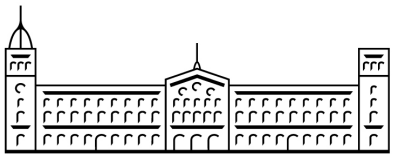

Treball final de grau

GRAU DE MATEMÀTIQUES

Facultat de Matemàtiques
Universitat de Barcelona

3D Reconstruction

Autor: Marc Grau Galofré

Directors: Dr. Joan Carles Naranjo i Dr. Oriol Pujol
Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, June 27, 2016

Abstract

The capacity of retrieving an object from images taken from different positions, known as 3D reconstruction, is being used nowadays in fields like medicine and robotics. In this project, we will give an approach on what are its basis and how is mathematically grounded. Finally, we will give an implementation for carrying out calibrated and uncalibrated reconstructions.

Acknowledgments

I would like to thank my tutors, for accepting this project a week before the beginning, and my friends, for sharing their time working with me. An special thank to my sister Anna.

Contents

1 Introduction

It's widely known that, with a camera, one can project points from space to a plane. But, it is possible the way around? Is there any way to recover the scene from images? In this project, we will answer this question. We will see how a camera can be modeled using projective geometry, and how, with only the images, the original scene can be recovered.

We may find the origin of this idea in the roots of projective geometry. During the 15th century, the Italian Renaissance brought the idea of capturing perspective into a plane. Filippo Brunelleschi (1337-1446), an Italian painter, is thought to be the first one to achieve scientific accuracy in representing linear perspective in paintings. This eagerness of giving a representation, as close as possible, of a real scene evolved for about 450 years later. On 1908, a German mathematician called Von Sanden, used in his PhD, which was focused on studying photometry, a matrix that related points between different views. This matrix is thought to be the precursor of the Essential matrix, from which we will talk in this project. On 1913, an algorithm for computing this matrix was given by Kruppa, a German mathematician. This algorithm needed 5 points to proceed, and involved finding the intersection of two sextic curves. In a time where computers didn't exist, this was a really hard task to do. On 1981, the Essential matrix, as we know nowadays, was introduced by Longuet-Higgins to the recent created computer vision community, with a linear method for computing it. This method used 8 correspondences and gave a single solution. Thirteen years later, on 1994, the Fundamental matrix was finally introduced by Luong and Faugeras. From this point, many algorithms have been given in order to compute it, either minimizing error or computational cost.

In this project, we will center on the deduction of the Fundamental matrix, and the Essential matrix from it. We will see how these matrices can be used in order to obtain points in space from its projection in two images and will take it into practice.

This is an interdisciplinary project, we will need tools from matrix algebra, linear algebra, numerical methods, geometry... This but, has a downside, since we won't deepen much in any of them; too many times we will have to end a section with only a little introduction of it. On the other side, we will be able to give an implementation of the theoretical results in computer. This implementation will be done in Python, and will introduce ourselves into digital image processing and computer vision.

Project Structure

This project is mainly divided in two sections. The first section, formed by the first three chapters, is focused on setting the mathematical background that we need to perform the reconstruction.

We begin by introducing the camera model, seeing how it can be expressed in terms of the camera matrix P from the extrinsic parameters of the camera (its

situation in terms of the coordinate frame) and its intrinsic parameters of the camera, represented through the calibration matrix K. This introduction is ended by relating the calibration matrix with metric properties, a relation that will allow us to compute K from images, without the need of knowing technical details of the camera.
On the following chapter, we will work on the geometry involving two images, known as two views or epipolar geometry. This geometry is focused on studying the relation between what are called correspondent points, i.e, points in different images that are the projection of the same 3D point. This relation is encoded in terms of the fundamental matrix F by what is called the epipolar constrain. We will see how this matrix can be computed using this constrain, and how it can be used to retrieve a pair of camera matrices corresponding to the images up to a projective transformation. We end this chapter by seeing that the knowledge of the calibration matrix reduces the number of possible camera matrices to only four.
The chapter closing this section is focused on the possible reconstructions that may arise, depending on the information we have about the scene and the camera. We also see how discrete image coordinates may be corrected in order to reduce error on the triangulated 3D point an give an algorithm to obtain it.

On the second section, we find the applied chapters. On chapter 5, we give a very minimal introduction to image processing, and how it can be applied to find correspondences between images. Finally, on chapter 6 we apply the results we have been seen along the project in order to perform a calibrated reconstruction.

2 Cameras and Calibration

In this first chapter, we will introduce the pinhole camera model as the basis of this project. We will see how its information can be encoded in matrix form, and how the different aspects of the camera, such as its situation on the world or its internal parameters, affect the image. We will talk about the importance of calibration, and will see how projective geometry allows us to use known information in the world in order to get information from the camera. This process will be developed in terms of the projective space $\mathbb{P}^3 := \mathbb{A}^3 \cup \mathbb{P}(\mathbb{R}^3)$, i.e. the real affine space, extended by adding the infinite directions. Finally, we will give a basic algorithm in order to compute the calibration matrix, which contains the information from the camera internal parameters.

2.1 The Pinhole Model

Without body, without lenses, under this model a camera is reduced to its simplest representation: a tiny hole were light goes through and a plane were it collides, generating an image. Under this approach, we present the model as follows:

Definition 2.1.0.1. *Under the pinhole model, a camera is the central projection from points in space onto a plane, i.e. is a map* $\mathbb{R}^3 \setminus \{C\} \to \pi$ *which sends points* X *in the 3D world to points* x *in the plane* π *as* $x = (C \vee X) \cap \pi$.

We now define the nomenclature: C is the camera center, and its also the center of projection. π is the image plane. The orthogonal ray from π to C is the principal ray, and its intersection with π is the principal point, f is the distance between C and π, called the focal length, the plane parallel to π containing C is known as the principal plane of the camera. Points in space may be referred as world points, whilst points in π will be referred as image points.

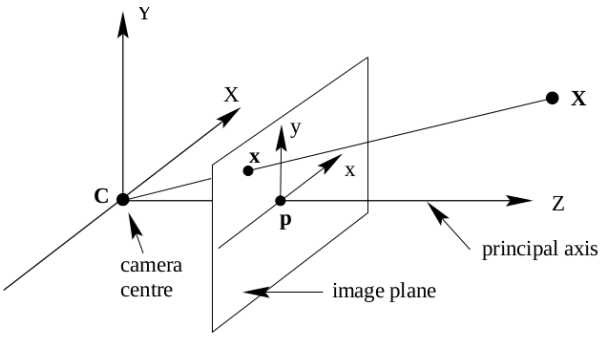

Figure 1: The Pinhole model

[6, chapter 6, pg. 206].

Let's take Euclidean coordinates as follows: let $C = (0, 0, 0)$, the camera center, be the origin, and π the plane $z = f$, the image plane. Let $X = (x, y, z)$ be a point

in the space, is easily seen that the ray $C \vee X$ intersects π at $x = (xf/z, yf/z, f)$. Taking homogeneous coordinates, C becomes $(0:0:0:1)$ and the image plane is $z = ft$. Then, the central projection expression can be expressed as a linear mapping $P : \mathbb{P}^3 \setminus \{C\} \to \pi$ which sends $X = (x:y:z:t)$ to $x = (fx:fy:z)$. That is:

$$\begin{pmatrix} x \\ y \\ z \\ t \end{pmatrix} \to \begin{pmatrix} fx \\ fy \\ z \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ t \end{pmatrix}. \tag{2.1}$$

This matrix may be written as $diag(f, f, 1) \cdot [I|0]$, where $diag(f, f, 1)$ is a 3×3 diagonal matrix and $[I|0]$ is a matrix divided by blocks, where I is a 3×3 identity matrix and 0 is a zero column. We may denote it by P, and it will be called the camera projection matrix. However, the simplicity of the expression is due to some facts that we are now going to discuss.

First of all, we supposed that the camera center C was the origin of our coordinate system, and the principal axis was over the Z-axis, but world points may be expressed in a different coordinate frame, usually referred as world coordinate frame, in which C may not be the origin. If that happens, there will be a rotation and a translation that will relate both systems: $X_{\text{cam}} = R(X - \tilde{C})$, where R is a rotation matrix and $\tilde{C}$ are the coordinates of the camera center in the world coordinate frame.
We were also considering that the principal point was the origin of coordinates in the image plane, which is not be always possible. In this case, the point $X = (x:y:z:1)$ is mapped to $x = (xf + p_x, yf + p_y, z)$, where (p_x, p_y) are the coordinates of the principal point. In this case, the 3x3 diagonal matrix is affected, and is no longer diagonal.
The camera may also be involved in the map expression; some cameras may have not square pixels, this introduce the effect of scale factors in each direction: m_x in the x-axis and m_y in the y-axis. Under this circumstances, the diagonal has the form $diag(\alpha_x, \alpha_y, 1)$, where $\alpha_x = f \cdot m_x$ and $\alpha_y = f \cdot m_y$.
Finally, the pixel coordinate system may not be orthogonal, in that case, an extra parameter s is added to our old diagonal matrix, called the skew parameter.

Adding this parameters to the original $diag(f, f, 1)$, we obtain the calibration matrix:

Definition 2.1.0.2. *The matrix K, containing the internal parameters of the camera, is called the calibration matrix.*

We have seen in a heuristic way which role plays K on codifying the image information. Being more precise, K is defined as the change of coordinate matrix from the image plane to the pixel plane. This can be seen as follows:

Let O be the origin of coordinates of the pixel coordinate frame on the top left corner of the pixel plane. Let x, y be the direction of the horizontal and quasi-vertical axes, considering that the angle θ between them may differ from 90 degrees.

Under this point of view, m_x and m_y may be seen as the amount of pixels per unity on the directions of the axes O_x, O_y respectively. Then, the change of coordinates between the image coordinate frame and the pixel coordinate frame is ruled by the expression:

$$v_1 = \tfrac{1}{m_x} e_1.$$

$$v_2 = \tfrac{1}{m_y}(e_1 \cos\theta + e_2 \sin\theta).$$

Where $(v_1, v_2), (e_1, e_2)$ are vectors on the (pixel, image) coordinate frame. Isolating e_1, e_2, we get the expression we are looking for.

$$e_1 = m_x v_1.$$

$$e_2 = \tfrac{1}{\sin\theta}(m_y v_2 - \cos\theta m_x v_1) = \tfrac{m_y}{\sin\theta} v_2 - m_x v_1 \cot\theta.$$

Multiplying by $diag(f, f, 1)$, we get an explicit expression for K:

$$K = \begin{pmatrix} \alpha_x & -\alpha_x \cot\theta & x_0 \\ 0 & \alpha_y / \sin\theta & y_0 \\ 0 & 0 & 1 \end{pmatrix}.$$

This matrix plays a crucial role in $3D$ reconstruction , since it encodes the transformation from image coordinates to pixel coordinates in the image plane. It will be seen how euclidean information can be received from pictures once this matrix is known, and so a complete metric reconstruction is possible. A camera from which the calibration matrix is known, is called a calibrated camera. Methods for computing this matrix won't be seen in depth, but a relation between this matrix and the absolute conic will be established.

Before that, we will retake our study on the projection matrix and see some of the information it entrains. First of all, we will introduce a compact form of the projection matrix as follows:

$$P = K[R|t].$$

where $t = -R\tilde{C}$, K is the calibration matrix of the camera and R is a rotation matrix. Note that this definition allows the immediate retrieve of many of the elements of the pinhole model. It's immediate to see that C is the right null space of P. The principal plane of the camera can also be directly obtained as the plane defined by the last row of P, while the last column of P defines the image of the world coordinate frame into the image plane. With these elements, the camera can be situated in the euclidean coordinate frame from its matrix.

In our model, P is a 3×4 matrix with its 3×3 left submatrix is non-singular. Later, we will mention other models where the non-singularity constrain is not required.

The camera matrix can present some changes depending where the camera center is supposed to be. We will not study in detail the differences it generates, but will talk briefly about it now. If C is located on the infinite plane and this plane coincides with the principal plane, the left 3×3 submatrix is singular and there's no principal point in the image. These cameras are called infinite cameras and are often used when points in space are in similar depth. we will focus on the cameras with invertible left 3×3 submatrix, called the finite cameras.

To summarize, P encodes the transformation from world coordinates to pixel coordinates, in other words, is a change of coordinates matrix. We can distinguish two separated parts on this process, the first one is carried out by the rotation matrix R by changing the world coordinate frame to the camera coordinate frame. In this part is where the called extrinsic parameters of the camera (such as situation or orientation under the world coordinate frame) play part. Once the camera coordinate frame is set, we procede with the projection into the image plane and hence, the pixel plane. This projection is encoded by the intrinsic parameters of the camera and therefore, ruled by the calibration matrix. This process is summarized on the next figure:

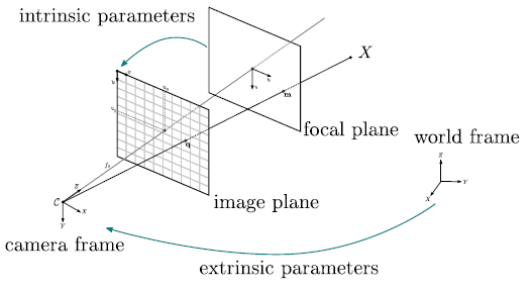

Figure 2: Projection by P.

2.2 Calibration

Now, we will take a closer look to the calibration matrix K. We will see its importance in terms of obtaining information from an image and will justify it using projective invariants. Finally, it will be seen how this matrix can be computed, but again, we won't enter in deep in this subject, since it could be a whole new project.

An image point x in π back-projects to a ray defined by X and C. On the camera coordinate frame, calibration relates the image point to the ray's direction. Let X be a point in space, X can be expressed as $X = \lambda\vec{d} + C$, where $\vec{d} = (X - C)$. In the camera's euclidean coordinate frame, $C = (0, 0, 0)$, so $X = \lambda\vec{d}$.
Going back to $\mathbb{P}^3$, this point is projected to $x = PX = K[I|0](\vec{d}, 1) = K\vec{d}$, if K is known, since it's not singular, $\vec{d}$ can be computed from the image point as $\vec{d} = K^{-1}x$.

So, it's clear that the knowledge of the calibration matrix provides us very interesting opportunities towards getting metric information from images. However,

this matrix is different for each camera, and its parameters are provided by technological components inside the camera (as for example, the focal length f, the pixel distortion...). A direct attempt from getting them would imply the dismantling of the camera, accurate measurements and the subsequent mounting of the camera, where the parameters that were so carefully measured, would almost be altered. Then, how can K be obtained? The following results will show how the projective character of P can be used for the obtainment of K. The projective geometry provides us very important tools for this calculation. Now, we'll leave K and P apart and take a little trip inside the projective geometry.

First of all, we'll suppose that there's a metric on E, consisting of the usual inner product $<,>: E \times E \to \mathbb{R}^3$ defined by the identity matrix Id. We'll assume that $\pi_\infty := \{(x : y : z : t) | t = 0\}$.

Our goal is to relate the inner product derived from the affine space with a projective invariant, so information from angles in the world can be used in the imaged angles. This invariant will prove to be the cross ratio; we'll show that's invariant under projective transformations and how angles can be related with it.

Definition 2.2.0.1. *Let $a, b, c, d \in \mathbb{P}^1$, no three of them equal. In a fixed reference on P_1, the cross ratio of this points is defined as:*

$$(a, b; c, d) = \frac{|ac||bd|}{|bc||ad|}.$$

Where $|xy|$ is the determinant of x and y with coordinates on the line coordinate frame.

The invariance of the cross ratio almost projective transformation can be easily seen as follows.

Lemma 2.2.0.1. *Let H be a projective transformation of a line. Then:*

$$(Ha, Hb; Hc, Hd) = (a, b; c, d).$$

In other words, the cross ratio does not depend on the choice of the reference.

Proof. $(Ha, Hb; Hc, Hd) = \dfrac{|HaHc||HbHd|}{|HbHc||HaHd|} = \dfrac{det(H)^2|ac||bd|}{det(H)^2|bc||ad|} = \dfrac{|ac||bd|}{|bc||ad|} = (a, b; c, d).$
□

The cross ratio needs four points to be computed; two concurrent lines l_1, l_2 on the world intersect π_∞ on two points p_1, p_2, representing the lines directions. So other two points are needed; since all points must lie on the same line and the known two points lie on π_∞, the other two lie on π_∞ too. Since every pair of concurrent lines defines an angle, and we want it to be invariant under the projection P, it's not a surprise that the two remaining points are determined once p_1, p_2 are given. We introduce a very important object, which will give us the relation we are looking between projective an affine geometry.

Definition 2.2.0.2. *The absolute conic Ω_∞ is the conic lying at π_∞ defined by the equation:*

$$\Omega_\infty := x^2 + y^2 + z^2 = 0.$$

This conic is defined by the Id matrix, and has no real points on it. Finally, joining all these results, we are in position of announcing the next proposition.

Proposition 2.2.0.1. *(Laguerre's Formula) Let l, m be two finite lines on $\mathbb{P}^2$ and let L, M be the corresponding intersection with the line at infinity l_∞. Then, the angle α between l_1 and l_2 is:*

$$\alpha = \tfrac{1}{2i} ln((L, M; I, J)).$$

Where I, J are the intersection between the line $L \vee M$ and the absolute conic.

Proof. Let $l = p_1 + < L >$, $m = p_2 + < M >$ be the two lines. Then, their respective infinite points are L, M respectively. Since they are defined up to scale, we can consider that L and M are unity vectors, dividing by their norm if its necessary. Therefore, their coordinates in l_∞ are $L = (cos\alpha : sin\alpha)$, $M = (cos\beta : sin\beta)$. We can suppose $\beta \leq \alpha$.

So, $\theta = (\alpha - \beta)\ mod(\pi)$ is the angle we are looking for. Let's start with the cross ratio expression:

$$(L, M; I, J) = \frac{\begin{vmatrix} 1 & i \\ cos\alpha & sin\alpha \end{vmatrix}}{\begin{vmatrix} 1 & i \\ cos\beta & sin\beta \end{vmatrix}} : \frac{\begin{vmatrix} 1 & -i \\ cos\alpha & sin\alpha \end{vmatrix}}{\begin{vmatrix} 1 & -i \\ cos\beta & sin\beta \end{vmatrix}} = \frac{sin\alpha - icos\alpha}{sin\beta - icos\beta} : \frac{sin\alpha + icos\alpha}{sin\beta + icos\beta}.$$

We may rewritte this expressions in polar coordinates, and using the pair, odd properties of the *cos* and *sin* funtions, we obtain:

$$(L, M; I, J) = \frac{i}{i}\frac{cos\alpha + isin\alpha}{cos(-\alpha) + isin(-\alpha)}\frac{i}{i}\frac{cos(-\beta) + isin(-\beta)}{cos\beta + isin\beta} = \frac{e^{i\alpha}}{e^{-i\alpha}}\frac{e^{-i\beta}}{e^{i\beta}}$$

$$= e^{2i(\alpha-\beta)} = e^{2i\theta}.$$

So, isolating θ, we obtain:

$$\theta = \frac{1}{2i} ln((L, M; I, J)).$$

As we wanted to prove. □

So far, we have seen that there's a relation between angles and projectives invariants, so the knowledge of, for example, orthogonality in the world is transferred to the image in terms of the absolute conic. We'll go further in this direction, and see how K and Ω_∞ are related. First of all, since points in π_∞ are of the form $X_\infty = (d : 0)$, where d is an homogeneous $3-$vector, its immediate to see that they are mapped to:

$$PX_\infty = KR[I|-\tilde{C}]\begin{pmatrix} d \\ 0 \end{pmatrix} = KRd.$$

Corollary 2.2.0.1. *The mapping between π_∞ and the image plane is given by the planar homography H, where:*

$$H = KR.$$

Since Ω_∞ lives in π_∞, we have to see how conics are affected by a planar homography. It was seen in the projective geometry course that the image of a conic Q under an homography H is:

$$Q' = H^{-t}QH^{-1}.$$

Finally, we get what we were looking for:

Theorem 2.2.0.1. *The image of the absolute conic Ω_∞ (IAC) is $\omega = (KK^t)^{-1}$.*

Proof. Ω_∞ lies on π_∞, so it's image under P is ruled under the planar homography $H = KR$, so Ω_∞ is mapped to $\omega = H^{-t}\Omega_\infty H^{-1} = (KR)^{-t}Id(KR)^{-1}$, since R is a rotation matrix, and therefore, orthogonal, $\omega = K^{-t}RR^{-1}K^{-1} = (KK^t)^{-1}$.

□

So ω depends only on the internal parameters of the camera, it does not depend on the camera orientation or position. We will use this fact to get an universal form for computing K.

Theorem 2.2.0.2. *Determining ω in an image also determines K.*

Proof. This result will follow immediately once the next lemma is proven:

Lemma 2.2.0.2. *Cholesky Factorization: Let A be a symmetric matrix positive-defined. Then A can be uniquely decomposed as $A = KK^t$, where K is an upper-triangular real matrix with positive diagonal entries.*

Proof. Let UDU^t be the SVD decomposition of A. Since A is symmetric, we know that D has positives entries and U is orthogonal. Let E be the square root of D, that is $D = EE^t$, being E diagonal. Then $A = UDU^t = UEE^tU^t = VV^t$, where $V = UE$. Now, taking the QR decomposition of V, we get $V = KQ$, where Q is orthogonal and K is an upper-triangular matrix. Replacing this on the last expression of A, we get $A = VV^t = KQ(KQ)^t = KQQ^tK^t = KK^t$. We can ensure that K has positive entries by multiplying by a diagonal matrix with entries ± 1 without changing the product KK^t.
To see the uniqueness of the factorization, let K_1, K_2 two upper triangular matrix satisfying $A = K_1K_1^t = K_2K_2^t$ then $K_2^{-1}K_1 = K_2^tK_1^t$. Since K_i are upper triangular, the right side of the equation is upper triangular, while the left side is lower triangular. Therefore, they both must be diagonal: $D = K_2^{-1}K_1 = K_2^tK_1^t$. However, $K_2^tK_1^t$ is the inverse transpose of $K_2^{-1}K_1$, and so D is it's own inverse transpose, and hence, it's entries must be ± 1. If K_1 and K_2 have positive entries, it must be $D = Id$, so $K_1 = K_2$. □

Since ω is a 3×3 symmetric positive defined matrix, we can apply the Cholesky factorization on it to obtain K. $\square$

So, the problem has been reduced to find ω. Now is when angles come into scene. We have proved they relation with Ω_∞, and this relation is where our this result will hold. Let α be the image of a known angle between the world lines d_1 and d_2. The cosine formula asserts:

$$cos(\alpha) = \frac{d_1^t d_2}{\sqrt{d_1^t d_1}\sqrt{d_2^t d_2}}.$$

Seeing d_1 and d_2 as back projections of the image plane points x_1 and x_2 by the camera matrix $P = KR[Id|t]$, we can develop this expression as follows:

$$cos(\alpha) = \frac{(R^{-1}K^{-1}x_1)^t(R^{-1}K^{-1}x_2)}{\sqrt{(R^{-1}K^{-1}x_1)^t(R^{-1}K^{-1}x_1)}\sqrt{(R^{-1}K^{-1}x_2)^t(R^{-1}K^{-1}x_2)}} =$$

$$= \frac{x_1^t(K^{-t}RR^{-1}K^{-1})x_2}{\sqrt{x_1^t(K^{-t}RR^{-1}K^{-1})x_1}\sqrt{x_2^t(K^{-t}RR^{-1}K^{-1})x_2}} = \frac{x_1^t(KK^t)^{-1}x_2}{\sqrt{x_1^t(KK^t)^{-1}x_1}\sqrt{x_2^t(KK^t)^{-1}x_2}} =$$

$$= \frac{x_1^t\omega x_2}{\sqrt{x_1^t\omega x_1}\sqrt{x_2^t\omega x_2}}.$$

It follows immediately that if d_1 and d_2 are orthogonal, x_1 and x_2 are conjugated by ω, so $x_1\omega x_2 = 0$, so it gives linear constrains on ω. In general, any known α gives constrains on ω, although if d_1 and d_2 are not orthogonal, these constrains are not linear.

We have enough tools to give an algorithm for calculating K. It uses what is called a calibration device, an object with very specific geometric information, such as a chess table. In this case, we will use the image of three squares, situated on none parallel, neither orthogonal planes. Consider one of the squares. The correspondence between its four corner points and the image define an homography H between the plane π of the square and the image. Applying H to the circular points on π (every plane intersects π_∞ in a line, and this line intersects ω in the two circular points of π), determines their images as $H(\pm i : 1 : 0)$. This gives us two points on the unknown ω. Applying this methodology with the other squares, we get a total of six points on ω, enough to compute it, since only 5 are necessary to determine a conic. So the algorithm is the following:

1. For each square compute the homography H that maps its corners: $(0,0), (1,0), (0,1), (1,1)$ to their image points.
2. Compute the imaged circular points for the plane of that square as $H(\pm i : 1 : 0)$.
3. Fit a conic ω to the six imaged circular points; this conic is determined up to scale.

4. Compute the calibration matrix K from $\omega = (KK^t)^{-1}$ with the Cholesky factorization.

[6, chapter 6, pg. 211].

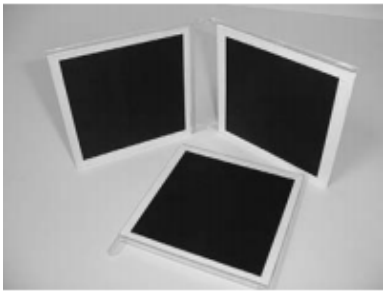

Figure 3: Calibration Device.

The three none perpendicular neither parallel planes allow the computation of the calibration matrix from Ω_∞.
[6, chapter 6, pg. 211].

Although this method may make finding the calibration matrix seem to be quite simple, this is far from being true. This method is only applicable on very controlled environments, were external issues as, for example, light influence can be handled, and even under this circumstances, the presence of errors is quite significant, either in form of image measurements or in the template geometry, so trusting in only one template may be dangerous. The only way to deal with this facts is through a large bank of data; without varying the camera parameters, a significant number of templates should be used in order to get the correspondent calibration matrix. Then optimization processes can be carried out from the data, and the most suitable calibration matrix can be obtained.

3 Two Views Geometry

Let's guess the next situation: we make a search on google images for a famous monument; for sure we will get a huge amount of pictures from every viable point of view of it. But how are this images related? The same point in space is likely to be mapped to different image locations in different photos. Intuitively, this images of the same point may seem to be, somehow, related, and is in this relation in which this chapter is focused. We will study the geometry of two views, see how the idea of correspondent points leads us to find a way for locating points on one image in the other, and how this correspondences can help us at determining the relative positions of the two cameras.

3.1 Epipolar Geometry

Let π_1, π_2 be two image planes, X a world point which is mapped to x, x' respectively. We call x, x' corresponding points. It will be seen that corresponding points are needed in order to compute the reconstruction from the images. Now we are going to study this relation of correspondence, and try to use it in order to search for corresponding points between images. We can begin observing that x, x' are coplanar with the camera centers, lying in the plane $\pi := X \vee l_b$, where $l_b = C_1 \vee C_2$ is called the base line. We can use this fact to try to determinate x' once x is known: π can be computed from x, C_1, C_2, and therefore, x' must lie in it's intersection with the second image plane.

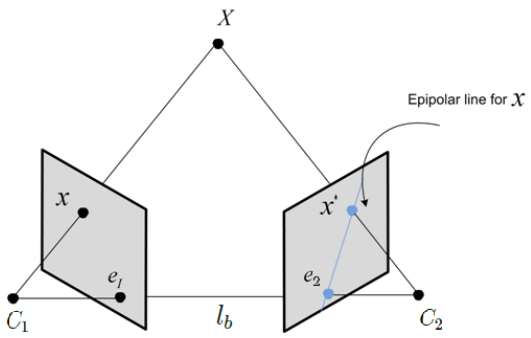

Figure 4: The main objects of the epipolar geometry.

In gray, the image planes. In blue, the epipolar line generated by the correspondent points x, x'.

First of all, let's introduce some notation:

Definition 3.1.0.1. *The main components of the epipolar geometry can be defined as follows:*

-We call epipol the point of intersection between the base line and the image plane:

$$e := (C_1 \vee C_2) \cap \pi_1,\ e' := (C_1 \vee C_2) \cap \pi_2.$$

-An epipolar plane is defined as a plane containing the baseline. An epipolar plane can be generated by every x in the image plane different from the epipole, so there is a one parameter family of epipolar planes.

-An epipolar line is defined as the intersection between an epipolar plane an image plane. Therefore, epipolar lines from different image planes are corresponding if they lie on the same epipolar plane and all epipolar lines contain the correspondent epipole.

Proposition 3.1.0.1. *Let x, x' be corresponding points in π_1, π_2 respectively. Then, x' must lie on the epipolar line $l' = (x \vee C_1 \vee C_2) \cap \pi_2$.*

Proof. Since x, x' are corresponding points, they are the projection from the same point X in space. Therefore, x' can be expressed as $x' = (X \vee C_2) \cap \pi_2$, but X lies on the epipolar plane $\pi_e = x \vee C_1 \vee C_2$, since it lies on the ray $C_1 \vee x$. Then the ray $X \vee C_2$ lies on the plane too, so the intersection between this ray and π_2 lies on the intersection between π_e with π_2, which is l'.

□

Thus, there is a map from a point in one image to its corresponding epipolar line in the other image. It leaves us to the next section, where we are going to characterize it and see some important properties in terms of our objective.

3.2 The Fundamental Matrix

Geometrically, this map can be seen as follows:

Let X be a world point, let π be a plane not containing neither of the camera centers such that $X \in \pi$. The ray $x \vee C_1$ meets π in X and then is projected to x' on the other image plane. This point must lie on the epipolar line l' by the last proposition. x and x' are images of the same point X, therefore, they are projectively equivalent. Expanding this result to the set of corresponding points between the images, we see that $\{x_i \leftrightarrow x_i'\}$ is projectively equivalent to planar points X_i. Thus, there is a planar transformation H_π mapping x_i to x_i'. This process is called transfer via a plane.

Now, the epipolar line corresponding to x can be constructed as the ray from e' to x', that is $l' = e' \vee x'$. We now introduce some notation:

Definition 3.2.0.1. *Let $v = (x, y, z)$ be a $3-$vector, then it's skew-symmetric matrix is defined as:*

$$[v]_x = \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix}.$$

Lemma 3.2.0.1. *Let v' be a $3-$vector, then:*

$$v \wedge v' = [v]_x v' = (v^t [v']_x)^t.$$

Proof. This follows immediately from a simple calculation: Let $v = (x, y, z), u = (a, b, c)$ be two two vectors in $\mathbb{R}^3$. Then:

$$v \times u = \begin{vmatrix} i & j & k \\ x & y & z \\ a & b & c \end{vmatrix} = (yc - zb, zc - xc, xb - ya).$$

$$[v]_x u = \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = (yc - zb, zc - xc, xb - ya).$$

□

Thus, we can see $[e']_x$ as the following map:

$$\begin{array}{rccc} [e']_x : & \mathbb{P}^2 \setminus \{e'\} & \to & e'^* \\ & [x'] & \mapsto & e' \vee x' =: l' \end{array}$$

where l' is the line containing $[e']$ and $[x']$ with coefficients $e' \wedge x'$. Since $x' = H_\pi$, we have:

$$l' = [e']_x H_\pi x = Fx$$

Definition 3.2.0.2. *The fundamental matrix F is defined as $F := [e']_x H_\pi$, where H_π is the transfer mapping from one image to another via any plane π and e' is the epipol from the second image plane.*

Observation 1. Since $[e']_x$ has rank 2 and H_π has rank 3, F is a rank 2 matrix. This can also be seen from the fact that F represents a mapping from the projective plane $\mathbb{P}^2$ from the first image to the pencil of epipolars lines through the epipol e'.

The fundamental matrix can be obtained in several ways. We will mention some of them, and later will give an algorithm to carry out this work. The simplest way is also the less useful, since it implies knowing everything about the two cameras, their position and both calibration matrices.

Definition 3.2.0.3. *Let $P \in M(m, n; K)$ be a general matrix. Then the right pseudo inverse of P is defined as a matrix $P^+ \in M(n, m; K)$ such as:*

$$PP^+ = Id.$$

Lemma 3.2.0.2. *Let P, P' be the camera matrices. F can be computed as:*

$$F = [e']_x P' P^+.$$

Where P^+ is a pseudo inverse of P.

Proof. This comes directly from the definition. $P^+(x)$ chooses a preimage of x, so $P'P^+$ determines with e' the epipolar line. □

Example 1. Let's suppose we know P and P'. This means to knowing the camera position, orientation and calibration matrix. For both cameras. As we can see is a very limited scenario, but in this controlled environment, the fundamental matrix is obtained straightforward from the last result. If the camera position and orientation is known, we can take new coordinates, allowing the first camera center to be the origin of the world coordinate frame. This will simplify the first camera matrix, while only affecting the rotation matrix from P' by a known affinity. So, we have:

$$P = K(Id|0),\ P' = K'(R|t),\ e' = P'C.$$

Note that P^+ can be obtained immediately as $\begin{pmatrix} K^{-1} \\ 0 \end{pmatrix}$. Then:

$$F = [e']_x P'P^+ = [P'C]_x P'P^+ = [K'(R|t)\begin{pmatrix} 0 \\ 1 \end{pmatrix}]_x K'(R|t)\begin{pmatrix} K^{-1} \\ 0 \end{pmatrix} = [t]_x K'RK^{-1}.$$

We will now try another approach, focusing on the correspondence relation between points. The fundamental matrix will give us algebraic conditions to verify if two points x, x' can be correspondent. We are going to formalize this result, and see some other important properties it hides.

Proposition 3.2.0.1. *For any pair of corresponding points x, x'. It is fulfilled:*

$$x'^t F x = 0.$$

Proof. If x, x' are corresponding points, then x' must lie on the epipolar line $l' = Fx$. Therefore, $x'^t l' = x'^t F x = 0$. If x, x' are points satisfying $x'^t F x = 0$, then, by the definition of F, the rays defined by x, x' must be coplanar, then are coplanar. □

So F can be characterized in terms of correspondent points. This is an important result, since it implies that F can be computed through image correspondences, without knowing any of the camera matrices. Later we will give an algorithm to compute F using this fact, now we are going to focus on the properties of F.

Proposition 3.2.0.2. *F satisfies:*

1. *If F is the fundamental matrix of the pair (P, P'), F^t is the fundamental matrix of the pair (P', P).*
2. *Let x, x' be points in the first and second images respectively. Then $l' = Fx$ is the epipolar line corresponding to x, and $l = F^t x'$ is the epipolar line corresponding to x'.*
3. *Let e, e' be the two epipoles. Then, $e'^t F = Fe = 0$, in other words, e' is the left null-vector of F, and e is the right null-vector of F.*

Proof. 1 is easily seen taking the expression of correspondence $x'^t F x = 0$, where F is the fundamental matrix of the pair (P, P'). Then $0 = (x'^t F x)^t = x^t F^t x'$, hence F^t is the fundamental matrix for the pair (P', P).
2 follows immediately from 1. If x, x' are corresponding points, then $x'^t F x = 0$ and therefore, $x'^t(Fx) = x'^t l' = 0$. Applying an analogous argument, we have that l is the epipolar line corresponding to x.
3 can be obtained directly from the epipole definition. Since every epipolar line contains the correspondent epipole, then, for every x in the first image, it will be satisfied $e'^t F x = 0$, hence e' is the left null-vector of F. We also deduce that $e^t F^t = 0$, therefore $(e^t F^t)^t = Fe = 0$, hence e is the right null-vector of F. □

We have seen that F can be determined (up to scale) by the camera matrices. One of the most interesting properties of F is that the other way is also true: F may be used to determine the camera matrices of the two views. The map $l' = Fx$ and the correspondence condition $x \leftrightarrow x'$ are projective relationships, and consequently, depend only on projective coordinates on the image, and not, for example, on the camera position or orientation. The following results are oriented in seeing how F reacts to changes on the camera matrices and image coordinates, with the aim to see that F determines the camera matrices.

Lemma 3.2.0.3. *Let F be the fundamental matrix, H a projective transformation affecting the image coordinates, $\tilde{x} = Hx, \tilde{x'} = H'x'$. Then, if x, x' are corresponding points ($x \leftrightarrow x'$), there is the corresponding map $\tilde{l'} = \tilde{F}\tilde{x}$, where $\tilde{F} = H'^{-t} F H^{-1}$.*

Proof. If $x \leftrightarrow x'$, is fulfilled $x' F x = 0$. We know $\tilde{x} = Hx$, $\tilde{x'} = H'x'$.
Then $H^{-1}\tilde{x} = x, H'^{-1}\tilde{x'} = x'$, so:

$$0 = x' F x = (H'^{-1}\tilde{x'})^t F (H^{-1}\tilde{x}) = \tilde{x'}^t H'^{-t} F H^{-1} \tilde{x} = \tilde{x'}^t \tilde{F} \tilde{x} = 0.$$

As we wanted to prove. □

This result involves only variations on the image coordinate frame, but we can derive a similar result involving transformations to the camera matrices themselves. That means that F is unaffected by changes on the world coordinate frame, this fact is now formalized:

Theorem 3.2.0.1. *Let (P_1, P_1'), (P_2, P_2') two pairs of camera matrices. Then, exists a projective transformation of 3-space H such that $P_1 = P_2 H, P_1' = P_2' H$ if and only if (P_1, P_1'), (P_2, P_2') have both the same fundamental matrix.*

Proof. $\Rightarrow$
Let's suppose that $P_2 = P_1 H, P_2' = P_1' H$. We begin observing that if X is a world point, $P_1 X = (P_1 H)(H^{-1} X)$, $P_1' X = (P_1' H)(H^{-1} X)$. Then, if x, x', are both projections of the point X by P_1, P_1' respectively, they are also projection of the point $H^{-1}X$ by $(P_1 H, P_1' H)$ and therefore, F is the fundamental matrix for this pair of cameras. □

What we have seen now implies that, although (P, P') determined a unique fundamental matrix, the converse is not true, since they are determined up to a projective transformation. By proving the other implication, we put an end to the ambiguity: the fundamental matrix will prove to determine up to a space projective transformation the two camera matrices. Given this ambiguity, it is natural to introduce a canonical form for the pair of camera matrices given by F. We will take the first camera matrix as $P = (Id|0)$, where Id is the 3×3 identity matrix and 0 the 3-null vector. This is a viable choice, since we can always take the projective transformation $H = P^{*-1}$, where P^* is the non singular augmented 4×4 P matrix. Taking this form for P, we have the next result:

Lemma 3.2.0.4. *The fundamental matrix corresponding to the pair of cameras $P = (Id|0)$ and $P' = (M|m)$ is:*

$$F = [m]_x M.$$

where M is a none singular 3×3 real matrix and $m \in \mathbb{R}^3$.

Proof. By the lemma 3.2.0.4., we know $F = [e']_x P' P^+$. Let's adapt this result to our case, where:

$$e' = P'C,\ P^+ = \begin{pmatrix} Id \\ 0 \end{pmatrix}, C = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

So we have:

$$F = [P'C]_x P' P^+ = [(M|m) \begin{pmatrix} 0 \\ 1 \end{pmatrix}]_x (M|m) \begin{pmatrix} Id \\ 0 \end{pmatrix} = [m]_x M.$$

□

This canonical form notation will serve us on simplifying the proof of the remaining implication.

Proof. $\Leftarrow$
So, we have the same fundamental matrix F for the two pair of cameras (P_1, P_1'), (P_2, P_2'). First of all, we assume that each pair is on the canonical form:

$$P_1 = P_2 = (Id|0),\ P_1' = (R_1|t_1),\ P_2' = (R_2|t_2).$$

Then, as we have just seen, F can be written as $F = [t_1]_x R_1 = [t_2]_x R_2$. Now, with the help of the following technical lemma, the proof will follow easily.

Lemma 3.2.0.5. *Suppose that a rank 2 matrix M can be decomposed in two different ways: $M = [v]_x V$ and $M = [u]_x U$, where $u, v \in \mathbb{R}^3$, and U, V are rank 3 real matrices. Then, $\exists k, t$, $k \in \mathbb{R}$, $t \in \mathbb{R}^3$ such that $u = kv$ and $U = k^{-1}(V + vt^t)$.*

Proof. The existence of k can be seen almost immediately, since v^t, u^t are both from the left ker of M:

$$0 = v^t[v]_x V = v^t M.$$

And similarly for u. Now, from $[v]_x V = [u]_x U$, it follows that $[u]_x U = [kv]_x U = [v]_x kU$. Thus, $0 = [u]_x U - [v]_x V = [v]_x(kU - V)$, and since the right ker of $[v]_x$ is generated by v, $Im(kU - V) \subset< v >$, so $kU - V = (vt_1\, vt_2\, vt_3)$ for $t_i \in \mathbb{R}$. Setting $t = (t_1, t_2, t_3)$, we have $kU - V = vt^t$. Then, we only have to isolate U from the expression $(kU - V) = vt^t$, obtaining $U = k^{-1}(V + vt^t)$. □

This lemma allows us to express P_2' in terms of P_1'; $P_2' = (k^{-1}(R_1 + t_1 t^t)|kt_1)$, and reduces the proof in constructing a suitable transformation. Let H be the following matrix:

$$H = \begin{pmatrix} k^{-1}Id & 0 \\ k^{-1}t^t & k \end{pmatrix}.$$

With this choice, we get that $P_1 H = k^{-1}(Id|0) = k^{-1}P_2$, and $P_1' H = (R_1|t_1)H = (k^{-1}(R_1 + t_1 t^t)|kt_1) = (R_2|t_2) = P_2'$ Thus the pairs (P_1, P_1'), (P_2, P_2') are projectively equivalent.

□

And directly from the theorem:

Corollary 3.2.0.1. *To have the same fundamental matrix F, is an equivalence relation in the set of the pair of camera matrices.*

3.3 Calculating the Fundamental Matrix

In this section we will talk about how the fundamental matrix can be obtained from the images, only through point correspondence. We will see some methods of calculating it, and will give a justification for every method. However, this will only be an introduction to the subject, since going in deep into it would require a wide study on algorithms, error treatment and applied statistics. In this section, we will suppose that correspondences are given. In later chapters, we will see how correspondent points may be obtained, and will put this into practice.
We are aiming to obtain F through point correspondence, so it's natural to take the equation $x'^t F x = 0$ as its definition. Now, let's suppose we start from the very beginning. What do we know from, the yet unknown, F? It's a 3×3 matrix, so it has 9 entries, which are translated as nine degrees of freedom. Since it's a projective entity, it's up to scale, so only 8 degrees are left. Last of all, we know F has rank 2 i.e. $det(F) = 0$, which is reflected as another constraint, hence 7 degrees are left. For a general fundamental matrix, we can go no further in reducing its degrees of freedom, so it turns out to be 7 the minimum number of correspondences required to calculate F.

$$F = \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix}.$$

So, our goal is to solve the equation:

$$x'^t F x = 0.$$

Which may be rewritten in terms of the points coordinates as follows:

$$\alpha'\alpha f_{11} + \alpha'\beta f_{12} + \alpha' f_{13} + \beta'\alpha f_{21} + \beta'\beta f_{22} + \beta' f_{23} + \alpha f_{31} + \beta f_{32} + f_{33} = 0.$$

Where $x = (\alpha, \beta, 1)$, $x' = (\alpha', \beta', 1)$. To simplify the notation, we may refer to the vector $(f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33})$ as f. This allows us to present the last expression as an inner product between vectors:

$$(\alpha'\alpha, \alpha'\beta, \alpha', \beta'\alpha, \beta'\beta, \beta', \alpha, \beta, 1) f = 0 \ .$$

Thus, from a set of n correspondences, we get a system of n linear equations:

$$Af = \begin{pmatrix} \alpha'_1\alpha_1 & \alpha'_1\beta_1 & \alpha'_1 & \beta'_1\alpha_1 & \beta'_1\beta_1 & \beta'_1 & \alpha_1 & \beta_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha'_n\alpha_n & \alpha'_n\beta_n & \alpha'_n & \beta'_n\alpha_n & \beta'_n\beta_n & \beta'_n & \alpha_n & \beta_n & 1 \end{pmatrix} f = 0.$$

So, f generates the ker of A, in consequence, A can have, at most, rank 8, and if the rank is exactly 8, f results to be the right ker of A. But we find a great mishap at putting this to practice: errors. Even with an ideal camera, the presence of errors cannot be ignored, due to the character of the P matrix. The projection from space to the image plain hides a considerable source of errors: a continuous to discrete map. A pixel in the image may be the projection of a neighborhood of space points; this can be easily seen while zooming an image, when the pixels become visible to human eye. So, when taking coordinates on an image point, there's an entire set of space points that are projected to these coordinates. This fact can cause the resulting matrix to have rank 3, which means that epipolar lines do not meet in one point. Lately, we will discuss how we face this problem. For now, we will just see how F can be computed; see how errors might be handled would require an entire project.

We begin talking about the case where less correspondences are required. If we have 8 different correspondences i.e. A has rank 8, F is obtained straight from the right ker of A. By this way, the singularity constrain does not appear, and hence F might not be singular. So, we reduce to have 7 correspondences. This leads to have a 7×9 matrix A with, we will suppose, rank 7. Under this circumstances, the equation $Af = 0$ has a 2-dimensional space as solution, which may be written as $aF_1 + (1-a)F_2$, where a is a variable scalar. F_1 and F_2 are matrices obtained from f_1, f_2, the two generators of A's kernel. The rank 2 constrain may be used by imposing:

$$det((aF_1 + (1-a)F_2) = 0.$$

Since F_1 and F_2 are known, this lead to a degree 3 polynomial equation in a. Solving this new equation, we might find that it can either have one or three real solutions (the complex solutions are discarded [7, Reconstruction from Other Configurations]). Substituting either of the solutions into $F = aF_1 + (1-a)F_2$, we get a possible solution for the fundamental matrix. This procedure is called the seven point algorithm.

So, solving the equation with the minimum number of point correspondence does not translate into a simple way to compute F, since it requires the solving of a cubic polynomial in order to satisfy the singularity constrain. This fact motivate the introduction of another method for computing F, called the eight-point algorithm. First of all, we will try to minimize the dependence on the image coordinates that F has by normalizing them. This means applying a translation over the image coordinates, in order to make the centroid of the image the origin of the coordinate frame, and scaling, in order to keep points inside the $\sqrt{2}$ circle. This process can be packed up into the matrix:

$$T = \begin{pmatrix} k & 0 & -k\mu_x \\ 0 & k & -k\mu_y \\ 0 & 0 & 1 \end{pmatrix}.$$

Where $k = \sqrt{2}/m$ is the scaling factors, being m the mean of the distances from origin, μ_x, μ_y the mean of the x, respectively y, coordinate of the point set. So why normalization of the data is important? We will only give a little example. Considering we have noisy data, let's suppose that there's a correspondence between the point $x = (100, 100, 1)$ in the first image and the point $x' = (100, 100, 1)$ in the second. Then, the quadratic entries of the vector $(\alpha'\alpha, \alpha'\beta, \alpha', \beta'\alpha, \beta'\beta, \beta', \alpha, \beta, 1)$ are of order 10^4, the linear entries of the order of 10^2 and the last entry will always be a unity. As we can see, this is an appropriate circumstance for error propagation. By normalizing data, we impose that all factors are of a similar order, which lead us into a more accurate solution for our problem. As the name might suggest, we start from eight different correspondences, which means that F is determined by A's kernel, having it rank 1. Here, we found ourselves in front of the same problem we have already mentioned: this null-space of A may not be singular. We will face this problem with the following result:

Lemma 3.3.0.1. *Let $F = UDV^t$ be the SVD from a real 3×3 matrix F, with $D = (\lambda_1, \lambda_2, \epsilon)$, $\epsilon \leq \lambda_2 \leq \lambda_1$, none of them 0. Then, the matrix*

$$\tilde{F} = Udiag(\lambda_1, \lambda_2, 0)V^t.$$

minimizes $d(\tilde{F}, F)^2$, satisfying $det(\tilde{F}) = 0$, where d is the euclidean distance in $\mathbb{R}^9$

Proof. $diag(\lambda_1, \lambda_2, 0)$ is being denoted as D'. First of all, we will prove that the matrices U and V play no role in this result.

Lemma 3.3.0.2. *Let M, N be real 3×3 matrices, U an orthogonal matrix. The following statements are satisfied:*

- $d(UM, UN)^2 = d(M, N)^2$.
- $d(MU, NU)^2 = d(M, N)^2$.

Proof. We will prove the first case, the second is analogous. First of all, we will see that orthogonal matrices preserves norms.
Let $x \in \mathbb{R}^n$, $U \in O(n)$ then:

$$\|Ux\|^2 = (Ux) \cdot (Ux) = (Ux)^t(Ux) = x^t U^t U x = x^t x = \|x\|.$$

In order to apply this result, we define the matrix $U' \in M_{9\times 9}$ in terms of the entries of U as follows:

$$U' := \begin{pmatrix} U_{11} & 0 & 0 & U_{12} & 0 & 0 & U_{13} & 0 & 0 \\ 0 & U_{11} & 0 & 0 & U_{12} & 0 & 0 & U_{13} & 0 \\ 0 & 0 & U_{11} & 0 & 0 & U_{12} & 0 & 0 & U_{13} \\ U_{21} & 0 & 0 & U_{22} & 0 & 0 & U_{23} & 0 & 0 \\ 0 & U_{21} & 0 & 0 & U_{22} & 0 & 0 & U_{23} & 0 \\ 0 & 0 & U_{21} & 0 & 0 & U_{22} & 0 & 0 & U_{23} \\ U_{31} & 0 & 0 & U_{32} & 0 & 0 & U_{33} & 0 & 0 \\ 0 & U_{31} & 0 & 0 & U_{32} & 0 & 0 & U_{33} & 0 \\ 0 & 0 & U_{31} & 0 & 0 & U_{32} & 0 & 0 & U_{33} \end{pmatrix}.$$

It's immediate to see that, expressing M as a 9-vector, $U'(M) = UM$ and also that U' is orthogonal, since $U'U'^t = Id$. So, what we are going to do is to substitute U by U' on the distance expression:

$$d(UM, UN)^2 = d(U'(M), U'(N))^2 = \|U'(M) - U'(N)\|^2 = \|M - N\|^2 = d(M, N).$$

As we wanted to see. □

Using this lemma, our problem is reduced in finding the matrix $\tilde{F}$ minimizing $d(\tilde{F}, D')^2$ under the condition $det(\tilde{F}) = 0$. Since it's an optimization problem, we will make use of the Lagrange multipliers. Let $\tilde{F} = (a, b, c, d, e, f, g, h, i)$ be a general vector in $\mathbb{R}^9$. Then, we must minimize:

$$d(\tilde{F}) = (a - \lambda_1)^2 + b^2 + c^2 + d^2 + (e - \lambda_2)^2 + g^2 + h^2 + (i - \epsilon)^2.$$

Under the condition:

$$aei + bfg + dhc - gec - ahf - bdi = 0.$$

Deriving $d(\tilde{F})$ by each variable and equaling 0, we get the conditions:

$$\begin{array}{ccc} 2(a - \lambda_1) = 0 & 2b = 0 & 2c = 0 \\ 2(e - \lambda_2) = 0 & 2d = 0 & 2f = 0. \\ 2(i - \epsilon) = 0 & 2h = 0 & 2g = 0 \end{array}$$

Applying the singular condition, we get that either of a, e or i must be 0. So, we have three possibilities for $\tilde{F}$:

1. $\tilde{F}_1 = (0, 0, 0, 0, \lambda_2, 0, 0, 0\epsilon)$, $d(\tilde{F}_1) = \lambda_1^2$.
2. $\tilde{F}_2 = (\lambda_1, 0, 0, 0, 0, 0, 0, 0, \epsilon)$, $d(\tilde{F}_2) = \lambda_2^2$.
3. $\tilde{F}_3 = (\lambda_1, 0, 0, 0, \lambda_2, 0, 0, 0, 0)$, $d(\tilde{F}_3) = \epsilon^2$.

So, the one which minimizes $d(\tilde{F})$ is, by hypothesis, $\tilde{F}_3$, which corresponds to the matrix $diag(\lambda_1, \lambda_2, 0)$, as we wanted to prove. □

So, with the singularity issue managed, we can now introduce the 8-point algorithm:

1. Normalize data: Apply the corresponding T transformation matrix to each image $\tilde{x}_i = Tx_i$, $\tilde{x'} = T'x'_i$, in order to normalize the image coordinates.
2. Compute the matrix $\tilde{F}'$ satisfying $\tilde{x'}_i \tilde{F}' \tilde{x}_i = 0 \forall i$:
   - Solve the linear system given by the correspondences $\tilde{x} \leftrightarrow \tilde{x'}$ in order to obtain a matrix $\tilde{F}$, which will usually be regular.
   - Enforce the singularity constrain replacing $\tilde{F}$ by $\tilde{F}'$ using Lemma 3.3.0.8.
3. Denormalize data: Set $F = T'\tilde{F}'T$. F is the fundamental matrix for the correspondences $x_i \leftrightarrow x'_i$, corresponding to the original data.

These are not the only methods for obtaining the fundamental matrix. Iterative oriented algorithms such as RANSAC are also used to carry out this task. Each method has it's own advantages and disadvantages. The 8-point algorithm is, maybe, the most efficient one, although it may not be the most precise. The Gold-Standard algorithm gives an accurate solutions, although it needs and initial approximation, such as the matrix given by the 8-Point Algorithm, in order to proceed. The RANSAC algorithm gives an approximation in every iteration, which can or not be better than the one given in the previous iteration, so when to decide that an approximation is good enough becomes an issue. External information can be used: the knowledge of the motion between the two cameras, geometric entities on images, such as planes or quadrics, give accuracy on the correspondences. Correspondences also affect the compute, a correspondence near an epipole will be very susceptible to errors, since an error on the measurement or in the coordinates may affect the epipolar line, and difficult the localization of the epipole.

3.4 The Essential Matrix

In this section, we will see how K helps to limit the ambiguity of the pair of cameras determined by a general F, and in the next we will see its importance in computing the reconstruction, allowing metric properties to be obtained. It will be proved that, the knowledge of both calibration matrices reduces from a projective transformation to only four possibilities the ambiguity of the possible pairs of cameras. This fact, along with what will be seen on the reconstruction chapter motivates the computation of K.
So, let's suppose that the calibration matrices are both known. Under this circumstance, we may introduce what are called normalized coordinates as follows:
A general camera matrix decomposes as $P = K(R|t)$, so a general point X in space has as image $x = PX$. Since K is known, we can define $\tilde{x} = K^{-1}x$. Then, $\tilde{x}$ is the image of the point X by the camera $\tilde{P} = (R|t)$, which has Id as calibration matrix. $\tilde{P} = K^{-1}P$ is called a normalized camera, the effect of the known calibration camera having been removed.

Now, let's consider a pair of normalized cameras $P = (Id|0), P' = (R|t)$. By Lemma 3.2.0.2., we know the expression that the fundamental matrix would have under this circumstances.

$$F = [t]_x R.$$

Definition 3.4.0.1. *The Essential matrix E is defined as the fundamental matrix corresponding to a normalized pair of cameras. It's given by the equation:*

$$\tilde{x'}^t E \tilde{x}.$$

Applying Lemma 3.2.0.3., taking K and K' as homography, we define the essential matrix in terms of the fundamental matrix and the calibration matrix as:

$$E = K'^t F K.$$

The first effect that the known calibration provides is reducing the degrees of freedom E has. Since only R and t are unknown, and each of this has 3 degrees of freedom (the three degrees of R come from the two parameters for the axis plus the angle), we fall from 9 to 6 degrees of freedom, and taking in account that there is a scale ambiguity, we end having only 5 degrees of freedom. This is translated into new properties that we are now going to discuss. First of all, we will give an algebraic characterization of the essential matrix.

Theorem 3.4.0.1. *A 3×3 matrix is an essential matrix if and only if two of its singular values are equal, and the third one is zero.*

Proof. Since E is a fundamental matrix, it may be decomposed as $E = [t]_x R = SR$, where S is a skew-symmetric matrix. We will use the next lemma to carry this prove out.

Lemma 3.4.0.1. *Let S be a 3×3 skew-symmetric matrix. Then, S may be decomposed as $S = kUZU^t$, where k is a scalar, U an orthogonal matrix. and $Z = [(0,0,1)]_x$*

Proof. Given S, we can divide it by a convenient scalar in order to express it as $S = [t]_x$, where t is a unity vector. We define $u_3 = t$, and take $u_1 \in < t >^{\perp}$, a unit vector, and $u_2 = -[t]_x u_1 = t \vee u_1$. Then, $[t]_x u_2 = t \vee (u_1 \vee t) = u_1$, thus, the application matrix $[t]_x$ expressed in the base $\{u_1, u_2, u_3\}$ is:

$$Z = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

By construction, $\{u_1, u_2, u_3\}$ is an orthogonal basis. So, we have constructed the decomposition:

$$[v]_x = kUZU^t.$$

Where U is an orthogonal matrix and k is a scalar. □

Now, we take the following matrix:

$$W = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Which is an orthogonal matrix. Applying the lemma to S, we have its decomposition as $S = kUZU^t$, where U is orthogonal and Z the one from the lemma. W satisfies $Z = diag(1,1,0)W$ up to sign, then $S = Udiag(1,1,0)WU^t$ up to scale, and $E = Udiag(1,1,0)(WU^tR)$, which is its SVD, with two equal singular values and the other zero, as required. To see the converse, we have only to apply this construction to a matrix with two singular values equal and factorize it as SR. □

What we want is to reduce the possible choices of P'. To do that, our goal is to constrain the possible factorizations of S and R from the decomposition of E. This way, we may be able to get geometrical information of P'. The following result is aiming to that direction, we will see that S has only one possible decomposition, while R can have two choices.

Proposition 3.4.0.1. *Suppose that the SVD of E is $Udiag(1,1,0)V^t$. Using the notation of the previous proposition, there are two possible factorizations, up to sign, $E = SR$ as follows:*

$$S = UZU^t,\ R = UWV^t \text{ or } UX^tV^t.$$

Proof. In both cases, the factorization is valid. This can be easily seen by doing the product:

- $R = UWV^t \rightarrow SR = UZU^tUWV^t = UZWV^t = Udiag(1,1,0)V^t = E.$

- $R = UW^tV^t \rightarrow SR = UZU^tUW^tV^t = UZW^tV^t = -Udiag(1,1,0)V^t = -E.$

Thus, we have only to see that there are no more possible factorizations.
S is determined as $S = UZU^t$, hence we have only to focus on R, which may be written as UXV^t, where X is a rotation matrix. Then:

$$E = Udiag(1,1,0)V^t = SR = (UZU^t)(UXV^t) = U(ZX)V^t.$$

Therefore, (ZX) must be $diag(1,1,0)$. Since X is a rotation matrix, we have:

$$ZX = Z \begin{pmatrix} \alpha & -\beta & 0 \\ \beta & \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} = diag(1,1,0).$$

Then:

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha & -\beta \\ \beta & \alpha \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

So, $\alpha = 0$, $\beta = 1$. It follows $X = W$ or $X = W^t$ (since it's up to sign), as we wanted to see. □

Taking in account that $E = [t]_x R$, this result gives us a way to get $[t]_x$, thus t can be determined (up to scale). Since $St = 0$, it follows that $t = U(0,0,1) = u_3$. However, the sign of E, and in consequent, the sign of t, cannot be determined. Given this fact, plus the ambiguity on the choice in R, we get the four possible pairs of cameras.

Corollary 3.4.0.1. *Let $E = Udiag(1,1,0)V^t$ be an essential matrix, $P = (Id|0)$ the first camera matrix determined by E. Then, P' can have the following forms:*

$$P' = (UWV^t| \pm u_3) \text{ and } P' = (UW^tV^t| \pm u_3).$$

The four possible choices are represented on the following figure,

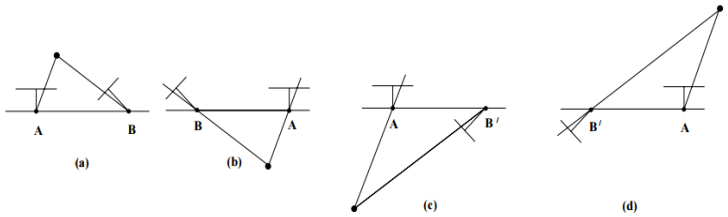

Figure 5: The four possible camera choices given by the essential matrix.

b) and d) have the baseline reversed comparing to a) and c); this reflects the sign ambiguity. In each case, there's a 90°rotation of camera B. Only in a) the reconstructed point is in front of both cameras.
[6, chapter 9, pg. 260]

4 Reconstruction

In this chapter, we will see how the tools that we have been developing along this work are joined in order to get the reconstruction.

The procedure for obtaining the estimated $3D$ point for a pair of correspondent points is quite straightforward. Our goal is to triangulate the position of the searched point as follows:
Let's suppose we have two images and its fundamental matrix F. We can obtain the pair of cameras P, P' corresponding to F using the results seen on the last section. Let x, x' be two points in the images such that the epipolar constrain is satisfied, i.e. $xFx' = 0$. That means that x lies on the epipolar line defined by x', while x' lies on the one defined by x. Putting this together, we have that x and x' lie on the same epipolar plane and in consequence, the rays defined by x and x' with their respective camera center lie on the plane, hence they intersect in one point X. This point X is the sought point, from which x and x' are image. This process, known as triangulation, can be used to obtain an estimation of the $3D$ position for all the correspondent points, and this set of estimated $3D$ point defines what is called a point cloud, which describes the external surface of the imaged scene.

4.1 Stratified Reconstruction

In this construction, only the fundamental matrix is needed, but what can we gain if we know the infinite plane position? And the calibration matrix? In this sections, we will see the limitation of the reconstruction obtained from uncalibrated cameras, and which role play the concepts we have been developing on the last two chapters upon refining the final reconstruction. Starting from only the fundamental matrix, step by step we will see which constrains are added when camera information is gained. We will suppose that a real reconstruction exists, with the exact measurements, orientation, situation as the original set of world points, and will measure the ambiguity of the obtained reconstruction in terms of its difference towards the real one.
We will use this pair of images as model to see how reconstruction is refined as further information is becoming known.

Figure 6: Model images

Extracted from [6, chapter 10, pg. 267].

4.1.1 Projective Reconstruction

In this scenario, we know nothing about the calibration of the cameras nor they relative position, we have only the fundamental matrix and the set of correspondences. Under this circumstances, we have a projective ambiguity coming from the camera determination. This transformation maps the obtained reconstruction to the real one. This implies that, for example, parallel lines on the real reconstruction may not be parallel on the obtained one, angles, distances may also differ. Only projective invariants, as the cross ratio and intersections are preserved.

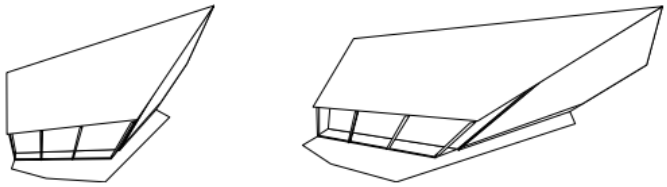

Figure 7:

Two possibles Projective reconstructions [6, chapter 10, pg. 267].

4.1.2 Affine Reconstruction

Having an affine reconstruction implies that affine properties, such as parallelism, mid point of two points and centroids, on the real reconstruction can be found on the obtained one. Starting from a projective reconstruction, affine refinement can be obtained by locating the infinite plane position on the projective reconstruction and finding the transformation that maps its coordinates to the canonical coordinates found on the real reconstruction. If the plane coordinates are the same, the ambiguity between the obtained reconstruction and the real one is defined by a transformation that fixes the infinite plane, in other words, an affinity.

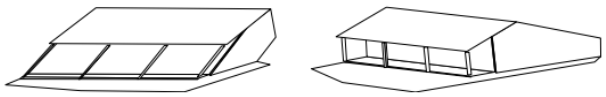

Figure 8:

Affine Reconstruction [6, chapter 10, pg. 270].

4.1.3 Metric Reconstruction

If the image of the absolute conic ω is known in either of the images, one can refine the affine reconstruction up to a metric reconstruction. This type of reconstruction shares with the real reconstruction metric information, such as angles, ratios of

lengths and areas. The idea to upgrade an affine reconstruction up to a metric one is to obtain the absolute conic from it's known image (called the IAC), and find the transformation that maps the conic to the conic on the real reconstruction. By doing this, the ambiguity between the two reconstructions will be a transformation that keeps Ω_∞ invariant.

Proposition 4.1.3.1. *Let H be a projective transformation. Ω_∞ is fixed under H if an only if H is a similarity.*

Proof. Since Ω_∞ lies on π_∞, if H fixes Ω_∞, H must also fix π_∞, hence H must be an affine transformation:

$$H = \begin{pmatrix} A & t \\ 0 & 1 \end{pmatrix}.$$

Since the image of Ω_∞ will lie on π_∞, we can restrict H to π_∞, $H_{|\pi_\infty} = A$. Then the image of Ω_∞ under $H_{|\pi_\infty}$ is given by the expression $H_{|\pi_\infty}(\Omega_\infty) = A^{-t}IA^{-1}$. Assuming Ω_∞ is fixed under $H_{|\pi_\infty}$, we have $I = A^{-t}IA{-}1$, so $A^tA = I$. Then A is orthogonal, hence H is a similarity.
On the other hand, if H is a similarity, we immediately get $H_{|\pi_\infty}(\Omega_\infty) = A^{-t}IA{-}1 = AA^t = I$. □

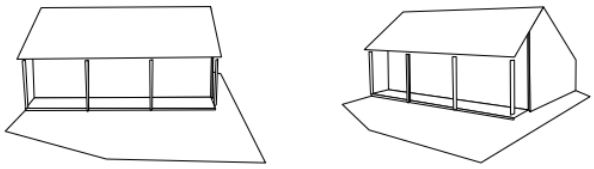

Figure 9:

Metric Reconstruction [6, chapter 10, pg. 274].

In conclusion, we have a similarity ambiguity. This means that the obtained reconstruction will differ from the real one by a rotation or a reflection. This stage is the final stratum, we can go no further in refining the reconstruction.

So, metric reconstruction requires the knowledge of, at least on one image, the IAC. In other words, it requires the calibration of at least one camera. That implies that, if both calibration are known, metric reconstruction can be done directly, with no need of previous stratum. Computing the essential matrix E of the two images, the two cameras can be determined, and triangulating the points position will lead us to four possible point clouds. Discarding the ones that locate the points behind the cameras, we obtain the metric reconstruction.

4.2 Epipolar Correction

As in the case of the fundamental matrix, the presence of errors cannot be obviated. In the case of triangulation, errors on the image coordinates may cause that the obtained rays by back projecting the image points do not intersect, so no $3D$ point X is obtained from $x \leftrightarrow x'$. This fact has also implication on the epipolar constrain, since the no existence of such point implies that epipolar lines are not coplanar, i.e. $xFx' \neq 0$. In order to deal with errors, estimation methods are introduced with the aim of locating the most suitable position for the point X to be.

In this section, we will suppose that either the fundamental or the essential matrices have been precisely obtained, so error is only found in image coordinates. Our objective is estimating the point $\tilde{X}$, satisfying $\tilde{x} = P\tilde{X}$ and $\tilde{x'} = P'\tilde{X}$ for the given P and P', from the measured points x, x'. First of all, we need a model for the error measurement, which is usually assumed to be a Gaussian error model (we will not enter in why this assumption is made). Thus, we suppose that image errors obey a $(0, \sigma^2)$ Gaussian distribution. Assuming that exists $\{\tilde{X}_i\}$ such that, if there is no error, $\tilde{x_i} = P\tilde{X_i}\ \forall i$, under the set Gaussian model, a general image point x_i will be given by the equation $x_i = \tilde{x_i} + Ex$, with Ex obeying a Gaussian distribution $(0, \sigma^2)$. If errors in the measurements are independent, the probability density function of each point is:

$$P(x_i) = \left(\frac{1}{2\pi\sigma^2}\right) e^{-d(x_i,\tilde{x_i})^2/(2\sigma^2)}.$$

Where d is the euclidean distance.
This can be applied in the case where the two images are affected by measurement errors. If the true correspondence are given by the points $\tilde{x_i} \leftrightarrow \tilde{x'_i}$, then the likelihood function is given by the expressions:

$$P(\{x_i \leftrightarrow x'_i\}) = \prod_i \left(\frac{1}{2\pi\sigma^2}\right) e^{-(d(x_i,\tilde{x_i})^2 + d(x'_i,\tilde{x'_i})^2)/(2\sigma^2)}.$$

Taking logarithm, we have:

$$log(P(\{x_i \leftrightarrow x'_i\})) = -\frac{1}{2\sigma^2} \sum_i d(x_i, \tilde{x_i})^2 + d(x'_i, \tilde{x'_i})^2 + cnt.$$

Thus the maximum of the function is achieved upon minimizing:

$$\sum_i d(x_i, \tilde{x_i})^2 + d(x'_i, \tilde{x'_i})^2.$$

Since this expression is given by a sum of squared factors, it is necessary to minimize each factor in order to reach the minimum. So, we define the cost function:

$$C(x, x') = d(x, \tilde{x})^2 + d(x', \tilde{x'})^2.$$

Where $\tilde{x}$, $\tilde{x'}$ satisfy $\tilde{x'}^t F\tilde{x} = 0$. Since these points maximize the likelihood function, they are the maximum likelihood estimators for the true image correspondences, and allow us to compute $\tilde{X}$ from triangulation, since the epipolar constrain is satisfied.
Therefore, our problem has become estimating $\tilde{x}$, $\tilde{x'}$. First of all, we will reformulate the approach in order to turn it to a one parameter minimization problem. This can be done as follows:
Since $\tilde{x}$ and $\tilde{x'}$ satisfy the epipolar constrain, $\tilde{x} \in F\tilde{x'}$, which we will denote as l, whilst $x' \in F\tilde{x}$, which will be denoted as l'. This is fulfilled by every pair of points in these lines, being the orthogonal projection of x and x' over them, $x_\perp$ and $x'_\perp$, the ones that minimize C. So, we can rewrite the distance expressions in terms of the epipolar lines as $d(x, \tilde{x}) = d(x, l)$, changing the minimization problem upon finding the pair of epipolar lines l, l' that minimize:

$$d(x, l)^2 + d(x', l')^2.$$

This expression might seem similar to the original one, but we have made a huge improvement, since l' is determined by points in l, hence by l. This motivates us to parametrize l by its angle t with the line parallel to the x-axis passing through the epipole e. Now, the cost function depends only on this parameter t:

$$C(t) = d(x, l(t))^2 + d(x', l'(t))^2.$$

Being this a one parameter minimizing problem, which solution can be computed as follows.
First of all, we will suppose that none of the image points is either of the epipoles, since $\tilde{X}$ cannot be found if x and x' are the correspondent epipoles, or is immediately found as the camera center if one of them is. Under this assumption, we apply a translation on each image, in order to express x and x' as the origin $(0, 0, 1)$ of the respective coordinate frame:

$$T = \begin{pmatrix} 1 & 0 & -x \\ 0 & 1 & -y \\ 0 & 0 & 1 \end{pmatrix} \; T' = \begin{pmatrix} 1 & 0 & -x' \\ 0 & 1 & -y' \\ 0 & 0 & 1 \end{pmatrix}.$$

So, F will be changed by $T'^{-1} F T^{-1}$. The epipoles obtained from this fundamental matrix, $e = (e_1, e_2, e_3)$, $e' = (e'_1, e'_2, e'_3)$ might be imaged to the points $(1, 0, f)$, $(1, 0, f')$ respectively:

$$R = \begin{pmatrix} \epsilon_1 & \epsilon_2 & 0 \\ -\epsilon_2 & \epsilon_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \; R' = \begin{pmatrix} \epsilon'_1 & \epsilon'_2 & 0 \\ -\epsilon'_2 & \epsilon'_1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Where $\epsilon = (\epsilon_1, \epsilon_2, \epsilon_3) = \lambda e$, $\lambda = \dfrac{1}{\sqrt{e_1^2 + e_2^2}}$ such that $\epsilon_1^2 + \epsilon_2^2 = \lambda^2 e_1^2 + \lambda^2 e_2^2 = 1$, and similar for ϵ'. Hence, now F will be changed by $R'TR$.

A similar result as Lemma 3.3.0.9. show that these transformation do not affect the distance expression, therefore the minimization problem is unchanged. In this case, since $F(1,0,f)^t = (1,0,f')F = 0$, the fundamental matrix has the form

$$F = \begin{pmatrix} ff'd & -f'c & -f'd \\ -fb & a & b \\ -fd & c & d \end{pmatrix}.$$

The line $l(t)$ is generated by the point $(0,t,1)$ and the epipole $(1,0,f)$, so it's coordinates will be given by the cross product $(0,t,1)\times(1,0,f) = (tf,1,-t)$. Then, we can give the concrete expression from $d(x,l(t))$ with the usual point to line distance equation:

$$d(x,l(t))^2 = \|l(t)x\|^2 = \frac{t^2}{1+(tf)^2}.$$

Obtaining $l' = F(0,t,1)^t = (-f'(ct+d), at+b, ct+d)^t$, we can get the correspondent expression for the distance as:

$$d(x,l'(t))^2 = \|l'(t)x\|^2 = \frac{(ct+d)^2}{(at+b)^2+f'^2(ct+d)^2}.$$

Now, we define $S(t)$ as the sum of both distance expressions:

$$S(t) = \frac{t^2}{1+(tf)^2} + \frac{(ct+d)^2}{(at+b)^2+f'^2(ct+d)^2}.$$

The minimum of this function will give us the parameter t that defines the epipolar lines which minimize the error on the image coordinates, hence the points $\tilde{x}$ and $\tilde{x'}$ might be computed and $\tilde{X}$ finally obtained. So, we will proceed by deriving $S(t)$ and forcing it to be 0. The derivative is given by the expression:

$$S'(t) = \frac{2t}{(1+f^2+t^2)^2} + \frac{2(ad-bc)(at+b)(ct+d)}{((at+b)^2+f'^2(ct+d)^2)^2}.$$

This expresson becomes 0 if an only if the numerator is 0, so taking common denominator an equating it to 0, we obtain:

$$g(t) = t((at+b)^2+f'^2(ct+d)^2)^2 - (ad-bc)(1+f^2t^2)^2(at+b)(ct+d) = 0.$$

$g(t)$ is a 6-degree polynomial, so it can have up to 6 real roots, corresponding to 3 minimums and 3 maximums of $S(t)$. Evaluating the real roots on $S(t)$ and on the asymptotic value $t\to\infty$, the absolute minimum may be obtained, so $\tilde{x}$ and $\tilde{x'}$ can now be computed as the closest points in a line from the origin (for a general line (α,β,γ), the closest point to the origin is $(-\alpha\gamma,-\beta\gamma,\alpha^2+\beta^2)$). Nevertheless, we still have to undo the transformations that have been done, so the points on the original coordinate frame will be given as $T^{-1}R^t\tilde{x}$ and $T'^{-1}R'^t\tilde{x'}$.

4.3 The Linear Triangulation Method

With the estimated $\tilde{x}$ and $\tilde{x'}$, the point $\tilde{X}$ can finally be computed. Here we give a general method in order to do it. The main idea is to combine the expressions $\tilde{x} = P\tilde{X}$ and $\tilde{x'} = P'\tilde{X}$ in order to get constrains on the unknown $\tilde{X}$. This combination will become a system of the form $A\tilde{X} = 0$, linear in $\tilde{X}$, which will lead us to the point. The matrix A comes from joining both expressions, therefore, it can be written as:

$$A = \begin{bmatrix} P & \tilde{x} \\ P' & \tilde{x'} \end{bmatrix}.$$

The solution $\tilde{X} = 0$ is of no use, so we seek $\tilde{X}$ minimizing $\|A\tilde{X}\|$. Since $\tilde{X}$ is determined up to scale, we can suppose $\|\tilde{X}\| = 1$. Let $A = UDV^t$ be the SVD of A, we want to minimize $\|UDV^t\tilde{X}\| = \|DV^t\tilde{X}\|$, considering $\|\tilde{X}\| = \|V^t\tilde{X}\|$. We may rewrite the expression, setting $y = \|V^t\tilde{X}\|$, so our problem is to minimize $\|Dy\|$ under the condition $\|y\| = 1$. Since D is a diagonal matrix, with its lowest value on its last column, we can deduce that $y = (0, ..., 0, 1)$, hence $\tilde{X} = Vy$ is the last column of V. This process is known as the least squared solution for the system $A\tilde{X} = 0$.

5 Image Processing in Correspondent Points Location

Given a pair of images, we have seen that once the fundamental matrix is computed, the reconstruction (projective if no further information is known) can be obtained. But for having a precise estimation of the fundamental matrix, a reliable set of correspondent points is needed. This task is easily carried out by a human, since its capacity of recognizing objects almost instantly allows it to look for common regions in both images and locate the correspondences within them. However, if this task is let for a computer, this direct approach is no longer possible. Then, how can correspondent points be automatically computed? This section is focused on giving an answer to this question.

5.1 Image Representation

First of all, we will talk about how images can be seen in order for a computer to work with them. An image has implicit a resolution $n \times m$, where its dimension in pixels is reflected. This fact can be used in order to represent an image as a matrix of dimension equals to its resolution. Here, the discrete nature of the projection mapping can be clearly seen, since every pixel in the i row and j column of the image is related to the element i, j in the matrix.
Color images are usually represented with the RGB code, this can be seen as a multidimensional array $I_{m,n,3}$, where $I_{1m,n}$ is the matrix containing the red values, $I_{2m,n}$ contains the green values and $I_{3m,n}$ the blue ones. In this section we will suppose that the image is given in gray scale, i.e. it is represented as a $n \times m$ matrix, with values between 0 and 1, where 0 represents black and 1 white. This is done for time efficiency, since in the RGB model, every process are introducing must be carried out in each matrix, and the information we seek can also be obtained from the gray scale model. Nevertheless, the conversion from a pixel expressed in RGB to gray scale can be done with the following expression:

$$Y = 0.2125R + 0.7154G + 0.0721B$$

Where Y is the luminescence of the pixel and coefficients are estimated to be the perception of human eye to the respective color.

5.2 Image Filtering

This section is a very brief introduction to image filtering. We will only give a first contact with the subject, see some of the main concepts and applications that filtering brings to our project and apply them in order to locate correspondent points. Let's suppose that we have a deficient camera, with a tiny stain on the objective. Every image taken by this camera will have this stain projected on it, so will the matrix representation. Many times, this will seem to be a kind of discontinuity on the values of the matrix, for example, if the image of a white sheet is taken. So,

it is interesting to have a method for extracting information from the matrix using local information given by small groups of pixels, rather than using individual pixel values. In this section we will talk about these methods, and see how they can be used in order to extract information from the image.
We shall begin differentiating two different kind of methods: methods over the spatial domain, which means that they are based on direct manipulation of the pixels, and methods over the transform domain, where first the image is transformed, the task is carried out over the transformed image, and the inverse transformation is applied, bringing back the results to the spatial domain. We won't enter in deep into these kind of methods, since they are based on signal theory and Fourier analysis, tools that differ too much from the direction of this project. Nevertheless, they are worth mentioning, since we make use of them in order of, for example, removing noise in images, as we can see in this picture:

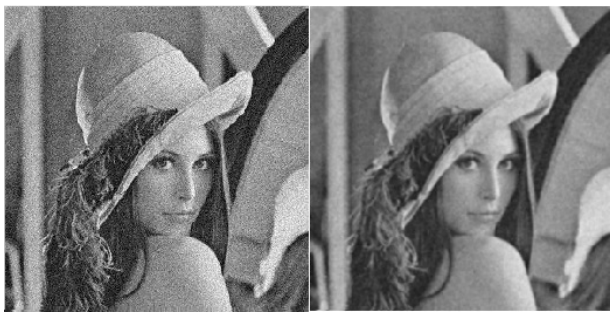

Figure 10: Noise removal using frequency domain techniques.

A gaussian filter has been used, with $\mu = 0$ and $\sigma = 1$.

So, we will talk briefly over spatial filtering, to give an intuitive idea its mechanics, before entering in gradient based methods. We will denote processes in the spatial domain as:

$$g(x, y) = T[f(x, y)].$$

Where $f(x, y)$ is the input image, $g(x, y)$ is the output image and T is an operator on f, defined over a neighborhood of the point (x, y), usually taken as a 3×3 square centered on (x, y). So, by moving the center of the square from pixel to pixel, and applying T in every neighborhood, we get the value of g at the correspondent point. If this value has the property of being shift invariant, meaning that it depends on the pattern on the neighborhood, rather than its position in the image and linear, with the usual interpretation, the procedure is known as linear filtering.
The values of the T operator used in linear image filtering is usually referred as the kernel of the filter, and the action of applying the filter is usually referred as convolution. Its output is given by the expression:

$$g(i, j) = \sum_{u,v} T_{i-u,j-v} I_{u,v}.$$

This is a particularization of the usual convolution expression, taking the input image and the filter as functions to convolve and the output image as the result of the

convolution. This particularization means that the usual properties of convolution are present, which combined with Fourier analysis, makes this operation to be very fast to compute.

5.2.1 Edges and Corners detection

Between many applications of filtering, we are focusing on the ones that allow the localization and detection of interests zones, which likely contain correspondent points. The best candidates ought to be visible from two views, keeping its properties under the camera motion, so they can be easily computed. That makes of corners a very interesting choice, since they satisfy these properties. In this section, we will study gradient-based filtering for edge and corner detection. This will allow us to built a simple algorithm for point detection.
How can an edge be characterized? They might be thought as strong changes in image intensity along a curve. This curve is what we call an edge, although not all edges are defined by this feature. Our goal is to detect points lying on the curve, called the edge points. First of all, we will introduce the image gradient. As we know, the gradient of any differentiable $2D$ function f is given by the expression:

$$\nabla f = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}).$$

Instead of a differentiable function but, we have a discrete image I, so we will net to give an approximation for $\dfrac{\partial I}{\partial x}$ and $\dfrac{\partial I}{\partial y}$. This approximation is given in terms of the finite differences, in other words:

$$I_x = \frac{\partial I}{\partial x} = \lim_{h \to 0} \frac{I(x+h, y) - I(x, y)}{h} \approx I(x+1, y) - I(x, y).$$

And similarly:

$$I_y = \frac{\partial I}{\partial y} \approx I(x, y+1) - I(x, y).$$

This definition of the gradient is very sensible big differences between pixel values, and in consequence it results to be very sensitive to noise, so smoothing the image with is almost obligatory in order not to have strong responses in wrong points. This expression of the gradient vector allows it to be represented as the following linear filter:

$$\frac{\partial I}{\partial x} = \begin{pmatrix} -1 & 1 \end{pmatrix} \quad \frac{\partial I}{\partial y} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}.$$

This representation has been improved in order to be less sensitive no noise, taking in account a hull neighborhood of pixels for the kernel:

$$\frac{\partial I}{\partial x} = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad \frac{\partial I}{\partial y} = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}.$$

These operators are known as Prewitt operators. Other operators, such as Sobel, are even less sensitive to noise. Its response to an image can be seen in the next picture:

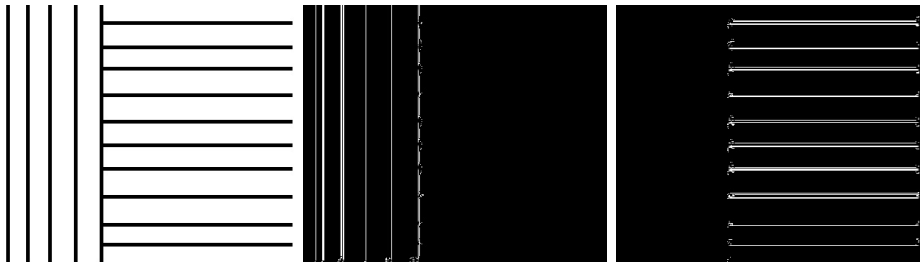

Figure 11: Gradient Based Filters Over an Image.

The first image has been prepares for the test, the second shows the effect of the horizontal Prewitt filter, the third one, shows the effect of the vertical Prewitt filter. Note that the direction of an edge at an arbitrary point (x, y) is orthogonal to the direction $\alpha(x, y) = \tan^{-1}\left(\frac{I_y}{I_x}\right)$ of the gradient at the point.

With these operators, we do detect directional edges, but only if they point in the direction of the kernel. In order to detect the whole edge curve, we proceed as follows. First of all, we locate a local maximum of the gradient value, which will play the role of the starting point of the edge curve. We know that the gradient direction is orthogonal with the edge, so we must seek the next edge point on the normal direction of the gradient. This point will be a local maximum on the gradient direction, so we can iterate the process, marking each point visited and its neighbors. Upon reaching a visited point, we will have a closed curve, that will determine our edge. In order not to have open curves, this method needs the use of two different thresholds, one for determining the start of a curve, an another lower, to keep the tracking of the edge points.
The kernel used in this method is usually the laplacian of a gaussian filter. The gaussian part of the operator gives the smoothing needed to reduce noise, and its capacity of being parametrized allows it to detect edges of different shapes, while the laplacian has the property of being isotropic (invariant to rotations), so it respond equally to changes in intensity in any directions, avoiding having to apply different filters for each direction.

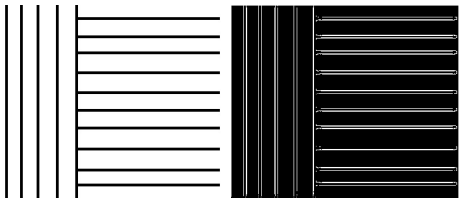

Figure 12: Improved Gradient Based Filters Over an Image.

The Canny edge detector has been applied to the original image. As we can see, all the edges pointing in different directions have been detected. Note that the vagueness on corners.

Now, we will leave edges a part and will focus in finding corners. As we said before, corners are potential candidates to be correspondent points, so having a methodology of computing them is important. A first approach in order to find a corner is locating an edge, and keeping track of it looking for a sudden change of its direction, which will mark the corner. This approach but, has an issue, since the smoothing applied in order to make the location of the edge more accurate, can make corners to disappear, as we can see on the last figure.
So, we will focus in what might make a corner characteristic in order to locate it. Intuitively, in a corner the gradient should be large, since we are on an edge, but also the its direction should suddenly change. Hence, we will try to locate corners looking at strong variations in orientations in small neighborhoods. In order to pack these information, we define the expression:

$$E(i,j) = \sum_{x,y} \omega(x,y)[I(x+i,y+j) - I(x,y)]^2.$$

Where $\omega(x,y)$ is a window operator representing the neighborhood, setting all values outside the ranges of (x,y) to 0. The factor $[I(x+i,y+j) - I(x,y)]^2$ will be large on windows with a high variety of values. These regions are of interest, since reflect a variation on the image, which could either be an edge or a corner. We can obtain a first order approximation of $I(x+i,y+j)$ using Taylor series:

$$I(x+i,y+j) = I(x,y) + iI_x(x,y) + jI_y(x,y).$$

So, for i, j smalls, $E(i,j)$ may be rewrite as:

$$E(i,j) = \sum_{x,y} [I(x,y) + iI_x(x,y) + jI_y(x,y) - I(x,y)]^2 = \sum_{x,y} i^2 I_x^2 + 2ijI_xI_y + j^2 I_y^2.$$

This can be written in terms of a matrix product:

$$E(i,j) = \begin{pmatrix} i & j \end{pmatrix} H \begin{pmatrix} i \\ j \end{pmatrix}.$$

where:

$$H = \sum \begin{pmatrix} I_x^2 & I_xI_y \\ I_yI_x & I_y^2 \end{pmatrix}.$$

This matrix can be used in order to describe the behavior of the orientation in our neighborhood. If both eigenvalues are small, then all values have to be small, meaning that we are in a neighborhood with poor variation on the gray level. If only one eigenvalue is large, our window will be situated over an edge, being the eigenvalue associated with the edge direction. On the other hand, if both eigenvalues are large, we might be over a corner.

Under this assumption, the Harris corner detector is defined, relating corners to the local maximums of the function:

$$H(x, y) = det(H) - k\Big(\frac{trace(H)}{2}\Big)^2.$$

where k is some constant, originally set to 0.5. Note that we are comparing the product of the eigenvalues with the square of the mean, hence local maximums of $H(x, y)$ correspond to both eigenvalues being large.

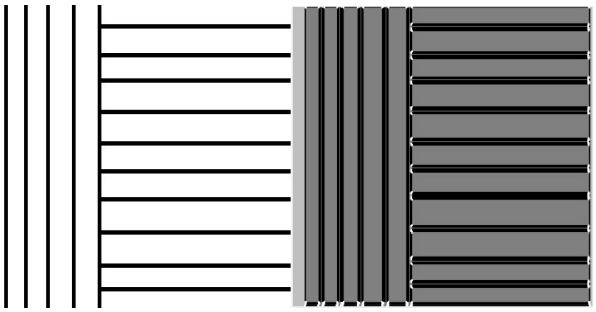

Figure 13: Detecting Corners Using Harris Algorithm

The Harris algorithm has been used over the test image. The corners have been detected and painted white.

5.2.2 Point Matching

A last tool is needed in order to build our algorithm. We have a list of possible correspondences given by Harris algorithm for corner detection, but now need to decide which point is related to which. In order to take this decision, we need a way of evaluating similarity between points, and Gabor filters will provide it. As [**?**, 1.1] introduces, these filters are obtained by combining a sinusoid with a Gaussian, so they will respond to some frequency, but only in a localized part of the signal.
Let $g(x, y, \theta, \phi)$ be the function defining a Gabor filter centered at the origin with θ as the spatial frequency and ϕ as the orientation. We can see it in terms of the following expression:

$$g(x, y, \theta, \phi) = \exp\Big(-\frac{x^2+^2}{\sigma^2}\Big)\exp(2\pi\theta i(x\cos\phi + y\sin\phi).$$

These parameters allow to set a Gabor filter to respond to a concrete feature, for example orientation. Hence, applying a prepared Gabor filter will provide information of a desired feature on our list of possible correspondences. This idea can be generalized as follows: Let $G = (G_1, ..., G_n)$, usually known as a bank of filters, a set of n Gabor filters, each filter prepared to respond to a different feature. We can define the function:

$$f : I \to \mathbb{C}^n$$
$$p \mapsto ((G_1 * I)(p), ..., (G_n * I)(p))$$

Where $*$ denotes convolution.

We can proceed intuitively, two points will be correspondent if they have similar responses to the bank of filters, i.e., if they are close in terms of the euclidean distance. This notion of closeness will be given in terms of a threshold parameter ϵ, so we have different scenarios. In the ideal one, we would have to list of possible correspondences, l_1 and l_2, of the same length, and would be able to find an ϵ such that for every point x of l_1 there is only one point y of l_2 such that $d(x, y) <= \epsilon$, and this y is truly the correspondent point to x. Sadly, this is very unlikely to happen. In the general case, a point x can have either multiple possible matches or none, the length of the two list will differ and sometimes we will assign the wrong match to a point.
Under this circumstances, there are two different approaches in order to get the matches. We can either minimize the global error or assign to each point $x \in l_1$ the closest point $y \in l_2$. Although the first approach will always give more matches than the second one, it is not recommendable to take it, since it has a high computational cost and it tends to give many wrong correspondences. So, we will take the second approach, and match the points with, usually called, brute force.

We can now give our first algorithm in order to get a set of correspondences:

Algorithm:
Input: Two images I_1 and I_2.
Output: A set of correspondences $\{x_i \leftrightarrow y_i\}$ between I_1 and I_2.

1. Get l_1, l_2 from I_1, I_2 using Harris algorithm.
2. Prepare a bank of Gabor filters G.
3. For each point in each list, compute its image under f and store it in two new lists l_1' and l_2'.
4. Match the points in l_1' and l_2' in terms of the euclidean distance with a threshold ϵ.
5. The matching in l_1' and l_2' corresponds to a matching in l_1 and l_2.

5.3 An Introduction to Feature Extractors, SIFT and BRIEF

In the last section, we talked about the difficulty in finding correspondences between images. In order of having a robust estimation of the fundamental matrix, we need a consistent set of correspondent points. Although the given algorithm is indeed a good start point, it is not reliable enough to be used in practice. For this

reason, we will use the BRIEF algorithm for point correspondences. In this section, we will give an introduction to feature extractors, in particular to SIFT as a general example, and later will deepen in BRIEF as our choice.
In this point, we assume to have a list of possible interest points given by the corner detector, and want to be able to compare such points in order to set the correspondences. We will describe the methodology as it is presented in [4, 5.4].

Our first step will consist in building a suitable neighborhood around the corner points. By doing this, we will be able to relate two points by comparing their neighborhoods, defining two points as correspondent if they neighborhoods are similar. These neighborhoods should not be affected by direct image transformations, i.e., if the image is translated, the neighborhoods should also be affected be the same translation, whereas if the image is scaled by a factor k, this scaling should also affect the neighborhood. In order to have these properties, and also benefit from smoothing, the Laplacian of a Gaussian filter is used.
Let (x, y) be the coordinates of a corner point, σ the smoothing parameter, k a scale parameter, the radius of the neighborhood is given by the expression:

$$r(x, y) = argmax_{\sigma} \nabla^2_{\sigma} I(x, y).$$

With this definition, the radius fulfills what we wanted, since now scales with the same parameter k as I. We can now build the neighborhood N as, for example, creating a circle of radius r with center (x, y). The next step for refining N is obtaining an estimation of the orientation of N. First of all, we will compute an orientation histogram $H(\theta)$ of gradient orientation within a radius kr of (x, y). We then define the orientation of the neighborhood as:

$$\theta_p = argmax_{\theta} H(\theta).$$

Then, we define the neighborhood N as $N := (x, y, r, \theta_p)$. Now we will see how we can extract features from these neighborhoods. We will begin talking about SIFT (Scale Invariant Feature Transform), which will serve as general example of how an extractor works. First of all, N is divided into an $n \times n$ grid, and each element of this grid is also divided into an $m \times m$ grid made of subcells. At the center of each subcell, an estimation of the gradient is computed from a weighted average of the gradients around the center of the cell. The magnitude of these gradient estimations are weighted by a Gaussian in distance from the center of the patch and used to create h histograms, each one corresponding to a different gradient direction. These histograms are joined into a $n \times n \times h$ vector v, which is normalized and truncated by a threshold parameter. This vector v is called a SIFT feature vector, containing information about the orientation of the gradient in the neighborhood, and as we said in the last section, allows to compute similarity between neighborhoods in terms of the euclidean distance.

This approach is very similar to what we did in the last section, computing estimations of the gradient direction and using them as values in a vector in order

to use the euclidean distance. This may be computationally expensive, since we are normalizing each vector and computing the euclidean distance in each comparison. In order to deal with this computational cost, BRIEF (Binary Robust Independent Elementary Features) was introduced. We begin computing the corner points as we did in SIFT, but instead of computing gradient histograms, we define the test τ between the image points u, v as:

$$\tau(p; u, v) = \begin{cases} 1 & \text{if } p(u) < p(v) \\ 0 & \text{otherwise.} \end{cases}$$

Where $p(u)$ (resp. $p(v)$) is the pixel intensity in a smoothed region of u (resp. v). Note that all neighborhood must be of the same size and orientation in order to have the same coordinates, so the correspondent transformations might be needed. Applying the test to a set of n pairs of points, we get a n binary string, which will be used as feature vector. Once we have obtained all the feature vectors, we proceed in computing the matching. This time but, we won't make use of the euclidean distance. Since we are working on binary strings, we can use the Hamming distance, and compare only bit by bit of each string. This operation is very efficient in terms of computability, hence this algorithm will be very useful in devices such as mobile phones and cameras, which are not provided with a powerful CPU.

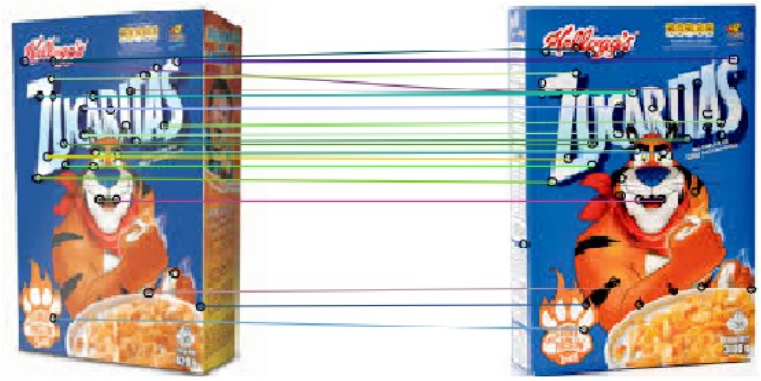

Figure 14: Matching Correspondent Points

The BRIEF algorithm has been used on this pair of images. The matches have been established and now the calculation of the fundamental matrix is available. Notice that some mismatches have occurred.

6 Implementation

In this section, we will show examples of some the topics than have been explained in this project. From the importance of blurring an image before applying certain kinds of filters on it to the plot of the epipolar lines resulting from a computed fundamental matrix. All the algorithms have been implemented with Python, making us of the numpy library for numeric calculations and the scikit-image library for image treatment.

We shall begin showing the impact that noise may cause to our results. As we have just said, this can easily be seen with the implementation of the simplest directional filter on the space domain. The implementation is very simple, just compute the difference between adjacent pixels and compare it with a threshold. If it's higher, set the corresponding pixel value to $(255, 255, 255)$, otherwise set it to $(0, 0, 0)$.

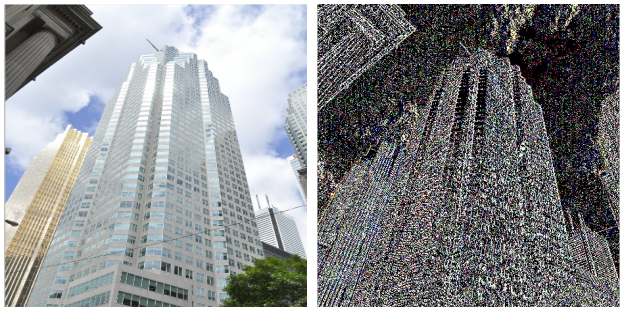

Figure 15: The effect of noise on an untreated image

The left image has been taken with a Nikon D500, a model of camera that was released on 2010. As we can see, the amount of noise that affects the output is considerable.

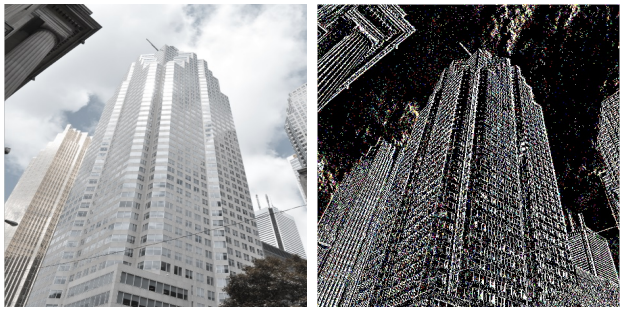

Figure 16: The effect of noise on a treated image

The left image is the result of applying a Gaussian kernel on the input image. The disparity on the pixel values has been reduced, and the output is now much more accurate.

As we can see, although the image seems clean from noise, it must be treated before carrying out our algorithms, since it may affect in a dramatic way the output result. This fact affects directly the computation of the calibration matrix, since

the corner detector used to locate the vertex of the square may not give accurate results. These facts, added to the lack of time have motivated the use of a pair of stereo images, obtained from an stereo bank, with the calibration matrix already computed. The chosen images are:

Figure 17: The pair of images we will use for now on.

Images taken from http://vision.middlebury.edu/stereo/data/2014/

6.1 General Procedure

First of all, we will obtain the fundamental matrix F corresponding to this pair of images. As we have seen on section 3.3, we need an initial set of at least eight corresponding points in order to execute the 8-point algorithm. We can obtain these correspondences using the BRIEF algorithm.

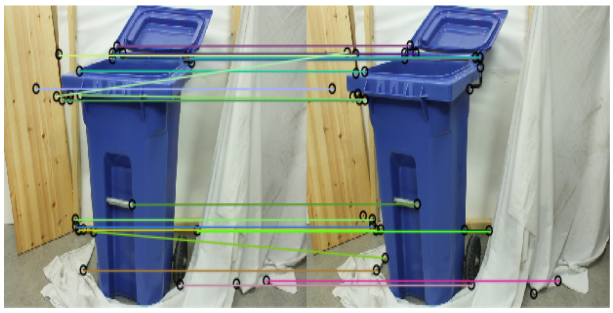

Figure 18: Corresponding points obtained by BRIEF

These are the correspondences that we will use in order to compute the fundamental matrix. Notice that there are two faulty correspondences. Since we have more than ten points, we might eliminate them from our set and use the remaining. In a full automatic methodology, we would have used an iterative algorithm like the Gold Standard algorithm from the full set in order to detect them and act before they affect the final result.

We shall now run the 8-point algorithm and get a first approximation of the fundamental matrix. We can also see the importance of normalizing the data before using SVD. The approximations we ha get is:

$$F = \begin{pmatrix} 5.99318723e-09 & -3.58949616e-06 & 1.22479712e-03 \\ 3.42188001e-06, & 6.95215185e-08 & 3.72863860e-03 \\ -1.16352337e-03 & -3.80782527e-03 & 1.64716197e-02 \end{pmatrix}.$$

And the corresponding epipolar lines on the first image:

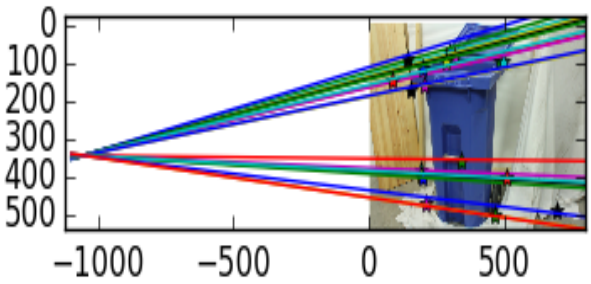

Figure 19: Epipolar lines corresponding to the equation $F^t x' = l$

Note that, thanks to the singularity enforcement, the epipole e is defined as the intersection of all epipolar lines. The plotted points are the correspondent points found by BRIEF on the first image. The origin of coordinates is taken on the top left corner of the image.

We can see that, even with the normalized data, there are some points in which the corresponding epipolar line does not pass exactly through them. We will fix this issue later, by finding more corresponding points using this estimation of F. This couldn't be done with the estimation given be the unnormalized data. In that case, the resulting matrix is:

$$\begin{pmatrix} -3.25189277e-06 & 3.77542843e-05 & -8.23638758e-03 \\ -3.69450974e-05 & -7.68805648e-06 & 6.94520615e-03 \\ 1.05995773e-02 & -2.10620074e-03 & -9.99883561e-01 \end{pmatrix}.$$

And the epipolar lines that we might get would be:

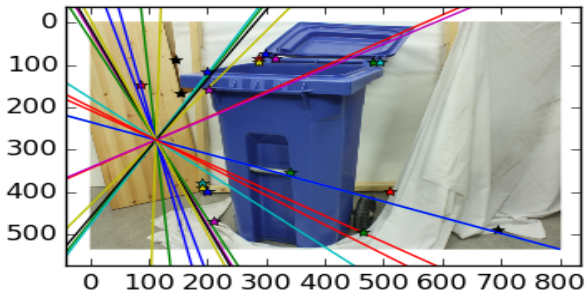

Figure 20: Epipolar lines corresponding to an unnormalized F

As we can see, the normalization of the data is almost obligatory if we want to make use of the fundamental matrix for further computations. Up to this point,

we will make use of the epipolar geometry to motivate the use of the RANSAC algorithm. Since these images have been taken using an stereo rig, we know that the two camera planes are parallel, hence the epioles must lie at infinity. In our approximation, although the epipole is quite far from the image, epipolar lines are far from being parallel. This may be the result of not having enough correspondences, or to have them in a poor distributed way over the image. To deal with this fact, we need a considerable amount of candidates to be corresponding points and a way of determining if they are truly or not. As before, we will start making use ob the Brief algorithm in order to get the possible correspondences. This time but, we will make use of a less restricting corner detector that will provide more candidates than Harris.

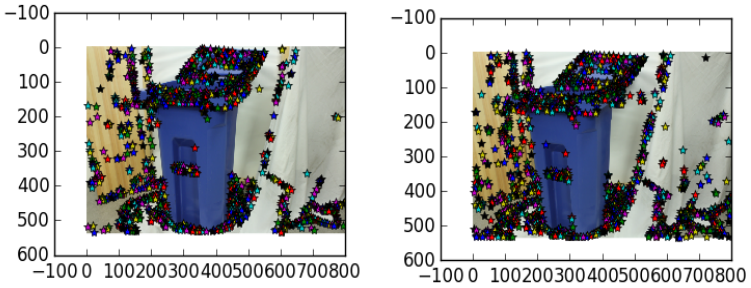

Figure 21: Points detected by the Kitchen-Rosenfeld algorithm

Note that, unlike Harris procedure, with this algorithm many detected corners do not correspond with the definition of corner that we are used to.

Now, Brief will give us the possible correspondences between the two sets of points:

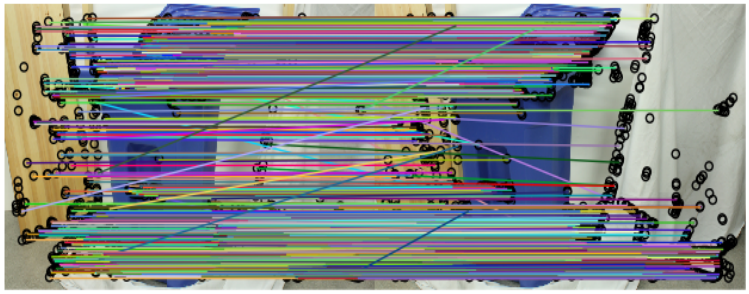

Figure 22: Matches resulting of the brief algorithm

There have been found 676 matches. We can immediately see that some of them are not correct.

This time, it is not viable to see one by one which of the matches are actually mismatches, so we will use RANSAC. In general terms, given a data set, a model that can be fitted by the data and an error measure, RANSAC (the abbreviation for RANdom SAmple Consensus) determines which of the data defines a model minimizing the error measure by taking random samples of the data, generating

the model with it and testing with the remaining data. After a certain number of iterations, the best model with the data that defines it will be the output. In our case, the model will be the normalized eight point algorithm, whilst the error measure will be given by the Sampson Error (a first order approximation of the geometric error). Applying RANSAC to our matching problem, we obtain the following estimation for the fundamental matrix:

$$\begin{pmatrix} 4.97461164e-21 & -2.59199018e-05 & 8.38552368e-03 \\ 3.10487321e-05 & 4.33960970e-06 & 4.75882516e+12 \\ -9.13146095e-03 & -4.75882516e+12 & 1.00000000e+00 \end{pmatrix}.$$

The correspondences used for generating this estimation are:

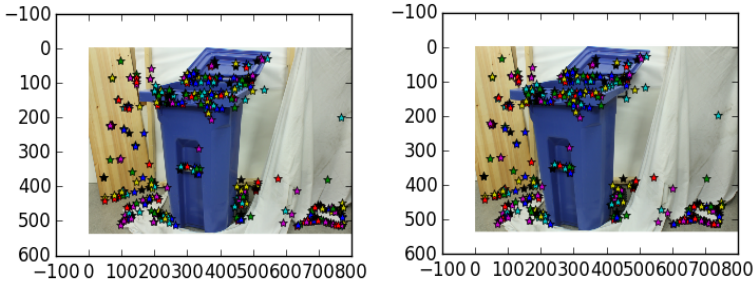

Figure 23: Matches used for the RANSAC estimation of F.

And the epipolar lines defined by this estimation of the fundamental matrix are:

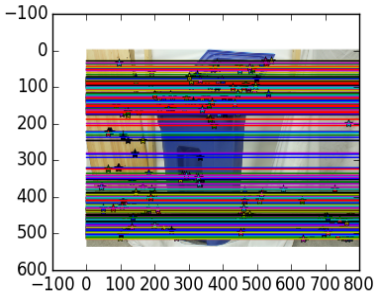

Figure 24: Epipolar lines drawn on the first image.

Note that in this case, they are indeed parallel.

This is a reliable estimation of the fundamental matrix, that we can use in order to obtain more correspondent points. The procedure will be very simple. We will make use of the epipolar constrain for a pair of correspondent points on a set of correspondences. In this case, we will use Brief over the edge points, detected by the canny algorithm. Once the possible correspondences are set, we will establish a threshold ϵ over the expression $x'^t F x$, accepting every pair of points as a true correspondence if it fulfills $x'^t F x <= \epsilon$. This will lead us to a smaller set of points,

which we will assume that are effectively correspondent. According to 3.4.0.3., there is only one of the four possible cameras that situates the 3D points in front of both cameras. Due to errors, this will no be true for all points, even for the correct camera. Therefore, we are selecting the one that has the most by triangulating all the points using 4.3, and checking the sign of the third coordinate of the resulting 3D point. Being it negative means that the point has been situated behind the camera. Once the second camera has been found, we can proceed to plot the triangulated points corresponding to it, generating the reconstruction:

Figure 25: Reconstructed Scene From the Initial Views

6.2 Observations on the Depth Levels

On the last days of the project, we came up with a surprising result concerning the distribution of the correspondent points in the image. We found out that, depending on which depth level these points are located, the reconstruction on certain objects in the scene will differ. For seeing this, we will take this pair of images:

Figure 26: Input Images

Note the difference between the deep of the bike compared to the shelf.

Following the general procedure, i.e., computing the correspondences from the corners, estimating the fundamental matrix with the RANSAC algorithm and filtering the correspondences from the canny algorithm, we produced the following reconstruction:

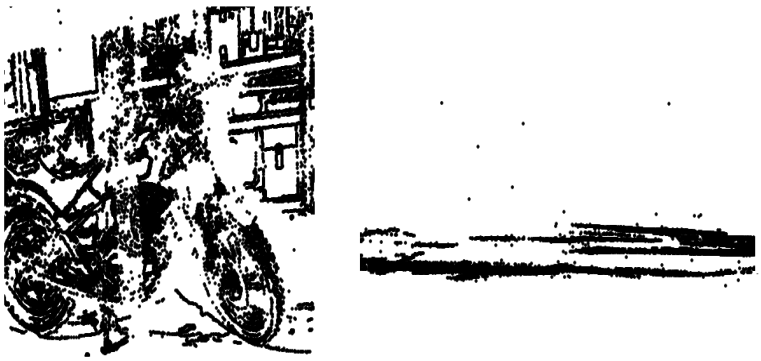

Figure 27: Reconstruction with an homogeneous distribution of points along the image.

On the first image, we can see the reconstructed scene from a frontal view, while on the second image, we see the reconstructed scene from a lateral view. Note that, although neither of the objects present a high level of deep, they are clearly separated.

As we can immediately see, this reconstruction lacks of volume, it's mainly composed by two parallel planes, representing the two principal levels of deep in the image: on the front, the plane containing the bike, while the other plane contains the shelf.
We shall now separate the two sets of points. We will do so by trying to approximate one of the planes equation, and then compute it's value for each reconstructed point.

The procedure to approximate the plane can be resumed in the following steps:

1. Pick a random point p.
2. Obtain all the points x such that $d(x, p) < \epsilon$, for a given ϵ, where d is the euclidean distance.
3. Stack all the found points in a matrix A and solve the system $AX = 0$ using least squares. The obtained X will correspond to the approximation of the plane coefficients.
4. Set a threshold $\tilde{\epsilon}$ and pick every reconstructed point x such that $X\Delta x < \tilde{\epsilon}$. This set of points will correspond to one of the deep levels.

Taking $\epsilon = 10^{-3}$, $\tilde{\epsilon} = 6.5^{-3}$, and the random point from the bike, we obtained an approximation of the points corresponding to the bike plane. An analogous procedure, with the random point from the shelf, might give the shelf plane, although its computation is not needed.
Note that these parameters have been given by observation, so these are the only parameters that need of the user in order to be set.

Figure 28: Bike separated from the shelf.

Now, taking a sample of of these points, we can recalculate the essential matrix and recompute the camera matrices. The reconstruction given from these cameras can be seen on the following images:

Figure 29: Reconstructed scene with the recalculated essential matrix.

We have gained in deep detail, even in the shelf. Clearly, objects are less similar to planes now, and the proportions of the bike are much closer to the real ones.

7 Conclusions

This project is a basic introduction to 3D reconstruction. We have seen, step by step, how starting from a simple camera model, we can deduce the existence of the fundamental matrix, and from it, the possibility of triangulating correspondent points, found automatically thanks to image processing techniques. As we have seen, it's possible to retrieve a scene of a quite surprising quality, taking in account that we have used basic tools in almost every step of the reconstruction. This process but, needs of an above average computer to be done properly. For generating these reconstructions, we have re sized all images to a size of 800×537 pixels, and although we apply brief by sectors, in some images we needed computers with more than 4GB of RAM in order to generate the matches.

We only have to look behind and see how many times is written "'it would require a whole project"' to see how many things we have left. From more advanced calibration methods, to a study on the accuracy of the fundamental matrix estimations, passing through a better implementation. What we have seen is only a little introduction into this field of computer vision.

The next natural step would be to increase the number of images to reconstruct. In this process, the fundamental matrix would be replaced by the trifocal tensor, requiring in consequence of tensor analysis in order to proceed.
Another possibility would be to improve the current reconstruction by adding colors, textures and surfaces, or by providing a GUI to ease the use of the implementation. We could also try to find a mathematical justification for the disparity in the reconstruction. The differences between the reconstruction obtained from an essential matrix coming from correspondences in the bike, with the one obtained from the essential matrix coming from the shelf might allow the computation of the distance between both deep levels, but this is just an hypothesis that might be tested.

A Additional Algorithms

Aiming to improve the reconstruction, we have developed additional algorithms as, for example, the already told algorithm for separating two planes in space. Another one, for cleaning out outlayers in a reconstruction of a single object, used in the bike reconstruction with the following results:

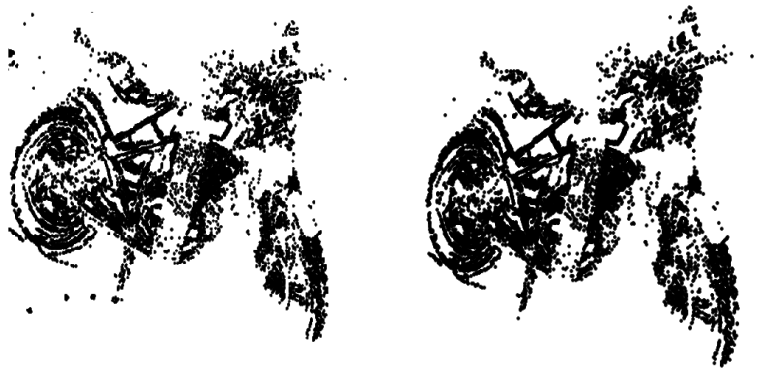

Figure 30: Result of the algorithm on the bike reconstruction.

Aiming to an algorithm for generating triangulations in a point cloud, we have developed an algorithm that computes the euclidean ball centered in a point p with an average complexity of $n \log n$. The procedure is simple. First, the data is sorted by, for example, its value on the first coordinate. Doing so, we prevent from comparing the euclidean distance from one point to all the rest for each ball we want to calculate. Instead we compare the distance with the adjacent points in the sorted list while they fulfill the desired distance with the coordinate used for the sorting. These points are the only ones that may be inside the ball.

References

[1] Calonder, M.; Lepetit, V.; Strecha, C.; Fua, P.: *Brief: Binary robust independent elementary features*, https://www.robots.ox.ac.uk/ vgg/rg/papers/CalonderLSF10.pdf Accessed on 30th March 2016, École Polytechnique Fédérale de Lausanne, 2010.

[2] Chojnacki, W.; Brooks, M.; Hengel, A.; Gawley, D.: *Revisiting Hartley's Normalized Eight-Point Algorithm*, https://cs.adelaide.edu.au/wojtek/papers/paminals2.pdf, Accessed on 3rd March 2016, University of Adelaide, 2004.

[3] Espuny, F.: Geometria de la Visió, Universitat de Barcelona, 2005.

[4] Forsyth, D; Ponce J.; Computer Vision: A Modern Approach, second edition, Prentice Hall Professional Technical Reference, 2012.

[5] Gonzalez, R.; Woods, R.: Digital Image Processing, Second Edition, Prentice Hall of India, 2003.

[6] Hartley, R.; Zisserman, A.: Multiple View Geometry in Computer Vision, Cambridge University Press, 2004.

[7] Hartley, R.: Projective reconstruction and invariants from multiple images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 16:1036-1041, October 1994.

[8] Hunter, J. D.: Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, Vol. 9, No. 3., 2007, pp. 90-95.

[9] Lamoureux, P.: *Numerical Stability of the 8-Point Algorithm*, https://www.ecse.rpi.edu/Homepages/qji/CV/8point.pdf, Accessed on 2nd March 2016, Rensselaer Polytechnic Institute, 2005.

[10] Maini, R.; Aggarwal, H.: *Study and Comparison of Various Image Edge Detection Techniques*, http://www.math.tau.ac.il/ turkel/notes/Maini.pdf, Accessed on 17th March 2016., University of Punjabi, 2007.

[11] Naranjo, J.C.: Fonaments Geomètrics de la Reconstrucció 3D, Universitat de Barcelona, Pre Published, 2016.

[12] Scharstein, D.: *Stereo datasets with ground truth*, http://vision.middlebury.edu/stereo/data/2014/, Accessed on 15th May 2016, Middlebury College, 2014.

[13] Shin Naga Prasad, V; Domke, J.: *Gabor Filter Visualization*, https://wwwold.cs.umd.edu/class/spring2005/cmsc838s/assignment-projects/gabor-f Accessed on 20th April 2016. Tech. Rep., University of Maryland, 2005.

[14] Smallwood, H.: *Projective Geometry: Perspectives from Art and Mathematics.*, http://eprints.fortlewis.edu/27/1/Projective_Geometry__Perspectives_from_Art_and_ Accessed on 23rd June 2016, Fort Lewis College, 2009.

[15] Solem, J. E.: Programming Computer Vision with Python, O'Reilly Media, 2012.

[16] Van der Walt, S.; Schönberger, J.; Nunez-Iglesias, F.; Boulogne, J.; Warner, N.; Yager, E.; Gouillart, T.: *scikit-image: image processing in Python*, http://scikit-image.org/, 2014.