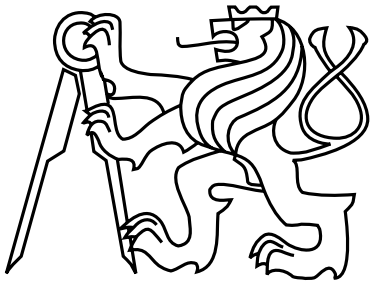
Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics



Bachelor Thesis

Reconstruction of Spheres in 3D for Robot Calibration

Nikita Litvishko

Supervisor: doc. Ing. Tomáš Pajdla, PhD.

Study Program: Cybernetics and Robotics, Bachelor

Field of Study: Systems and Control

May 24, 2018

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Litvishko Nikita** Personal ID number: **456867**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

Branch of study: **Systems and Control**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Reconstruction of spheres in 3D for robot calibration

Bachelor's thesis title in Czech:

Rekonstrukce koulí v prostoru pro kalibraci robotů

Guidelines:

1. Study principles of sphere projection to uncalibrated and calibrated images [1,2].
2. Suggest an approach to reconstructing a pair of touching spheres in 3D space.
3. Implement the approach and verify it on real data.

Bibliography / sources:

[1] Hui Zhang, Kwan-Yee K. Wong, and Guoqiang Zhang. Camera Calibration from Images of Spheres. IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 29, NO. 3, MARCH 2007 (http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4069265)
[2] M. Agrawal and L. S. Davis, "Camera calibration using spheres: a semi-definite programming approach," Proceedings Ninth IEEE International Conference on Computer Vision, Nice, France, 2003, pp. 782-789 vol.2.
doi: 10.1109/ICCV.2003.1238428

Name and workplace of bachelor's thesis supervisor:

doc. Ing. Tomáš Pajdla, Ph.D., Applied Algebra and Geometry, CIIRC

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **15.01.2018** Deadline for bachelor thesis submission: **25.05.2018**

Assignment valid until: **30.09.2019**

doc. Ing. Tomáš Pajdla, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Ing. Pavel Ripka, CSc.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Aknowledgements

I would like to thank my advisor Tomáš Pajdla for introducing me into the world of research, showing me how to creatively solve the problems in projective geometry and for his guidance and advices that helped me to finish this work.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date ....

signature

Abstract

Computer vision is a fast and widely developing branch of science. One of the problems and tasks of computer vision is the determining of a particular object on the image and its position in space. In this work, we propose our algorithm, which allows us to uniquely determine the sphere in space. We concentrate on studying the theory, which directly relates to the theme of this work and apply this theory in practice to obtain a solution.

As a result, we have a working algorithm that gives a satisfactory result on real data. In the future, this algorithm can be used, for example, to calibrate the robot in such a way that determining the position of the sphere will determine the position of the last joint of robotic arm.

Keywords: computer vision, conic sections, quadrics, camera calibration

Abstrakt

Počítačové vidění je rychle rozvijející se oblast vědy. Jedním z problemů a úkolů počítačového vidění je detekce konkretního objektu na obrázku a určení jeho polohy v prostoru. V této práci navrhujeme algoritmus, který umožňuje jednoznačně určit polohu sféry v prostoru. Soustřeďujeme se na teorii, která se přímo vztahuje k tématu této práce, a používáme ji v praxi pro získání řešení.

Ve výsledku dostáváme funkční algoritmus, který dobře funguje při zkoušení na reálných datech. V budoucnu tento algoritmus může být použit, například, pro kalibraci robota, a to tak, že určení pozice sféry určí i pozici konce robotické ruky.

Klíčová slova: počítačové vidění, kuželosečka, kvadrika, kalibrace kamery

Contents

List of Figures

Chapter 1

Introduction

Computer vision is a fast and widely developing branch of science. In our time it is closely related to artificial intelligence, namely, with the recognition and detection of objects. Extracting information from 2d images of 3D objects is also one of the most common branches of computer vision. It combines the solution of polynomial equations, linear algebra and projective geometry.

In this work, we focused our attention on recreating a sphere on the basis of its image. Our main task was to propose an algorithm that would allow unambiguous determination of the position of the sphere in space. Similar problem has been solved by Zisserman and Cross in [3]. They used two cameras to get stereo image and to determine the position of the object, called quadric.

The motivation for this work was a group of researchers who are engaged in calibrating the robot. They need to determine the position of the sphere in space, which is located on the end of the robotic arm, in order to accurately determine the coordinates of the last joint and to calibrate the robot.

This thesis can be divided to the following sections:

1. In the beginning we will conduct theoretical introduction in our problem, that is needed for better understanding the problem and that we needed to suggest the solution.

2. Then, we will move on with the main part of this thesis, namely we will suggest a solution to the given problem and discuss problems, that could show up.

3. After, we will test the solution on the real data and we will see the limitations of the algorithm.

4. Finally, we will summarize this work.

Chapter 2

Basic theory

In this chapter we will go through the theory, that we had to study to successfully tackle the issue. Theory consists of basic knowledge in computer vision and geometry, such as surfaces in 3D and different conic sections. We also studied the interaction between surfaces in 3D and its projections on the planes. Thus, we divide our theoretical introduction into the following sections

1. Perspective camera model
2. Conic sections
3. Quadrics
4. Projection of the sphere onto the image plane

2.1 Perspective camera model

There exist many types of cameras. We will concentrate ourselves on the perspective camera model, that maps points from 3D space onto the image plane. Basically, points on the image plane are given by the intersection between the vector, connecting the camera center with the point, and the image plane.

Chapters 6 and 7 in book [10] describe the basics of perspective camera model. In this section we will briefly summarize the main of these two chapters. We will use the notation of [10] and [5].

Let us consider following illustration 2.1

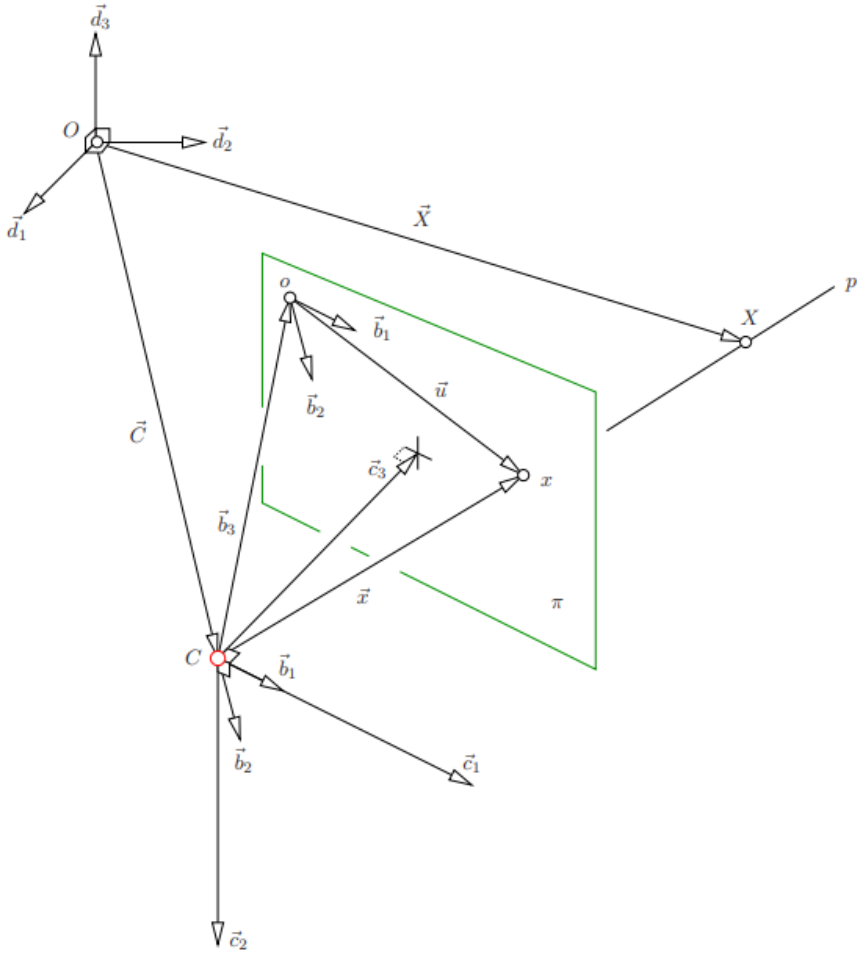


Figure 2.1: Coordinate systems of perspective camera [10].

Figure 2.1 shows the geometry of the perspective camera. We can see there the camera center C, the image plane π and the point O, which is the origin for the main coordinate system.

To solve the problems of computer vision, such as extracting information about 3D object from 2D image, we will define some handful coordinate systems.

First, we introduce the *world coordinate system* (O, δ), so we can express any point in the space with its help. This coordinate system consists of the origin O and orthonormal basis δ containing vectors $\mathbf{d}_1, \mathbf{d}_2$ and $\mathbf{d}_3$, so every point can be written as

$$\mathbf{X}_\delta = \delta_1 \mathbf{d}_1 + \delta_2 \mathbf{d}_2 + \delta_3 \mathbf{d}_3 \tag{2.1.1}$$

World coordinate system allows us to define the *camera projection center* $\mathbf{C}_\delta$.

Next we define the *image coordinate system* (o, α) of the image plane π. We use this coordinate system to express the points on the image plane through its origin o and two non-orthogonal basis vectors $\mathbf{b}_1$ and $\mathbf{b}_2$. So, any point on the image plane can be represented as vector in basis α

$$\mathbf{u}_{i_\alpha} = u_i\mathbf{b}_1 + v_i\mathbf{b}_2 = \begin{bmatrix} u_i \\ v_i \end{bmatrix} \tag{2.1.2}$$

Finally, to associate the coordinates of points on the image with the camera center, we introduce the *camera coordinate system* (C, β). Its origin is always placed in the projection center and its basis consists of basis α with added vector $\mathbf{b}_3$. We define vector $\mathbf{b}_3$ as a vector connecting origins of two coordinate systems β and α. Thus, it is more convenient to write coordinates of the points in the image plane with respect to basis β, because the third coordinate will always be equal to one. At this point, we can express above vector $\mathbf{u}_\alpha$ in basis β by simply adding one, i.e.

$$\mathbf{u}_{i_\beta} = \begin{bmatrix} \mathbf{u}_{i_\alpha} \\ 1 \end{bmatrix} \tag{2.1.3}$$

Now let us take an arbitrary point X in the space. If we start drawing the line from the camera center C to point X, we will reach the image plane at the point x. We define such point x as the projection of point X onto the image plane and we can represent it with respect to β as

$$\mathbf{x}_\beta = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{2.1.4}$$

If we want to represent point X with respect to the camera coordinate system, we can imagine, that point X in β can be obtained from $\mathbf{x}$ as its multiple by some non-zero number η. Thus, we can write

$$\eta\mathbf{x}_\beta = \mathbf{X}_\beta - \mathbf{C}_\beta \tag{2.1.5}$$

Now, suppose that we have the matrix $\mathbf{A}$, that converts any vector from space, given with respect to the world coordinate system δ, to the vector represented by the camera coordinate system β. We can write this as

$$\mathbf{x}_\beta = \mathbf{A}\mathbf{x}_\delta \tag{2.1.6}$$

The matrix $\mathbf{A}$, as we will see later, contains the *extrinsic* and *intrinsic* parameters of the camera. We will give the definition to these parameters a little bit later.
Using Equation 2.1.6 we can re-write Equation 2.1.5 as

$$\eta\mathbf{x}_\beta = \mathbf{A}(\mathbf{X}_\delta - \mathbf{C}_\delta) \tag{2.1.7}$$

$$\eta\mathbf{x}_\beta = \mathbf{A}\begin{bmatrix} \mathbf{I} | & -\mathbf{C}_\delta \end{bmatrix}\begin{bmatrix} \mathbf{X}_\delta \\ 1 \end{bmatrix} \tag{2.1.8}$$

$$\eta \mathbf{x}_\beta = \mathbf{P}_\beta \begin{bmatrix} \mathbf{X}_\delta \\ 1 \end{bmatrix} \tag{2.1.9}$$

We just have introduced the 3×4 *image projection matrix* $\mathbf{P}_\beta$.
With help of Equation 2.1.9 we can describe the relationship between the point in the space $\mathbf{X}_\delta$ and the point on the image plane $\mathbf{u}_\alpha$. If we know the value of η and matrix $\mathbf{A}$, we can obtain the coordinates of the point in space from Equation 2.1.7 as

$$\mathbf{X}_\delta = \eta \mathbf{A}^{-1} \mathbf{x}_\beta + \mathbf{C}_\delta \tag{2.1.10}$$

Now we will look closer to the matrix $\mathbf{A}$. We define it as

$$\mathbf{A} = \frac{1}{f} \mathbf{K}\mathbf{R} \tag{2.1.11}$$

where we introduce two new matrices $\mathbf{K}$, $\mathbf{R}$ and parameter f, which is the *focal length* and is defined as the distance between the camera center and the image plane.

We have already told, that the matrix $\mathbf{A}$ contains parameters of the camera. We have mentioned *extrinsic* and *intrinsic* parameters of the camera. Matrix $\mathbf{K}$ determines the intrinsic parameters, that are given by camera construction and do not change when moving or rotating the camera. This matrix is given as follows

$$\mathbf{K} = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ 0 & k_{22} & k_{23} \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1.12}$$

The elements of matrix $\mathbf{K}$ are defined as

$$k_{11} = \frac{f}{\|\mathbf{b}_1\|} \tag{2.1.13}$$

$$k_{12} = -\frac{f \cos\angle(\mathbf{b}_1, \mathbf{b}_2)}{\|\mathbf{b}_1\| \sin\angle(\mathbf{b}_1, \mathbf{b}_2)} \tag{2.1.14}$$

$$k_{22} = \frac{f}{\|\mathbf{b}_2\| \sin\angle(\mathbf{b}_1, \mathbf{b}_2)} \tag{2.1.15}$$

Elements k_{13} and k_{23} determine the coordinates of the *principal point* in α, that is given as intersection between vector $\mathbf{c}_3$ and the image plane.

Now we will look at the second matrix introduced in Equation 2.1.11. This is 3×3 rotation matrix, that determines the extrinsic parameters of the camera, i.e. its orientation. Matrix $\mathbf{R}$ and vector $\mathbf{C}_\delta$ uniquely define the position and orientation of the camera.

To continue, we will define new orthogonal camera coordinate system (C, γ) with three basis vectors $\mathbf{c}_1, \mathbf{c}_2$ and $\mathbf{c}_3$. These three vectors are chosen in such way, that $\text{Span}\{\mathbf{c}_1, \mathbf{c}_2\}$ defines the plane parallel to the image plane π. Vector $\mathbf{c}_3$ is orthogonal to both $\mathbf{c}_1$ and $\mathbf{c}_2$ and thus is orthogonal to the image plane and has the length equals to the focal length.

The basis γ holds

$$\mathbf{c}_1 = k_{11} \mathbf{b}_1 \tag{2.1.16}$$

$$\mathbf{c}_2 = k_{12} \mathbf{b}_1 + k_{22} \mathbf{b}_2 \tag{2.1.17}$$

$$\mathbf{c}_3 = k_{13}\mathbf{b}_1 + k_{23}\mathbf{b}_2 + 1\mathbf{b}_3 \tag{2.1.18}$$

By using matrices $\mathbf{K}$, $\mathbf{R}$, $\mathbf{A}$ and focal length we can provide next vectors transformations between coordinate systems

$$\mathbf{y}_\beta = \mathbf{K}\mathbf{y}_\gamma \tag{2.1.19}$$

$$\mathbf{y}_\gamma = \frac{1}{f}\mathbf{R}\mathbf{y}_\delta \tag{2.1.20}$$

$$\mathbf{y}_\beta = \mathbf{A}\mathbf{y}_\delta \tag{2.1.21}$$

Inverse transformations could be obtained by applying matrices $\mathbf{K}^{-1}$, $\mathbf{R}^{-1}$ and $\mathbf{A}^{-1}$.

Using above equations we can express given vector with respect to any of the coordinate systems.

Using Equation 2.1.11 and Equation 2.1.8 we obtain

$$f\eta\mathbf{x}_\beta = \mathbf{K}\mathbf{R}\begin{bmatrix}\mathbf{I}| & -\mathbf{C}_\delta\end{bmatrix}\begin{bmatrix}\mathbf{X}_\delta \\ 1\end{bmatrix} \tag{2.1.22}$$

$$f\eta\mathbf{x}_\beta = \mathbf{P}\begin{bmatrix}\mathbf{X}_\delta \\ 1\end{bmatrix} \tag{2.1.23}$$

where 3×3 matrix $\mathbf{P}$ is called the *camera projection matrix*.

2.2 Conic sections

Due to the perspective camera model all projections of the objects do not preserve their actual size and in most cases may have different form, than in the real world. For example, if we project the circle, that does not lie on the plane, which is parallel to the image plane and its center does not intersect with the ray given by vector $\mathbf{c}_3$, then we will obtain an ellipse. When increasing an angle between the image plane and the plane, where circles lie, the difference between circle and ellipse would be more evident. Figure 2.2 clearly demonstrates this.

Figure 2.2: (a) The images of circles are almost represented as circles. (b) The circles are represented as ellipses [5].

When working with the images of spheres in real life, we are also dealing with circles and ellipses. By [4] conic section is the name given to the shapes that we obtain by taking different plane slices through a double cone. In our case the cone is produced by inscribed sphere and a vertex placed in the camera projection center. In general, we may consider the following conic sections

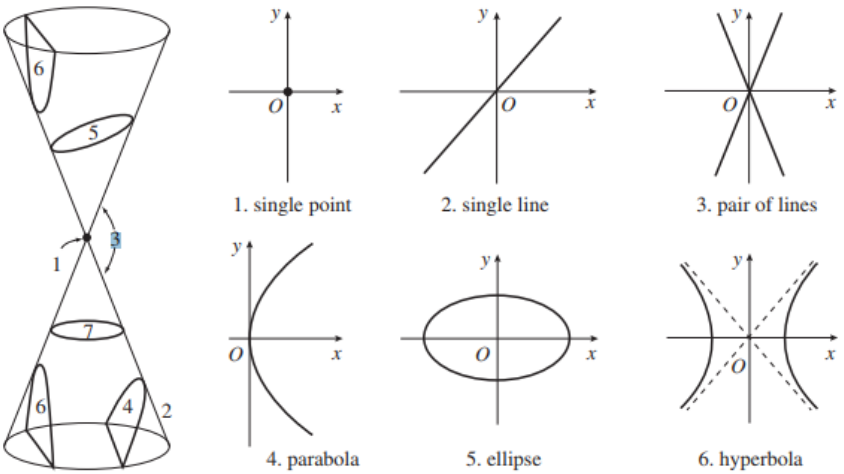


Figure 2.3: Different conic sections [4].

We can see, that there are 6 types of conic sections. However, in the real life the plane will never pass through the vertex, which follows from non-zero focal length. Also, we may consider finite image planes, which implies, that plane will never intersect given sphere. From above words follows, that the only conic section, which can be obtained is ellipse. To suggest the solution to our problem of reconstructing sphere in the space by given image we had to study basics of conic sections.

By [4] any conic has an equation of the form

$$Ax^2 + Bxy + Cy^2 + Fx + Gy + H = 0 \tag{2.2.1}$$

where A, B, C, F, G and H are real numbers, and not all of A, B and C are zero.
To work with conic sections in computer vision, we have to express them in the matrix form as

$$\mathbf{x}^{\mathrm{T}}\mathbf{A}_{33}\mathbf{x} + \mathbf{J}^{\mathrm{T}}\mathbf{x} + H = 0 \tag{2.2.2}$$

where matrices $\mathbf{A}_{33}$ and $\mathbf{J}$ contain coefficients A, B, C, F, G and the vector $\mathbf{x}$ represents a point on the conic. These two matrices are given as

$$\mathbf{A}_{33} = \begin{bmatrix} A & B/2 \\ B/2 & C \end{bmatrix} \tag{2.2.3}$$

$$\mathbf{J} = \begin{bmatrix} F \\ G \end{bmatrix} \tag{2.2.4}$$

Equation 2.2.2 is the equation of the conic in inhomogeneous coordinates [5]. We can use this equation if we work only with α coordinate system of the perspective camera model. To work with another coordinate systems, for example, β we should use homogeneous coordinates.

So, we can re-write above equation as

$$\mathbf{X}^{\mathrm{T}}\mathbf{q}\mathbf{X} = 0 \tag{2.2.5}$$

where matrix $\mathbf{q}$ is the symmetric invertible (in case of ellipse, hyperbola and parabola) matrix and can be written as

$$\mathbf{q} = \begin{bmatrix} A & B/2 & F/2 \\ B/2 & C & G/2 \\ F/2 & G/2 & H \end{bmatrix} \tag{2.2.6}$$

We can see, that the upper left 2×2 submatrix is the matrix $\mathbf{A}_{33}$. The matrix $\mathbf{A}_{33}$ is very useful for us, because with its help we can determine if obtained conic section is a circle or an ellipse. If coefficients A and C are equal and $B = 0$, then conic section is a circle.

We have defined conics through the points. Due to the duality we can represent the same conic through lines. We call this *dual conic* and it is also represented by 3×3 matrix $\mathbf{q}^*$ [5]. For dual conics we can write

$$\mathbf{l}^{\mathrm{T}}\mathbf{q}^*\mathbf{l} = 0 \tag{2.2.7}$$

where $\mathbf{l}$ is the tangent to a conic line. If matrix $\mathbf{q}^*$ is not singular and is symmetric, then $\mathbf{q}^* = \mathbf{q}^{-1}$. Later we will use this to obtain the projection of the sphere onto the image plane.

2.3 Quadrics

Spheres are wide-used objects in computer vision [6][7][9]. If given sphere without any markers on it, then such sphere does not have an orientation and thus is easier to describe. Also in many cases it is easy to detect the apparent contour of the sphere on the image. To work with spheres in computer vision, we would have to express them generally in matrix form.

First, we will write general equation for quadric surface [1]. Such surface is the set of points, that satisfy the equation

$$Ax^2 + By^2 + Cz^2 + 2Dxy + 2Eyz + 2Fxz + 2Gx + 2Hy + 2Jz + K = 0 \tag{2.3.1}$$

The above equation can be represented in the matrix form by 4×1 homogeneous vector $\mathbf{X}$, that represents a point on the quadric, and a 4×4 matrix $\mathbf{Q}$ as

$$\mathbf{X}^{\mathrm{T}}\mathbf{Q}\mathbf{X} = 0 \tag{2.3.2}$$

where matrix $\mathbf{Q}$ is

$$\mathbf{Q} = \begin{bmatrix} A & D & F & G \\ D & B & E & H \\ F & E & C & J \\ G & H & J & K \end{bmatrix} \tag{2.3.3}$$

If we set $A = B = C = -K = 1$ and $D = E = F = G = H = J = 0$, we get a unit sphere with center placed at the origin.

The sphere in general case is represented by its center a and radius r. If we define γ as $a^T a - r^2$ [6], sphere equation in matrix form will be represented as

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & -a_1 \\ 0 & 1 & 0 & -a_2 \\ 0 & 0 & 1 & -a_3 \\ -a_1 & -a_2 & -a_3 & \gamma \end{bmatrix} = \begin{bmatrix} \mathbf{I} & -a \\ -a^{\mathrm{T}} & \gamma \end{bmatrix} \tag{2.3.4}$$

Obtained matrix $\mathbf{Q}$ will be used in the next section to construct a projection onto the image plane.

2.4 Projection of a sphere into a perspective image

In this section we will use knowledge obtained in the previous two sections to obtain conic section for a concrete given sphere.

Let us assume, that we are given the sphere

$$\mathbf{S} \equiv \mathbf{Q} = \begin{bmatrix} \mathbf{I} & -a \\ -a^{\mathrm{T}} & \gamma \end{bmatrix} \tag{2.4.1}$$

and matrices $\mathbf{K}$, $\mathbf{R}$ and a focal length. Thus, we can construct the camera projection matrix $\mathbf{P}$. Now we will write the equation for the dual conic $\mathbf{q}^*$ [5][6] of conic $\mathbf{q}$ as

$$\mathbf{q}^* = \mathbf{P}\mathbf{Q}^*\mathbf{P}^{\mathrm{T}} \tag{2.4.2}$$

where $\mathbf{Q}^*$ is the dual quadric of $\mathbf{Q}$.

By substituting $\mathbf{q}^* = \mathbf{q}^{-1}$ and $\mathbf{Q}^* = \mathbf{Q}^{-1}$ into Equation 2.4.2 we get

$$\mathbf{q}^{-1} = \mathbf{P}\mathbf{Q}^{-1}\mathbf{P}^{\mathrm{T}} \tag{2.4.3}$$

$$\mathbf{q} = (\mathbf{P}\mathbf{Q}^{-1}\mathbf{P}^{\mathrm{T}})^{-1} \tag{2.4.4}$$

Equation 2.4.4 is the projection of quadric $\mathbf{Q}$ onto the image plane.

Chapter 3

Sphere reconstruction

In this chapter we will suggest a solution to the problem of sphere reconstruction from the given image. Briefly, our idea is that conic section and the camera center define the concrete cone and there is only one point, where center of the sphere can be placed so that sphere would be inscribed in this cone.

When we want to recover the sphere from given conic section, we can obtain two cases:

1. Obtained conic section is an ellipse
2. Obtained conic section is a circle

3.1 Ellipse

Basically, we can divide our solution into the following steps:

1. Calculate eigenvectors and eigenvalues of the matrix $\mathbf{A}_{33}$
2. Calculate the center and vertices of an ellipse
3. Find an angle θ between two vectors connecting major vertices and calculate a point M, where angle bisection and major axis intersect
4. Determine points X and Y where orthogonal to the major axis line passing through M intersects with ellipse.
5. Determine sphere's parameters by given radius and compute the distance between sphere's center and the camera projection center
6. Using obtained direction and distance, calculate the actual position of the sphere

Now we will go through each item above and look closer at the suggested solution.

3.1.1 Eigenvectors and eigenvalues of $\mathbf{A}_{33}$

First, we need to extract the matrix $\mathbf{q}$, representing the conic section. In the real scene we used "ellipse detection" for given photos and then more precisely found the contour of an ellipse and used program for ellipse fitting [12].

Now assume we are given matrix $\mathbf{q}$. We easily obtain matrix $\mathbf{A}_{33}$ from $\mathbf{q}$ and then calculate its eigenvectors and eigenvalues by the definition

$$\mathbf{A}_{33}\mathbf{e} = v\mathbf{e} \tag{3.1.1}$$

where $\mathbf{e}$ is the eigenvector corresponding to the eigenvalue v. Thus, we obtain two eigenvectors corresponding to two different eigenvalues, which implies that these vectors are orthogonal [8]. By the principal axis theorem, the axes of an ellipse are orthogonal, so they will be parallel to the eigenvectors of matrix $\mathbf{A}_{33}$. Thus, by far we have two eigenvectors $\mathbf{e}_1$ and $\mathbf{e}_2$ and corresponding to them eigenvalues v_1 and v_2.

3.1.2 Ellipse's vertices

Our next step is to calculate the coordinates of the ellipse's center and then obtain the equations for the axes. We define the center of an ellipse as a point, where major and minor axes intersect. If we multiply first two columns of matrix $\mathbf{q}$ by vector $\mathbf{x} = \begin{bmatrix} x & y & 1 \end{bmatrix}^{\mathrm{T}}$ we get two equations that give us one solution, which is our required center [2]. Thus, we will get

$$\begin{cases} Ax + (B/2)y + D/2 = 0 \\ \\ (B/2)x + Cy + G/2 = 0 \end{cases} \tag{3.1.2}$$

By solving it we obtain the point (x_c, y_c).

We have already seen, that the minor and major axes are parallel to two eigenvectors. The eigenvector corresponding to the smaller eigenvalue will be parallel to the major axis, and eigenvector corresponding to the larger eigenvalue will be parallel to the minor axis [2]. We need to find two points, where major axis crosses an ellipse. Thus, we again solve the system of non-linear equations, which are

$$\begin{cases} \mathbf{x}^{\mathrm{T}}\mathbf{q}\mathbf{x} = 0 \\ \\ y = \dfrac{e_2 x - x_c e_2 + y_c e_1}{e_1} \end{cases} \tag{3.1.3}$$

where $\mathbf{x}$ is 3×1 vector representing homogeneous coordinates, $\mathbf{q}$ is the conic matrix and (e_1, e_2) is the eigenvector corresponding to the smaller eigenvalue. Thus, we obtained two points in the image coordinate system α, which can be represented in β as vectors

$$\mathbf{a} = \begin{bmatrix} a_1 & a_2 & 1 \end{bmatrix}^{\mathrm{T}} \tag{3.1.4}$$

$$\mathbf{b} = \begin{bmatrix} b_1 & b_2 & 1 \end{bmatrix}^{\mathrm{T}} \tag{3.1.5}$$

3.1.3 Projection of the sphere's center

In the previous part we obtained two vectors $\mathbf{a}$ and $\mathbf{b}$ and now we will calculate the angle between them. We will use well known formula

$$\cos\theta = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \cdot \|\mathbf{b}\|} \tag{3.1.6}$$

Angle Θ will be used a little bit later.

The second part of this step is to calculate the coordinates of point M and relevant vector $\mathbf{m}$, given by intersection between bisection-line and the major axis. From geometry we know, that bisection will divide line into two parts that are proportional to the lengths of surrounding lines. This is illustrated in Figure 3.1

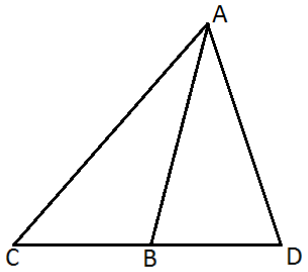


Figure 3.1: Angle bisection.

$$\frac{|BD|}{|BC|} = \frac{|AD|}{|AC|} \tag{3.1.7}$$

With its help we can calculate vector $\mathbf{m}$ as

$$\mathbf{m} = \frac{\|\mathbf{a}\|}{\|\mathbf{a}\| + \|\mathbf{b}\|}\mathbf{a} + \frac{\|\mathbf{b}\|}{\|\mathbf{a}\| + \|\mathbf{b}\|}\mathbf{b} \tag{3.1.8}$$

Now we will deviate from our train of thought a bit and image the process of obtaining an ellipse as illustrated in Figure 3.2. Imagine, we have a cone and a plane Ω, that intersects it, is parallel to the cone's base. Now we choose an axis around which we will rotate the plane Ω. Our chosen axis u is the line, that lies in the plane Ω and intersects with cone's height. Also, we denote two points, where u intersects with the cone, as X and Y. To achieve the rotation, we use some rotation matrix $\mathbf{R}_u(\phi)$. By the definition of any rotation matrix we have

$$\mathbf{R}\mathbf{v} = \mathbf{v} \tag{3.1.9}$$

where $\mathbf{v}$ is vector that is parallel to the rotation axis. Equation 3.1.9 tells us, that after rotation vector $\mathbf{v}$ will be unchanged. We have a little bit more difficult situation, because we do not rotate the conic section, but we want to rotate the plane Ω, so we could get new conic sections. In our case the only unchanged line in every obtained ellipse (and a circle, which we had at the beginning) after application the rotation, is the line that coincides with

axis u. Such line will maintain its length and orientation, unlike major and minor axes of obtained ellipse.
This approval can be demonstrated in the following illustrations in Figure 3.2

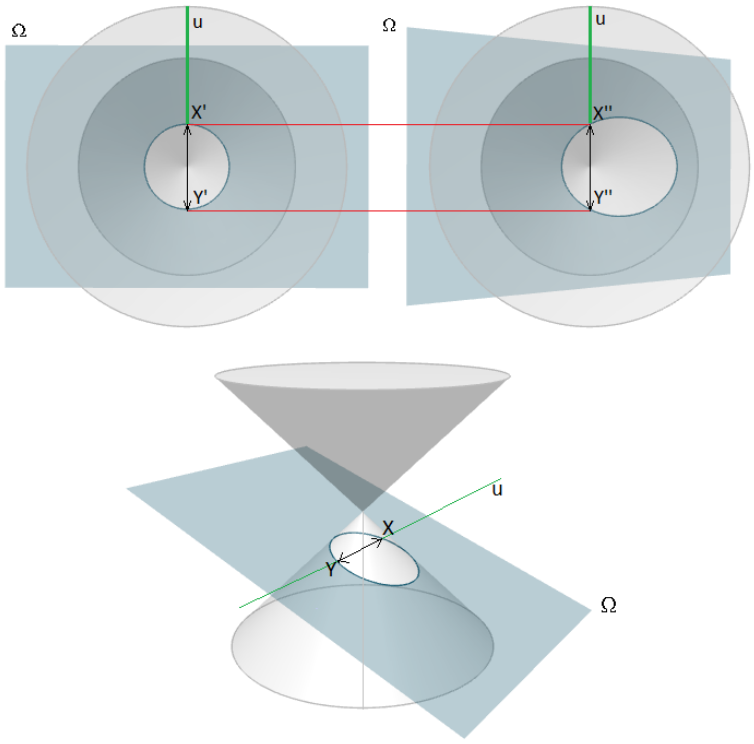


Figure 3.2: Line lying on the rotation axis does not change the length [11].

We see, that lengths $X'Y'$ and $X''Y''$ are equal. Due to the fact, that cone is constructed by given sphere and vertex in the camera center we can tell, that the cone's height will pass through the sphere's center and if we take any two opposite rays, that have beginning in the vertex and if these to rays and the cone's height form a plane, then the cone's height will be the bisection of the angel between two rays. In the set of such pairs of rays only two rays can easily determine an angle θ. We can see in Figure 3.2, that such rays will pass through major vertices of obtained conic section. From all of the above we can conclude, that some multiple of the vector obtained in Equation 3.1.8 will pass through the sphere's center.

3.1.4 Rotation axis

From the above part we can see how useful are points, where orthogonal to the major axis line, which also contains point M, intersects with an ellipse. We have already obtained the equation of the line, that is the major axis, by solving system of equations 3.1.3. We also know, that this line was given by the point (x_c, y_c) and direction vector $\begin{bmatrix} e_1 & e_2 \end{bmatrix}^{\mathrm{T}}$. The line, that is orthogonal to the major vertex will have the direction vector equals to $\begin{bmatrix} -e_2 & e_1 \end{bmatrix}^{\mathrm{T}}$. If we assume, that the point M has coordinates (m_1, m_2), we can obtain the intersection

between the line and conic by solving next system of non-linear equations

$$\begin{cases} \mathbf{x}^{\mathrm{T}}\mathbf{q}\mathbf{x} = 0 \\ \\ y = -\dfrac{e_1 x - e_1 m_1 - e_2 m_2}{e_2} \end{cases} \tag{3.1.10}$$

The solution will give us two points X and Y, which, as we saw in Figure 3.2, correspond to X' and Y' (same as X'' and Y''). Now we will compute the length l of XY by very well known formula

$$l = \sqrt{(X_1 - Y_1)^2 + (X_2 - Y_2)^2} \tag{3.1.11}$$

We will use obtained length l in the next step.

3.1.5 Sphere's parameters

In this part of solution we need to switch to 2D to easily calculate the distance between the camera center and the sphere's center.

If we build a plane, that passes through the cone's vertex and line XY, we will get the following cross section

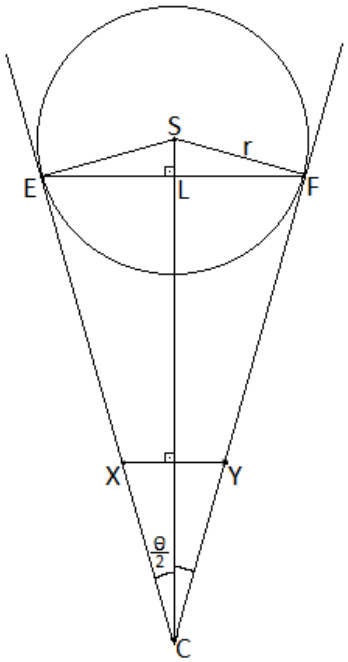


Figure 3.3: 2D view of sphere's recovering problem.

The sphere inscribed in the cone is reduced to the circle inscribed in the triangle with radius r. Our goal in this step is to calculate the distance CS, which we define as d. Further, we define the diameter of the visible contour of the sphere, that is circle with diameter

$EF = d_c$. The distance to the sphere's center will be a sum of $SL = n$ and $CL = p$. Both lengths we can compute as

$$n = r\sin\frac{\theta}{2} \tag{3.1.12}$$

$$p = \frac{\|\mathbf{m}\|d_c}{l} \tag{3.1.13}$$

where $d_c = 2r\cos\frac{\theta}{2}$. Then the distance between the camera center and a sphere is given by $d = (n + p)$. In the next step we will finally determine the position of the sphere in the space.

3.1.6 Recovering the position

The sphere's center is given by vector $\mathbf{s}$, that has the direction and magnitude. We have already obtained the direction of this vector in β. We need to normalize vector $\mathbf{m}$ and multiply it by distance d. Then we have to transform obtained vector into δ coordinate system. It could be done as follows

$$\mathbf{s}_\delta = \mathbf{A}^{-1}\frac{\mathbf{m}}{\|\mathbf{m}\|} \cdot d = f\mathbf{R}^{-1}\mathbf{K}^{-1}\mathbf{s}_\beta \tag{3.1.14}$$

Now we have to add vector $\mathbf{C}_\delta$ to compute the actual sphere's center. Thus, we have

$$\mathbf{X}_{s_\delta} = \mathbf{C}_\delta + \mathbf{s}_\delta \tag{3.1.15}$$

where $\mathbf{X}_{s_\delta}$ is the vector, determining the position of the sphere.

3.2 Circle

The second case can be achieved only if the sphere's center lies along $\mathbf{c}_3$ of γ coordinate system. Unlike first case, we do not have to calculate the direction, because it is already given by $\mathbf{c}_3$. All we need to do is to calculate the distance from the camera center to the sphere's center. Methods for circle detecting in the images return the circle's center and its radius, so we can use Equations 3.1.12, 3.1.13, 3.1.14 and 3.1.15 to obtain the sphere's center.

Chapter 4

Real data

We used MATLAB camera calibrator application [14] to calibrate camera and simultaneously took one of the calibration positions as reference position for camera to test our solution. Yellow point, named as checkerboard origin, was chosen as origin for the world coordinate system, where axis Z directs from an observer. In Figure 4.1 we can see, where is the origin of the world coordinate system.

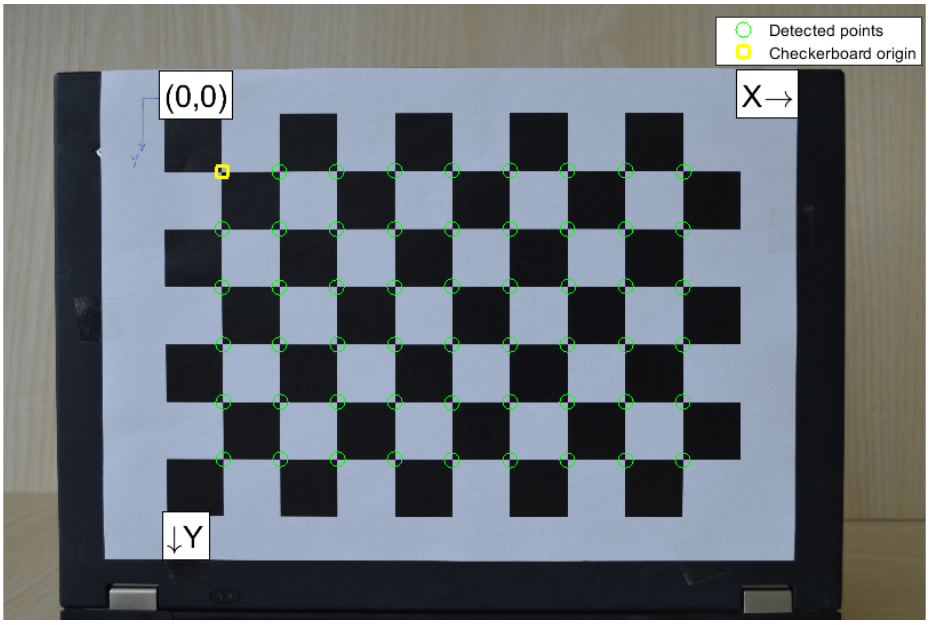


Figure 4.1: Checkerboard used for calibrations and yellow square determining the world origin.

We chose such origin for easier determining the real position of the sphere, because sphere was always touching the checkerboard, so its third coordinate was always equal to $-r$, where r is the radius of sphere.

4.1 Single sphere

In the following sections we will look at the results of different experiments, in which we were changing the distance between the camera center and a sphere.

Each measurement we were calibrating the camera to get more accurate results and to get exact camera position and orientation in the space, i.e. to get precise extrinsic parameters. Matrix $\mathbf{K}$ always had following form

$$\mathbf{K} = \begin{bmatrix} f_x & s & p_1 \\ 0 & f_y & p_2 \\ 0 & 0 & 1 \end{bmatrix}$$

We have used Nikon D5100 for our experiments and a checkerboard printed on A4 with 24.5 mm boards for calibrations [13].

All of the distances, except for focal length and a principal point are shown in millimeters. For radius measuring we have used digital caliper, where diameter was measured ten times. Then we have computed the mean value as

$$\bar{r} = \frac{1}{n} \sum_{i=1}^{n} r_i \tag{4.1.1}$$

Using Equation 4.1.1 we have obtained the following radii for white sphere and a tennis ball

$$r_1 = 26.96$$

$$r_2 = 31.05$$

Next, we denote $\mathbf{s}_r$ as the real sphere's position and $\mathbf{s}_c$ as calculated position of the sphere.
All results were rounded to one decimal place in s_1, s_2 and two decimal places in case of radius and s_3.

4.1.1 Distance 420 mm

After calibrating the camera we obtained following intrinsic and extrinsic parameters

$$f_x = 4181$$

$$f_y = 4179$$

$$s = 1.5$$

$$\mathbf{p} = \begin{bmatrix} 2492 & 1638 \end{bmatrix}^{\mathrm{T}}$$

$$\mathbf{K} = \begin{bmatrix} f_x & s & p_1 \\ 0 & f_y & p_2 \\ 0 & 0 & 1 \end{bmatrix}$$

The extrinsic parameters of camera are as follows

$$\mathbf{R} = \begin{bmatrix} 0.999 & 0.005 & -0.006 \\ -0.005 & 0.999 & 0.005 \\ 0.006 & -0.005 & 0.999 \end{bmatrix}$$

$$\mathbf{C}_\delta = \begin{bmatrix} 103.9 & 61.3 & -420.8 \end{bmatrix}^{\mathrm{T}}$$

Then we tested our algorithm for the following case:

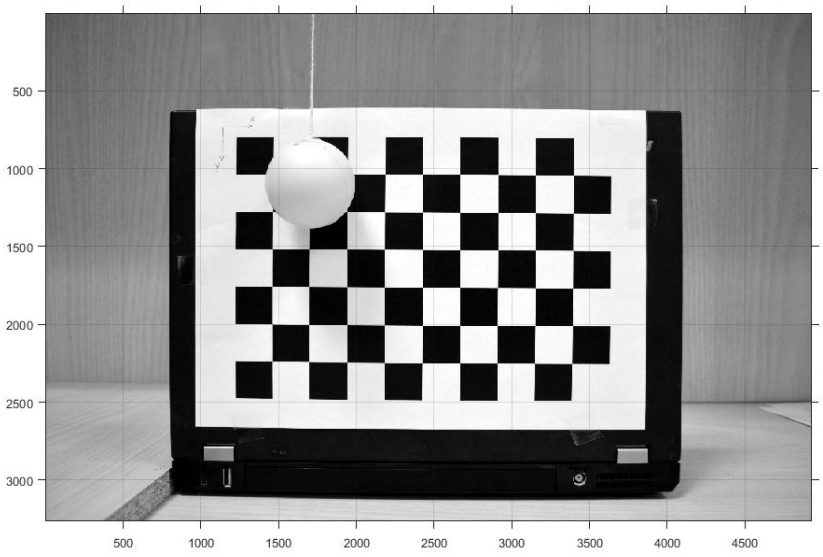


Figure 4.2: One sphere. First case.

$$\mathbf{s}_r = \begin{bmatrix} 28 \\ 8 \\ -26.96 \end{bmatrix}$$

$$\mathbf{s}_c = \begin{bmatrix} 32.4 \\ 7.9 \\ -27.82 \end{bmatrix}$$

The Euclidean distance between $\mathbf{s}_r$ and $\mathbf{s}_c$ is

$$d = 4.46$$

The error can be attributed to mistakes in camera calibrating, or error caused by ellipse fitting program.

4.1.2 Distance 710 mm

Now we will move camera away from the sphere to the distance 718 mm. After calibrating we obtained

$$f_x = 5306$$

$$f_y = 5303$$

$$s = 1.4$$

$$\mathbf{p} = \begin{bmatrix} 2473 & 1634 \end{bmatrix}^{\mathrm{T}}$$

The extrinsic parameters of camera are as follows

$$\mathbf{R} = \begin{bmatrix} 0.999 & 0.005 & -0.003 \\ -0.005 & 0.999 & 0.001 \\ 0.003 & -0.001 & 0.999 \end{bmatrix}$$

$$\mathbf{C}_{\delta} = \begin{bmatrix} 100 & 62 & -718 \end{bmatrix}^{\mathrm{T}}$$

Then we tested our algorithm for the following case:

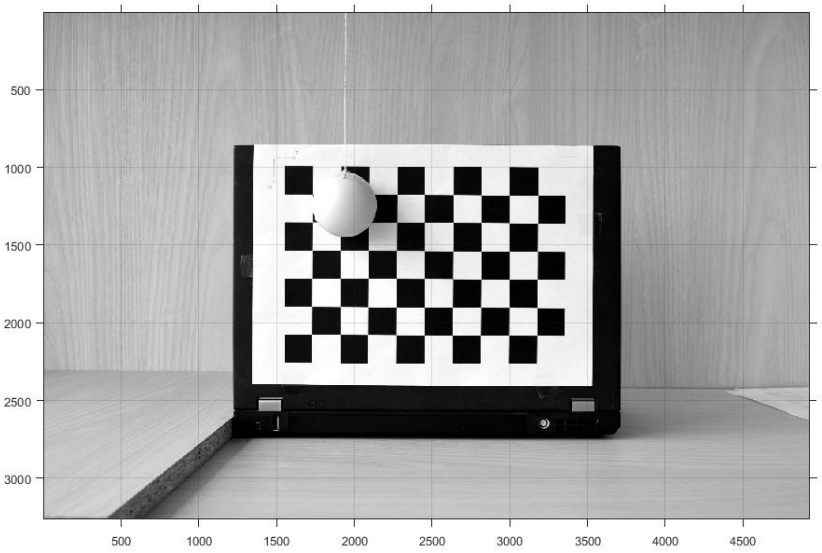


Figure 4.3: One sphere. Second case.

Real sphere's position was about

$$\mathbf{s}_r = \begin{bmatrix} 28 \\ 9 \\ -26.96 \end{bmatrix}$$

And computed

$$\mathbf{s}_c = \begin{bmatrix} 32.7 \\ 10 \\ -29.3 \end{bmatrix}$$

Now we compute the error, that is expressed in the Euclidean distance

$$d = 5.37$$

We can see, that the error in this case is a little bit higher. For example, when distance between the camera center and a sphere was about 420 mm, then the difference between the real third coordinate of the sphere and computed coordinate was

$$d_{420} = 0.86$$

In the case, when the distance is almost twice higher than 420 mm, the error in calculation the third coordinate increased to

$$d_{710} = 2.34$$

Obtained d_{420} and d_{710} tell us, that calculated distance to the sphere is very much influenced by correctly defined radius, since the main idea of our algorithm was that in concrete cone we can inscribe only one sphere with given radius. Especially since the height of the cone is much bigger than sphere's radius, an angle θ is very small, thus little changes in radius cause huge changes in the distance. Assume, for example, that the distance between the sphere and camera center d_{x_1} is 750 mm and radius is equal to 26 mm. Then angle θ is approximately equal to

$$\theta \approx \frac{\frac{r}{2}}{d_{x_1}} \approx 0.0173[rad] \approx 1.98^o$$

Now we can compute, how much will change the distance, if the radius is changed by millimeter.

$$d_{x_2} = 778.8 \Rightarrow \Delta d_x = 28.8$$

From the above result we can conclude, that it is not very optimal to calculate the position of the sphere, when sphere is placed far away from the camera.

4.1.3 Distance 280 mm

In this experiment we placed the sphere close to the camera. After camera calibrating we obtained the following parameters

$$f_x = 5306$$

$$f_y = 5304$$

$$s = 2.3$$

$$\mathbf{p} = \begin{bmatrix} 2485 & 1635 \end{bmatrix}^{\mathrm{T}}$$

The extrinsic parameters of camera are as follows

$$\mathbf{R} = \begin{bmatrix} 0.999 & 0.004 & -0.009 \\ -0.004 & 0.999 & -0.002 \\ 0.009 & 0.002 & 0.999 \end{bmatrix}$$

$$\mathbf{C}_\delta = \begin{bmatrix} 99 & 58 & -286 \end{bmatrix}^{\mathrm{T}}$$

Then we tested our algorithm for the following case:

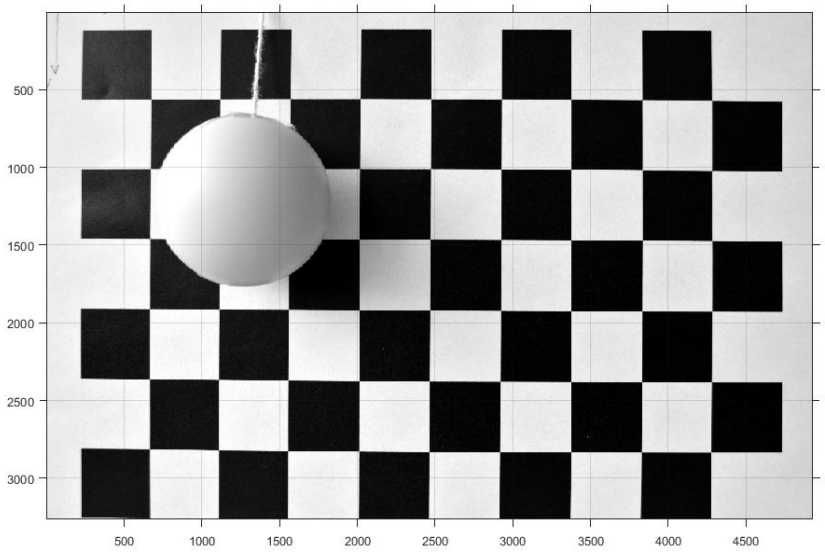


Figure 4.4: One sphere. Third case.

Real sphere's position

$$\mathbf{s}_r = \begin{bmatrix} 41 \\ 41 \\ -26.96 \end{bmatrix}$$

Computed sphere's position

$$\mathbf{s}_c = \begin{bmatrix} 39.3 \\ 40.3 \\ -19.94 \end{bmatrix}$$

And an error

$$d = 7.24$$

At such distance radius measurement has lesser influence on the result, but correctly defined ellipse will affect accuracy more, than in the previous experiments. In Figure 4.5 we can see, that the apparent contour of the sphere is not precise. There are seven pixels that could possibly determine the same point on the ellipse.

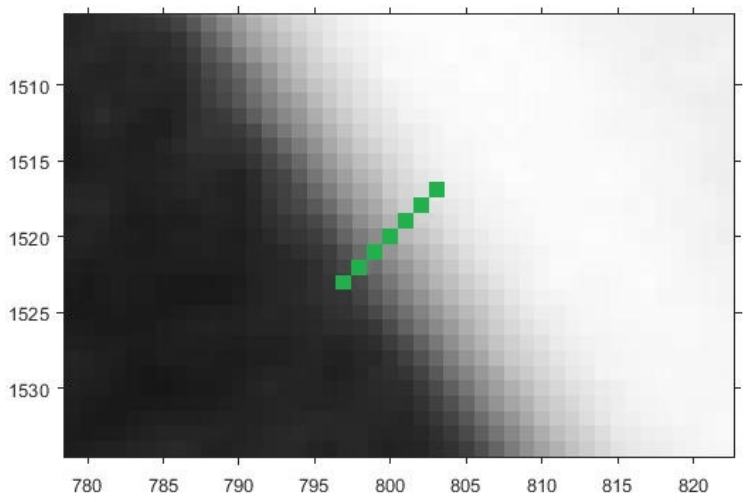


Figure 4.5: Not precise contour of the sphere.

Let us conclude the results of three experiments.

1. If sphere is placed far away from the camera, then it is necessary to determine the radius of the sphere in the most precise way. Also, the noise in determining the sphere's contour will also increase with increasing distance between the sphere's center and the camera center. If sphere is placed at the very long distance, then just few pixels will determine its outline.

2. By placing the sphere close to the camera, we face the problem of determining the contour of an ellipse in the sense, that many pixels may define the same point of the ellipse.

3. Based on the results of the experiments, the most optimal way is to have the sphere at such distance from the camera, so that each point of the ellipse will be determined by two or three pixels.

4.2 Two Spheres

In the following two sections we will look, how does our algorithm work with two spheres. We leaved one sphere from the previous experiments and added a tennis ball. Then we tried to determine the position of the spheres in two cases

1. Both spheres' centers were lying in the plane, which was almost parallel to the image plane, thus spheres did not overlap one another.

2. One sphere was closer to the camera center, so that not entire contour of the second sphere could be seen.

In general, there are some cases, when reconstruction of two touching spheres could not be completed with high accuracy or could not be completed at all. For example, if one sphere is behind another, then it is not possible to recover the second sphere from single image.

When the second sphere is behind first one with particularly hidden contour, then it is possible to recover the second sphere, even though the accuracy will be low. Decreasing in accuracy can be explained by the fact, that points, needed to fit the ellipse, will be on the little interval, so it will be worst, than in the case, when whole contour is visible.

4.2.1 **Distance** 410 mm

After calibrating the camera we have obtained following parameters

$$f_x = 5297$$

$$f_y = 5295$$

$$s = 1.3$$

$$\mathbf{p} = \begin{bmatrix} 2481 & 1629 \end{bmatrix}^{\mathrm{T}}$$

The extrinsic parameters of camera are as follows

$$\mathbf{R} = \begin{bmatrix} 0.999 & 0.004 & -0.006 \\ -0.004 & 0.999 & -0.005 \\ 0.006 & 0.005 & 0.999 \end{bmatrix}$$

$$\mathbf{C}_\delta = \begin{bmatrix} 100 & 60 & -417 \end{bmatrix}^{\mathrm{T}}$$

Then we tested our algorithm for the following case:

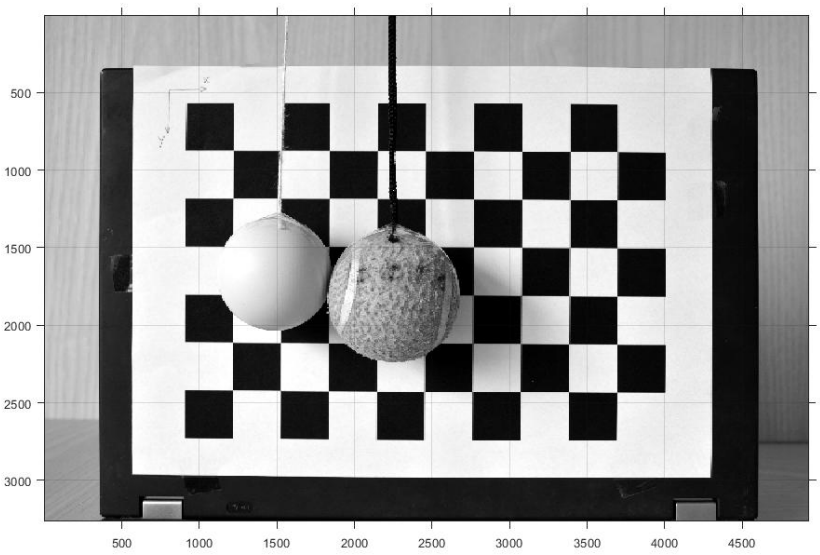


Figure 4.6: Two spheres. First case.

Real 1st sphere's position was about

$$\mathbf{s}_r = \begin{bmatrix} 30 \\ 67 \\ -26.96 \end{bmatrix}$$

And computed

$$\mathbf{s}_c = \begin{bmatrix} 28.7 \\ 67.9 \\ -24.8 \end{bmatrix}$$

$$d_1 = 2.68 \tag{4.2.1}$$

Real 2nd sphere's position was

$$\mathbf{s}_r = \begin{bmatrix} 80 \\ 90 \\ -31.05 \end{bmatrix}$$

And computed

$$\mathbf{s}_c = \begin{bmatrix} 86.6 \\ 78.9 \\ -23.06 \end{bmatrix}$$

$$d_2 = 15.16$$

The error obtained for the second sphere can be attributed to the wrong ellipse contour detecting, since tennis ball does not have an ideal surface, what can be illustrated in the Figure 4.7.

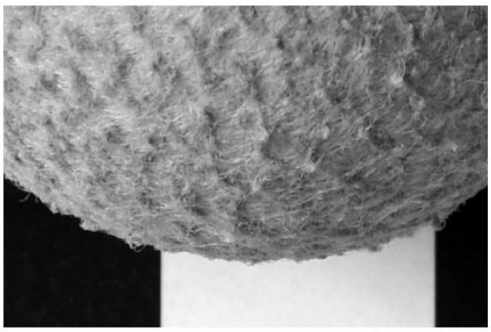


Figure 4.7: Structure of the surface of tennis ball.

As we can see, it is not clear, where is the contour of the sphere. The points of the contour of the tennis ball were taken manually. We were always taking twenty points to fit an ellipse. Then we fitted an ellipse ten times with different contour points to obtain the mean value of the ellipse's center and then to calculate the standard deviation. Mean value was computed using Equation 4.1.1 and the standard deviation was given by

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\bar{x} - x_i)^2} \tag{4.2.2}$$

After calculations, we have obtained

$$\bar{\mathbf{c}} = \begin{bmatrix} 2264.7 & 1795.3 \end{bmatrix}^{\mathrm{T}}$$

$$\mathbf{s} = \begin{bmatrix} 15.6 & 15.9 \end{bmatrix}^{\mathrm{T}}$$

Then, we have computed the ratio of standard deviations s_1 and s_2 to the mean values c_1 and c_2, where s_1 and s_2 are deviations along x and y axes, c_1 and c_2 are the mean values of x-position and y-position of the ellipse's center.

$$p_x = \frac{s_1}{c_1} = 0.687\% \tag{4.2.3}$$

$$p_y = \frac{s_2}{c_2} = 0.886\% \tag{4.2.4}$$

Now we will do the same for the sphere with smooth surface in Figure 4.6

$$\bar{\mathbf{c}} = \begin{bmatrix} 1484.3 & 1667.2 \end{bmatrix}^{\mathrm{T}}$$

$$\mathbf{s} = \begin{bmatrix} 2.7 & 1.9 \end{bmatrix}^{\mathrm{T}}$$

$$p_x = \frac{s_1}{c_1} = 0.183\% \tag{4.2.5}$$

$$p_y = \frac{s_2}{c_2} = 0.114\% \tag{4.2.6}$$

From the above results we can see, that p_x and p_y computed for tennis ball are 4-8 times bigger, than p_x and p_y of the sphere with smooth surface, so the error in determining the contour of the ellipse is higher, when dealing with balls, which do not have an ideal surface.

4.2.2 Distance 360 mm

In this experiment we tried to recover spheres in the case, that one was covering the silhouette of another. After camera calibrating we obtained

$$f_x = 5291$$

$$f_y = 5289$$

$$s = 0.99$$

$$\mathbf{p} = \begin{bmatrix} 2477 & 1622 \end{bmatrix}^{\mathrm{T}}$$

The extrinsic parameters of camera are as follows

$$\mathbf{R} = \begin{bmatrix} 0.999 & 0.005 & -0.01 \\ -0.005 & 0.999 & 0 \\ 0.01 & 0 & 0.999 \end{bmatrix}$$

$$\mathbf{C}_\delta = \begin{bmatrix} 100 & 60 & -477 \end{bmatrix}^{\mathrm{T}}$$

Then we tested our algorithm for the following case:

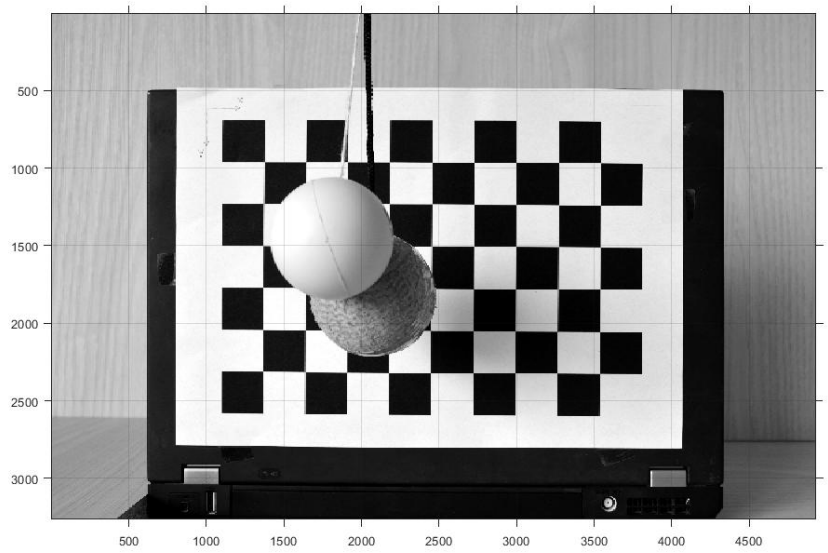


Figure 4.8: Two spheres. Second case.

Real 1st sphere's position was about

$$\mathbf{s}_r = \begin{bmatrix} 54 \\ 48 \\ -118 \end{bmatrix}$$

And computed

$$\mathbf{s}_c = \begin{bmatrix} 58.2 \\ 51.4 \\ -112.4 \end{bmatrix}$$

$$d_1 = 7.78$$

Real 2nd sphere's position was about

$$\mathbf{s}_r = \begin{bmatrix} 66 \\ 74 \\ -65 \end{bmatrix}$$

And computed

$$\mathbf{s}_c = \begin{bmatrix} 76.2 \\ 77.7 \\ -60.9 \end{bmatrix}$$

$$d_2 = 11.57$$

In this experiment we cover the apparent contour of the tennis ball with the scotch tape, so that its surface was smoother. We can see, that this increased the accuracy of computing the position of the tennis ball.

4.3 Changing the radius

We have already told, that the accuracy in radius measuring is very important for the good result. To confirm this words, we will take a closer look at the Equations 3.1.12 and 3.1.13. Since the distance to the sphere is given by

$$d = n + p = r\sin\frac{\theta}{2} + \frac{\|\mathbf{m}\|d_c}{l} = r\left(\sin\frac{\theta}{2} + \frac{2\|\mathbf{m}\|}{l}\cos\frac{\theta}{2}\right) \tag{4.3.1}$$

From Equation 4.3.1 we see, that the distance d is linearly dependent on the defined radius. The farther the radius from the correct value, the greater the error in determining the sphere's position. In taken photo, parameters such as $\|\mathbf{m}\|$, θ and l are uniquely calculated and only the radius is independent on the experiment. So, we can think of Equation 4.3.1 as follows

$$d = k \cdot r \tag{4.3.2}$$

In Equation 4.3.2 k determines the constant, that is uniquely defined for each experiment. The computed distance to the sphere will change when decreasing or increasing the radius. The dependence of the error on a correctly defined radius can be given by Equation 4.3.3

$$y_{err} = k \cdot |\Delta r| \tag{4.3.3}$$

In Equation 4.3.3 y_{err} defines the error in computing the position of the sphere, k is the constant, computed for each photo, r is the chosen radius. Graphically the relation can be shown in Figure 4.9

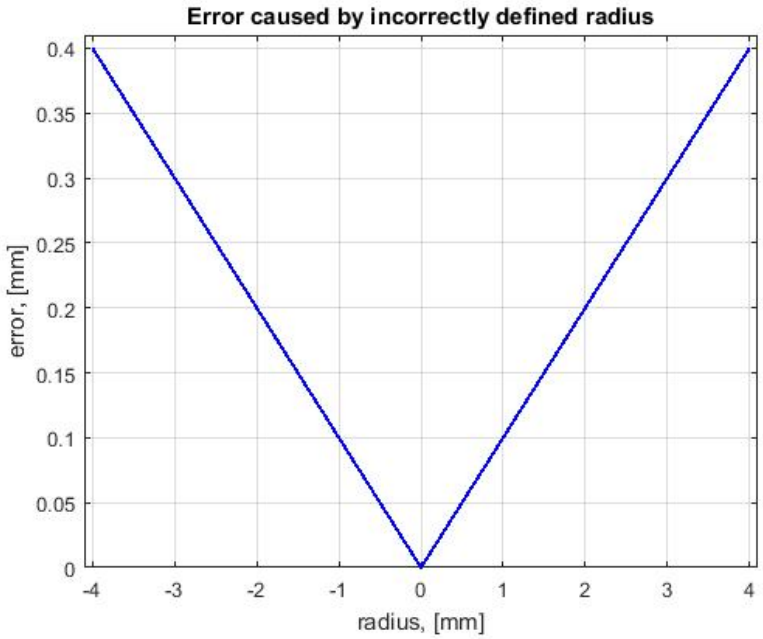


Figure 4.9: The relation between an error in computing and error in radius measurement.

Constant k, as we can see in Equations 4.3.1 and 4.3.2 is partially defined by computed angle, so we can say, that with increasing angle, the influence of the radius on the result decreases.

4.4 Relative position of the sphere

In the previous experiments we calculated the absolute position of the sphere, i.e. where given sphere is placed with respect to the world coordinate system. The results were depended on many factors, such as correctly calibrated camera, correctly defined ellipse, correctly measured real position of the sphere and what is most important - radius measurements. To evaluate the accuracy of our algorithm, in this section we will calculate the relative position of the sphere, i.e. we would not have to know, where the sphere is placed, what are extrinsic parameters of the camera and what is the real radius of the sphere.
We made such experiment in the following way:

1. We fixed the camera and manually set such a focal length, that the sphere was always in focus. Equipment is illustrated in Figure 4.10.

Figure 4.10: Equipment used during the experiment.

2. We calibrated the camera, so we knew the elements of matrix **K** and we were able to represent the result in the world units.

3. We placed the sphere on the device, that allowed us to accurately change the position by one tenth of millimeter. Used device is shown in Figure 4.11

Figure 4.11: Device, which allowed accurate changes in the sphere's position.

4. Beginning with a starting position, we were moving the sphere by 4 mm each time and took 39 photos in total.

5. Finally, for each photo we were calculating the position of the sphere $\eta\mathbf{x}$ and then the Euclidean distance between positions in two neighbouring measurements.

Supposedly, all computed distances had to be the same, since the position of the sphere has always been changed by the same value. The result of this experiment is shown below

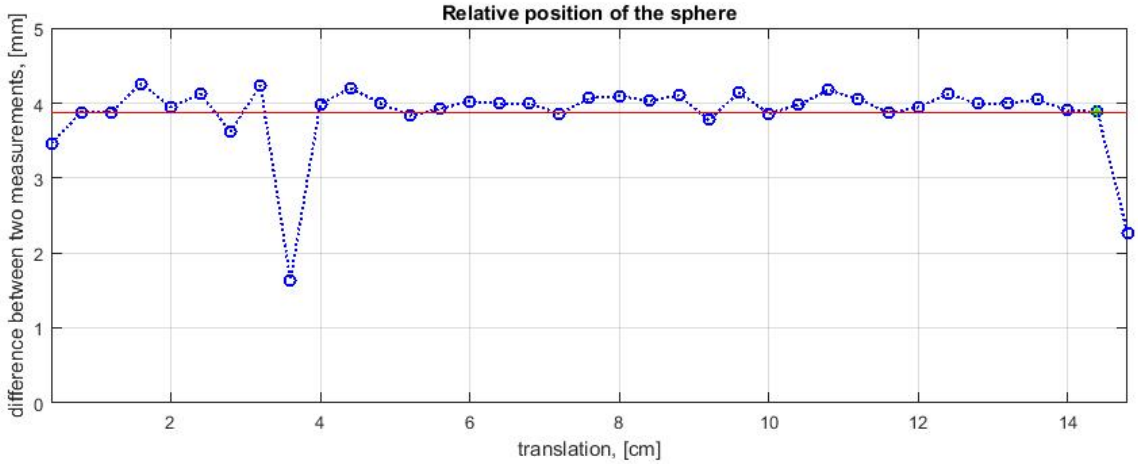


Figure 4.12: Graph illustrating the relation between computed translation and an actual translation.

Figure 4.13 demonstrates the functionality of the ellipse detecting program. The red dot denotes the center of an ellipse and the red curve denotes the contour of the detected ellipse.

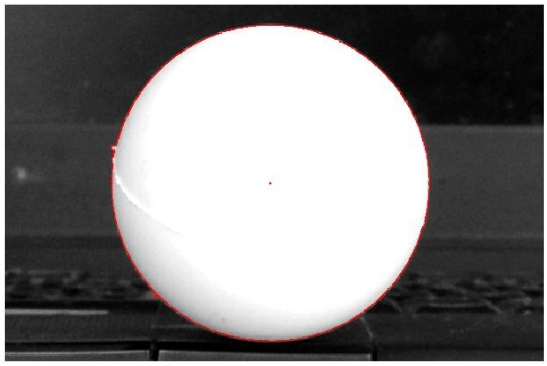


Figure 4.13: Detected ellipse.

Red line in Figure 4.12 determines the mean value of all computed distances. As we can see, just two measurements have significant deviation from the mean value. This can be attributed to the small mistakes in the ellipse detecting program. For example, let us take one measurement, that is marked by green star in Figure 4.12. The difference between the centers' positions in two photos is 1.882 pixels. The difference of centers between photos, that gave us big error, is 0.876 pixels. So, we can see, that even one pixel in detecting the ellipse's center can cause such an error.

Without taking in consideration measurements, where errors appeared, we can see in Figure 4.14, that the remaining translations were very close to 4 mm.

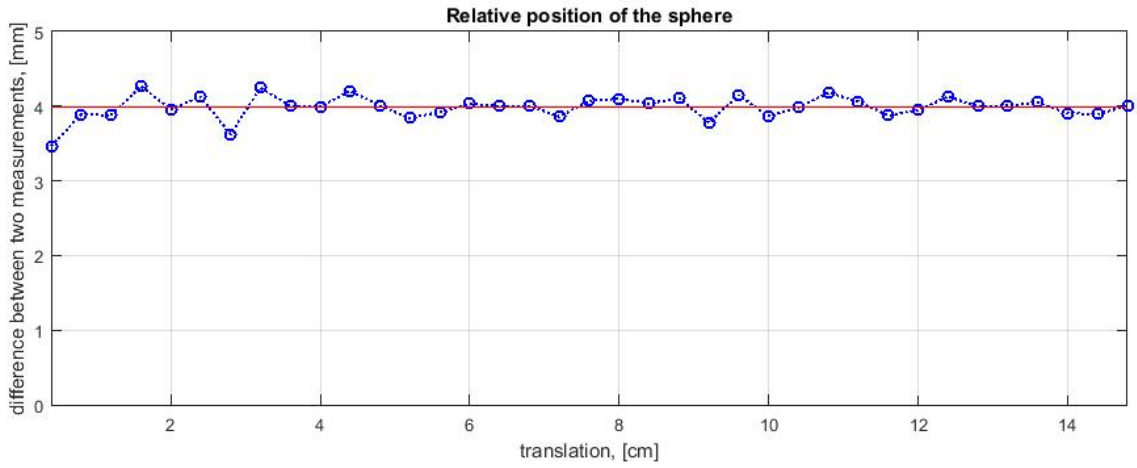


Figure 4.14: Graph illustrating the relation between computed translation and an actual translation. Two measurements with big errors were removed.

Since it is not possible to remove two photos, because the distance has always been computed between neighbouring measurements, we just have changed the value of the computed translation to four, so it does not influence the mean value and the standard deviation. Computed mean value $\bar{t}$ of all translations t and the standard deviation, based on the results shown in Figure 4.14, are equal to

$$\bar{t} = 3.98 \tag{4.4.1}$$

$$\sigma = 0.16 \tag{4.4.2}$$

Based on the results of experiment we can conclude, that the position of the sphere is calculated with satisfied accuracy.

Chapter 5

Conclusion

In this work, we have focused on sphere's reconstruction in 3D space from 2D image.

In the second chapter, we looked at the basic theory that was needed to better understand the problem and for its successful solution. We studied the perspective model of the camera and its geometry, the conic sections and planes of the second order, called quadrics. At the end of the second chapter, we applied the formula described in [5] and [6] to construct the projection of the sphere onto the plane.

In the third chapter we showed that it is possible to recreate the sphere in space based on the received photo and taking into account the knowledge of the extrinsic and intrinsic parameters of the camera. We proposed an algorithm that uniquely determines the position of the sphere in space.

In the last chapter we saw how our algorithm behaves in practice and checked its functionality on real data. The undoubted benefit of testing an algorithm on the real data was that we could see and analyze affection of extraneous factors, such as lightening, background and surface of a sphere. We have seen, how they influence the accuracy of determining the position of the sphere and how important it is to accurately determine the radius of a sphere, especially if the sphere is at a great distance from the camera. Then, we tried to recreate a couple of touching balls and saw that the position of the ball, whose surface was not smooth, was calculated with much less accuracy than the position of the ball with a perfect surface.

In the end, we conducted an experiment in which the camera calibration errors and radius determination errors were removed due to the fact that the relative position of the sphere was considered, and not the absolute.

Bibliography

[1] J. R. Miller. "Analysis of Quadric-Surface-Based Solid Models". In: *University of Kansas* (1988).

[2] A. B. Ayoub. "The central conic sections revisited". In: *Mathematics Magazine* (1993), pp. 322–325. URL: https://www.jstor.org/stable/2690513?newaccount=true&read-now=1&seq=3#page_scan_tab_contents.

[3] A. Zisserman G. Gross. "Quadric Reconstruction from Dual-Space Geometry". In: *Proceedings of the Sixth International Conference on Computer Vision* (1998), p. 25.

[4] Matthew F. Esplen David A. Brannan and Jeremy J. Gray. *Geometry*. Cambridge University Press, 1999.

[5] A. Zisserman R. Hartley. *Multiple view geometry in computer vision*. volume 2. Cambridge Univ Press, 2000.

[6] M. Agrawal and L. S. Davis. "Camera Calibration Using Spheres: A Semi-definite Programming Approach". In: *Proceedings of IEEE International Conf. on Computer Vision* (2003).

[7] G. Zhang H. Zhang and K.-Y.K. Wong. "Camera Calibration with Spheres: Linear Approaches". In: *Proc. Int'l Conf. Image Processing, vol. II* (2005), pp. 1150–1153.

[8] G. Strang. *Linear Algebra and its Applications*. Fourth Edition. Thomson, Brooks/Cole, Belmont, CA, 2006.

[9] G.Q. Zhang H. Zhang K.K. Wong. "Camera calibration from images of spheres". In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2007), pp. 499–503.

[10] T. Pajdla. *Elements of Geometry for Computer Vision*. FEE CTU, 2013.

[11] Murray Bourne. *Conic Sections - interactive 3-D graph*. URL: https://www.intmath.com/plane-analytic-geometry/conic-sections-summary-interactive.php.

[12] N. Chernov. *Ellipse Fit (Direct method)*. URL: https://www.mathworks.com/matlabcentral/fileexchange/22684-ellipse-fit--direct-method-.

[13] *Nikon Digital SLR Camera D5100 Specifications*. URL: http://imaging.nikon.com/lineup/dslr/d5100/pdf/d5100_16p.pdf.

[14] *Single Camera Calibrator App*. URL: https://www.mathworks.com/help/vision/ug/single-camera-calibrator-app.html.

Appendix A
Contents of the attached CD

- `/`
  - `Bachelor_thesis` ..... folder with main documents
    - `Images` ..... folder contaning images of all experiments
      - `RelativePosition` ..... folder containing photos of the relative position experiment
        - `_DSC0328.jpg` ..... first photo of the relative position of the sphere
        - ⋮ ..... 39 photos from _DSC0328 to _DSC0365 of the relative position of the sphere
        - `_DSC0365.jpg` ..... last photo of the relative position of the sphere
      - `SingleSphere` ..... folder containing photos of the single sphere experiment
        - `OneSphere280.jpg` ..... distance 280 mm
        - `OneSphere420.jpg` ..... distance 420 mm
        - `OneSphere710.jpg` ..... distance 710 mm
      - `TwoSpheres` ..... folder containing photos of the two spheres experiment
        - `TwoSpheres360.jpg` ..... distance 360 mm
        - `TwoSpheres410.jpg` ..... distance 410 mm
    - `MATLABCODE` ..... folder containing MATLAB scripts
      - `DetectEllipseByTomasPajdla` ..... folder containing the scripts written by Tomáš Pajdla, which allow us to detect an ellipse on the images in the relative position experiment
      - `RelativePosition` .....
        - `ComputeCentersConcreteq` ..... script for computing centers of concrete **q**
        - `ComputeCentersForErrorDetermining` script for computing centers to determine an error in ellipse detecting
        - `distu2Without2Meas` ..... script for graph plotting without two bad measurements
        - `distuGraph` ..... script for plotting the graph

- `distuScript` — script for computing the relative position of the sphere
- `EllipseDirectFit` — script used to fit the ellipse from given points
- `ShowSpheres` — script for showing the spheres
- `TestOfAllImages` — script for identifying an ellipse in each image
- `TestOfBadImagesInExperiment` — script for identifying an ellipse in images with big error and in their neighbouring images

`SingleSphere` — folder containing scripts for computing the position of the single sphere

- `dist280` — distance 280 mm
- `dist420` — distance 420 mm
- `dist710` — distance 710 mm

`TwoSpheres` — folder containing scripts for computing the position of two spheres

- `dist360mm` — distance 360 mm
- `dist410mm` — distance 410 mm

`addPath.txt` — text document containing three commands to add two paths and run file for successive using our scripts

`Bachelor_Thesis.pdf` — digital copy of this thesis