CS6320: 3D Computer Vision
Project 1 : Geometric Camera Models and Calibration

Arthur Coste: *coste.arthur@gmail.com*

January 2013

Contents

1 Introduction

The objective of this first project of the semester is to study various aspects related to geometry and cameras. In a first part we are going to review and present some important concepts regarding theoretical questions related to image acquisition using cameras and especially the way we convert a three dimensional world into a two dimensional image. In fact this aspect deals with geometry and is the main basis of computer vision because two dimensional images are the basis input data we have.

In a second part, we will present and implement a very important part of Computer Vision which is camera calibration. Indeed to have good images and study them properly we need to be able to have information regarding the device we use to get the images and it's possible issues. We also need to know the position of the camera to be able to make accurate results that will also depend on the internal parameters of the camera.

Due to the scientific nature of this work we will use the Scientific metric system to estimate all our measurements.

In this report, the implementation is made using MATLAB, the functions we developed are included with this report and their implementation presented in this document.

The following functions are associated with this work :

- $select\_points.m : [x, y] = select\_points(InputImage)$
- $calibration.m : [] = calibration()$ stand alone function

Note : All the images and drawings were realized for this project so there is no Copyright infringement in this work.

2 Theoretical Problems

In this section we are going to present and explain theoretical points regarding the geometry at stake when acquiring images.

2.1 Pin hole camera

As we have seen the geometrical properties of the pin hole camera imply that a point, no matter how far it is from the camera will be projected into the image plane. The following figure is showing this aspect:

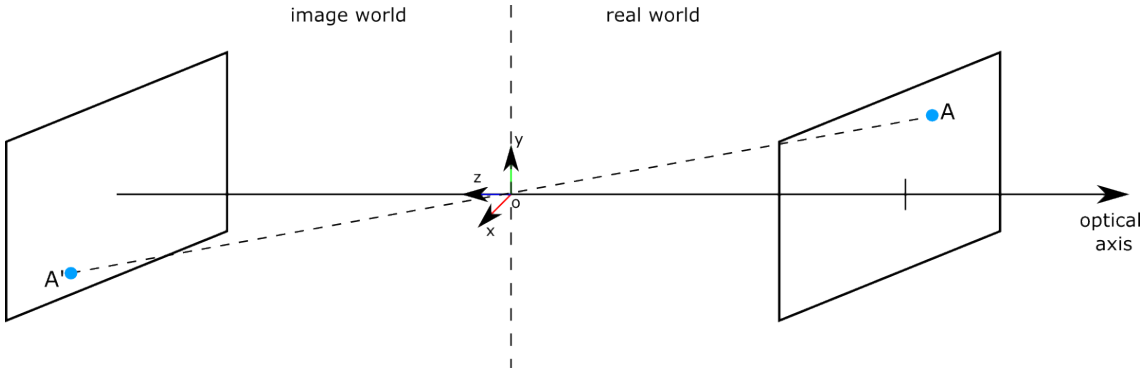

Figure 1: A point in the real world projects into a point

An easy way to answer the question regarding a line would be to say that basically a line is a continuum of points so a line is an infinite number of points. Each point will be projected as a point in the plane image and the geometry will be preserved if we considerate that all the rays coming from the line in the world are strictly linear and all goes exactly through the pin hole and if we don't consider a possible diffraction.

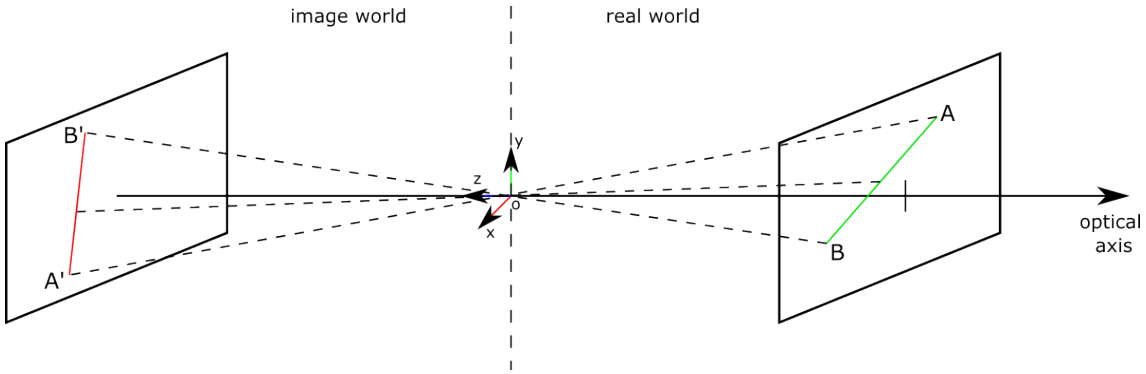

Figure 2: A line in the real world projects into a line

Note that the geometry is preserved but not the orientation which will be flipped, the drawing is not perfectly accurate on that aspect.

Let's now prove the previous statement using a formal presentation and considering three collinear points. In the real world, let's define the following points:

$$P_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \qquad \text{with} \qquad i \in [\![1, 3]\!] \tag{1}$$

If we consider each of this point as a vector, the collinear points are then collinear vector, so it implies that it exists a linear relationship between them.
If we build a matrix of this system and compute its determinant we will obtain 0 because each column of this matrix is a linear combination of the others *i.e* $P_1 = \alpha P_2 + \beta P_3$ and we end up with a single vector and the determinant is 0:

$$det(P_1, P_2, P_3) = \begin{vmatrix} x & \alpha x & \beta x \\ y & \alpha y & \beta y \\ z & \alpha z & \beta z \end{vmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \cdot \begin{vmatrix} 1 & \alpha & \beta \\ 1 & \alpha & \beta \\ 1 & \alpha & \beta \end{vmatrix} = 0 \tag{2}$$

So if we use the perspective projection equation we can determine the position on the plane of our points. The perspective projection is given by:

$$\begin{cases} x' = \frac{x}{z} \\ y' = \frac{y}{z} \\ z' = 1 \end{cases} \tag{3}$$

So we have :

$$P_i^{'} = \begin{pmatrix} \frac{x_i}{z} \\ \frac{y_i}{z} \\ 1 \end{pmatrix} \qquad \text{with} \qquad i \in [\![1, 3]\!] \tag{4}$$

So if we build again the determinant matrix with the projected points we have:

$$det(P_1^{'}, P_2^{'}, P_3^{'}) = \begin{vmatrix} \frac{x}{z} & \frac{\alpha x}{z} & \frac{\beta x}{z} \\ \frac{y}{z} & \frac{\alpha y}{z} & \frac{\beta y}{z} \\ 1 & \alpha & \beta \end{vmatrix} = \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \\ 1 \end{pmatrix} \cdot \begin{vmatrix} 1 & \alpha & \beta \\ 1 & \alpha & \beta \\ 1 & \alpha & \beta \end{vmatrix} = 0 \tag{5}$$

So according to the previous result, using the perspective projection we are showing that if a set of points are originally aligned in the world space their projection on the image plane is also made of aligned points. The link with the previous question is what we mentioned about the nature of a real line as being a continuum of points which are aligned.

2.2 Perspective projection

In this part, we want to study the projection of two parallel lines of the world into the image plane and show that there is a convergence to an horizon plane. We will firstly do it geometrically and

then present the formal solution of to this question.

The book has a very nice illustration to show this phenomenon so I will use it to support my geometrical explanation.

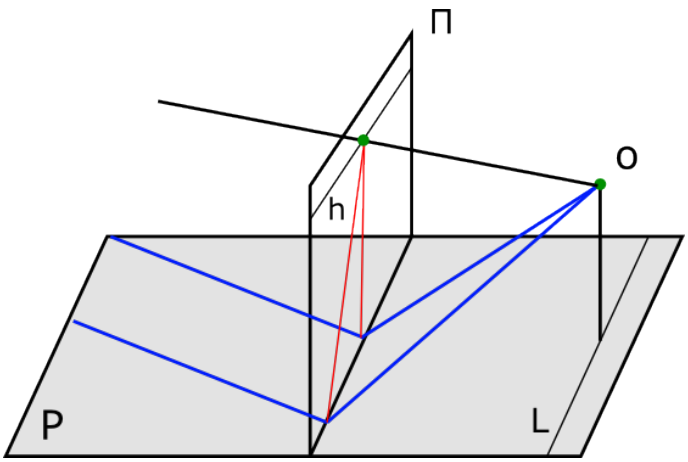

Figure 3: Two parallel lines meeting at infinity

So the geometrical explanation is relying on two things, the first thing is that when we acquire an image of parallel lines for instance we depend on the optical system we use, which could be as simple as a pinhole or more complicated involving lenses. But the point is that we have an optical center where everything converge to. This is the point O in our figure. Then the Π plan is the image plan that have been symmetrically flipped according to point O.
Based on the previous two points, our two parallel lines are going to converge through the optical system to be projected on the image plane Π and because they are going to be viewed from a special viewpoint which gets light which also comes from this optical system, we are going to see the intersection of the two lines.

To conclude this geometrical explanation, we can say that the explanation lies in that two parallel lines do not actually intersect, but only appear to do so on the image plane. In fact, under perspective projection model, any set of parallel lines that are not parallel with the projection plan Π will converge to a vanishing point located on an horizon line.

The theoretical proof relies on a very important point which is as we can see on the previous picture the duality between affine and projective spaces. This aspect is important to consider that the Euclidean property of non crossing parallel lines will not hold in a different space.
Indeed, let us prove this mathematically using the perspective equation to convert an Euclidean space into a projective space.

If we consider the set up of the previous picture, we need to associate a spatial reference such as:

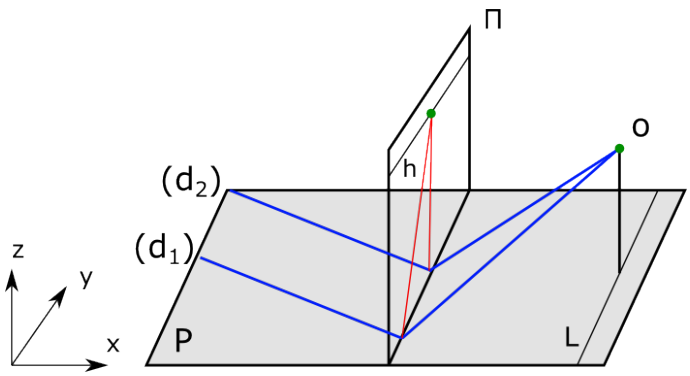

Figure 4: Spatial reference added

If we consider the Euclidean plane $(\mathcal{P})$ of $\mathbb{R}^3$ such as $z = 0$ and the two parallel lines (d_1) and (d_2). In this space, the two lines will verify the property of parallel lines and will never cross.
Let's write the canonical Euclidean equations of those two lines with respect to our spatial basis:

$$\begin{cases} (d_1) : a_1x + b_1y + e_1z = c_1 \\ (d_2) : a_2x + b_2y + e_2z = c_2 \end{cases} \tag{6}$$

As said before if $(\mathcal{P})$ has height 0 it's going to simplify a bit the previous equations but it's not going to modify the following steps so we will keep it to solve the general parallel lines projection problem.
Indeed we are now going to use the perspective equation to project our parallel lines (d_1) and (d_2) and see what happens.

$$\begin{cases} y' = \frac{y}{x}f' \\ z' = \frac{z}{x}f' \end{cases} \tag{7}$$

Our projection plane is defined by it's position on the x axis so we only project on the Π plane the y and z axis.
From the lines equation let's extract the expressions of y and z:

$$(d_1) : a_1x + b_1y + e_1z = c_1 \Rightarrow \begin{cases} y = \frac{c_1 - a_1x - e_1z}{b_1} \\ z = \frac{c_1 - a_1x - b_1y}{e_1} \end{cases} \tag{8}$$

$$(d_2) : a_2x + b_2y + e_2z = c_2 \Rightarrow \begin{cases} y = \frac{c_2 - a_2x - e_2z}{b_2} \\ z = \frac{c_2 - a_2x - b_2y}{e_2} \end{cases} \tag{9}$$

Let's plug those expression to determine the perspective projection of those lines:

$$\begin{cases} y' = \frac{y}{x}f' \\ z' = \frac{z}{x}f' \end{cases} \Rightarrow \begin{cases} y' = \frac{c_1 - a_1x - e_1z}{b_1x}f' \\ z' = \frac{c_1 - a_1x - b_1y}{e_1x}f' \end{cases} \tag{10}$$

In this case if we make x go to infinity it yields to:

$$\begin{cases} y' = \frac{a_1}{b_1} f' \\ z' = \frac{a_1}{e_1} f' \end{cases} \qquad \text{and for the second line} \qquad \begin{cases} y' = \frac{a_2}{b_2} f' \\ z' = \frac{a_2}{e_2} f' \end{cases} \tag{11}$$

If lines are parallel their slope is the same so $a_1 = a_2$ and their y intercept is linearly connected. So we can conclude that they intercept in the same point.

The same kind of demonstration can be done using the homogeneous representation of a line :

$$\begin{cases} \text{line 1 : } P_1 + \alpha \mathbf{l} \\ \text{line 2 : } P_2 + \alpha \mathbf{l} \end{cases} \quad \Rightarrow \quad \begin{cases} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} + \alpha \begin{pmatrix} l_1 \\ l_2 \\ l_3 \end{pmatrix} \\ \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} + \alpha \begin{pmatrix} l_1 \\ l_2 \\ l_3 \end{pmatrix} \end{cases} \tag{12}$$

The lines being parallel imply that $l_1 = l_2$ and if we project them and again take the limit of x going to infinity:

$$\begin{cases} y' = \frac{y_1}{x_1} f' + \frac{l_2}{l_1} \\ z' = \frac{z_1}{x_1} f' + \frac{l_3}{l_1} \end{cases} \qquad \text{if } x \to +\infty \qquad \begin{cases} y' = \frac{l_2}{l_1} \\ z' = \frac{l_3}{l_1} \end{cases} \tag{13}$$

And so by doing the same with the other line, define with the same parameters, we also end with a single intersection point.

Note that we did not use the same notation as the book with the z axis, but the development is the same. It just requires to change the orientation of the coordinate system we chose initially.

2.3 Depth of field

2.3.1 Infinite depth of field

Given the fact that a pinhole camera does not have any optical lenses no focal distances are technically implied opposed as if we were using lenses. It means that each ray that goes through the pinhole and which come from a point wherever it is in the real world will be projected into the image plane. It implies that the depth of field which is the the maximum distance of points that will provide a sharp image does not depend on any lenses so is technically infinite.

So, the point relies on the fact that there is no focus to be done and that everything is supposed to be sharp. There are some physical limits of course but this absence of optical focus provides the infinite depth of focus property.

2.3.2 Lenses and blur circle

Here is the set up of two points, one in the focal object plan which will create a sharp point on the image plane and an other one which will contribute to generate a blur circle.

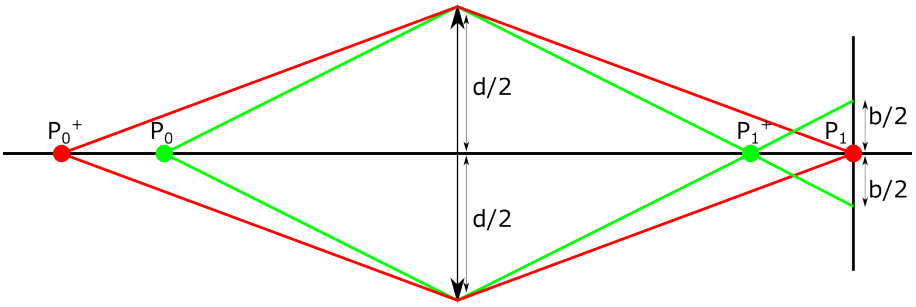

Figure 5: Blur circle with lens set up

Our goal is to determine the value of b which is the diameter of the blur circle in function of the other distance between points.
The mathematical solution rely on similar triangles according to the following set up:

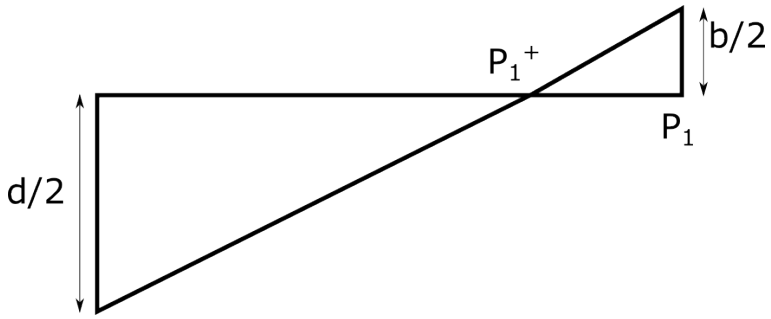

Figure 6: Similar triangles in the lens set up

If we use the assignment notation we can define that the image plane is located at position z we have the following relation:

$$\frac{P_i - P_i^+}{P_i^+} = \frac{\frac{b}{2}}{\frac{d}{2}} \quad \Rightarrow \quad \frac{|z - P_i^+|}{P_i^+} = \frac{b}{d} \quad \Rightarrow \quad b = d\frac{|z - P_i^+|}{P_i^+} \tag{14}$$

Now let's use the thin lens assumption to derive the equation of $\Delta Z_o^+ = Z_o^+ - Z_o$
Let's determine the expression of Z_o^+ using the lens equation:

$$\frac{1}{Z_o^+} + \frac{1}{Z_i^+} = \frac{1}{f} \quad \Rightarrow \quad f' = \frac{Z_o^+ Z_i}{Z_o^+ + Z_i} \quad \Rightarrow \quad Z_o^+ = f\frac{Z_i^+}{Z_i^+ - f} \tag{15}$$

Using the fact that Z_o and Z_i are conjugate points in focal points we have $z = Z_o = Z_i$ let's determine the expression of Z_i^+ to plug it in the previous equation:

$$b = d\frac{|Z_o - P_i^+|}{P_i^+} \qquad \Rightarrow \qquad \frac{b}{d} = 1 - \frac{Z_o}{Z_i^+} \qquad \Rightarrow \qquad Z_i^+ = \frac{dZ_o}{d-b} \tag{16}$$

Which leads to :

$$Z_o^+ = f\frac{Z_i^+}{Z_i^+ - f} \qquad \Rightarrow \qquad Z_o^+ = f\frac{dZ_o}{dZ_o - f(d-b)} \tag{17}$$

So we can determine $\Delta Z_o^+ = Z_o^+ - Z_o$:

$$\Delta Z_o^+ = Z_o^+ - Z_o \qquad \Rightarrow \qquad \Delta Z_o = f\frac{dZ_o}{dZ_o - f(d-b)} - Z_o \qquad \Rightarrow \qquad \Delta Z_o^+ = \frac{fbZ_o - dZ_o^2}{dZ_o + f(d-b)} \tag{18}$$

We can rewrite this in a simpler way:

$$\Delta Z_o^+ = \frac{fbZ_o - dZ_o^2}{dZ_o + f(d-b)} \qquad \Rightarrow \qquad \Delta Z_o^+ = \frac{Z_o\left(f\frac{b}{d} - Z_o\right)}{f(\frac{d}{b} - 1) + Z_o} \tag{19}$$

We can now say that if the diameter d increase the depth of field decrease, to prove that let's consider the limit of the previous equation :

$$\lim_{d\to+\infty} \Delta Z_o^+ = \frac{\lim\limits_{d\to+\infty} Z_o\left(f\frac{b}{d} - Z_o\right)}{\lim\limits_{d\to+\infty}\left(f(\frac{d}{b} - 1) + Z_o\right)} = \frac{-Z_o^2}{+\infty} = 0 \tag{20}$$

The previous equation uses an improper notation but shows that if d increase, the depth of field decrease. Let's now do the same analysis with regard to Z_o. If we use the dominant exponent rule and negligible operations we obtain the following :

$$\Delta Z_o^+ \approx \frac{Z_o^2}{Z_o} \approx Z_o \qquad \Rightarrow \qquad \text{if} \qquad Z_o \nearrow \qquad , \qquad \Delta Z_o^+ \nearrow \tag{21}$$

So if Z_o increase it also increase the depth of field.

2.4 3D Rotation

Rotation matrices play a very important role in computer vision and particularly in this project to help us calibrate the spatial orientation of the camera. Rotation matrices is an orthogonal matrix with a determinant equal to 1. The set of all matrices of a given dimension is a group called the Special Orthogonal Group of dimension n : $\mathcal{SO}_n$.

2.4.1 Commutativity in general case

Due to this last aspect, rotation matrices have very nice properties such as associativity for instance but they don't have the commutativity property except for $\mathcal{SO}_2$. In this first part, we are going to

show this property using both theoretical definitions and practical examples.
Let's consider the rotation of angle θ around x.

$$\mathbf{R_x}(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{pmatrix} \tag{22}$$

Let's consider the rotation of angle ϕ around y.

$$\mathbf{R_y}(\phi) = \begin{pmatrix} cos(\phi) & 0 & sin(\phi) \\ 0 & 1 & 0 \\ -sin(\phi) & 0 & cos(\phi) \end{pmatrix} \tag{23}$$

Let's consider the rotation of angle ψ around z.

$$\mathbf{R_z}(\psi) = \begin{pmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{24}$$

Now, let's try to combine them to show that it's not commutative. It's basically due to the fact that matrix product is not commutative.
Let's compare $\mathbf{R_x}(\theta)\mathbf{R_y}(\phi)\mathbf{R_z}(\psi)$ with $\mathbf{R_z}(\psi)\mathbf{R_y}(\phi)\mathbf{R_x}(\theta)$

$$\mathbf{R_x}(\theta)\mathbf{R_y}(\phi)\mathbf{R_z}(\psi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{pmatrix} \begin{pmatrix} cos(\phi) & 0 & sin(\phi) \\ 0 & 1 & 0 \\ -sin(\phi) & 0 & cos(\phi) \end{pmatrix} \begin{pmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{25}$$

$$\mathbf{R_x}(\theta)\mathbf{R_y}(\phi)\mathbf{R_z}(\psi) = \begin{pmatrix} cos(\phi) & 0 & sin(\phi) \\ sin(\theta)sin(\phi) & cos(\theta) & -sin(\theta)cos(\phi) \\ -cos(\theta)sin(\phi) & sin(\theta) & cos(\theta)cos(\phi) \end{pmatrix} \begin{pmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{26}$$

$$= \begin{pmatrix} cos(\phi)cos(\psi) & -cos(\phi)sin(\psi) & sin(\phi) \\ sin(\theta)sin(\phi)cos(\psi) + cos(\theta)sin(\psi) & -sin(\theta)sin(\phi)sin(\psi) + cos(\theta)cos(\psi) & -sin(\theta)cos(\phi) \\ -cos(\theta)sin(\phi)cos(\psi) + sin(\theta)sin(\psi) & cos(\theta)sin(\phi)sin(\psi) + sin(\theta)cos(\psi) & cos(\theta)cos(\phi) \end{pmatrix} \tag{27}$$

$$\mathbf{R_z}(\psi)\mathbf{R_y}(\phi)\mathbf{R_x}(\theta) = \begin{pmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} cos(\phi) & 0 & sin(\phi) \\ 0 & 1 & 0 \\ -sin(\phi) & 0 & cos(\phi) \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{pmatrix} \tag{28}$$

$$\mathbf{R_z}(\psi)\mathbf{R_y}(\phi)\mathbf{R_x}(\theta) = \begin{pmatrix} cos(\psi)cos(\phi) & -sin(\psi) & sin(\phi)cos(\psi) \\ cos(\phi)sin(\phi) & cos(\phi) & sin(\phi)sin(\psi) \\ -sin(\phi) & 0 & cos(\phi) \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{pmatrix} \tag{29}$$

$$= \begin{pmatrix} cos(\psi)cos(\phi) & -cos(\theta)sin(\psi) + sin(\theta)sin(\phi)cos(\psi) & sin(\phi)sin(\psi) + cos(\theta)sin(\phi)cos(\psi) \\ cos(\phi)sin(\psi) & cos(\theta)cos(\psi) + sin(\theta)sin(\phi)sin(\psi) & -sin(\theta)cos(\psi) + cos(\theta)sin(\phi)sin(\psi) \\ -sin(\phi) & sin(\theta)cos(\phi) & cos(\theta)cos(\phi) \end{pmatrix} \tag{30}$$

If we compare the analytical results of the rotation we performed, we can see some similarity but they are not the same so it implies that the order to perform a rotation in a 3 dimensional space is dependent of the order. Let's illustrate this aspect with a picture to make things more clear:

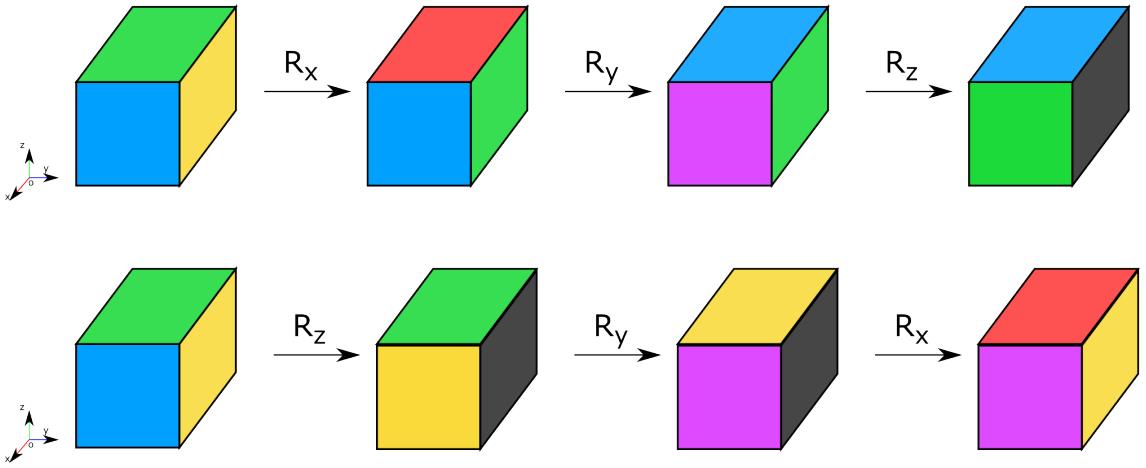

Figure 7: Illustration of the two rotations

As we can see, if we start with the same initial situation and apply the three dimension rotation matrices in a different order the result obtained is different.

2.4.2 Small angles

Let's considerate now that this 3D rotation matrices are parametrized with very small angles that models some small issues. In this case we have the following properties:

$$\lim_{\theta \to 0} cos(\theta) = 1 - d\theta \approx 1 \qquad \text{and} \qquad \lim_{\theta \to 0} sin(\theta) = d\theta \tag{31}$$

The two previous properties allow us to rewrite and simplify the rotation matrices we presented before.

$$\mathbf{R_x}(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -d\theta \\ 0 & d\theta & 1 \end{pmatrix} \text{ and } \mathbf{R_y}(\phi) = \begin{pmatrix} 1 & 0 & d\phi \\ 0 & 1 & 0 \\ -d\phi & 0 & 1 \end{pmatrix} \text{ and } \mathbf{R_z}(\psi) = \begin{pmatrix} 1 & -d\psi & 0 \\ d\psi & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{32}$$

Where each one of them can be decomposed according to:

$$\mathbf{dR_x} = I_3 + Ad\theta \text{ and } \mathbf{dR_y} = I_3 + Bd\phi \text{ and } \mathbf{dR_z} = I_3 + Cd\psi \tag{33}$$

If we compute again the rotation we presented before but using the previous matrices we have:

$$\mathbf{dR_x}(\theta)\mathbf{dR_y}(\phi)\mathbf{dR_z}(\psi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -d\theta \\ 0 & d\theta & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & d\phi \\ 0 & 1 & 0 \\ -d\phi & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & -d\psi & 0 \\ d\psi & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{34}$$

$$\mathbf{dR_x}(\theta)\mathbf{dR_y}(\phi)\mathbf{dR_z}(\psi) = \begin{pmatrix} 1 & 0 & d\phi \\ d\theta d\phi & 1 & -d\theta \\ -d\phi & d\theta & 1 \end{pmatrix} \begin{pmatrix} 1 & -d\psi & 0 \\ d\psi & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{35}$$

$$\mathbf{dR_x}(\theta)\mathbf{dR_y}(\phi)\mathbf{dR_z}(\psi) = \begin{pmatrix} 1 & -d\psi & d\phi \\ d\theta d\phi + d\psi & -d\theta d\phi d\psi + 1 & -d\theta \\ -d\phi + d\theta d\psi & d\phi d\psi + d\theta & 1 \end{pmatrix} \tag{36}$$

$$\mathbf{dR_z}(\theta)\mathbf{dR_y}(\phi)\mathbf{dR_x}(\psi) = \begin{pmatrix} 1 & -d\psi & 0 \\ d\psi & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & d\phi \\ 0 & 1 & 0 \\ -d\phi & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -d\theta \\ 0 & d\theta & 1 \end{pmatrix} \tag{37}$$

$$\mathbf{dR_z}(\theta)\mathbf{dR_y}(\phi)\mathbf{dR_x}(\psi) = \begin{pmatrix} 1 & -d\psi & d\phi \\ d\psi & 1 & -d\phi d\psi \\ -d\phi & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -d\theta \\ 0 & d\theta & 1 \end{pmatrix} \tag{38}$$

$$\mathbf{dR_z}(\theta)\mathbf{dR_y}(\phi)\mathbf{dR_x}(\psi) = \begin{pmatrix} 1 & -d\psi + d\theta d\phi & d\theta d\psi + d\phi \\ d\psi & 1 + d\theta d\phi d\psi & -d\theta + d\phi d\psi \\ -d\phi & d\theta & 1 \end{pmatrix} \tag{39}$$

We kind of end up again with something similar as before where the two result matrices are different. But here, our condition of infinitesimal angles is going to provide us simplifications. Indeed our infinitesimal angles are really small let's call : $d\theta$, $d\phi$ and $d\psi$ the first order infinitesimal angles. When we multiply them with each other as we can find in the two previous matrices we create second and third order infinitesimal angles which are even smaller. So we are going to invoke the following:

$$d\theta d\phi \approx 0 \text{ and } d\phi d\psi \approx 0 \text{ and } d\theta d\psi \approx 0 \text{ and } d\theta d\phi d\psi \approx 0 \tag{40}$$

So we can rewrite the two previous results and simplify:

$$\mathbf{dR_x}(\theta)\mathbf{dR_y}(\phi)\mathbf{dR_z}(\psi) = \begin{pmatrix} 1 & -d\psi & d\phi \\ d\theta d\phi + d\psi & -d\theta d\phi d\psi + 1 & -d\theta \\ -d\phi + d\theta d\psi & d\phi d\psi + d\theta & 1 \end{pmatrix} = \begin{pmatrix} 1 & -d\psi & d\phi \\ d\psi & 1 & -d\theta \\ -d\phi & d\theta & 1 \end{pmatrix} \tag{41}$$

$$\mathbf{dR_z}(\theta)\mathbf{dR_y}(\phi)\mathbf{dR_x}(\psi) = \begin{pmatrix} 1 & -d\psi + d\theta d\phi & d\theta d\psi + d\phi \\ d\psi & 1 + d\theta d\phi d\psi & -d\theta + d\phi d\psi \\ -d\phi & d\theta & 1 \end{pmatrix} = \begin{pmatrix} 1 & -d\psi & d\phi \\ d\psi & 1 & -d\theta \\ -d\phi & d\theta & 1 \end{pmatrix} \tag{42}$$

And we can conclude that in the case of infinitesimal rotation angles we do have the commutative property and that order does not matter in regard to a consideration on what is being a small angle of rotation.

3 Practical Problem

In this practical problem, we are going to perform a camera calibration. This step consists in modelling the process used in cameras to convert a three dimensional real scene into a two dimensional image. This technique is aiming at determining the parameters that are mathematically involved to convert the coordinates between the two spaces.

In a first part we are going to present theoretically the whole mathematical framework supporting this operation. Then we will present the experiment we made and the camera we used and finally we will present our implementation to determine these parameters and the result we obtained.

3.1 Theoretical presentation

In this framework we are going to present the whole mathematical pipeline to perform camera calibration and the parameters that need to be determined.

The first thing to introduce here is the camera model we are going to use. In this framework we are going to use a standard pinhole camera model, this model is valid only if the optical system of the camera can be considered as respecting Gauss conditions which are the conditions of the paraxial approximation involving the linear model of ray lights and small angles. This last aspect is linked to the small angles in the rotation example.

The whole point of camera calibration is then to find the relationship between the point A in the real world with physical coordinates and its associated projection in the image A'. The issue here is that we change the dimension of the space between A and A' so we need to take this aspect into account.

$$A = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \qquad \text{and} \qquad A' = \begin{pmatrix} u \\ v \end{pmatrix} \tag{43}$$

The following picture illustrate this conversion:

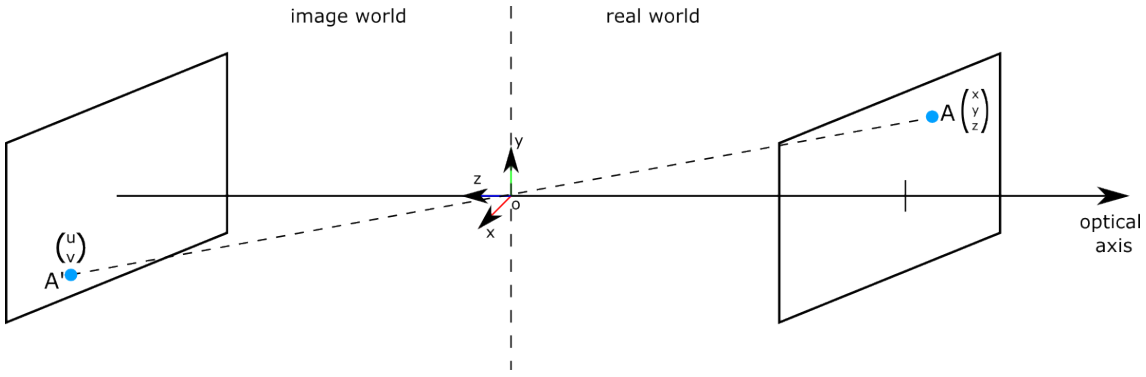

Figure 8: Illustration of projection of A in real space to A' on image plane

On the previous picture we can clearly see the different coordinate references at stake here, one associated to the 3D object space, one associated to the camera and the last one associated to the image plane. There is a relationship between them and this is what we are going to present.

3.1.1 Homogeneous coordinates

In this framework we are going to use the homogeneous coordinates due to the projection at stake here. We remind that the homogeneous coordinates are defined by adding one component to the coordinated of a point or a vector. For instance, if we write the two previous points A and A' we have the following :

$$A = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \qquad \text{and} \qquad A' = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \tag{44}$$

3.1.2 Intrinsic parameters

Intrinsic parameters are the parameters that depend on the camera and the way it has been manufactured. Modelling intrinsic parameters will allow us to establish a relationship between the coordinates on the image plane and the coordinate system associated to the camera.

There are 5 intrinsic parameters which model the following aspects : the definition of the pixel size, the location of the optical center on the sensor array and the manufacturing quality of this sensor.

Firstly lets talk about the pixel definition. Indeed, the pixel is the fundamental element that compose all digital images but which also exists and is defined on the sensor used in the camera. They are rectangular surface elements characterize by the parameters k and l defined as follow :

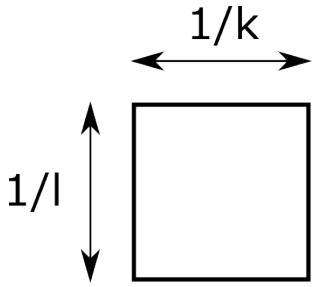

Figure 9: k and l parameters of a pixel element

We can introduce those two parameters in our perspective projection equation which becomes :

$$\begin{cases} x' = u = kf\frac{x}{z} \\ y' = v = lf\frac{y}{z} \\ z' = 1 \end{cases} \tag{45}$$

Now we can define the two parameters we are going to use in our calibration framework:

$$\begin{cases} \alpha = kf \\ \beta = lf \end{cases} \tag{46}$$

The second aspect modelled by the intrinsic set of parameters is the location of the optical center on the array because it might not lay exactly in the center of the array depending on the quality of the sensor and the way it has been manufactured and built into the camera. So there could be a small shift of its location regarding where we expect it to be.

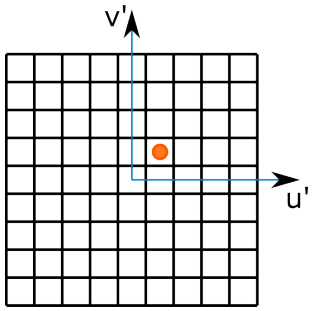

Figure 10: Location of the optical center

This location of the optical center will be parametrised using (u_0, v_0) in the modified perspective equation presented before:

$$\begin{cases} x' = u = \alpha \frac{x}{z} + u_0 \\ y' = v = \beta \frac{y}{z} + v_0 \\ z' = 1 \end{cases} \tag{47}$$

Finally, it takes in account then it takes in account a possible distortion of the CCD or CMOS array used to acquire images. In fact there could be a small tilt between what we think is the coordinate frame and what it physically is on the device. The following picture is explaining this aspect.

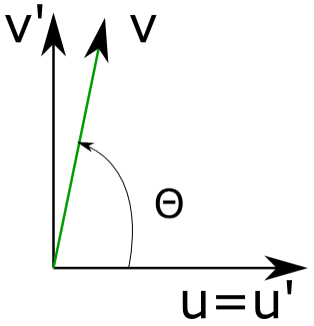

Figure 11: Ideal (u',v') frame and possible tilt of the physical (u,v) frame

We are going to model this tilt using an angular parameter called θ. In the ideal case this value should be equal to 90° and we can here estimate using this parameter the quality of the sensor used in our camera.

Applying general trigonometry provide us the relationship between the two frames according to the angular parameter θ.

$$\begin{cases} u' = u - v'cos(\theta) \\ v = v'sin(\theta) \end{cases} \quad \Rightarrow \quad \begin{cases} u' = u - v'cos(\theta) \\ v' = \frac{v}{sin(\theta)} \end{cases} \tag{48}$$

$$\begin{cases} u' = u - v\frac{sin(\theta)}{cos(\theta)} \\ v = v'sin(\theta) \end{cases} \quad \Rightarrow \quad \begin{cases} u' = u - vcot(\theta) \\ v' = \frac{v}{sin(\theta)} \end{cases} \tag{49}$$

So we can use the previous result to plug it into the modified perspective projection equation and create an intrinsic matrix:

$$\mathbf{M_I} = \frac{1}{z} \begin{pmatrix} \alpha & -\alpha cot(\theta) & u_0 & 0 \\ 0 & \frac{\beta}{sin(\theta)} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{50}$$

So we finally end with the Intrinsic parameters matrix to solve for the following set of unknown : $(\alpha, \beta, \theta, u_0, v_0)$

3.1.3 Extrinsic parameters

Extrinsic parameters are dependent on the physical position and orientation of the camera in the coordinate reference frame.
In fact compared to the reference coordinate basis, the position of the object changes each time we take a picture. So to be able to make meaningful measurement and analysis based on images we need to know exactly where the object is located regarding to the camera reference. Mathematically the position of the object can be defined with a translation and we can also add a possible rotation to model the object orientation which is not necessary the same as the camera.
This will provide us a way to convert the object space coordinate in the camera basis.

To stay consistent with the previous notation we chose we are also going to use the homogeneous coordinates to build this extrinsic matrix.

$$\mathbf{M_E} = \left(\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \\ \hline 0 & 0 & 0 & 1 \end{array} \right) \tag{51}$$

Where $(T_x, T_y, T_z)^T$ vector is the translation vector and r_{ij} with $(i, j) \in [\![1, 3]\!]^2$ are the components of the 3D rotation matrix presented before.
As we can see here, there are technically twelve unknowns but due to the nature of the rotation matrix, we can reduce it to a set of three angles : (θ, ϕ, ψ).

3.1.4 Final formulation

If we gather what we presented before, we can obtain the full mathematical framework to express the relationship between the world coordinate and there projection on the image plane.

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \frac{1}{z} \begin{pmatrix} \alpha & -\alpha cot(\theta) & u_0 & 0 \\ 0 & \frac{\beta}{sin(\theta)} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \left(\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{52}$$

So we have a set of 11 unknown parameters that are used to calibrate our camera. But this notation using several matrices is not going to be easy to handle in our implementation using MATLAB this is why we are going to gather them into a single matrix that will contain all the parameters to estimate.

$$M_{EI} = \begin{pmatrix} \alpha r_1^T - \alpha cot(\theta) r_2^T + u_0 r_3^T & \alpha T_x - \alpha cot(\theta) T_y + u_0 T_z \\ \frac{\beta}{sin(\theta)} r_2^T + v_0 r_3^T & \frac{\beta}{sin(\theta)} T_y + v_0 T_z \\ r_3^T & T_z \end{pmatrix} \tag{53}$$

3.2 Practical set up

To perform the calibration of our camera, we need to use special patterns and a special set up to be able to make meaningful measurements.
The pattern we are going to use is a checker board that we are going to stick on a corner of a wall to use the difference of depth in our measurement. Here is a picture of the set up taken with a Panasonic Lumix camera:

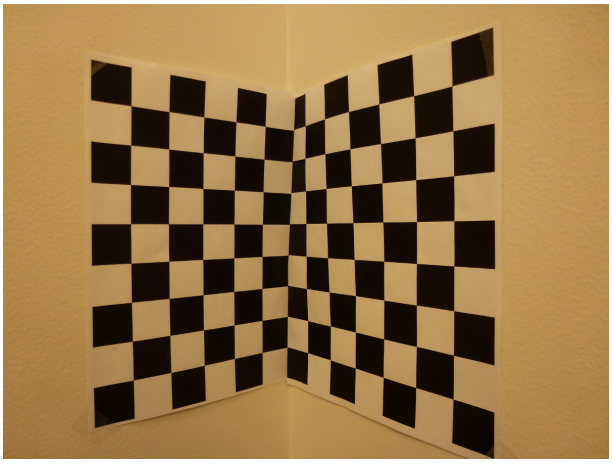

Figure 12: Checker board in a corner of a room

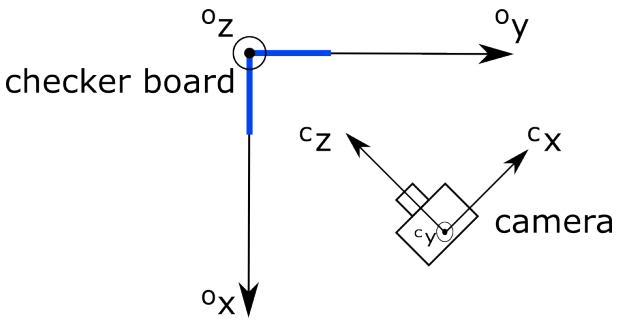

Figure 13: Set up to get the image

On the previous image we can also see the spatial basis we chose to use in the real world space. The bottom of the checker board is centred on the x and y and has a height of 30 cm. We made some approximative measures of the position of our camera in the framework we presented. The camera has this approximative position (40,30,45) in the object coordinate frame and thanks to calibration we will obtain it in the camera frame.

3.3 Processing

We selected a set of points into the image and defined their coordinates in the space according to the coordinate frame we set up before. Here is a picture of the location of our set of 34 points.

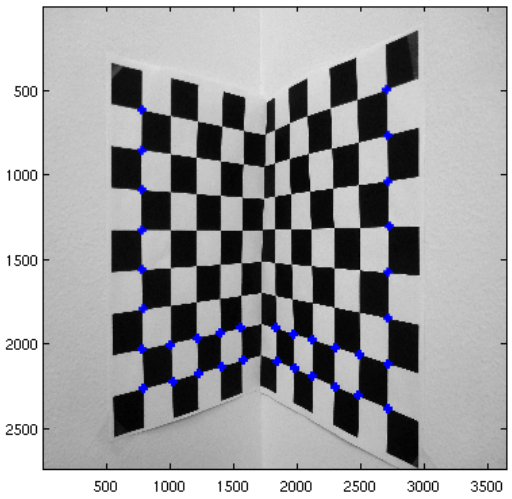

Figure 14: Selected set of points used for calibration

We then set up the matrix problem to solve to estimate our parameters:

$$\mathcal{M} = \begin{pmatrix} P1_x & P1_y & P1_z & 1 & 0 & 0 & 0 & 0 & -u_1P1_x & -u_1P1_y & -u_1P1_z & -u_1 \\ 0 & 0 & 0 & 0 & P1_x & P1_y & P1_z & 1 & -v_1P1_x & -v_1P1_y & -v_1P1_z & -v_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ Pn_x & Pn_y & Pn_z & 1 & 0 & 0 & 0 & 0 & -u_nPn_x & -u_nPn_y & -u_nPn_z & -u_n \\ 0 & 0 & 0 & 0 & Pn_x & Pn_y & Pn_z & 1 & -v_nPn_x & -v_nPn_y & -v_nPn_z & -v_n \end{pmatrix} \tag{54}$$

We want to solve the following system $\mathcal{M}X = 0$ where X is a vector containing the twelve unknowns of the matrix M_{EI} presented before. To do so we are going to rely on the Singular Value Decomposition of this matrix to solve the problem.

The Singular Value Decomposition is a technique used to find diagonal matrices associated to rectangular matrices. In fact, the spectral theorem states that a square matrix can be diagonalized into an eigenvector basis, the Singular value decomposition generalize this to non square matrices such as the matrix $\mathcal{M}$ we presented before. So it exists a decomposition so we can write $\mathcal{M}$ as follow:

$$\mathcal{M} = U\Sigma V^* \tag{55}$$

3.4 Results

Here are the results we get from the mathematical resolution presented before.

3.4.1 Intrinsic parameters

We remind that we try to estimate a set of 5 intrinsic parameters which are associated to the physical characteristics of the camera and mostly its CCD array.

$$\begin{cases} \theta = 89.919^\circ \\ \alpha = 2.7482 \times 10^3 \\ \beta = 2.7325 \times 10^3 \\ u_0 = 1.7538 \times 10^3 \\ v_0 = 1.3811 \times 10^3 \end{cases} \tag{56}$$

Let's analyse them. Firstly we can say that the θ angle is pretty good, it should theoretically be equal to 90° and the result we get is only off by 0.08 which seems to be very low, so it seems that our camera has a pretty good manufactured CCD sensor.
Then the α and β values are pretty close from each other which is interesting. Indeed, they are defined as being the k and l pixel densities scaled by the focal length, so having similar values indicate that the k and l parameters are similar so our pixels are close to be square.
Finally let's talk about the location of the optical center in the array defined by the position (u_0, v_0) in the image frame. The values we obtained have to be compared to the center of our image. The dimension of the image we acquired are : 3648×2736 so the assumption is that the optical center should lie on the middle of the image. So let's compare our results to the middle of the image.

$$\begin{cases} u_0 - \frac{3648}{2} = 1754 - 1824 = -70 \; pixels \\ v_0 - \frac{2736}{2} = 1381 - 1368 = 13 \; pixels \end{cases} \tag{57}$$

So our optical center does not lie exactly in the center of the image, there is a small shift in the y direction of few pixels and a bigger one on the x direction. We will maybe try to see if this results remain the same with further experiments.

3.4.2 Extrinsic parameters

Now let's look at the extrinsic parameters which depend on the position of the camera in the real world and which will change each time we take a different picture and move the camera.

$$\begin{cases} t_x = -0.4251 \\ t_y = 40.8839 \\ t_z = 44.8017 \end{cases} \tag{58}$$

The values we obtain are close to what we roughly measured so they are totally meaningful. In fact, they express the position of the object (here our checker board) with regard to the camera frame as we presented them in a previous figure. In fact the y and z values are close to the raw measurement we made. Here is now the determination of rotation vectors :

$$\begin{cases} r_1 = [-0.7748, 0.6322, -0.0070] \\ r_2 = 1.0e-05 * [-0.0062, -0.0020, 0.5048] \\ r_3 = 1.0e-05 * [0.3191, 0.3911, 0.0055] \end{cases} \tag{59}$$

According to the way we set up the matrix, we can estimate the rotation angles. Due to the previous issue with the coordinate reference orientation I doubt that my result for rotation angle is good, in fact it might suffer from this issue. We use in this section the same notation as the one used in the theoretical section about 3D rotation matrices.

$$\begin{cases} \theta = \frac{arcsin(r_3(2))}{cos(\phi)} \\ \phi = arcsin\left(-r_3(1)\right) \\ \psi = arctan\left(\frac{r_2(1)}{r_1(1)}\right) \end{cases} \tag{60}$$

$$\begin{cases} \theta = -89.3786^\circ \\ \phi = 39.2146^\circ \\ \psi = -89.9591^\circ \end{cases} \tag{61}$$

Based on the fact that I was holding the camera, the angle obtained make some sense so it seems that the calibration might not be to much wrong because the angles if they are correctly estimated seems to traduce the difference of orientation between the object and camera frame.

3.5 Other experiments

3.5.1 Minimum number of points

According to the mathematical equation we presented, we can say that the minimum number of points required to estimate the parameter is 6. In fact each point will provide two equations so we will end up with 12 equations which will allow us to estimate our parameters with a minimal set of equations. This will allow us to compare with the results we previously found.
Here is a picture of our set up :

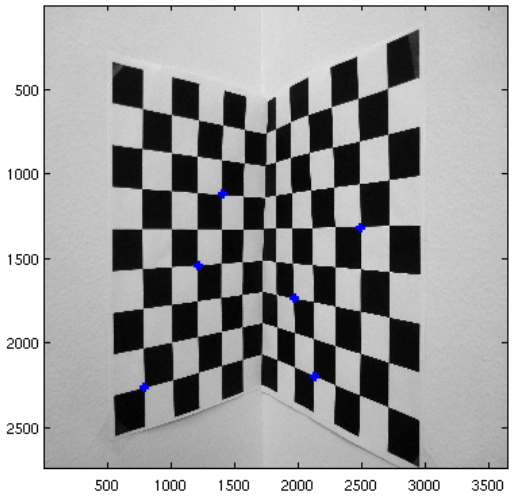

Figure 15: Selected set of 6 points used for calibration test

Here are the results after we used this set up and the calibration program we used.
Intrinsic parameters:

$$\begin{cases} \theta = 114.6460^\circ \\ \alpha = 2.7943 \times 10^3 \\ \beta = 2.1732 \times 10^3 \\ u_0 = 3.9134 \times 10^3 \\ v_0 = -827.4368 \end{cases} \tag{62}$$

Extrinsic parameters:

$$\begin{cases} t_x = 59.2857 \\ t_y = -81.7169 \\ t_z = -15.5060 \\ r_1 = [0.9402, -0.3401, 0.0164] \\ r_2 = 1.0e-05 * [0.1445, 0.3775, -0.4521] \\ r_3 = 1.0e-05 * [0.1476, 0.4274, 0.4041] \end{cases} \tag{63}$$

Which provide the following angles :

$$\begin{cases} \theta = 48.2087^\circ \\ \phi = 19.8867^\circ \\ \psi = -89.9794^\circ \end{cases} \tag{64}$$

As we can see, our estimation of some parameters is wrong but some other make sense. In fact, except our v_0 parameter which is completely off firstly due to its sign and secondly because the value is low. The estimated parameters for α, β and u_0 remains with the same kind of values as before. Regarding the extrinsic parameters all of them are wrong with respect to the previous measurement we made and the physical measurement.

3.5.2 Points lying on a line

If points lie on a line of the image plane, this will create a set of collinear point within the image which will affect the way we solve our equation system.
Here is our set up for this experiment:

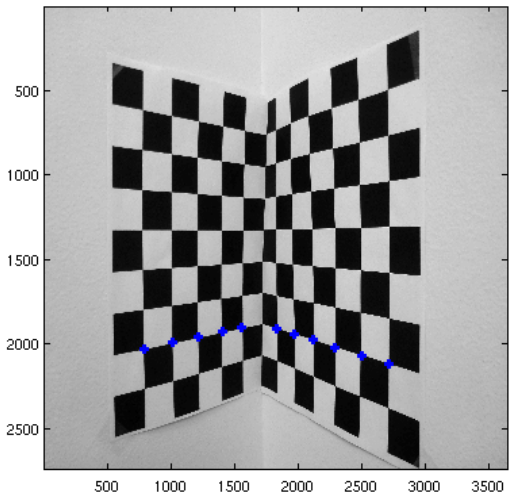

Figure 16: Selected set of points lying on a line and used for calibration test

Here are the results after we used this set up and the calibration program we used.
Intrinsic parameters:

$$\begin{cases} \theta = 179.1628^\circ \\ \alpha = 1.2159 \times 10^{-9} \\ \beta = 2.4582 \times 10^{-7} \\ u_0 = 3.2824 \times 10^3 \\ v_0 = 7.5087 \times 10^5 \end{cases} \tag{65}$$

Extrinsic parameters:

$$\begin{cases} t_x = 7.1120 \times 10^3 \\ t_y = -0.0940 \times 10^3 \\ t_z = 0.0360 \times 10^3 \\ r_1 = [0.9814, -0.1920, 0.0000] \\ r_2 = 1.0e-07 * [0.0710, 0.3629, 0.0000] \\ r_3 = 1.0e-07 * [0.0000, 0.0000, 0.3698] \end{cases} \tag{66}$$

Which provide the following angles :

$$\begin{cases} \theta = 0^\circ \\ \phi = 11.0695^\circ \\ \psi = 0^\circ \end{cases} \tag{67}$$

3.5.3 Pin hole image

On my camera there is a mode to take a picture in a pin hole mode so we tried the same experiment with our 34 points. This pin hole mode simulates a pin hole camera.

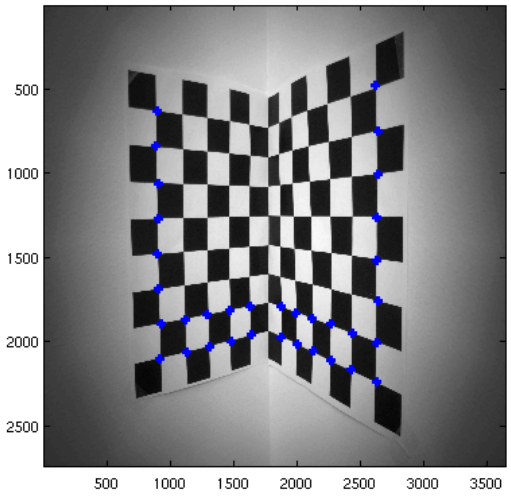

Figure 17: Selected set of points on a pin hole image and used for calibration test

Here are the results after we used this set up and the calibration program we used.
Intrinsic parameters:

$$\begin{cases} \theta = 90.0186^\circ \\ \alpha = 2.6230 \times 10^3 \\ \beta = 2.6136 \times 10^3 \\ u_0 = 1.7994 \times 10^3 \\ v_0 = 1.3584 \times 10^3 \end{cases} \tag{68}$$

The values we obtain here are pretty similar to what we obtained in the regular camera mode to acquire our image the first time. The values of α and β are similar which again proves that our pixel array is close to be made of square elements. And our u_0 and v_0 values are close to before. To compare the position of the center we are again going to calculate the difference with what we theoretically expect:

$$\begin{cases} u_0 - \frac{3648}{2} = 1799 - 1824 = -25 \ pixels \\ v_0 - \frac{2736}{2} = 1358 - 1368 = -10 \ pixels \end{cases} \tag{69}$$

So here the result seems to be closer to what we expected. The reason could be due to the optical system which in the first case might be using the lens system more than with this mode. Extrinsic

parameters:

$$\begin{cases} t_x = -0.1070 \\ t_y = 38.2084 \\ t_z = 47.8641 \\ r_1 = [-0.8131, 0.5821, -0.0101] \\ r_2 = 1.0e-05 * [-0.0175, -0.0156, 0.5058] \\ r_3 = 1.0e-05 * [0.2942, 0.4114, 0.0229] \end{cases} \tag{70}$$

Which provide the following angles :

$$\begin{cases} \theta = -87.4077^\circ \\ \phi = 35.5990^\circ \\ \psi = 89.9859^\circ \end{cases} \tag{71}$$

Regarding the analysis of the extrinsic parameters they are close to our estimation and with what we found before so they seem to be accurate and consistent. Again they traduce the difference between the two orientations we had for our spatial coordinates frames associated to the object and the camera.

Regarding the other experiment we made in the previous parts, we showed the mathematical limits of the method with the number and location of points. So in the first experiment we developed was using a large number of points widely spread on the grid to ensure the best possible least square estimation (LSE). The last experiment using a pinhole mode in the camera allow us to have almost the same estimation of parameters. So the difference between this two modes remains small for this kind of analysis.

3.6 Implementation

3.6.1 Select Points on Image

To be able to estimate the parameters we need to be able to get the location of a set of landmarks so we implemented a function to get the coordinates of each selected points.

```
but = 1;
while but == 1
   clf
   imagesc(I);
   colormap(gray)
   hold on
   plot(x, y, 'b+','linewidth',3);
   axis square

   [s, t, but] = ginput(1);

x = [x;s];
y = [y;t];

end
```

3.6.2 Determination of calibration parameters

Here is our code to implement the calibration to estimate the parameters.

```
%read image
I=imread('P1020660.JPG');
I2=double(I(:,:,1));

imagesc(I2)

%get points
[x,y] = select_points(I2)
P = [\textit{associated 3D coordinates of points}]

% build matrix
for i = 1:length(x)-1
     L1=[P(i,1),P(i,2),P(i,3),1,0,0,0,0, -x(i)*P(i,1),-x(i)*P(i,2),-x(i)*P(i,3),-x(i)]
     L2= [0,0,0,0,P(i,1),P(i,2),P(i,3),1,-y(i)*P(i,1),-y(i)*P(i,2),-y(i)*P(i,3),-y(i)]
     L=[L;L1;L2]
end

%perform SVD and determine parameters
[U,S,V]=svd(L)
```

```
x = V(:,end)
a1=[x(1);x(2);x(3)]
a2=[x(5);x(6);x(7)]
a3=[x(9);x(10);x(11)]

rho = -1 / norm(a3)
r3 = rho*a3
u_0 = rho^2*(a1'*a3)
v_0 = rho^2*(a2'*a3)

theta = acosd(-(cross(a1,a3)'*cross(a2,a3))/(norm(cross(a1,a3))'*norm(cross(a2,a3))))
alpha = rho^2 * norm(cross(a1,a3))*sind(theta)
beta = rho^2 * norm(cross(a2,a3))*sind(theta)

r1= (cross(a2,a3)/norm(cross(a2,a3)))
r3=a3
r2=cross(r3,r1)

% building intrinsic parameter matrix
k = [alpha, -alpha*(cosd(theta)/sind(theta)),u_0;
       0,beta/sin(theta), v_0;
       0,0,1];
t = rho*inv(k)*[x(4);x(8);x(12)]
```

4 Conclusion

In this first project, we had the opportunity to work and study some important aspects regarding the basis of computer vision which is the camera used to acquire images.

In fact we firstly studied the theoretical model and technique of perspective projection which is the fundamental principle at stake. We saw that lines where projected into lines on the image plane, and that a scaling factor was at stake. We also discussed the 3D implications which could appear different as we expect them to be such as non commutativity of rotations in space and the intersection of parallel lines in the projective space.

Then we studied the camera calibration which allow us to consider some issues due to the hardware we used to acquire images. In fact to be able to create an accurate representation of the world using sensors such as cameras we need to know their internal parameters and physical properties to be able to make accurate measurement. In this work we studied the main intrinsic and extrinsic parameters of my personal camera. Other aspects could be taken in account such as radial distortion and noise in the CCD array. But it would require a more complicated set up of our mathematical framework to estimate those parameters. Another aspect that could influence a bit our results is the uncertainty in the coordinates of selected points. Indeed our measurement in the real world were almost accurate and we tried to keep the same accuracy in our point selection but we cannot avoid a possible bias.

Finally, the coordinate orientation issue is a serious issue that affect the accuracy of the extrinsic parameters. I unfortunately realized late that this could be the origin of the strange value of some parameters.But this issue might not be too bad, because it's just a permutation of axis that we can understand through the rotation angles (if the method for their computation is accurate). I really hope that this is not a too serious issue for this assignment, but I found out that maybe I should have taken an other orientation of the object coordinates frame to have maybe an easier understanding of the estimation of our extrinsic parameters.

The framework we used to solve and estimate the parameters is linked to Tsai method without radial distortion estimation but an other method was possible. Indeed we could have develop a Levenberg-Marquardt type algorithm, but this iterative method requires a suitable initialization which require a prior knowledge of the range of values of these parameters and the constrains to ensure convergence to solution might stay unknown. So the method we used in this assignment appears to be the most suitable method and the easier to implement to solve our camera calibration problem. Mathematically other techniques could have been used to solve the matrix system such as the use of Pseudo Inverse matrix or other matrix decomposition techniques.

References

[1] Wikipedia contributors. Wikipedia, The Free Encyclopaedia, 2013. Available at: http://en.wikipedia.org, Accessed January, 2013.

[2] D. A. Forsyth, J. Ponce, Computer Vision: A modern approach, Second Edition, Pearson, 2011.

List of Figures