includes four independent parameters:  $s_x f$ ,  $s_y f$ ,  $C_x$ ,  $C_y$ , where we denote with  $f$ , the focal length, with  $s_x$ ,  $s_y$ , the scale factors and with  $C_x$ ,  $C_y$  the center of the image (the intersection of the optical axis with the CCD chip plane). A better simulation of a real camera is given by the model, which includes the radial distortion. We denote the coefficient of the radial distortion with  $k$ . There are camera models, which include also other types of distortions, decentering and thin prism distortion [WCH92]. Theoretically, we should also consider the skew factor [Fau93]. The *skew factor* is a function of the angle between the axes defined by two adjacent sides of the CCD chip. Normally, this angle is 90 degrees and then the skew factor will have no influence to the projective matrix. Other intrinsic parameters can be introduced to model the fact that the optical axis is not orthogonal to the CCD chip. This is one of next problems to be solved in our future work.

We considered that in order to reach the required accuracy it is enough to consider a model, which includes the four classical intrinsic parameters  $s_x f$ ,  $s_y f$ ,  $C_x$ ,  $C_y$  and the coefficient of the radial distortion  $k$ . Because of the technological progresses in building lenses and CCD chips the effect of distortions, other than the radial distortion, and the effect of the skew factor are very small. We used for the radial distortion the same model as it was presented in the sub-chapter 3.1.2.

## 4.2 Contributions at the Calibration Procedure

As it was explained in the chapters before, if we want metric information we have to know both internal and external camera parameters. The process of computing all the camera parameters is called camera calibration. In the following part we will describe first the calibration device we used and then the calibration procedure developed, underlining the contributions to this procedure.

### 4.2.1 Description of the calibration device

In figure 4.5, one can see the calibration plate. This was made from glass, in order to reduce the modifications, which can appear because of the temperature variation. The uncertainty of the circle positions is between  $-0.01$  mm and  $+0.01$  mm.

One can also see, from the figure 4.5, that the calibration plate is fixed on a special device. This device can provide movements in three orthogonal directions ( $x, y, z$ ) with an uncertainty situated between  $-0.01\text{mm}$  and  $+0.01\text{mm}$ . The alignment between the special device frame and the calibration plate frame is done mechanically and is adjusted and controlled using Leica 3D measurement system with an uncertainty of  $0.01\text{mm}$ . Finally, the total uncertainty of the position of the circles is situated between  $-0.025\text{mm}$  and  $+0.025\text{mm}$ .

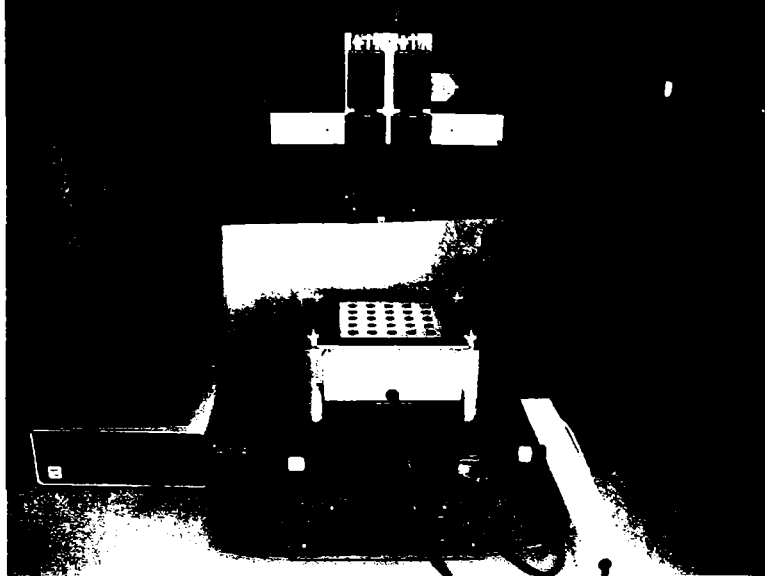


Fig. 4.5. The calibration device.

#### 4.2.2 Description of the calibration procedure

As we saw in sub-chapter 4.2.1, using the calibration device, we are able to generate 3D points whose coordinates are known very precisely. In chapter 5 we will describe a procedure, which will allow us to find also very precisely the 2D coordinates of corresponding 3D points in the image. Knowing these 3D and 2D coordinates of a set of points we will be able to compute the camera parameters.

We begin the description of the calibration procedure by finding the relation between the 3D coordinates and the 2D pixel coordinates of a point and the camera parameters. We start from the next equations:

$$\frac{1}{1+k\left(\frac{(X_p - C_x)^2}{s_x^2} + \frac{(Y_p - C_y)^2}{s_y^2}\right)} \frac{X_p - C_x}{s_x} = f \frac{x}{z}, \quad (4.1)$$

$$\frac{1}{1+k\left(\frac{(X_p - C_x)^2}{s_x^2} + \frac{(Y_p - C_y)^2}{s_y^2}\right)} \frac{Y_p - C_y}{s_y} = f \frac{y}{z}, \quad (4.2)$$

where  $(X_p, Y_p)$  are the pixel coordinates and  $(x, y, z)$  are the 3D coordinates of a calibration point with respect to the camera frame. Between the 3D coordinates of a calibration point with respect to the camera frame and the 3D coordinates of the same point but with respect to the world frame one can write the next relation:

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix}^T = {}^{cam}_{w} \mathbf{T} \cdot \begin{bmatrix} x_w & y_w & z_w & 1 \end{bmatrix}^T, \quad (4.3)$$

where  $(x_w, y_w, z_w)$  are the 3D coordinates of the calibration point with respect to the world frame.

The transformation from the camera frame to the world frame can be written as a function of  $t_x, t_y, t_z, \alpha, \beta$  and  $\gamma$  as follows:

$${}^{cam}_{w} \mathbf{T} = \begin{bmatrix} \cos \gamma \cos \beta & \cos \gamma \sin \beta \sin \alpha - \sin \gamma \cos \alpha & \cos \gamma \sin \beta \cos \alpha + \sin \gamma \sin \alpha & t_x \\ \sin \gamma \sin \beta & \sin \gamma \sin \beta \sin \alpha + \cos \gamma \cos \alpha & \sin \gamma \sin \beta \cos \alpha - \cos \gamma \sin \alpha & t_y \\ -\sin \beta & \cos \beta \sin \alpha & \cos \beta \cos \alpha & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This relation is denoted (4.4). From the relations (4.3) and (4.4) we obtain, the next three relations, denoted (4.5), (4.6) and (4.7):

$$\begin{aligned} x &= \cos \gamma \cos \beta x_w + (\cos \gamma \sin \beta \sin \alpha - \sin \gamma \cos \alpha) y_w + (\cos \gamma \sin \beta \cos \alpha + \sin \gamma \sin \alpha) z_w + t_x \\ y &= \sin \gamma \cos \beta x_w + (\sin \gamma \sin \beta \sin \alpha + \cos \gamma \cos \alpha) y_w + (\sin \gamma \sin \beta \cos \alpha + \cos \gamma \sin \alpha) z_w + t_y \\ z &= -\sin \beta x_w + \cos \beta \sin \alpha y_w + \cos \beta \cos \alpha z_w + t_z \end{aligned}$$

Starting from the equation (4.1) one can write, as follows:

$$(X_p - C_x)z - s_x f \left( 1 + k \left( \frac{(X_p - C_x)^2}{s_x^2} + \frac{(Y_p - C_y)^2}{s_y^2} \right) \right) x = 0. \quad (4.8)$$

With the notations:  $p_x = s_x f$ , and  $d = kf^2$  the relation (4.8) becomes:

$$(X_p - C_x)z - p_x \left( 1 + d \left( \frac{(X_p - C_x)^2}{p_x^2} + \frac{(Y_p - C_y)^2}{p_y^2} \right) \right) x = 0 \quad (4.9)$$



For each calibration point one can write the relation (4.9). This relation is an equation having the coefficients determined by the 3D coordinates and the 2D pixel coordinates of the calibration point  $P$ . One can make the following notation:

$$F_{x_p, y_p, x_w, y_w, z_w}^x(\alpha, \beta, \gamma, t_x, t_z, p_x, p_y, C_x, C_y, d) = 0. \quad (4.10)$$

If we use  $N$  ( $N > 10$ ) calibration points we will obtain an over-determined system of equations. To solve this system we use the Newton Algorithm [Man81], [Nas99], [Lip01] and [RM01]. First, we must make this nonlinear system to be linear. According to Newton Algorithm starting from relation (4.10) one can write the next relation:

$$\begin{aligned} F_{x_0} + \frac{\partial F_{x_0}}{\partial \alpha}(\alpha - \alpha_0) + \frac{\partial F_{x_0}}{\partial \beta}(\beta - \beta_0) + \frac{\partial F_{x_0}}{\partial \gamma}(\gamma - \gamma_0) + \frac{\partial F_{x_0}}{\partial t_x}(t_x - t_{x_0}) + \\ + \frac{\partial F_{x_0}}{\partial t_z}(t_z - t_{z_0}) + \frac{\partial F_{x_0}}{\partial p_x}(p_x - p_{x_0}) + \frac{\partial F_{x_0}}{\partial p_y}(p_y - p_{y_0}) + \frac{\partial F_{x_0}}{\partial C_x}(C_x - C_{x_0}) + \\ + \frac{\partial F_{x_0}}{\partial C_y}(C_y - C_{y_0}) + \frac{\partial F_{x_0}}{\partial d}(d - d_0) = 0 \end{aligned} \quad (4.11)$$

where  $t_{x_0}$ ,  $t_{z_0}$ ,  $\alpha_0$ ,  $\beta_0$ ,  $\gamma_0$ ,  $p_{x_0}$ ,  $p_{y_0}$ ,  $C_{x_0}$ ,  $C_{y_0}$  and  $d_0$  are initial values for the camera parameters.

$F_x$ ,  $F_{x_0}$  and  $\frac{\partial F_{x_0}}{\partial v}$  ( $v$  is anyone from  $t_x$ ,  $t_z$ ,  $\alpha$ ,  $\beta$ ,  $\gamma$ ,  $p_x$ ,  $p_y$ ,  $C_x$ ,  $C_y$  and  $d$ ) are given by the next

three relations:

$$F_x = F_{x_p, y_p, x_w, y_w, z_w}^x(\alpha, \beta, \gamma, t_x, t_z, p_x, p_y, C_x, C_y, d), \quad (4.12)$$

$$F_{x_0} = F_{x_p, y_p, x_w, y_w, z_w}^x(\alpha_0, \beta_0, \gamma_0, t_{x_0}, t_{z_0}, p_{x_0}, p_{y_0}, C_{x_0}, C_{y_0}, d_0), \quad (4.13)$$

$$\frac{\partial F_{x_0}}{\partial v} = \frac{\partial F_x}{\partial v}(\alpha_0, \beta_0, \gamma_0, t_{x_0}, t_{z_0}, p_{x_0}, p_{y_0}, C_{x_0}, C_{y_0}, d_0). \quad (4.14)$$

Going further we will write the explicit relations for  $\frac{\partial F_x}{\partial v}$ :

$$\begin{aligned} \frac{\partial F_x}{\partial \alpha} &= (\cos \beta \cos \alpha \cdot y_w - \cos \beta \sin \alpha \cdot z_w)(X_p - C_x) - \\ &- (\cos \gamma \sin \beta \cos \alpha \cdot y_w + \sin \gamma \sin \alpha \cdot y_w - \cos \gamma \sin \beta \sin \alpha \cdot z_w + \\ &+ \sin \gamma \cos \alpha \cdot z_w) p_x \left( 1 + d \left( \frac{(X_p - C_x)^2}{p_x^2} + \frac{(Y_p - C_y)^2}{p_y^2} \right) \right) \end{aligned} \quad (4.15)$$

$$\begin{aligned} \frac{\partial F_x}{\partial \beta} &= (-\cos \beta \cdot x_w - \sin \beta \sin \alpha \cdot y_w - \sin \beta \cos \alpha \cdot z_w)(X_p - C_x) - \\ &- (-\cos \gamma \sin \beta \cdot x_w + \cos \gamma \cos \beta \sin \alpha \cdot y_w + \\ &+ \cos \gamma \cos \beta \cos \alpha \cdot z_w) p_x \left( 1 + d \left( \frac{(X_p - C_x)^2}{p_x^2} + \frac{(Y_p - C_y)^2}{p_y^2} \right) \right) \end{aligned} \quad (4.16)$$

$$\begin{aligned} \frac{\partial F_x}{\partial \gamma} &= -(-\sin \gamma \cos \beta \cdot x_w - \sin \gamma \sin \beta \sin \alpha \cdot y_w - \cos \gamma \cos \alpha \cdot y_w - \\ &- \sin \gamma \sin \beta \cos \alpha \cdot z_w + \cos \gamma \sin \alpha \cdot z_w) p_x \left( 1 + d \left( \frac{(X_p - C_x)^2}{p_x^2} + \frac{(Y_p - C_y)^2}{p_y^2} \right) \right) \end{aligned} \quad (4.17)$$

$$\frac{\partial F_x}{\partial t_x} = -p_x \left( 1 + d \left( \frac{(X_p - C_x)^2}{p_x^2} + \frac{(Y_p - C_y)^2}{p_y^2} \right) \right), \quad (4.18)$$

$$\frac{\partial F_x}{\partial t_z} = X_p - C_x, \quad (4.19)$$

$$\frac{\partial F_x}{\partial p_x} = - \left( \left( 1 + d \left( \frac{(X_p - C_x)^2}{p_x^2} + \frac{(Y_p - C_y)^2}{p_y^2} \right) \right) - \frac{1}{2} d \frac{(X_p - C_x)^2}{p_x^2} \right) x, \quad (4.20)$$

$$\frac{\partial F_x}{\partial p_y} = \frac{1}{2} p_x d \frac{(Y_p - C_y)^2}{p_y^3} x, \quad (4.21)$$

$$\frac{\partial F_x}{\partial C_x} = -z + 2d \frac{X_p - C_x}{p_x} x. \quad (4.22)$$

$$\frac{\partial F_x}{\partial C_y} = 2p_x d \frac{Y_p - C_y}{p_y^2} x. \quad (4.23)$$

$$\frac{\partial F_x}{\partial d} = -p_x \left( \frac{(X_p - C_x)^2}{p_x^2} + \frac{(Y_p - C_y)^2}{p_y^2} \right) x. \quad (4.24)$$

The relation (4.11) can be written in the following form:

$$\begin{aligned} & \frac{\partial F_{x0}}{\partial \alpha} \alpha + \frac{\partial F_{x0}}{\partial \beta} \beta + \frac{\partial F_{x0}}{\partial \gamma} \gamma + \frac{\partial F_{x0}}{\partial t_x} t_x + \frac{\partial F_{x0}}{\partial t_z} t_z + \frac{\partial F_{x0}}{\partial p_x} p_x + \frac{\partial F_{x0}}{\partial p_y} p_y + \frac{\partial F_{x0}}{\partial C_x} C_x + \\ & + \frac{\partial F_{x0}}{\partial C_y} C_y + \frac{\partial F_{x0}}{\partial d} d = -F_{x0} + \frac{\partial F_{x0}}{\partial \alpha} \alpha_0 + \frac{\partial F_{x0}}{\partial \beta} \beta_0 + \frac{\partial F_{x0}}{\partial \gamma} \gamma_0 + \frac{\partial F_{x0}}{\partial t_x} t_{x0} + \frac{\partial F_{x0}}{\partial t_z} t_{z0} + \\ & + \frac{\partial F_{x0}}{\partial p_x} p_{x0} + \frac{\partial F_{x0}}{\partial p_y} p_{y0} + \frac{\partial F_{x0}}{\partial C_x} C_{x0} + \frac{\partial F_{x0}}{\partial C_y} C_{y0} + \frac{\partial F_{x0}}{\partial d} d_0 \end{aligned} \quad (4.25)$$

One can see that the relation (4.25) is a linear equation. For each calibration point one can write this equation and finally obtain an over-determined system of linear equations. We write this system using the matrices, as follows:

$$\begin{aligned} & \mathbf{Coef}_x \cdot [\alpha \quad \beta \quad \gamma \quad t_x \quad t_z \quad p_x \quad p_y \quad C_x \quad C_y \quad d]^T = \\ & \left[ \begin{array}{l} -F_{x0}^1 + \frac{\partial F_{x0}^1}{\partial \alpha} \alpha_0 + \frac{\partial F_{x0}^1}{\partial \beta} \beta_0 + \frac{\partial F_{x0}^1}{\partial \gamma} \gamma_0 + \frac{\partial F_{x0}^1}{\partial t_x} t_{x0} + \frac{\partial F_{x0}^1}{\partial t_z} t_{z0} + \\ \quad + \frac{\partial F_{x0}^1}{\partial p_x} p_{x0} + \frac{\partial F_{x0}^1}{\partial p_y} p_{y0} + \frac{\partial F_{x0}^1}{\partial C_x} C_{x0} + \frac{\partial F_{x0}^1}{\partial C_y} C_{y0} + \frac{\partial F_{x0}^1}{\partial d} d_0 \\ \dots \\ -F_{x0}^N + \frac{\partial F_{x0}^N}{\partial \alpha} \alpha_0 + \frac{\partial F_{x0}^N}{\partial \beta} \beta_0 + \frac{\partial F_{x0}^N}{\partial \gamma} \gamma_0 + \frac{\partial F_{x0}^N}{\partial t_x} t_{x0} + \frac{\partial F_{x0}^N}{\partial t_z} t_{z0} + \\ \quad + \frac{\partial F_{x0}^N}{\partial p_x} p_{x0} + \frac{\partial F_{x0}^N}{\partial p_y} p_{y0} + \frac{\partial F_{x0}^N}{\partial C_x} C_{x0} + \frac{\partial F_{x0}^N}{\partial C_y} C_{y0} + \frac{\partial F_{x0}^N}{\partial d} d_0 \end{array} \right], \end{aligned} \quad (4.26)$$

where  $\mathbf{Coef}_x$  is a matrix with  $N$  rows and 10 columns. Each row of the matrix is formed by the coefficients from the relation (4.25).

The next problem is to compute the inverse of the matrix  $\mathbf{Coef}_x$ . This problem will be solved using a special algorithm detailed presented in the reference [PTVF92]. Knowing this inverse matrix, the solution for our system is given by the relation (4.27), as follows:

$$\begin{aligned}
 & [\alpha \quad \beta \quad \gamma \quad t_x \quad t_z \quad p_x \quad p_y \quad C_x \quad C_y \quad d]^T = \\
 & = (\mathbf{Coef}_x)^{-1} \cdot \begin{bmatrix} -F_{x_0}^1 + \frac{\partial F_{x_0}^1}{\partial \alpha} \alpha_0 + \frac{\partial F_{x_0}^1}{\partial \beta} \beta_0 + \frac{\partial F_{x_0}^1}{\partial \gamma} \gamma_0 + \frac{\partial F_{x_0}^1}{\partial t_x} t_{x_0} + \frac{\partial F_{x_0}^1}{\partial t_z} t_{z_0} + \\ + \frac{\partial F_{x_0}^1}{\partial p_x} p_{x_0} + \frac{\partial F_{x_0}^1}{\partial p_y} p_{y_0} + \frac{\partial F_{x_0}^1}{\partial C_x} C_{x_0} + \frac{\partial F_{x_0}^1}{\partial C_y} C_{y_0} + \frac{\partial F_{x_0}^1}{\partial d} d_0 \\ \dots \\ -F_{x_0}^N + \frac{\partial F_{x_0}^N}{\partial \alpha} \alpha_0 + \frac{\partial F_{x_0}^N}{\partial \beta} \beta_0 + \frac{\partial F_{x_0}^N}{\partial \gamma} \gamma_0 + \frac{\partial F_{x_0}^N}{\partial t_x} t_{x_0} + \frac{\partial F_{x_0}^N}{\partial t_z} t_{z_0} + \\ + \frac{\partial F_{x_0}^N}{\partial p_x} p_{x_0} + \frac{\partial F_{x_0}^N}{\partial p_y} p_{y_0} + \frac{\partial F_{x_0}^N}{\partial C_x} C_{x_0} + \frac{\partial F_{x_0}^N}{\partial C_y} C_{y_0} + \frac{\partial F_{x_0}^N}{\partial d} d_0 \end{bmatrix}. \quad (4.27)
 \end{aligned}$$

We will use these values as the new initial values and we write the equation (4.25) again and we will solve the system (4.26) and obtain another set of values for the camera parameters. We repeat this process until the difference between the last solutions and the last initial values is less than a certain value.

Using the same steps, as we made starting with the relation (4.1) and ending with the relation (4.27), for the relation (4.2) we will obtain, as follows:

$$(Y_p - C_y)z - s_x f \left( 1 + k \left( \frac{(X_p - C_x)^2}{s_x^2} + \frac{(Y_p - C_y)^2}{s_y^2} \right) \right) y = 0. \quad (4.28)$$

With the notations:  $p_x = s_x f$  and  $d = kf^2$  the relation (4.8) becomes:

$$(Y_p - C_y)z - p_x \left( 1 + d \left( \frac{(X_p - C_x)^2}{p_x^2} + \frac{(Y_p - C_y)^2}{p_y^2} \right) \right) y = 0. \quad (4.29)$$

One can make the following notation:

$$F_{x_p, y_p, z_p, x_w, y_w, z_w}^y(\alpha, \beta, \gamma, t_x, t_z, p_x, p_y, C_x, C_y, d) = 0. \quad (4.30)$$

According to Newton Algorithm starting from relation (4.30) one can write the next relation:

$$\begin{aligned} & F_{y,0} + \frac{\partial F_{y,0}}{\partial \alpha}(\alpha - \alpha_0) + \frac{\partial F_{y,0}}{\partial \beta}(\beta - \beta_0) + \frac{\partial F_{y,0}}{\partial \gamma}(\gamma - \gamma_0) + \frac{\partial F_{y,0}}{\partial t_x}(t_x - t_{x,0}) + \\ & + \frac{\partial F_{y,0}}{\partial t_z}(t_z - t_{z,0}) + \frac{\partial F_{y,0}}{\partial p_x}(p_x - p_{x,0}) + \frac{\partial F_{y,0}}{\partial p_y}(p_y - p_{y,0}) + \frac{\partial F_{y,0}}{\partial C_x}(C_x - C_{x,0}) + \\ & + \frac{\partial F_{y,0}}{\partial C_y}(C_y - C_{y,0}) + \frac{\partial F_{y,0}}{\partial d}(d - d_0) = 0 \end{aligned} \quad (4.31)$$

where  $t_{x,0}, t_{z,0}, \alpha_0, \beta_0, \gamma_0, p_{x,0}, p_{y,0}, C_{x,0}, C_{y,0}$  and  $d$  are initial values for the camera parameters.

Going further we will write the explicit relations for  $\frac{\partial F_y}{\partial \alpha}$ :

$$\begin{aligned} \frac{\partial F_y}{\partial \alpha} &= (\cos \beta \cos \alpha \cdot y_w - \cos \beta \sin \alpha \cdot z_w)(Y_p - C_y) - \\ & - (\sin \gamma \sin \beta \cos \alpha \cdot y_w - \cos \gamma \sin \alpha \cdot y_w - \sin \gamma \sin \beta \sin \alpha \cdot z_w - \\ & - \cos \gamma \cos \alpha \cdot z_w)p_x \left( 1 + d \left( \frac{(X_p - C_x)^2}{p_x^2} + \frac{(Y_p - C_y)^2}{p_y^2} \right) \right) \end{aligned} \quad (4.32)$$

$$\begin{aligned} \frac{\partial F_y}{\partial \beta} &= (-\cos \beta \cdot x_w - \sin \beta \sin \alpha \cdot y_w - \sin \beta \cos \alpha \cdot z_w)(Y_p - C_y) - \\ & - (-\sin \gamma \sin \beta \cdot x_w + \sin \gamma \cos \beta \sin \alpha \cdot y_w + \\ & + \sin \gamma \cos \beta \cos \alpha \cdot z_w)p_x \left( 1 + d \left( \frac{(X_p - C_x)^2}{p_x^2} + \frac{(Y_p - C_y)^2}{p_y^2} \right) \right) \end{aligned} \quad (4.33)$$

$$\begin{aligned} \frac{\partial F_y}{\partial \gamma} &= -(\cos \gamma \cos \beta \cdot x_w + \cos \gamma \sin \beta \sin \alpha \cdot y_w - \sin \gamma \cos \alpha \cdot y_w + \\ & - \cos \gamma \sin \beta \cos \alpha \cdot z_w + \sin \gamma \sin \alpha \cdot z_w)p_x \left( 1 + d \left( \frac{(X_p - C_x)^2}{p_x^2} + \frac{(Y_p - C_y)^2}{p_y^2} \right) \right), \end{aligned} \quad (4.34)$$

$$\frac{\partial F_y}{\partial t_y} = -p_y \left( 1 + d \left( \frac{(X_p - C_x)^2}{p_x^2} + \frac{(Y_p - C_y)^2}{p_y^2} \right) \right), \quad (4.35)$$

$$\frac{\partial F_y}{\partial t_z} = Y_p - C_y, \quad (4.36)$$

$$\frac{\partial F_y}{\partial p_x} = \frac{1}{2} p_y d \frac{(X_p - C_x)^2}{p_x^3} y, \quad (4.37)$$

$$\frac{\partial F_y}{\partial p_y} = - \left( \left( 1 + d \left( \frac{(X_p - C_x)^2}{p_x^2} + \frac{(Y_p - C_y)^2}{p_y^2} \right) \right) - \frac{1}{2} d \frac{(Y_p - C_y)^2}{p_y^2} \right) y, \quad (4.38)$$

$$\frac{\partial F_y}{\partial C_x} = 2 p_y d \frac{X_p - C_x}{p_x^2} y, \quad (4.39)$$

$$\frac{\partial F_y}{\partial C_y} = -z + 2d \frac{Y_p - C_y}{p_y} y, \quad (4.40)$$

$$\frac{\partial F_x}{\partial d} = -p_y \left( \frac{(X_p - C_x)^2}{p_x^2} + \frac{(Y_p - C_y)^2}{p_y^2} \right) y. \quad (4.41)$$

In the same way, as we made for the relation (4.11), one can obtain for the relation (4.31) the following form:

$$\begin{aligned} & \frac{\partial F_{y0}}{\partial \alpha} \alpha + \frac{\partial F_{y0}}{\partial \beta} \beta + \frac{\partial F_{y0}}{\partial \gamma} \gamma + \frac{\partial F_{y0}}{\partial t_y} t_y + \frac{\partial F_{y0}}{\partial t_z} t_z + \frac{\partial F_{y0}}{\partial p_x} p_x + \frac{\partial F_{y0}}{\partial p_y} p_y + \frac{\partial F_{y0}}{\partial C_x} C_x + \\ & + \frac{\partial F_{y0}}{\partial C_y} C_y + \frac{\partial F_{y0}}{\partial d} d = -F_{y0} + \frac{\partial F_{y0}}{\partial \alpha} \alpha_0 + \frac{\partial F_{y0}}{\partial \beta} \beta_0 + \frac{\partial F_{y0}}{\partial \gamma} \gamma_0 + \frac{\partial F_{y0}}{\partial t_y} t_{y0} + \frac{\partial F_{y0}}{\partial t_z} t_{z0} + \\ & + \frac{\partial F_{y0}}{\partial p_x} p_{x0} + \frac{\partial F_{y0}}{\partial p_y} p_{y0} + \frac{\partial F_{y0}}{\partial C_x} C_{x0} + \frac{\partial F_{y0}}{\partial C_y} C_{y0} + \frac{\partial F_{y0}}{\partial d} d_0 \end{aligned} \quad (4.42)$$

For each calibration point one can write this equation and finally obtain an over-determined system of linear equations. One can write this system using the matrices, as follows:

$$\text{Coef}_y \cdot [\alpha \ \beta \ \gamma \ t_x \ t_z \ p_x \ p_y \ C_x \ C_y \ d]^T = \begin{bmatrix} -F_{y0}^1 + \frac{\partial F_{y0}^1}{\partial \alpha} \alpha_0 + \frac{\partial F_{y0}^1}{\partial \beta} \beta_0 + \frac{\partial F_{y0}^1}{\partial \gamma} \gamma_0 + \frac{\partial F_{y0}^1}{\partial t_x} t_{x0} + \frac{\partial F_{y0}^1}{\partial t_z} t_{z0} + \\ + \frac{\partial F_{y0}^1}{\partial p_x} p_{x0} + \frac{\partial F_{y0}^1}{\partial p_y} p_{y0} + \frac{\partial F_{y0}^1}{\partial C_x} C_{x0} + \frac{\partial F_{y0}^1}{\partial C_y} C_{y0} + \frac{\partial F_{y0}^1}{\partial d} d_0 \\ \dots \\ -F_{y0}^N + \frac{\partial F_{y0}^N}{\partial \alpha} \alpha_0 + \frac{\partial F_{y0}^N}{\partial \beta} \beta_0 + \frac{\partial F_{y0}^N}{\partial \gamma} \gamma_0 + \frac{\partial F_{y0}^N}{\partial t_x} t_{x0} + \frac{\partial F_{y0}^N}{\partial t_z} t_{z0} + \\ + \frac{\partial F_{y0}^N}{\partial p_x} p_{x0} + \frac{\partial F_{y0}^N}{\partial p_y} p_{y0} + \frac{\partial F_{y0}^N}{\partial C_x} C_{x0} + \frac{\partial F_{y0}^N}{\partial C_y} C_{y0} + \frac{\partial F_{y0}^N}{\partial d} d_0 \end{bmatrix} \quad (4.43)$$

where  $\text{Coef}_y$  is a matrix with  $N$  rows and 10 columns. Each row of the matrix is formed by the coefficients from the relation (4.42). We will solve this system in the same way as we made with the system (4.26) and we obtain, the relation (4.44), as follows:

$$[\alpha \ \beta \ \gamma \ t_x \ t_z \ p_x \ p_y \ C_x \ C_y \ d]^T = (\text{Coef}_y)^{-1} \begin{bmatrix} -F_{y0}^1 + \frac{\partial F_{y0}^1}{\partial \alpha} \alpha_0 + \frac{\partial F_{y0}^1}{\partial \beta} \beta_0 + \frac{\partial F_{y0}^1}{\partial \gamma} \gamma_0 + \frac{\partial F_{y0}^1}{\partial t_x} t_{x0} + \frac{\partial F_{y0}^1}{\partial t_z} t_{z0} + \\ + \frac{\partial F_{y0}^1}{\partial p_x} p_{x0} + \frac{\partial F_{y0}^1}{\partial p_y} p_{y0} + \frac{\partial F_{y0}^1}{\partial C_x} C_{x0} + \frac{\partial F_{y0}^1}{\partial C_y} C_{y0} + \frac{\partial F_{y0}^1}{\partial d} d_0 \\ \dots \\ -F_{y0}^N + \frac{\partial F_{y0}^N}{\partial \alpha} \alpha_0 + \frac{\partial F_{y0}^N}{\partial \beta} \beta_0 + \frac{\partial F_{y0}^N}{\partial \gamma} \gamma_0 + \frac{\partial F_{y0}^N}{\partial t_x} t_{x0} + \frac{\partial F_{y0}^N}{\partial t_z} t_{z0} + \\ + \frac{\partial F_{y0}^N}{\partial p_x} p_{x0} + \frac{\partial F_{y0}^N}{\partial p_y} p_{y0} + \frac{\partial F_{y0}^N}{\partial C_x} C_{x0} + \frac{\partial F_{y0}^N}{\partial C_y} C_{y0} + \frac{\partial F_{y0}^N}{\partial d} d_0 \end{bmatrix} \quad (4.44)$$

We will use these values as the new initial values and we write the equation (4.42) again and we will solve it and obtain another set of values for the camera parameters. We repeat this process until the difference between the last solutions and the last initial values is less than a certain value.

In order to obtain the final solution we make for the camera parameters, which were computed in both systems, the average of their computed values. The results of the calibration procedure can be seen in figure 4.6. The units for  $x$ ,  $y$ , and  $z$ , which correspond to the camera parameters  $t_x$ ,  $t_y$ , and  $t_z$  are millimeters, for *Alfa*, *Beta*, and *Gama*, which correspond to the camera parameters  $\alpha$ ,  $\beta$ , and  $\gamma$  are degrees, and for *CenterX*, *CenterY*, *Px*, and *Py*, which correspond to the camera parameter  $C_x$ ,  $C_y$ ,  $p_x$ , and  $p_y$  are pixels. The parameter noted *Distortion*, which corresponds to camera parameter  $d$  has no unit.

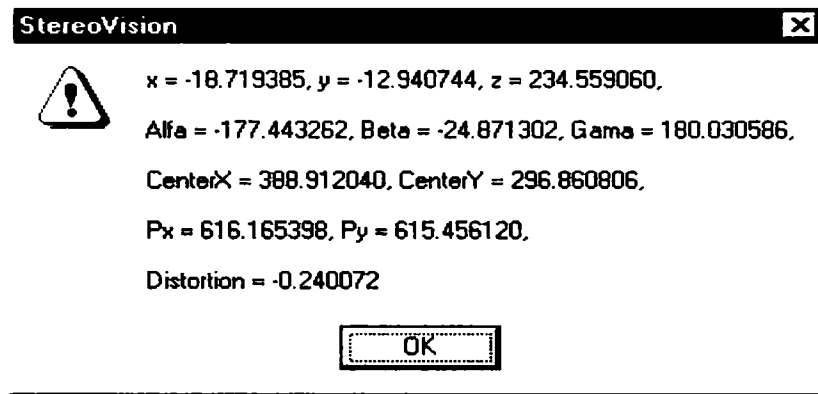


Fig. 4.6. Results of the calibration procedure.



## Image Processing and Shape Recognition

### 5.1 Theoretical Introduction in Image Processing

In this sub-chapter the goal is to present the theoretical support, which was necessary to develop a new image-processing algorithm, as one will see in sub-chapter 5.2.

#### 5.1.1 Image enhancement techniques

Using image enhancement techniques, one has the possibility to improve certain characteristics from an image, characteristics, which presents special interest for the user. According to the reference [GLP99], one can divide these techniques in four categories, which will be presented in the following parts of this sub-chapter.

##### *A. Point operators*

The mathematical definition for a point operator is given by the following relation:

$$g(m, n) = O_{m, n} \{f(m, n)\}. \quad (5.1)$$

One can divide this type of operators in two categories, *spatial invariant* and *spatial variant*.

The equation (5.1) becomes for a spatial invariant operator, as follows:

$$g(m, n) = O \{f(m, n)\}, \quad (5.2)$$

where  $m$  and  $n$  are passive variables. In the following part we will present some examples of operators described by different forms of the relation (5.2).

##### *a) contrast modification*

$$g = \begin{cases} m \cdot f, & 0 < f \leq f_L \\ m_1 \cdot f_L + m_2 \cdot (f - f_L), & f_L < f \leq f_H \\ m_1 \cdot f_L + m_2 \cdot (f_H - f_L) + m_3 \cdot (f - f_H), & f_H < f \leq f_{\max} \end{cases} \quad (5.3)$$

b) *binarysation*

$$g = \begin{cases} g_{\min} & f \leq t_h \\ g_{\max} & f > t_h \end{cases} \quad (5.4)$$

One used operator, which belongs to the special variant category is named *grey level correction operator*. This operator is described by the relation (5.5), as follows:

$$g(m, n) = c_{m,n} \cdot f(m, n), \quad (5.5)$$

where  $c_{m,n}$  is a correction coefficient dependent on the pixel position. If we suppose that the obtained image is  $R$  and the ideal image is  $I$  then the coefficients are computed with the next relation:

$$c_{m,n} = \frac{I_{m,n}}{R_{m,n}}. \quad (5.6)$$

### **B. Geometrical transformations**

A geometrical transformation realize a projection of a pixel from the coordinates  $(x, y)$  to the coordinates  $(x', y')$ . This can be described mathematically with the next two relations:

$$x' = T_x(x, y), \quad (5.7)$$

$$y' = T_y(x, y). \quad (5.8)$$

Computing the Jacoby,  $J$  we will obtain important information about the transformation proprieties. To compute the Jacoby we will use relation (5.9), as follows:

$$J = \begin{vmatrix} \frac{\partial x'}{\partial x} & \frac{\partial x'}{\partial y} \\ \frac{\partial y'}{\partial x} & \frac{\partial y'}{\partial y} \end{vmatrix}. \quad (5.9)$$

We will present now three types from the usual geometrical transformations.

a) *linear transformations (translation, rotation)*

$$x' = a_0 + a_1x + a_2y, \quad (5.10)$$

$$y' = b_0 + b_1x + b_2y, \quad (5.11)$$

$$J = a_1b_2 - a_2b_1. \quad (5.12)$$

b) *bilinear transformations*

$$x' = a_0 + a_1x + a_2y + a_3xy \quad (5.13)$$

$$y' = b_0 + b_1x + b_2y + b_3xy, \quad (5.14)$$

c) *perspective transformations*

$$x' = \frac{a_0 + a_1x + a_2y}{a_3x + a_4y + 1}, \quad (5.15)$$

$$y' = \frac{b_0 + b_1x + b_2y}{b_3x + b_4y + 1}. \quad (5.16)$$

### C. Image smoothing

The goal of image smoothing is to eliminate the noise or small variations of the illumination intensity in an image. All the smoothing operators have the same disadvantage because they eliminate some details from the image and they reduce the accuracy of the edges. These operators are divided in two categories: *linear smoothing operators* and *non-linear smoothing operators*.

From the first category we will present shortly two operators. We will start with the *average operator*. If we have  $N$  frames from the same image  $f$  we will make a temporal average according to the following relation:

$$g = \frac{1}{N} \sum_{k=1}^N (f_k + n_k) = f + \frac{1}{N} \sum_{k=1}^N n_k. \quad (5.17)$$

One can see that the white noise having the standard deviation  $\sigma$  remains white also in the output image  $g$  and the standard deviation decreases with  $N$  square, reference [GLP99].

If we have for an image only one frame then we will make a *spatial average*. We will use a uniform filter having the dimensions  $L \times L$  defined by the relation (5.18), as follows:

$$\mathbf{h} = \frac{1}{L^2} \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{bmatrix}. \quad (5.18)$$

We will convolve the input image  $f$  with this convolution mask and we will obtain the output image  $g$ . In the figure 5.1, one can see the effect of this filter to an ideal edge.

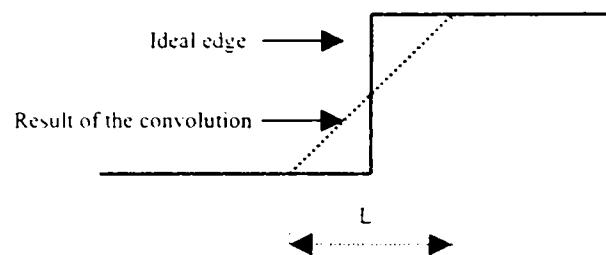


Fig. 5.1. Effect of the special average to an ideal edge.

If we see this filter as a modality of estimating the grey level at the location situated in the middle of the mask one can see that all the pixels from the mask have the same influence. In order to reduce the effect of the pixels, which are situated at the edges of the mask we will create another filter called binomial filter. A 2D binomial filter is built on the base of 1D filter. A 1D filter of any range can be built by convolving several times the following mask:

$$\mathbf{b}_1 = \frac{1}{2} [1 \ 1]. \quad (5.19)$$

According to the explanation before, a 1D filter having the range 2 will be computed as follows:

$$\mathbf{b}_2 = \frac{1}{2} [1 \ 1] * \frac{1}{2} [1 \ 1] = \frac{1}{4} [1 \ 2 \ 1]. \quad (5.20)$$

A 2D filter having the range 2 will be defined by the relation (5.21):

$$\mathbf{b}^2 = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \frac{1}{4} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}. \quad (5.21)$$

All these linear filters give good results when the image is affected by white noise, but they are not so efficient to the binary noise. To eliminate this type of noise we will use a non-linear filter called *median filter*. Using this filter we will replace the grey level for a pixel with the value, which is calculated as being the value situated at the middle position in a row. This row was created by arranging in increasing order all the values of the pixels situated in a window centered at that pixel.

This filter is a particular case of the filters called *statistical ordering filters*. All the filters from this category have the propriety that before any operation with samples from the image these samples are arranged in a certain sequence.

#### ***D. Contour emphasizing and details enhancement techniques***

There are also situations where we are interested to analyze some local structures having small dimensions, for example thin lines or points. For these cases we need to use filters, which will increase the quality of these details. We will present in the following part two types of such filters.

We start by presenting a high pass filter. We will define such a filter using the next two relations:

$${}^L f = f * {}^L \mathbf{h}, \quad (5.22)$$

$$g = cf - (c-1) * {}^L f \quad (5.23)$$

where  $f$  is the input image,  $g$  is the output image,  ${}^L \mathbf{h}$  is the transfer function of a low-pass filter, which was studied in sub-chapter 5.1.1.3., and  $c$  is a constant having values greater than 1. With this constant we can increase or decrease the effect of the filter to the edges.

The second type of filters is represented by *band-pass filters*. We will build a 1D band-pass filter starting from the binomial filters presented in sub-chapter 5.1.1.C. The relation (5.24) describes a 1D band-pass filter having the range 4:

$${}^{BP}h_x^4 = 4(b_x^2 - b_x^4). \quad (5.24)$$

### 5.1.2 Edge detection techniques

Edge detection is one of the most commonly used operations in the image analysis. An edge is the boundary between an object and the background, and indicates the boundary between overlapping objects. Edge detection is a part of the process called segmentation, which means the identification of regions within an image.

One can see in figure 5.2 an ideal step edge.

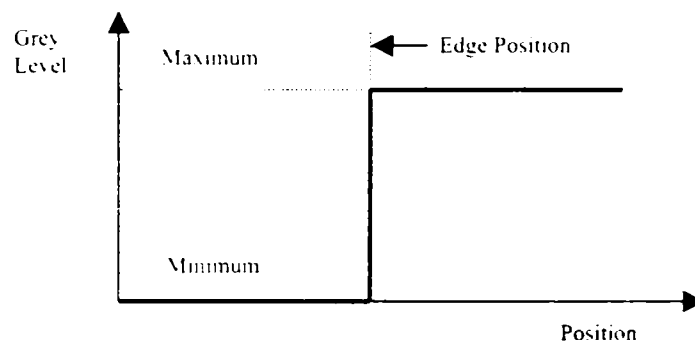


Fig. 5.2. Example of step edge.

This step edge is an ideal model, which never occurs in an image because of the following three reasons:

- objects rarely have such a sharp outline;
- a scene is never sampled so that edges occur at margins of a pixel;
- due to noise, which affects an image.

There are essentially three common types of operators for locating edges. The first type is a *derivative operator* designed to identify places where there are large intensity changes. The second resembles a *template-matching scheme*, where the edge is modeled by a small image showing the abstracted properties of a perfect edge. Finally, there are operators that use a *mathematical model of the edge*. In the following part we will present all these three types of operators, starting with the derivative operators.

### A. Derivative operators

Our goal is to detect the position where is located the boundary between two different grey levels. This position is given by the maximum value of the 2D gradient operator. This operator is defined by the following relation:

$$\nabla A(x, y) = \left( \frac{\partial A}{\partial x} \quad \frac{\partial A}{\partial y} \right). \quad (5.25)$$

Because an image is not a continuous function and can't be differentiated in the usual way we will use differences. We will define the operators  $\nabla_1$  and  $\nabla_2$ , as follows:

$$\nabla_{x1} A(x, y) = A(x, y) - A(x-1, y), \quad (5.26)$$

$$\nabla_{y1} A(x, y) = A(x, y) - A(x, y-1), \quad (5.27)$$

$$\nabla_{x2} A(x, y) = \frac{1}{2} (A(x+1, y) - A(x-1, y)), \quad (5.28)$$

$$\nabla_{y2} A(x, y) = \frac{1}{2} (A(x, y+1) - A(x, y-1)). \quad (5.29)$$

We will define the edge response  $G_{mag}$  and the direction of the edge  $G_{dir}$  with the next two relations:

$$G_{mag} = \sqrt{(\nabla_{x1} A(x, y))^2 + (\nabla_{y1} A(x, y))^2}, \quad (5.30)$$

$$G_{dir} = \arctan\left(\frac{\nabla_{y1} A(x, y)}{\nabla_{x1} A(x, y)}\right). \quad (5.31)$$

In figure 5.3, one can see the block scheme of an edge detector using a 2D gradient operator.

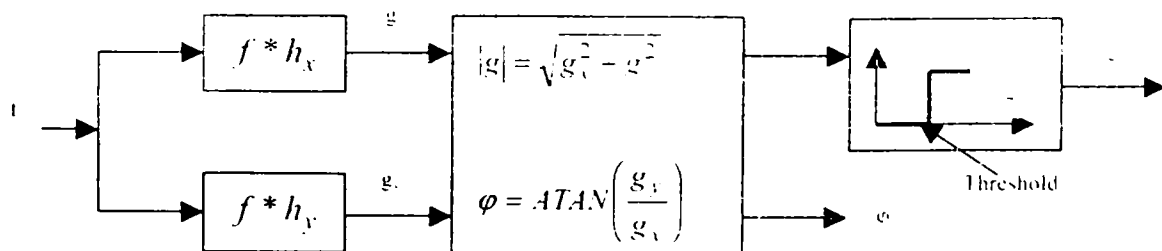


Fig. 5.3. Block scheme for an edge detector using a 2D gradient.

In figure 3,  $f$  represents the input image. The gradient operator is then applied in two directions  $x$  and  $y$ . The next block computes the edge response and the edge direction. In the last block is taken the decision if the current pixel situated at the location  $(x, y)$  is black or white. Of course the variable  $c$  can take only two values one for the black level and another one for the white level.

### B. Template-matching edge detection

The idea behind template matching edge detection is to use a small, discrete template as a model of an edge instead of using a derivative operator directly (as in sub-chapter 5.1.2.A) or a complex, more global model (as in sub-chapter 5.1.2.C.). There are a lot of models for the possible existing edges.

One edge detector, which belongs to this category, is *Sobel edge detector*. The templates for this detector, as convolution masks have the following values:

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}. \quad (5.32)$$

For a pixel having the coordinates  $(i, j)$  one can compute  $S_x$ , and  $S_y$ , as follows:

$$S_x = \mathbf{I}[i-1][j+1] + 2\mathbf{I}[i][j+1] + \mathbf{I}[i+1][j+1] - (\mathbf{I}[i-1][j-1] + 2\mathbf{I}[i][j-1] + \mathbf{I}[i+1][j-1]). \quad (5.33)$$

$$S_y = \mathbf{I}[i+1][j+1] + 2\mathbf{I}[i+1][j] + \mathbf{I}[i+1][j-1] - (\mathbf{I}[i-1][j+1] + 2\mathbf{I}[i-1][j] + \mathbf{I}[i-1][j-1]). \quad (5.34)$$

A second example of the use of templates is the one described by Kirsch. The templates for this detector, as convolution masks have the following values:

$$\mathbf{K}_0 = \begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix} \quad \mathbf{K}_1 = \begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix} \quad \mathbf{K}_2 = \begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix} \quad \mathbf{K}_3 = \begin{bmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}. \quad (5.35)$$

$$\mathbf{K}_4 = \begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix} \quad \mathbf{K}_5 = \begin{bmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{bmatrix} \quad \mathbf{K}_6 = \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix} \quad \mathbf{K}_7 = \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix}. \quad (5.36)$$



These masks are produced to model the kind of grey level change near an edge having various orientations, rather than an approximation to the gradient. There is one mask for each of eight compass directions. For example, a large response to mask  $K_n$  implies a vertical edge (horizontal gradient) at the pixel corresponding to the center of the mask. In order to find the edges an image is convolved with all of the masks at each pixel position. The response of the operator at a pixel is the maximum of any of the eight masks. The direction of the edge pixel is quantized into eight possibilities here, and is  $\pi/4 * i$ , where  $i$  is the number of the mask having the largest response.

### C. Operators using a mathematical model

In the following part we will describe shortly three detectors, which are included in this category.

1. *Marr-Hildreth Edge Detector*. In order to build an edge detection algorithm we have to carry out the following three steps:

- convolve the image  $I$  with a two-dimensional Gaussian function:
- complete the Laplacian of the convolved image; call this  $L$ :
- edge pixels are those for which there is a zero crossing in  $L$ .

A convolution in two dimensions can be expressed as follows:

$$I * G(i, j) = \sum_n \sum_m I(n, m) \cdot G(i - n, j - m). \quad (5.37)$$

The function  $G$  being convolved with the image is a two-dimensional Gaussian. This is defined with the next relation:

$$G_\sigma(x, y) = \sigma^{-2} \cdot e^{-\frac{(x^2 + y^2)}{2\sigma^2}} \quad (5.38)$$

The Laplacian operator is defined as follows:

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}. \quad (5.39)$$

The order doesn't matter in this case, because both convolution and Laplacian are linear operators so, we can make first the Laplacian of the Gaussian (*LoG*), reference [Par97], and then use it to convolve the image  $I$ . The relation (5.40) defines the Laplacian of the Gaussian:

$$\nabla^2 G_\sigma = \left( \frac{r^2 - 2\sigma^2}{\sigma^4} \right) \cdot e^{-\frac{r^2}{2\sigma^2}}, \quad (5.40)$$

where

$$r = \sqrt{x^2 + y^2}. \quad (5.41)$$

2. *The Canny Edge Detector*. Canny specified three issues that an edge detector must address, references [Par97], and [Can86]. These issues are:

- error rate – the edge detector should respond only to edges, and should find all of them;
- localization – the distance between the edge pixels as found by the edge detector and the actual edge should be as small as possible;
- response – the edge detector should not identify multiple edge pixels where only a single edge exists.

The goal was to find a filter, which accomplish all these three criteria. Canny decided to use the first derivative of a Gaussian function as an approximation for the ideal filter, which is too complex to be analytically computed.

A 1D Gaussian function is given by the relation (5.42) and the derivative with respect to  $x$  is given by the relation (5.43), as follows:

$$G(x) = e^{-\frac{x^2}{2\sigma^2}}, \quad (5.42)$$

$$G'(x) = \left( -\frac{x}{\sigma^2} \right) \cdot e^{-\frac{x^2}{2\sigma^2}}. \quad (5.43)$$

A 2D Gaussian function is given by the relation (5.44):

$$G(x) = e^{-\left( \frac{x^2}{2\sigma^2} + \frac{y^2}{2\sigma^2} \right)}. \quad (5.44)$$

In order to simplify the implementation of the convolution procedure we will separate the convolution with a 2D operator in two convolutions with a 1D operator. Due to this the magnitude of the result is computed at each pixel  $(x, y)$  as follows:

$$M(x, y) = \sqrt{I_x^2(x, y) + I_y^2(x, y)}, \quad (5.45)$$

where  $I_x$  and  $I_y$  are the results of the convolutions with 1D operator.

The final step in the Canny edge detector is a non-maximum suppression step where pixels there are not maxima are removed.

3. *The Shen-Castan (ISEF) Edge Detector.* This method use as an optimal filter an infinite symmetrical exponential filter (ISFE), defined by the relation (5.46):

$$f(x) = \frac{p}{2} e^{-p|x|}. \quad (5.46)$$

In two dimensions the filter is given by the next relation:

$$f(x) = \frac{p}{2} e^{-p(|x|+|y|)}. \quad (5.47)$$

This can be applied to an image in the same way, as was the derivative of the Gaussian. But Shen and Castan went one step further and gave a realization of their filters as one-dimensional recursive filters.

Finally, this method make use of false zero-crossing suppression, which has the same goal as the non-maximum suppression used in the Canny edge detector.

### 5.1.3 Methods in grey level segmentation

This sub-chapter is divided in two parts. In the first part we will present some basic information about what image segmentation means and we will give some examples of segmentation methods. In the second part we will discuss about the use of regional thresholds.

#### *A. Basics of Gray-Level Segmentation*

Grey-level segmentation or thresholding is a conversion between a grey-level image and a bi-level image. The idea is that a bi-level image will contain all the essential

information about the number, position and shape of objects while containing a lot less information than a grey-level image.

The most common way to convert an image is to select a single threshold value. All the grey levels below this value will be black (0) and all the values above this level will be white (1). The relation (5.48) describes mathematically this fact:

$$I_b(x, y) = \begin{cases} 0, & \text{if } I(x, y) < T \\ 1, & \text{if } I(x, y) \geq T \end{cases} \quad (5.48)$$

where  $I(x, y)$  is the grey level image,  $I_b(x, y)$  is the bi-level image and  $T$  is the chosen threshold. The problem is now how to find the right value for the threshold. We will present in the following parts some methods to compute this value.

One used method, called sometimes  $p$ -tile method, is to compute the threshold from the following equation:

$$ratio = \frac{\sum_{i=0}^T h(i)}{\sum_{i=0}^{255} h(i)}, \quad (5.49)$$

where  $h(i)$  represents the number of the pixels having the grey level  $i$ . Of course this method can be applied when we know the ratio between the black pixels and the white pixels.

Another common method to compute the threshold is obtained by using histograms. The threshold is determined according to the position where a minimum occurs between two peaks of the histogram.

Starting from the fact that the threshold value is influenced by the grey level of the pixels situated at the boundary between an object and the background we can make a histogram only with the edge pixels. We will compute then the threshold as it was described in the method presented before. In order to determine the edge pixels one method is to convolve the input image with the following mask:

$$\mathbf{L} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \quad (5.50)$$

We will select then only the pixels, which have a large value of the resulting Laplacian and we will build the histogram with these pixels.

*Iterative selection* is a method, which is based on an initial guess of the threshold. Then the threshold is adjusted until the final value is reached. The initial value of the threshold is equal to the mean grey level of the image. Then the mean grey level for all pixels below the threshold is computed and called  $T_b$ , and the mean level of the pixels greater or equal to the initial threshold is computed and called  $T_w$ . The new estimation of the threshold will be computed with the next relation, as follows:

$$T_i = \frac{T_b + T_w}{2}. \quad (5.51)$$

The algorithm is repeated until the difference between two consecutive values for the estimation of the threshold is smaller than a certain value.

The relation (5.52) computes the  $k$ -th estimation of the threshold when the first threshold was noted  $T_0$ :

$$T_k = \frac{\sum_{i=0}^{T_{k-1}} i \cdot h(i)}{2 \cdot \sum_{i=0}^{T_{k-1}} h(i)} + \frac{\sum_{j=T_{k-1}+1}^N j \cdot h(j)}{2 \cdot \sum_{j=T_{k-1}+1}^N h(j)}. \quad (5.52)$$

where  $h(i)$  represents the number of the pixels having the grey level  $i$ .

We will present now very shortly another three methods. For details see the reference [Par97].

We start with the method of grey level histograms. This is based on a statistical method, which is called *analysis of variance*. We will compute first the total variance of the grey level values in the image  $\sigma_t^2$ . For any given threshold  $T$  one can compute the variance of the object pixels and the variance of the background pixels. These represent the within-class variance values noted  $\sigma_w^2$ . Finally the variation of the mean values for each class from the overall mean of the pixels defines a between-classes variance, which will be noted  $\sigma_b^2$ . We can find an optimal value for the threshold  $T$  by minimizing the ratio of the between-class variance to the total variance, reference [Par97].

The second method makes use of the entropy. The entropy is a measure of information content of an image. If there are  $n$  possible symbols  $x_i$  and the symbol  $x_i$  appears with the probability  $p(x_i)$  then the entropy associated to the source of symbols  $X$  is defined as follows:

$$H(X) = -\sum_{i=1}^n p(x_i) \cdot \log(p(x_i)). \quad (5.53)$$

An image can be thought as a source symbols represented by the grey levels, which appear in the image. Having a threshold  $T$  one can compute the entropy of the black pixels and the entropy of the white pixels as follows:

$$H_b = -\sum_{i=0}^T p_i \cdot \log(p_i), \quad (5.54)$$

$$H_w = -\sum_{i=T+1}^{255} p_i \cdot \log(p_i). \quad (5.55)$$

We consider that the grey levels take values from 0 to 255. We will find an optimum for the threshold  $T$  by maximizing the sum between  $H_b$  and  $H_w$ , reference [Par97].

The last method studied by us is called *minimum error thresholding*. The histogram of an image composed by an object and background can be expressed with the next relation:

$$h(i) = \frac{P_1}{\sigma_1 \sqrt{2\pi}} \cdot e^{-\frac{(i - \mu_1)^2}{2\sigma_1^2}} + \frac{P_2}{\sigma_2 \sqrt{2\pi}} \cdot e^{-\frac{(i - \mu_2)^2}{2\sigma_2^2}}, \quad (5.56)$$

where  $\sigma$  and  $\mu$  are the standard deviation and mean of the classes and  $P$ , called also *scaling factor*, is the probability that a pixel belongs to one of the classes. One solution to determine the threshold is to express the probability of the wrong decision and to minimize it. But, the problem is we don't know the values for  $\sigma$ ,  $\mu$  and  $P$  and it is also difficult to estimate them.

Kittler and Illingworth, reference [Par97], created a new criterion to be minimized:

$$J(T) = 1 + 2(P_1(T) \log \sigma_1(T) + P_2(T) \log \sigma_2(T)) - 2(P_1(T) \log P_1(T) + P_2(T) \log P_2(T)), \quad (5.57)$$

where

$$P_1(T) = \sum_{i=1}^T h(i) \quad P_2(T) = \sum_{i=T+1}^{255} h(i), \quad (5.58)$$

$$\mu_1(T) = \frac{\sum_{i=0}^T i \cdot h(i)}{P_1(T)} \quad \mu_2(T) = \frac{\sum_{i=0}^T i \cdot h(i)}{P_2(T)}, \quad (5.59)$$

$$\sigma_1^2(T) = \frac{\sum_{i=0}^T h(i)(i - \mu_1(T))^2}{P_1(T)} \quad \sigma_2^2(T) = \frac{\sum_{i=0}^T h(i)(i - \mu_2(T))^2}{P_2(T)}. \quad (5.60)$$

The value  $T$  that minimized  $J(T)$  will be the best threshold.

### ***B. The use of the regional thresholds***

In practice in most of the cases, a segmentation with a single threshold is not enough to obtain all the necessary information, references [Ber86], [JM03], and [NR79]. One reason is the illumination, which can be different in an image and this way will influence the form and the position of the edges in the segmentation process.

The first problem, which must be solved is to decide in how many regions we should divide an image and how big should be these regions. Afterwards, in order to find the threshold for a region, one can use anyone of the methods presented before in sub-chapter 5.1.3.A.

We will present in the following parts the solution proposed by Chow and Kaneko, according to the reference [Par97]. They divided an image of 256 x 256 pixels in 49 overlapping regions, each one being 64 x 64 pixels. Going further we will make a histogram for each region. We make then a bimodality test for all the histograms. Each bimodal histogram has a pair of Gaussian curves fit to it, using least-squares method. The thresholds for the regions, which have not a bimodal histogram will be interpolated from those that have. The explanation why these regions have not a bimodal histogram is that they include only parts from the object or from the background. Finally, a pixel-by-pixel interpolation of the thresholds values is done, giving every pixel in the image its own threshold. This algorithm historically forms the foundation of the regional thresholding methods, and is frequently cited in the literature, reference [Par97].

A bimodal histogram is expressed as a sum of two Gaussians, as we have seen in the relation (5.56). Our goal is to obtain the mean, the standard deviation and the scaling factors for each of the two Gaussians. First, in order to reduce the influence of the noise the histogram for the current window is found, and is smoothed, as one can see in the relation (5.61):

$$h_s(i) = \frac{h(i-2) + 2 \cdot h(i-1) + 3h(i) + 2h(i+1) + h(i+2)}{9}. \quad (5.61)$$

We will divide now the smoothed histogram in two parts. The separation point will be noted  $\nu$  and represents the grey level where the histogram reaches the minimum value. We can estimate now the initial guess of the parameters for the Gaussian functions using the next relations:

$$N_1 = \sum_{i=0}^{\nu} h(i) \quad N_2 = \sum_{i=\nu+1}^{255} h(i), \quad (5.62)$$

$$\mu_1 = \sum_{i=0}^{\nu} h(i) \cdot i \quad \mu_2 = \sum_{i=\nu+1}^{255} h(i) \cdot i, \quad (5.63)$$

$$\sigma_1 = \sqrt{\frac{1}{N} \sum_{i=0}^{\nu} h(i) \cdot (i - \mu_1)^2} \quad \sigma_2 = \sqrt{\frac{1}{N} \sum_{i=\nu+1}^{255} h(i) \cdot (i - \mu_2)^2}, \quad (5.64)$$

$$P_1 = \frac{\sigma_1 \cdot N_1}{\sum_{i=0}^{\nu} e^{-\frac{(i-\mu_1)^2}{2\sigma_1^2}}} \quad P_2 = \frac{\sigma_2 \cdot N_2}{\sum_{i=\nu+1}^{255} e^{-\frac{(i-\mu_2)^2}{2\sigma_2^2}}}. \quad (5.65)$$

We will note the Gaussian described by one set of parameters as follows:

$$G_i(x) = \frac{P_i}{\sigma_i} \cdot e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}, \quad (5.66)$$

where  $i$  can take the values 1 and 2.

Going further we find the exact value for the parameters by minimizing the next equation:

$$R(P_1, \mu_1, \sigma_1, P_2, \mu_2, \sigma_2) = \sum_{i=0}^{255} (G_1(i) + G_2(i) - h(i))^2. \quad (5.67)$$



Finally, we determine the threshold, as being the intersection point of the two Gaussians by solving the next quadratic equation:

$$\left(\frac{1}{\sigma_1^2} - \frac{1}{\sigma_2^2}\right) \cdot T^2 + 2 \cdot \left(\frac{\mu_1}{\sigma_1^2} - \frac{\mu_2}{\sigma_2^2}\right) \cdot T + \left(\frac{\mu_1^2}{\sigma_1^2} - \frac{\mu_2^2}{\sigma_2^2}\right) + 2 \log\left(\frac{P_1 \sigma_1}{P_2 \sigma_2}\right) = 0. \quad (5.68)$$

If the equation has two solutions we chose the one, which is situated between  $\mu_1$  and  $\mu_2$ .

For the regions where it was not possible to compute a threshold using the method described before we will estimate one from the neighbors, using a linear interpolation or a simple weighted scheme. Finally, we will smooth them by local averaging using the following mask:

$$S = \begin{bmatrix} \frac{1}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} \\ 1 & 2 & 1 \\ \frac{1}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} \end{bmatrix}. \quad (5.69)$$

The last step is to compute a threshold for each pixel. We will consider the situation as one can see in figure 5.4.

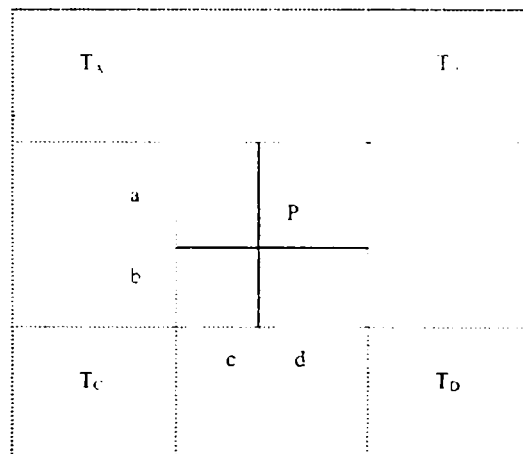


Fig. 5.4. Linear interpolation of individual pixel thresholds.

The threshold value for the pixel noted in the figure 5.4 with  $P$  is computed using the next relation:

$$T = \frac{b \cdot d \cdot T_A + b \cdot c \cdot T_B + a \cdot d \cdot T_C + a \cdot c \cdot T_D}{(a + b) \cdot (c + d)}. \quad (5.70)$$

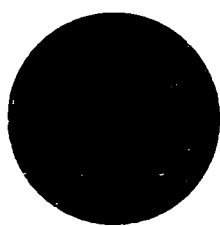
$T_A$ ,  $T_B$ ,  $T_C$  and  $T_D$  are the thresholds for the four adjacent regions to the region where our pixel is situated. The position of the pixel in the window is given by the dimensions  $a$ ,  $b$ ,  $c$  and  $d$ .

## 5.2 New Image Processing for Stereo Vision

In stereovision the main problem is to identify the same point in both pictures obtained from the two cameras, which belong to the stereo sensor. This problem is also known as the *correspondence problem*, reference [EF03a]. In our practical experiments we developed two types of stereo sensors, one in non-parallel configuration and the other one in parallel configuration. A complex analysis of the measurement errors for these two types of stereo sensors was presented in reference [TSIN02]. The conclusion was that the accuracy of the image-processing algorithm has a big influence to the accuracy of the measurement results.

### 5.2.1 Marks selection

The first problem was to decide what types of marks one can use in order to identify them in the pictures obtained from the CCD cameras, reference [NITS03]. One solution was to use crosses, see figure 5.5.b: the other one was to use circles, see figure 5.5.a. We decided to use circles and the reasons why we chose them are presented in the following parts of this chapter.



a) Circle



b) Cross

Fig. 5.5. Marks used to be identified in image processing.

The accuracy of the information obtained from the marks is directly dependent to the accuracy of the detected edge points. If we take a circle having a radius  $r$  the total length of the edges is given by the next relation:

$$L_{circle} = 2\pi * r. \quad (5.71)$$

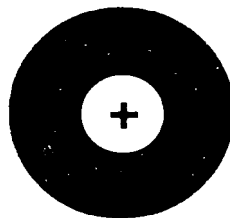
For a cross, having the dimensions  $2r$  horizontal and  $2r$  vertical, the total length of the edges is given by the following relation:

$$L_{cross} = 8 * r. \quad (5.72)$$

With these two relations we show the fact that the number of the edge points for a circle is smaller than the number of the edge points for the corresponding cross. So, we have to detect more edge points in case we use a cross than in case we use a circle. Each one of the edge points is detected with an error and influences the information used in our further calculations. That means, more edge points more errors and finally, bigger influence to the useful information.

In the first phase we identify the circle making use of specialized software. This software offers the possibility to recognize a model, which was taught in a stage before. The problem is that, this software is specialized to identify a form, as an entire, but the information we want is at a pixel level. There is a possibility to use a correlation function implemented in this software, as a second step for the model recognition, but even if we use it the results are not good enough. A detailed analysis of the errors of the 3D stereo measurement system, where we used this correlation function for the model recognition, is presented in references [NIT02], and [TSIN02].

According to these results, we decided to use from this software only the functions necessary to recognize a form as an entire and we developed our own functions to go to the pixel and further to the sub-pixel level. In figure 5.6, one can see the model we used to be recognized in both calibration and measurement procedures implemented for the 3D stereo measurement system.



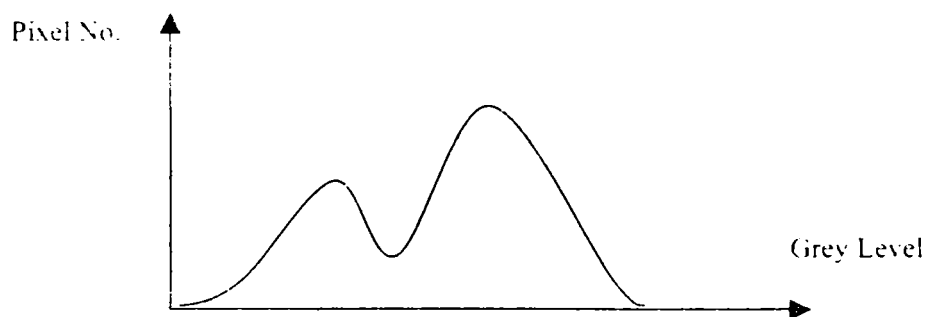
*Fig. 5.6. Model to be recognized.*

We used, at the beginning this type of circle instead of the type presented in figure 5.5.a, because our calibration plate was ordered to make use of the correlation function from the image processing software we had at that moment. To use the correlation function the best model was this one from the figure 5.6.

### 5.2.2 Mathematical approach

As we have already explained, in the first phase we recognize all the circles from the calibration plate using the model from the figure 5.6. With that procedure we will obtain the approximate pixel coordinates of the middle of the circles for each circle. In the second phase we will take each circle separately and we will determine the pixel coordinates of its weight point, as it will be described in the following part of this chapter.

We started by making a segmentation of the image with a fix threshold. The threshold value is taken 127, which means the middle between 0 and 255. 0 represents the black level and 255 the white level. Between these values we have different grey levels. It's very important before we make the segmentation to analyze the image histogram and according to the histogram to adjust the illumination. We have to avoid the situation when there is not enough light (too many pixels having the value 0, in the image histogram) and the situation when there is too much light (too many pixels having value 255 in the image histogram). An image histogram represents the number of the pixels having a certain level of grey for each one of the grey levels, defined between black and white. In figure 5.7, we represented the histogram for an image composed by background having, in reality, only one grey level and an object also, in reality in one grey level, but of course different to the background grey level.



*Fig. 5.7. Example of histogram.*

The results after this segmentation with a fix threshold were not so good and we decided to make a new image segmentation with a dynamic threshold. This way, we had to replace for each circle the fix threshold with a new value, which would provide us a better segmentation of the image. Our idea was to take a square region around each circle so that, in this region the ratio between white points and black points is the same. We will consider the square having the dimensions  $4R \times 4R$ , where  $R$  is the radius of the circle. From the

geometrical dimensions we can compute now the ratio between the black points and the total number of the points situated in one square region, as one can see in relation (5.73).

$$ratio = \frac{TotalBlackPOINTS}{TotalPOINTS} = 0.13. \quad (5.73)$$

Relation (5.74) is a detailed form for relation (5.73):

$$ratio = \frac{\pi \cdot R^2 - \pi \cdot r^2 + 2 \cdot x \cdot y - x^2}{16 \cdot R^2}, \quad (5.74)$$

where the meaning of the notations becomes clear looking at figure 5.8.

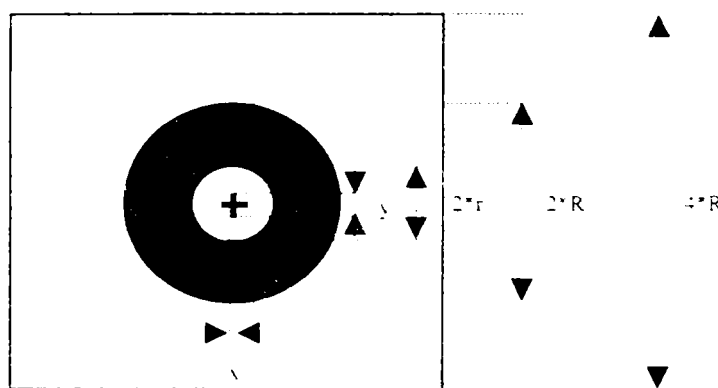


Fig. 5.8. Details of a square region from the calibration plate.

In order to simplify our method and to make it more accurate we will make the small circle, situated inside of the big circle, black. There are two possibilities. One is to make it in the image; it means to set all the pixels, which define the small circle to the value zero. The second possibility is to paint in black the small circle directly on the calibration plate. The second solution has two advantages. First one is that the value for the ratio is computed more simply and more accurate. The second one is that we don't need any software function to find the points, which belong to the small circle and to set them to the black level. If we chose the second solution we have to change also the relation to compute the ratio. This way, the relation (5.74) becomes, as follows:

$$ratio = \frac{\pi \cdot R^2}{16 \cdot R^2} = 0.19625 \quad (5.75)$$

Also, figure 5.8 will be changed, as one can see in figure 5.9.

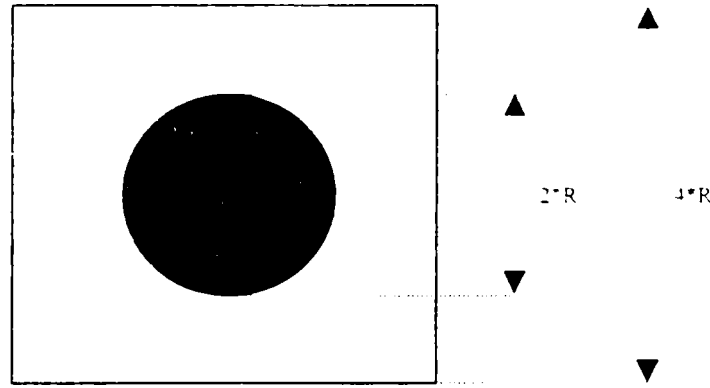


Fig. 5.9. Details of a region from the calibration plate with a black point.

The problem is now that in order to divide the image in 25 regions where the ratio of the black points is constant, given by the relation (5.75), we need for each circle the value of the radius  $R$ . The method to compute these values is presented in the following parts of this chapter.

We identify first an approximate position for the middle of the black circle in the image taken with a camera. We used an image processing software, which gives the possibility of teaching one model and then this model is identified in the picture. We will define now a square region centered on the position where a circle was identified. The side of the square is 50 pixels. We took this number because the maximum radius of one circle in our case is 20 pixels and this way, we are sure that the circle is inside of the square.

Going further, we will analyze now the grey level of pixels, belonging to such a square region, starting from left to right ( $x$  direction), for different  $y$  values. The values for  $y$  should be situated between 0 and 5 pixels in negative direction relative to the detected middle of the circle and also between 0 and 5 pixels in positive direction. This way, we are sure that we will analyze the entire region where the real middle of the circle is located. We start by computing the next relation:

$$\Delta g(x_L, y_i) = \text{Max}_i \{g(x_{i-1}, y_i) - g(x_{i+1}, y_i)\}, \quad (5.76)$$

where  $g(x_i, y_j)$  is the grey level for the pixel whose coordinates are  $x_i, y_j$ . The value  $L$  is equal to the corresponding value for  $i$  where the maximum was determined. We continue by computing now the next relation:

$$\Delta g(x_R, y_i) = \text{Max}_i \{g(x_{i+1}, y_i) - g(x_{i-1}, y_i)\}. \quad (5.77)$$

As we said, we repeat this procedure for different  $y$  values. For each value  $y_j$  we compute the difference according to the relation (5.78):

$$R_x(y_j) = x_R - x_L. \quad (5.78)$$

Finally, we compute the maximum radius  $R_x$  in  $x$  direction, using the following relation:

$$R_x = \text{Max}\{R_x(y_j)\}. \quad (5.79)$$

We will repeat the same procedure from down to up ( $y$  direction) for different  $x$  values in the same conditions as we did before from left to right. This way, we will compute step by step the following four relations:

$$\Delta g(x_i, y_D) = \text{Max}\{g(x_i, y_{j-1}) - g(x_i, y_{j+1})\}, \quad (5.80)$$

$$\Delta g(x_i, y_U) = \text{Max}\{g(x_i, y_{j+1}) - g(x_i, y_{j-1})\}, \quad (5.81)$$

$$R_y(x_i) = y_U - y_D, \quad (5.82)$$

$$R_y = \text{Max}\{R_y(x_i)\}. \quad (5.83)$$

The final value for the circle radius will be computed, as one can see just below:

$$R = \frac{R_x + R_y}{2}. \quad (5.84)$$

We obtain two different values for the circle radius,  $R_x$  and  $R_y$  because what is a circle in reality suffers some modifications, by projection in the image and becomes an ellipse. We can approximate the surface of this ellipse with the surface of a circle having the radius  $R$ , according to the relation (5.84).

In this moment we have the approximate coordinates of the middle of each circle and we know also the radius in pixels for all of them. We can define now a square region around each circle. We know that the total number of the pixels from one region can be computed using the next relation:

$$\text{TotalPOINTS} = 16 * R^2 = \sum_{z=0}^{255} p(z), \quad (5.85)$$

where  $p(z)$  represents the number of the pixels whose grey level is  $z$ . The total number of the black points is computed with relation (16), as follows:

$$TotalBlackPOINTS = \sum_{z=0}^I p(z), \quad (5.86)$$

where  $T$  is the threshold. Knowing the ratio between the black points and the total number of the points of a square region one can compute the threshold  $T$ . Having a new threshold for each square region of the image, we will make a new segmentation of the image in all of the 25 regions. We make again the observation that now each region will have its own threshold.

Going further, we have to compute the weight point of each circle. We will use the next two formulas:

$$C_x = \frac{\sum_{i=0}^{(4R-1)} \sum_{j=0}^{(4R-1)} g_S(x_i, y_j) \cdot x_i}{\sum_{i=0}^{(4R-1)} \sum_{j=0}^{(4R-1)} g_S(x_i, y_j)}, \quad (5.87)$$

$$C_y = \frac{\sum_{i=0}^{(4R-1)} \sum_{j=0}^{(4R-1)} g_S(x_i, y_j) \cdot y_j}{\sum_{i=0}^{(4R-1)} \sum_{j=0}^{(4R-1)} g_S(x_i, y_j)}, \quad (5.88)$$

where  $g_S(x, y)$  is defined by relation (19), as follows:

$$g_S(x, y) = \begin{cases} 1, & \text{if } g(x, y) < T \\ 0, & \text{if } g(x, y) \geq T \end{cases} \quad (5.89)$$

If we note with  $N$  the total number of the black points from a region the relations (5.87) and (5.88) become, as follows:

$$C_x = \frac{\sum_{k=0}^{N-1} x_k}{N}, \quad (5.90)$$

$$C_y = \frac{\sum_{k=0}^{N-1} y_k}{N}. \quad (5.91)$$



In this moment we have two new coordinates for the middle of each circle. We have to say that the formulation “the middle of the circle” is not the right one, because of two reasons. First reason is that we talk now about the weight point of a geometrical entity. Second reason is that the circle from the calibration plate is not a circle anymore, in the image taken with the CCD camera, but an ellipse.

### 5.2.3 Sub-pixel approach

Until now we worked only at the pixel level. The results were better than in case of using only the functions from the dedicated software, but we considered that we could obtain more. The solution was to go to the sub-pixel level, reference [Dev95]. We will start by explaining a real situation. We will consider a simple plate half white and half black.

If we look with a camera to this plate the image, which will be stored on the chip, will be a little distorted, as one can see in figure 5.10.d.

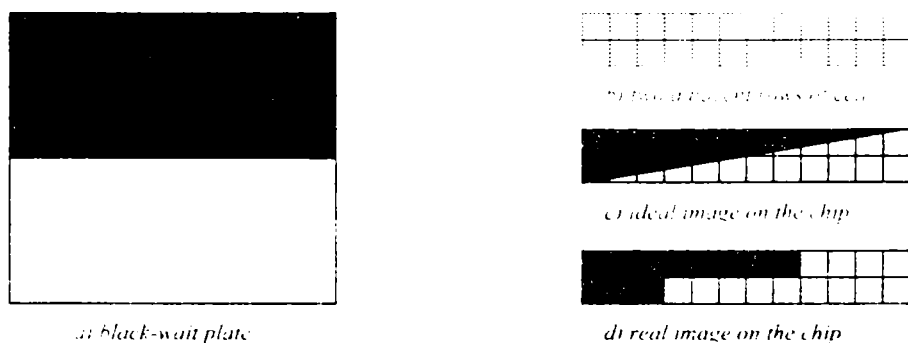


Fig. 5.10. Details at the cells level on the chip of a CCD camera.

We consider that the transfer from the chip image to the computer image take place without errors so the situation from figure 5.10.d is valid also at the pixel level. The idea is that in most of the cases the border between an object and the background in the pixel image should be situated on the surface of one pixel not at the border between two pixels. From physical considerations we can't have in one cell of the chip two different levels of electricity and also the corresponding pixel can't have two levels of grey. We developed a mathematical algorithm, which will determine a sub-pixel value associated to the location where the border between two levels of grey should be placed.

Our goal is to reach an accuracy of a tenth of pixel. To realize that we have to explore each circle in the following way: we start from the weight point of the “circle” with lines to the edges of the “circle”. There are two problems that must be solved. The first one is: how many lines we have to use? The second one is to compute the grey level in certain sub-pixel positions situated on this line.

The number of the lines we should use is determined by the value of the angle between two consecutive lines. The length of the circle is computed using the relation (5.92), as follows:

$$L_c = 2\pi \cdot R \cdot \Delta p, \quad (5.92)$$

where  $\Delta p$  is the length of one edge of a square pixel. We want to make an exploration from tenth to tenth of pixel. The angle between two consecutive exploration lines will be computed with the next relation:

$$\Delta\alpha = \frac{180}{\pi \cdot R \cdot n} [\text{deg}]. \quad (5.93)$$

where  $n$  is equal to the number of parts in which we want to divide a pixel. In our case we take  $n$  equal to 10 and the maximum value for the radius,  $R$ , equal to 20. This way, we obtain for  $\Delta\alpha$  the value 0.28, which means we have to use approximate 1285 exploration lines.

For each of these lines we will analyze a part of it having the length equal to the length of 5 pixels. The middle of this part is situated at a distance equal to the circle radius  $R$ . Between the Cartesian coordinates of one point situated in this part of the line and the polar coordinates of the same point one can write the next two relations:

$$x = C_x + (R + d) \cdot \cos \alpha, \quad (5.94)$$

$$y = C_y + (R + d) \cdot \sin \alpha, \quad (5.95)$$

where  $d$  takes values between  $-2.5$  and  $+2.5$ . The difference between two consecutive values  $d$  is 0.1.  $C_x$  and  $C_y$  are the coordinates of the “circle” weight point.

In figure 5.11, one can see the correspondence between the coordinates  $(x, y)$  and the coordinates  $(d, \alpha)$ .

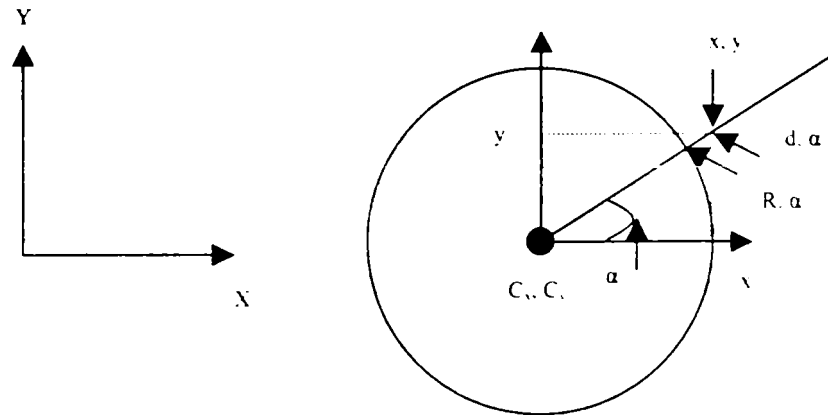


Fig. 5.11. Correspondence between Cartesian and Polar coordinates.

As we said before we want to have the grey level of the points situated at any location  $d$  on the exploration line. Using the relations (5.94) and (5.95) we are able to compute for each of the 51 values of  $d$  his corresponding coordinates  $(x, y)$ .

In figure 5.12, one can see details at the sub-pixel level for one exploration line.

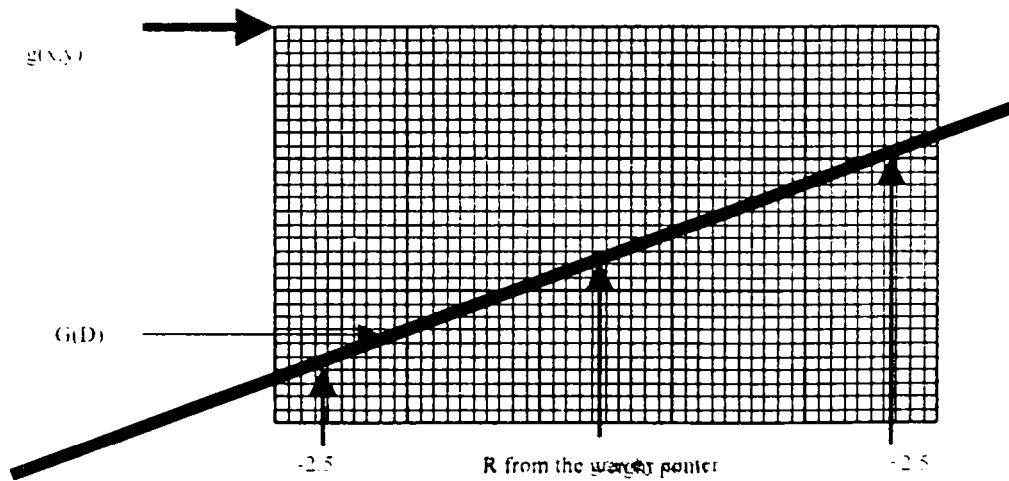


Fig. 5.12. Details for an exploration line at sub-pixel level.

The problem is now that these coordinates  $(x, y)$  have float values and we know the grey level only for those, which have integer values. In the following part we will present a solution to compute the grey level of a point whose coordinates take float values.

In figure 5.13, we represented a square formed by nine pixels.

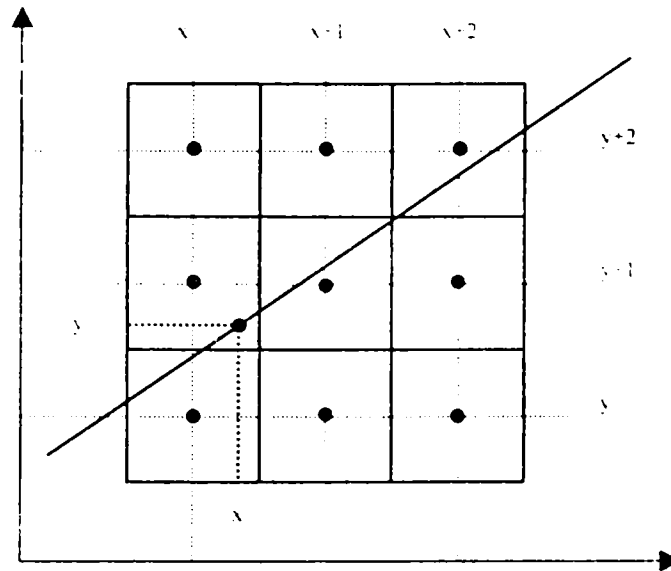


Fig. 5.13. Sub-pixel resolution.

The values  $x$  and  $y$  are positive integers. They represent the location of the pixel in the image. With small circles we represented the grey level of the one pixel and we placed this in the middle of the pixel. We are interested to compute the grey level of the point situated at the location  $(x_1, y_1)$ , as one can see in figure 5.13. To simplify the further calculation we make first the next notations:

$$\Delta x = x_1 - x, \quad (5.96)$$

$$\Delta y = y_1 - y. \quad (5.97)$$

Now we can compute the grey level of the point situated at location  $(x_1, y_1)$  using the relation (5.98):

$$g(x_1, y_1) = g(x, y) \cdot (1 - \Delta x) \cdot (1 - \Delta y) + g(x, y + 1) \cdot (1 - \Delta x) \cdot \Delta y + g(x + 1, y + 1) \cdot \Delta x \cdot \Delta y + g(x + 1, y) \cdot \Delta x \cdot (1 - \Delta y) \quad (5.98)$$

In order to simplify the mathematical calculation and to avoid working with float numbers we will define a new variable  $D$  as follows:

$$D = 10 \cdot d - 25. \quad (5.99)$$

The new variable  $D$  will take integer values between  $-25$  and  $+25$ .

One can compute the function  $G(D)$  using the next relation:

$$G(D) = g\left(\frac{D-25}{10}\right) = g(d) = g(x, y). \quad (5.100)$$

This way, we divided an interval of five pixels in fifty sub-pixels intervals and we computed for each sub-pixel interval the corresponding grey level.

Going further we must find a mathematical relation, which approximates as good as possible the function  $G(D)$ . We will make first a graphical interpretation of the computed values for  $G(D)$ . In figure 5.14, one can see the approximate graphical representation of this function.

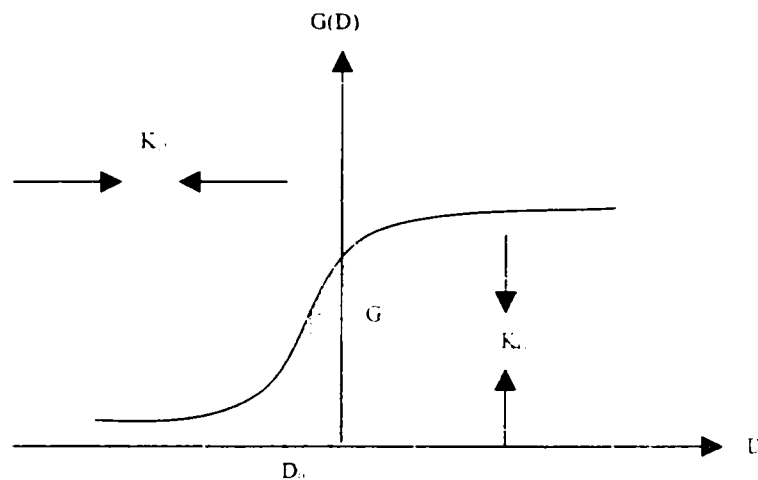


Fig. 5.14. Graphical representation for  $G(D)$ .

One can see that this function can be approximated with an *arctan* function, as follows:

$$G(D) = G_0 + K_G \cdot \arctan(K_D \cdot (D - D_0)). \quad (5.101)$$

Our final goal is to compute  $D_0$ . Unfortunately, we can't compute  $D_0$  without computing the other three unknowns  $G_0$ ,  $K_G$ ,  $K_D$ , from the equation (5.101). One can write the equation (5.101), as follows:

$$F_{D_i, G_i}(D_0, G_0, K_D, K_G) = 0, \quad (5.102)$$

where  $D_i$  and  $G_i$  are calculated in the steps before. As, we told we have fifty pairs of points  $(D_i, G_i)$ . For each pair we can write the equation (5.102). This way, we will obtain an over determined system of nonlinear equations. To solve system we have to accomplish two steps.

First step is to find a good set of starting values for  $D_0$ ,  $G_0$ ,  $K_D$ , and  $K_G$ . The second step is to make the nonlinear equations linear. After making these two steps, we obtain an over determined system of linear equations, which will be solved using least square method. With the new solutions we will repeat again the algorithm. We will stop when the difference between two sets of consecutive solutions is less as a certain value.

The obtained value for  $D_0$  will be used to calculate the corresponding  $x_0$  and  $y_0$ . For that we will use the relations (5.104) and (5.105). The new coordinates for the circle weight point will be computed using the next two relations:

$$C_x = \frac{\sum_{k=0}^{N-1} x_0(k)}{NP}, \quad (5.103)$$

$$C_y = \frac{\sum_{k=0}^{N-1} y_0(k)}{NP}, \quad (5.104)$$

where  $NP$  is the total number of the edge points and  $x_0(k)$  and  $y_0(k)$  are the computed coordinates of the edge points. These new coordinates were used in our further calculations. The results of the 3D measurements made with our stereo sensor will be presented in the chapter three of this paper. Also a detailed analysis of the errors of the stereo sensor will be presented in chapter three too.

In the following part of this chapter we will explain the solution developed by us to find a good set of starting values for our variables.  $D_0$ ,  $G_0$ ,  $K_D$ , and  $K_G$ . reference [TSNI04]. We will compute first the next difference:

$$\Delta G = G_{i+1} - G_{i-1}, \quad (5.105)$$

where  $i$  takes values from  $-24$  to  $+24$ . We will determine maximum of  $\Delta G$  and according to this maximum we will obtain the starting values for  $D_0$  and  $G_0$ .

To obtain the starting values for  $K_D$  and  $K_G$  we will take two pairs of points  $(D_i, G_i)$  and  $(D_j, G_j)$  and we write the next two equations:

$$G_i - G_0 = K_G \cdot \arctan(K_D (D_i - D_0)), \quad (5.106)$$

$$G_j - G_0 = K_G \cdot \arctan(K_D (D_j - D_0)). \quad (5.107)$$

We will solve this system and we will obtain the starting values for  $K_D$  and  $K_G$ .

The solution to solve this system will be presented in the following part of this chapter. To simplify the form of the equations, described by the relations (5.106) and (5.107) we make the following notations:

$$G_i - G_0 = a_1, \quad (5.108)$$

$$G_j - G_0 = a_2, \quad (5.109)$$

$$D_i - D_0 = b_1, \quad (5.110)$$

$$D_j - D_0 = b_2, \quad (5.111)$$

$$K_D = x, \quad (5.112)$$

$$K_G = y. \quad (5.113)$$

With these notations the relations (5.106) and (5.107) will become as follows:

$$a_1 = y \cdot \arctan b_1 \cdot x, \quad (5.114)$$

$$a_2 = y \cdot \arctan b_2 \cdot x, \quad (5.115)$$

where  $x$  and  $y$  are our unknowns and  $a_1$ ,  $a_2$ ,  $b_1$  and  $b_2$  are known. We eliminate  $y$  from the equations (5.114) and (5.115) and we obtain, as follows:

$$a_1 \cdot \arctan b_2 \cdot x = a_2 \cdot \arctan b_1 \cdot x. \quad (5.116)$$

We will make now a graphical analysis of the relation (5.116). We will define two functions  $f_1$  and  $f_2$ , as one can see in the next two relations:

$$f_1(x) = a_1 \cdot \arctan b_2 \cdot x, \quad (5.117)$$

$$f_2(x) = a_2 \cdot \arctan b_1 \cdot x. \quad (5.118)$$

The graphical representation of these two functions is presented in figure 5.15.

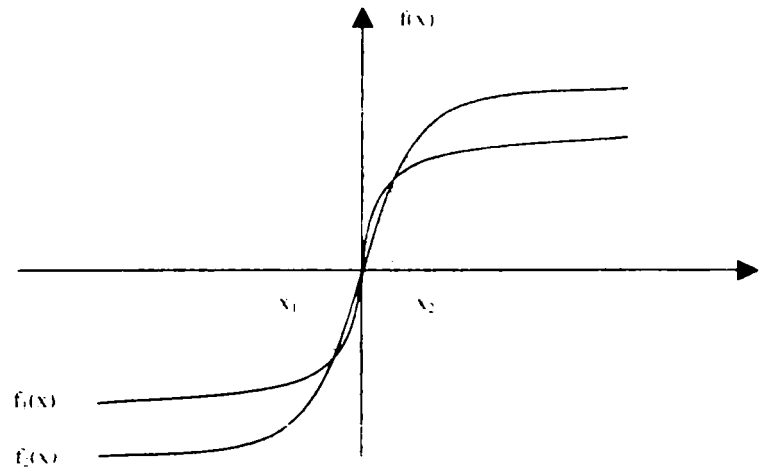


Fig. 5.15. Graphical representation of the functions  $f_1$  and  $f_2$ .

We have to say that the representation from the figure 5.15 shows the situation when the equation (5.116) has three solutions  $0$ ,  $x_1$  and  $x_2$ . This is the case we should have because we obtained this equation from a real situation. It means, that the equation must have these three solutions. We are only interested to calculate  $x_2$ , because we know from the definition of function  $G(D)$ , relation (5.101), that the coefficient  $K_D$ , which is the same with  $x$ , relation (5.112), has only positive values.

The situation when the equation (5.116) has only the solution  $0$  is graphically represented in figure 5.16.

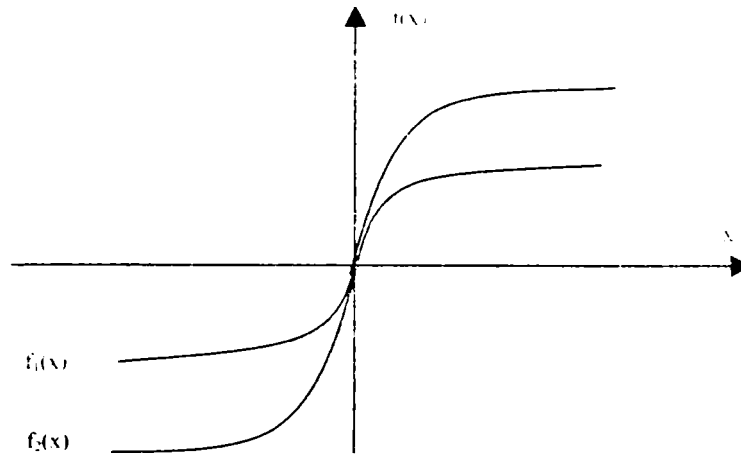


Fig. 5.16. Functions  $f_1$  and  $f_2$  - theoretical situation.

To solve the equation (5.116) we will compute first the derivations of the functions in the origin. For a function  $f$  having the form described by the relation (5.119), the derivation of this function will be calculated using the relation (5.120).

$$f(x) = a \cdot \arctan b \cdot x \quad (5.119)$$

$$f'(x) = \frac{a \cdot b}{1 + (\arctan b \cdot x)^2} \quad (5.120)$$



That means that for our functions  $f_1$  and  $f_2$  the derivations in the origin will have the following values:

$$f_1'(0) = a_1 \cdot b_2, \quad (5.121)$$

$$f_2'(0) = a_2 \cdot b_1. \quad (5.122)$$

Comparing the values for these two derivations we will know what sign has the difference between  $f_1$  and  $f_2$  in the positive vicinity of 0 and also in the negative and positive vicinity of the solution  $x_2$ , see figure 5.15. The solution  $x_2$  will be computed numerically using a starting value and then several iterations.

We will present in the next part of this chapter the linearisation procedure. The equation (5.102) can be written also in the following form:

$$F_{D_i, G_i} = G_0^0 + K_G^0 \cdot \arctan(K_D^0 \cdot (D_i - D_0^0)) - G_i = 0, \quad (5.123)$$

where  $D_0^0, G_0^0, K_D^0, K_G^0$  are initial values for  $D_0, G_0, K_D,$  and  $K_G$  and  $F_{D_i, G_i}$  is the short notation from  $F_{D_i, G_i}(D_0^0, G_0^0, K_D^0, K_G^0)$ .

We will compute then the next four derivations:

$$\frac{\partial F}{\partial D_0} = \frac{\partial F_{D_i, G_i}}{\partial D_0} \Big|_{D_0^0, G_0^0, K_D^0, K_G^0} = -K_G^0 \cdot K_D^0 \cdot D_0^0 \cdot \frac{1}{1 + (K_D^0 \cdot (D_i - D_0^0))^2}, \quad (5.124)$$

$$\frac{\partial F}{\partial G_0} = \frac{\partial F_{D_i, G_i}}{\partial G_0} \Big|_{D_0^0, G_0^0, K_D^0, K_G^0} = 1, \quad (5.125)$$

$$\frac{\partial F}{\partial K_D} = \frac{\partial F_{D_i, G_i}}{\partial K_D} \Big|_{D_0^0, G_0^0, K_D^0, K_G^0} = K_G^0 \cdot (D_i - D_0^0) \cdot \frac{1}{1 + (K_D^0 \cdot (D_i - D_0^0))^2}, \quad (5.126)$$

$$\frac{\partial F}{\partial K_G} = \frac{\partial F_{D_i, G_i}}{\partial K_G} \Big|_{D_0^0, G_0^0, K_D^0, K_G^0} = \arctan(K_D^0 \cdot (D_i - D_0^0)). \quad (5.127)$$

With these last relations we can obtain a linear equation for (5.102) as follows:

$$F_{i,j} + \frac{\partial F}{\partial D_{i,j}} \cdot (D_{i,j} - D_{i,j}^0) + \frac{\partial F}{\partial G_{i,j}} \cdot (G_{i,j} - G_{i,j}^0) + \frac{\partial F}{\partial K_{i,j}} \cdot (K_{i,j} - K_{i,j}^0) + \frac{\partial F}{\partial K_{i,j}'} \cdot (K_{i,j}' - K_{i,j}'^0) = 0. \quad (5.128)$$

We will make linear all the nonlinear equation of our obtained over-determined system of equations, using the equation (5.128). The new over-determined system of linear equations will be solved using least square method, in the same way as we made with the systems solved in sub-chapter 4.2.2.

## Analysis of the 3D Measurements

### 6.1 Contributions at the Development of Accurate Measurement Procedures

To measure we will use the same device as in the calibration procedure, see figure 4.5, chapter 4. We will move the plate in different positions and we will measure with the stereo sensor the 3D coordinates of the points from the calibration plate. The big advantage of this device is that we can control very precisely the  $x$ ,  $y$ , and  $z$  movements of the plate (accuracy 0.025 mm, see 4.2.1). This way, we could verify the accuracy of the measurements made with the calibrated stereo sensor and we developed an algorithm to improve substantially the accuracy of the measurements.

The goal is to measure the 3D coordinates of a point with respect to the world frame using the stereo sensor. We consider a point  $P$  having the coordinates  $x_w, y_w, z_w$  with respect to the world frame. This point will have the coordinates  $x_R, y_R, z_R$  with respect to the camera right frame and the coordinates  $x_L, y_L, z_L$  with respect to the camera left frame. With these notations one can write the next relation:

$${}^w_L \mathbf{T} \cdot \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = {}^w_R \mathbf{T} \cdot \begin{bmatrix} x_R \\ y_R \\ z_R \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \quad (6.1)$$

The relation (6.1) can be written, as follows:

$${}^w_L \mathbf{R} \cdot \begin{bmatrix} x_L \\ y_L \\ z_L \\ 1 \end{bmatrix} = {}^w_R \mathbf{R} \cdot \begin{bmatrix} x_R \\ y_R \\ z_R \\ 1 \end{bmatrix} = \begin{bmatrix} t_x & -t_x \\ t_y & -t_y \\ t_z & -t_z \end{bmatrix}, \quad (6.2)$$

where  ${}^w_L \mathbf{R}$  represents the rotation from the world frame to the camera left frame and  ${}^w_R \mathbf{R}$  the rotation from the world frame to the camera right frame.  $t_x, t_y, t_z$  are the translations from

the world frame to the camera left frame and  $t_{x_R}, t_{y_R}, t_{z_R}$  are the translations from the world frame to the camera right frame.

From the two images made with the stereo sensor we can find the pixel coordinates of the point P. We denote these coordinates with  $X_L, Y_L$  for camera left and with  $X_R, Y_R$  for camera right.

Between the 3D coordinates and the pixel coordinates of the point P one can write the nest relations:

$$\frac{1}{\left(1 + d^L \left( \frac{(X_p - C_x)^2}{p_x^2} + \frac{(Y_p - C_y)^2}{p_y^2} \right)\right)} \frac{X_L - C_x^L}{p_x^L} \cdot z_L = X_L, \quad (6.3)$$

$$\frac{1}{\left(1 + d^L \left( \frac{(X_p - C_x)^2}{p_x^2} + \frac{(Y_p - C_y)^2}{p_y^2} \right)\right)} \frac{Y_L - C_y^L}{p_y^L} \cdot z_L = Y_L. \quad (6.4)$$

$$\frac{1}{\left(1 + d^R \left( \frac{(X_p - C_x)^2}{p_x^2} + \frac{(Y_p - C_y)^2}{p_y^2} \right)\right)} \frac{X_R - C_x^R}{p_x^R} \cdot z_R = X_R. \quad (6.5)$$

$$\frac{1}{\left(1 + d^R \left( \frac{(X_p - C_x)^2}{p_x^2} + \frac{(Y_p - C_y)^2}{p_y^2} \right)\right)} \frac{Y_R - C_y^R}{p_y^R} \cdot z_R = Y_R. \quad (6.6)$$

where  $p_x^L, p_y^L, C_x^L, C_y^L, d^L$  are the internal parameters for camera left and  $p_x^R, p_y^R, C_x^R, C_y^R, d^R$  are the internal parameters for camera right. The values for these parameters are known because they were computed in the calibration procedure. We will make the following notations:

$$\frac{1}{\left(1 + d^L \left( \frac{(X_p - C_x)^2}{p_x^2} + \frac{(Y_p - C_y)^2}{p_y^2} \right)\right)} \frac{X_L - C_x^L}{p_x^L} = c1x^L, \quad (6.7)$$

$$\frac{1}{\left(1 + d^L \left( \frac{(X_p - C_x)^2}{p_x^2} + \frac{(Y_p - C_y)^2}{p_y^2} \right)\right)} \frac{Y_L - C_y^L}{p_y^L} = c1y^L. \quad (6.8)$$

$$\frac{1}{\left(1 + d^R \left( \frac{(Y_p - C_v)^f}{p_v^2} + \frac{(Y_p - C_v)^f}{p_v^2} \right)\right)} \frac{X_R - C_v^R}{p_v^R} = ctx^R, \quad (6.9)$$

$$\frac{1}{\left(1 + d^R \left( \frac{(Y_p - C_v)^f}{p_v^2} + \frac{(Y_p - C_v)^f}{p_v^2} \right)\right)} \frac{X_R - C_v^F}{p_v^R} = cty^R. \quad (6.10)$$

With these notations the relations (6.3), (6.4), (6.5) and (6.6) will become, as follows:

$$ctx^L \cdot z_L = x_L, \quad (6.11)$$

$$cty^L \cdot z_L = y_L, \quad (6.12)$$

$$ctx^R \cdot z_R = x_R, \quad (6.13)$$

$$cty^R \cdot z_R = y_R. \quad (6.14)$$

From the equation (6.2) and the relations (6.11), (6.12), (6.13) and (6.14) one can obtain:

$$(r_1^L ctx^L + r_2^L cty^L + r_3^L) z_L - (r_1^R ctx^R + r_2^R cty^R + r_3^R) z_R = t_{x_R} - t_{x_L}, \quad (6.15)$$

$$(r_4^L ctx^L + r_5^L cty^L + r_6^L) z_L - (r_4^R ctx^R + r_5^R cty^R + r_6^R) z_R = t_{y_R} - t_{y_L}, \quad (6.16)$$

$$(r_7^L ctx^L + r_8^L cty^L + r_9^L) z_L - (r_7^R ctx^R + r_8^R cty^R + r_9^R) z_R = t_{z_R} - t_{z_L}, \quad (6.17)$$

where

$${}^L R = \begin{bmatrix} r_1^L & r_2^L & r_3^L \\ r_4^L & r_5^L & r_6^L \\ r_7^L & r_8^L & r_9^L \end{bmatrix}. \quad (6.18)$$

and

$${}^R R = \begin{bmatrix} r_1^R & r_2^R & r_3^R \\ r_4^R & r_5^R & r_6^R \\ r_7^R & r_8^R & r_9^R \end{bmatrix}. \quad (6.19)$$

The equations (6.15), (6.16) and (6.17) can be written using the matrices, as follows:

$$\mathbf{coef}_z \cdot \begin{bmatrix} \bar{z}_L \\ \bar{z}_R \end{bmatrix} = \begin{bmatrix} t_{x_s} - t_{x_i} \\ t_{y_s} - t_{y_i} \\ t_{z_s} - t_{z_i} \end{bmatrix}, \quad (6.20)$$

where

$$\mathbf{coef}_z = \begin{bmatrix} r_1^L c t x^L + r_2^L c t y^L + r_3^L & -(r_1^R c t x^R + r_2^R c t y^R + r_3^R) \\ r_4^L c t x^L + r_5^L c t y^L + r_6^L & -(r_4^R c t x^R + r_5^R c t y^R + r_6^R) \\ r_7^L c t x^L + r_8^L c t y^L + r_9^L & -(r_7^R c t x^R + r_8^R c t y^R + r_9^R) \end{bmatrix}. \quad (6.21)$$

The problem now is to compute the inverse of matrix  $\mathbf{coef}_z$ . This problem is solved using a special algorithm developed in the reference [PTVF92]. Knowing this inverse matrix, the solution for our system is, as follows:

$$\begin{bmatrix} \bar{z}_L \\ \bar{z}_R \end{bmatrix} = (\mathbf{coef}_z)^{-1} \begin{bmatrix} t_{x_s} - t_{x_i} \\ t_{y_s} - t_{y_i} \\ t_{z_s} - t_{z_i} \end{bmatrix}. \quad (6.22)$$

Using the relations (6.11), (6.12), (6.13) and (6.14) we can compute the values for  $x_L$ ,  $y_L$  and for  $x_R$ ,  $y_R$ . Knowing now the 3D coordinates of the point  $P$  with respect to the cameras frame we can easy compute the 3D coordinates of this point with respect to the world system:

$$\begin{bmatrix} x_W \\ y_W \\ z_W \\ 1 \end{bmatrix} = {}^W_L T \cdot \begin{bmatrix} x_L \\ y_L \\ z_L \\ 1 \end{bmatrix} = {}^W_R T \cdot \begin{bmatrix} x_R \\ y_R \\ z_R \\ 1 \end{bmatrix}. \quad (6.23)$$

In figure 6.1 one can see the measured coordinates of the calibration points situated in a position which is different than the positions used in the calibration procedure.

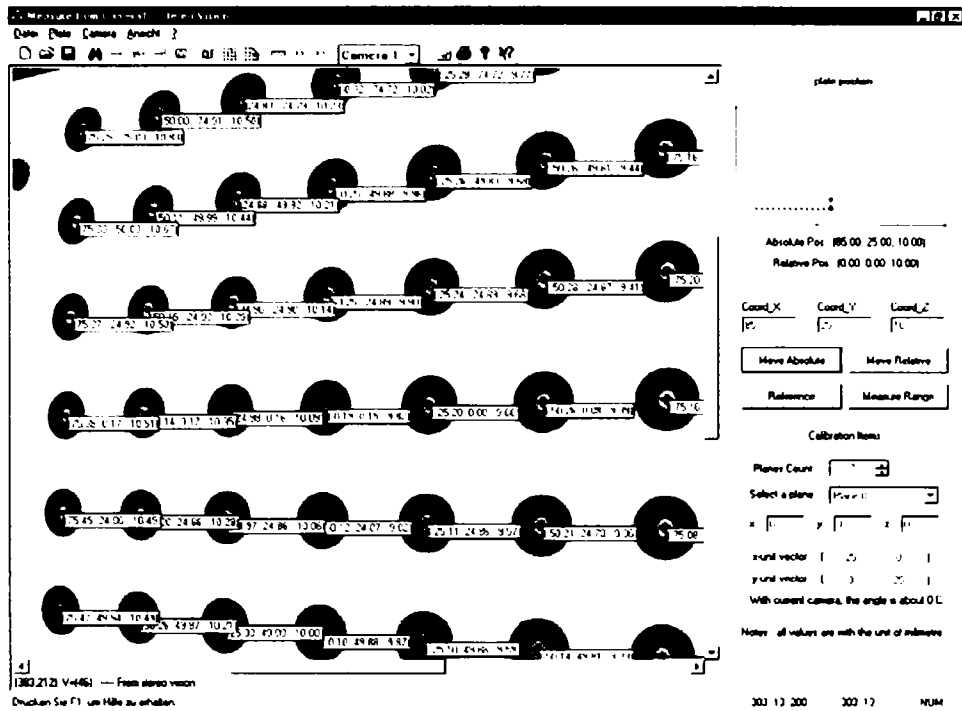


Fig. 6.1. Measured 3D coordinates.

In figure 6.2 one can see the real coordinates of the same points.

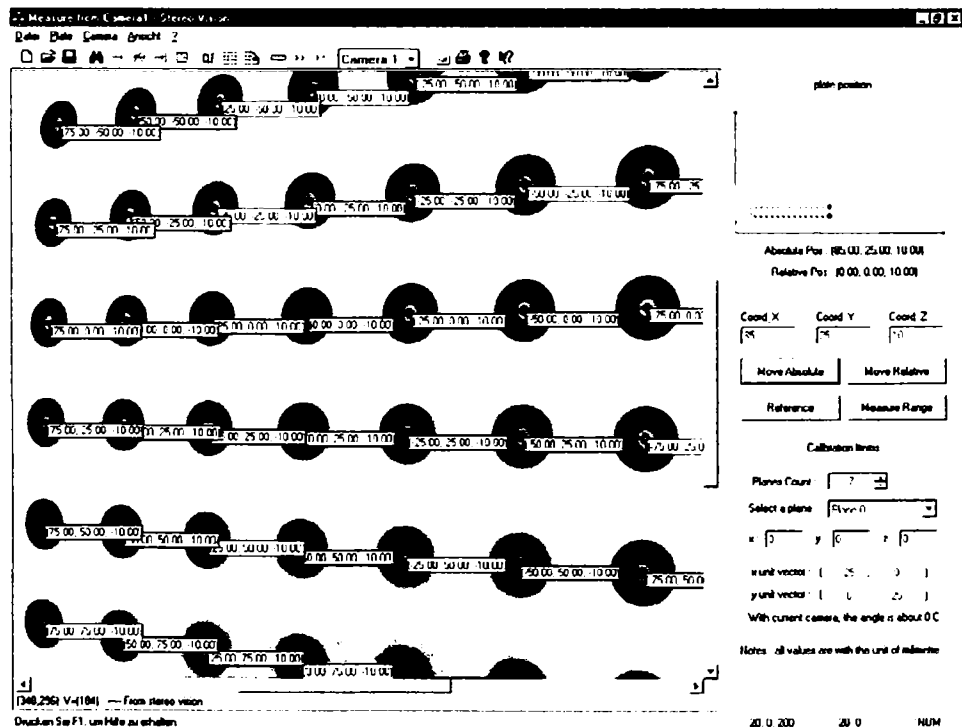


Fig. 6.2. Real 3D coordinates.

In the following parts of this chapter we will discuss the results obtained for the two possible configurations of the stereo sensor: non-parallel configuration in 6.2 and parallel configuration in 6.3.

## 6.2 Non-parallel Configuration

In this sub-chapter we will present first an analysis of the errors obtained with the non-parallel stereo sensor. In the second part a special procedure to reduce these errors will be explained, followed by a new error analysis. All of these are also presented in the reference [NIT02].

### 6.2.1 Analysis of the errors

In figure 6.3 we represented the distribution of the measurement errors for the coordinate  $x$ . The measured points were situated in a plan parallel with the plan defined by the axis  $x$  and  $y$  of the stereo sensor frame. We made this representation because the variation of the errors, if only  $z$  coordinate is changed, is much smaller than the variation of the errors if  $x$  or  $y$  coordinates are changed. The measured points are uniformly placed, on the calibration plate, in a square area with a side equal to 150 mm (see sub-chapter 4.2.1). The coordinates of these points will be measured relative to the stereo sensor frame. The origin of the stereo sensor frame is situated very close to the middle of the square area where the points to be measured are located. This means that the points situated on the edges of this square area will have their  $x$  and  $y$  coordinates approximately  $-75$  mm or  $+75$  mm. To simplify the representation and the calculations we consider the  $x$  and  $y$  coordinates of the measured points to be situated between 0 and 6. This way, 0 will correspond to  $-75$  mm and 6 to  $+75$  mm.

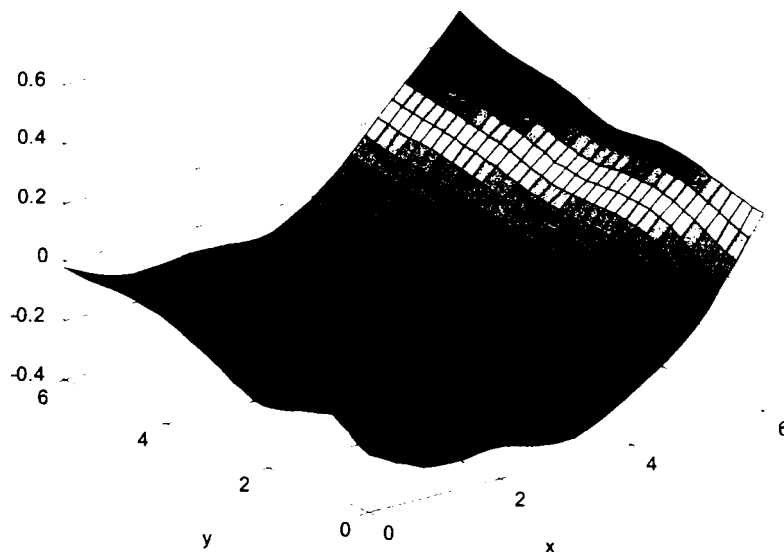


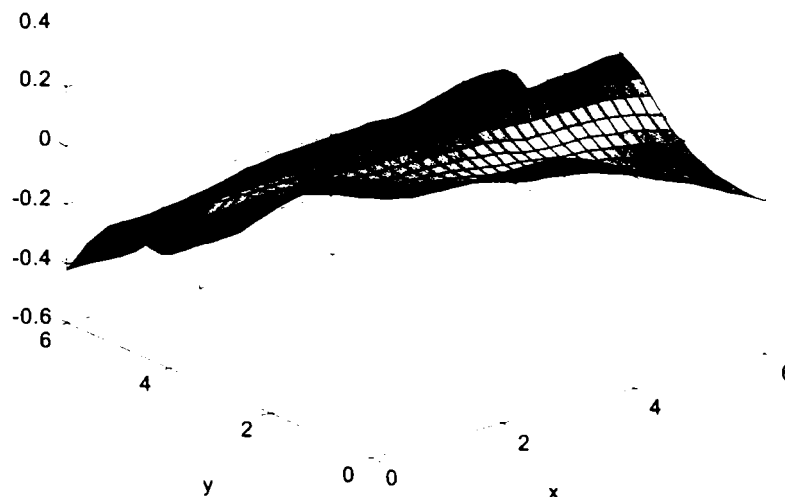
Fig. 6.3. The distribution of the errors for the coordinate  $x$ .

The errors for the coordinate  $x$  are situated between  $-0.33$  mm and  $+0.5$  mm. These errors are computed as the difference between the measured coordinate  $x$  obtained from the



stereo sensor and the real coordinate  $x$  generated by our special calibration device (see subchapter 4.2.1).

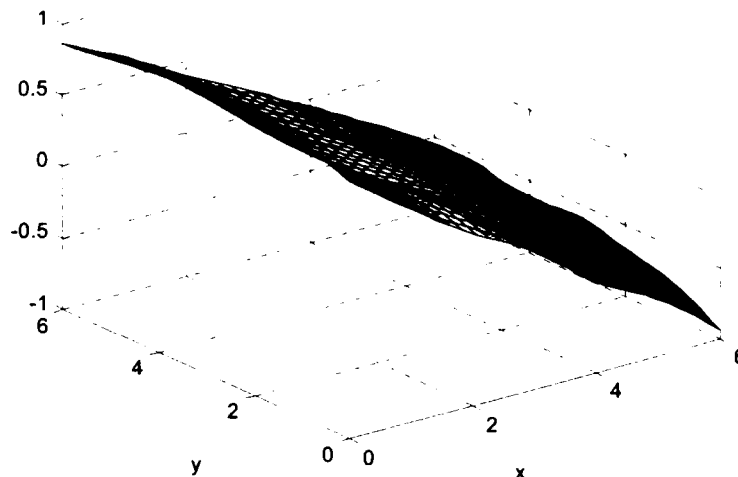
In the figure 6.4 we represented the distribution of the measurement errors for the coordinate  $y$ .



*Fig. 6.4. The distribution of the errors for the coordinate  $y$ .*

The errors for the coordinate  $y$  are situated between  $-0.42$  mm and  $+0.34$  mm.

In the figure 6.5 we represented the distribution of the measurement errors for the coordinate  $z$ .



*Fig. 6.5. The distribution of the errors for the coordinate  $z$ .*

The errors for the coordinate  $z$  are situated between  $-0.95$  mm and  $+0.94$  mm.

## 6.2.2 New method to eliminate the systematic errors

Looking to the graphic from the figure 6.4, it is obvious that a systematic error appear in the measurement process. We will define a correction function  $f_i$ , which has as variables the coordinate  $x_0$  and  $y_0$  of the measured point. One can write the following relation:

$$x = x_0 + f_x(x_0, y_0), \quad (6.24)$$

where  $x$  is the new value for the coordinate  $x$  of the measured point. Our next problem is to find a mathematical relation for the function  $f_x$ , knowing its values in all the 49 measured points represented in figure 6.4. If we look in figure 6.4 to the same  $x$  we can see that the variation of the errors is approximately linear. So, one can write the following relation:

$$f_x(x_0, y) = B_1 y + B_0. \quad (6.25)$$

where  $B_1$  and  $B_0$  must be computed. For the same  $y$  we will approximate the variation of the errors with the following relation:

$$f_x(x, y_0) = A_4 x^4 + A_3 x^3 + A_2 x^2 + A_1 x + A_0. \quad (6.26)$$

where  $A_4$ ,  $A_3$ ,  $A_2$ ,  $A_1$  and  $A_0$  must be computed. This represents in fact a polynomial approximation. With these two relations one can write the function  $f_x$ , as follows:

$$f_x(x, y) = A_4 x^4 + A_3 x^3 + A_2 x^2 + A_1 x + B_1 y + C_0. \quad (6.27)$$

To find all the coefficients we use the least squares method. We start from the next relation:

$$\sum_{i=0}^6 \sum_{j=0}^6 (A_4 x_i^4 + A_3 x_i^3 + A_2 x_i^2 + A_1 x_i + B_1 y_j + C_0 - f_i(x_i, y_j))^2 = \min. \quad (6.28)$$

We will note this sum with  $S$ . Going further, one can write for each coefficient the next relation:

$$\frac{\partial S}{\partial C_i} = 0. \quad (6.29)$$

This way, we will obtain a linear system composed from six equations and having six unknowns. We solve the system and we obtained the following correction function:

$$f_x(x, y) = 0.0016x^4 - 0.0109x^3 + 0.0238x^2 - 0.0167x + 0.038y - 0.2244. \quad (6.30)$$

Using this correction function the new distribution of the measurement errors for the coordinate  $x$  is as one can see in figure 6.6.

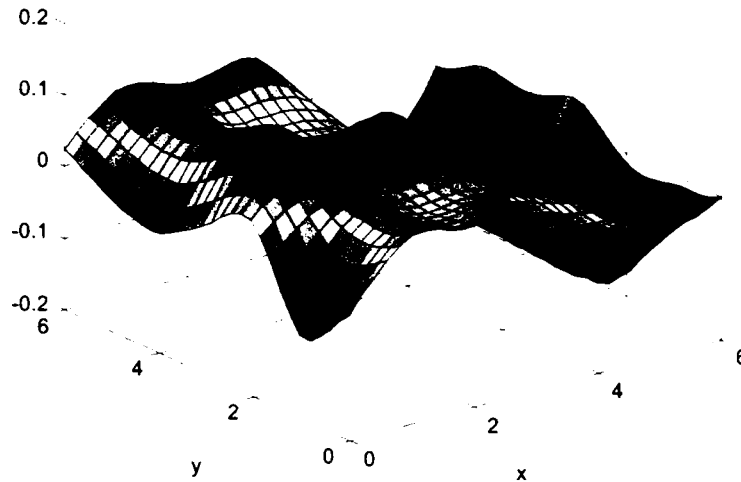


Fig. 6.6. The distribution of the errors for the coordinate  $x$  after correction.

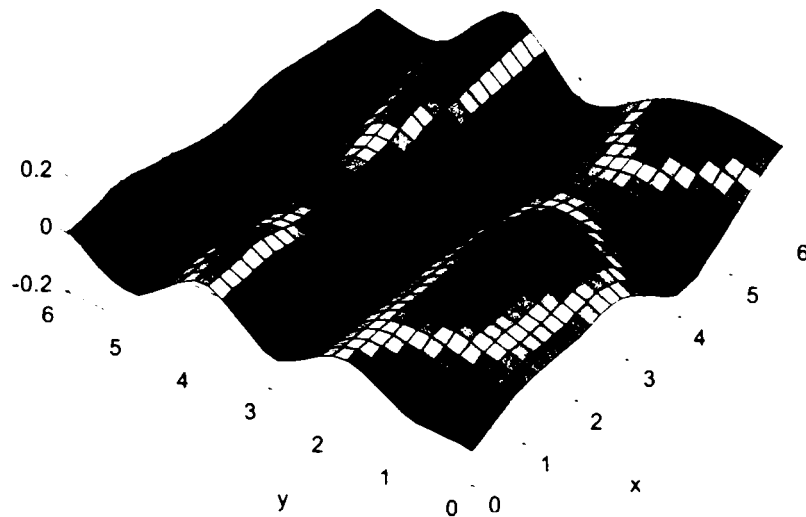
The errors of the coordinates  $x$  are now situated between  $-0.125$  mm and  $+0.125$  mm. These errors are more than 3 times smaller than in the case when no correction was used (see figure 6.3). It is also obvious that the systematical errors, which clearly appear in figure 6.3, became insignificant after correction, as one can see in figure 6.6. The distribution of the errors, presented in figure 6.6, includes also the effect of the uncertainty of the calibration device (see 4.2.1).

We will apply the same procedure for the coordinates  $y$  and  $z$  to reduce the systematical errors. The correction function will be as follows:

$$f_y(x, y) = \begin{cases} -0.058y + 0.142 & x \in [0,3], y \in [0,3) \\ 0.03y + 0.02 & x \in (3,6], y \in [0,3). \\ 0.027x - 0.123y + 0.49 & x \in [0,6], y \in [3,6] \end{cases} \quad (6.31)$$

In this case it was not possible to approximate all the measurement area with only one function, so we had to divide this area in three sub-areas.

Using this correction function the new distribution of the measurement errors for the coordinate  $y$  is as one can see in figure 6.7.



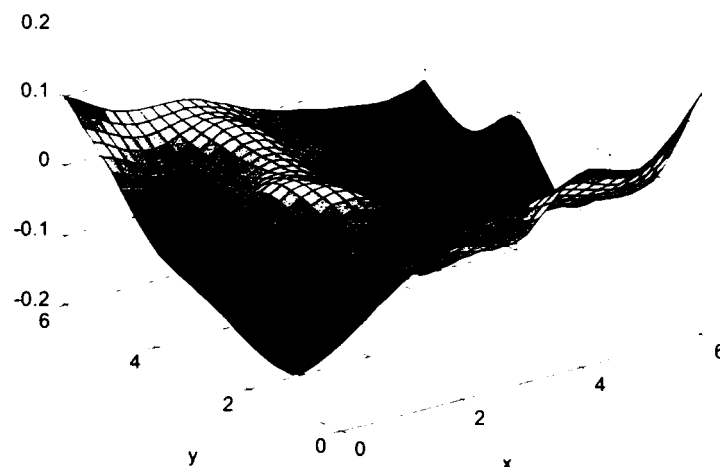
*Fig. 6.7. The distribution of the errors for the coordinate  $y$  after correction.*

The errors after correction are situated now between  $-0.12$  mm and  $+0.12$  mm.

For the coordinate  $z$  we use the following correction function:

$$f_z = -0.256x + 0.036y - 0.79. \quad (6.32)$$

Applying this correction function the new distribution of the measurement errors for the coordinate  $z$  is as one can see in figure 6.8.



*Fig. 6.8. The distribution of the errors for the coordinate  $z$  after correction.*

The errors are situated now between  $-0.149$  mm and  $+0.149$  mm. So, the errors range was reduced more than 6 times.

In the table 6.1, one can see better the obtained improvement of the stereo sensor after the correction functions were applied.

*Table 6.1. A comparison between the errors before and after the correction function was applied*

Errors		Min. (mm)	Max. (mm)	Range (mm)	Ratio(before/after)
x	before correction	-0.33	0.50	0.83	<b>3.32</b>
x	after correction	-0.13	0.13	0.25	
y	before correction	-0.42	0.34	0.76	<b>3.17</b>
y	after correction	-0.12	0.12	0.24	
z	before correction	-0.95	0.94	1.89	<b>6.34</b>
z	after correction	-0.15	0.15	0.30	

### 6.3 Parallel configuration

In the following parts of this sub-chapter we will make an analysis of the errors of the measurements obtained with a stereo sensor built in parallel configuration. We will present four cases. In the first case we used Lenz calibration method (see 3.3.2) to calibrate the cameras combined with our best image-processing algorithm (see 5.2.3). In the other three situations we used our calibration method (see 4.2.2), but for the image processing algorithms, three different types: algorithm developed only with the functions from the dedicated software (see 5.2.2), algorithm based on the weighted point of a circle (see 5.2.2), and algorithm which uses the sub-pixel approach. **The last one represents our best image-processing algorithm.**

Going further, for each case we will present four graphics. The first three graphics show the errors distribution obtained for the coordinates  $x$ ,  $y$ ,  $z$  and the last one the errors of the position vector  $v$ . The error of the position vector was computed using the next relation:

$$\Delta v = \sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2} . \quad (6.33)$$

We have measured 25 points situated in a plan. In our graphics the coordinates  $x$  and  $y$  indicate the position of the measured point in this plan and the coordinate  $z$  indicates step by step the four errors presented before.

As we said before, the first group of four graphics presents the errors obtained when we used only the Lenz method to calibrate the stereo sensor. In figure 6.9, we represented the errors for the coordinate  $x$ . These errors are situated between  $-0.117$  mm and  $+0.165$  mm.

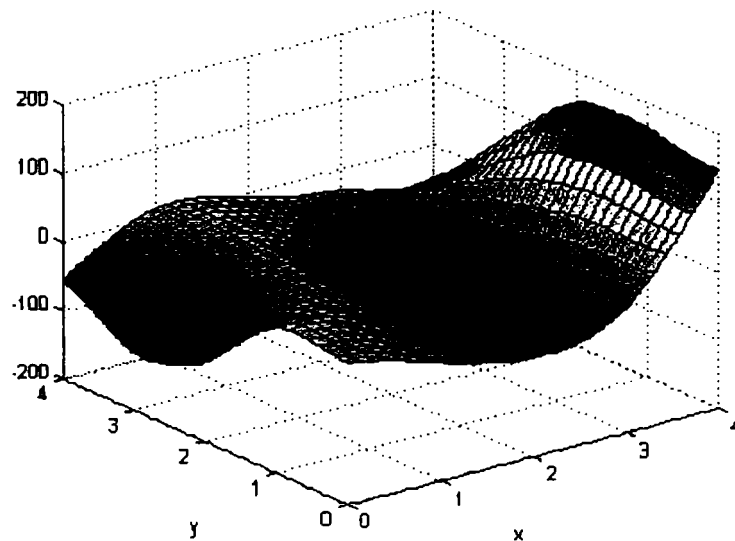
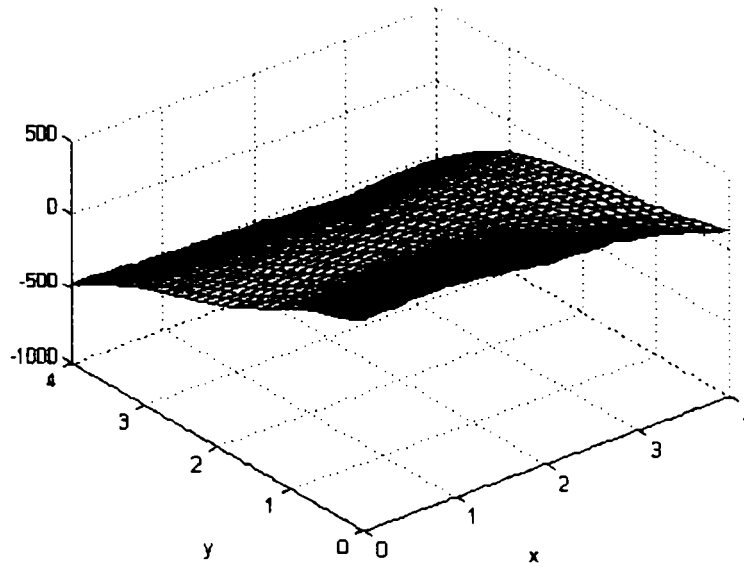


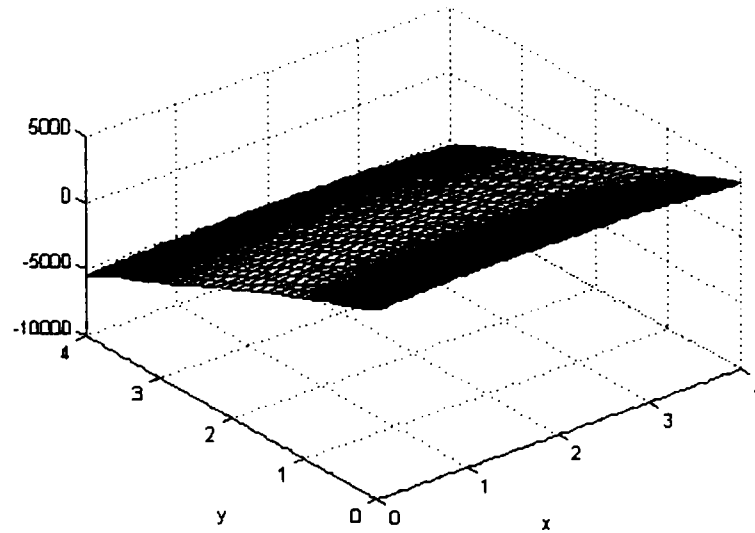
Fig. 6.9. The distribution of the errors for the coordinate  $x$ .

In figure 6.10, we represented the errors for the coordinate  $y$ . These errors are situated between  $-0.553$  mm and  $+0.437$  mm.



*Fig. 6.10. The distribution of the errors for the coordinate  $y$ .*

In figure 6.11, we represented the errors for the coordinate  $z$ . These errors are situated between  $-5.516$  mm and  $+4.469$  mm.



*Fig. 6.11. The distribution of the errors for the coordinate  $z$ .*

In figure 6.12, we represented the errors for the position vector  $\Delta v$ . These errors are smaller than 6 mm.

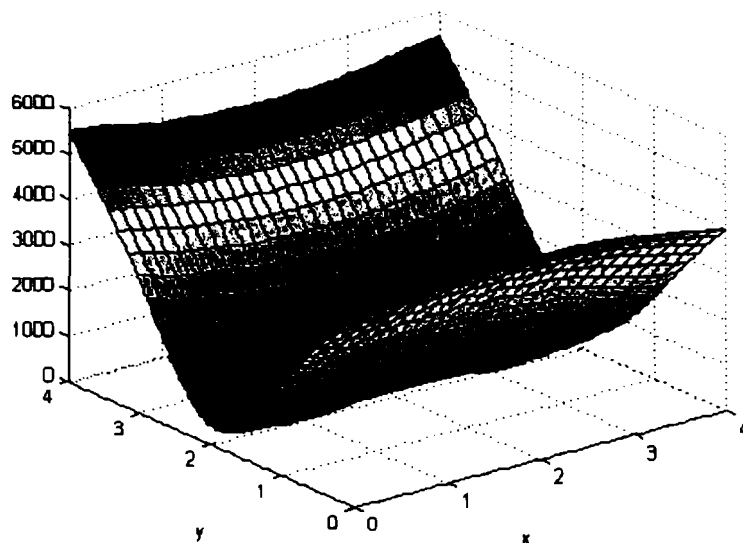


Fig. 6.12. The distribution of the errors for the position vector.

We can see from these first four graphics that the errors are very high. This is due to the fact that we used a model for the camera in which the scale factors and the image center are not calibrated. Only approximate values are used. Also, the effect of the distortion is not considered. The calibration points are situated only in one plan not in a 3D space, as it will be considered in our final calibration method.

The second group will show the errors obtained when the camera are calibrated with our calibration method, using in the image processing algorithm only the functions from the dedicated software.

In figure 6.13, we represented the errors for the coordinate  $x$ . These errors are situated between  $-17 \mu\text{m}$  and  $+37 \mu\text{m}$ .

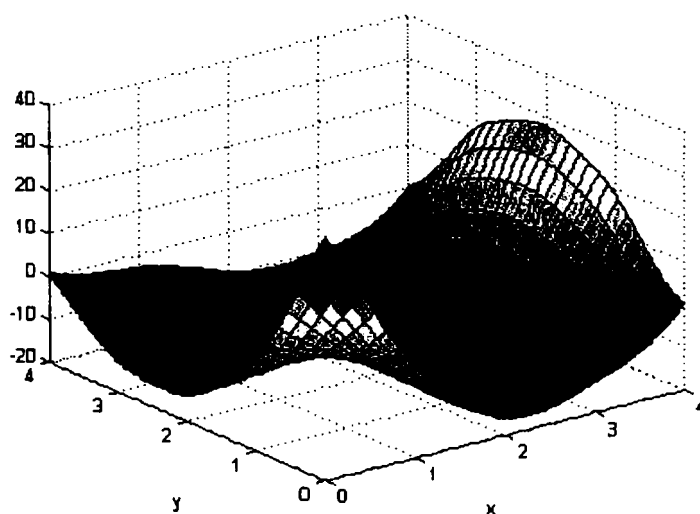


Fig. 6.13. The distribution of the errors for the coordinate  $x$ .



In figure 6.14, we represented the errors for the coordinate  $y$ . These errors are situated between  $-116 \mu\text{m}$  and  $+88 \mu\text{m}$ .

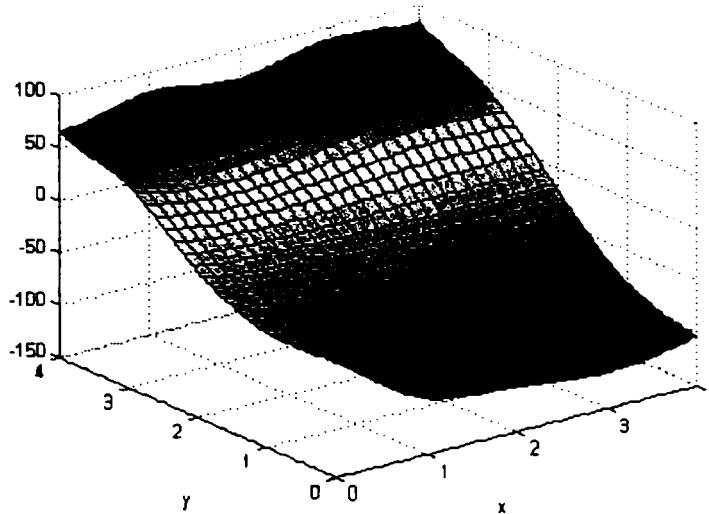


Fig. 6.14. The distribution of the errors for the coordinate  $y$ .

In figure 6.15, we represented the errors for the coordinate  $z$ . These errors are situated between  $-295 \mu\text{m}$  and  $+47 \mu\text{m}$ .

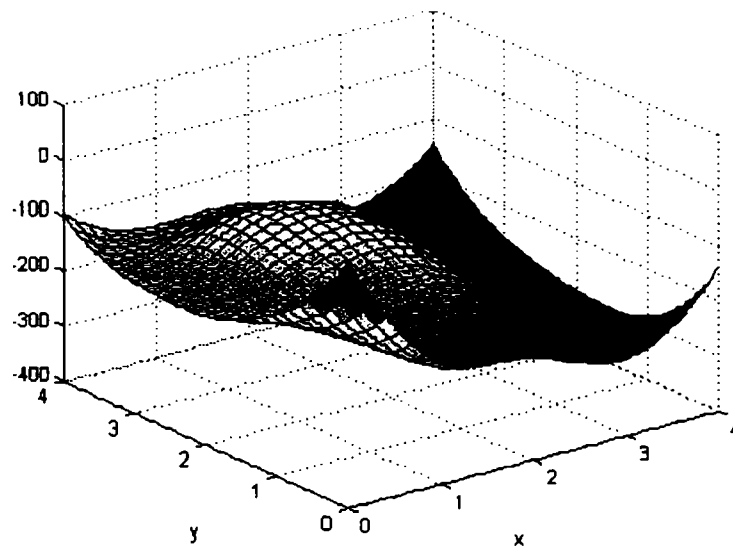
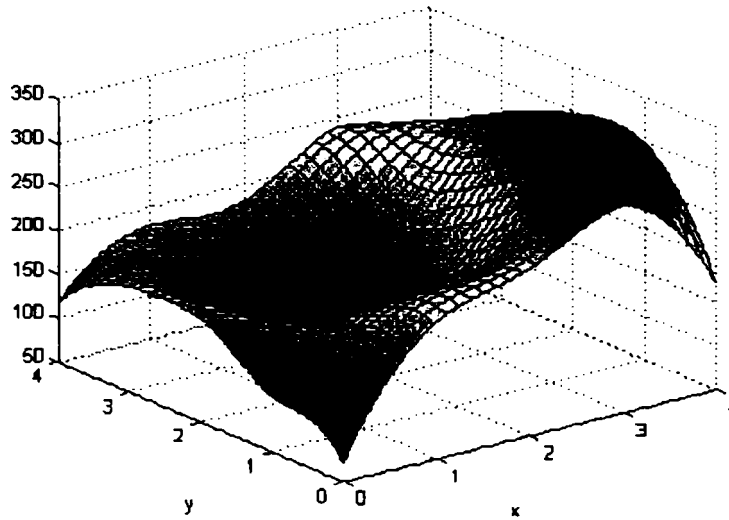


Fig. 6.15. The distribution of the errors for the coordinate  $z$ .

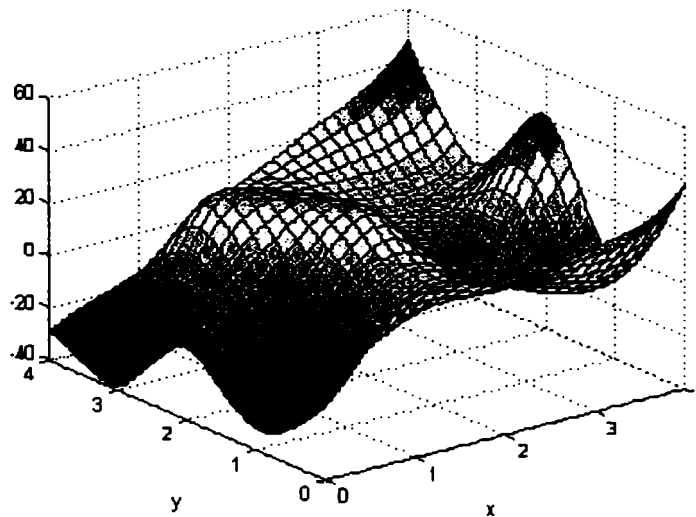
In figure 6.16, we represented the errors for the position vector  $\Delta v$ , defined by the relation (6.33). These errors are smaller than  $350 \mu\text{m}$ .



*Fig. 6.16. The distribution of the errors for the position vector.*

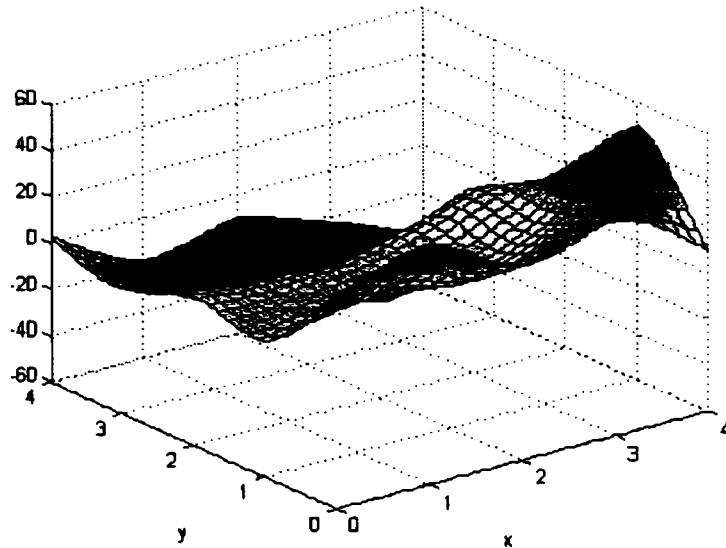
The next four graphics are made for the method where we calculated the weighted point of a circle. As we will see, the errors will be situated almost in the same range as for the method presented before.

In figure 6.17, we represented the errors for the coordinate  $x$ . These errors are situated between  $-39 \mu\text{m}$  and  $+47 \mu\text{m}$ .



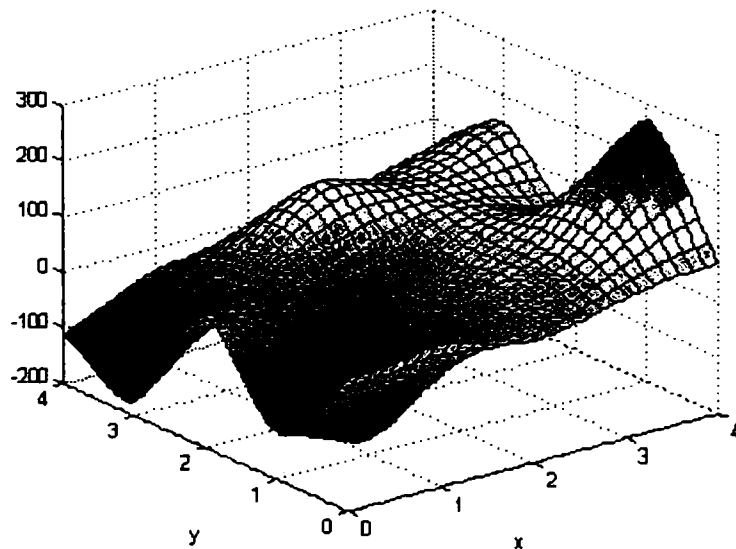
*Fig. 6.17. The distribution of the errors for the coordinate x.*

In figure 6.18, we represented the errors for the coordinate  $y$ . These errors are situated between  $-41 \mu\text{m}$  and  $+50 \mu\text{m}$ .



*Fig. 6.18. The distribution of the errors for the coordinate  $y$ .*

In figure 6.19, we represented the errors for the coordinate  $z$ . These errors are situated between  $-175 \mu\text{m}$  and  $+272 \mu\text{m}$ .



*Fig. 6.19. The distribution of the errors for the coordinate  $z$ .*

In figure 6.20, we represented the errors for the position vector  $\Delta v$ . These errors are smaller than  $300 \mu\text{m}$ .

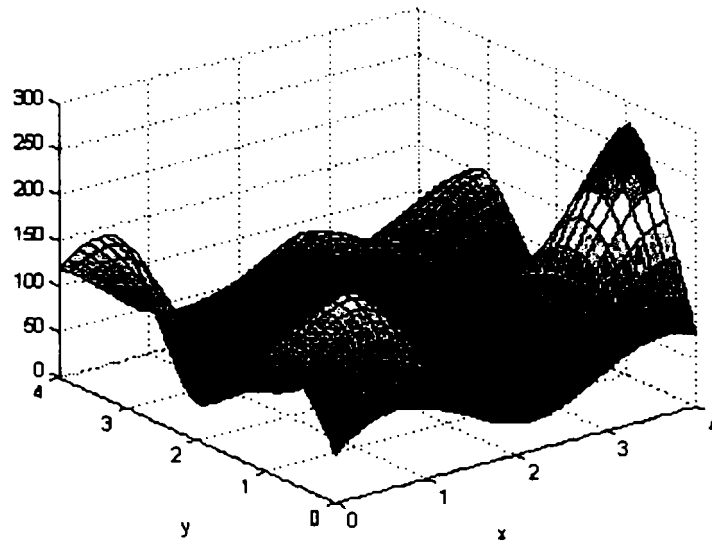


Fig. 6.20. The distribution of the errors for the position vector.

The last graphics are made for the method where we used the sub-pixel algorithm. This method was presented detailed in sub-chapter 5.2.3. As we said in 5.2.3, the problem was that the border between white and black is in most of the cases located somewhere between two adjacent cells of the CCD camera, respectively somewhere between two adjacent pixels of an image. So, in order to determine the exact position of this border a sub-pixel approach is needed. This is the best method and we will obtain the smallest errors, as we will see in the following part.

In figure 6.21, we represented the errors for the coordinate  $x$ . These errors are situated between  $-19 \mu\text{m}$  and  $+24 \mu\text{m}$ .

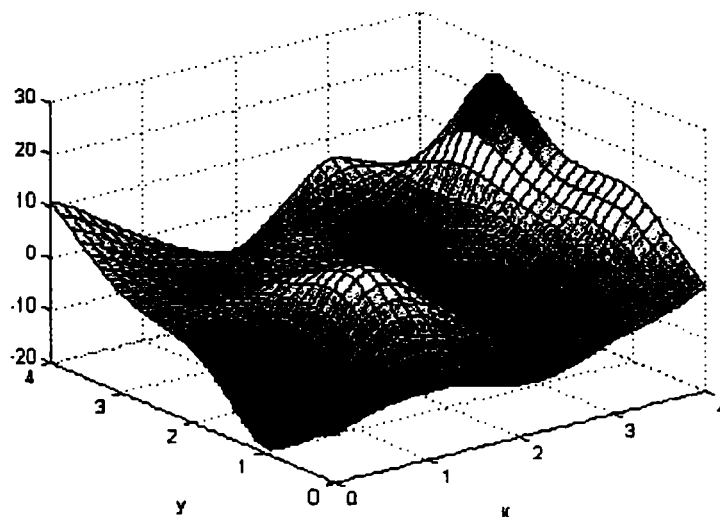
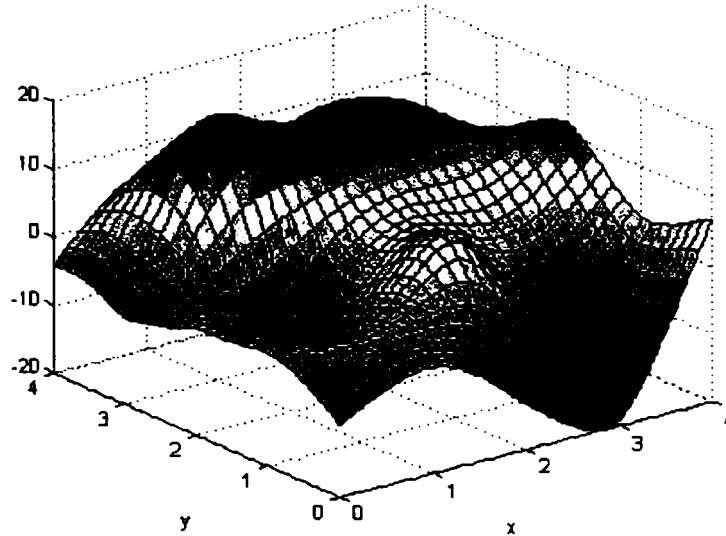


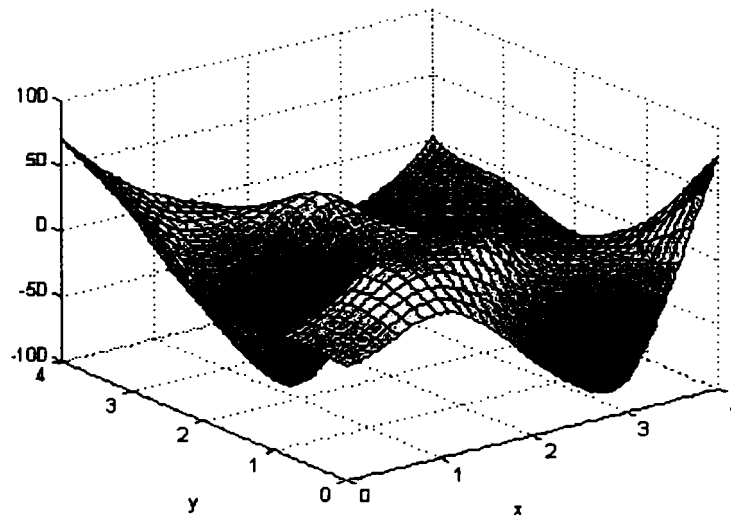
Fig. 6.21. The distribution of the errors for the coordinate  $x$ .

In figure 6.22, we represented the errors for the coordinate  $y$ . These errors are situated between  $-20 \mu\text{m}$  and  $+17 \mu\text{m}$ .



*Fig. 6.22. The distribution of the errors for the coordinate  $y$ .*

In figure 6.23, we represented the errors for the coordinate  $z$ . These errors are situated between  $-91 \mu\text{m}$  and  $+79 \mu\text{m}$ .



*Fig. 6.23. The distribution of the errors for the coordinate  $z$ .*

In figure 6.24, we represented the errors for the position vector  $\Delta v$ . These errors are smaller than  $100 \mu\text{m}$ .

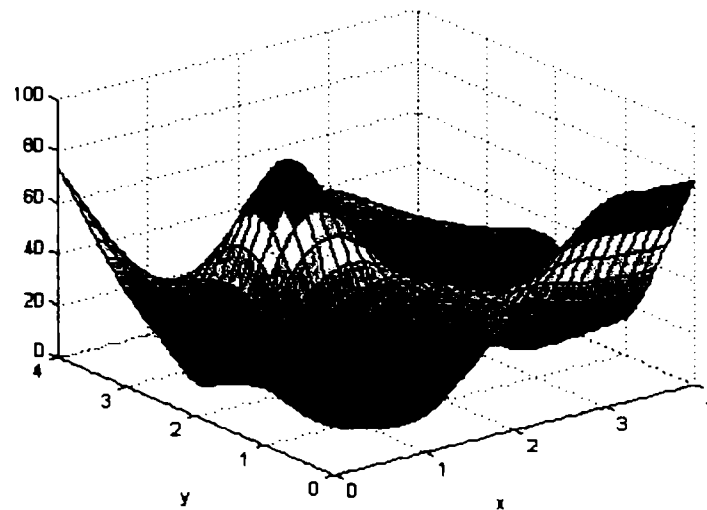


Fig. 6.24. The distribution of the errors for the position vector.

All these graphics were obtained using MATLAB programs. The measurement data obtained from the stereo sensor were used in these programs to generate the graphics discussed before. These programs are presented in Annex E.

In the table 6.2 we present a brief comparison between all the four cases presented before starting with figure 6.9 and finishing with figure 6.24.

Table 6.2. A brief comparison between all the cases analyzed before for the parallel sensor

Errors		Min. ( $\mu\text{m}$ )	Max. ( $\mu\text{m}$ )	Range ( $\mu\text{m}$ )	Ratio ( $i/(i+1)$ )
x	Lenz Method +sub-pixel	-117	165	282	
x	Our Method + software	-17	37	54	5.22
x	Our Method + weight point	-39	47	86	0.63
x	Our Method + sub-pixel	-19	24	43	<b>2.00</b>
y	Lenz Method +sub-pixel	-553	437	990	
y	Our Method + software	-116	88	204	4.85
y	Our Method + weight point	-41	50	91	2.24
y	Our Method + sub-pixel	-20	17	37	<b>2.46</b>
z	Lenz Method +sub-pixel	-5516	4469	9985	
z	Our Method + software	-295	47	342	29.20
z	Our Method + weight point	-175	272	447	0.77
z	Our Method + sub-pixel	-91	79	170	<b>2.63</b>
$\Delta w$	Lenz Method +sub-pixel	0	6000	6000	
$\Delta w$	Our Method + software	0	350	350	17.14
$\Delta w$	Our Method + weight point	0	300	300	1.17
$\Delta w$	Our Method + sub-pixel	0	100	100	<b>3.00</b>

# Industrial Applications

In industrial applications, a stereo sensor can be used in two configurations: as a fixed sensor, references [KK99], [JT02], and [TSIN04] or, as a mobile sensor mounted on the robot hand, references [DK02], and [iK98].

The first configuration can be employed in measuring the angle between the axles of a vehicle and the plane in which the wheels are rotating. The accuracy in such applications has to be very high. In sub-chapter 7.1 is presented a measurements system based on stereo sensors, which provides the required accuracy. This system can replace the current solution, which uses very expensive laser devices.

The second configuration, mobile sensor, is found useful in automatic processes, such as robotic hands mounting of windows for passenger cars. Here as well, this solution with a stereo sensor mounted on the robot hand can replace, with better results, the current solution. It needs only two cameras instead of four or eight, which are needed for the multi-camera method, which is presently used. In sub-chapter 7.2 is presented a test application, which makes use of a mobile stereo sensor.

## 7.1 Fixed Sensor Configuration

The wheel alignment problem is an important task and concerns all car producers. There were developed a lot of measurement systems to be used for solving this problem. At the beginning there were produced systems based only on mechanical methods. The disadvantage of these methods was the time for measuring which was too long. Also, the accuracy of the measured results was influenced by the errors of the tire surfaces. The second step was to build measurement systems, which use both mechanical and optical methods for measuring. In this category we have systems based on laser technology and systems, which use cameras. The first ones have the disadvantage that they are very expensive. For the second type the accuracy is the task that must be improved. It is also important to know that the systems, which use cameras, can be based on the multi-camera concept or stereo camera concept. We used the second possibility.

In the following parts we define and then explain the two angles we want to measure with our vision system. Camber is the inward or outward tilt of the wheel measured from top to bottom. This angle is adjusted to prevent excessive tire deterioration and to enhance straight ahead stability. It is measured in degrees and has several methods of adjustment. In figure 7.1, one can understand better the definition of this angle. In this figure, there are presented three possible situations for this angle: positive Camber, negative Camber and zero Camber.

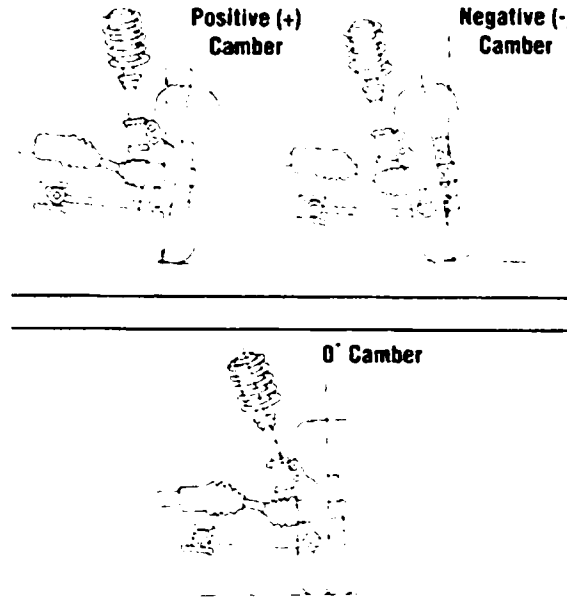


Fig. 7.1. Definition of Camber.

The angle formed by a horizontal line through the plane of one wheel versus a perpendicular line to the centerline is called the individual toe. This is the most critical tire angle. When a horizontal line is drawn through the plane of each wheel, and they intersect in front of the wheels, this is called toe-in or positive toe. When they intersect behind the wheels, this is called toe-out or negative toe. In figure 7.2, one can understand better the definition of this angle.

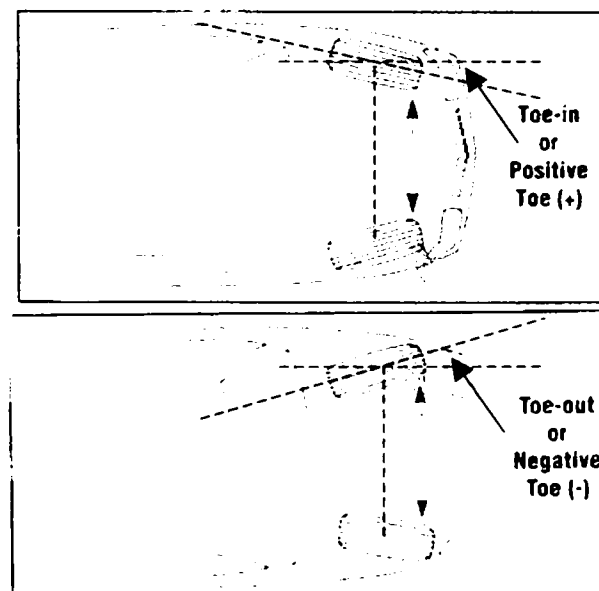


Fig. 7.2. Definition of Toe.

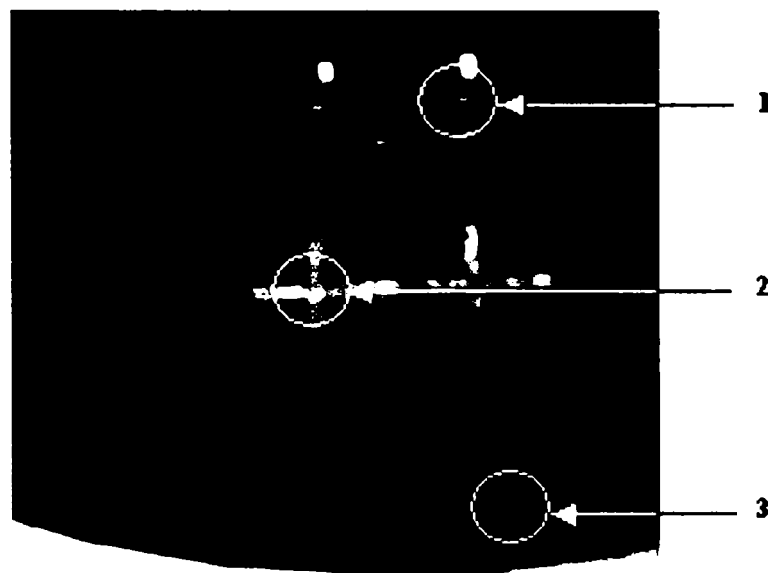


In figure 7.3, one can see the stereo sensor and the light projector used to build the required measurement system. For the stereo sensor we used the parallel configuration as it was presented in sub-chapter 4.1.3 and 6.3.



*Fig. 7.3. Stereo sensor and light projector.*

The light projector is used to create on the tire surface, some marks, which could be further measured with the stereo sensor. In figure 7.4, one can see the shape of the structured light created by the light projector on the tire surface.



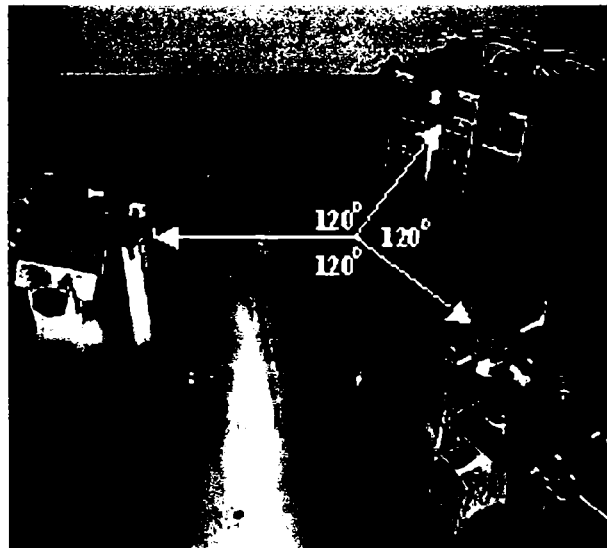
*Fig. 7.4. Structured light projected on the tire surface.*

There are two possibilities to make use of this structured light. First one is to use as marks the intersections between the light and different forms existing on the tire surface. As one can see in figure 4, in this category are included points 1 and 3. The second one is to use as marks the crosses defined by the structured light itself on the tire surface. To this category belongs point 2.

Having this information about the angles to be measured and the tools to be used, in the next step we have to define a mathematical model for the wheel in order to be able to measure these angles.

First of all we define a coordinate frame for the wheel. We call this, the wheel frame. The origin of this frame is situated in the middle of the tire. Axe  $z$  is perpendicular to the tire so that the plane determined by axes  $x$  and  $y$  is parallel to the tire. Axe  $x$  is horizontal. One can see all these details in figure 7.7. With these notations, Camber is determined by measuring the rotation of the wheel frame around  $x$  axe and Toe is given by the measured value of the rotation of the wheel frame around  $y$  axe.

The angle information we need is obtained by knowing the orientation of the tire plane (the plane defined by axes  $x$  and  $y$ ) relative to a reference plane. So, the task is to measure this tire plane. It is known that a plane is determined by at least three points, which are not all situated on the same line. Starting from the plane definition we decided to use three stereo sensors placed on a circle at equal relative distances between them, as one can see in figure 7.5.



*Fig. 7.5. Description of the measurement system.*

We explained in chapter 4 that in the calibration procedure of the stereo sensor is defined a stereo sensor frame, reference [TSNI04]. It means, the coordinates of the points measured with a calibrated stereo sensor are given relative to its defined stereo sensor frame. The three stereo sensors, which are fixed on a rigid plate, as one can see in figure 7.5, are first calibrated (see chapter 4). This means, each one has its own frame. The next step is to find the relative position and orientation of these three frames with respect to a reference frame. This is in fact the calibration procedure of our measurement system.

In figure 7.6, one can see the calibration plate we have used in order to compute the position and orientation of the stereo sensors frames with respect to the reference frame. We

denoted with  $S_R$  the reference frame situated in the middle of the plate and with  $S_{S1}$ ,  $S_{S2}$  and  $S_{S3}$  the stereo sensors frames. In figure 7.6, it is drawn only one sensor frame, because the situation is similar for the other two. The mathematical explanation, which follows for one sensor, will be applied in the same way for the other two sensors. With  $T(S_R-S_{Si})$  we denoted the transformation from the reference frame to one sensor frame.

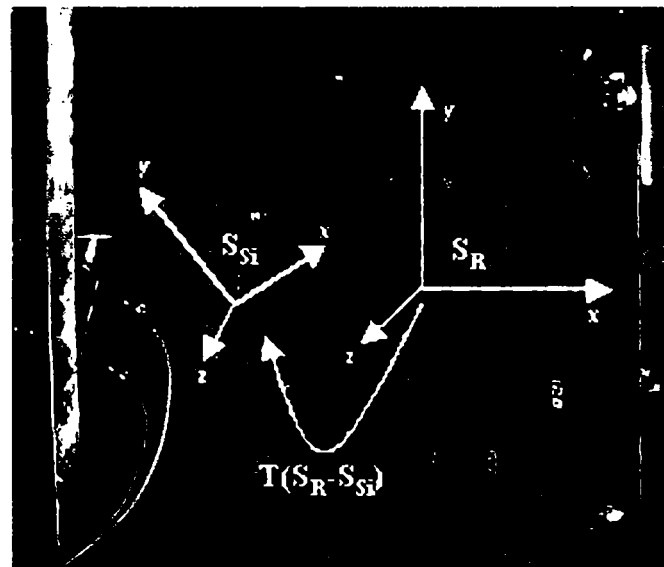


Fig. 7.6. Calibration of the measurement system.

The calibration plate, we used, has 121 points and we know very precisely their position with respect to the reference frame. We denote the coordinates of one point from this plate with  $x_R$ ,  $y_R$  and  $z_R$ . The same point will be measured with the stereo sensor and we obtain the coordinates  $x_{Si}$ ,  $y_{Si}$ , and  $z_{Si}$ . According to the reference [Pau81], between these coordinates we have the following relation:

$$\begin{pmatrix} x_R & y_R & z_R & 1 \end{pmatrix}^T = \begin{matrix} S_R \\ S_{Si} \end{matrix} \mathbf{T} \cdot \begin{pmatrix} x_{Si} & y_{Si} & z_{Si} & 1 \end{pmatrix}^T. \quad (7.1)$$

Using more than four points for each sensor we obtain an over determinate system of nonlinear equations. According to the references [Man81], [Nas99] and [PTVF92] such systems are solved in two steps. First step, we make the system linear and second step we use least square methods to find the solution of the system.

Going further there are two different methods of measuring. Until now, we have implemented in practice only one method and obtained test results, which are presented at the end of this sub-chapter.

The method, which we have implemented, is based on identifying marks of type noted with 3, as one can see in figure 7.4. This method used the fact that on the surface of the tire there are several profiles, which modify the shape of the structured light projected on it.

Because these shape modifications are very small we had to develop image processing algorithms to provide us enough and accurate information. We have used sub-pixel accuracy and segmentation methods according to the references [GLP99], [Par97] and [TSIN04].

Our goal is to identify points, which are in the same plane and of course this plane must be parallel to the tire plane. As one can see in figure 7.7, there are some profiles having circle shape on the surface of the wheel. The big advantage for these circles is that they define, at least theoretically, each one a plane, which is parallel to the tire plane.

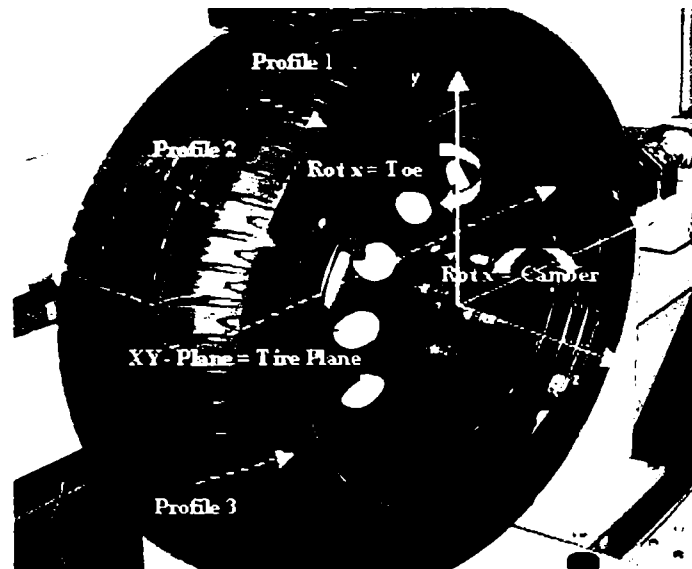


Fig. 7.7. Explanations for the measurement procedure.

The structured light allows us to take for each stereo sensor maximum three points per circle. This way, we can use maximum nine points to compute the plane where the circle is situated. Using the best-fit method we eliminate from these points those, which have big errors and finally, compute a plane parallel with the tire plane. Having this plane, we can compute the values for Camber and Toe.

The second method is based on identifying marks of type noted with 2, as one can see in figure 7.4. The idea is to use the light crosses for identifying which pixel from the image obtain with one camera of the sensor corresponds to a certain pixel from the image obtained with the other camera. This way, we can measure the 3D coordinates for a lot of points belonging to the light lines projected on the tire. With this information the next step is to calculate the tire plane and its orientation relative to a reference plane.

To test our system we use a special device having a wheel and the possibility to adjust it at different angles between  $-3$  and  $3$  degrees for both Camber and Toe. Before we start a normal measurement the wheel is fixed so that, the special device indicates 0 for both Camber and Toe. For this position, we make a zero measurement. It means all the measurements, which follow to this zero measurement, are done relative to this zero wheel frame.

For the diagrams, which follow we have measured ten different orientations of the wheel, but sometimes keeping one angle fixed. On the horizontal scale we have represented the real value of the angle in degrees. On the vertical scale we have represented the difference between the measured value and the real value of the angles. The unit used for this difference is the minute.

In figure 7.8, one can see the distribution of the errors for Camber. They are situated between 0.40 and 4.06 minutes. It is important to mention that for the same Camber angle we obtained different errors, because the measurements system is unfortunately influenced by the value of the Toe angle.

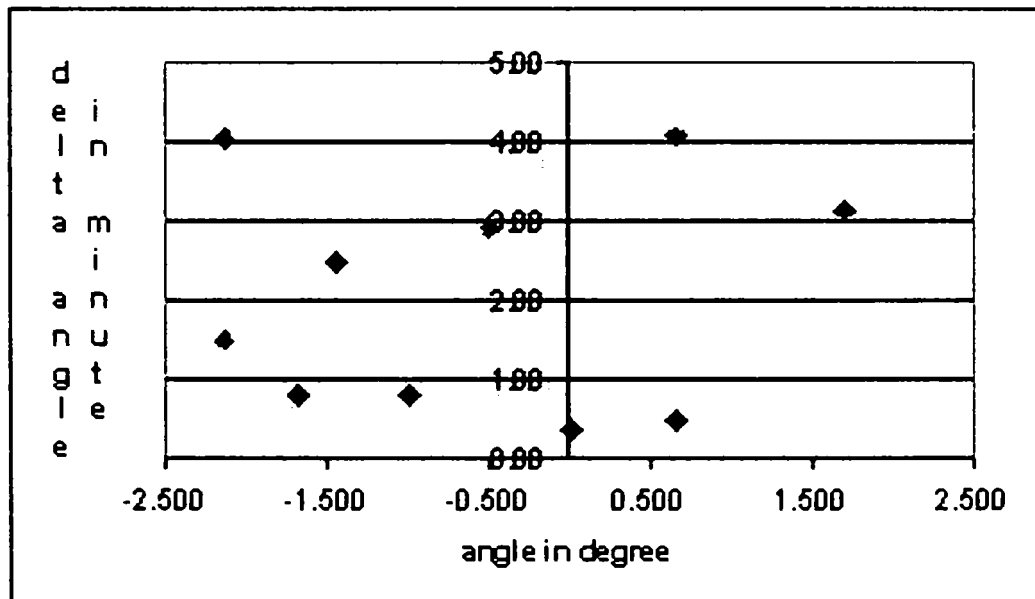


Fig. 7.8. Distribution of the errors for Camber.

In figure 7.9, one can see the distribution of the errors for Toe. They are situated between -0.38 and 4.86 minutes. Also, here we obtained different errors for the same Toe angle due to the influence of the Camber angle to the measurement system.

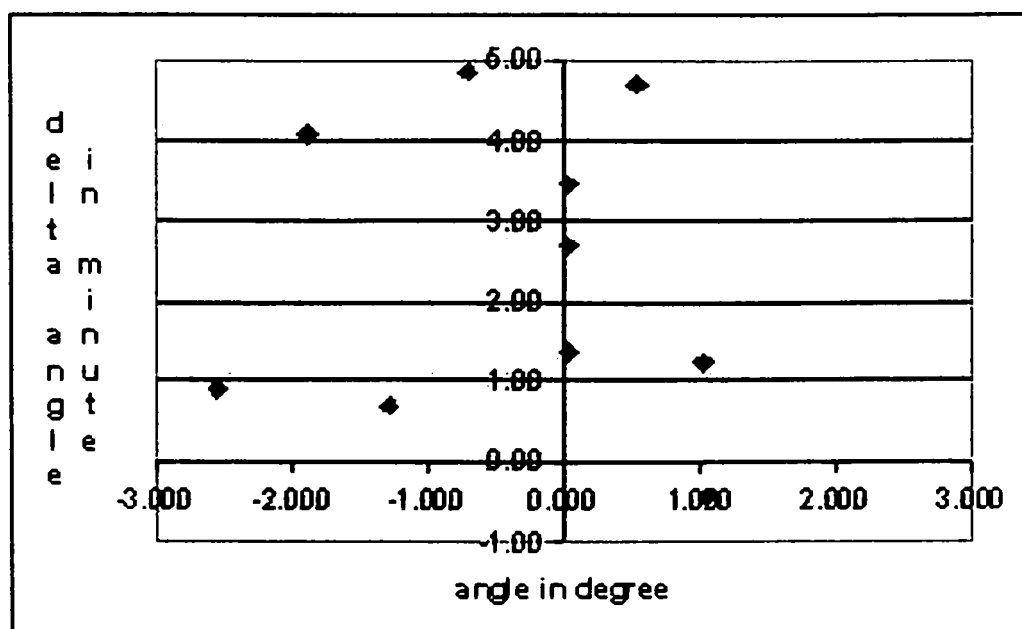


Fig. 7.9. Distribution of the errors for Toe.

Summarizing one can say that we have succeeded to build a vision system using simple methods and cheap components with a good accuracy. These first results obtained in the measurement procedure confirm us the fact that our vision system could be further developed and improved. Using the second method, improving the quality of the structured light and developing better image processing algorithms and mathematical algorithms we will be able to reach the accuracy of 0.1 minutes with our vision system.

## 7.2 Mobile Sensor Configuration

In this sub-chapter a test application, which use a stereo sensor in a mobile configuration will be presented. The stereo sensor will be mounted on the robot tool. The robot will be driven in different positions in such a way that certain parts of an object can be viewed and measured. In the following parts we will describe both theoretically and practically the test application we have developed.

First step is to establish a frame for the object whose position we want to measure. The robot tool is moved in a position near the object, position 0, as one can see in figure 7.10. We consider the robot tool frame for this position of the robot tool as being the object frame. We consider also this frame as being the reference frame for the next measurements. We measure four fixed points from the object using the stereo sensor. These points are noted 1, 2, 3 and 4, as one can see in the figure 7.10. We will obtain for each point its 3D coordinates with respect to the reference frame, which is the same with the object frame. In this moment the object is well defined by an object frame and four points whose 3D coordinates are known with respect to the object frame. For each point  $i$  we denote its coordinates  $^{Obj}P_i$ .

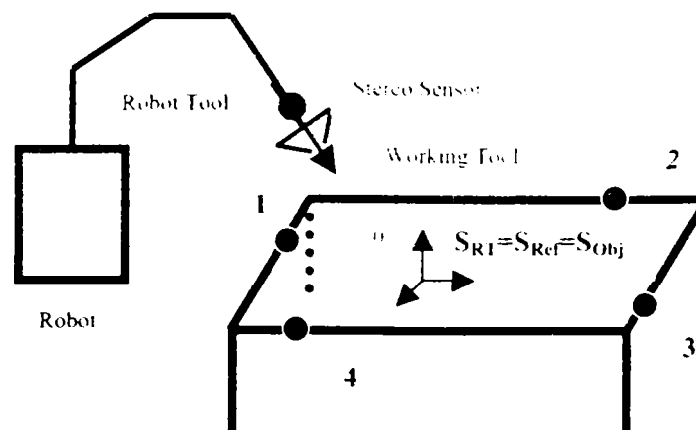


Fig. 7.10. The object frame definition.

According to the explanations we can write the following relations:

$$S_{RT} = S_{Ref} = S_{Obj} \quad (7.2)$$

$${}^{Ref} P_i = {}^{Obj} P_i \quad (7.3)$$

In the figure 7.10 one can see the practical results obtained when we introduced one test object. The box named object frame represents the transformation from the robot tool frame to the robot base frame. By pressing the button Set Object Frame we set this frame to be our reference frame for the next measurements. In the box Relative Coords we have the coordinates of the measured point with respect to reference frame.

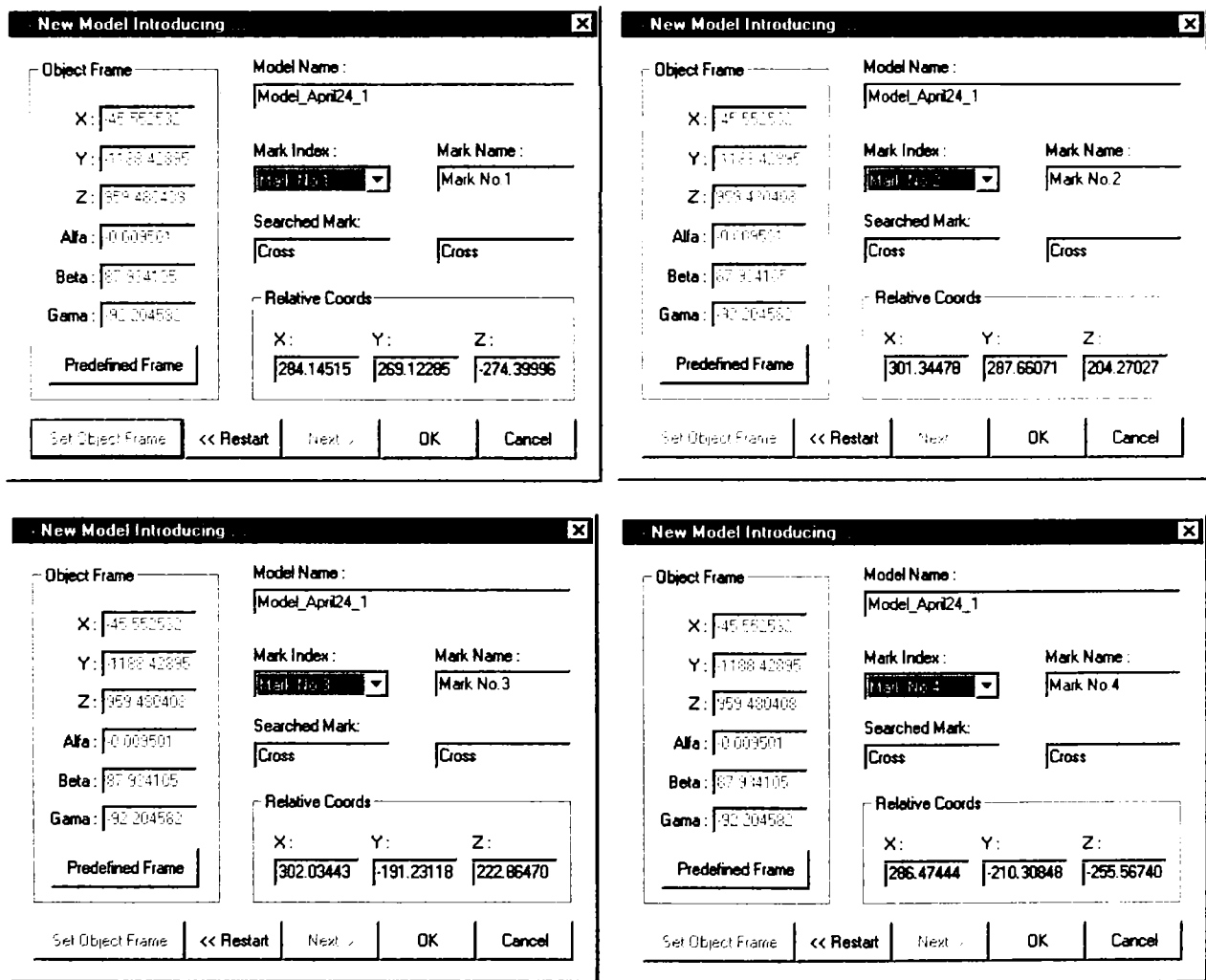


Fig. 7.11. Introducing the object – C program.

After the object was defined, we move the object from the reference position and we measure the deviation of the actual position of the object with respect to the reference position.

In figure 7.12 one can see the frames, which were used in the mathematical description.

To measure the new coordinates of the fixed points from the object the robot will be moved in the same positions as it was done when the coordinates of the fixed points were measured first time. One can make this because the actual position of the object is only a little different from the reference position and the fixed point can be seen by the stereo sensor. Using the same positions for the robot we will eliminate the absolute error of the robot.

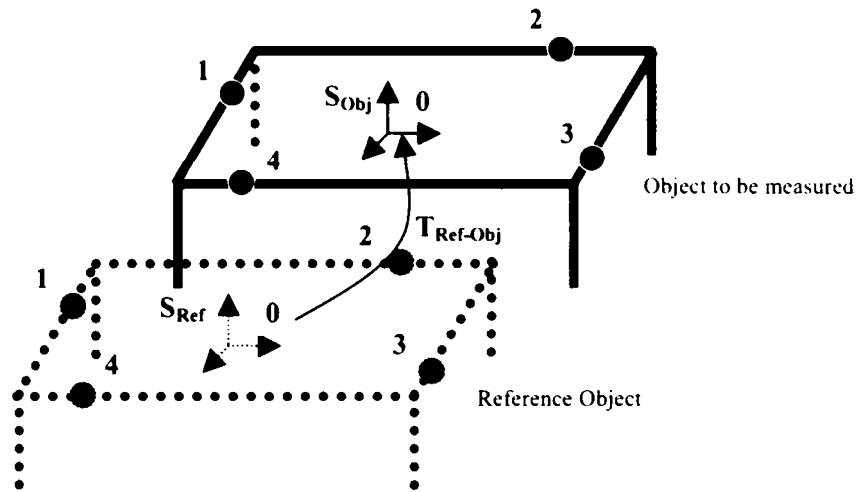


Fig. 7.12. The deviation of the actual object with respect to the reference object.

We denote the measured coordinates of the points  $i$  with  ${}^{Ref}P_i$ . One can write now for a point the following equation, according to the reference [Pau81]:

$${}^{Ref}P_i = {}^{Ref}_{Obj}T \cdot {}^{Obj}P_i, \quad (7.4)$$

where  ${}^{Ref}_{Obj}T$  is the transformation from the reference frame to the actual object frame. This transformation represents the deviation of the actual object position with respect to the reference object position. This deviation will be sent to the robot.

For each point one can write the relation (4.3). This way, we obtain an over determinate system of non-linear equations, which will be solved according to the method presented in sub-chapter 4.2.2.

In the figure 7.13 we measured the deviation of the object frame relative to the reference frame when the object is still in the reference position. The units for  $x$ ,  $y$  and  $z$  are millimeters and for  $alfa$ ,  $beta$  and  $gama$  are degrees.

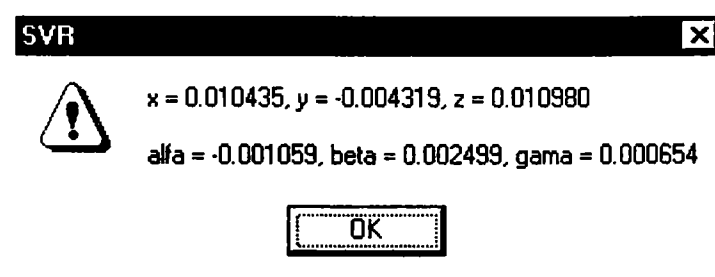


Fig. 7.13. Errors for the reference position of the object.



If there were no errors all the deviations would be zero. As one can see the errors are very small. The greatest influence in producing these errors comes from the image-processing algorithm used to recognize the marked points. Concerning the image-processing algorithm it must be said that the stability of lighting conditions is very important for the stability of the measurement results. The relative error of the robot is the second factor, which produces the final errors.

In the figure 7.14 we measured the deviation of the object frame relative to the reference frame when the object was moved from the reference position. The units for  $x$ ,  $y$  and  $z$  are millimeters and for  $\alpha$ ,  $\beta$  and  $\gamma$  are degrees.

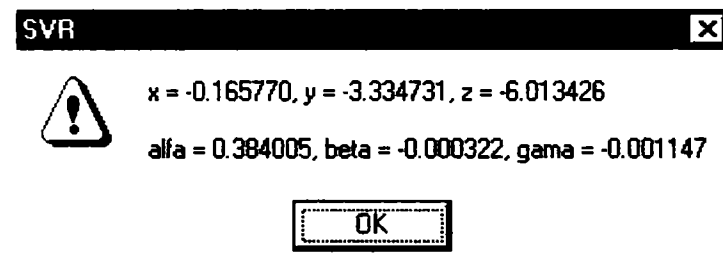


Fig. 7.14. Measured relative position of the object frame.

The measurement system works practically without errors when the range of the deviations is between  $-20$  mm and  $+20$  mm for  $x$ ,  $y$  and  $z$  and between  $-1.5$  and  $1.5$  degree for  $\alpha$ ,  $\beta$  and  $\gamma$ .

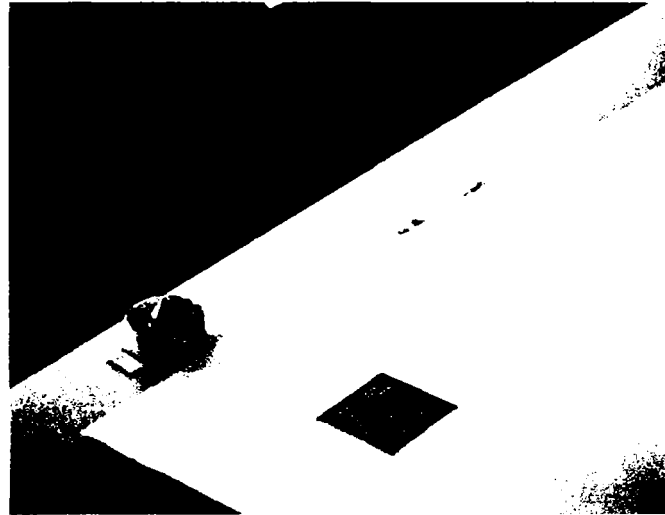
For testing our system we developed a simple application. Our testing object will be a table. A plate was fixed on this table, as one can see in figure 7.15.



Fig. 7.15. The description of the test application.

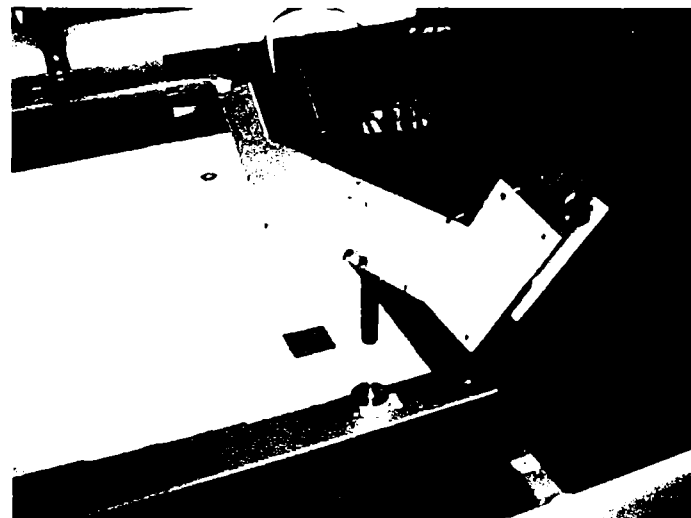
On the plate there are four points, which are used to make the calibration between the robot tool and the stereo sensor. This calibration means to find the transformation from the robot tool frame to the stereo sensor frame. The same points are used as fixed points of the object, according to the description made in the first part of this sub-chapter. We call these

points, *visual points*. We fix on the table another four points, which will be used in our testing application. We call these points, *application points*. One can see clearly one pair of these points in figure 7.16, the black point is a visual point and the other one is an application point.



*Fig. 7.16. Visual point and application point.*

In the test application, our four application points will be touched with the working tool. First step is to teach the robot this test application. We teach manually the robot to touch, with an accuracy of 0.1 mm, step by step the application points. This teaching part will be made with respect to the robot base. In figure 7.17, one can see the final position of the working tool in the teaching procedure for one point.



*Fig. 7.17. The taught position for one application point.*

Second step is to establish a frame for the table and to measure the 3D coordinates of the visual points, in order to define completely the table. The established object frame will be the reference frame in this test application. It is very important to have the table in the same position during step one and two. At the end of step two we make some changes in the software to have the application taught in the reference frame.

Third step is to move the table, to measure the deviation from the reference position, to send this deviation to the robot and finally to see how precisely the working tool will touch the application points. After a lot of practical tests we can say that the total error of the system is maximum 1 mm. It means that the maximum distance between the application point and the end of the working tool is less than 1 mm.

This error comes from two reasons. One reason is the error of the measurement system, but we explained in the first part of this sub-chapter that the shift of the object is measured with high accuracy, so the influence of this error is very small. The second reason is the absolute error of the robot. We have seen in our practical tests that the error was different for our four application points when we sent to the robot the same shift.

# Conclusions and Contributions

Most of the stereo applications are built to obtain 3D information starting from 2D information. This 3D information allows the possibility to reconstruct a real scene and to identify the form of different objects and their relative position in this real scene. This means, the stereo vision was used mostly to obtain qualitative information about the real world. There is also the possibility to use stereo vision in order to make 3D accurate measurements. This means to obtain quantitative information about the real world. As it was said in the introduction, the goal of this PhD thesis was to analyze the problem of using stereo cameras to realize accurate 3D measurements.

My PhD thesis starts by making in chapter 1 a brief introduction in the field of computer vision in order to place the treated subject at the right place in this field. From the field of computer vision we go in the direction called scene reconstruction and then further in the direction called accurate visual metrology, which represents the domain where my PhD thesis belongs.

In chapter 2 are presented basic knowledge about Projective geometry, which is a very important tool for solving the scene reconstruction problem. We need this tool to be able to present in a systematic way the camera models. Also, all the on-line calibration methods are make use of this Projective geometry.

In chapter 3 we present an overview of the existing camera models. They are divided in two categories: ideal camera models, called also distortion free models and real models, which includes the distortion effect. Then an overview of the existing calibration methods is presented. They are also divided in two categories: off-line calibration methods, called traditional calibration methods and on-line calibration methods, called camera self-calibration methods. The calibration method we developed belongs to the first categories. This way, in the last part of chapter 3 we analyze two important off-line calibration methods with contributions at the simulation and analysis of errors produced by measuring with a calibrated camera.

In chapter 4 are presented first the types of stereo sensors, which were built in order to test our calibration method. We developed two types of stereo sensors: in parallel configuration and in non-parallel configuration. Then the chosen camera model is presented.

This model is a real model, which includes effect of the radial distortion. The calibration device is presented in the following parts and finally, the calibration procedure, including my contributions, ends this chapter.

In the first half of chapter 5 a theoretical introduction in image processing is made. Basic knowledge about image enhancement techniques, about edge detection techniques and about grey level segmentation methods is presented here. In the second half of this chapter our sub-pixel image processing algorithm developed in order to identify a certain point is described.

In chapter 6 is described at the beginning the measurement procedure developed in order to obtain accurate 3D results. A detailed analysis of the measurement errors for both parallel and non-parallel configurations of the stereo sensors is presented. For the non-parallel configuration a new method to eliminate the systematic errors is developed. The efficiency of the method is demonstrated by the analysis of the errors presented in the sub-chapter 6.2.2. For the parallel configuration, we implemented different image processing algorithms. A comparative error analysis, when these different image processing algorithms were used, is presented in sub-chapter 6.3. This demonstrates that our sub-pixel image processing algorithm is better than the other tested algorithms.

In chapter 7 some different industrial applications of the stereo sensor as 3D accurate measurement tool are described. There are two possibilities of using the stereo sensor: in a fixed configuration or in a mobile configuration. For each configuration is presented one practical application.

To realize the practical part of this project we had to develop a series of programs. We used Visual C++ environment to develop them. One part of the programs is made to control the calibration device to generate the calibration points and the points to be measured for testing the stereo sensor. Another part is made to implement the calibration procedure and the measurement procedure.

The analysis of the errors was made using MATLAB programs because it was easier to use this software than the Visual C++. The programs for the errors analysis are presented in the Annex E. We developed also programs in C++ to implement the two off-line calibration methods presented in chapter 3. These programs are presented in Annex B and Annex D.

The most important contributions developed in this PhD thesis will be briefly presented in the following parts:

1. In figure 3.12, from the sub-chapter 3.3.4, one can see the method we consider to define the simulation of a camera measurement. During the calibration procedure we compute

the camera parameters, which generates a transformation from the world frame to the camera frame. Unfortunately, this transformation is affected by different errors, which appear in the calibration process. Our idea was to obtain a measure of these errors by computing the error of the position vector given by the relation (3.144). This error of the position vector includes all the calibration errors and this way, shows the quality of the used calibration method.

2. To analyze the quality of the two discussed calibration methods presented in chapter 3 (3.3.2 Lenz calibration method and 3.3.3. Tsai calibration method) we introduced two types of errors in the calibration phase. The first type is referring to the uncertainty of determining the 3D position of the calibration points in the real scene and the second type is referring to the uncertainty of determining the 2D position of the calibration points in the image. The goal was to see how sensitive are these two calibration methods to these types of errors. The results are presented in the graphics from the figure 3.13 and 3.14.

3. Knowing the fact that our stereo sensor will be used to measure 3D coordinates of points situated at small distance from the cameras (200 mm – 500 mm) we decided to use both parallel and non-parallel configurations. At the beginning of chapter 4 is explained way it is not possible to use the parallel configuration when we want to measure points situated at big distance from the cameras.

4. In sub-chapter 4.1.4 is described the model we considered for the cameras. We have 6 internal camera parameters: two for scale factors, denoted  $s_x$ , and  $s_y$ , two parameters for the image center, denoted  $C_x$ , and  $C_y$ , one parameter for the focal lens, denoted  $f$ , and finally, the parameter called coefficient of the radial distortion, denoted  $k$ . Our idea was to make the following notations:  $p_x = s_x f$ ,  $p_y = s_y f$ ,  $d = kf^2$ . This way, we reduced the number of the unknowns, which must be computed in the calibration phase, without affecting the complexity of our defined camera model.

5. To generate the calibration points we used a special device, as one can see in figure 4.5, in sub-chapter 4.2.1. With this device one can obtain a lot of calibration points covering all the working place of the camera. This way, the camera parameters will be computed according to the requirements of the user. If a user needs a stereo sensor to measure only in a certain space we will use for calibrating the sensor only points situated in this certain space. This way, the errors obtained by measuring with the sensor will be smaller than in the case when the calibration is made in a space, which is different from the working space.

6. Having the camera model defined in sub-chapter 4.1.4 the mathematical description of the calibration procedure is presented in sub-chapter 4.2.2. Our way to compute the camera parameters was to obtain two non-linear, over-determined equation systems including all

these parameters. The systems were solved separately, using Newton algorithm and least squares method. In order to obtain the final solution we made for the camera parameters, which were computed in both systems, the average of their computed values. The results of the calibration procedure can be seen in figure 4.6.

7. In sub-chapter 5.2.1 we had to solve the problem of selecting a certain mark to be used as a calibration point. Comparing the relation (5.71) and (5.72) we decided to use a circle as a mark because a circle has less edge points than a cross within the same dimensions.

8. In sub-chapter 5.2.2 we developed an image processing algorithm to compute the pixel coordinates of a mark, which was identified in the recognition process. The entire algorithm presented here is original. It is important to say that we make use of a segmentation with a variable threshold and that we compute the coordinates of the weight point of the identified mark. The value for the threshold is computed with the relation (5.86). The coordinates of the weight point for the identified mark are computed with the relations (5.90) and (5.91).

9. In sub-chapter 5.2.3 we developed also an original sub-pixel algorithm in order to identify the coordinates of the weight point of an identified mark at sub-pixel level. We used the figure 5.10 to explain why it is necessary to adopt a sub-pixel approach if we want high accuracy for the stereo sensor. Going further it is explained our idea of using exploration lines starting from the weight point determined with the algorithm developed in sub-chapter 5.2.2. The relation (5.98) is used to compute the grey level for a point situated at any location between 4 adjacent pixels. The relation (5.101) represents the approximation function used to identify the place where an edge is located.

10. In sub-chapter 6.1 is developed the mathematical part for the measurement procedure. We solved the correspondence problem by making use of the sub-pixel image processing algorithm presented in sub-chapter 5.2. Having the pixel coordinates in both images one can write the relation between the 3D coordinates of a point and its 2D corresponding pixel coordinates making use of the notations from the calibration procedure ( $p_x = s_x f$ ,  $p_y = s_y f$ , and  $d = kf^2$  - see chapter 4). This way, we obtain a linear equation system. To solve the system we use some special notations, as one can see in the relations: (6.7), (6.8), (6.9), and (6.10). Finally we will obtain the 3D coordinates of a measured point with respect to the stereo sensor frame. These results are showed in figure 6.1.

11. From the errors analysis of the stereo sensor built in non-parallel configuration it is obvious that a systematic error appears in the measuring process. One can see that from the graphics presented in figure 6.3, 6.4, and 6.5. In sub-chapter 6.2.2 we present a method to

eliminate this systematic error. We will use a correction function defined by the equation (6.24). This correction function will be computed for each of the 3 coordinates  $x$ ,  $y$  and  $z$ . This function represents a polynomial approximation of the errors. To find its coefficients we used the least squares method. After we applied the correction the results are much better as one can see in the graphics presented in figure 6.6, 6.7, and 6.8. It is also obvious that the influence of the systematic errors after the correction is insignificant.

12. In subchapter 6.3 is presented the analysis of the errors for the stereo sensor built in parallel configuration. This analysis shows the fact that the smallest errors are obtained when the sub-pixel algorithm is used. In table 6.2 we presented a comparison between the obtained errors in the four analyzed cases. The table shows of course the same results as the graphics presented in this sub-chapter namely that our calibration method combined with the sub-pixel algorithm gives the best results.

13. In sub-chapter 7.1 we presented an industrial application, which uses three pairs of stereo sensors fixed on a rigid plate as one can see in figure 7.5. The system was build to solve a part of the wheel alignment problem. This means to measure the angles called Camber and Toe. These angles are defined in figure 7.1 and 7.2. The system produced by us is totally new being a prototype.

14. In sub-chapter 7.2 we presented a test application to show the advantages of using a stereo sensor mounted on the robot hand as a 3D measurement tool. The first advantage is represented by the accuracy of the 3D measurements made using the stereo sensor. Another advantage is represented by the low costs necessary to produce and to calibrate this type of stereo sensor.

For the future we think to improve the camera model by introducing new camera parameters and to develop a self-calibration method making use of the tool called Projective geometry (see chapter 2), tool that helped considerably to solve different camera problems in the last 10 years. Another very important aspect that we will try to solve in the future is the recalibration of the sensor when by different reasons at least one of the camera parameters was changed.



# References

- [AHH99] L. de Agapito, R. Hartley, E. Hayman, "*Linear calibration of a rotating and zooming camera*", Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Fort Collins, Colorado, 1999, pp. 15-21
- [AHR01] L. de Agapito, E. Hayman, I. Reid, "*Self-calibration of rotating and zooming camera*", International Journal of Computer Vision, Kluwer Academic Publishers Hingham, MA, USA, Volume 45, Issue 2, November 2001, pp. 107-127
- [Arm96] M. N. Armstrong, "*Self-Calibration from Image Sequences*", PhD, University of Oxford, 1996
- [AP95] A. Azarbajani, A. Pentland, "*Camera self-calibration from one point correspondence*", Technical Report no. 341, MIT Media Lab, 1995
- [AS99] L. Alvarez, J. Sanchez, "*Non-linear estimation of the 3D geometry from two perspective views*", 1999
- [ASB00] X. Armangué, J. Salvi, J. Balle, 2000. "*A comparative review of camera calibrating methods with accuracy evaluation*", Proceedings of 5<sup>th</sup> Ibero-American Symposium on Pattern Recognition, SIAPR 2000, Lisboa, Portugal, pp. 183-194
- [Atk96] K. B. Atkinson, "*Close Range Photogrammetry and Machine Vision*", Whittles Publishing, Roseleigh House, Latheronwheel, Caithness, KW5 6DW, Scotland, UK
- [BAA98] J. Batista, H. Araujo, A. T. de Almeida, "*Iterative Multi-Step Camera Calibration*", Proceedings of the 6th International Conference on Computer Vision, 1998, pp. 709-714
- [BAHB98] M. J. Brooks, L. de Agapito, D. Q. Huynh, L. Baumela, "*Towards robust metric reconstruction via a dynamic un-calibrated stereo head*", Image and Vision Computing, vol. 16, 1998, pp. 989-1002
- [Bas93] A. Basu, "*Active Calibration: Alternative Strategy and Analysis*", Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, New York, USA, June 1993, pp. 495-500
- [BC97] H. Bacakoglu, M. S. Kamel, "*A Three-Step Camera Calibration Method*", IEEE Transactions on Instrumentation and Measurement, vol. 46, no. 5, October 1997, pp. 1165-1171
- [Ber86] J. Bernsen, "*Dynamic thresholding of grey-level images*", Proceedings of 8<sup>th</sup> International Conference on Pattern Recognition, 1986, pp. 1251-1255.

- [BP98] J. -Y. Bouguet, P. Perona, "3D photography on your desk", Proceedings of IEEE International Conference on Computer Vision Bombay, India, January 1998, pp. 43-50
- [Bro71] D. C. Brown, "Close-range camera calibration", Photogrammetric Engineering, 37(8), 1971, pp. 855-866
- [BT99] J. Bates, T. Tompkins, "Utilizare Visual C++6", Editura Teora 1999
- [Can86] J. Canny, "A computational approach to edge detection", IEEE Transactions on Pattern Analysis and Machine Vision, November 1986, vol. 8, no. 6, pp. 679-698
- [CDR99] R. Cipolla, T. W. Drummond, D. Robertson, "Camera calibration from vanishing points in images of architectural scenes", Proceedings of British Machine Vision Conference, vol. 2, September 1999, UK, pp. 382-391
- [Crim99] A. Criminisi, "Accurate Visual Metrology from Single and Multiple Uncalibrated Images", PhD, University of Oxford, 1999
- [CT90] B. Caprile, V. Torre, "Using vanishing points for camera calibration", International Journal of Computer Vision, vol. 4, no. 2, March 1990, pp. 127-140
- [Cum01a] A. Cumani, "Simple and Accurate Camera Calibration", IEN Technical Report no. 631, July 2001
- [Cum01b] A. Cumani, "Minimalist Camera Calibration", IEN Technical Report no. 641, December 2001
- [Cum02] A. Cumani, "A simple camera calibration method", Proceedings of the 2<sup>nd</sup> WSEAS Conf. Signal Processing, Computational Geometry and Vision, Rethymno 2002
- [Dev95] F. Devernay, "A Non-maxima Suppression method for Edge Detection with Sub-Pixel Accuracy", Technical Report 2724, INRIA Sophia-Antipolis, November 1995
- [DF01] F. Devernay, O. Faugeras, "Straight Lines Have to Be Straight", Machine Vision and Applications, vol. 13, no. 1, 2001, pp. 14-24
- [DG97] F. Dornaika, C. Garcia, "Robust Camera Calibration using 2D-to-3D feature correspondence", Proceedings of Videometrics V - SPIE's Optical Science, Engineering and Instrumentation'97, August 1997, San Diego, pp. 123-133
- [DJK02] G. N. DeSousa, A. H. Jones, A. C. Kak, "An world-independent approach for the calibration of mobile robotics active stereo heads", Proceedings of IEEE International Conference on Robotics and Automation (ICRA'02), vol. 4, 2002, pp. 3336-3341

- [DK02] G. N. DeSousa, A. C. Kak, "*Vision for Mobile Robot Navigation: A Survey*", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 2, February 2002, pp. 237-267
- [Dod82] Petre Dodoc "*Teoria si constructia sistemelor optice*", Editura Technica, Bucuresti, 1982
- [EF03a] M. P. Eklund, A. A. Farag, "*Robust correspondence methods for stereo vision*", International Journal of Pattern Recognition and Artificial Intelligence, 2003, vol. 17, no. 7, pp. 1059-1079
- [EF03b] M. T. El-Melegy, A. A. Farag, "*Statistically Robust Approach to Lens Distortion Calibration with Model Selection*", Conference on Computer Vision and Pattern Recognition Workshop, June 2003, vol. 8, pp 91-99
- [EF03c] A. Eid, A. A. Farag, "*On the performance characterization of stereo and space carving*", Proceedings of Advanced Concepts for Intelligent Vision Systems, Gent, Belgium, 2003, pp. 291-296
- [Fau93] O. Faugeras, "*Three dimensional computer vision*", Massachusetts Institute of Technology, London, 1993
- [Fau95] O. Faugeras, "*Stratification of three-dimensional vision: projective, affine, and metric representation*", Journal of Optical Society of America, 1995, pp. 465-484
- [FL01] O. Faugeras, T. -Q. Luong, "*The Geometry of Multiple Images*", Massachusetts Institute of Technology, London, 2001
- [FLM92] O. Faugeras, T. -Q. Luong, S. Maybank, "*Camera self-calibration: theory and experiments*", Proceedings of the 2<sup>nd</sup> European Conference on Computer Vision, Santa Margherita, Italy, May 1992, pp. 321-334
- [FR97] A. Fusiello, V. Roberto, "*Efficient stereo with multiple windowing*", Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, Puerto Rico 1997, pp. 858-863
- [FT86] O. Faugeras, G. Toscani, "*The calibration problem for stereo*", Proceedings CVPR'86, Miami Beach, Florida, June 1986, pp. 15-20
- [FT87] O. Faugeras, G. Toscani, "*Camera Calibration for 3D Computer Vision*", Proceedings of International Workshop on Industrial Applications of Machine Vision and Machine Intelligence, Tokyo, Japan, 1987, pp. 240-247
- [GLP99] V. Gui, D. Lacrama, D. Pescaru, "*Image Processing*" (in Romanian), Editura Politehnica, Timisoara, 1999
- [GK86] L. A. Gerhard, W. I. Kwak, "*An improved adaptive stereo ranging method for 3D Measurements*", Proceedings CVPR'86, Miami Beach, Florida, June 1986, pp. 21-26

- [HA97] A. Heyden, K. Astrom, "*Euclidean reconstruction from image sequences with varying and unknown focal length and principal point*", Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 1997, pp. 477-492
- [HA99] A. Heyden, K. Astrom, "*Flexible Calibration: Minimal Cases for Auto-Calibration*", Proceedings of 7<sup>th</sup> International Conference on Computer Vision, Kerkyra, Greece, 1999, pp. 350-355
- [Har92] R. Hartley, "*Estimation of relative camera position for un-calibrated cameras*", Proceedings of the 2<sup>nd</sup> European Conference on Computer Vision, Santa Margherita, Italy, May 1992, pp. 579-587
- [Har94] R. Hartley, "*Self-Calibration from Multiple Views with a Rotating Camera*", Proceedings of 3<sup>rd</sup> European Conference on Computer Vision, Stockholm, vol. 1, 1994, pp. 471-478
- [Har97] R. Hartley, "*Self-Calibration of Stationary Cameras*", International Journal of Computer Vision, vol. 22, no. 1, February 1997, pp. 5-23
- [HHAR99] R. Hartley, E. Hayman, L. de Agapito, I. Reid, "*Camera calibration and the search of infinity*", Proceedings of 7<sup>th</sup> International Conference on Computer Vision, Kerkyra, Greece, 1999, pp. 510-517
- [HHN88] B.K.P. Horn, H.M. Hilden, S. Negahdaripour, "*Closed-form solution of absolute orientation using orthonormal matrices*", Journal of the Optical Society of America, vol. 5, July 1988, pp. 1127-1135
- [HM03] E. Hayman, D. W. Murray, "*The Effects of Translational Misalignment when Self-Calibrating Rotating and Zooming Cameras*", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 8, 2003, pp. 1-6
- [Hor87] B. Horn, "*Closed-form solution of absolute orientation using unit quaternions*", Journal of the Optical Society of America, vol. 4, April 1987, pp. 629-642
- [HTMS82] E. Hall, J. Tio, C. McPherson, F. Sadjadi, "*Measuring curved surfaces for robot vision*", *Computer Journal*, vol. December, 1982, pp. 42-54
- [HZ03] R. Hartley, A. Zisserman, "*Multiple View Geometry*", Cambridge University Press, 2003, Second edition
- [IK98] L. Iocchi, K. Konolige, "*A Multi-resolution Stereo Vision System for Mobile Robots*", Proceedings of the AIIA'98 Workshop on New Trends in Robotics Research, pp.317-321
- [Ito86] M. Ito, A. Ishii, "*Range and shape measurement using three - view stereo analysis*", Proceedings CVPR'86, Miami Beach, Florida, June 1986, pp. 9-14
- [Ito91] M. Ito, "*Robot vision modeling - camera modeling and camera calibration*", *Advanced Robotics*, vol.5, no.3, 1991, pp. 321-335

- [Jim93] Jim Z. C. Lai, "*On the Sensitivity of Camera Calibration*", Image and Vision Computing Volume 11, Number 10, December 1993
- [JM03] X. Jiang, D. Mojon, "*Adaptive local thresholding by verification-based multi-threshold probing with application to vessel detection in retinal Images*", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 1, January 2003, pp. 131-137
- [JT02] D. Jurca, **L. Toma Jr.**, "*Medicine Tracking: Implementation of a 3D Trecker with a Multi Camera System*", Buletinul Științific al Universității "Politehnica" din Timișoara, Tomul 47(61), 2002 Fascicola 1-2, Vol. x, pp. 208-213
- [KO94] T. Kanade, M. Okutomi, "*A stereo matching algorithm with an adaptive window: theory and experiments*", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 16, 1994, pp. 920-932
- [Lan58] G. S. Landsberg, "*Optica*", Editura Technica, Bucuresti, 1958, Editia a doua
- [LD99] M. I. A. Lourakis, R. Deriche, "*Camera self-calibration using the singular value decomposition of the fundamental matrix: From point correspondence to 3D measurements*", Technical Report 3748, INRIA Sophia-Antipolis, August 1999
- [LD00] M. I. A. Lourakis, R. Deriche, "*Camera self-calibration using the singular value decomposition of the fundamental matrix*", 2000 Asian Conference on Computer Vision, (ACCV'00), Taiwan, January 2000, vol. 1, pp. 403-408
- [Len87] R. K. Lenz, "*Linsenfehlerkorrigierte Eichnung von Halbleiterkamas mit Standardobjektiven für hochgenaue 3D – Messungen in Echtzeit*", Informatik Fachberichte 149, 9. DAGM-Symposium 1987, Braunschweig, 29. Sep.-1 Oct., Seiten 212 – 216, Springer, 1987
- [LF96] Q. -T. Luong, O. Faugeras, "*The Fundamental Matrix: Theory, Algorithms, and Stability Analysis*", International Journal of Computer Vision, vol. 17, no. 1, 1996, pp. 43-76
- [LF97] Q.-T. Luong, O. Faugeras, "*Self-calibration of a moving camera from point correspondence and fundamental matrix*", International Journal of Computer Vision 22(3), 1997, pp. 261-289
- [Lip01] O. Lipovan, "*Analiza matematica – calcul diferential*", Editia a 3-a, Editura Politehnica, 2001
- [LL96] M. Li, J. -M. Lavest, "*Some Aspects of Zoom Lens Camera Calibration*", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, no. 11, November 1996, pp. 1105-1110
- [LT88] R. K. Lenz, R. Y. Tsai, "*Techniques for Calibration of the Scale Factor and Image Center for High Accuracy 3-D Machine Vision Metrology*", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 10, no. 5, September 1988, pp. 713-720

- [LZ99] D. Liebowitz, A. Zissermann, "*Combining Scene and Auto-Calibration Constraints*", Proceedings of International Conference on Computer Vision, 1999, pp. 293-300
- [KK99] A. Kosaka, A. C. Kak, "*Stereo vision for industrial applications*", Handbook of Industrial Robotics, Second Edition, John Wiley & Sons, New York, 1999, pp.269-294.
- [Ma96] S. D. Ma, "*A self-calibration technique for active vision systems*", IEEE Transactions on Robotics and Automation, vol.12, no. 1, 1996, pp. 114-120
- [Man81] St. Manusar "*Metode numerice in rezolvarea ecuatiilor neliniare*", Editura Technica, Bucuresti, 1981
- [MC99] P. Mendonca, R. Cipolla, "*A simple technique for self-calibration*", Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Fort Collins, Colorado, USA, June 1999, pp. 112-116
- [Men01] P. Mendonca, "*Multiview Geometry: Profiles and Self Calibration*", PhD, University of Cambridge, 2001
- [MF92] S. J. Maybank, O. Faugeras, "*A theory of self-calibration of moving camera*", International Journal of Computer Vision, 8(3), August 1992, pp. 123-151
- [MNK04] P. Mittrapiyanuruk, G. N. DeSouza, A. C. Kak, "*Calculating the 3D-Pose of Rigid Objects Using Active Appearance Models*", Proceedings of International Conference in Robotics and Automation, 2004
- [MS97] J. Majumdar, Seethalakshmy, "*Efficient parallel processing for depth calculation using stereo*", Robotics and Automation Systems, vol. 20, 1997, pp. 1-13
- [Nas99] Pavel Naslau, "*Metode numerice*", Editura Politehnica, Timisoara, 1999
- [NIT02] W. Neddermeyer, A. Ignea, **L. Toma Jr.**, "*Stereo Vision Sensor for Industrial Applications - A new solution for measuring and control of 3D movements of different objects with high accuracy*", SCI2002, Orlando - Florida, USA, Proceedings, Volume IX, pp. 157-161
- [NITS03] W. Neddermeyer, A. Ignea, **L. Toma Jr.**, F. Shu "*Stereo Vision - A new solution for matching the left and the right image models with high accuracy*", SCI2003 Orlando - Florida, USA, Proceedings, Volume X, pp. 145-149
- [NR79] Y. Nakagawa, A. Rosenfeld, "*Some experiments on variable thresholding*", Pattern Recognition, vol. 11, no. 3, 1979, pp. 191-204
- [Par97] J. R. Parker, "*Algorithms for image processing and computer vision*", Copyright © 1997 by John Wiley & Sons, Inc.

- [Pau81] Richard P. Paul, "*Robot Manipulators: Mathematics, Programming and Control*", The MIT Press Cambridge, Massachusetts and London, 1981, pp. 1-63
- [Pen91] M. A. Penna, "*Camera Calibration: A quick and easy way to determine the scale factor*", IEEE Transactions Pattern Analysis and Machine Intelligence, no. 13, 1991, pp. 1240-1245
- [PH86] M. Pietikainen, D. Harwood, "*Depth from three camera stereo*", Proceedings CVPR'86, Miami Beach, Florida, June 1986, pp. 2-8
- [PTVF92] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, "*Numerical Recipes in C: The Art of Scientific Computing*", Cambridge University Press, New York, 1992
- [RM01] D. M. Rendi, I. Mihut, "*Algebra liniara, geometrie analitica si diferenciala*", Editura Politehnica, 2001
- [RSAS01] R. Rosales, M. Siddiqui, J. Alon, S. Sclaroff, "*Estimating 3D Pose using Uncalibrated Cameras*", Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'01), vol. 1, 2001, pp. 821-827
- [RW02] G. Roth, A. Whitehead, "*Some Improvements on Two Autocalibration Algorithms from the Fundamental Matrix*", Proceedings of 16<sup>th</sup> International Conference on Pattern Recognition (ICPR'02), vol. 2, Quebec City, Canada, August 2002, pp. 312-315
- [SK79] J. Semple, G. Kneebone "*Algebraic Projective Geometry*", Oxford University Press, 1979
- [Sp64] C. E. Springer, "*Geometry and Analysis of Projective Spaces*", Freeman, 1964
- [Ste95] G.P. Stein, "*Large accurate internal camera calibration using rotation with analysis of sources of error*", 5<sup>th</sup> International Conference on Computer Vision, 1995 (ICCV'95), pp. 230-236
- [Ste97] G. P. Stein, "*Lens distortion calibration using point correspondence*", IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Juan, Puerto Rico, June 1997, (CVPR'97), pp. 602-608
- [Stu97] P. Sturm, "*Vision 3D Non Calibree: Contributions a la reconstruction projective et etude des mouvements critiques pour l'auto-calibrage*", PhD, l'Institut National Polytechnique de Grenoble, 1997
- [Stu02] P. Sturm, "*Critical Motion Sequences for the Self-Calibration of Cameras and Stereo Systems with Variable Focal Length*", Image and Vision Computing, no. 5-6, vol. 20, March 2002, pp. 415-426
- [TBW03] T. Thormaehlen, H. Broszio, I. Wassermann, "*Robust line-based calibration of lens distortion from single view*", Proceedings of Mirage 2003, INRIA Rocquencourt, France, March 2003, pp. 105-112

- [TH84] R. Y. Tsai, T. S. Huang “*Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces*”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.6, January 1984, pp. 13-27
- [TIMN96] A. Takahashi, I. Ishii, H. Makino, M. Nakashizuka, “*A Camera Calibration Method Using Parallelogramatic Grid Points*”, IEICE Trans. on Inf. & Syst., vol. E79-D, no. 11, Novembre 1996, pp. 1579-1587
- [TIN00] **L. Toma Jr.**, A. Ignea, W. Neddermeyer, “*A Comparison Between Two Camera Calibration Methods*”, Buletinul Științific al Universității "Politehnica" din Timișoara, Tomul 45(59), 2000 Fascicola 1, vol. 2, pp. 222-227
- [TM00] B.J. Tordoff, D.W. Murray, “*Violating Rotating Camera Geometry: The Effect of Radial Distortion on Self-Calibration*”, Proceedings of 15<sup>th</sup> International Conference on Pattern Recognition, 2000, pp. 423-427
- [Tsa86] R. Y. Tsai, “*An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision*”, Proceedings CVPR’86, Miami Beach, Florida, June 1986, pp. 364-374
- [Tsa87] R. Y. Tsai, “*A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf Cameras and Lenses*”, IEEE Journal of Robotics and Automation, vol. RA-3, No.4, August 1987, pp. 323-344
- [TSIN02] **L. Toma Jr.**, F. Shu, A. Ignea, W. Neddermeyer, “*The analysis of the errors for a stereo sensor built in two configurations*”, Buletinul Științific al Universității "Politehnica" din Timișoara, Tomul 47(61), 2002 Fascicola 1-2, Vol. 2, pp. 174-178
- [TSIN04] **L. Toma Jr.**, F. Shu, A. Ignea, W. Neddermeyer, “*The development of a new system to measure Camber and Toe using stereo cameras*”, Buletinul Științific al Universității "Politehnica" din Timișoara, Tomul 49(63), 2004 Fascicola 1-2, Vol. x, pp. 236-239
- [TSNI04] **L. Toma Jr.**, F. Shu, W. Neddermeyer, A. Ignea, “*Stereo Vision Sensor for 3D Measurements – A complete solution to produce, calibrate and verify the accuracy of the measurements results*”, ICINCO2004 - Setubal, Portugal, Proceedings, Volume 2, pp. 410-416
- [Tom00] **L. Toma Jr.**, “*A comparison between two methods for camera calibration*”, Master Theses, 2000
- [TVDF89] G. Toscani, R. Vaillant, R. Deriche, O. Faugeras. “*Stereo camera calibration using the environment*”, Proceedings of the 6<sup>th</sup> Scandinavian conference on image analysis, 1989, pp. 953-960

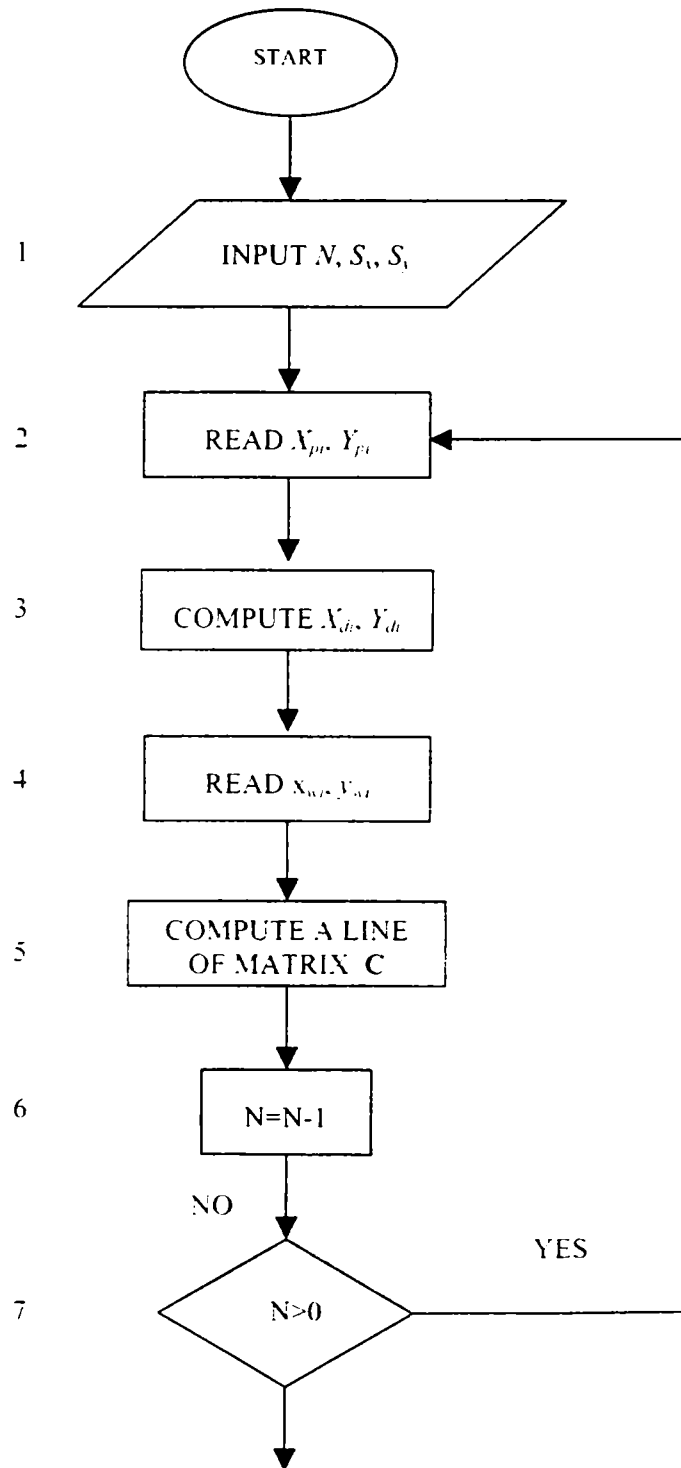


- [Wie92] T. Wieland, "*Untersuchung eines linearen Direkten Verfahrens zur Kamerakalibration*", Vision and Voice Magazine, vol. 6, no. 1, 1992, pp. 31-37
- [WCH92] J. Weng, P. Cohen, M. Herniou, "*Camera calibration with distortion models and accuracy evaluation*", IEEE Transactions on Pattern Analysis and Machine Intelligence. vol 14, 1992, pp.965-980
- [WM93] G. -Q. Wei, S. D. Ma, "*A complete two-plane camera calibration method and experimental comparisons*", Proceedings of 4<sup>th</sup> International Conference on Computer Vision, Berlin, May 1993, pp. 439-446
- [WM94] G. -Q. Wei, S. D. Ma, "*Implicit and explicit camera calibration: Theory and experiments*", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 16, October 1994, pp. 469-480
- [WMC00] K. -Y. Wong, P. Mendonca, R. Cipolla, "*Camera calibration from symmetry*", Mathematics of Surfaces, Cambridge, UK, September 2000, pp. 214-226
- [WMC03] K. -Y. Wong, P. Mendonca, R. Cipolla, "*Camera calibration from surface of revolution*", IEEE Transactions on Pattern Analysis and Machine Intelligence, February 2003, pp. 147-161
- [WR02] A. Whitehead, G. Roth, "*Evolutionary Based Autocalibration from the Fundamental Matrix*", Proceedings of the Applications of Evolutionary Computing on EvoWorkshops, Springer-Verlag, London, UK, 2002, pp. 292-303
- [WXW01] Q. Wu, G. Xu, L. Wang, "*A Three-stage system for camera calibration*", SPIE Conference Proceedings: Multispectral Image Processing and Pattern Recognition, 2001, pp. 22-24
- [Vra62] Gh. Vrânceanu, "*Geometrie analitică, proiectivă și diferențială*", București, ESDP, 1962
- [YDK03] Y. Yoon, G. N. DeSouza, A. C. Kak, "*Real-time tracking and pose estimation for industrial objects using geometric features*", IEEE International Conference on Robotics and Automation, 2003, pp. 3473-3478
- [ZBR95] A. Zissermann, P. Beardsley, I. Reid, "*Metric calibration of a stereo rig*", IEEE Workshop on Representation of Visual Scenes, Boston 1995
- [Zha00] Z. Zhang, "*A flexible new technique for camera calibration*", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 11, November 2000, pp. 1330-1334
- [Zha04] Z. Zhang, "*Camera Calibration with One-Dimension Objects*", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 7, July 2004, pp. 892-899

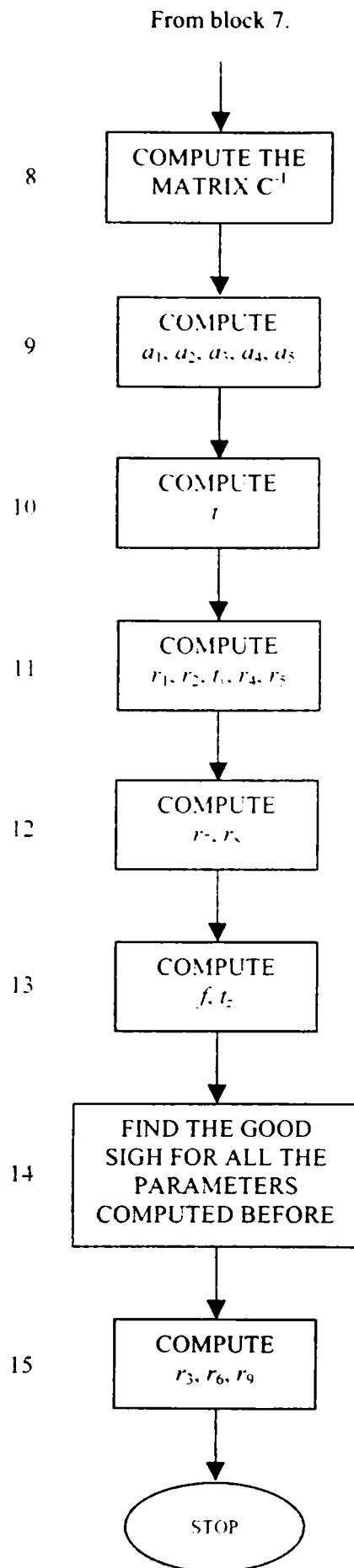
- [ZRF94] Y. Zhou, D. Renoux, J. M. Faure, "*Application of the metric tensor to camera calibration*", *Measurement*, vol. 13, 1994, pp. 47-54
- [Zhu95] H. Zhuang, "*A self-calibration approach to extrinsic parameters estimation of stereo cameras*", *Robotics and Automation Systems*, vol. 15, 1995, pp. 189-197
- [ZRXW93] H. Zhuang, Z. S. Roth, X. Xu, K. Wang, "*Camera calibration Issues in Robot Calibration Eye-on-hand Configuration*", *Robotics and Computer-Integrated Manufacturing*, vol. 10, no. 6, 1993, pp 401-412

## Logical Algorithm for Lenz Calibration Method

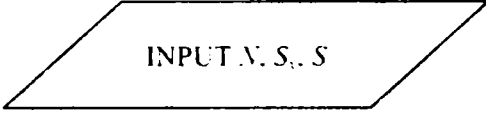
In this annex it is presented a logical algorithm for Lenz camera calibration method. This algorithm will be used to create a C program.

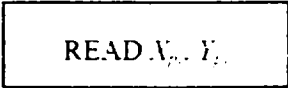



Continue to block 8.



Each block of the logical algorithm will be described in the next parts of this annex.

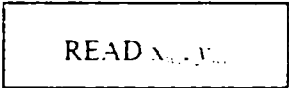
1.   $N$  represents the number of calibration points, which are going to be used in order to find the camera parameters.  $S_x$ ,  $S_y$  are the scale factors, whose values are given by the relations (3.62), and (3.63). At the beginning of our C program we must introduces the values for all these three variables.


2.  The pixels coordinates of a point  $P_i$  are read from a file containing the coordinates for every calibration point used in this case.

3.  Using the values for  $S_x$ ,  $S_y$ ,  $X_{pi}$ , and  $Y_{pi}$  one can compute the values for  $X_{di}$ , and  $Y_{di}$ :

$$X_{di} = S_x^{-1} X_{pi}, \quad (\text{A.1})$$

$$Y_{di} = S_y^{-1} Y_{pi}. \quad (\text{A.2})$$

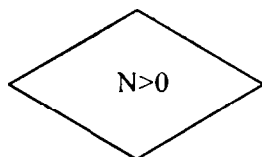
4.  In the same way as it was made in block 2, the 3D coordinates, in millimeters, for a point  $P_i$  are read from a file, which contains the 3D coordinates for all the calibration points, with respect to the world system. There are only two coordinates to read because all the calibration points are in the same plane, and the world system was chosen in such a way that  $z_{wi}$  is zero for all the points.

5.  Using matrices for every point  $P_i$  it is possible to obtain an equation, as follows:

$$\begin{bmatrix} Y_{di} x_{wi} & Y_{di} y_{wi} & Y_{di} & -X_{di} x_{wi} & -X_{di} y_{wi} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} X_{di} \\ X_{di} \\ \cdot \\ \cdot \\ X_{di} \end{bmatrix}. \quad (\text{A.3})$$

So, here using the values for  $x_{wi}$ ,  $y_{wi}$ ,  $z_{wi}$ ,  $X_{di}$ , and  $Y_{di}$ , the C program must compute the values for  $Y_{di} x_{wi}$ ,  $Y_{di} y_{wi}$ ,  $Y_{di}$ ,  $-X_{di} x_{wi}$ , and  $-X_{di} y_{wi}$ .

6. N=N-1 The C program will use the variable  $N$  to count the number of the calibration points  $P_i$ . At the beginning  $N$  is equal with the total number of the calibration points, and it is decreased with one, for each execution of the loop formed by the blocks noted 2, 3, 4, and 5.



7. This loop will be repeated for every calibration point, it means until  $N$  is equal with zero. Finally, the matrix  $C$  will have the following form:

$$C = \begin{bmatrix} Y_{d1}x_{w1} & Y_{d1}y_{w1} & Y_{d1} & -X_{d1}x_{w1} & -X_{d1}y_{w1} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ Y_{di}x_{wi} & Y_{di}y_{wi} & Y_{di} & -X_{di}x_{wi} & -X_{di}y_{wi} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ Y_{dN}x_{wN} & Y_{dN}y_{wN} & Y_{dN} & -X_{dN}x_{wN} & -X_{dN}y_{wN} \end{bmatrix}. \quad (\text{A.4})$$

COMPUTE THE  
MATRIX  $C^{-1}$

8. From the reference [PTVF92], in the C program will be used the function which gives the inverse of a matrix, in order to compute the matrix  $C^{-1}$ .

COMPUTE  
 $a_1, a_2, a_3, a_4, a_5$

9. In the same way as before, it will be used a function from the reference [PTVF92]. This function makes the multiplication between a matrix and a vector, as follows:

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = C^{-1} * \begin{bmatrix} X_{d1} \\ X_{d2} \\ \cdot \\ \cdot \\ X_{dN} \end{bmatrix}. \quad (\text{A.5})$$

COMPUTE  
 $t_p$

10. In this block the C program must implement the following relation in order to compute the value for  $t_p$ :

$$t_p = \frac{2}{\left[ (a_1 + a_5)^2 + (a_2 - a_4)^2 \right]^{\frac{1}{2}} + \left[ (a_1 - a_5)^2 + (a_2 + a_4)^2 \right]^{\frac{1}{2}}}. \quad (\text{A.6})$$

11. **COMPUTE**  
 $r_1, r_2, t_1, r_4, r_5$

The values for  $r_1, r_2, t_1, r_4,$  and  $r_5$  are computed in the C program using the next relations:

$$r_1 = a_1 t_1, \quad r_2 = a_2 t_1, \quad t_1 = a_3 t_1, \quad r_4 = a_4 t_1, \text{ and } r_5 = a_5 t_1 \quad (\text{A.7})$$

12. **COMPUTE**  
 $r_7, r_8$

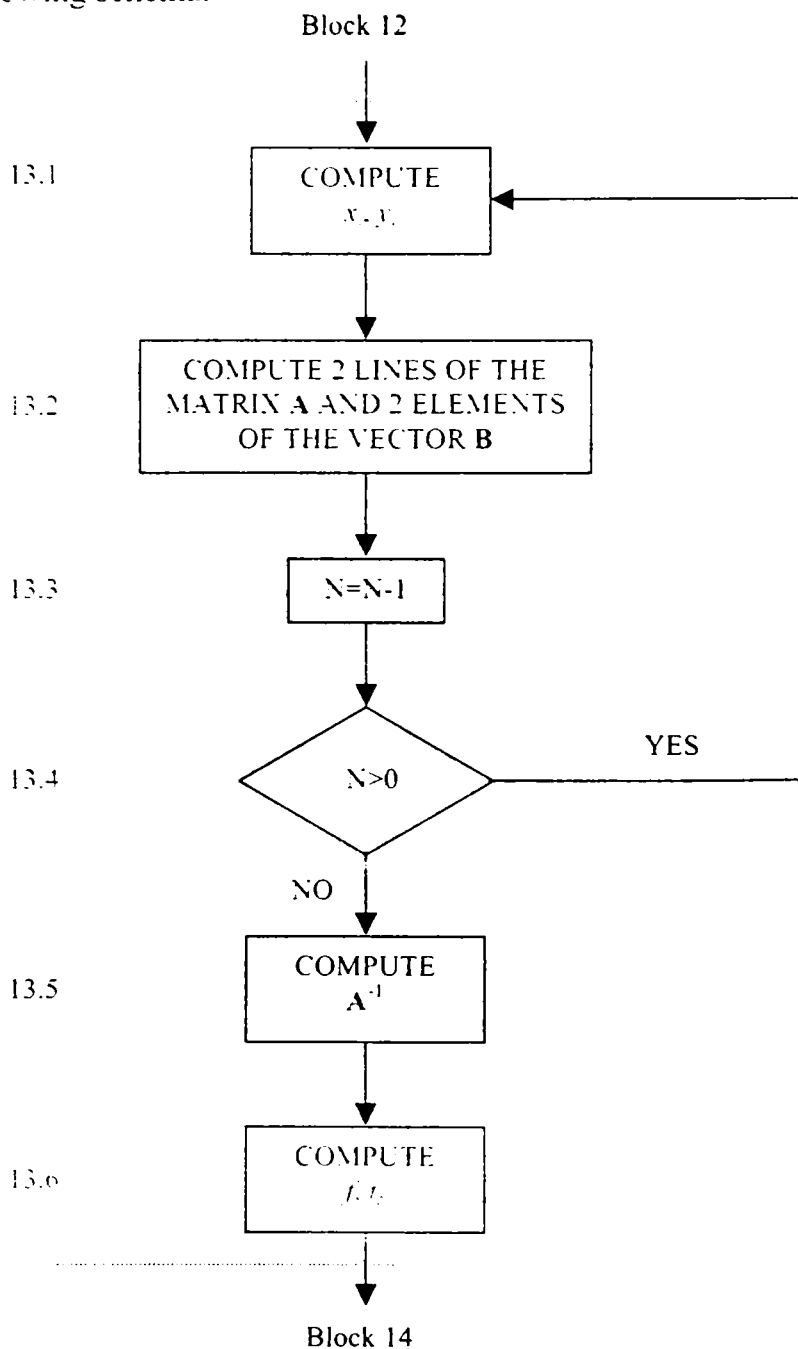
Using the following relations the C program computes the values for  $r_7,$  and  $r_8$ :

$$r_7 = (1 - r_1^2 - r_4^2), \quad (\text{A.8})$$

$$r_8 = -(1 - r_2^2 - r_5^2) \text{sign}(r_1 r_2 + r_4 r_5). \quad (\text{A.9})$$

13. **COMPUTE**  
 $t, t_1$

This block can be divided in some smaller blocks, as one can see in the following schema:



13.1. 

COMPUTE $x, y$
-------------------

 For this block, the C program must implement the next two relations:

$$x_i = x_{wi}r_1 + y_{wi}r_2 + t_v, \quad (\text{A.10})$$

$$y_i = x_{wi}r_4 + y_{wi}r_5 + t_v. \quad (\text{A.11})$$

13.2. 

COMPUTE 2 LINES OF THE MATRIX A AND 2 ELEMENTS OF THE VECTOR B
--

 For each point  $P_i$  it is possible to obtain the following two equation:

$$-x_i f - X_{di} t_z = x_{wi} r_7 X_{di} + y_{wi} r_8 X_{di}, \quad (\text{A.12})$$

$$-y_i f - Y_{di} t_z = x_{wi} r_7 Y_{di} + y_{wi} r_8 Y_{di}. \quad (\text{A.13})$$

Using matrices we obtain, as follows:

$$\begin{bmatrix} -x_i & -X_{di} \\ -y_i & -Y_{di} \end{bmatrix} \begin{bmatrix} f \\ t_z \end{bmatrix} = \begin{bmatrix} x_{wi} r_7 X_{di} + y_{wi} r_8 X_{di} \\ x_{wi} r_7 Y_{di} + y_{wi} r_8 Y_{di} \end{bmatrix}. \quad (\text{A.14})$$

So, here the C program only has to put the values for  $-x_i$ ,  $-X_{di}$ , and  $-y_i$ ,  $-Y_{di}$ , as two lines of a matrix noted **A**, because these values were computed before, but for the vector **B** it must be first, computed the values for  $x_{wi} r_7 X_{di} + y_{wi} r_8 X_{di}$ ,  $x_{wi} r_7 Y_{di} + y_{wi} r_8 Y_{di}$ , and then they must be put as two elements of the vector **B**.

With the blocks 13.3, and 13.4 the loop formed by the blocks 13.1, and 13.2 is repeated for  $N$  times. It means, that for each point  $P_i$  two new lines will be put in the matrix **A**, and two new elements in the vector **B**. Finally, we will obtain the next equation:

$$\begin{bmatrix} -x_i & -X_{di} \\ \cdot & \cdot \\ -x_i & -X_{di} \\ -y_i & -Y_{di} \\ \cdot & \cdot \\ -y_i & -Y_{di} \end{bmatrix} \begin{bmatrix} f \\ t_z \end{bmatrix} = \begin{bmatrix} x_{wi} r_7 X_{di} + y_{wi} r_8 X_{di} \\ \cdot \\ x_{wi} r_7 X_{di} + y_{wi} r_8 X_{di} \\ x_{wi} r_7 Y_{di} + y_{wi} r_8 Y_{di} \\ \cdot \\ x_{wi} r_7 Y_{di} + y_{wi} r_8 Y_{di} \end{bmatrix}. \quad (\text{A.15})$$



- 13.5. 

COMPUTE $A^{-1}$
---------------------

 From the reference [PTVF92], it will be used the function which gives us the inverse of a matrix, in order to compute the matrix  $A^{-1}$

- 13.6. 

COMPUTE $t_1, t_2$
-----------------------

 In the same way, as it was made in block 13.5 be, it will be used a function from the reference [PTVF92], function which makes the multiplication between a matrix and a vector. This way, we will implement the following operation:

$$\begin{bmatrix} f \\ t_2 \end{bmatrix} = A^{-1} B. \quad (\text{A.16})$$

14. 

FIND THE GOOD SIGN FOR ALL THE PARAMETERS COMPUTED BEFORE
--

 We must write a function in the C program, which will implement the next operations, (see step 5 from Lenz Calibration Method):

$$\{r_1, r_2, r_3, r_4, r_5, t_1, t_2\} = \frac{\{r_1, r_2, r_3, r_4, r_5, t_1, t_2\}}{\text{sign}\left(\frac{b}{t_2}\right)}. \quad (\text{A.17})$$

$$\{r_7, r_8, t_3\} = \frac{\{r_7, r_8, t_3\}}{\text{sign}(t_2)}. \quad (\text{A.18})$$

$$\{b\} = \frac{\{b\}}{\text{sign}(b)}. \quad (\text{A.19})$$

15. 

COMPUTE $r_3, r_6, r_9$
----------------------------

 The C program must make the next three operations, (see step 6 from Lenz Calibration Method):

$$r_3 = r_4 r_8 - r_5 r_7, \quad (\text{A.20})$$

$$r_6 = r_2 r_7 - r_1 r_8, \quad (\text{A.21})$$

$$r_9 = r_1 r_5 - r_2 r_4. \quad (\text{A.22})$$

Using now this logical algorithm it will be easier to write the C program than starting directly, and making use only of the information from these six steps of the Lenz method, described in sub-chapter 3.3.2.

## C Program for Lenz Calibration Method

In this annex is presented the C program developed following the logical algorithm from Annex A. This C program implements Lenz calibration method.

```
//Liviu Toma C Program for Lenz Calibration Method

#include "matrix.h"
#include "Includes.h"
#include "calibration.h"

void main(void)
{
    int N=121,M;           //number of calibration points
    int i,j,inv;
    double Tx,Ty,Tz,r1,r2,r3,r4,r5,r6,r7,r8,r9;
    double x,y;
    double dScale_in_X; // = 0.008509;
    double dScale_in_Y; // = 0.008316;
    double sx=120; //90.5;
    double sy=120; //92.75;
    double Cx=384; //380;
    double Cy=287; //295;
    double f;           //focal length
    double **coordXYZ_in_refSystem;
    double **coordXY_in_pixels;
    double **coordXY_on_chip;
    double **C;         //coefficients matrix
    double **INVC;
    double *Xd;
    double *A;
    double **CF;        //coefficients matrix in used to compute f and Tz
    double **INVCF;
    double *BF;
    double *FTc;        //vector which contains two elements, f and Tz
    double **Trans;    //the real transformation
    double **ComputedTrans;
    double *TranslAngles;
    double *ComputedTranslAngles;
    double *DeltaTranslAngles;

    FILE *fvp;
    char field[10];
    M=2*N;

    //realize the correct correspondance
    dScale_in_X=1/sx;
    dScale_in_Y=1/sy;

    //we reserve memory for all the vectors and matrices
    coordXYZ_in_refSystem = dmatrix(1,N,1,3);
    coordXY_in_pixels = dmatrix(1,N,1,2);
    coordXY_on_chip = dmatrix(1,N,1,2);
    C = dmatrix(1,N,1,5);
    INVC = dmatrix(1,5,1,N);
    Xd = dvector(1,N);
```

```

A = dvector(1,5);
CF = dmatrix(1,M,1,2);
INVCF = dmatrix(1,2,1,M);
BF = dvector(1,M);
FIz = dvector(1,2);
Trans = dmatrix(1,4,1,4);
ComputedTrans = dmatrix(1,4,1,4);
TranslAngles = dvector(1,6);
ComputedTranslAngles = dvector(1,6);
DeltaTranslAngles = dvector(1,6);

//we open the file 3Dcoordinates.txt
fvp = fopen ("Myl21vectorboard.txt","r");
if(fvp==0)
{
    printf("Error while opening 3Dcoordinates.txt\n");
    getchar();
    exit(1);
}
for(i=1;i<=N;i++)
{
    //printf("\n");
    for(j=1;j<=3;j++)
    {
        fscanf(fvp,"%s",feld);
        coordXYZ_in_refSystem[i][j]=atof(feld);
        //printf("\t%f",coordXYZ_in_refSystem[i][j]);
    }
}
fclose (fvp);

//we open the file Myl21chipcoordinates.txt
fvp = fopen ("Myl21chipcoordinates.txt","r");
if(fvp==0)
{
    printf("Error while opening pixelsCoordinates.txt\n");
    getchar();
    exit(1);
}
for(i=1;i<=N;i++)
{
    //printf("\n");
    for(j=1;j<=2;j++)
    {
        fscanf(fvp,"%s",feld);
        coordXY_in_pixels[i][j]=atof(feld);
        //printf("\t%f",coordXY_in_pixels[i][j]);
    }
}
fclose (fvp);

//compute the coefficients matrix C
for(i=1;i<=N;i++)
{
    //compute Xdi and Ydi
    coordXY_on_chip[i][1] = dScale_in_X*(coordXY_in_pixels[i][1]-
    Xd[i] = coordXY_on_chip[i][1];
    coordXY_on_chip[i][2] = dScale_in_Y*(coordXY_in_pixels[i][2]-
    //compute a line of matrix C
    C[i][1]=coordXY_on_chip[i][2]*coordXYZ_in_refSystem[i][1];

```

```

        C[i][2]=coordXY_on_chip[i][2]*coordXYZ_in_refSystem[i][2];
        C[i][3]=coordXY_on_chip[i][2];
        C[i][4]=-coordXY_on_chip[i][1]*coordXYZ_in_refSystem[i][1];
        C[i][5]=-coordXY_on_chip[i][1]*coordXYZ_in_refSystem[i][2];
    }

    //compute the matrix INVC
    inv = invers(INVC,C,N,5);

    //compute the vector A
    mulmatrixVector(A,INVC,Xd,5,N,N);

    //compute the translation Ty
    Ty=2/(sqrt((A[1]+A[5])*(A[1]+A[5])+(A[2]-A[4])*(A[2]-
A[4]))+sqrt((A[1]-A[5])*(A[1]-A[5])+(A[2]+A[4])*(A[2]+A[4])));
    //printf("Ty is: %f",Ty);
    //compute r1,r2,r4,r5 and Tx
    r1=Ty*A[1];
    r2=Ty*A[2];
    Tx=Ty*A[3];
    r4=Ty*A[4];
    r5=Ty*A[5];

    //compute r7 and r8
    r7=sqrt(1-r1*r1-r4*r4);
    if(r1*r2+r4*r5<0)
    {
        r8=sqrt(1-r2*r2-r5*r5);
    }
    else
    {
        r8=-sqrt(1-r2*r2-r5*r5);
    }

    //compute x and y
    x=r1*coordXYZ_in_refSystem[3][1]+r2*coordXYZ_in_refSystem[3][2]+Tx;
    y=r4*coordXYZ_in_refSystem[3][1]+r5*coordXYZ_in_refSystem[3][2]+Ty;

    //verify if the chosen sign for Ty is the good one
    if((x*(coordXY_in_pixels[3][1]-Cx)<0) || (y*(coordXY_in_pixels[3][2]-
Cy)<0))
    {
        Ty=-Ty;
        Tx=-Tx;
        r1=-r1;
        r2=-r2;
        r4=-r4;
        r5=-r5;
    }

    //compute the matrix CF and the vector BF

    for(i=1;i<=N;i++)
    {
        CF[i][1]=r4*coordXYZ_in_refSystem[i][1]+r5*coordXYZ_in_refSystem[i][2
]+Ty;
        CF[i][2]=coordXY_on_chip[i][2];
        BF[i]=-
CF[i][2]*(coordXYZ_in_refSystem[i][1]*r7+coordXYZ_in_refSystem[i][2]*r8);
    }

```

```

for(i=1;i<=N;i++)
{
    CF[N+1][1]=r1*coordXYZ_in_refSystem[i][1]+r2*coordXYZ_in_refSystem[i]
[2]+Tx;
    CF[N+1][2]=coordXY_on_chip[i][1];
    BF[N+1]=-
CF[N+1][2]*(coordXYZ_in_refSystem[i][1]*r7+coordXYZ_in_refSystem[i][2]*r8);
}

//compute INVCF
inv = invers(INVCF,CF,M,2);

//compute i and Tz
mulmatrixVector(FTz,INVCF,BF,2,M,M);
f=FTz[1];
Tz=FTz[2];

//establish the right sign for the elementes of the transformation

if(Tz<0)
{
    r7=-r7;
    r8=-r8;
    Tz=-Tz;
}
if(f<0)
{
    f=-f;
}
//compute r3,r6,r9
r3=r4*r8-r5*r7;
r6=r2*r7-r1*r8;
r9=r1*r5-r2*r4;

//write the matrix ComputedTrans
ComputedTrans[1][1]=r1;
ComputedTrans[1][2]=r2;
ComputedTrans[1][3]=r3;
ComputedTrans[1][4]=Tx;
ComputedTrans[2][1]=r4;
ComputedTrans[2][2]=r5;
ComputedTrans[2][3]=r6;
ComputedTrans[2][4]=Ty;
ComputedTrans[3][1]=r7;
ComputedTrans[3][2]=r8;
ComputedTrans[3][3]=r9;
ComputedTrans[3][4]=Tz;
ComputedTrans[4][1]=0.0;
ComputedTrans[4][2]=0.0;
ComputedTrans[4][3]=0.0;
ComputedTrans[4][4]=1.0;

//compute the vector ComputedTranslAngles
TransTo6Vector3(ComputedTrans,ComputedTranslAngles);

//print the solutions
printf("\nThe focal lenght is: %f", f);

printf("\nThe computed transformation vector is: \n");
for(i=1;i<=3;i++)
{

```

```

        printf("\n%f",ComputedTranslAngles[i]);
    }

    for(i=4;i<=6;i++)
    {
        printf("\n%f",ComputedTranslAngles[i]*90/asin(1));
    }

    printf("\n\n");
    //read the real transformation from the file RealTransformation.txt
    //inv=loadmatrix(Trans,4,4,"My37RealTransformation.txt");
    inv=loadmatrix(Trans,4,4,"camltow.txt");
    //printdmatrix(Trans,4,4);

    //compute the vector TranslAngles
    TransTo6Vector3(Trans,TranslAngles);
    printf("\nThe real transformation vector is: \n");
    for(i=1;i<=3;i++)
    {
        printf("\n%f",TranslAngles[i]);
    }

    for(i=4;i<=6;i++)
    {
        printf("\n%f",TranslAngles[i]*90/asin(1));
    }

    free_dmatrix(ccoordXYZ_in_refSystem, 1,N,1,3);
    free_dmatrix(ccoordXY_in_pixels, 1,N,1,2);
    free_dmatrix(C, 1,N,1,5);
    free_dmatrix(INVC, 1,5,1,N);
    free_dvector(Xd,1,N);
    free_dvector(A,1,5);

    free_dmatrix(CF, 1,M,1,2);
    free_dmatrix(INVCF, 1,2,1,M);
    free_dvector(BF, 1,M);
    free_dvector(FTz, 1,2);

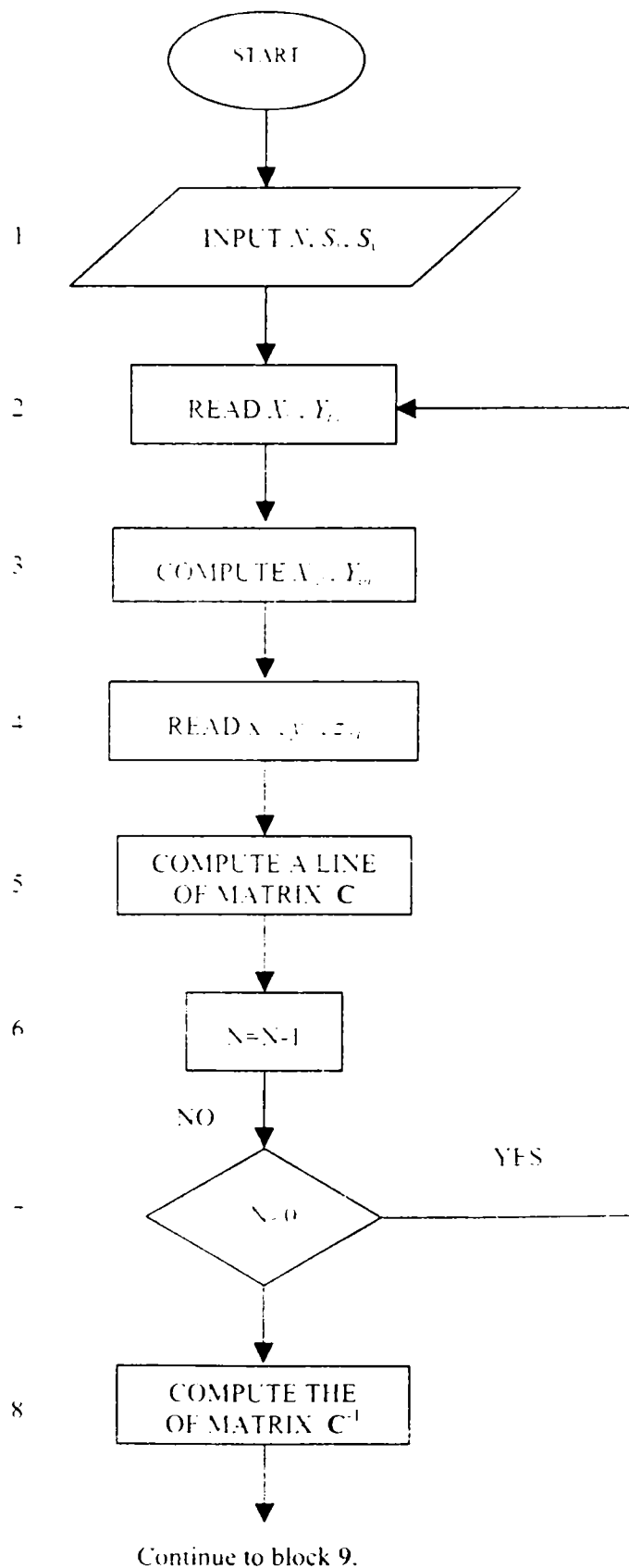
    free_dmatrix(Trans, 1,4,1,4);
    free_dmatrix(ComputedTrans, 1,4,1,4);
    free_dvector(TranslAngles, 1,6);
    free_dvector(ComputedTranslAngles, 1,6);
    free_dvector(DeltaTranslAngles, 1,6);

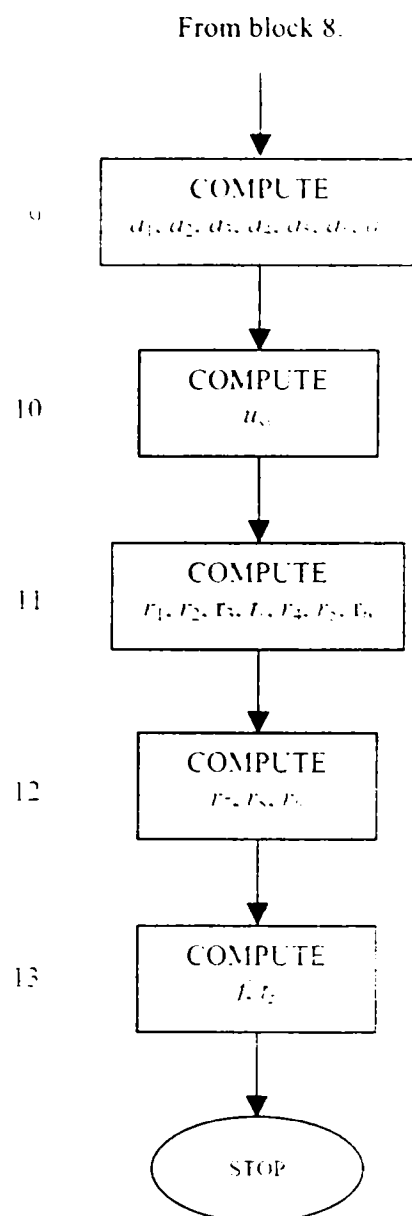
    getchar();

```

## Logical Algorithm for Tsai Calibration Method

In this annex it is presented a logical algorithm for Tsai camera calibration method. This algorithm will be used to create a C program.





As one can see, there are no big differences between the logical schematic presented here and that one from Annex A. In the following parts of this annex there will be presented only the blocks, which are different from those described in Annex A.

The blocks noted 1, and 2 are the same as the blocks 1, and 2 from the Annex A.

The block 3 differs only by the fact that instead of computing  $X_{di}$  the C program will compute  $X_{di}^*$ , using the following relation:

$$X_{di}^* = S_x^{-1} X_{pi}. \quad (\text{C.1})$$

In block 4 the C program will read instead of two coordinates, as we did in block 4 from Annex A, three coordinates because in this case, the calibration points are not in the same plane, so we have different values for  $z_{wi}$ .



In block 5 appear some differences because the equation which must be implemented is different from the one which has already been implemented in block 5 from Annex A. In this case the equation, which is obtained for each point  $P_i$  has the following form:

$$Y_{di}x_{wi}a_1 + Y_{di}y_{wi}a_2 + Y_{di}z_{wi}a_3 + Y_{di}a_4 - X_{di}x_{wi}a_5 - X_{di}y_{wi}a_6 - X_{di}z_{wi}a_7 = X_{di}. \quad (C.2)$$

If we use the matrices one can write this equation, as follows:

$$\begin{bmatrix} Y_{d1}x_{w1} & Y_{d1}y_{w1} & Y_{d1}z_{w1} & Y_{d1} & -X_{d1}x_{w1} & -X_{d1}y_{w1} & -X_{d1}z_{w1} \\ Y_{d2}x_{w2} & Y_{d2}y_{w2} & Y_{d2}z_{w2} & Y_{d2} & -X_{d2}x_{w2} & -X_{d2}y_{w2} & -X_{d2}z_{w2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ Y_{dN}x_{wN} & Y_{dN}y_{wN} & Y_{dN}z_{wN} & Y_{dN} & -X_{dN}x_{wN} & -X_{dN}y_{wN} & -X_{dN}z_{wN} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = [X_{di}]. \quad (C.3)$$

So, here using the values for  $x_{wi}$ ,  $y_{wi}$ ,  $z_{wi}$ ,  $X_{di}$ , and  $Y_{di}$ , the C program computes the values for  $Y_{di}x_{wi}$ ,  $Y_{di}y_{wi}$ ,  $Y_{di}z_{wi}$ ,  $Y_{di}$ ,  $-X_{di}x_{wi}$ ,  $-X_{di}y_{wi}$ , and  $-X_{di}z_{wi}$ .

The blocks 6 and 7 have the same goal as the blocks 6, and 7 from the annex A. After the loop formed by the blocks 2, 3, 4, and 5 has been executed for  $N$  times, where  $N$  is the number of the calibration points we will have the matrix  $C$ , as follows:

$$C = \begin{bmatrix} Y_{d1}x_{w1} & Y_{d1}y_{w1} & Y_{d1}z_{w1} & Y_{d1} & -X_{d1}x_{w1} & -X_{d1}y_{w1} & -X_{d1}z_{w1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ Y_{dN}x_{wN} & Y_{dN}y_{wN} & Y_{dN}z_{wN} & Y_{dN} & -X_{dN}x_{wN} & -X_{dN}y_{wN} & -X_{dN}z_{wN} \\ Y_{dN}x_{wN} & Y_{dN}y_{wN} & Y_{dN}z_{wN} & Y_{dN} & -X_{dN}x_{wN} & -X_{dN}y_{wN} & -X_{dN}z_{wN} \end{bmatrix}.$$

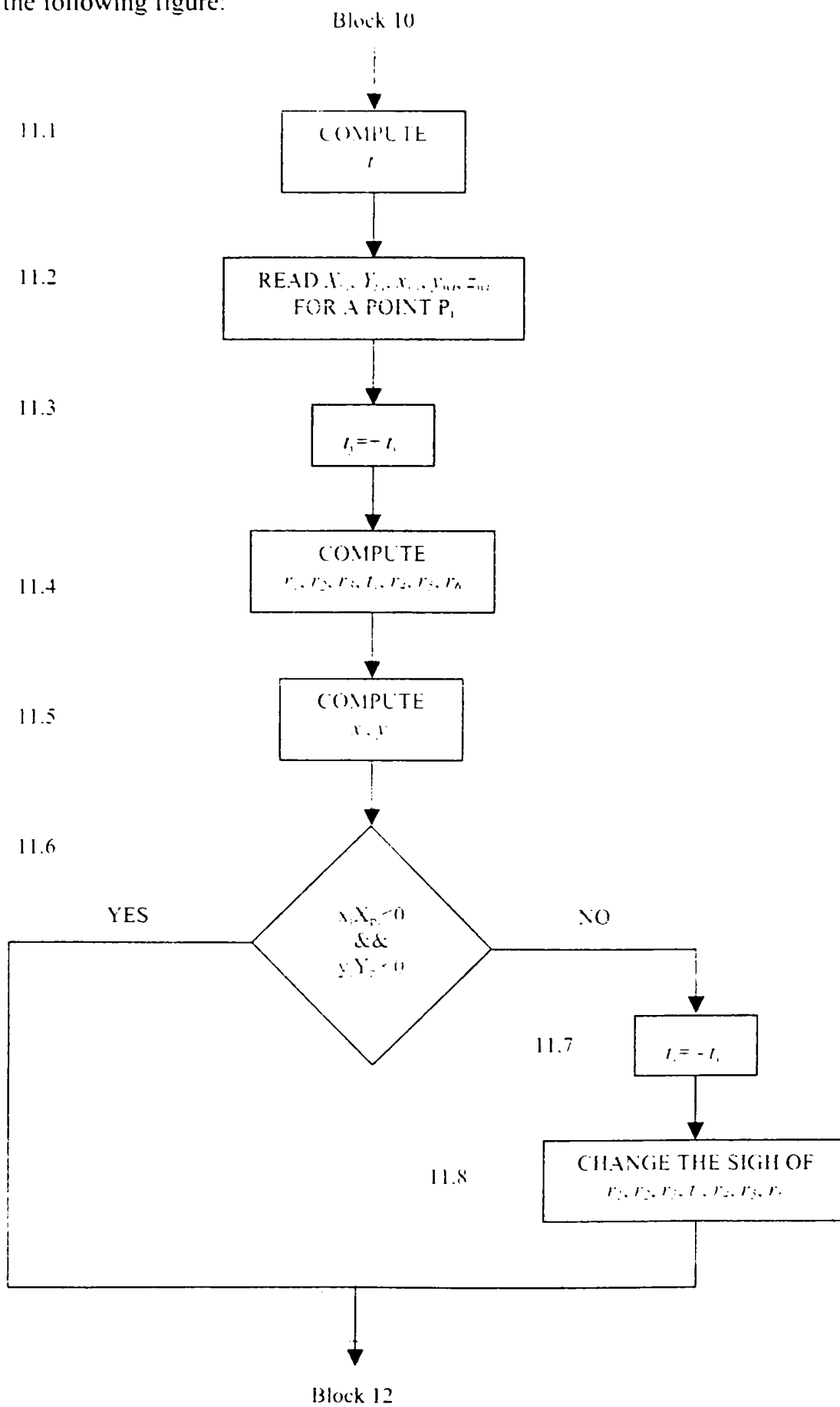
The block 8 realizes the same function as the block 8 described in Annex A.

The function implemented in the block 9 is the same function as that implemented in the block 9 from Annex A, the differences are only in the dimension of the vector and the matrix which must be multiplied.

The block 10 is new. This means that we have not this block in the algorithm presented in Annex A. Here, in the C program must be implemented a function which will compute the uncertainty of the  $x$  scale factor  $u_{x,c}$ , given by the following relation:

$$u_{x,c} = (a_1^2 + a_2^2 + a_3^2)^{\frac{1}{2}} (a_5^2 + a_6^2 + a_7^2)^{\frac{1}{2}}. \quad (5.4)$$

The block 11 is also new. This block can be divided in some smaller blocks, as one can see in the following figure:



- 11.1 

COMPUTE $t_v$
------------------

 The C Program will compute the value for  $|t_v|$  using the following relation:

$$|t_v| = (a_5^2 + a_6^2 + a_7^2)^{-\frac{1}{2}}. \quad (C.5)$$

- 11.2 

READ $X_{pi}, Y_{pi}, X_{wi}, Y_{wi}, z_{wi}$ FOR A POINT $P_i$
--

 In order to establish the right sign for  $t_v$  we need this information in the next steps. It would be good if the chosen point  $P_i$  will be situated far from the image center.

- 11.3 

$t_v = -t_v$
--------------

 In the C program it will be chosen the sign +1 for  $t_v$ .

- 11.4 

COMPUTE $r_1, r_2, r_3, t_x, r_4, r_5, r_6$
--

 Using the values for  $a_1, a_2, a_3, a_4, a_5, a_6, a_7, u_{sx}$ , and  $t_v$ , computed before, the C program computes  $r_1, r_2, r_3, r_4, r_5, r_6$ , and  $t_x$  using the following relations:

$$r_1 = a_1 u_{sx}^{-1} t_v, \quad (C.6)$$

$$r_2 = a_2 u_{sx}^{-1} t_v, \quad (C.7)$$

$$r_3 = a_3 u_{sx}^{-1} t_v, \quad (C.8)$$

$$t_x = a_4 u_{sx}^{-1} t_v, \quad (C.9)$$

$$r_4 = a_5 t_v, \quad (C.10)$$

$$r_5 = a_6 t_v, \quad (C.11)$$

$$r_6 = a_7 t_v. \quad (C.12)$$

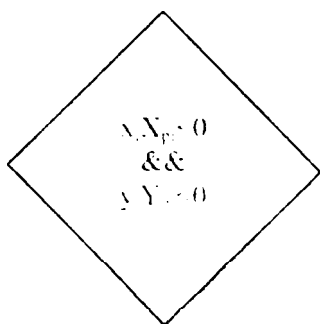
- 11.5 

COMPUTE $x_i, y_i$
-----------------------

 It must be computed  $x_i$  and  $y_i$ , which are two of the three coordinates of the point  $P_i$  with respect to the camera system. We will use the following two relations:

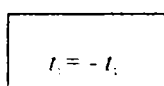
$$x_i = x_{wi} r_1 + y_{wi} r_2 + z_{wi} r_3 + t_x, \quad (C.13)$$

$$y_i = x_{wi} r_4 + y_{wi} r_5 + z_{wi} r_6 + t_x. \quad (C.14)$$



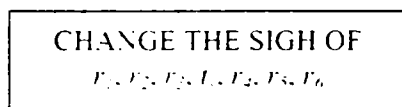
11.6 The C program tests here if it was made the right choice in the block 11.3 for the sign of  $t_i$ . With our systems and notations, if the right choice was done, we should have  $x_i$  and  $X_{pi}$  with opposite sign, and in the same time  $y_i$  and  $Y_{pi}$  also with opposite sign. If not, it means that the sign for  $t_i$  must be change, and also we must make other changes, as one can see in the next two blocks.

11.7



In the C program it is chosen the sign -1 for  $t_i$ .

11.8



From the relations (C.6)..... (C.12) it's obvious that if the sign of  $t_i$  was changed, also the signs of  $r_1, r_2, r_3, r_4, r_5, r_6$ , and  $t_x$  must be changed. The C program must make here the following operations:

$$r_1 = -r_1, \quad (C.15)$$

$$r_2 = -r_2, \quad (C.16)$$

$$r_3 = -r_3, \quad (C.17)$$

$$t_x = -t_x, \quad (C.18)$$

$$r_4 = -r_4, \quad (C.19)$$

$$r_5 = -r_5, \quad (C.20)$$

$$r_6 = -r_6. \quad (C.21)$$

In the block 12 it must be computed the values for  $r_7, r_8$ , and  $r_9$  so, we have to implement in the C program the following relations:

$$r_7 = r_2 r_6 - r_3 r_5, \quad (C.22)$$

$$r_8 = r_3 r_4 - r_1 r_6, \quad (C.23)$$

$$r_9 = r_1 r_5 - r_2 r_4. \quad (C.24)$$

The block 13 has the same structure and the same function as the block 13 from Annex A, which was explained detailed in Annex A. The differences appears when it must be computed the values for  $x_i$ ,  $y_i$ , and the elements of the vector  $\mathbf{B}$ . The new relations are, as follows:

$$x_i = x_w r_1 + y_w r_2 + z_w r_3 + t_i, \quad (\text{C.25})$$

$$y_i = x_w r_4 + y_w r_5 + z_w r_6 + t_i, \quad (\text{C.26})$$

$$\begin{bmatrix} -x_i & -X_{di} \\ \cdot & \cdot \\ -x_N & -X_{dN} \\ -y_i & -Y_{di} \\ \cdot & \cdot \\ -y_N & -Y_{dN} \end{bmatrix} \begin{bmatrix} f \\ t_i \end{bmatrix} = \begin{bmatrix} x_{wi} r_1 X_{di} + y_{wi} r_2 X_{di} + z_{wi} r_3 X_{di} \\ \cdot \\ x_{wN} r_4 X_{dN} + y_{wN} r_5 X_{dN} + z_{wN} r_6 X_{dN} \\ x_{wi} r_7 Y_{di} + y_{wi} r_8 Y_{di} + z_{wi} r_9 Y_{di} \\ \cdot \\ x_{wN} r_{10} Y_{dN} + y_{wN} r_{11} Y_{dN} + z_{wN} r_{12} Y_{dN} \end{bmatrix} \quad (\text{C.27})$$

We also have to say that  $X_{di}$  is given by the relation following relation:

$$X_{di} = u_{di}^{-1} X_{di}^*, \quad (\text{C.28})$$

Using now this logical algorithm it will be easier to write the C program than starting directly, and making use only of the information from these five steps of Tsai calibration method, described in sub-chapter 3.3.3.

## C Program for Tsai Calibration Method

In this annex is presented the C program developed following the logical algorithm from Annex C. This C program implements Tsai calibration method.

```
//Liviu Toma C Program for Tsai Calibration Method
#include "matrix.h"
#include "Includes.h"
#include "calibration.h"

void main(void)
{
    int N=363,M;          //number of calibrationpoints
    int i,j,inv;
    double usx;
    double Tx,Ty,Tz,r1,r2,r3,r4,r5,r6,r7,r8,r9;
    double x,y;
    double dScale_in_X = 0.008509;
    double dScale_in_Y = 0.008316;
    double f;           //focal length
    double **coordXYZ_in_refSystem;
    double **coordXY_in_pixels;
    double **coordXY_on_chip;
    double **C;         //coefficients matrix
    double **INVC;
    double *Xd;
    double *A;
    double **CF;       //coefficentes matrix in used to compute f and Tz
    double **INVCF;
    double *BF;
    double *FTz;       //vector which contains two elements, f and Tz
    double **Trans;   //the real transformation
    double **ComputedTrans;
    double *TranslAngles;
    double *ComputedTranslAngles;
    double *DeltaTranslAngles;

    FILE *fvp;
    char field[10];
    M=2*N;

    //we reserve memory for all the vectors and matrices
    coordXYZ_in_refSystem = dmatrix(1,N,1,3);
    coordXY_in_pixels = dmatrix(1,N,1,2);
    coordXY_on_chip = dmatrix(1,N,1,2);
    C = dmatrix(1,N,1,7);
    INVC = dmatrix(1,7,1,N);
    Xd = dvector(1,N);
    A = dvector(1,7);
    CF = dmatrix(1,M,1,2);
    INVCF = dmatrix(1,2,1,M);
    BF = dvector(1,M);
    FTz = dvector(1,2);
    Trans = dmatrix(1,4,1,4);
    ComputedTrans = dmatrix(1,4,1,4);
    TranslAngles = dvector(1,6);
}
```

```

ComputedTranslAngles = dvector(1,6);
DeltaTranslAngles = dvector(1,6);

//printf("\nSemnul este: %d", SIGN(1));

//we open the file 3Dcoordinates.txt
fvp = fopen ("My3633Dcoordinates1.txt","r");
if(fvp==0)
{
    printf("Error while opening 3Dcoordinates.txt\n");
    getchar();
    exit(1);
}
for(i=1;i<=N;i++)
{
    //printf("\n");
    for(j=1;j<=3;j++)
    {
        fscanf(fvp,"%s",feld);
        coordXYZ_in_refSystem[i][j]=atof(feld);
        //printf("\t%f",coordXYZ_in_refSystem[i][j]);
    }
}
fclose (fvp);

//we open the file pixelsCoordinates.txt
fvp = fopen ("My363pixelsCoordinates1.txt","r");
if(fvp==0)
{
    printf("Error while opening pixelsCoordinates.txt\n");
    getchar();
    exit(1);
}
for(i=1;i<=N;i++)
{
    //printf("\n");
    for(j=1;j<=2;j++)
    {
        fscanf(fvp,"%s",feld);
        coordXY_in_pixels[i][j]=atof(feld);
        //printf("\t%f",coordXY_in_pixels[i][j]);
    }
}
fclose (fvp);

//compute the coefficients matrix C
for(i=1;i<=N;i++)
{
    //compute Xdi and Ydi
    coordXY_on_chip[i][1] = dScale_in_X*coordXY_in_pixels[i][1];
    Xd[i] = coordXY_on_chip[i][1];
    coordXY_on_chip[i][2] = dScale_in_Y*coordXY_in_pixels[i][2];

    //compute a line of matrix C
    C[i][1] = coordXY_on_chip[i][2]*coordXYZ_in_refSystem[i][1];
    C[i][2] = coordXY_on_chip[i][2]*coordXYZ_in_refSystem[i][2];
    C[i][3] = coordXY_on_chip[i][2]*coordXYZ_in_refSystem[i][3];
    C[i][4] = coordXY_on_chip[i][2];
    C[i][5] = coordXY_on_chip[i][1]*coordXYZ_in_refSystem[i][1];
    C[i][6] = coordXY_on_chip[i][1]*coordXYZ_in_refSystem[i][2];
    C[i][7] = coordXY_on_chip[i][1]*coordXYZ_in_refSystem[i][3];
}

```

```

//compute the matrix INVC
inv = invers(INVC,C,N,7);

//compute the vector A
mulmatrixVector(A, INVC, Xd, 7, N, N);

//compute the uncertainty of the scale factor usx
usx=sqrt(A[1]*A[1]+A[2]*A[2]+A[3]*A[3])/sqrt(A[5]*A[5]+A[6]*A[6]+A[7]
*A[7]);

//compute the translation Ty
Ty=1/sqrt(A[5]*A[5]+A[6]*A[6]+A[7]*A[7]);

//establish the right sign for Ty
//compute r1, r2, r3, Tx, r4, r5 and r6 for Ty chosen positive   if(
Ty<0)
if(Ty<0)
{Ty=-Ty;}

r1=(A[1]*Ty)/usx;
r2=(A[2]*Ty)/usx;
r3=(A[3]*Ty)/usx;
Tx=(A[4]*Ty)/usx;
r4=A[5]*Ty;
r5=A[6]*Ty;
r6=A[7]*Ty;

//compute x and y
x=r1*coordXYZ_in_refSystem[40][1]+r2*coordXYZ_in_refSystem[40][2]+r3*
coordXYZ_in_refSystem[40][3]+Tx;
y=r4*coordXYZ_in_refSystem[40][1]+r5*coordXYZ_in_refSystem[40][2]+r6*
coordXYZ_in_refSystem[40][3]+Ty;

//verify if the chosen sign for Ty is the good one
if((x*coordXY_in_pixels[40][1]<0)&&(y*coordXY_in_pixels[40][2]<0))
{}
else
{
    Ty=-Ty;
    Tx=-Tx;
    r1=-r1;
    r2=-r2;
    r3=-r3;
    r6=-r6;
}
//compute r7, r8 and r9
r7=r2*r6-r3*r5;
r8=r3*r4-r1*r6;
r9=r1*r5-r2*r4;

//compute the matrix CF and the vector BF

for(i=1;i<=N;i++)
{
    CF[i][1]=(r4*coordXYZ_in_refSystem[i][1])+(r5*coordXYZ_in_refSystem[i]
][2])+(r6*coordXYZ_in_refSystem[i][3])+Ty;
    CF[i][2]=coordXY_on_chip[i][2];
    BF[i]=-
CF[i][2]*((coordXYZ_in_refSystem[i][1]*r7)+(coordXYZ_in_refSystem[i][2]*r8)
+(coordXYZ_in_refSystem[i][3]*r9));
}

```



```

for(i=1;i<=N;i++)
{
    CF[N+i][1]=r1*coordXYZ_in_refSystem[i][1]+r2*coordXYZ_in_refSystem[i]
[2]+r3*coordXYZ_in_refSystem[i][3]+Tx;
    CF[N+i][2]=coordXY_on_chip[i][1]/usx;
    BF[N+i]=-
CF[N+i][2]*(coordXYZ_in_refSystem[i][1]*r7+coordXYZ_in_refSystem[i][2]*r8+c
oordXYZ_in_refSystem[i][3]*r9);
}

//compute INVCF
inv = invers(INVCF,CF,M,2);

//compute f and Tz
mulmatrixVector(FTz,INVCF,BF,2,M,M);
f=FTz[1];
Tz=FTz[2];

//write the matrix ComputedTrans
ComputedTrans[1][1]=r1;
ComputedTrans[1][2]=r2;
ComputedTrans[1][3]=r3;
ComputedTrans[1][4]=Tx;
ComputedTrans[2][1]=r4;
ComputedTrans[2][2]=r5;
ComputedTrans[2][3]=r6;
ComputedTrans[2][4]=Ty;
ComputedTrans[3][1]=r7;
ComputedTrans[3][2]=r8;
ComputedTrans[3][3]=r9;
ComputedTrans[3][4]=Tz;
ComputedTrans[4][1]=0.0;
ComputedTrans[4][2]=0.0;
ComputedTrans[4][3]=0.0;
ComputedTrans[4][4]=1.0;

//compute the vector ComputedTranslAngles
TransTo6Vector3(ComputedTrans,ComputedTranslAngles);

//print the solutions
printf("\nThe focal length is: %f", f);
printf("\nThe uncertainty of the scale factor is: %f",usx);

//printf("\nThe computed transformation is: \n");
//printdmatrix(ComputedTrans,4,4);
/*for(i=1;i<=4;i++)
{
    printf("\n");
    for(j=1;j<=4;j++)
    {
        printf("\t%f",ComputedTrans[i][j]);
    }
}*/

printf("\nThe computed transformation vector is: \n");
for(i=0;i<=5;i++)
{
    printf("\n%f",ComputedTranslAngles[i]);
}

```

```

//read the real transformation from the file RealTransformation.txt
inv=loadmatrix(Trans,4,4,"My3RealTransformation.txt");
//printdmatrix(Trans,4,4);

//compute the vector TranslAngles
TransTo6Vector3(Trans,TranslAngles);
/*printf("\nThe real transformation vector is: \n");
for(i=0;i<=5;i++)
{
    printf("\n%f",TranslAngles[i]);
}*/

//compute the difference vector DeltaTranslAngles
printf("\nThe vector DeltaTranslAngles is: ");
for(i=0;i<=5;i++)
{
    DeltaTranslAngles[i]=ComputedTranslAngles[i]-TranslAngles[i];
    printf("\n%f",DeltaTranslAngles[i]);
}

/*
printf("\nTx is: %f", Tx);
printf("\nTy is: %f", Ty);
printf("\nTz is: %f", Tz);
printf("\nf is: %f", f);
printf("\n%f", r1);
printf("\t%f", r2);
printf("\t%f", r3);
printf("\n%f", r4);
printf("\t%f", r5);
printf("\t%f", r6);
printf("\n%f", r7);
printf("\t%f", r8);
printf("\t%f", r9);          */

free_dmatrix(coordXYZ_in_refSystem, 1,N,1,3);
free_dmatrix(coordXY_in_pixels, 1,N,1,2);
free_dmatrix(C, 1,N,1,7);
free_dmatrix(INVC, 1,7,1,N);
free_dvector(Xd,1,N);
free_dvector(A,1,7);
free_dmatrix(CF, 1,M,1,2);
free_dmatrix(INVCF, 1,2,1,M);
free_dvector(BF, 1,M);
free_dvector(FTz, 1,2);
free_dmatrix(Trans, 1,4,1,4);
free_dmatrix(ComputedTrans, 1,4,1,4);
free_dvector(TranslAngles, 1,6);
free_dvector(ComputedTranslAngles, 1,6);
free_dvector(DeltaTranslAngles, 1,6);

getchar();
}

```

## MATLAB Programs for making the analysis of the measurement errors

In this annex are presented two MATLAB programs developed for making the analysis of the measurement errors. This analysis was discussed in chapter 6.

The first program is done only to represent the error distribution. The measurement errors are saved in a matrix having 5 columns. The first two columns represent the  $x$ ,  $y$  position of the measured point and the other three represents the errors for  $x$ ,  $y$  and  $z$  measured coordinates.

```
//Liviu Toma MATLAB Program for Error Distribution
a=[0 0 10 33 34
1 0 2 9 -37
2 0 0 19 -19
3 0 8 16 9
4 0 7 40 47
0 1 -16 -2 -45
1 1 -17 8 -41
2 1 -11 11 12
3 1 -9 14 -17
4 1 -4 25 57
0 2 -15 5 -33
1 2 -4 -8 -2
2 2 -16 -8 4
3 2 -1 3 -22
4 2 -2 1 76
0 3 -26 -1 -60
1 3 -8 -12 -47
2 3 -6 5 -22
3 3 -6 -13 -1
4 3 -1 -20 22
0 4 3 -15 28
1 4 -13 13 -93
2 4 -6 -41 41
3 4 0 -35 41
4 4 -21 -57 92

];

for i=1:25
    z(i,1)=sqrt(a(i,3)*a(i,3)+a(i,4)*a(i,4)+a(i,5)*a(i,5));
end

NX=30;
NY=30;

zz=zeros(5,5);
XI=linspace(0,4,NX);
YI=linspace(0,4,NY);
[XI,YI]=meshgrid(XI,YI);
X=a(:,1);
Y=a(:,2);
```

```

Z=a(:,5);
Z=u;

ZI=griddata(X,Y,Z,XI,YI,'w4');
plot(XI,YI,ZI)
contour(XI,YI, ZI)
surf(XI,YI,ZI)
colorbar
xlabel('x')
ylabel('y')
title('Z')
disp('media=')
disp(mean(a))
disp('max=')
disp(max(a))
disp('min=')
disp(min(a))
disp(max(a)-min(a))

```

The second program is done to represent the error distribution after the correction function was applied, (see 6.2.2).

```

//Liviu Toma MATLAB Program for Error Correction
a=[0 0 -0.20 0.34 0.82
1 0 -0.30 0.26 0.50
2 0 -0.29 0.25 0.21
3 0 -0.33 0.24 -0.03
4 0 -0.22 0.14 -0.36
5 0 -0.05 0.04 -0.60
6 0 0.27 -0.08 -0.94
0 1 -0.10 0.27 0.93
1 1 -0.25 0.14 0.53
2 1 -0.24 0.13 0.32
3 1 -0.24 0.08 0.02
4 1 -0.15 0.06 -0.25
5 1 0.10 -0.01 -0.49
6 1 0.32 -0.07 -0.75
0 2 -0.19 0.12 0.95
1 2 -0.24 0.11 0.56
2 2 -0.22 0.11 0.28
3 2 -0.23 0.11 0.09
4 2 -0.08 0.08 -0.13
5 2 0.14 0.08 -0.42
6 2 0.36 0.04 -0.66
0 3 -0.12 0.05 0.95
1 3 -0.25 0.08 0.57
2 3 -0.19 0.09 0.32
3 3 -0.22 0.15 0.12
4 3 -0.03 0.16 -0.09
5 3 0.15 0.16 -0.38
6 3 0.34 0.19 -0.58
0 4 -0.04 -0.22 0.95
1 4 -0.17 -0.21 0.65
2 4 -0.14 -0.13 0.35
3 4 -0.11 -0.11 0.14
4 4 -0.01 -0.14 -0.11
5 4 0.20 -0.13 -0.25
6 4 0.40 -0.12 -0.64
0 5 -0.02 -0.19 0.90

```

```

1 5 -0.12 -0.17 0.67
2 5 -0.08 -0.12 0.37
3 5 -0.07 -0.10 0.16
4 5 0.01 -0.08 -0.06
5 5 0.24 -0.09 -0.27
6 5 0.43 -0.03 -0.55
0 6 -0.02 -0.42 0.86
1 6 -0.10 -0.42 0.65
2 6 -0.08 -0.34 0.40
3 6 -0.09 -0.27 0.18
4 6 0.04 -0.27 -0.05
5 6 0.27 -0.21 -0.29
6 6 0.50 -0.16 -0.56

];

B = a(:,3);

N=7;
for J = 1:N
    for I = 1:N
        a((J-1)*7+I,3)=0.00156*(I-2.75)*(I-2.75)*(I-2.75)*(I-
2.75)+0.038*(J-4)-0.125;
    end
end

NX=30;
NY=30;

zz=zeros(7,7);
XI=linspace(0,6,NX);
YI=linspace(0,6,NY);
[XI,YI]=meshgrid(XI,YI);
X=a(:,1);
Y=a(:,2);
Z=a(:,3);

R = Z-B;
a(:,3)=R;
ZI=griddata(X,Y,R,XI,YI,'nearest');
surf(XI,YI,ZI)
colorbar
xlabel('x')
ylabel('y')
zlabel('E_x')
disp('media=')
disp(mean(a))
disp('max=')
disp(max(a))
disp('min=')
disp(min(a))
disp(max(a(:,3))-min(a(:,3)))

```