MASTER THESIS

TITLE: Multiple-Camera Capture System Implementation

MASTER DEGREE: Master in Science in Telecommunication Engineering & Management

AUTHOR: Jaume Civit Rovira

DIRECTOR: Leonardo Lizcano Esteras

SUPERVISOR: Francesc Tarrés Ruiz

DATE: March 26 th 2008

Title: Multiple-Camera Capture System Implementation

Author: Jaume Civit Rovira

Director: Leonardo Lizcano Esteras

Date: March 26 th 2008

Overview

Nowadays, the consumer looks for more and more realism in what he sees through visual communications electronic devices, the change of the TV from conventional PAL resolution to High Definition or the new 3D productions are only a few examples, so why do not offer the TV in three dimensions?

To accomplish with the user requirements and be able to offer the video content, with its depth information, is necessary to take advantage of the new visualization devices (auto-stereoscopic monitors, holographic devices, etc.), and to look for methods to capture, extract and reproduce the volume information which is in background.

In this thesis, algorithms to obtain the depth information, of a scene captured from different points of view will be studied and evaluated. Moreover, this depth information will be used to synthesize new views of the scene, simulating a capture from a virtual camera. Therefore, the following topics will be studied, implemented and evaluated: capture techniques and methods to obtain the geometrical relations between different views, correspondence computing between two views and, in the final stage, how to use this data to build realistic new synthesized images.

A solution based in *Image Based Rendering will be used to* build new synthetic images. A proof of concept about a solution based in *Light Field Rendering*, where, a priori, there is no knowledge about the scene geometry. On the other side, to take some advantage from the depth information obtained before, a 3D points viewer will be implemented in order to be able to visualize a 3D representation of the scene.

This thesis is organized as follows: the first chapter introduces the fundamental concepts of projective geometry and the geometry of multiple views, required for the understanding the basis of this thesis. In chapter 2, the technique used for depth computation is described. Chapter 3 deals with the Image Based Rendering methods mentioned above. Chapter 4 describes the developed applications, whereas, in the last chapter, the main conclusions of the implemented work are presented.

Títol: Multiple-Camera Capture System Implementation

Autor: Jaume Civit Rovira

Director: Leonardo Lizcano Esteras

Data: March 26 th 2008

Resum

Actualment, el consumidor busca cada cop més realisme en el que veu en els dispositius de comunicació audiovisual, el canvi te la televisió en format PAL convencional a la Alta Definició i les noves produccions en 3D en son exemples, doncs per què no oferir la televisió en tres dimensions?

Per complir amb els requeriments de l'usuari i ser capaços d'oferir el contingut de vídeo, amb la seva informació de profunditat, és necessari aprofitar els nous dispositius de visualització (auto-estereoscòpics, dispositius hologràfics, etc.) i buscar mètodes per capturar, extreure i reproduir la informació de volum que fins ara quedava en un segon pla.

En aquesta tesi s'estudiaran algoritmes per obtenir la informació de profunditat d'un escenari a partir d'imatges preses des de diferents punts de vista, i fer servir aquesta informació per sintetitzar noves vistes de l'escena, donant més flexibilitat a l'usuari. S'estudiaran les tècniques de captura i els mètodes per obtenir les relacions geomètriques que existeixen entre les diferents vistes, el càlcul de correspondències entre dos imatges i, després, com aprofitar aquestes dades per poder generar imatges sintètiques realistes.

Un cop realitzat l'estudi previ s'implementarà una solució basada en *Image Based Rendering*, transferència a partir del tensor trifocal, que té com a objectiu aprofitar les restriccions en la geometria, de tres vistes, per generar imatges sintètiques. També es realitzarà una proba de concepte sobre una solució basada *Light Field Rendering,* en la que, a priori, no es té coneixement de la geometria de l'escena. Per altra banda, aprofitant la informació de profunditat calculada de la imatge, s'implementarà un visor per punts 3D, per tal de poder visualitzar la representació tridimensional de l'escenari.

Pel que fa l'organització del treball: el capítol 1 introdueix els conceptes bàsics de geometria projectiva, i la geometria de múltiples vistes, necessaris per a la comprensió de la resta de la tesis. En el capítol 2 s'estudia la tècnica de càlcul de profunditat utilitzada en la implementació. El capítol 3 tracta dels mètodes basats en *Image Based Rendering*, abans esmentats. El següent capítol descriu les aplicacions realitzades i a l'últim capítols es presenten les conclusions i les possibles futures línies de treball.

CONTENTS

INTRODUCTION

This work has been developed in Telefónica I+D as part of an innovation project, to study the creation of new video services for next generation networks.

The evolution of video communications requires a new approach to the user's demand, who wants more and more realistic communications which involves the necessity of transmitting 3D video content.

Due to the convergence of these factors, many techniques have been developed over the last few years.

The analysis of scenes from multiple cameras is currently a very active area in digital image processing. Two main areas should be highlighted: capturing 3D information from the scene, which consists in extracting depth information from a 3D scene, through the analysis of two or more images obtained from different viewpoints; and the generation of synthetic images for immersive environments. With the goal of providing the possibility that the user can vary his point of view, and thus provide a more realistic feeling to communications. This area is closely linked to the previous one, because it is necessary to know very precisely the geometry of the scene in order to generate synthetic views.

The project is aimed at the possible establishment of a 3D video conferencing and 3D TV services, with open research lines in the study of Tele-immersion systems. Tele-immersion is based on the concept that a user, of this system, must have the feeling of being physically in a remote location (the same room that his partner, medical center, academic ceremony, and so on). Both, 3D videoconferencing and these environments, require advanced communications systems (with substantially higher bandwidth), technologies to analyze user activity and methods to display information (pictures and sound) so that seem the most realistic possible.

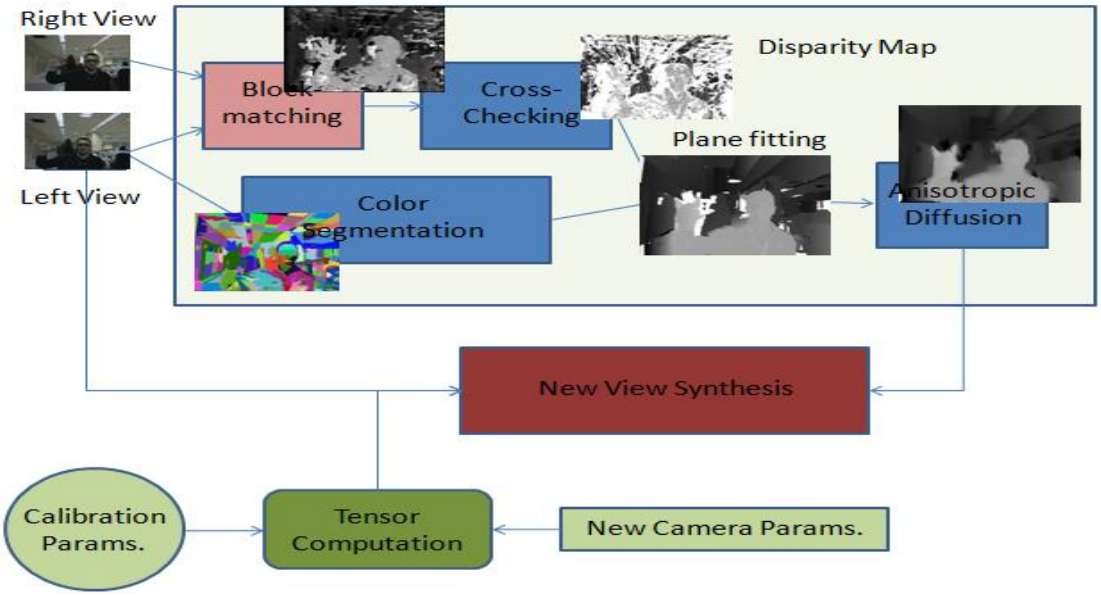
Fig. 0.1 An example of a Tele-immersion system [39].

This thesis will conduct studies aimed at the two main stages introduced above: extracting 3D information from the scene, as well as the generation of synthetic viewpoints.

The objective of this project is to create a system capable of generating intermediate images that exist between the cameras of a scenario such as the following figure, with the aim of playing a video of the scene in which it is possible to vary the user's viewpoint, in the arc between the cameras, offering a sense of continuity.

In fact, the problem of working with multiple cameras has been solved by modeling the system as groups of two cameras, which are chosen depending on the point of view taken by the user, with a much simpler geometry. This solution provides scalability to the environment, that is to say, from a basic system of two cameras can be gaining field of vision by adding more capture points.

The final application can be described with this block diagram:



To have an overview of what has been done in this thesis, following there is a brief description of each of the modules of the above diagram.

To calculate the depths of a scene two or more images, captured from different perspectives, are needed. Once the images have been captured it is necessary to find their correspondences to recover the depth of each pixel in the images. The calculation of the depths of the scene, from a stereo pair, is reduced to a geometric problem as long as the cameras are properly aligned with one another and that the defects in the lens distortion have been corrected earlier. In practice this never happens so it is necessary to call a previous module, rectification, which consists into modify the images to align them and reduce the

correspondence matching into 1D problem. This takes advantage of knowledge of the epipolar geometry, further described in this thesis.

Depth estimation is a very noisy process; therefore, finding a way to improve the quality of the disparity map is required. The first step is deprecate the wrong correspondences with the process known as cross-checking, it consists in comparing the disparity maps from both views and discarding the wrong pixels, this process will be described later. In this thesis, another solution adopted is to fit the original images into color regions and, then, model the depth associated to these regions as a planar surface which should keep the continuity in the depth map, for segments of the image with similar characteristics.

Another post-processing will be done to the disparity map in order to fuse the depth in the boundaries of similar segments, in terms of color. An anisotropic diffusion filter is included in the algorithm. It consists in a filter that keeps the original image's boundaries taking into account the gradients in the image, that is to say, the filter propagates the information of one pixel to its neighborhood until a border in the image is found.

Once the disparity map is computed and corrected, it is possible to synthesize new images corresponding to new viewpoints. To achieve this goal a technique based on point transference is proposed. The trifocal tensor contains the geometrical relations between three views (in this case two original and one virtual). Taking advantage of the geometrical constraints between views, this method allows create new synthetic images, knowing the correspondences between two images the corresponding point in the third view can be calculated. The algorithm to recover the three view geometry, and the trifocal tensor to render the new views, will be described later in the thesis.

Also, an application to show the reconstruction of the 3D position for each pixel has been implemented. The main idea is to triangulate the position of the point in the space that corresponds to a pixel in the image, taking advantage of the previous disparity computation. The result is a 3D reconstruction of the scene.

Finally, a proof of concept about techniques that do not need any information about the scene geometry will is implemented. The main idea is to parameterize the plenoptic function, which models the behavior of a ray of light coming from an object. The goal is store the light from an object in a light field, an array of images captured closely one from another, and synthesize the new view interpolating information from several images from the array. A light field viewer has been implement, using as input the data sets provided by Stanford University.

This document is organized as follows:

Chapter 1, **Multiple view geometry**, introduces the essentials on projective geometry, as well as the homogeneous notation. It also describes the pinhole camera model, as well as insights on camera calibration (concepts of lens distortion, etc.); the epipolar geometry for the architectures of two cameras, and the three view geometry, making an introduction to trifocal tensor to know the

relationships between three cameras, and the tensor notation. How final section presents a summary of architectures with more than three cameras.

In chapter 2, **3D Reconstruction**, detailing the depth extraction algorithm implemented in the final application. Each section of the chapter describes a different stage of disparity map calculation. The first section discusses the concept of epipolar rectification to simplify the process of finding disparities, explained in section 2. Then, it introduces the concept of image color segmentation and how to model the depth of these segments as a planar surface to improve the results of the initial search. It also introduces the concept of anisotropic diffusion to merge the depth of the segments resulting from the preceding paragraph. To conclude the chapter, it shows the method of triangulation, to regain the position of the 3D point, corresponding to each pixel in the image, in the 3D space.

In chapter 3, **Image Based Rendering**, Two types of techniques are explained: those which have some knowledge of the scene geometry, in particular the point transference based on the tensor trifocal, and those that do not depend, in any way, on the scene complexity, specifically Light Field Rendering. They will be shown some of the results achieved during the implementation of these algorithms.

The chapter 4, **Description of the applications**, describes the applications implemented during this thesis, to demonstrate the concepts introduced, as well as a guide detailing the use of input parameters for each of them.

In order to conclude, in chapter 5, **Conclusions and future lines**, we summarize the main results and the development of the applications, along with a proposal for future lines of investigation.

CHAPTER 1. MULTIPLE VIEW GEOMETRY

1.1. Projective Geometry Introduction

In this section, the main geometrical ideas and notations, which will be used later, are introduced. Particularly, the geometric basics of plane projective transformations are explained. This transformation models the geometric distortion introduced over a plane, when a picture of it is taken by a perspective camera. Under a perspective camera some geometrical properties are preserved, like collinearity, but not others like parallelism, for example. The projective geometry models the acquisition process of the image and, at the same time, gives an appropriate mathematical representation. Basically, all the information in this chapter is extracted from the book Multiple View Geometry in computer vision, Richard Hartley and Andrew Zisserman **[27]**. This chapter is completely necessary to understand later operations.

1.1.1. Homogeneous Notation

Next, the homogeneous notation for points and lines of the plane will be introduced.

The geometrical entities will be represented like columns by default. Taking into account this consideration a point in a plane will be represented as a column vector $x = (x, y)^T$.

A line in the plane is represented by an equation like $ax + by + c = 0$. Then, a line can be represented as a vector $(a, b, c)^T$. The correspondence between lines and vectors $(a, b, c)^T$ is not one by one, because the lines $ax + by + c = 0$ and $(ka)x + (kb)y + (kc) = 0$ are the same for every constant k different from 0. In fact, two vectors related by a global scale are considered equivalents. A type of vector equivalence, under this relation, is known as homogenous vector. The set of equivalence classes of vectors in $R^3 - (0,0,0)^T$ constitute the projective space P^2. The notation $-(0,0,0)^T$ means that the vector $(0,0,0)^T$, which does not correspond, to any line is excluded.

A point $x = (x, y)^T$ is incident with the line $l = (a, b, c)^T$ if and only if $ax + by + c = 0$. This equation can be written as a dot-product of vectors $(x, y, 1)(a, b, c)^T = (x, y, 1)l = 0$, that is to say, the point x has been represented as a 3-vector including a third coordinate 1. Must be noted that for every constant k, different from 0, will keep verifying $(kx, ky, k)l = 0$. Consequently, it is correct to consider the set of vectors $(kx, ky, k)^T$ for different k values as the representation of the point $x = (x, y)^T$ into R^2. Then the points, like the lines, are represented by homogeneous vectors. An arbitrary homogeneous vector representative of a

point has the form $x = (x1, x2, x3)^T$, representing the point $x = (x1/x3, x2/x3)^T$ in R^2.

From the above result can be deduced that a point x is over the line l only if $x^T l = 0$.

Given two lines $l = (a,b,c)^T$ and $l' = (a',b',c')^T$ their intersection must be computed. The vector $x = l' \times l$, where $\times$ means the cross-product. From the identity of the triple cross-product $l' \cdot (l \times l') = l \cdot (l \times l') = 0$ can be seen that $l^T x = l'^T x = 0$. Then if x is considered as a point representative, that point will be over the lines l and l', consequently, in the intersection of both.

In the same way, can be deduced that the vector defined by the straight line through two points x and x' is given by:

$$l = x' \times x \qquad \textbf{(1.1)}$$

1.1.2. Transformations of the projective plane

A projectivity of the plane is an invertible application from P^2 into P^2 (is that to say, 3 homogenous vectors) that applies lines into lines. This transformation are usually called homography or projectivity.

The following theorem allows to use the algebraic properties of a projectivity: A map h from P^2 into P^2 is a projectivity if only if exists a non singular matrix 3x3 H such that for every point in P^2 is represented by a vector $h(x) = Hx$ is true. Consequently, it can be stated that a projective transformation between planes is a linear transformation over homogeneous vectors of 3 components x, represented by a matrix 3x3 H, as $x' = Hx$.

The figure **Fig.1.1** shows an example of projectivity: a projection between planes produced when all the rays coming from all the points of a plane coincide in the same point (projection centre).

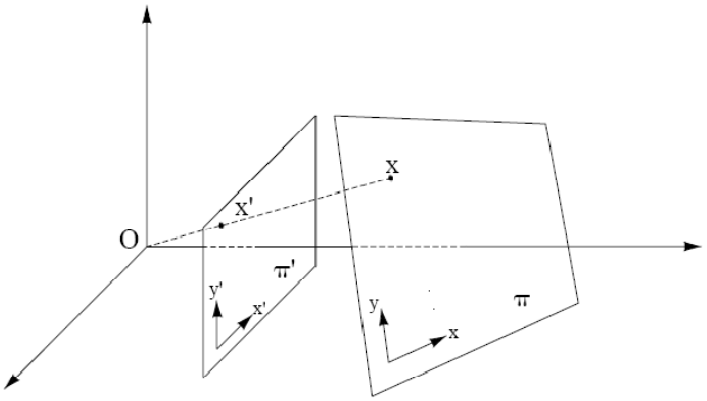


Fig. 1.1 Example of Projectivity [27].

It can be observed how this projection system preserves the lines, a line in a plane is transformed as a line into the other plane. After the above results, it is obvious that if coordinates systems are defined in each plane and the points are expressed in homogeneous coordinates; the central projection system can be expressed as $x' = Hx.$, being H an invertible 3x3 matrix. If the coordinate systems of both planes are Euclidean then the transformation is called projective and only depends of six degrees of freedom.

The shapes are distorted under the perspective effect, for example the rectangular shapes are shown as deformed quadrilaters. Usually, the parallel lines are not preserved. In the above example, the central projection has been shown as a particular case of the projective transformations and then can be expressed like a result of an invertible linear transformation. Then, it is possible undo the deformation calculating the inverse transformation and applying it to the image. The figure **Fig 1.2** shows an example:

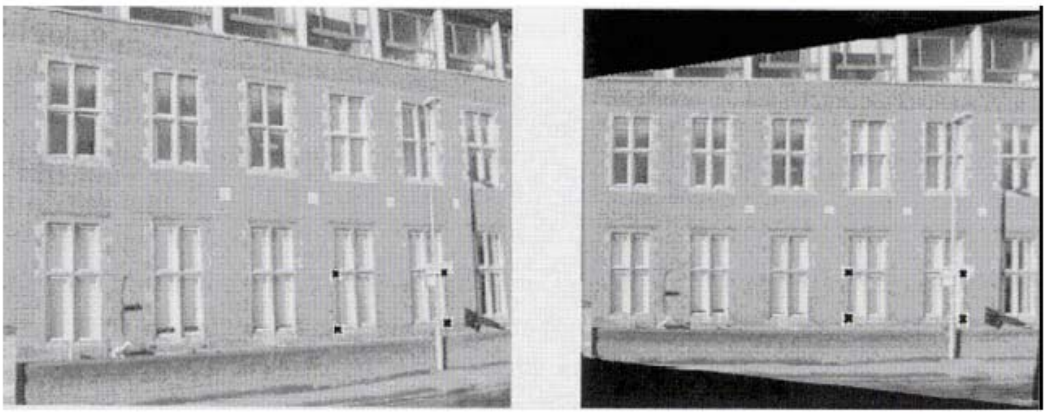


Fig. 1.2 Correction of the perspective distortion [27].

The equation that transforms a point of a plane by a projectivity is given as $x' = Hx$, now, the equation defining a straight line and a conic transformation is

needed. Knowing that a projectivity preserves the collinearity, given a straight line l which contains three points will also exist a straight line l' which will contain these three points $x_i' = Hx_i$, and its relation with l will be given by

$$l' = H^{-T} l \tag{1.2}$$

For three points x_i is verified $l^T x_i = 0$, given a singular matrix H appears

$$l^T H^{-1} H x_i = l^T H^{-1} x_i' = 0 \tag{1.3}$$

Then,

$$l'^T = l^T H^{-1} \tag{1.4}$$

The equation of a conic in homogeneous coordinates is $x^T C x = 0$. For a projective transformation of the points of a plane $x_i' = Hx_i$, the equation of the conic becomes:

$$x'^T C' x' = x^T H^T C' H x = x^T C x \tag{1.5}$$

Then transformation, due to a projectivity, of a planar conic of matrix C is another conic with matrix $C' = H^{-T} C' H^{-1}$.

1.2. The camera model

A model is the representation of geometrical and physical properties of a system. The process of obtaining these parameters is known as calibration, and allows working with a mathematical model of the camera that gives a relation between the camera coordinate system and the world coordinate system. In addition, in a stereo vision system there is a relative component that relates the two cameras. Recovering the cameras' parameters will simplify later procedures like obtaining the correspondences between views.

1.2.1. The pin-hole model

The pin-hole model is the most commonly used to convert the real 3D coordinates into the camera 2D coordinates. This is a perspective projection model where all the rays coming from certain object go through a thin hole to impact over the image sensor. Due to the lens do not have this linear behavior the pin-hole model must be corrected with a distortion value, that is to say, it must be complemented with parameters that correct its ideal behavior.

The system of reference is placed in the projection centre of the camera, fitting the system's z axis with the camera's optical axis. With this distribution, the camera plane, with coordinates (u,v), is placed at a distance equal to the focal length from the objective. The intersection between the optical axis and the focal plane is named principal point.

The projection centre of the camera is supposed constant, but it is, a priori, unknown. The image plane is, usually, placed in front of the projection centre C to have an image without inversion. **Fig. 1.3** shows a scheme of the pin-hole model.

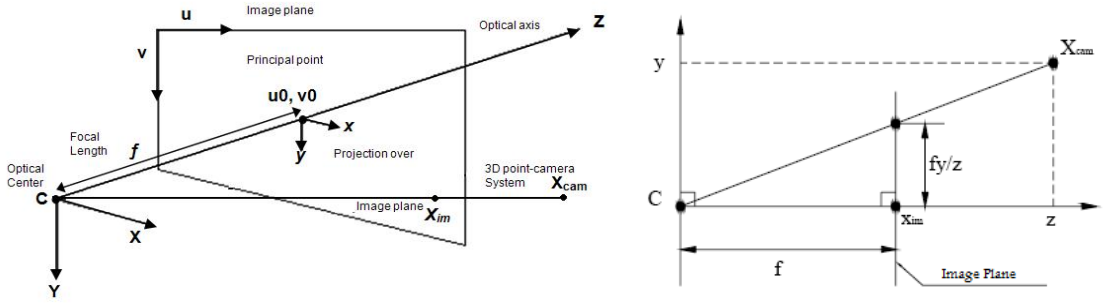


Fig. 1.3 Pin-hole model [27].

1.2.2. The pin-hole equations

A point X_{cam} expressed in the camera's system of coordinates, $[X,Y,Z]$, is projected over a point in the image plane x_{im} with coordinates (x,y) when the pin-hole model is applied. The values of the coordinates in the image plane can be calculated from the perspective projection equations for an objective with focal length f :

$$\begin{bmatrix} x \\ y \end{bmatrix} = f / Z \begin{bmatrix} X \\ Y \end{bmatrix} \qquad \textbf{(1.6)}$$

To obtain the coordinates in terms of pixel (u,v) in the image plane, the vertical and horizontal sizes of the pixels are used, s_x and s_y usually expressed in mm. This coordinates must be referenced with the top-left corner of the image as origin (adding the coordinates, in pixels, of the principal point (u_0,v_0)).

$$\begin{aligned} u &= s_x x + u_0 \\ v &= s_y y + v_0 \end{aligned} \qquad \textbf{(1.7)}$$

In this way is done the transformation between the camera's Euclidean space (image plane with distance $z=-f$) and the image sensor 2D, origin (0,0) placed at the top-left corner of the image

The tridimensional point, corresponding to the pixel (u,v), is not unique. All the points belonging to the straight line, that links the projection centre C with the initial point, are possible original points. Then, any point M from the straight line R, with coordinates (X,Y,Z), accomplish the following equality when it is projected over the pixel (u,v):

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{pmatrix} fs_x & 0 & u_0 \\ 0 & fs_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{cam} \quad \text{Where} \quad \begin{aligned} f_x &= s_x f \\ f_y &= s_y f \end{aligned} \qquad \textbf{(1.8)}$$

The sub index cam for the coordinates (X,Y,Z) means that the point M is expressed with the camera's system of reference.

If it is considered the coordinates of a point M from a certain system of reference (3D space), this must be expressed in the camera coordinates to be able applying the pin-hole equations. The transformation between both systems is expressed as a rotation and translation matrix, named extrinsic matrix (RT_{ext}). With this matrix the change of reference can be expressed as, in homogeneous coordinates:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}_{cam} = \left(\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ r_{41} & r_{42} & r_{43} & 1 \end{array}\right) \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}_W \qquad \textbf{(1.9)}$$

Usually, a projection P of a point $M(X,Y,Z)_W$ over the pixel in the image plane with coordinates $m(u,v)$ is expressed as $m \sim PM$, where P can be expressed as:

$$P = \begin{pmatrix} fs_x & 0 & c_0 & 0 \\ 0 & fs_y & r_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \left(\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{array}\right) \quad \textbf{(1.10)}$$

In this way, a final expression of perspective projection is presented, which depends on the scale factor λ. Usually written:

$$P = K[R|t] \quad \textbf{(1.11)}$$

R is the rotation matrix, t the translation vector and K is known as the camera calibration matrix.

The matrix K depends on the intrinsic parameters from the camera, specific for every lens and camera. These parameters should be constant.

The optical centre C is the unique point which projection is not defined:

$$P\begin{pmatrix} C \\ 1 \end{pmatrix} = 0 \Rightarrow C = -{P_{3x3}}^{-1} P_{:,4} \quad \textbf{(1.12)}$$

$P_{:,4}$ is the fourth column of P.

The thin lens can be modeled accurately with the pin-hole model because the thickness of these lenses is considered negligible. Nevertheless, for the thick lens case the ideal perspective projection of the pin-hole model must be corrected with other parameters that allow us to consider the real effects.

The lens distortion is introduced by means of a polynomial, in function of the position (u,v) of each pixel. The distortion is the distance between the real projection and the coordinates given by the pin-hole model.

In general, the calibration considers two types of distortion: radial and tangential.

$$u - u_0 = f_x \frac{X_{cam}}{Z_{cam}} + (u - u_0) dist\_rad + dist\_tg\_u$$
$$v - v_0 = f_x \frac{Y_{cam}}{Z_{cam}} + (v - v_0) dist\_rad + dist\_tg\_v \quad \textbf{(1.13)}$$

The radial distortion can be observed more intense when the pixel position is far from the principal point. Usually modeled with two parameters, there are some systems that are modeled with fifth order polynomials:

$$r^2 = \left(\frac{u - u_0}{f_x}\right)^2 + \left(\frac{v - v_0}{f_y}\right)^2$$
$$dist\_rad = a_1 r^2 + a_2 r^4 + a_3 r^6 + a_4 r^8 + a_5 r^{10} \quad \textbf{(1.14)}$$

The tangential distortion is used to correct the perpendicularity between the lens' optical axis and the image plane. Two parameters p_1 and p_2, produce a second order polynomial:

$$u = \left(\frac{u - u_0}{f_x}\right); v = \left(\frac{v - v_0}{f_y}\right); r^2 = u^2 + v^2$$
$$dist\_tg\_u = p_1(r^2 + 2u^2) + 2p_2 uv$$
$$dist\_tg\_v = p_2(r^2 + 2v^2) + 2p_1 uv \quad \textbf{(1.15)}$$

1.3. Two-view geometry

This section will introduce the existing relation between the points of two images captured from two different points of view.

1.3.1. Epipolar Geometry and Fundamental Matrix

The epipolar geometry describes the relation between two views in perspective from the same tridimensional scene. Basically, this is the geometry of the intersection between the image planes and the set of planes that have the baseline as axis (the baseline is the line which contains the both cameras' centers). This geometry is usually motivated for a later stereo-matching process.

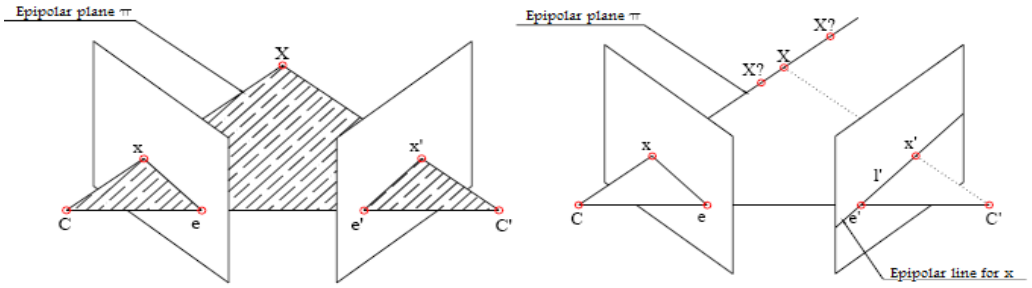


Fig. 1.4 Point correspondence geometry.

If a point X in the 3D space is projected on two views (x in the first one and x' in the second one) these points are coplanar with the camera centers, like can be seen in **Fig. 1.4**. This plane will be denoted π. The rays projected from x and x' intersect into X and belong to π. This is the most significant property for the correspondence computation. Knowing the point x, the plane π is defined by the baseline and the ray coming from x. By definition the ray corresponding to the point x' belongs to π, and then the point x' will be over the line l' defined as the intersection of the plane π with the second image plane. This line l' is the image in the second view of the line Cx. l' is the epipolar line corresponding to x.

The entities involved in the epipolar geometry are illustrated in the figure **Fig. 1.5**, in the right scheme the rotation of the epipolar planes over the baseline with the changing of X position.

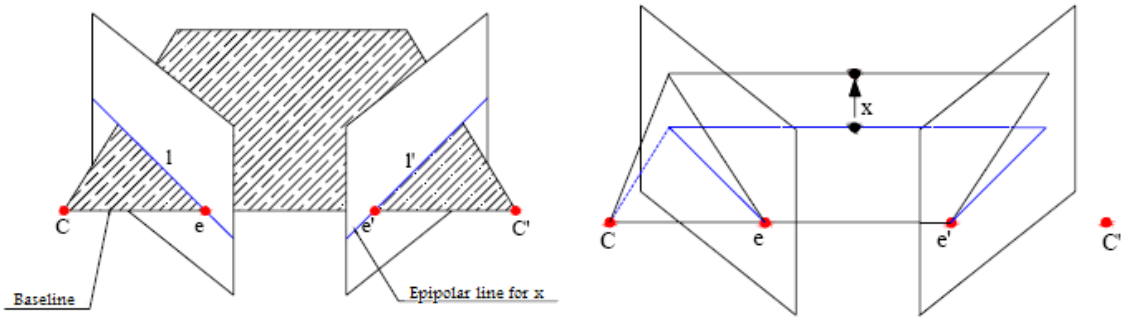


Fig. 1.5 Epipolar geometry.

Introduced concepts:

Epipole: Intersection point between the line, which connect the camera centers, and the image plane. (e and e' in figures **Fig.1.4** and **Fig.1.5**).
Epipolar plane: Plane which contains the baseline. There are a set of epipolar planes.
Epipolar line: Intersection between an epipolar plane and the image plane. The epipolar lines intersect in the epipole. An epipolar plane intersects in both image planes defining the epipolar lines.

Next, the techniques to find the map between points of the left image and the epipolar lines in the right image will be described, in order to restrict the searching of x (left image) only along the corresponding epipolar line (right image).

1.3.2. Essential and fundamental matrices

The epipolar geometry can be described by several ways taking into account the level of knowledge of the system. There are three general cases:

- If the intrinsic and extrinsic parameters are known for the both cameras, the epipolar geometry can be described in terms of projection matrices.
- If only the intrinsic parameters are known, the epipolar geometry is described through the essential matrix, and normalized coordinates.
- If no parameters are known, the fundamental matrix is required.

Every camera identifies a 3D system of reference, with the origin in the projection centre of the camera and the Z axis on the optical axis. The systems of reference of the both cameras are related by the extrinsic parameters. These define a rigid transformation in the 3D space, represented by a translation vector $t = C - C'$ and a rotation matrix R.

The equation of the epipolar line can be described with the equation of the optical ray. The right epipolar line l', corresponding to the point x of the left image, represents geometrically the projection, in the right image plane, of the optical ray through x:

$$\varsigma' x' = P' X = P' \begin{pmatrix} C \\ 1 \end{pmatrix} + \lambda P' \begin{pmatrix} P_{3x3}{}^{-1} x \\ 0 \end{pmatrix} \qquad \textbf{(1.16)}$$

Where the ray has been described as the parametric line that links the projection centre C from the left camera and the point at the infinite projected on x.

C is expressed in homogenous coordinates. ς' is the distance between X and the focal plane of the right camera and $\lambda \in \Re$

Simplifying the above equation, the result is the description of the right epipolar line:

$$\varsigma' x' = e' + \lambda P_{3x3}' P_{3x3}{}^{-1} x \qquad \textbf{(1.17)}$$

This is the equation of the line which go through the right epipole e' and a point of the right image $\tilde{x} = P_{3x3}' P_{3x3}^{-1} x$ which represents the projection of the point at the infinite placed on the optical ray from x.

1.1.2.1. The essential matrix

If the intrinsic parameters are known, it is possible to work with the normalized coordinates:

$$x \leftarrow K^{-1} x \tag{1.18}$$

Considering a pair of normalized cameras, the global system of reference can be placed on the first camera, such:

$$P = [I \mid 0] \text{ and } P' = [R \mid t] \tag{1.19}$$

With this choice, the unknown extrinsic parameters have become explicit. Now, replacing these projection matrices in the equation **(1.17)**:

$$\varsigma' x' = t + \lambda R x \tag{1.20}$$

This means that the point x' belongs to the line which links the points t and Rx. The result expressed in homogenous coordinates (the line which goes through two points is its cross-product).

$$x'^T \, t \times (Rx) = 0 \tag{1.21}$$

Taking into account that the cross-product of two vectors can be written as the result of the product of a skew-symmetric matrix and one of them, such:

$$x'^T \, [t]_\times (Rx) = 0 \tag{1.22}$$

Where

$$[t]_\times = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}_\times = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \quad \textbf{(1.23)}$$

Multiplying the matrices from the equation **(1.22)** the result is a matrix that describes the relation between the points x and x' in normalized coordinates. This matrix is named essential matrix E:

$$E = [t]_\times R \quad \textbf{(1.24)}$$

The essential matrix only provides information about the extrinsic parameters. It has rank 2, because the determinant of $[t]_\times$ is zero. It has only two degrees of freedom: a 3D rotation and a 3D translation.

It should be necessary to know the intrinsic parameters, because the values of the measured projection points is given in pixel coordinates, therefore, knowledge of the transformation between the world coordinates and the pixel coordinates is necessary.

2.1.2.1. The fundamental matrix

The fundamental matrix can be obtained in a similarly way than the essential matrix. All the camera parameters are unknown, for this reason the equation can be written in a general form **(1.19)**:

$$P = K[I \mid 0] \text{ and } P' = K'[R \mid t] \quad \textbf{(1.25)}$$

Replacing these projection matrices in the equation **(2.17)**, the result is:

$$\varsigma' x' = e' + \lambda K' R K^{-1} x \qquad \text{with} \qquad e' = K' t \quad \textbf{(1.26)}$$

This expression shows how the point x' belongs to the line which links the right epipole e' and, then is obvious that the parameter λ has the same value than ς, the depth of the point X from the left camera. This result can be written in homogeneous coordinates:

$$x'^T [e']_\times K' R K^{-1} x = 0 \quad \textbf{(1.27)}$$

The matrix

$$F = [e']_{\times} K' R K^{-1} x \qquad \textbf{(1.28)}$$

It is named Fundamental Matrix F It represents of the epipolar geometry.

$$x'^{T} F x = 0 \qquad \textbf{(1.29)}$$

It is a rank 2 homogeneous 3x3 matrix. It has only seven degrees of freedom because is defined for a scale factor and its determinant is zero. F can be defined completely from the pixel correspondences, therefore it does not need the intrinsic parameters.

For each point x in the left image, the corresponding epipolar line in the right image can be expressed as:

$$l' = Fx \qquad \textbf{(1.30)}$$

In the same way, an epipolar line in the left image corresponding to a point x' in the right image can be expressed as:

$$l = F^{T} x' \qquad \textbf{(1.31)}$$

The left epipole e is the null vector of the left product of the fundamental matrix and the right epipole e' is the null vector for the right product.

$$Fe = 0 \qquad \text{and} \qquad e'^{T} F = 0 \qquad \textbf{(1.32)}$$

The relation between the fundamental matrix and the camera matrices, K and K' is:

$$F = K'^{T} E K^{-1} \qquad \textbf{(1.33)}$$

1.4. Three-view geometry

Considering three views it is possible to group them in pairs and to get the two view relationships introduced in the last section. Using these pair wise epipolar relations, the projection of a point in the third image can be predicted from the coordinates in the first two images. This is illustrated in **Fig. 1.5**. The point in the third image is determined as the intersection of the two epipolar lines. This computation, however, is not always very well conditioned. When the point is located in the trifocal plane (i.e. the plane going through the three centers of projection), it is completely undetermined.

1.4.1. The trifocal Tensor

The trifocal tensor is like the fundamental matrix for three views. It contains all the geometrical relations, between three views, independently from the scene structure.

1.1.2.1. Incidence relations for lines

If a line in the 3D space is projected into three views, like is shown in the figure **1.6**, there are constraints which must be described.

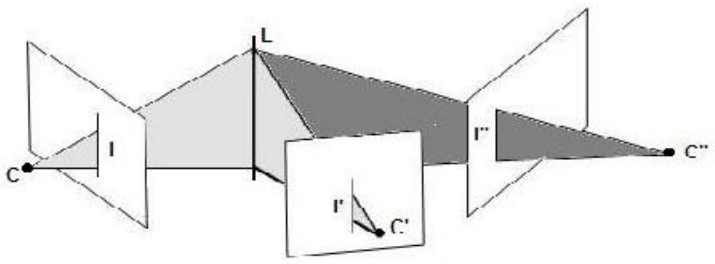


Fig. 1.6 Projection of a line, in 3D space, into three views. [27]

It is obvious that the plane defined for each of the three lines must contain the 3D line. In general, three planes in the space do not intersect in a line, therefore this fact introduce one constrain on the possible position of the three lines.

$l \leftrightarrow l' \leftrightarrow l''$ are a set of lines in correspondence, and $P = [I \mid 0], P' = [A \mid a_4]$, $P'' = [B \mid b_4]$ the three matrices associated to the three views. Later a_i and b_i will be notated as the columns of these matrices.

Calculating the equations of the planes generated by three lines in correspondence by retro-projection, using the camera matrices, the result is:

$$\pi = P^T l = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \ \pi' = P'^T l' = \begin{pmatrix} A^T l' \\ a_4^T l' \end{pmatrix}, \ \pi'' = P''^T l'' = \begin{pmatrix} B^T l'' \\ b_4^T l'' \end{pmatrix} \quad \textbf{(1.34)}$$

These three planes intersect in a line of the space, then it must be verified that the matrix 4x3 $M = [\pi, \pi', \pi'']$, defined by the vectors of the planes, should be of rank 2, and therefore will exist a linear relation between the column vector of this matrix.

The most left element of the matrix M has zero value, taking advance of this, the coefficients $\alpha = kb_4^T l''$ and $\beta = -ka_4^T l'$ for a scalar k. Applying this linear relation to the other rows, except for a constant of proportionality, the result is:

$$l = (b_4^T l'')A^T l' - (a_4^T l')B^T l'' = (l''^T b_4)A^T l' - (l'^T a_4)B^T l'' \quad \textbf{(1.35)}$$

The i-thm coordinate l_i of l can be written as:

$$l_i = l''^T (b_4 a_i^T) l' - l'^T (a_4 b_i^T) l'' = l'^T (a_i b_4^T) l'' - l'^T (a_4 b_i^T) l'' \quad \textbf{(1.36)}$$

Introducing the notation

$$T_i = a_i b_4^T - b_i a_4^T \quad \textbf{(1.37)}$$

The incidence relation can be written as:

$$l_i = l'^T T_i l'' \quad \textbf{(1.38)}$$

The set of three matrices $\{T_1, T_2, T_3\}$ form the trifocal tensor in its matrix notation.

$$l^T = l'^T [T_i] l'' = l'^T [T_1, T_2, T_3] l'' = [l'^T T_1 l'', l'^T T_2 l'', l'^T T_3 l''] \quad \textbf{(1.39)}$$

Given that the found relation only use the image coordinates, these are independent from the camera model. The tensors are independent from the cameras.

The tensor can be considered as a 3 x 3 x 3 cube operator, defined by 27 parameters in total, only 18 of which are independent due to additional nonlinear constraints.

2.1.2.1. Homographies induced by a plane

A fundamental property of the trifocal tensor is the homography between the first and third view, given a line in the second view. The following figure shows the situation

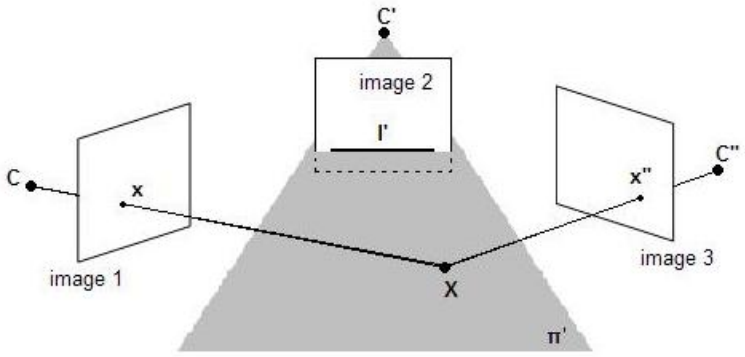


Fig. 1.7 Homography between the first and the third view [27].

A line in the second image defines the plane π' 3D by backward projection and this plane induces a homography between the first and the second view, which can be written for points and lines as:

$$x'' = Hx \quad \text{or} \quad l = H^T l'' \tag{1.40}$$

Comparing the equation **(1.38)** with the homography for three lines, $l_i = l'^T T_i l''$ and $l = H^T l''$, the columns of H can be expressed in function of the tensor matrices.

$$H = [h_1 h_2 h_3] \text{ with } h_i = T_i^T l' \tag{1.41}$$

This represents a homography between the first and third view given a line in the second view.

Incidence relations:

1. line-line-line correspondence: $l'^T [T_1, T_2, T_3] l'' = l^T$ or $(l'^T [T_1, T_2, T_3] l'')[l]_\times = 0$

2. point-line-line correspondence: $l'^T [\sum_i x^i T_i] l'' = 0$

3. point-line-point correspondence: $l'^T [\sum_i x^i T_i][x'']_\times = 0^T$

4. point-point-line correspondence: $[x']_\times [\sum_i x^i T_i] l'' = 0$

5. point-point-point correspondence $[x']_\times [\sum_i x^i T_i][x'']_\times = 0_{3x3}$

Since we are concerned mainly with points in images, further derivation of the constraints is not necessary to understanding this thesis. A complete set of derivations of the trilinear constraints for points and lines can be found at **[27]**.

1.4.2. Tensor Notation

In this section the tensor notation will be introduced to simplify later expressions and make them easier for manipulation.

Points and lines are represented as homogeneous vectors of 3 coordinates; instead they are distinguished, essentially, by the transformation equations with a change of the space base. The vectors defining line changes according the matrices of base transform, for this reason they are named covariant, instead, the coordinates of the points change with the base changing matrix inverted and transposed, and for this they are named contravariant.

The typical notation use subscripts for the covariant coordinates and superscripts for the contravariant coordinates. A point is represented by $x = (x^1, x^2, x^3)$ and a line by $l = (l_1, l_2, l_3)$. In the case of 2D matrices or bigger order can be observed by the indices of the tensor. For example, a transformation matrix between points has a contravariant dimension (columns) another covariant (rows), therefore it will be notated $A = (a_i^j)$ and the equation $x' = Ax$ is written $x'^i = \sum_j a_i^j x_j$ or, using the Einstein convention, $x'^i = a_i^j x_j$.
With this notation the trifocal tensor is written as:

$$\Im_i^{jk} = a_i^j b_4^k - a_4^j b_i^k \qquad \textbf{(1.42)}$$

The indices of the tensor, two contravariant and one covariant are established by the indices of the right expression. With this notation the incidence relation between straight lines is written as:

$$l_i = l'_j \, l''_k \, \mathfrak{I}_i^{jk} \qquad \textbf{(1.43)}$$

Hartley **[12]** proposed an algorithm for the practical computation of the quadrifocal tensor and a complete and formal mathematically based review of multiple view geometric relationships is given in **[6]**.

CHAPTER 2. 3D RECONSTRUCTION

In this chapter the transformation from 2D to 3D space will be discussed. Once calibration process finishes, is possible to proceed with methods developed to extract 3D information from sets of calibrated cameras

2.1. Dense depth estimation

With the camera calibration given for all viewpoints of a scene, the next step is proceeding with methods developed for calibrated cameras. The goal of obtaining realistic models of 3D scenes is accomplished by a dense disparity matching that estimates correspondences from the intensity images directly by exploiting additional geometrical constraints.

This chapter is organized as follows. In the first section rectification is discussed. This makes it possible to use standard stereo matching techniques on image pairs, which is the next topic. Finally, an implementation of a color based segmentation stereo matching will be shown.

2.1.1. Image pair rectification

The stereo matching problem can be solved much more efficiently if images are rectified. This step consists of transforming the images then the epipolar lines are aligned horizontally. In this case stereo matching algorithms can easily take advantage of the epipolar constraint and reduce the search space to one dimension (i.e. corresponding rows of the rectified images).

2.1.1.1. Rectification of camera matrices

Given a pair of stereo images, the epipolar rectification defines the transformation to apply at each image plane in order to make the pairs of epipolar lines being coincident and parallel to one axis of the image (usually the horizontal one). The rectified images can be understood like pictures captured by two virtual cameras, obtained rotating the original cameras and, possibly, modifying their intrinsic parameters. The rectification process is important because of the simplification in the correspondence matching problem, due to the search is restricted just to the horizontal lines.

A pair of calibrated cameras is assumed, it means the internal camera parameters are known, as well as their relative position. This assumption is not strictly necessary but simplifies the procedure. There are techniques that permit to fix pairs of weak calibrated cameras, only when some correspondent points between the two views are known (i.e. at Hartley **[29]**, Isgrò y Trucco **[28]**), but

this is not the case. The rectification process introduces some distortion in the final images due to the interpolation done after the projection of the images to the new planes.

Given the projection matrices of the real cameras P and P', the idea behind the rectification is to define two new cameras, $\hat{P}$ and $\hat{P}'$, obtained rotating the original camera over their optical centers until their planes be coplanar, which means they will contain the baseline. This ensures that the epipoles will be at the infinite and, therefore, the epipolar line will be parallel. In order to have horizontal epipolar lines, the base line must be parallel to the x axis of both cameras; next, the coincident point must have the same vertical coordinate.

The optical centers of the virtual cameras are the same than the real cameras' centers. The intrinsic parameters are the same for both virtual cameras, therefore, both resultant virtual cameras can be considered as the same camera displaced along the x axis of the system of reference.

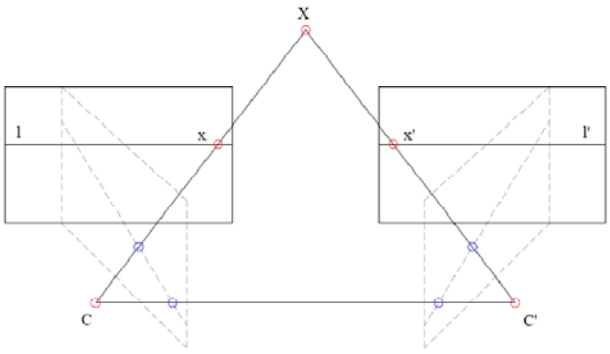


Fig. 2.1 Original cameras (discontinuous line) and rectified.

Writing the matrices of the virtual cameras in terms of their factorization:

$$\hat{P} = K[R - RC] \qquad \hat{P}' = K[R - RC'] \tag{2.1}$$

To define them assigning K, R, C and C' are needed. The intrinsic parameters for the matrix K can be chosen in an arbitrary way. The R matrix, which gives the position of the cameras, is specified through its row-vectors:

$$R = \begin{bmatrix} r_1^T \\ r_2^T \\ r_3^T \end{bmatrix} \tag{2.2}$$

These vectors are the x y and z axes, respectively, the system of reference of the virtual camera, expressed in global coordinates.

According to this:

- The x axis parallel to the base line:

$$r_1 = (C' - C) / \| C' - C \| \quad \textbf{(2.3)}$$

- The y axis orthogonal to x (required) and to a arbitrary unit vector k.

$$r_2 = k \times r_1 \quad \textbf{(2.4)}$$

- The z axis orthogonal to x and y axes (required)

$$r_3 = r_1 \times r_2 \quad \textbf{(2.5)}$$

In the second case, k defines the position of the axis in the orthogonal plane to x axis. k takes the direction of one real camera's optical axis, in this way, the virtual cameras are oriented in the same direction as the original cameras

It has been assumed that the new cameras have the same intrinsic parameters. In fact, the horizontal component of the centre could be different, which gives another level of freedom to center the rectified images to the interesting view, applying a horizontal translation.

To rectify the left image, for example, the transformation from the image plane P into $\hat{P}$ must be calculated. It is useful to imagine that the image is the result of the intersection between the plane and the cone of rays, coming from the points in 3D space, and the optical centre. The plane is moved while the cone is still static.

From the equation of the optical ray, for every 3D point X, projected over x in the real image and $\hat{x}$ in the rectified image, there are two parameters λ and $\hat{\lambda}$ such as:

$$\begin{cases} X = C + \lambda {P_{3x3}}^{-1} x \\ X = C + \hat{\lambda} \hat{P}_{3x3}^{-1} \hat{x} \end{cases} \quad \textbf{(2.6)}$$

Therefore:

$$\hat{x} = \frac{\lambda}{\hat{\lambda}} \hat{P}_{3x3}^{-1} {P_{3x3}}^{-1} x \quad \textbf{(2.7)}$$

The above transformation is a linear transformation of projective plane (named collineation) given by the 3x3 matrix $H = \hat{P}_{3x3}{}^{-1} P_{3x3}{}^{-1} x$. It must be realized that the unknown factor scale $\frac{\lambda}{\hat{\lambda}}$ can be depreciated because H transformation is defined until a scale factor (homogeneous). The same result is applied to the right image.

2.1.1.2. Implementation & Results

During this project the OPENCV library has been used, which implements a set of computer vision functions and tools that simplify some operations.

The calibration stage, which recovers the parameters of the cameras, has been performed with a tool provided by the above mentioned library. This tool gives the: extrinsic and intrinsic parameters, the lens distortion and the inverse of the rectification matrix for each camera view, in order to simplify the reprojection of the image.

The first step is release the lens distortion from each image applying the equations **(1.15)**. The second step is compute the rectified images using the concepts introduced in the above section. This process is done by multiplying the coordinates, in pixels, of every point of the image by the corresponding rectification matrix. This procedure has been performed as a look-up table in order to reduce the computation cost for applications with live video.

The resultant image should have holes due to the position of the pixels is discrete in the image, this should require a later interpolation, but working with the inverted rectification matrix it is possible to remap (reproject) without an interpolation, in other words, for each pixel in the resultant image, the pixel that should be located there, is calculated backwards.

Fig. 2.2 First row - original images, second row – rectified images.

In the figure above can be realized that the epipolar line drawn, red line, differs from less than one pixel in the rectified images.

Another operation is performed, due to the changing of the image plane; there are points that do not appear in the final image. This step computes the intersection of the resultant images and then obtains the common area for both images. 4 points are calculated in order to build a scale matrix to scale the images, vertically and horizontally. The resultant images fill completely the available area. The figure shows the procedure and the resultant images.

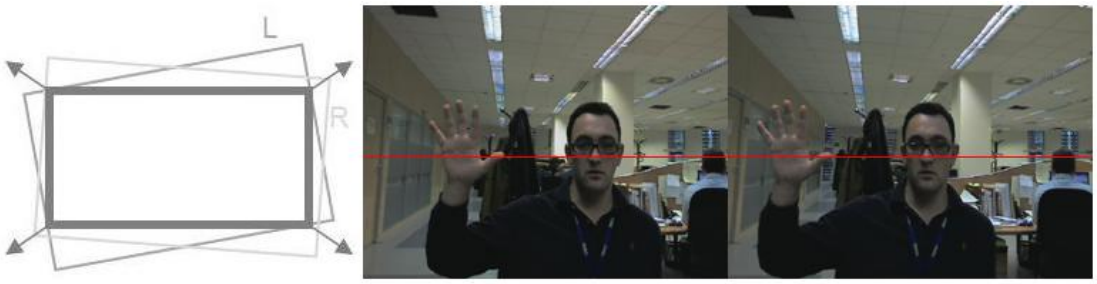


Fig. 2.3 Intersection, rectified and scaled images.

As is shown in the figure above, the red line ensures that the epipolar rectifications is reached, one pixel in the left can be found in the same line in the right image.

All the computations in this step are made only once and integrated in a look-up table.

2.1.2. Stereo matching

The properly 3D information must be obtained from the difference between the images involved in the stereo pair, rectified above. Finding the correspondences between the images, looking for which pixel in the right image corresponds to each pixel in the left image. This is one of the most well known problems in computer vision.

This section is focused to introduce the methods based in intensity matching because the area based correlation method, one of them, is the one implemented in this thesis, but there are a great number of techniques to perform the same process. The reason for this choice is because this method can be implemented in dedicated hardware because it is completely parallelizable, and can be performed in real time.

2.1.2.1. Disparity and depth

The main objective of the correspondence matching is finding the pixels from the two images $m = (u, v)$ and $m' = (u', v')$ which are projections from the same point P in the 3D scene. The resultant vector associates the matching pixels giving a value of difference, disparity, between their positions in each image:

$$d = m - m' = (d_u, d_v) \tag{2.8}$$

Once the stereo rig is rectified, the correspondence matching can be restricted to 1D, along the epipolar line. Once rectified, is coincident with a scanline of the image.

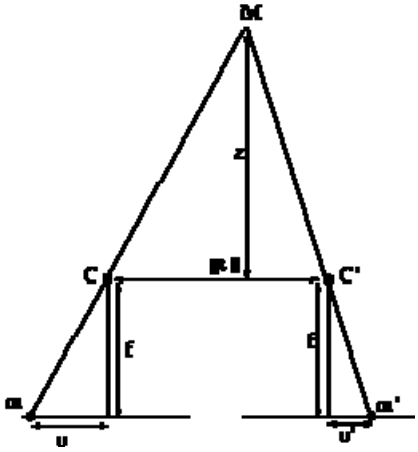


Fig. 2.4 Disparity and depth measures for parallel cameras, Faugeras [30].

Given a scene point M and its two projection points m of coordinates (u, v) and m' of coordinates(u', v'), the *disparity value d* is defined as $d = u' - u$

Note that $v = v'$ as there is no vertical parallax between the two cameras. The depth measure z of M is related to the disparity value d as follows:

$$z = \frac{f||t||}{d} \tag{2.9}$$

In parallel camera configurations, the epipolar lines coincide with the horizontal scanlines, and the epipoles are at infinity. The stereo matching is greatly simplified for parallel cameras.

Approaches to the correspondence problem can be broadly classified into two categories: the intensity-based matching and the feature-based matching techniques. In the first category, the matching process is applied directly to the

intensity profiles of the two images, in the second category features are first extracted from the images and the matching process is applied to the features. In next section we will discuss about the intensity-based stereo matching because it is the method implemented and makes denser disparity maps.

2.1.2.2. Intensity-based stereo matching

As shown in the previous section, the epipolar lines coincide with the horizontal scanlines once the images are rectified, the corresponding points in both images must lie on the same horizontal scanline. Such stereo configurations reduce the search for correspondences from two-dimensions to one-dimension. In fact, a close look at the intensity profiles from the corresponding row of the image pair, reveals that the two intensity profiles differ only by a horizontal shift and a local foreshortening.

An approach in intensity-based stereo matching, commonly known as the *area-based-correlation method*, is only to match those regions in the images that are interesting, for instance, regions that contain high variation of intensity values in the horizontal, vertical, and diagonal directions. A simple correlation scheme is applied in the matching process; a match is assigned to regions that are highly correlated in the two images.

The problem associated with this area-based-correlation approach is that size of the correlation windows must be carefully chosen. If the correlation windows are too small, the intensity variation in the windows will not be distinctive enough, and many false matches may result. If they are too large, resolution is lost, since neighboring image regions with different disparities will be combined in the measurement. Even worse, the two windows may not correlate unless the disparity within the windows is constant, which suggests that the multi-resolution scheme is appropriated.

A little window will preserve the contours, while a bigger window produce a low-pass filter effect that smooth the disparity map, losing the details information of the scene, specially the discontinuities. The prize for a bigger precision is more noise in the map, especially in the homogeneous regions, where the bigger window achieves better results.

A way to estimate the disparity between two images, applying the area-based-correlation technique is the Sum of Squared Differences (SSD):

$$SSD(u,v) = \sum_{\substack{area \\ around \\ (u,v)}} (I_l(i,j) - I_r(i,j))^2 \qquad \textbf{(2.10)}$$

where the disparity corresponds at the lowest value.

The advantage of this method is the recursive calculation, which can be implemented in hardware devices.

This procedure can be done for both images, from left to right and right to left, and then correct the disparity map for both views, with a process named cross-checking.

Another usual post-processing is the calculus of the sub-pixel resolution from the disparity map. This smoothing step consists in interpolate the curve of matching costs between the best value ant its neighbors. The improvement introduced by this process is relevant for images with a low disparity range. The reliability of the implemented algorithm is not enough to consider a fractional disparity computation. In the other side, with high resolutions, the real size of the pixel is little enough to achieve a good resolution in the disparity.

2.1.3. Cross-checking

A simple cross-checking process is applied to ensure consistency between bidirectional disparity estimations.

Once the disparities are calculated for both images, that is to say, left to right and right to left using each image as reference, the resultant disparity vectors are concatenated. The difference between the initial point and the final should be zero. In other case the estimated correspondence is obviously wrong, because of occlusions, regions with low texture or other error sources. Therefore, if the difference exceeds certain tolerance Δ the disparity value is deprecated.

2.1.3.1. Implementation & Results

Images in VGA resolution (640x480 pixels) are considered. The area-based correlation has been implemented so that all the operations inside the correlation window have been reused to reduce the computation cost, in C code. Then the changing in the correlation window size will change the time cost linearly.

Figure **2.3** shows how the disparity map changes with different sizes of the correlation window.

Fig. 2.5 Reference view and disparity maps window 5x5 and 15x15 with 80 disparity levels.

The figure **2.4** shows how the number of wrong pixels changes, in terms of disparity, with different sizes of the correlation window. With Δ=0. The deprecated points are marked in white.

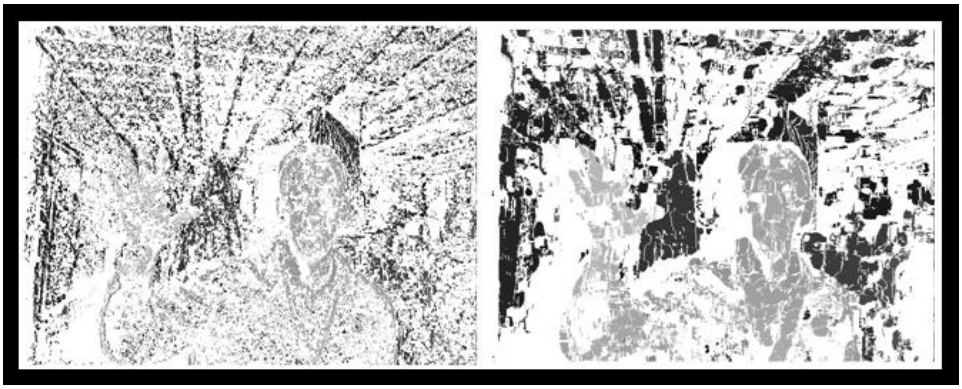


Fig. 2.6 Deprecated points (in white): window 5x5 and 13x13.

The following figure shows how the number of deprecated points grows with the correlation window size.

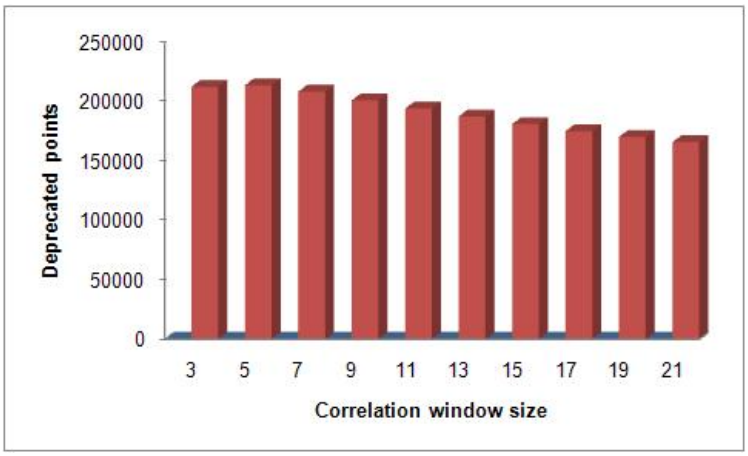


Fig. 2.7 Number of deprecated points depending on the correlation window size.

2.1.4. Color segmentation

A difficult problem, in common stereo algorithms, is to find correct depth in regions with low texture. Because of the similarity of matching costs for all the pixels in these regions, the resultant depth map by picking the best matching score is usually noisy.

Supposing that for a region with homogenous color usually there is no large depth discontinuities then the noise resultant from initial depth estimation can be removed for textureless region. The affirmation above implies a depth representation based on splitting the reference image into homogeneous color regions.

Once the reference image is segmented, in each color segment, the depth surface can be modeled as a plane surface with small depth variations for each pixel. This model guarantees smoothness in textureless regions. However, depth estimation is usually more reliable in high texturized areas.

It should be emphasized that the color segmentation is not one of the main goals of this thesis. Over-segmentation of smooth surfaces is tolerated, while it still keeping the boundary information. Following above affirmation, the depth boundaries should coincide with color segmentation boundaries.

2.1.4.1. Color Segmentation Algorithm

Any algorithm that decomposes an image into homogeneous color regions should be enough for the proposed method. As has been said, the most important issue is that the algorithm keeps the boundaries of the image but, in addition, the minimum segment size should be modified by parameter. Since the algorithm enforces the depth continuity inside each segment strictly, under-segmentation should be avoided. Two methods has been tested, taking account the code availability and the references in related works (i.e. **[31] [32] [33]**).

Color Segmentation based on the Mean-Shift Algoritm

The first method tested is based on the Mean shift filtering that is a data clustering algorithm commonly used in computer vision and image processing. For each pixel from an image, the set of neighboring pixels, defined by a radius and a color distance, is determined. Once the pixel neighborhood is selected the next step is calculate a new a spatial center and a new color mean value. The computed mean values are used as the centers in the next iteration. This operation is repeated recursively until these values stop changing. Once the pixel clusters are established, the next step is the segment indexing, for a later managing.

The Mean-shift filtering algorithm is included in the OpenCV library, but it has been deprecated due to the slow performance, and it requires an indexing stage, implemented using the cvFloodFill function provided by OpenCV, which returns an structure with information about the neighborhood of the selected pixel. This algorithm is further explained by D. Comaniciu and P. Meer at **[34]**. The following image shows an example result.

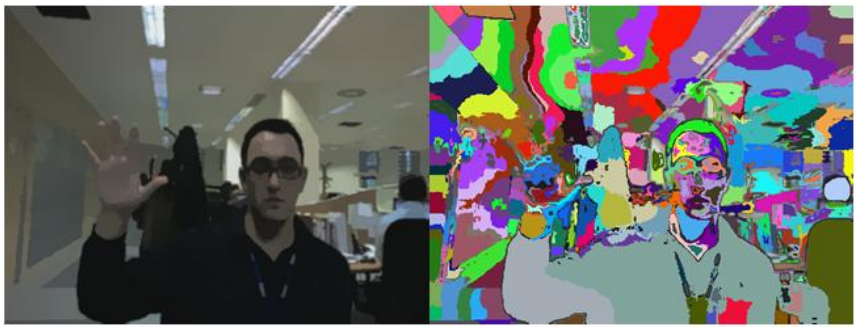


Fig. 2.8 Mean-Shift filtering results: left image after mean-shift filtering (15 spacial radius and 15 color radius), right false color representation.

Color Segmentation – Graph Based Segmentation

Another technique to solve the problem of segmenting an image into regions is using a graph-based representation of the image **[35]**. This technique represents the problem in terms of a graph $G = (V, E)$ where each node $v_i \in V$ corresponds to a pixel in the image, and the edges in E connect certain pairs of neighboring pixels. A weight is associated with each edge based on some property of the pixels that it connects, such as their image intensities.

An efficient implementation of this algorithm has been found online **[35]**, providing good results and being flexible for later improvements. This is the selected method for the final implementation, in addition, it does not require the indexing stage performed with cvFloodFill due to the segments data can be extracted from the graph structure and it requires less interaction, that is to say, the configuration parameters do not have to be changed for each scene.

This algorithm receive three parameters as input: σ, as the deviation of the previous Gaussian filter; k, is a value used by the algorithm to calculate a threshold function that establishes the number of segments in the image, larger k values produces larger segments; and the minimum segment size in pixels.

Fig. 2.9 Graph based segmentation results. Left: σ=0.3 k=400. Right: σ=0.6 k=300.

The figures above shows how the boundaries are preserved.

2.1.5. Plane fitting

This step will help to recover the depth information from occlusions, according to the affirmation that near regions with similar color must have no disparity discontinuities.

The depth in each segment is then represented as 3D planar surface according **[32]**:

$$\frac{1}{Z} = Ax + By + C \qquad \textbf{(2.11)}$$

As shown in **(2.9)** depth of a pixel is inversely proportional to its disparity, then

$$d = ax + by + c \qquad \textbf{(2.12)}$$

The goal is to find the least square solution (a,b,c) of this linear system. Considering that the least square method is very sensitive to the effects of outliers, only those points with reliable disparities are used to fit the plane, the points marked as occlusions or wrong matches are not considered.

The method chosen to find the solution to the above equation is the SVD (Singular Value Decomposition, further explained at **[36]**). The essential trick consists in subtracting the mean of the x, y and d data. Figure **2.8** illustrates which is the possible result:

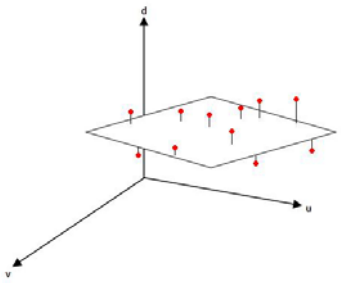


Fig. 2.10 Plane fitting.

Other methods can be used to find a better solution, i.e. RANSAC, but are computationally more expensive.

2.1.5.1. Implementation & Results

In the above step an array with all the color segments is constructed. Each segment of the image is analyzed independently, following the matlab code, which illustrates the procedure, later implemented in C (u and v are the image coordinates in pixel form).

```
%d=Au+Bv+C
K=[mean(u),mean(v),mean(d)];
[U,S,V]=svd([u-K(1),v-K(2),d-K(3)],0);
M=-1/V(end,end)*V(:,end);
A=M(1), B=M(2), C=-K*M
```

As a post-processing for the resulting image the segments that exceeds a certain slope, determined as a thresholding, are deprecated considering that contain outliers.

The following figure shows the result after the above process for all the segments of the image:

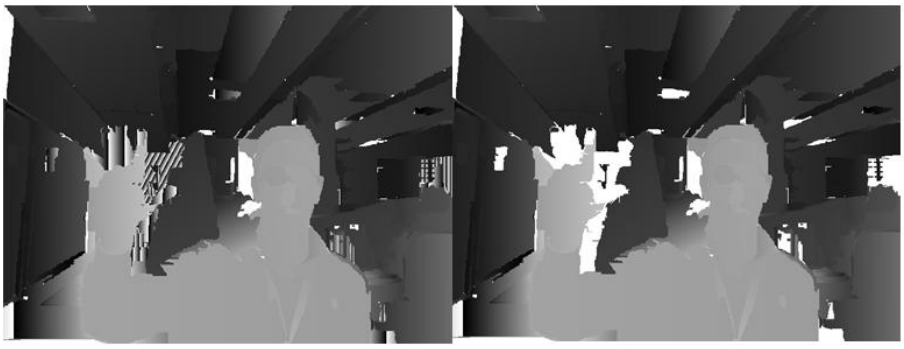


Fig. 2.11 Plane fitting with deprecated segments marked in white (segments with more than 30 degrees of slope are deprecated).

Another post-process is performed over the resultant disparity map. A smoothing step for the above resultant disparity is introduced too.

2.1.6. Anisotropic Diffusion

Once the plane fitting is done, the remaining map contains depth discontinuities between segments with color similarities. The next step should be fuse or merge the depth information between neighboring segments. There are several techniques, like belief propagation **[28]** which uses global constraints to propagate information between segments, but are so computationally

expensive. The goal is to find a way to fuse the depth information in the segment boundaries without losing the edges of the original images.

Diffusion refers to an aggregating or averaging operation that is implemented by repeatedly adding to each pixel the weighted values of its neighboring pixels. Non-linear and anisotropic diffusion has been proposed for a variety of early vision tasks, including edge detection, introduced by Perona and Malik **[24]**. Proesmans **[25]** detect discontinuities in optical flow by comparing forward and backward flow estimates and then using a diffusion process to smooth the discontinuity maps. Proesman also use an anisotropic diffusion process to smooth the disparity estimates. Sapiro and Tannenbaum **[26]** used a similar technique to perform edge preserving smoothing of images.

Perona and Malik formulate the anisotropic diffusion filter as a diffusion process that encourages intraregion smoothing while inhibiting interregion smoothing. Mathematically, the process is defined as follows:

$$\frac{\delta}{\delta t} I(\bar{x},t) = \nabla \bullet (c(\bar{x},t)\nabla I(\bar{x},t)) \qquad \textbf{(2.13)}$$

In this case, $I(\bar{x},t)$ is the reference image. $\bar{x}$ refers to the image axes (i.e. (x,y,z)) and t refers to the iteration step. $c(\bar{x},t)$ is called the *diffusion function* and is a monotonically decreasing function of the image gradient magnitude:

$$c(\bar{x},t) = f(\nabla I(\bar{x},t)) \qquad \textbf{(2.14)}$$

It allows locally adaptive diffusion strengths; edges are selectively smoothed or enhanced based on the evaluation of the diffusion function. Although any monotonically decreasing continuous function of ∇I would suffice as a diffusion function, two functions have been suggested by Perona and Malik:

$$c_1(\bar{x},t) = \exp\left(-\left(\frac{|\nabla I(\bar{x},t)|}{K}\right)^2\right) \qquad \textbf{(2.15)}$$

$$c_2(\bar{x},t) = \frac{1}{1+\left(\frac{|\nabla I(\bar{x},t)|}{K}\right)^{1+\alpha}} \qquad \alpha > 0 \qquad \textbf{(2.16)}$$

K is referred to as the *diffusion constant* or the *flow constant*. Obviously, the behavior of the filter depends on K. Could be helpful to define a flow function:

$$\Phi(\bar{x},t) = c(\bar{x},t)\cdot\nabla I(\bar{x},t) \quad \textbf{(2.16)}$$

Equation **(2.13)** can be rewritten as:

$$\frac{\delta}{\delta t} I(\bar{x},t) = \nabla \bullet (\Phi(\bar{x},t)) \quad \textbf{(2.17)}$$

2.1.6.1. Implementation & Results

In this thesis a discrete implementation of the algorithm has been performed, based on Perona and Malik definition. Three ideas must be considerated:

1. In the discrete domain, a gradient can be approximated as the difference in intensity between neighboring elements, pixels, in the image.
2. The flow function introduced in Equation **(2.17)** can be calculated independently for each of the neighboring elements.
3. The filter is iterative; the right hand side of Equation **(2.13)** describes the change in image intensity produced by one iteration of the filter.

The filtering process consists of updating each pixel in the disparity image by an amount equal to the flow contributed by its nearest neighbors, using as gradient input the intensity of the reference image.

$$\frac{\delta}{\delta t} I(\bar{x},t) = \Phi_{west} + \Phi_{east} + ... + \Phi_{nord-west} + \Phi_{nord-east} \mid \Delta t = 1 \quad \textbf{(2.18)}$$

$$I(x,y,t+\Delta t) = I(x,y,t) + \Phi_{west} + \Phi_{east} + ... + \Phi_{nord-west} + \Phi_{nord-east} \quad \textbf{(2.19)}$$

In the implementation the Equation **(2.16)** has been used as a diffusion function, with a K =5 and α = 1. The filter application has been divided in two stages: the first one, only the deprecated are filtered and in the second stage all the pixels in the image are filtered.

Following some result with different number of iterations:

Fig. 2.12 Anisotropic Diffusion, 50 and 100 iterations.

2.2. Reconstruction of 3-D coordinates

Given that the cameras have been calibrated and the two perspective projection matrices $P = [q_{ij}]$ and $P' = [q'_{ij}]$ are known (modified to work with rectified images, is that to say, each projection matrix is multiplied by the corresponding rectification matrix), then for any scene point M with unknown 3-D coordinates (X, Y, Z), that projects onto the two retinal planes at (u, v) and (u', v') (known from the disparity computation, $v' = v$ and $u' = u + d$, in the rectified images) the result is

$$P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \; and \; P' \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = s \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} \tag{2.20}$$

Where s is a scale factor, depending on the pixel size.

Eliminating *s* and *s*' and combining the two equations into matrix form gives

$$\begin{bmatrix} q_{11} - uq_{31} & q_{12} - uq_{32} & q_{13} - uq_{33} \\ q_{21} - vq_{31} & q_{22} - vq_{32} & q_{23} - vq_{33} \\ q'_{11} - u'q'_{31} & q'_{12} - u'q'_{32} & q'_{13} - u'q'_{33} \\ q'_{21} - v'q'_{31} & q'_{22} - v'q'_{32} & q'_{23} - v'q'_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} u - q_{14} \\ v - q_{14} \\ u' - q'_{24} \\ v' - q'_{24} \end{bmatrix} \tag{2.21}$$

This is a linear system in(X, Y, Z). The 3-D coordinates of M can be easily computed.

2.2.1.1. Implementation & Results

An OpenGL window has been implemented to enable the navigation through the cloud of points generated by the transformation 2D into 3D.

Now, applying the equation **(2.21)** it is possible the computation of the 3D coordinates for every pixel, and then render the resultant 3D model over an OpenGL window, combining the disparity map, the image coordinates of each pixel and the geometrical information obtained during the calibration step.

A cloud of points is obtained, with pixels located in its spatial position. We can navigate clicking over the window changing the point of view.

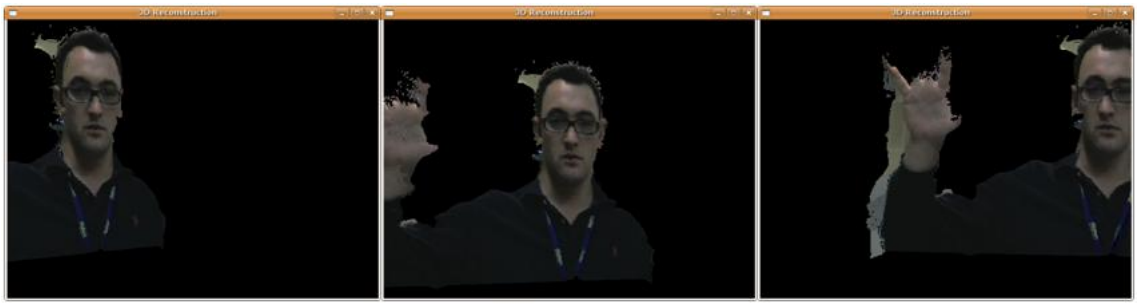


Fig. 2.13 3D Reconstruction observed from different points of view.

CHAPTER 3. IMAGE BASED RENDERING

Once implemented the technique to recover the 3D information of the scene that has been introduced, it is time to find a way to render this information as much realistic as possible.

In the previous section it has shown a way to render the scenario reprojecting the 2D points into its real position in the 3D space. But the results of this technique are so far of being realistic. It should be necessary introduce another kind of techniques, for example using geometrical models, that are computationally expensive (the computation time increases with the scene complexity) and poor realistic.

This topic is introduced in this thesis because makes possible achieve interactive rates in rendering, with certain quality. Image Based Rendering introduces several techniques those can produce photorealistic synthesized novel images. It is possible generate 2D images from a 3D object without know, a priori, the geometry of the scene. This new images can be generated using fewer operations than geometric models.

3.1. Image Based Rendering Techniques

These techniques are classified depending on the degree of knowledge of the scene geometry. Two techniques in different ways will be introduced: one that exploits the geometrical constraints between three views, to reproject the pixels on an image to its appropriate position given a new virtual camera position (it requires certain knowledge of capture geometry); another technique that produces photorealistic images without any knowledge of the scene geometry will be introduced, based on the parameterization of the plenoptic function.

3.1.1. Transfer methods

Transfer methods are characterized by the use of a relatively small number of images with application of geometric constraints (either recovered at some stage or known a priori) to reproject image pixels appropriately at a given virtual camera viewpoint. The geometric constraints can be in form of known depth values at each pixel, epipolar constraints between pairs of images, or trifocal tensors that link correspondences between triplets of images; that is to say, these methods require certain knowledge of the scene geometry.

Laveau and Faugeras **[17]** use a collection of images called reference views and the principle of the fundamental matrix to produce virtual views. The new viewpoint, which is chosen by interactively choosing the positions of four control image points, is computed using a reverse mapping or ray-tracing process. For

each pixel in the new target image, a search is performed to locate the pair of image correspondences in two reference views. The search is facilitated by using the epipolar constraints and the computed dense correspondences between the two reference views.

Avidan and Shashua **[18]** introduce a method to synthesize photorealistic novel views exploiting the constraints of the trifocal tensor. Knowing the camera parameters of two views and a new virtual camera, they propose a method to transfer the points from the first view into the new image plane. This method is named tensor transfer, and uses the incidence relation point-line-point between the three views.

3.1.2. Transference problem

Given three views of a scene and a pair of corresponding points in two views, is possible to know the position of the point in the third view. Knowing the camera matrices, independently of the image content, it is possible establish the position of the point in the third view. This is the problem of the point transference.

3.1.2.1. Point transfer with the fundamental matrices

The problem of the transference can be solved knowing only the fundamental matrices. If the fundamental matrices between the three views are known, F_{21}, F_{31}, F_{32}, and a pair of corresponding points x and x' of the first two views, from the equations studied above (see **Two-view geometry**) is deductible that the point in the third view could be given by:

$$x'' = (F_{31}x) \times (F_{32}x') \qquad \textbf{(3.1)}$$

The third point can be defined as the intersection of the epipolar lines of the points x and x' in the third view. It can be observed that the matrix F_{21} does not appear in the equation. The equation **(3.1)** requires a perfect matching of the points x and x', however the noise makes that this condition becomes inexact, given error in the calculus of x''. This method will require techniques to minimize this error. This algorithm is named epipolar transfer.

Nevertheless, this method presents serious problems when the epipolar lines defined in the third view are coincident, or almost coincident. This is a degenerated case due to x'', e_{31} and e_{32} are collinear in the third view, which means that the camera centers (C and C') and the point X define a plane with C'', the centre of the third camera. Therefore, the point X is on the trifocal plane defined by the cameras' centres.

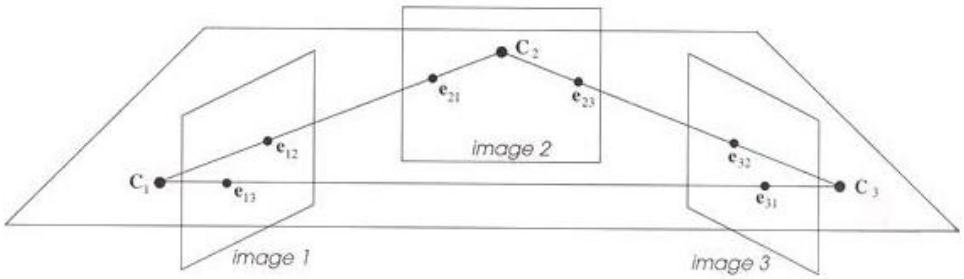


Fig. 3.1 Degenerated case for epipolar transfer [31].

For all the points lying over this plane the method will fail.

3.1.2.2. Point transfer with the trifocal tensor

The above degenerations can be solved using the trifocal tensor. Consider the correspondence $x \leftrightarrow x'$. If a line l', containing the point x', in the second view, then the corresponding point x'' can be obtained transferring the point x from the first view through the homography induced by the plane defined with the chosen straight line

$$x''^{k} = x^{i} l'_{j} \, \Im_{i}^{jk} \qquad \textbf{(3.2)}$$

The above equation is the tensor form of the point-line-point incidence relation, known as tensor point transfer. The line l' cannot be the epipolar line corresponding to the point x because of $x^{i} l'_{j} \, \Im_{i}^{jk} = 0^{k}$, then the point x'' is undefined. A good chosen is the perpendicular line to the epipolar line, going through the point x'.

This is the method finally implemented in this thesis.

3.1.2.3. Implementation & Results

In this section the algorithm implemented to solve the tensor point transfer will be explained. An overview of the algorithm can be seen in the next figure:

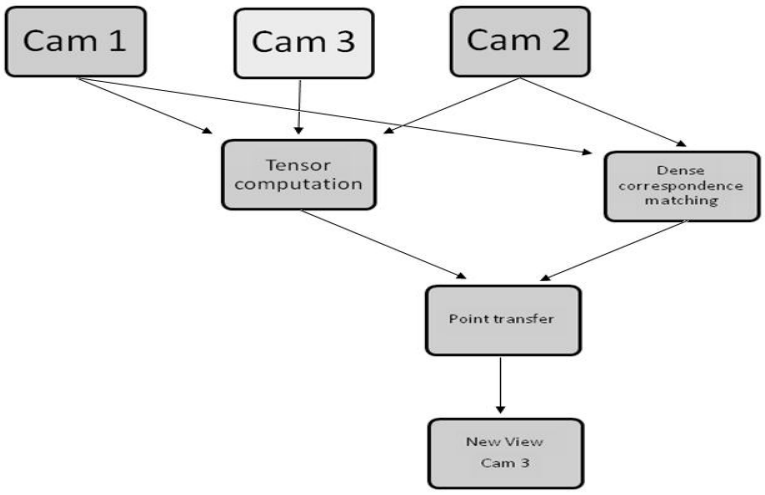


Fig. 3.2 Algorithm overview

The Cam 3 represents the position of the new virtual camera; Cam 1 and Cam 2 are the known cameras with known parameters:

Left Extrinsics $[R \mid t]$	**Intrinsics** K
-0,727539 -0,076493 -0,681789 0,580370 0,580370 -0,148053 0,987855 0,047155 0,047155 -0,257863 0,669901 0,135247	530,669373 0,000000 315,305389 0,000000 519,082275 245,085922 0,000000 0,000000 1,000000
Right Extrinsics $[R' \mid t']$	**Intrinsics** K'
-0,719657 -0,082746 -0,689381 0,264228 0,264228 -0,085841 0,995859 -0,029922 -0,029922 -0,154369 0,689003 0,037644	534,017334 0,000000 318,688416 0,000000 521,390259 245,504974 0,000000 0,000000 1,000000

The first step is computing the Trifocal Tensor from the camera parameters. From the equation **(1.42)** the tensor can be computed if, $P = [I \mid 0]$, $P' = [A \mid a_4]$ and $P'' = [B \mid b_4]$ (matrices associated to the cameras) are known. P and P' are known, then, is necessary obtain the matrix of the new camera.

Note that the matrix P is given in its canonical form. Then, P' and P'' are given related to P. Will be necessary convert the original matrix P to its canonical form and later apply the same transformation to P' and P''.

The application has been designed to compute images from views located between the two original cameras, in this way, receive as input the number of virtual views required. Then, to obtain the new camera matrix, the solution

adopted is dividing the space between the real cameras in discrete locations. From the definition $P = K[R \mid t]$ and $P' = K'[R' \mid t']$, it is possible compute the rotation and the translation between this two image planes and then generate a extrinsic matrix for one of the discrete locations. The instrinsic matrix K'' for the new virtual camera will be $K'' = K$. An example of virtual camera's matrix computed for a 5th view, when the space between the both original cameras is segmented in 10 possible views.

New Extrinsics 5th View $[R' \mid t']$
-0,723155 -0,079445 -0,686101 0,422299
0,422299 -0,117235 0,993067 0,008578
0,008578 -0,206116 0,680663 0,086638

Once the new camera matrix is obtained, the trifocal tensor can be computed easily.

This process is performed twice, that is to say, two tensors are computed: one taking the Cam 1 as reference and one taking the Cam 2.

The next step is to find the correspondences between the pixels from the two original views. To perform this operation there are several possibilities: techniques based in optical flow, when the points of view are near in the space and exists low displacements between the images, but as a generalization, to give more flexibility at the system, the solution proposed at section **2.1** will be adopted: stereo matching approach.

The disparity maps of both reference views will give the information about the correspondences between the images. Note that the disparity maps are computed from the rectified images, to render the new view into its real position the disparity maps will be unrectified for later computations.

The following step is the reprojection. The equation $x''^k = x^i l'_j \, \Im_i^{jk}$ will be applied to all pixels of both reference images to reproject them into the new view. Note that l' is the line perpendicular to the epipolar line, going through the point x' in the second image, computed from the product of fundamental matrix F and x'.

$$F = K^{-1^T} R [t']_\times R'^{-1} K'^{-1} \qquad \textbf{(3.3)}$$

Where $[t']_\times$ is the skew-symmetric matrix of t', in this case is $C - C'$ being C and C' the optical centers.

Once both images have been reprojected to the same place is time to combine the two views. The contribution of each view to the final images is weighted according the distance of the novel view to each reference view.

Next, some results with the above camera matrices.

Fig. 3.3 Left image reprojected (first row) and Right image reprojected (second row), Novel view, combination of both.

Can be seen there are errors in the reprojected views due to the errors in the stereo matching step or in the calibration process. This technique produces good result in systems with short baselines between the cameras.

This stage has been implemented in order to create a Look-up table, then all the operations are performed only once per virtual view. There are two look-up tables, one to project the left image and one for the right. Only merging is done each time.

3.1.3. Light Field Rendering

The other group of techniques, included in the set of Image Based Rendering, is constituted by algorithms which do not exploit any knowledge of the scene geometry. These methods are independent of the scene complexity and seem a good solution for future applications towards free viewpoint video.

In this thesis, a light field rendering method, the first introduction presented by Marc Levoy and Pat Hanrahan **[22]**, has been implemented. The light field is the radiance value going from a point in a particular direction. In general, this

ray can be modeled as a 5D function, known as plenoptic function. This function can be simplified for a region with no occluders reducing the function to 4D.

If it is possible capture this light field this data can be used to generate novel images, given a viewpoint, because the image synthesis requires sampling a set of rays going from the viewpoint to the image plane.

The light reflected from an object's surface fills the transparent space around it. This light field around static objects can be quantified and recorded by capturing images from many view points. The positions of the cameras for these images can be arranged on a plane in a regular grid. This 2D array of 2D images parameterizes the 4D light field. In the ideal case, the light field should be sampled densely; such the apparent motion between images does not exceed one pixel. In the other side, if intermediate positions are generated by interpolation from several images, the rendered image will appear blurred. The number of images required to guarantee less than one pixel apparent motion between adjacent images is incredibly big even for low resolution images. Then, light fields from real scenes are usually a sub-sampled representation of the complete light field information.

A set of conventional 2D images is used to render arbitrary views of a static 3D scene. The images store the light distribution, or light field, around the scene. The result is independent from scene complexity or illumination.

3.1.3.1. Representation of a light field

Since the light field measures radiance samples along rays in space, the rays in space must be parameterized. In **[22]** a two planes parameterization is proposed. Any ray in space can be defined by the intersection with two planes in the 3D space. The planes are chosen to be parallel one to other. One of the planes is parameterized as by u and v and the other plane is parameterized by s and t. This basically means that the potential rays considered form a slab, which is named the light slab. Also, it is assumed that the viewpoint looks towards the $s-t$ plane from the $u-v$ plane. Any ray can be indexed with the light slab by a (u, v, s, t) quadruple.

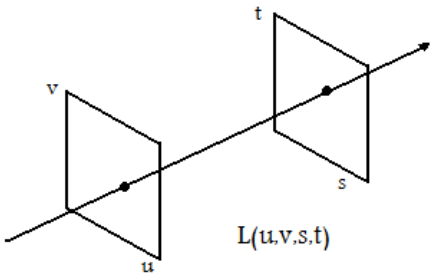


Fig. 3.4 Parameterization of a light field [22].

3.1.3.2. Building a light field

The light fields are usually generated for synthetic scenes, due to the complexity in the capture stage. A light field can be generated by assembling a collection of images. The radiance samples for all rays which intersect the $u-v$ plane at a single point can be generated by rendering the scene with the point in the $u-v$ plane as the viewpoint and the $s-t$ plane as the image plane. If this procedure is done for all sampled points on the $u-v$ plane the complete light slab is produced.

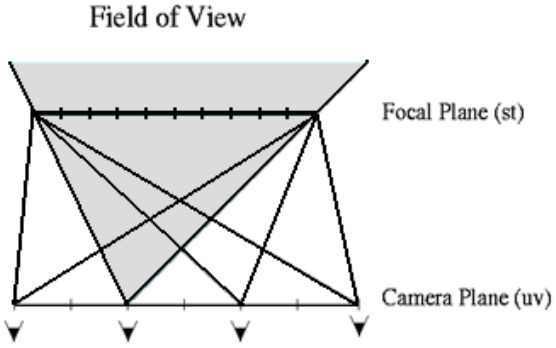


Fig. 3.5 Creation of a light field [22].

3.1.3.3. Displaying a light field

Constructing and displaying an image from the light slab requires a resampling of the 2D slice of lines from the 4D light field. Each line is a ray through the eye point and a pixel centre. To generate an image from a novel viewpoint, the radiance samples along each ray which goes from the viewpoint to the image plane must be recovered.

The algorithm has two steps: the first computes the (u, v, s, t) line parameters for each image ray, and the second consists in resample the radiance at those line parameters. The (u, v, s, t) parameters are associated with the points on the planes, where the ray intersects them. Then, the $u-v$ and $s-t$ parameters are calculated by determining the point of intersection of an image ray with each plane.

The second step involves resampling the radiance. Resampling implies reconstructing the image from the original samples for a novel viewpoint. First of all, an interpolation of the nearest neighbors is done.

3.1.3.4. Implementation & Results

In this thesis some light fields from the Stanford data-sets **[37]** are used to test the behavior of this method, and a light field viewer has been implemented, that is to say, only the resampling stage has been performed.

Equations considered for resampling:

$$\begin{aligned} X1 &= ((Z - Z1)/Z)(i_x / i\_widht - 0.5 - X) + X \\ Y1 &= ((Z - Z1)/Z)(i_y / i\_height - 0.5 - Y) + Y \\ U &= uvi\_width((X1 + 1000/uv\_width) - 1 \\ V &= uvi\_height((Y1 + 1000/uv\_height) - 1 \end{aligned} \quad \textbf{(3.4)}$$

The point of view is (X, Y, Z), all the rays coming from the image will meet at this point. i_x and i_y which are the coordinates of the pixel in the novel image.

If the ray intersects UV plane at coordinate (u_1, v_1) and ST plane at (s_1, t_1) then, the desired ray will be $L(u_1, v_1, s_1, t_1)$. The value of the pixel $L(u_1, v_1, s_1, t_1)$ is taken but, this is the value only for one pixel of the image. This procedure is done for all the pixels in the image to create the novel image.

The equations above are an example for the specific case of the dragon data-set where the real dimensions are: ST plane - 1x1, UV plane - 2000x2000 ($uv\_widht \times uv\_height$). The real data is ST - 256x256 pixels ($i\_widht \times i\_height$), UV -8x8 images ($uvi\_widht \times uvi\_height$).

Fig. 3.6 Left: original image - Right: synthesized image.

The synthesized image appears blurred due to the interpolation process, between the chosen pixel and its neighbors in the $u - v$ plane.

CHAPTER 4. DESCRIPTION OF THE APPLICATIONS

The applications core has been developed in C due to the computational cost of the process, but some libraries have been used to enhance the behavior of the applications, described in **ANNEX A**. The hardware description is available in **ANNEX B**.

Two applications have been developed: the first one implements the concepts introduced in **CHAPTER 2** and **CHAPTER 3** since section **3.1.2.** It has been designed modularly to accept several cameras and switch between stereo rigs depending on the desired point of view. Nowadays, only two cameras are plugged two check the behavior of the algorithms.

The second application is a stand-alone tool to render light fields, is much more simply than the first one, because the only goal is give an overview of the algorithms based in Image Based Rendering without any knowledge of the scene geometry.

The first application receives the parameters from the command line, like: the correlation window size, maximum and minimum disparity values, number of iterations for anisotropic diffusion, and the operation mode:

```
Usage: 3DRecovery
       [-W ]                # Window Width
       [-H ]                # Window Height
       [-maxD]              # Max Disparity
       [-minD]              # Min Disparity
       [-Cw]                # Correlation Window Size
       [-It]                # Number of iterations for AF
       [-hf]                # Hole filling with anisotropic Diffusion
       [-camera]            # If set use video Input
       [-tensor]            # IBR view synthesis
```

The application has two operation modes, the first one for 3D reconstruction, and the second one developed to perform the Image Based Rendering technique.

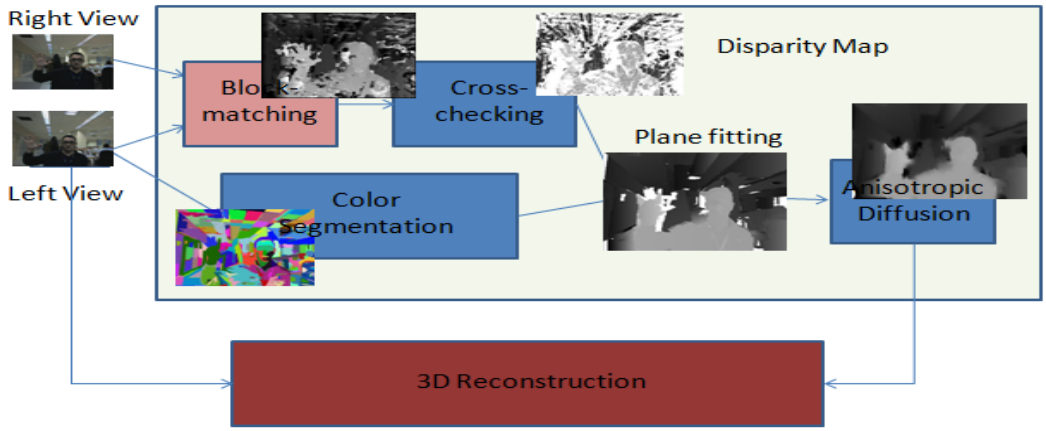


Fig. 4.1 3D Reconstruction mode.

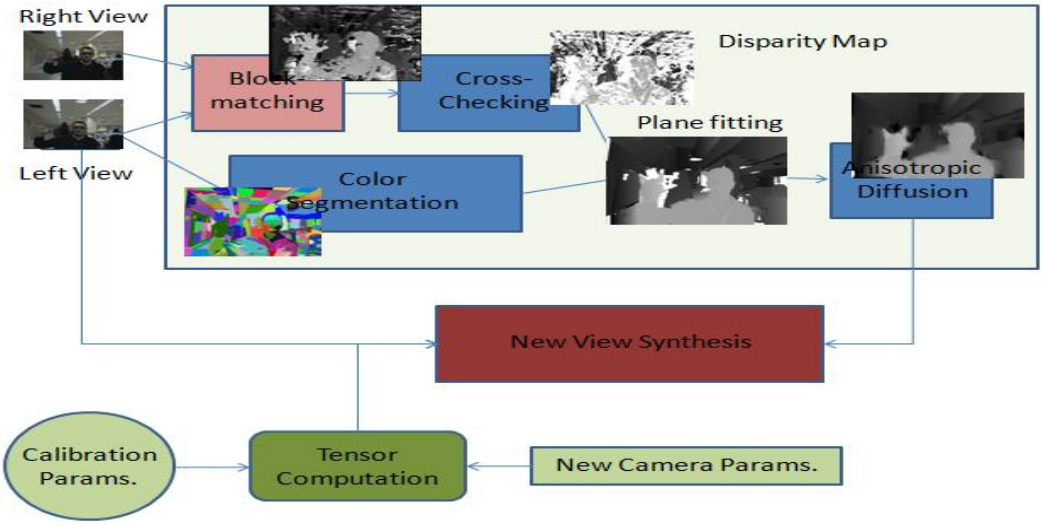


Fig. 4.2 Image Based Rendering mode.

The camera parameters are recovered using a tool provided in OpenCV Library. OpenCV library also helps to manage the images, structure **IplImage**, and perform some algebraic operations as: **SVD** solving (cvSVD for plane fitting), matrices, etc.

Both modes have a common step, disparity map computation.

The color segmentation and intensity stereo matching are performed in two different threads to take advantage of the available dual processor, decreasing the computation time, because they are the heaviest steps. POSIX Thread programming Library helps to perform this process.

The 3D Reconstruction mode shows a cloud of points formed by the pixels located in 3D coordinates (X,Y,Z) once they have been recovered, in an OpenGL window that allows mouse navigation. The OpenGLUT library makes easier construct this kind of windows, but the management of the changes with the mouse interaction is developed by OpenGL.

For the Image Based Rendering Mode, the parameters from calibration step will be recovered to compute the Trifocal Tensor and then solve the transfer problem for the new view. The number of views to be rendered can be established in the launch parameters, and then navigate through the rendered views, shown in a GTK window, with the arrow keys.

In the **ANNEX C** a table with the time cost, for each step of algorithm, is shown.

The light field viewer, receives as parameters the name of the light field to load (dragon, buda, huvee, are the available) and the size of the $u - v$ plane (sizes of the images grid that composes the light field). Then, with the arrow keys is possible change the point of view. The synthesized image is rendered in GTK Window.

The light field is stored as a set of PNG images. The filename of every image codifies its position in the grid (i.e. budda.2.1.png means that this images is placed at second row first column). The images are loaded in an array of Iplimages, and are selected depending on the results of the equations **(3.4)**.

```
Usage: LightFieldView
      [-lf ]            # Name of the light field to load
      [-s ]             # Grid size (sxs)
```

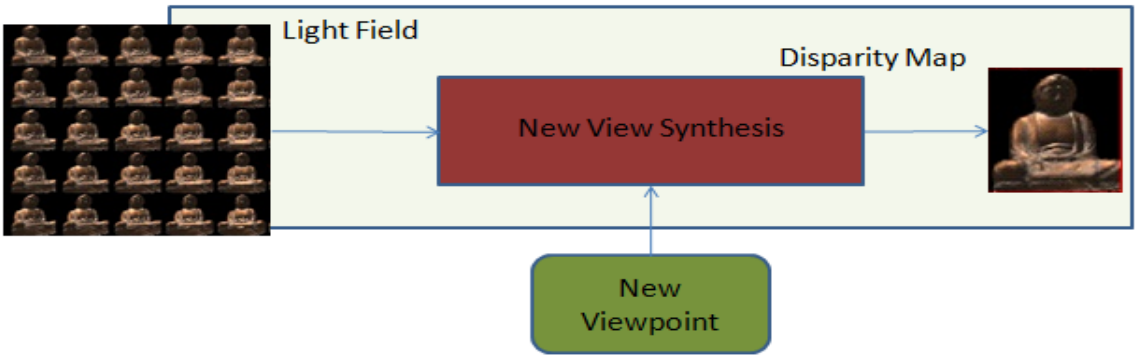


Fig. 4.3 Light field viewer.

CHAPTER 5. CONCLUSIONS & FUTURE LINES

Nowadays, the consumers are becoming more and more demanding in terms of communications and leisure. Therefore, it is necessary to offer innovations that cover their necessities.

Observing the progress of the media, it is possible to think that every time a greater realism is required and this can be offered with the current technology.

The application implemented in this thesis extends from the capture stage of the scene until the rendering of synthetic images.

The developed system is modular and scalable, with the ability to replace or upgrade any stage independently from the others. It has been designed so that, starting from a basically system of two cameras, being able to add more devices without reform the core of the application, but all the tests and results are based on a stereo pair.

The rectification of the stereo pair has been implemented as a look-up table, for both distortion corrections and the epipolar rectification, reducing the computation time to the time required for memory accesses.

The system performs a stereo matching stage over stereoscopic images, based on an area based correlation algorithm. It has been observed that the result of depth maps depends on the size of the window correlation. It is concluded that a larger window gives a better behavior in textureless regions, but paying the prize of losing resolution in the image boundaries

It has also been shown that cross-checking improves the later results due to the elimination of wrong correspondences after checking the disparities left-right. How varied the number of discarded pixels according to the correlation window size is also tested, lower with the increase of the window size.

Two methods were tested in order to perform the color segmentation step, mean-shift filtering and graph based segmentation. Concluding that which best performing shows, requiring less interaction (it is not necessary to change the parameters constantly), is based on graph based segmentation, being less sensitive to changes in the scene and preserving the boundaries of the original image.

It has been found that the plane fitting technique produces good results, in terms of preserving the continuity of depth in homogeneous regions. In addition, it helps to recover the contours of the objects, from the original image, on the disparity map, which most of them are lost because of the usage of high correlation window size.

An anisotropic diffusion filter has been integrated in the algorithm, noting interesting results, reducing the depth discontinuities between segments belonging to the same object in the image.

In summary, the end result of the depth extraction step, prior to an assessment with test data-sets, is usable for a later image synthesis step.

It is also observable in the **ANNEX C** that the bottleneck of the application is in the filtering stage, because it is a very iterative algorithm, although, the real limitation is in the color segmentation step. The other modules can be implemented on dedicated hardware, because they are parallelizable. The color segmentation algorithm proposed is recursive.

The reprojection stage, generating intermediate views from the trifocal tensor, has been implemented so that the two original images are projected on the position of the virtual camera. Both projections are combined depending on the distance between the original cameras and the virtual one. A good way to see that the correspondence matching produce good results is to check, that the projection of the two views over the same position, produces low noisy results.

In the other hand, the module above has been designed to build a look-up table for each new viewpoint. The time spent to build each view point is around one second, but this procedure is made only once per view and frame. It should be interesting try to build some kind of 3D lookup table, considering all the possible depths in the image, in order to make this only once per stream. The initial cost would be bigger but it would not require more computation. It could be achieved real time ratios for the rendering stage.

For the proof of concept done about Light Field Rendering, it is noted that the processing time is minimal. Because the transactions made on the images are not expensive, but the memory required is very high because of the volume of input data needed. The results are better with the larger size of the light field, that is to say, lower relative displacement between images of the array.

In addition, once presented the results obtained with 3D reconstruction of the pixel coordinates, it has been possible to observe that the reconstruction, of the complete scene, requires a greater input data, more seeing of reference, to obtain feasible results, however with the method based on generating virtual views just by two views we can generate a great amount of points of view, although all in a limited space margin, maintaining a good quality.

There are many possibilities to continue this development, and then suggest some of them:

Depth estimation: The algorithm does not consider exploit the temporary redundancy between images. Under common circumstances, it can be assumed that the depth does not change dramatically from one image to the next. Incorporating information from previous frames could help reduce errors on the disparity map and the computation time in between consecutive frames. Although is no considered that a final solution passes through a software implementation, but hardware, because of the high computational cost required.

It could consider implementing some window adaptive algorithm, which alter the correlation window size according to the texture of the area to go.

Cross-checking: In addition to the consistency verification of left - right, it could be incorporated some other verification of the results to eliminate more false correspondences. For example, including the observation of the curvature of similarity to detect areas homogeneously textured and periodicals, or restriction of unity, this means that each point in an image can only be matched by a maximum of one from the other.

Color segmentation: It appears that it is a good approximation but the proposed methods require a search for the optimal parameters manually, i.e. it is necessary to look for the parameters that are best suited to each scene. Therefore, it would be interesting lines of investigation seek alternative algorithms, or some method to analyze the image to allow estimating the optimal parameters for segmentation.

Plane fitting: modeling the depth of each segment as a flat surface is an acceptable approximation to the actual depth of each segment, but the real depth of the objects is not flat. Therefore, it would be interesting to study techniques to adjust curved surfaces. On the other hand, once adjusted levels, study how to propagate information between segments belonging to the same object in the image and, thus, avoid discontinuities in depth, as an alternative to the anisotropic diffusion, could be a future line.

Point transference with the tensor trifocal: This method gives very good results in cases where the separation between the cameras is short. It could be explored alternative methods that provide greater flexibility in the positioning of cameras (i.e. N-view algorithms).

Regarding to the techniques based on Light Field Rendering, several possible courses of study could be opened, some suggests here:

Study compression techniques that exploit the great spatial redundancy, and achieve compression ratios that make available the transmission of the light fields.

These methods are very sensitive to that image arrays accomplish the epipolar constrain (images must be perfectly aligned both vertically and horizontally) to get good results in the synthesis. That is why deployment with real images is not robust. One might consider using methods of extracting features to correct deviations from the restriction during the synthesis.

BIBLIOGRAPHY

[1] O. Faugeras, “What can be seen in three dimensions with an uncalibrated stereo rig", *Computer Vision - ECCV'92*, Lecture Notes in Computer Science, Vol. 588, Springer-Verlag, pp. 563-578, 1992.

[2] R. Hartley, “Estimation of relative camera positions for uncalibrated cameras", *Computer Vision - ECCV'92*, Lecture Notes in Computer Science, Vol. 588, Springer-Verlag, pp. 579-587, 1992.

[3] G. Golub and C. Van Loan, *Matrix Computations*, John Hopkins University Press, 1983.

[4] R. I. Hartley. Lines and points in three views - an integrated approach. *In Proceedings of the ARPA IU Workshop*. DARPA, 1994.

[5] Shashua and M. Werman. On the trilinear tensor of three perspective views and its underlying geometry. *International Conference on Computer Vision*, 1995.

[6] Theo Moons, A Guided Tour Through Multiview Relations. *In SMILE*. 1998, pp 304-346

[7] O. Faugeras and T. Papadopoulo. A nonlinear method for estimating the projective geometry of 3 views. *Sixth International Conference on Computer Vision*, 1998 pp 477-484.

[8] P.H.S. Torr and A. Zisserman. Robust Parameterization and Computation of the Trifocal Tensor. *Proc. British Machine Vision Conference.* pp 655-664. 1996

[9] R. I. Hartley. In Defense of the Eight Point Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997* pp. 580-593.

[10] Heyden, *Geometry and Algebra of Multiple Projective Transformations*, Ph.D.thesis, Lund University, 1995.

[11] Triggs, ``The geometry of projective reconstruction I: Matching constraints and the joint image", *Proc. International Conference on Computer Vision*, IEEE Computer Soc. Press, pp. 338-343, 1995.

[12] R. Hartley, ``Computation of the Quadrifocal Tensor", *Computer Vision-ECCV'98*, Lecture Notes in Computer Science, Vol. 1406, Springer-Verlag, pp. 20-35, 1998.

[13] Eric Weisstein, “*Least Squares Fitting*”. In MathWorld—A Wolfram Web Resource, http://mathworld.wolfram.com/LeastSquaresFitting.html

[14] McMillan, L. “An Image-based Approach to Three-dimensional Computer Graphics”. Ph.D. Dissertation. Department of Computer Science, University of North Carolina at Chapel Hill, 1997.

[15] S. M. Seitz and C. M. Dyer. View morphing. In *Computer Graphics Proceedings, Annual Conference Series*, pages 21–30,Proc. SIGGRAPH’96 (New Orleans), August 1996. ACM SIGGRAPH.

[16] D. Scharstein. Stereo vision for view synthesis. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’96)*, pages 852–857, San Francisco, California, June 1996.

[17] S. Laveau and O. Faugeras. 3-D scene representation as a collection of images and fundamental matrices. Technical Report 2205, INRIA-Sophia Antipolis, February 1994.

[18] Avidan, S. and Shashua. M. ***"Novel View Synthesis by Cascading Trilinear Tensors".*** IEEE Transactions on Visualization and Computer Graphics, 4(4), October to December 1998, pp. 1077-2626.

[19] Adelson, E. H., and J. R. Bergen, ***"The Plenoptic Function and the Elements of Early Vision,"*** Computational Models of Visual Processing. Chapter 1, Edited by Michael Landy and J. Anthony Movshon. The MIT Press, Cambridge, Massachusetts. 1991.

[20] T. Wong, P. Heng, S. Or, and W. Ng. Image-based rendering with controllable illumination. In *Proceedings of the 8-th Eurographics Workshop on Rendering*, pages 13–22, St. Etienne, France, June 1997.

[21] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. Computer Graphics (SIGGRAPH'95), pages 39–46, August 1995.

[22] Levoy, M. and Hanrahan, P. "**Light Field Rendering**" Proceedings of SIGGRAPH 1996, July 1996.

[23] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *Computer Graphics Proceedings, AnnualConference Series*, pages 43–54, Proc. SIGGRAPH'96 (New Orleans), August 1996. ACM SIGGRAPH.

[24] Perona P. and Malik J. 1990. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence 12,7,629-63.*

[25] Marc Proesmans, Luc J. Van Gool, Eric J. Pauwels, André Oosterlinck: Determination of Optical Flow and its Discontinuities using Non-Linear Diffusion. ECCV (2) 1994: 295-304

[26] Guillermo Sapiro and Allen Tannenbaum. Edge preserving geometric smoothing of MRI data. Technical report, University of Minnisota, April 1994. Dept. of Electrical Engineering.

[27] Hartley R. I. and A. Zisserman. Multiple View Geometry in Computer Vision. CUP, Cambridge, 2000.

[28] Isgrò F and Trucco E 1999 Projective rectification without epipolar geometry. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Fort Collins, CO, pp. I:94-99.

[29] Hartley R 1999 Theory and practice of projective rectification. International Journal of Computer Vision 35(2), 1-16.

[30] Faugeras, O. (1993), *Three-Dimensional Computer Vision -- A Geometric Viewpoint*, The MIT Press, Cambridge.

[31] D.Scharstein, R.Szeliski, "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms," International Journal of Computer Vision, 47(1): 7-42, May 2002.

[32] H.Tao, H.S.Sawhney, and R.Kumar, "A Global Matching Framework for Stereo Computation," Proc. International Conference on Computer Vision, pp.532-539, 2001.

[33] L. Hong and G. Chen. Segment-based stereo matching using graph cuts. In CVPR04, pages I: 74–81, 2004.

[34] D. Comaniciu and P. Meer, "Robust analysis of feature spaces: color image segmentation," in Proc. of IEEE conference on Computer Vision and Pattern Recognition, pp. 750-755, 1997.

[35] P.F. Felzenszwalb and D.P. Huttenlocher. Efficient Graph-Based Image Segmentation. International Journal of Computer Vision, 59(2):167–181, 2004. http://people.cs.uchicago.edu/~pff/segment/
[36] Singular Value Decomposition (SVD) tutorial http://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm
[37] Light field Rendering- Stanford University. http://graphics.stanford.edu/papers/light/
[38] Stereo matching using belief propagation (2002) by J Sun, H-Y Shum, N-N Zheng In ECCV
[39] www.seas.upenn.edu/whatsnew/2001/tele-im2.html

ANNEX A. EXTERN LIBRARIES

OpenCV

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real time computer vision.

Example applications of the OpenCV library are Human-Computer Interaction (HCI); Object Identification, Segmentation and Recognition; Face Recognition; Gesture Recognition; Motion Tracking, Ego Motion, Motion Understanding; Structure From Motion (SFM); and Mobile Robotics.

OpenGL

OpenGL (Open Graphics Library) is a standard specification defining a cross-language cross-platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives. OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992[1] and is widely used in CAD, virtual reality, scientific visualization, information visualization, flight simulation. It is also used in video games, where it competes with Direct3D on Microsoft Windows platforms (see Direct3D vs. OpenGL).

OpenGL Utility Toolkit

The OpenGL Utility Toolkit (GLUT) is a library of utilities for OpenGL programs, which primarily perform system-level I/O with the host operating system. Functions performed include window definition, window control, and monitoring of keyboard and mouse input. Routines for drawing a number of geometric primitives (both in solid and wireframe mode) are also provided, including cubes, spheres, and the Utah teapot. GLUT even has some limited support for creating pop-up menus.

POSIX Threads Programming

In shared memory multiprocessor architectures, such as SMPs, threads can be used to implement parallelism. Historically, hardware vendors have implemented their own proprietary versions of threads, making portability a concern for software developers. For UNIX systems, a standardized C language threads programming interface has been specified by the IEEE POSIX 1003.1c standard. Implementations that adhere to this standard are referred to as POSIX threads, or Pthreads.

ANNEX B. HARDWARE DESCRIPTION

A pair of Logitech Quickcam Ultra Vision are used for the capture stage. Working at 640x480 (VGA) pixel resolution

Fig. 5.1 Logitech Quickcam Ultra Vision.

As computer a Desktop Dell Optiplex 755 with an Intel Core 2 Duo at 2.33GHz with 2 GByte of RAM.

The graphic card is a NVIDIA FX 350 with the following specifications:

Memory Size: 128 MB
Memory Interface: 64-bits
Graphic Memory Bandwidth: 6.48 GB/s
Graphics Bus: PCI Express

ANNEX C. COMPUTATION TIME

Time cost for each step of the application

Disparity Computation (80 levels, 15x15 correlation window)	1533.09ms
Color Segmentation (σ =0.6 K=300) Segments Left 141 Segments Right 146	2530.67ms
Plane fitting	Left = 84.1132ms Right = 71.5394ms
Anisotropic Diffusion	Left = 3316.55ms Right = 3319.86ms
Total stereo stage	9489.2ms
Time to create a new view	1090,49ms