Rectification and Pose Recovery for Spherical Images

by

Florian Kangni and Robert Laganière

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in
Electrical and Computer Engineering

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Abstract

Acknowledgements

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A possible illustration of a machine vision application is the camera pose recovery problem. It consists essentially in using visual information extracted from a sequence of images to estimate the motion parameters of each of the cameras involved in the capture process. As such, it fits in a the concept where visual information is used to solve a particular problem, concept that defines machine vision. The reason why this particular example was cited is the interest it represents for this thesis as it will be seen later. For the time being, it is just one of the many applications of machine vision. For that reason, it can be defined or described by one (or more) of the activities involved in a common vision problem and briefly described next.

The capture process is related among other things to the development of sensors and other devices that allow the user or the machine to capture the environment that will be studied by a subsequent process. The industry of cameras and other sensors has gained quite some popularity with the introduction of digital cameras, camera phones, for common use and panoramic and other complex shaped sensors for specialized markets.

The processing step is the most important one. It also where most of the research is done and where the variety in algorithms in quite noticeable. Processes vary with the problems they deal with and as a consequence are as diverse as the needs of the industry. They include applications such as visual processing with for example image enhancement and color correction or content-based analysis with feature extraction and pattern recognition. The main goal here is usually to design an algorithm as elegant, efficient, simple and portable as possible.

The output or display module is the most exposed and interactive aspect of a vision system since the encountered devices are used either to display results or are themselves

a result of the previous steps. This aspect is linked to all types of displays, screens, devices - microscopes, X-ray, LCD, projectors - etc.

Last but not least, the transmission aspect is as important as any of the previous activities. Less noticeable, it is a very specialized field that has had some popularity lately due to the high demand in multimedia data traffic on wireless networks. The industry in that domain mainly has to insure the safe transport and the fidelity in delivering the information of interest. This transported information is often coded and/or compressed after the capture step or before visualization.

All the previous activities are not usually considered independently, instead, they are quite often incorporated as parts of a solution, the ultimate goal being to give a machine or device the ability to deal with a problem autonomously literally using its vision of it.

The work presented here is part of the NAVIRE project[1] developed at the University of Ottawa. This project aims at achieving a virtual navigation in remote real environments that are rendered from captured panoramas. As such, this project fits all the aspects mentioned above concerning a machine vision problem : from the capture process all the way to the output that in this case is used for navigation purposes. As part of such a project, we are to investigate properties related to the type of images used for rendering and possibly develop efficient algorithms. We therefore focus essentially on the processing aspect despite a few notions about the capture process and abundant outputs. More specifically, we aim at establishing a formal study of spherical or 360^o panoramas through their geometry. This of course, makes great use of well established properties and methods especially as far as epipolar geometry is concerned. The validity of the theory that is developed throughout the chapters is tested by confronting it to problems of increasing size starting from plain images with the notion of rectification and ending up in the context of the pose recovery problem in the case of cubic panoramas. Ultimately, one could state that the presented study uses known and efficient tools to develop new tools which is, in our opinion, a very suitable definition of engineering.

1.1 Thesis Objective

This thesis addresses the problem of camera pose estimation in the context of image-based virtual navigation in remote environment. The ultimate goal is to recover the $3D$

[1] Visit the website http://www.site.uottawa.ca/research/viva/projects/ibr/ for more information

motion and structure from spherical images sparsely distributed over the scene of interest considering that no assumption on the camera locations and on the scene structure are taken into account. A progression is made from the classical epipolar geometry of planar images to pose recovery from spherical images.

1.2 Thesis Outline

In the first chapter, we will present a short introduction to epipolar geometry since the subject has been extensively described and heavily documents in a recent past. This will be done by giving the reader an explanation of the geometry of two images. We will mention the epipolar constraint and the epipolar entities that we judged pertinent for our study. Finally a section will be dedicated to a popular application of the epipolar geometry which is image rectification. The concept has been once more well documented through the years especially in the case of stereo images, but here we will add a contribution with our adapted algorithm to three views in horizontal and "L" configuration based on homography transfer. As it will be the case for all upcoming chapters, this introductory part is ended by some results to demonstrate the explained procedures.

In the second chapter, we will introduce the concept of cubic panoramas and their geometry. The work here is largely based on the preliminary study of epipolar geometry of the first chapter. The novel aspect is the type of images used for they are spherical panoramas thus essentially implying an implicit multi-camera system. The first section will present the capture process through a description of the sensor used in our experiments, the PointGrey Ladybug camera. Then will follows a brief explanation of the cube generation. The most important part of this chapter will be dedicated to a formal presentation of the geometry of cubic panoramas that can somewhat be seen as an extension of the classic epipolar geometry to some extent as mentioned above. An application of this will be the description of the concepts of the fundamental matrix and the essential matrix in the case of cubic panoramas as well as related epipolar entities : lines and planes.The final section of the chapter will discuss mainly the rectification process in the case of cubes as it is inspired from the epipolar image rectification. We will introduce our motivation for such an application and the methods that were developed to solve such a problem. A section on some pertinent results obtained during our study will end this chapter.

The next chapter - fourth - of our study will take us through the steps of a two-stage algorithm designed to solve the pose recovery problem in the case of spherical panoramas. This situation is the natural progression of the rectification process since it depicts a more general configuration with a random number of cubic panoramas instead of just two. The first stage is what will be designated as cube alignment. The alignment will consist of finding the optimal configuration of all cubes such that the only unknowns of the pose estimation problem are the inter-panorama translations. This is done in a bundle adjustment type of approach that finds the optimal aligning rotations for all cubes. The resulting configuration, with only translations as unknowns, is then used in the second stage that is naturally the translation estimation. This completes the pose recovery problem and ends the fourth chapter accompanied by results obtained on different sets of panoramas.

Our study ends with a general conclusion of our research especially as far as the cubic panoramas are concerned, as well as the possible ramification of the presented algorithms and our suggestions for future studies that align on this one.

1.3 Thesis Contributions

In the process of achieving the objective stated earlier, this thesis offers three original contributions :

- An epipolar rectification method for triplets of planar images based on a method presented in [21] and on the use of homography composition.

- A cubic panorama rectification procedure as an extension of the concept of rectification as presented in [13] as well as in [21]. This method has the advantage of being easily adaptable to any kind of spherical panorama.

- A pose recovery algorithm from spherical images. Using features tracked across all panoramas, one can extract the camera positions and elements of the scene structure in $3D$. It is partly based on the bundle adjustment approach described in [5].

1.4 Application to Virtual Navigation

It was mentioned earlier that this thesis was part of a much larger virtual navigation project. Thus, it is only suitable that it contributes to the latter project in some way. As a matter of fact, cubic panorama rectification and pose recovery applied to spherical images are both essential tools for some modules of the project. On one hand, cube rectification eases greatly the process of panorama interpolation necessary for smooth user navigation by providing rectified panoramas in a more favorable configuration. On the other hand, pose recovery allows one to locate, up to a scale, the panoramas with respect to one another if the capture process is not assisted by any positioning system such as GPS. Pose recovery from spherical images can also improve user navigation by maintaining a consistent viewing direction when switching from one panorama to another, given that all relative orientations can be estimated before hand. Overall, our work can improve the navigation of a user in spherical image-based environment.

Chapter 2

Epipolar Geometry and Projective Image Rectification

2.1 Introduction

The term epipolar geometry is generally associated to a configuration where a scene is being observed, captured, analyzed from 2 viewpoints (or more). The intricate relationships that exist between the corresponding points through the images - usually two in what is called stereo vision - and the scene structure is what really defines the latter term. In [14] resp. [8], it is defined as "the intrinsic projective geometry between two views" resp. "the basic constraint which arises from the existence of two viewpoints". One then uses these different constraints or relationships to compute or refine correspondences, disparity or to a further extent solve pose estimation problems for example. Ultimately, one can observe that the more views one has of a scene the better the knowledge of the "real" scene geometry is since the potential of information source is larger. This accordingly trades off with the number of existing constraints, the complexity and the efficiency of the methods to process the views.

The objective of this chapter is to present some of the basics of the epipolar geometry to establish a proper starting point for the study presented in this text. To achieve this, we will limit ourselves to the entities and properties that are of interest for our study. For a more complete insight in epipolar geometry, the reader is referred to [7, 14] that pursue the development far beyond the scope of what is needed here.

This chapter thus starts with a presentation of important algorithms and definitions as an introductory part of the study. State of the art algorithms such as the singular

value decomposition and the normalized direct linear transform are the main subjects of this section that is completed by definitions of terms such as projective geometry, homography, and camera model. These elements are the tools that, from that point on, will appear to be quasi omnipresent in the study presented here.

Next, follows a section that presents the concepts of epipolar constraint and fundamental matrix. Properties arising from the particular relationship between two views of a scene are summarized here and some entities such as the epipoles, the epipolar lines and the essential matrix are introduced.

Finally, rectification is cited as an illustration of the concepts presented in preceding sections. The idea is based mainly on a procedure described in [21, 13]. Extension of the concept to trinocular vision is also explored for some particular configurations and this will be the first example of multi-image algorithms throughout all the chapters.

2.2 Introductory notes and useful algorithms

2.2.1 Definitions

Projective geometry is the type of geometry that is often associated to the process of image formation. Naturally, it is the kind that is referred to in machine vision. Unlike the euclidian kind, distances are not necessarily conserved by a transformation under projective considerations. In an euclidian context, one could take the example of a square in a $2D$ plane. If a rotation or a translation is applied to the latter square, the result will be a square of same dimensions only in a different position. This is not the case in projective geometry. [8] gives the curious reader a more complete insight in projective geometry. We will limit ourselves to the latter concept in the case of image formation for a camera.

As far as the camera model is concerned, the one that is used in our study is the well known and very popular pinhole model. The camera is basically "reduced" to its trivial form : its optical center through which pass all captured light rays. For such a model displayed in Fig.2.1, the important entities are the distance to the image plane, the *focal length* noted f and the position O of the center of the camera frame in the image plane, with respect to the upper left corner of the image.

Under such these considerations, we can also use the example of the square mentioned earlier. Depending on the camera's location, the projection of the square on the image plane could be a square, an irregular quadrilateral or even a line as it seen on Fig.2.2.

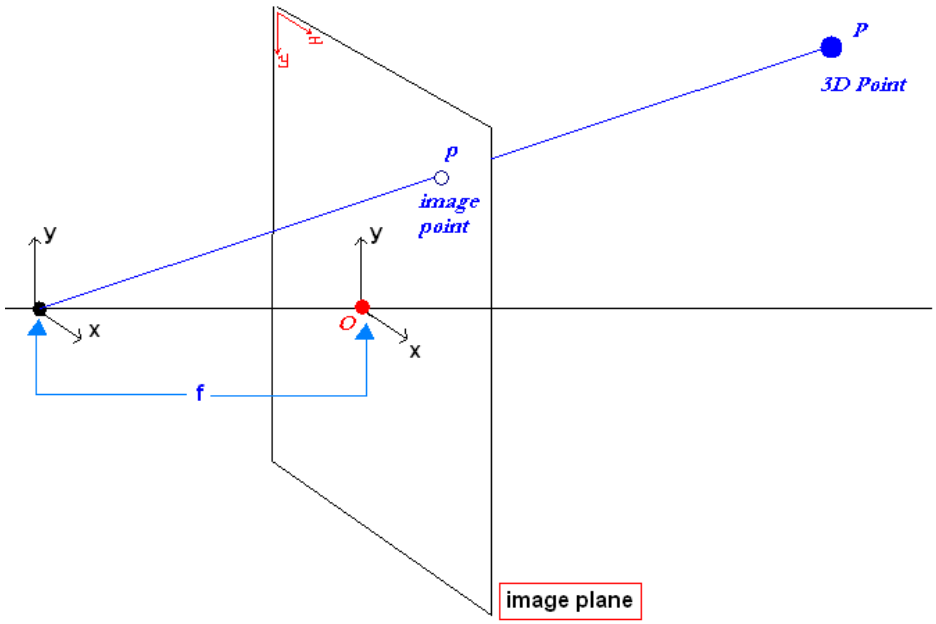


Figure 2.1: The pinhole model of a camera.

Thus intervenes the concept of projective transformation that actually defines quite well the idea of projective geometry. The process of image formation that was just described in Fig.2.1 and Fig.2.2 is an example of projective transformation where all points in the scene are projected onto a plane with respect to one unique point.

As it is given in [8], good illustrations of projective transformations are the planar homography and the calibration matrix. A planar homography H is a plane-to-plane transformation that could be anything from a simple rotation to a combination of skew, scale and rotation. It is often used to describe the relationship between two different projections of the same object onto two different destination planes as shown in Fig.2.3.

Points p_2 in image plane 2 are related to the points p_1 in image plane 1 particularly for the shaded planar section by a homography H such that:

$$(p_{2_x}, p_{2_y}, 1)^T = H(p_{1_x}, p_{1_y}, 1)^T \tag{2.1}$$

The equality in equation (2.1) is up to a scale due to the use of homogenous coordinates. $2D$ Homogenous coordinates are obtained as planar euclidian coordinates with a third added coordinate equal to 1 to take into the "scale" factor : $(p_x, p_y, 1)^T$ is essentially the same point as $(sp_x, sp_y, s)^T$ with s a non zero real number.

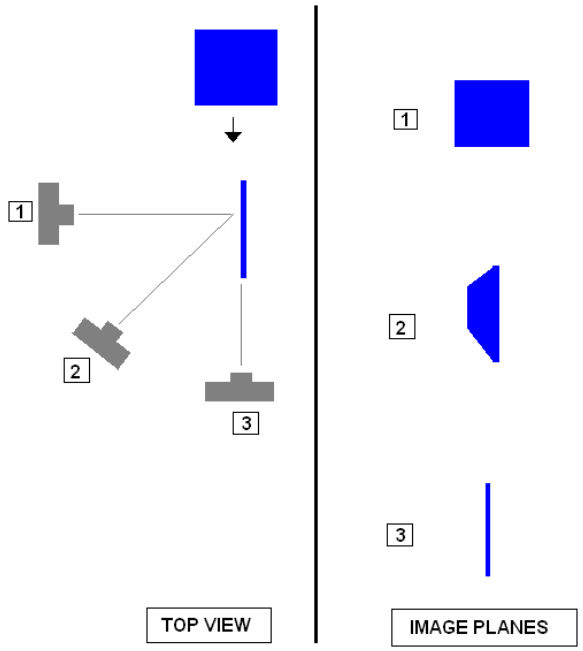


Figure 2.2: A square in different image planes.

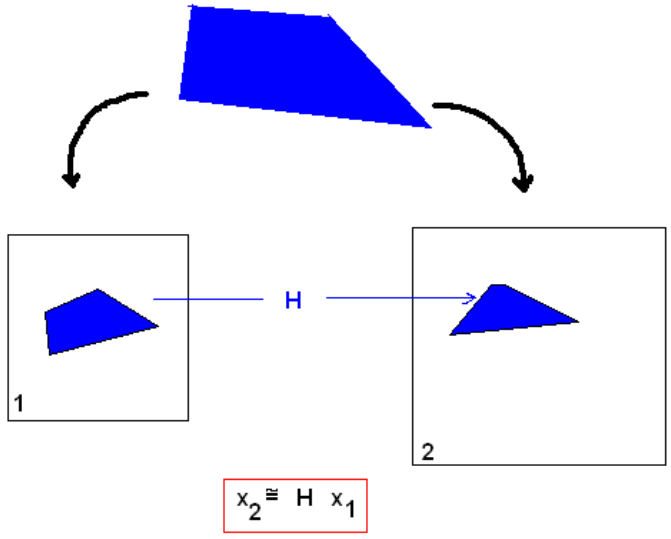


Figure 2.3: Illustration of the concept of homography.

Finally let us note that, the process of image formation described by Fig.2.1 is summed up by the *calibration matrix* K, also a projective transformation such that, for a point $P = (P_x, P_y, P_z, 1)^T$ of the scene expressed with respect to the camera frame coordinate - these coordinates are called *normalized camera coordinates* - its projection $p = (p_x, p_y, 1)^T$ onto the image plane is given by:

$$p = [K\ 0]P \tag{2.2}$$

The latter equality is up to a scale of course and K is explicitly given by:

$$K = \begin{pmatrix} -f & 0 & O_x \\ 0 & f & O_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.3}$$

Given that f is a positive entity and P_Z is always negative since the subject captured is always in front of the camera.

2.2.2 Algorithms

Singular Value Decomposition and Linear Systems

This section is essentially a summary of [1]. The singular value decomposition is also mentioned in [14, 8, 7, 27] where it is described as a simple and powerful tool for solving least squares problems. Let us first take a look at the algebra associated with this process.

For a given real m by n matrix A, there exists a decomposition in matrices U, S and V such that:

$$A = USV^T \tag{2.4}$$

Each of the decomposing matrices has particular properties. U and V are both orthogonal matrices meaning $U^TU = I_m$ and $V^TV = I_n$. Moreover, they are respectively constructed from the eigenvectors of AA^T and A^TA and as such are respectively squares m by m and n by n matrices. The matrix S is a m by n matrix with the only eventual non null elements on its diagonal; these are the square roots of the eigenvalues of AA^T or A^TA arranged in decreasing order. The elements of the diagonal in S are called the singular values and the column vectors associated to these values in U an V are consequently called left and right singular vectors. Singular vectors associated with null

or the smallest singular value are usually the main point of interest when solving a linear system of equations in a *least square sense*.

This decomposition is very often used in computer vision where linear system of equations are regularly encountered for example in epipolar geometry to compute the fundamental and essential matrices. The situation that requires the use of the *SVD* can be presented as a problem of the form (see estimation of the fundamental matrix, homography, essential matrix in [7, 8, 31]):

$$v_i^T H u_i = 0 \tag{2.5}$$

Where u_i and v_i are known vectors and the elements of H are sought after. Equation (2.5) expresses the fact that the problem defining constraint is verified by the known i^{th} pair of vectors. Usually many pairs (u_i, v_i) are given producing as many equations as there are pairs frequently resulting in over-determined systems. These systems are re-written as :

$$Ah = 0 \tag{2.6}$$

Where h are the elements of H represented in a single column vector. [1] mentions that optimal solution of (2.6), in a *least square sense*, with the constraint $|h| = 1$ is simply the right singular vector of A associated to the smallest singular value. The vector h that is found minimizes the norm of Ah and as such is the solution sought after. This very useful will appear frequently throughout this text and will be often referred to as the *SVD* solution.

The Normalized Direct Linear Transform algorithm

Equation (2.5) was said to represent a frequent problem in computer vision. As a matter of fact, from correspondences in two images or two frames or two spaces, an entity (for example a homography, a fundamental matrix, an essential matrix)is sought after that verifies a well known constraint. The explanation given by Hartley and Zisserman in [14] is summarized here.

The sets of points u_i and v_i involved in (2.5) must first be normalized. The normalization as indicated by [14] aims at transforming each set in a scatter of points with its centroid at position 0 of the space of interest. Moreover the average euclidian distance from a point to the centroid is to be set to $\sqrt{2}$. The reason behind this is the uniformiza-

tion of the weight of each point in the computation of the fore-mentioned entity and strongly recommended - with proof - in [14].

Let us take the example of points on two images planes expressed in projective coordinates $u_i = (u_{i_x}, u_{i_y}, 1)^T$ and $v_i = (v_{i_x}, v_{i_y}, 1)^T$. If $\bar{u}$ and $\bar{v}$ are the average points, [14] gives us the normalization transformations :

$$T_u = \begin{pmatrix} 1 & 0 & -\bar{u}_x \\ 0 & 1 & -\bar{u}_y \\ 0 & 0 & \sqrt{2} \end{pmatrix} \; ; \; T_v = \begin{pmatrix} 1 & 0 & -\bar{v}_x \\ 0 & 1 & -\bar{v}_y \\ 0 & 0 & \sqrt{2} \end{pmatrix} \tag{2.7}$$

T_u and T_v are applied to each point of the sets u_i and v_i resulting in new sets $\tilde{u}_i$ and $\tilde{v}_i$. The associated sought after entity is noted $\tilde{H}$ is computed using the *SVD* solution mentioned in the previous section. The final step of the normalized direct linear algorithm, that we will refer to as *normalized DLT* or simply *DLT* in the future, is the de-normalization of $\tilde{H}$ by applying the following formula :

$$H = T_v^{-T} \tilde{H} T_u^{-1} \tag{2.8}$$

H thus obtained is the entity that links the original sets of points u_i and v_i.

2.3 Epipolar Constraint and Fundamental Matrix

2.3.1 Fundamental matrix

Fig.2.4 summarizes the epipolar constraint. For two cameras, of respective centers C and C', observing the same point P of a scene (expressed with respect the camera frame in C), the respective images of the latter point are p and p'. In [8], the epipolar constraint states that the point p' has to lie on the projection of the ray sustained by the line through C and P in the camera of center C'. This is equivalent to the fact that, in [14], the same constraint is given as the fact that the rays CP, CP' and the line CC' are coplanar. This constraint is expressed by the *fundamental matrix* that links the points p and p' as follows :

$$p'^T F p = 0 \tag{2.9}$$

Moreover, if, as seen in Fig.2.4, the camera frames involved are separated by a translation t and a rotation R, and each of the cameras has a calibration matrix K and K' and

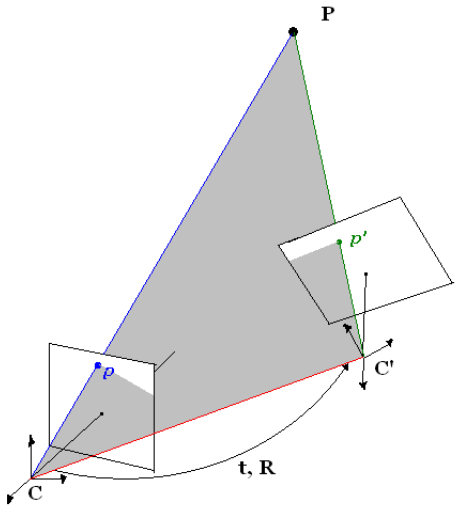


Figure 2.4: Epipolar constraint.

finally, the world coordinate frame coincides with the camera coordinate system centered in C, [8] and [14] establish an explicit expression of F:

$$F = K'^{-T}[t]_{\times}RK^{-T} \tag{2.10}$$

With $[t]_x$ defined as the antisymmetric associated to t:

$$[t]_{\times} = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 - t_x & \\ -t_y & t_x & 0 \end{pmatrix} \tag{2.11}$$

F therefore allows us to characterize the relationship between two cameras using images points. If instead, normalized coordinate points P and P' defined in (2.3) were used, combining (2.3) ,(2.9) and (2.10) results in:

$$P'^{T}EP = 0 \tag{2.12}$$

With E the essential matrix, concept fathered by Longuet-Higgins cited in [8, 14], and defined by:

$$E = [t]_{\times}R \tag{2.13}$$

E thus establishes the relationship between two cameras using normalized coordinate points. More details on the essential matrix will be presented in Chapter 3.

2.3.2 Properties

Earlier we mentioned that for the image point p and p', p' had to lie on the projection of the ray CP in the camera of center C'. The latter projection is the *epipolar line* l associated to p and computed as:

$$l = Fp \tag{2.14}$$

Another important entity associated to the fundamental matrix F is the *epipole*. Each of the camera possesses one epipole which is none other that the image point of the center of the opposite camera in the current camera. Noted respectively e and e', they are related to F in the sense they can be extracted as the right and left singular vectors of F of null singular value and as such:

$$Fe = 0 \text{ and } F^T e' = 0 \tag{2.15}$$

The epipoles, as we will see in later sections and chapters, are important entities involved in the process of rectification. Moreover let us note that, both epipolar lines and epipoles points are identical concepts for the essential matrix except that they are represented in higher dimension. Thus the epipolar lines become epipolar planes and the epipoles points become epipoles direction vectors.

Finally, F and E are computed following the same model i.e from matches automatically or hand-selected by the user and fed to the previously discussed *normalized DLT* algorithm. Note the similarity of (2.5) discussed in sections 2.2.2 and 2.2.2 with equations (2.9) and (2.12).

2.4 Epipolar rectification

The need for epipolar rectification is justified by the improvements it provides to classical machine vision application such as feature matching and disparity estimation. The objective of epipolar rectification is to create a configuration where the set of epipolar lines corresponding to a set of matches of a stereo pair is transformed into a set of lines that are either vertical or horizontal. Identifying corresponding points in both involved

images thus becomes a scanning and matching problem along the x or y direction which rather simplifies the feature matching process for example.

Many methods exist and have been implemented to solve the rectification problem for stereo and trinocular vision. Hartley worked on finding the rectifying transformation from the fundamental matrix with a strong mathematical justification [13]. Loop et al. developed a method to find the rectifying homographies and added some constraints to reduce the distortion introduced by rectification [18]. These methods are applicable to un-calibrated cameras and, in the case of two views, are close in theory to an algorithm presented by Mallon and Whelan [21]. The method they proposed follows Hartley's in principle but has its own original distortion reduction procedure. This approach is the one that is used in this section. One might then wonder how the problem of rectification extends to more than two views. We will limit our study to the case of 3 views mainly by interest for trinocular systems.

In the case of three views, Luping et al. presented a technique to rectify a triangular triplet of images using the perspective projection matrices (PPM) [2]. This technique uses camera calibration and is therefore not suitable for un-calibrated environments. Zhang et al. proposed a method to obtain the rectification homographies using the fundamental matrices, minimizing the distortion by adjusting 6 free parameters [30]. This method uses a set of three constraints on the triplet of images which allow the recovery of the three rectifying homographies in a closed form. Sun presents three methods that compute the projection matrices for the three images also using pair wise fundamental matrices [25]. The projection matrix of the reference image is a composition of 4 transformations; the other two are derived from the latter. In all these cases, the algorithm is designed for three views and uses three views constraints to achieve its goals.

The method presented here to solve the three view case is close to the one used in [30, 25] but is based on the method presented in [21]. This latter method uses the fundamental matrix in a similar way as in [13] but the novel aspect is the distortion reduction. It is a method developed for stereo. We therefore mainly describe how this 2-view algorithm can be adapted to the three view case in conjunction with an intermediate plane transfer by homography.

2.5 Projective rectification from the fundamental matrix

The algorithm presented in this chapter to solve the trinocular case is an extension of the method defined by Mallon and Whelan [21]. It is only suitable that the latter method is first described to the user. The objective of the authors was to obtain homographies that will simply be applied to each image to obtain its rectified counterpart, thus solving the stereo case. The rectification process, which somewhat follows Hartley's blueprint in [13], can be summarized as follows:

Fundamental matrix

First, one has to recover the fundamental matrix F using the 8 point algorithm mentioned in [31]. Eight or more matches are enough to compute the fundamental matrix. The Projective Vision Toolkit (*PVT* [29]) developed by Whitehead and Roth could be used to automatically find matches for a pair of images. In it latest version, it uses the Lowe's SIFT feature detector which provides large numbers of points [19].

These matches verify (2.9) and the *normalized DLT* algorithm mentioned earlier is used the estimate F. Note however that one additional step occurs before the denormalization : for the estimated *normalized* matrix $\tilde{F}$, its least singular value is forced to 0 to respect the rank 2 constraint.

Epipoles

At this stage, the epipoles e_{12} (in left image or image 1) and e_{21} (in right image or image 2) are extracted from an SVD decomposition of F. They are respectively given as right and left singular vectors of F associated with the null or least singular value as given previously in the introductory definitions section.

"Left" Homography H_1

From the epipoles, on can compute the rectifying homography H_1 to be applied on the left image by forcing the corresponding epipole to infinity in the horizontal direction i.e from $e_{12} = (e_{12_x}, e_{12_y}, 1)^T$ to $(e_{12_x}, 0, 0)^T$ in projective coordinates. H_1 is given as:

$$H_1 = \begin{pmatrix} 1 & 0 & 0 \\ -e_{12_y}/e_{12_x} & 1 & 0 \\ -1/e_{12_x} & 0 & 1 \end{pmatrix} \tag{2.16}$$

So that :

$$H_1 e_{12} = (e_{12_x}, 0, 0)^T \tag{2.17}$$

This is the first condition to obtain a rectified pair of images. As a matter of fact, this homography implies that all the epipolar lines corresponding to matches in the right image will all be horizontal and this is partially what is needed.

"Right" Homography H_2

Once H_1 is found, an additional constraint on the problem mentioned in [21] is used to solve for H_2. As a matter of fact, the fundamental matrix of the original setup being F, the resulting rectified fundamental matrix should equal to the trivial matrix F_h. [14] mentions this property when introducing the trivial stereo configurations particularly the one where both cameras differ only by a translation along the x axis. In such a configuration, both epipoles are projected at infinity in the x direction forcing the epipolar lines to be horizontal ultimately resulting in the fact a point in one image has its correspondent on the horizontal line of same y coordinate in the other image. Applying the suitable rotation and translation parameters t and R in (2.10) leads us to the following expression of F_h for such a trivial configuration:

$$F_h = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \tag{2.18}$$

Expressing the rectification in terms of the homographies H_1 and H_2 leads to the mathematical constraint by combining (2.1) and (2.9):

$$H_2^T F_h H_1 = \alpha F \tag{2.19}$$

With α a scale factor. We know F_h and have computed F and H_1. We want to solve for H_2 and α with :

$$H_2 = \begin{pmatrix} 1 & 0 & 0 \\ h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \end{pmatrix} \tag{2.20}$$

Equation (2.19) is transformed to a system of the type $AX = 0$ where X stands for elements h_i of the homography H_2 in a column with α as its last element . The system is then solve by using the *SVD* solution given in section 2.2.2. Steps 2.5 to 2.5 produce satisfying rectifying homographies with the restricting condition that original epipoles e_{12} and e_{21} do not appear within the left and right images respectively as noted by [21]. The last step of the method summarized in the present section is the distortion reduction introduced by [21]. It is an additional stage that improves the visual appearance of rectified images often severely distorted by the fore-mentioned process.

Distortion reduction

The distortion mentioned here is not related to lens distortion. It is "inserted" in the homographies after rectification. The reduction step is not mandatory but it makes the images look more natural as noted above. Essentially, the final transformations to be applied to each image of the pair are noted $K_i = A_i H_i$ with i = 1,2. The additional improving transformations A_i are of the form :

$$A_i = \begin{pmatrix} a_1^i & a_2^i & a_3^i \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.21}$$

The values of a_1 and a_2 are found by simplex minimization ($C++$ implementation Nelder-Mead or amoeba algorithm given in [22]) of the function:

$$f(a_1, a_2) = \sum_{i=1}^{n} \left[(\sigma_1(\mathbf{J}(\mathbf{K_i}, \mathbf{p_i})) - \mathbf{1})^{\mathbf{2}} + (\sigma_{\mathbf{2}}(\mathbf{J}(\mathbf{K}, \mathbf{p_i})) - \mathbf{1})^{\mathbf{2}} \right] \tag{2.22}$$

Where J is the jacobian of the transformation $K_i = A_i H_i$ at a point p_i contained in a grid over the image plane and σ_i are its singular values. Interestingly enough the jacobian J describes "the creation and loss of pixels as a result of the application of K" [21]. Note that the value of a_3 is left to the user for flexibility in centering the final image along the x axis since it only implies an horizontal offset.

This ends a summary of the method presented in [21] that allows the computation of rectifying homographies from the fundamental matrix of a stereo pair of images. This method was chosen for its simplicity and the distortion reduction stage that radically improves the visual aspect of resulting images. In the following sections, we present extensions of this latter method that ultimately lead us to the proposed solution to the trinocular case.

2.5.1 Extension to a "vertical" pair

As announced previously , this section describes one of the extensions added to the method presented in [21]. For a "vertical" pair of images, the process of rectification is very similar. By vertical we mean that the cameras of the stereo system have their centers located one above the other. In the classical case, the cameras are assumed to almost lie on the same horizontal plane. The differences in each step of the rectification are only due to the difference of configuration. For the vertical case, the ideal fundamental matrix F_v -and homolog of the previously introduced F_h in (2.18)- is given by [14]:

$$F_v = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \tag{2.23}$$

The modified rectification algorithm follows:

a. Recover the fundamental matrix F

b. Recover the epipoles e_{12} and e_{21} of the top and bottom images.

c. Recover the homography H_1 corresponding to the top image. Applying this transformation to the image sends the epipole e_{12} to infinity in the vertical direction : from $e_{12} = (e_{12_x}, e_{12_y}, 1)^T$ to $e_{12} = (0, e_{12_y}, 0)^T$ in projective space). Thus :

$$H_1 = \begin{pmatrix} 1 & -e_{12_x}/e_{12_y} & 0 \\ 0 & 1 & 0 \\ 0 & -1/e_{12_y} & 1 \end{pmatrix} \tag{2.24}$$

d. Using the same type of constraint as in the horizontal case (2.5), we obtain a linear system that is solved the same way using the same formulas but with H_1 and F_h replaced respectively by H_1 of equation (2.24) and F_v. This allows us to recover H_2 which in this case is of the form :

$$H_2 = \begin{pmatrix} h_1 & h_2 & h_3 \\ 0 & 1 & 0 \\ h_4 & h_5 & h_6 \end{pmatrix} \tag{2.25}$$

e. The distortion reduction step is exactly the same except the transformations are of the type A_i:

$$A_i = \begin{pmatrix} 1 & 0 & 0 \\ a_1^i & a_2^i & a_3^i \\ 0 & 0 & 1 \end{pmatrix} \tag{2.26}$$

This reflects the fact that the distortion and centering steps will affect the vertical coordinate and the user-defined value of a_3 corresponds to a translation along the vertical axis of the rectified image.

This modified version of the rectification procedure insures that, provided epipoles not within in each image, a point in one image will have its correspondent lying on the vertical line - its associated epipolar line - of same x coordinate in the other image. Both presented stereo rectification algorithms solve the trivial horizontal and vertical case for two images; they also help to solve some trivial trinocular cases when used suitably as shown in the next sections.

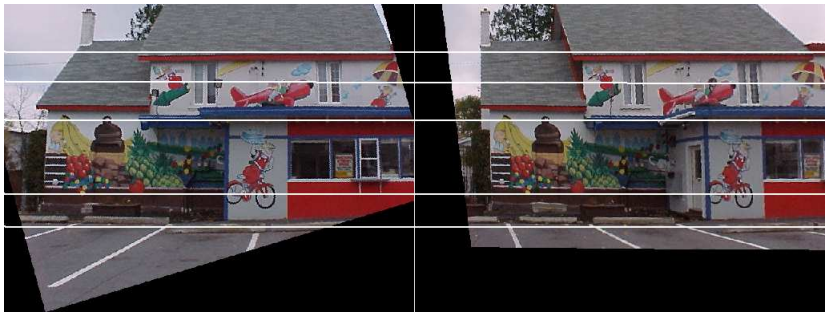
2.5.2 Preliminary Results

An example of image rectification for a horizontal stereo pair is displayed in Fig.2.5(c). The original images are shown in Fig.2.5(a) and the rectified versions with no distortion reduction in Fig.2.5(b). The visual improvement as well as the horizontal epipolar lines are easily observable.

The same is done for a vertical pair and some results are observable in Fig.2.6.

(a) Original horizontal pair

(b) Rectified pair with no distortion reduction



(c) Rectified pair after distortion reduction

Figure 2.5: Rectification for a "horizontal" stereo pair

(a) Original vertical pair (b) Rectified vertical pair

Figure 2.6: Rectification for a "vertical" stereo pair

2.6 Rectification of 3 views

The extension to triplets of images is different conceptually but uses the horizontal and vertical rectification at different stages. The algorithm presented in this section constitutes a first original contribution of this thesis. It has been presented at ICASSP 2006 [16]. The method has been developed as an introductory study to the problem of cubic panorama rectification presented in the next chapter. The concept is illustrated in Figure 2.7. The triplet is processed pair by pair therefore producing 4 homographies. The images are denoted 1,2 and 3. For 1 and 2, the rectification without the distortion reduction step gives us H_1 and H_2. Similarly, for images 2 and 3 the rectification without distortion reduction gives us H_2' and H_3. The rectification does not include the distortion since we want to stay consistent on the type of images we are working on : they are all affected by the same type of effects. The distortion reduction will therefore be the last phase of this process.

In both cases, unifying the results is the main objective. The solution is based on an attempt to find a common plane on which lie all rectified images affected by the proper type of rectification. For that, we explored an homography based solution that relies on composing the proper plane transformations on the suitable images to achieve our goal

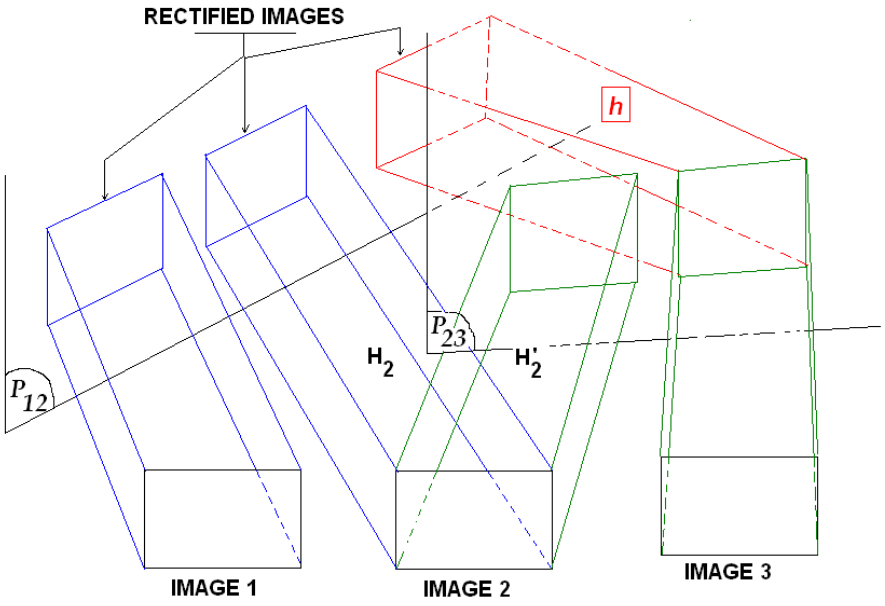


Figure 2.7: Rectification principle for a triplet of images.

as shown in Fig.2.7.

2.6.1 Horizontal triplets

The middle image 2 is common to the two pairs so we have H_2 and H'_2. Each of the computed homographies 'sends' the image plane 2 on two different planes containing respectively the rectified image 1 i.e $\mathcal{P}_{12}$ and the rectified image 3 i.e $\mathcal{P}_{23}$ (see Figure 2.7).

Our goal here is to find a way to transfer the plane $\mathcal{P}_{23}$ to $\mathcal{P}_{12}$; as a matter of fact we want to find the homography $\mathbf{h}$ between these two planes. This is done as follows:

- Image 2 is transferred to plane $\mathcal{P}_{12}$ with H_2
- Image 2 is transferred to plane $\mathcal{P}_{23}$ with H'_2
- Image 3 is transferred to plane $\mathcal{P}_{23}$ with H_3
- $\mathbf{h}$ between $\mathcal{P}_{12}$ and $\mathcal{P}_{23}$ is therefore given by $\mathbf{h} = H_2 H_2'^{-1}$: this is the "unification" mentioned earlier. Using the projection of the middle image in two different planes to deduct the relationship between both involved planes.
- Image 3 can therefore transferred to plane $\mathcal{P}_{12}$ by transiting through $\mathcal{P}_{23}$ using H'_3 given by :

$$H'_3 = \mathbf{h} H_3 = H_2 H_2'^{-1} H_3 \tag{2.27}$$

These steps essentially evaluate the homography H'_3 that sends the rectified version of image 3 to the plane containing the already rectified versions on image 1 and 2 by using the redundant data provided by the middle image. Finally, distortion reduction for the horizontal configuration is applied to each homography H_1, H_2 and H'_3 to insure that the y coordinates are left untouched.

2.6.2 "L-triplets"

The case of 'L'-shaped triplets is a combination of a vertical pair and a horizontal pair. All steps in the horizontal triplet procedure are repeated except for what follows:

- The pair 1, 2 is rectified using the vertical pair approach without the distortion reduction procedure (Section 2.5 to 2.5).

Figure 2.8: Original triplet of images : horizontal configuration

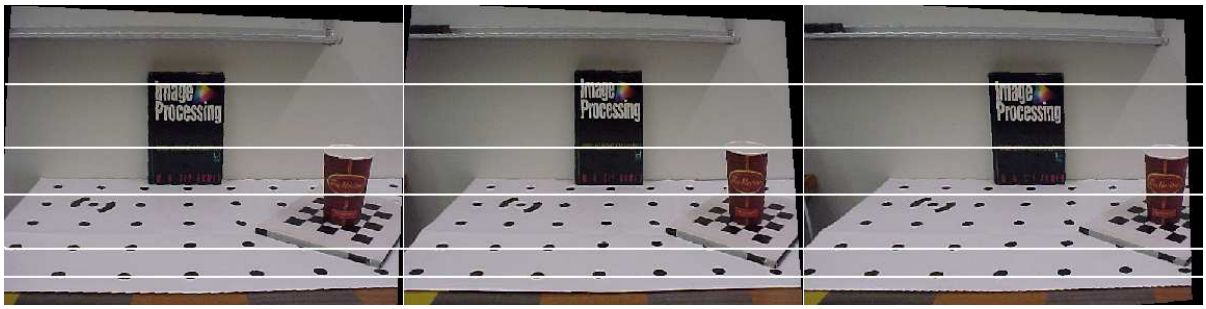


Figure 2.9: Rectified triplet of images : horizontal configuration

- The distortion rectification step uses the vertical distortion reduction approach for the rectified images 1 and 2. For image 3, the distortion reduction is also applied with the vertical approach described in section 2.5 to level the images 2 and 3 along the vertical axis.

2.7 Results and Observations for image triplets

For triplets of images, we have an example of a horizontal rectified triplet in Fig.2.8 with the original images in Fig.2.9. A few epipolar lines are drawn across the 3 images to show the consistency in the rectification process.

For comparison sake, the first example of "L"-shaped triplet is the same the one processed in [25]. The original triplet is shown in Fig.2.10. The result obtained in [25] are given in Fig.2.11. The result obtained using the homography-based approach presented in this chapter is given Fig.2.12. The desired epipolar lines are obtained in both cases. The effect of the distortion reduction is however well noticeable when comparing

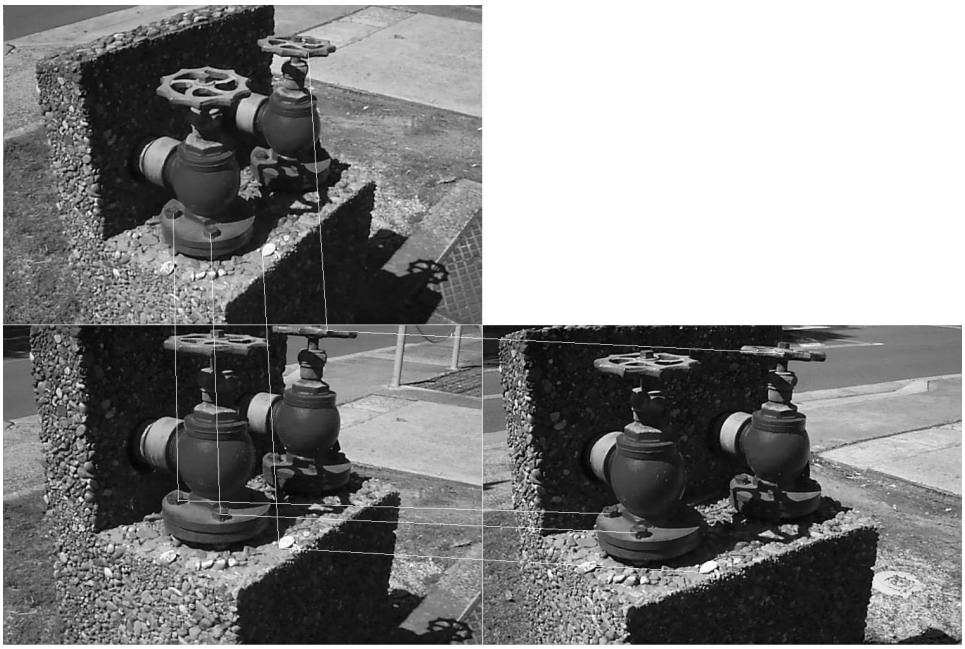


Figure 2.10: Original "L"-triplet (courtesy of C. Sun [25])

both results the set in Fig.2.12 looking less distorted and closer to the original images than the set in Fig.2.11.

Fig.2.13 and Fig.2.14 show another example of rectified "L"-shaped triplet of images.

2.7.1 Observations

An important observation mentioned earlier and in [21] is the fact that the rectification is ineffective for images where the epipoles appear in the image plane; suitable images are therefore to be used. This limitation concerning the capture process is not however detrimental to stereo system that usually use a quasi parallel setup for the image planes.

Another observation, that is rather obvious, is that a pair or triplet of images has to be taken close to the ideal configuration before using the corresponding rectification algorithm: i.e. it is impossible to rectify a vertical stereo pair of images with the horizontal stereo rectification approach.

Finally an important source of error is clearly the fundamental matrix approximation. For example note that well spread matches over the images help improve radically the fundamental matrix which otherwise ends up being very localized and valid only for a few points. It is therefore a very important step that should be handled with care and

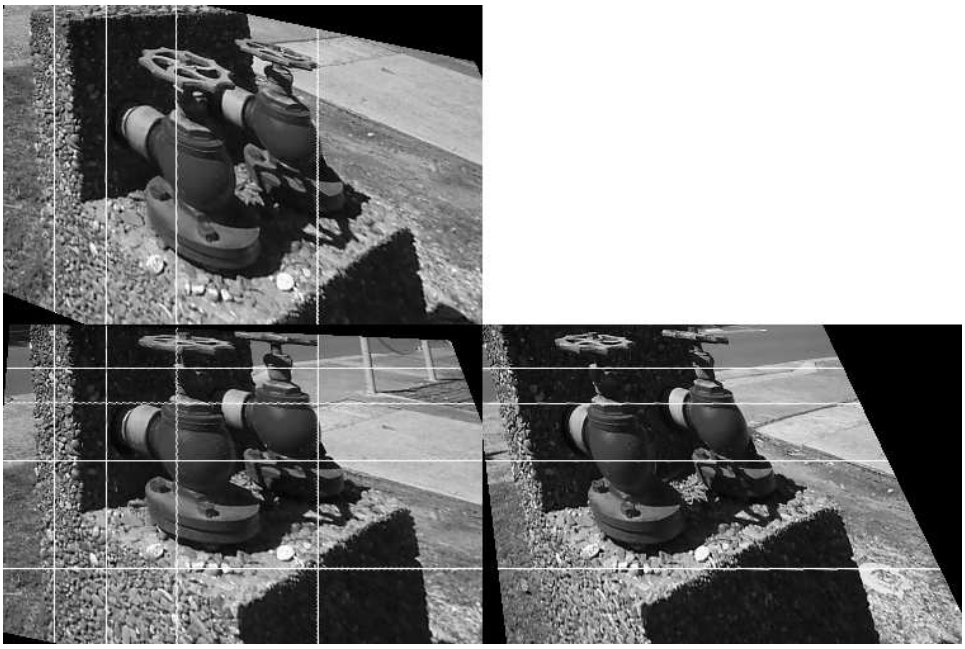


Figure 2.11: Results obtained by C. Sun (courtesy of C. Sun [25])

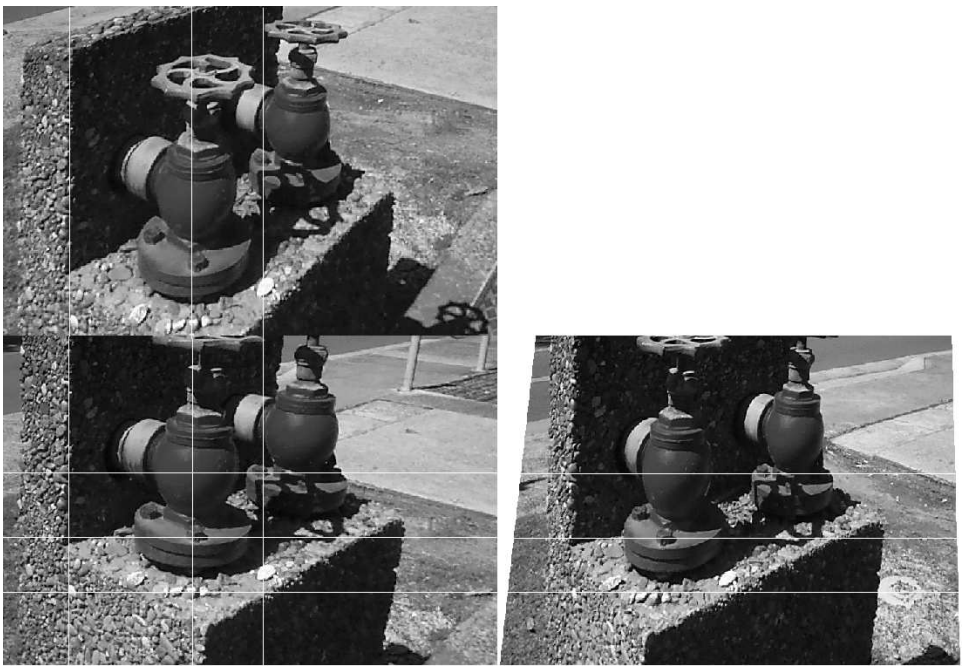


Figure 2.12: Results obtained by homography-based approach

Figure 2.13: Original triplet of images of the second example : L configuration

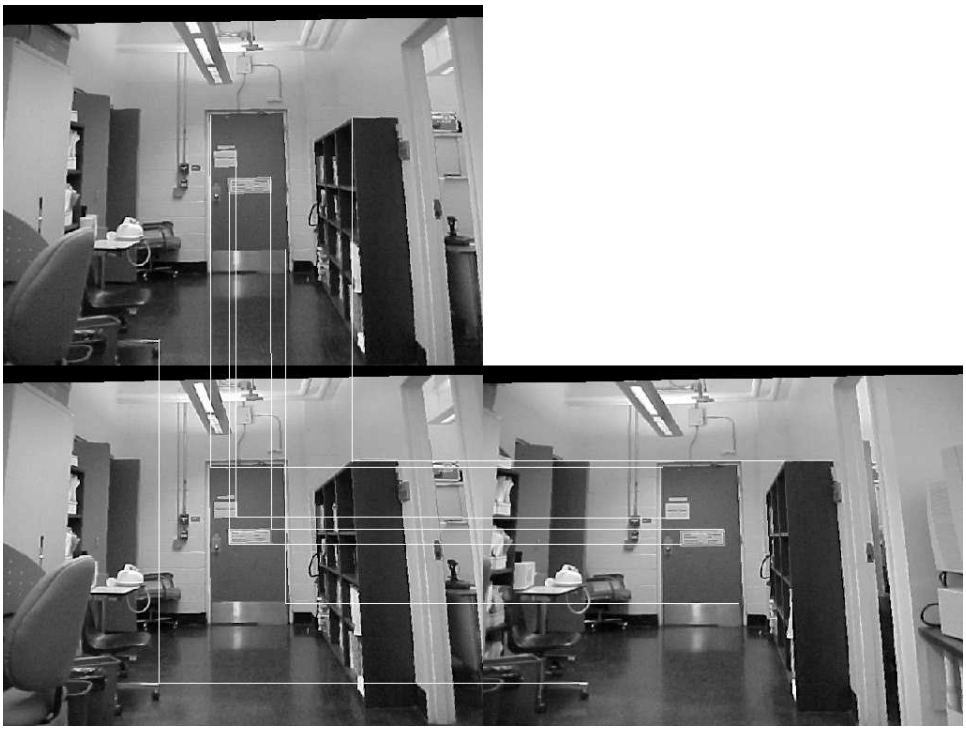


Figure 2.14: Rectified triplet of images of the second example : L configuration

carried out following one of the many existing techniques. For a set of algorithms, we suggest the reader to refer to [31].

2.8 Conclusion

This chapter presented an extension to an image pair rectification method based on fundamental matrix. Some notions related to the fundamental matrix and to algorithm used in the diverse computations were given to establish the basis of our work. Thus, definitions of epipolar entities were given. Moreover, important state-of-the-art algorithms were also summarized (*SVD* and *normalized DLT*). All this led us to the problem of stereo image rectification. For the classical horizontal pair configuration, an method developed in [21] was used.

The latter method was summarized and proved to have the advantage of being suitable for uncalibrated environments as well as producing a pair of rectifying homographies with a low distortion effect using solely the fundamental matrix. The abundance in the documentation concerning the computation of the fundamental matrix and the additional step of distortion reduction both proved to be the deciding factors in our choice of this rectification method as the basis of our algorithm. An extension to vertical image pair was also presented to set up the solution in the case of triplet of images.

Extensions to different three-view configurations were thus introduced in section 2.6. The approach presented , in the case of triplet of images, used a homography composition in order to rectify all images by projecting them on a common plane with the constraint of epipoles to infinity in the destination image plane. This proved to be a simple operation to carry out once the pair-wise rectifications were completed. The cases of horizontal triplets and "L"-shaped triplets were both treated.

Results were obtained on different sets of images and these were further visually improved when the proper distortion reduction was applied as the final step. A few observations were made as far as the performance of the basic stereo algorithm is concerned and the influence of matches and the fundamental matrix on the overall process.

Chapter 3

Cubic Panorama Rectification

3.1 Introduction

This chapter consists essentially in presenting the concept of epipolar geometry applied to cubic panoramas. Up to now, the reader has been introduced to different concepts and entities that help describe the geometry of a scene viewed under multiple angles particularly two and three. Cubic panoramas as their name indicate it are panoramic representations of a scene and as such, cover a 360 degrees viewing angle of that scene. This can be seen as a multiple camera configuration as well. In addition, panoramas intrinsically represent and carry more information than simpler images and are intuitively more meaningful to us when displayed in the right format since our vision system is "quasi" panoramic. Thus, in this chapter, we exploit the common traits between cubes and images and formally adapt and extend the idea of epipolar geometry, fundamental matrix and essential matrix.

Cubic panoramas or *cubes* have some embedded characteristics that make them quite attractive as a format. As a matter of fact, cubes can be seen as sets of images generated by 6 cameras in a particular configuration. This allows the stereo algorithms to carry over in quite a straightforward way to cubes. Moreover, their structure implies a simple calibration that does depends only on one variable. Finally, [9, 10] shortly mentioned the hardware advantages of the cubes with a graphic processor : a panoramic image with a work-around with 6 simple images is quite an efficient representation for further processing.

The first section of this chapter will introduce the cube capture and generation processes. This short paragraph is meant to address the curiosity of the reader on the kind

of sensor used in our experiment : the PointGrey Ladybug. We also discuss very briefly the cube generation procedure designed by *M. Fiala* of the National Research Council Canada (NRC).

Next, some interesting properties of the cube are explained as an introduction to the subsequent sections. The concepts of general $3D$ coordinates conversion to cube coordinates, face homography and cube calibration are the subjects of this preparatory section to cube geometry.

The following section describes the adaptation of the fundamental matrix to the cubic panoramas. Faces in correspondence in a pair of cubes are treated as a stereo system. Extensively discussed and well known algorithms are then applied to the cubes to demonstrate the epipolar constraint applied to the latter through the observation of epipolar lines.

In a natural succession, we present the essential matrix in the case of cubes. The epipolar constraint and trivial calibration of the panoramas lead to the establishment of the essential matrix for a given pair. Some results are also shown through the observation of the epipolar plane and lines.

Finally, an application illustrating the concept of essential matrix and more generally the epipolar geometry of cubic panoramas is discussed in the last section. Cube rectification extends the concepts of image rectification to the $3D$ space by using a similar approach with some interesting differences and results. The goal here is mainly to prove the validity and reliability of the previously established cube epipolar geometry.

3.2 Cubic panoramas : capture and generation

Cubic panoramas being the subject of the study it is necessary to provide some elements on the way they are obtained. The capture of the images that are composed into panoramas is done using the Point Grey Ladybug camera extensively presented in [24, 11] and shown in Fig.3.2(b). It is essentially a camera composed of 6 sensors (1024×768 pixels each), 5 laterals and 1 pointing upwards that capture a view of the world at 360 degrees around the azimuth completed by a top view.

Since the camera's sensors have been accurately calibrated, it is possible to fuse the six images to form an almost complete spherical panorama (see Fig.3.1). This panorama can therefore be considered to have been produced by a central projection camera that collects all light ray coming from all directions, incident on a point in space. The resulting two-dimensional plenoptic function can then be re-projected on any type of surface :

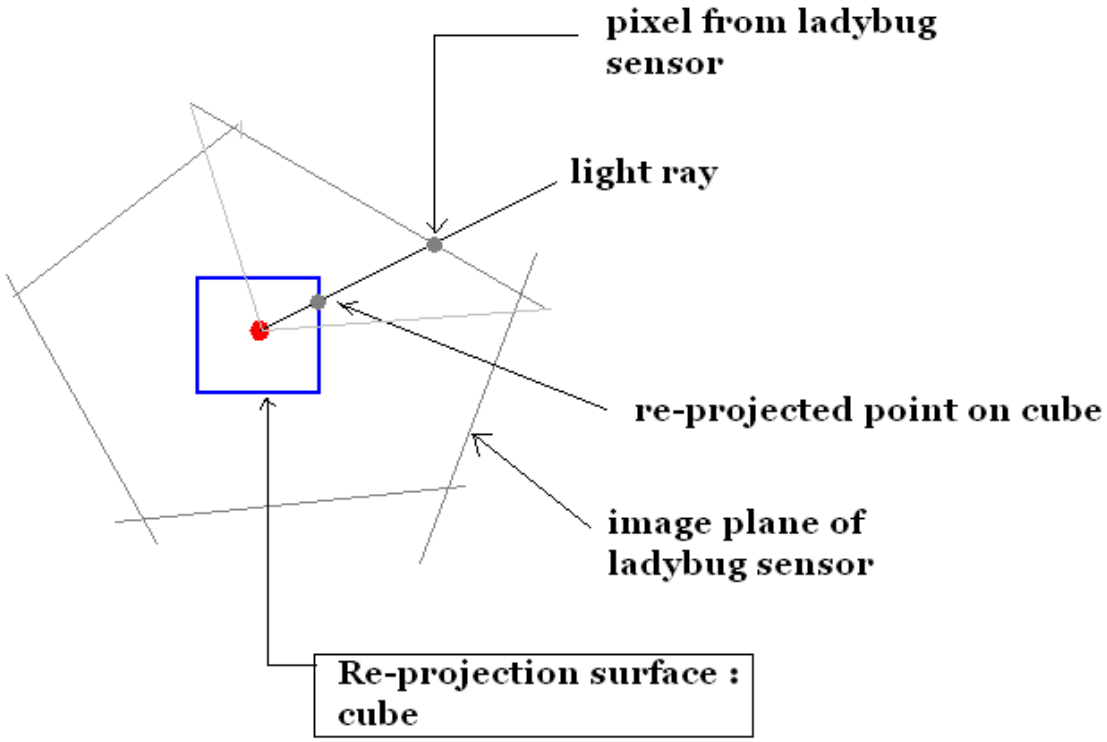
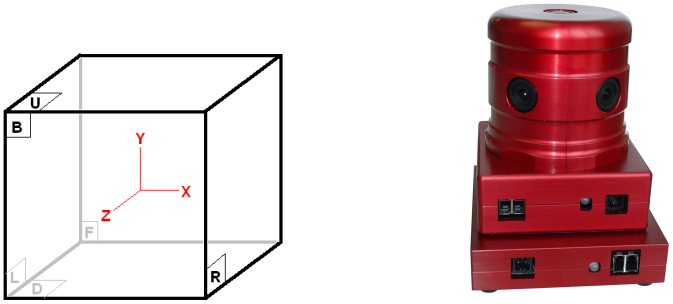


Figure 3.1: Fusion of the camera views to generate a cubic panorama

sphere, cylinder, cube, etc. We use here a cubic representation that have been shown to be easily manipulable and that can be stored and rendered very efficiently on standard graphic hardware [4]. This work on the cube generation has been done by M. Fiala from the NRC in [11].

We extended this procedure by allowing a user to specify a rotation matrix and obtain as an output the corresponding rotated cube. It remains simply a matter of querying for the color with the correctly computed light ray. The correspondence between pixels on cube faces and corresponding $3D$ light rays through the center of the cube is explained further in section 3.4 and in more details in appendix A

A cube can be seen on Fig.3.3 laid out in a cross or flat pattern with the faces in the order (from top to bottom and left to right): up, left, front, right, back, down. The reference frame chosen in our study is the standard openGL frame that can be seen inside a cube on figure 3.2(a) with x axis pointing toward the"right" face, the y axis toward the "up" face and the z axis consequently pointing toward the"back" face.



(a) Cube reference frame.

(b) The Point-grey Ladybug camera.

Figure 3.2: Cube frame and camera used in cube capture.

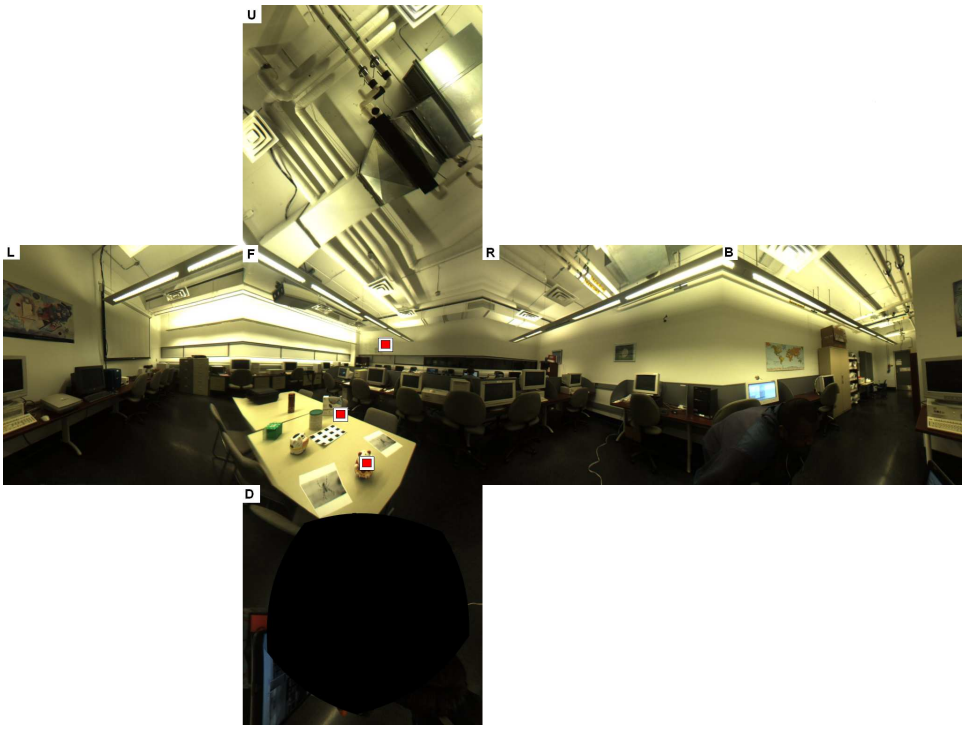


Figure 3.3: Cube laid out in cross pattern.

The fact that a cubic panorama is effectively made of six identical faces, each of them acting as a standard perspective projection camera with 90^o field of view, makes the representation very convenient to handle; all standard linear projective geometry concepts still being applicable.

3.3 Notations

We introduce some notations and conventions use throughout this chapter for clarity sake. Let us consider two cubes (C) and (C'). For each cube, a tag i in the set $\{\mathtt{U}, \mathtt{L}, \mathtt{F}, \mathtt{R}, \mathtt{B}, \mathtt{D}\}$ is given to each of the faces with `U` standing for the "up" face, `L` for the left face and so on. For a pair of faces in correspondence, the associated fundamental matrix is noted F_i where i is the tag of the faces. A 3D point is noted $\mathbf{X}$ and is equivalent to $(X, Y, Z)^T$. Its projection on the face i of the cube is noted $\tilde{\mathrm{x}}_i = (x_i, y_i, 1)^T$. $R(\theta)_x$ stands for a rotation around the axis x of amplitude θ and t stands for a translation vector. We will note P_i the projection matrix for a given face and K the common calibration matrix since the 6 faces have identical characteristics. We can therefore write: $P_i = K[R_i|t_i]$ following the model in [14]. As a reminder, the projection matrix allows us to obtain the image coordinates of any $3D$ point by a simple multiplication operation, provided the calibration and the extrinsic parameters of the camera are known, that is

$$x_i = P_i\mathbf{X} \tag{3.1}$$

3.3.1 Exponential representation of rotations

Rotations are important geometric entities referred to quite often in this study. [7, 17] present a complete insight into what is needed to be known in machine vision about these particular 3 by 3 matrices of the special orthogonal group also called SO(3). A plethora of sources will confirm that there exist many representations of rotations. One of the most popular being the usual 3 by 3 matrix M verifying : $MM^T = I_3$ and $det(M) = +1$ (I_3 is the 3 by 3 identity matrix). In kinematics, as well as machine vision where the problem of estimating a rotation often occurs, the exponential representation is a popular choice. It is closely related to the concept of antisymmetric matrix and the Rodrigues formula.

Given a 3D vector $v = (v_1, v_2, v_3)$, [7, 14] describe the associated antisymmetric matrix associated as :

$$[r]_\times = \begin{pmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{pmatrix} \tag{3.2}$$

For a rotation of non zero angle θ around the axis of unit vector v, the related Rodrigues vector ω is defined in [17] by :

$$\theta = |\omega| \tag{3.3}$$
$$v = \frac{\omega}{\theta} \tag{3.4}$$

To end this section, a rotation R of associated Rodrigues axis $\omega = \theta v$ can be represented thanks to an exponential notation as follows :

$$R = I_3 + sin\theta[v]_\times + (1 - cos\theta)([v]_\times)^2 \tag{3.5}$$

Rotation estimation problems in this text - mainly Chapter 2 and 3 - will make extensive use of this representation. One of the main advantages of such a notation is that 4 parameters are estimated instead of 9 for the matrix representations, even if the complexity of the problem in the case of the matrix is decreased by the special orthogonality constraint. For more details on the $SO(3)$ group, we suggest consulting [3, 7, 14].

3.4 Properties

3.4.1 Calibration Matrix

Further in this study will appear the need to define the calibration matrix K associated with each sensor for each face of the cube. One of the main advantages of the cubes, as mentioned earlier in section (3.1), are the constraints that are linked to it. If L is the size of a cube in pixels - L is set by the user at the generation stage - we can easily deduct the fore mentioned calibration matrix K in the ideal case. The image plane is at a distance $\frac{L}{2}$, the projection of the camera center in the image plane is always at $(\frac{L}{2}, \frac{L}{2}, 1)$. The ideal focal distance of all 6 cameras (one for each face) is also $\frac{L}{2}$. Thus we can write :

$$K = \begin{pmatrix} -\frac{L}{2} & 0 & \frac{L}{2} \\ 0 & \frac{L}{2} & \frac{L}{2} \\ 0 & 0 & 1 \end{pmatrix} \tag{3.6}$$

This notation is with respect with the classical pinhole model of a camera. The same model is obviously used for each virtual camera associated to each face. (3.6) will prove to be extremely useful throughout our study despite the fact that the approximation to the ideal case was made here. The accuracy of the solutions will prove not to suffer at all from this hypothesis.

3.4.2 Conversion Procedure

In this section we mention a simple and useful method to convert any ray to its cube coordinates with respect to the cube frame shown previously in Fig.3.2(a). For any ray guided by a given vector through the origin of the frame, its cube coordinates can be found by intersecting the direction of the ray with each image plane standing for each cube face. This will provide us with two possible intersections over the cube. A test of direction consistency then allows us to recover the proper point on the cube and therefore obtained the coordinates that are sought after.

How is this useful in our study ? This process is used especially when applying a rotation on a given cube. After we apply a given rotation to a point of the cube, nothing guarantees that the new coordinates are those of a point on the surface of the cube. It is usually a vector placed in the proper direction but not on the cube. To obtain its correspondent on the cube frame, we therefore have to apply the conversion procedure mentioned here to recover the proper point and thus the new coordinates of the rotated point or pixel. The mathematical details of such a procedure are given in appendix B.

3.4.3 Homography between two faces

It was mentioned in the previous section that a ray going through the center of the cube, intersects exactly 2 faces within their viewing area. As a matter of fact, considering the infinite extent of each of the images planes (for each face), such a ray actually has an intersection with each of the latter provided we consider loosely the notion of intersection at infinity in case of parallelism ray-plane. It is important to note that, unlike the case of procedure mentioned in the previous section, the resulting point is not necessarily constrained within the viewable area. Fig.3.4 displays such a situation. The 2 exact viewable intersections are the points A and B but we can see that the ray of interest here, through the center C also intersects the extended planes sustaining the top and bottom faces at the points D and E respectively, points that are clearly out of the faces themselves.

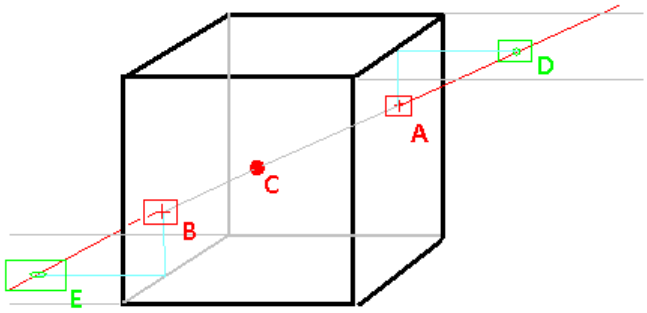


Figure 3.4: Intersections ray-cube.

With that in mind, we are interested to know, for any point of a given face, where its correspondent lies on the *plane* sustaining any other face. Faces being planes to some extent, we are therefore simply trying to find the transformation that transfers us from the plane of one face to the other and that provides us with a 1 to 1 point correspondence : this is an acceptable definition of a homography that is none other than a plane transformation. This will prove very useful in the study of fundamental matrices over the cubes particularly when it will come to computing epipolar lines over the cubes. This will be later discussed in the next section.

Recall the notations in section 3.3. Let us consider in this section the faces $f_{\mathtt{F}}$ and f_j for any j in $\{\mathtt{U}, .., \mathtt{B}\}$.Without loss of generality, if we consider the world coordinate system to be attached to the center of the cube with axis 'aligned' with the front face $f_{\mathtt{F}}$, the respective projection matrices for $f_{\mathtt{F}}$ and f_i are the following:

$$P_{\mathtt{F}} = K[I_3|0] \tag{3.7}$$

and

$$P_i = K[R_i|0] \tag{3.8}$$

With I_3 the identity matrix of order 3. The projection matrices P_f and P_i differ only up to a rotation R_i since the focal center is the same for all faces. Note that for all i, $R_i = R(\theta)_{axis}$ with θ in the set $\{\frac{-\pi}{2}, 0, \frac{\pi}{2}, \pi\}$ and axis standing for x, y, or z depending on the face : for example $R_r = R_x(\frac{-\pi}{2})$. For the detail of the other matrices consult appendix C. For a point X in space, its projections are $x_{\mathtt{F}}$ and x_i and are given by :

$$\tilde{\mathrm{x}}_{\mathtt{F}} = P_{\mathtt{F}}\mathbf{X} \tag{3.9}$$

and

$$\tilde{\mathrm{x}}_i = P_i\mathbf{X} \tag{3.10}$$

From (3.7) and (3.9) we can extract an expression for $\mathbf{X}$ as done in [14]:

$$\mathbf{X} = \begin{pmatrix} ZK^{-1}\tilde{\mathrm{x}}_{\mathtt{F}} \\ 1 \end{pmatrix}$$

Replacing this expression of $\mathbf{X}$ in (3.10) we obtain :

$$\tilde{\mathrm{x}}_i = ZKR_iK^{-1}\tilde{\mathrm{x}}_{\mathtt{F}} \tag{3.11}$$

Let us note :

$$H_i = KR_iK^{-1} \tag{3.12}$$

As a consequence, (3.11) becomes :

$$\tilde{\mathrm{x}}_i = ZH_i\tilde{\mathrm{x}}_f$$

which in projective space is equivalent to (notice Z is scalar) :

$$\tilde{\mathrm{x}}_i = H_i\tilde{\mathrm{x}}_{\mathtt{F}} \tag{3.13}$$

In the general case, i.e between any two faces i and j, the previous equation applied to $\tilde{\mathrm{x}}_i$ and $\tilde{\mathrm{x}}_j$ allows us to write :

$$\tilde{\mathrm{x}}_j = KR_jR_i^{-1}K^{-1}\tilde{\mathrm{x}}_i = H_{ij}\tilde{\mathrm{x}}_i \tag{3.14}$$

with :

$$H_{ij} = KR_jR_i^{-1}K^{-1} \tag{3.15}$$

To conclude this section, we have established a formula (3.14) that allows us to transfer any point of the cube to its projection through the center of the cube to the plane sustaining any given face.

3.5 Fundamental matrices and Epipolar lines

Let us start by mentioning that many elements of the first chapter related to epipolar geometry for two images appear in this section. As a matter of fact, this section can be seen as an extension of the concept of fundamental matrix to the case of cubic panoramas. The formulation of the problem is somewhat different and some constraints need to be considered to be able to transfer the methodology deriving from conventional stereo configuration. For any point on a cube, we want to define on what hyper-plane lies its correspondent in the other cube. The main constraint in this approach is the necessity to view the pair of cubes as 6 stereo image pairs. This means that the portion of the scene visible in one face is partly or ideally entirely visible in the corresponding face of the other cube to ease matching and avoid intermediate re-projection steps as it will be explained next. It is important to note that this part of the study gives us a starting point for our study of cubic epipolar geometry by establishing a good theoretical and mathematical basis illustrated by interesting preliminary results : if a pair of cubes is a system of 6 stereo pairs therefore generating 6 fundamental matrices, is there any relation between these matrices given the intrinsic structure of the cube ?

It is obvious that one of the fundamental matrices at least is needed to start answering this interrogation. First let us note that the standard 8 point algorithm mentioned in [12, 13, 31] and briefly discussed in the first chapter, is used to compute the "generating" fundamental matrix between the faces of concern. As mentioned in the preceding paragraph, we chose to have a slight constraint on the cubes consisting in corresponding faces displaying enough overlap for an easier matching process. In case the matches are not all on corresponding faces, a simple and direct remedy to this situation is to apply (3.14) to the concerned matches to re-project them on the right plane resulting in matches all in the same face plane, and this, for both cubes.

The matches are extracted from - possibly - corresponding faces manually or using the PVT Tool described in [29]. The rest of the procedure explained below is carried on once one of the possible 6 fundamental matrix is computed. The previous relationship (3.13) established between projections of a $3D$ point on 2 faces can be combined to the fundamental matrix expression. For 2 faces $i \in (\mathtt{U}, \cdots, \mathtt{B})$ in correspondence in 2 cubes, we have the following form of the epipolar constraint :

$$\tilde{\mathrm{x}}_i'^T F_i \tilde{\mathrm{x}}_i = 0 \tag{3.16}$$

Where $\tilde{\mathrm{x}}_i$ and $\tilde{\mathrm{x}}_i'$ are matches on the i face of respectively C and C'. Finally, (3.13)

in (3.16) yields :

$$\tilde{\mathrm{x}}_{\mathtt{F}}'^{T} H_i^T F_i H_i \tilde{\mathrm{x}}_{\mathtt{F}} = 0 \tag{3.17}$$

From which we conclude :

$$F_{\mathtt{F}} = {H_i}^T F_i H_i \tag{3.18}$$

or :

$$F_i = {H_i}^{-T} F_{\mathtt{F}} {H_i}^{-1} \tag{3.19}$$

With $H_i = K R_i K^{-1}$ as defined in (3.12). A general equation can be found as in the case of the homography to be able to swap between any two faces i and j when considering their fundamental matrices. We thus have what follows :

$$F_j = H_{ij}^{-T} F_i H_{ij}^{-1} \tag{3.20}$$

with H_{ij} defined in (3.14)

Thus, given that we are able to compute one of the possible 6 fundamental matrices for a pair of cubic panoramas, we have proved that it is also possible to recover all the other fundamental matrices. An illustration of this interesting results is the computation of the epipolar lines over the cubes.

As a matter of fact, let us consider two cubic panoramas C and C'. For any point $\tilde{\mathrm{x}}_i$ on a face i of C, it is possible to recover the corresponding epipolar line on face i of C' by simply applying the basic formula : $l_i = F_i \tilde{\mathrm{x}}_i$. For the other lines -exactly 3 since a plane *through the center* of a cube intersects the latter in 4 lines - the point $\tilde{\mathrm{x}}_i$ is re-projected onto all the remaining faces of interest using 3.14 resulting in points $\tilde{\mathrm{x}}_j$ and associated epipolar lines given by the same formula as previously :

$$l_j = F_j \tilde{\mathrm{x}}_j = F_j H_{ij} \tilde{\mathrm{x}}_i \tag{3.21}$$

This formula is actually also valid for the "generating" face i since $H_{ii} = I_3$. (3.21) is therefore a general formula to obtain the epipolar lines associated with a given point provided that one of the fundamental matrices was recovered before hand causing all matrices to be easily computable. Some examples of lines associated to points for a pair of cubes are displayed in Fig.3.7 as junctions of white segments. The points marked by red squares on one panorama and the epipolar are the white lines in the other panorama.

Note the consistency of the geometry since all lines are joined when considering adjacent faces. This is an illustration of the epipolar plane that will be mentioned in the next section in more details : one point is associated to a set of lines due to the fact that it is actually associated to a plane through the center of the other cube !

3.6 Essential Matrix and Epipolar plane

In the previous sections we established the epipolar geometry based on a stereo approach that relied on a calibration matrix considered ideal and on the restraining consideration that we were dealing with 6 stereo pairs. The notion of essential matrix is introduced here for many reasons and helps making the study of the geometry simpler as it is a compact form of all relationships for all faces. It also a way to reinforce the geometry already established with fundamental matrices as all approaches should fuse in a single, uniform, consistent and elegant solution.

In this section we mainly try to define properly the simplest relationship in a cubic panoramas correspondence. For two images, we were introduced to the link point-line. For two cubes, we had in the previous section the link point-lines. Here we establish the link point-plane and show the consistency with the previous method. Only here, the process proves to be simpler and more efficient. It will therefore become our tool of choice when dealing with cubic panoramas epipolar geometry. First some more justifications on the study of the essential matrix will be given, then will follow some computation details. This section ends with some results to illustrate the methodology adopted.

3.6.1 Essential matrix vs. Fundamental matrices

The fundamental matrix of two cameras, as we have seen in the previous chapter, is given by solving the epipolar constraint provided some matches are known (at least 8):

$$\tilde{\mathrm{x}}'^{T} F \tilde{\mathrm{x}} = 0 \tag{3.22}$$

If the rotation and the translation between both cameras are respectively R and t, K being their common calibration matrix, the expression of F was given as :

$$F = K^{-T} t_{\times} R K^{-1} \tag{3.23}$$

With $t_{\times}$ the antisymmetric matrix derived from the vector t. Recall it is given by (3.2).

By definition [14, 31], the essential matrix embeds more information than its counterpart discussed in the previous section. As a matter of fact, it is an expression of the epipolar constraint in terms of the normalized coordinates of the points of interest (i.e their coordinates in the $3D$ camera frame): if an image point is noted $\tilde{\mathrm{x}}$ then, given the calibration K, the associated point p of corresponding normalized coordinates is defined by :

$$\tilde{\mathrm{x}} = Kp \quad \text{or} \quad p = K^{-1}\tilde{\mathrm{x}} \tag{3.24}$$

As done in [14], by replacing (3.24) and (3.23) in (3.22), we obtain the following :

$$p'^{T} t_{\times} R p = 0 \tag{3.25}$$

By definition the essential matrix is :

$$E = t_{\times} R \tag{3.26}$$

Up to a scale, E characterizes completely the geometry between two cubes, just as completely as the fundamental matrix in the case of stereo images. This will be referred to often, in the rest of the text, as *E establishing the geometry of a pair of cubes*. That being said, (3.25) becomes :

$$p'^{T} E p = 0 \tag{3.27}$$

We need to know the calibration information to be able to estimate E since normalized coordinates are to be used (Refer to equations (3.24) and (3.27)). Nonetheless, in the case of the cubic panoramas, we have seen that the calibration matrix was as intuitive as it is simple and intrinsically linked to the structure of a cube; previously established equation (3.6) specifies its value.

Moreover, normalized coordinates in the case of cubes are equivalent to cube frame coordinates. Estimating E then becomes a matter of solving the classical problem of the epipolar constraint adapted to the case of cube frame coordinates with for example the 8 point algorithm used previously for the fundamental matrices and discussed in [12, 13, 31, 14, 8].

As a summary, the essential matrix presents the advantage of using cube coordinates that coincide with normalized coordinates. What is usually a difficult entity to recover - i.e normalized coordinates - is well known here in the case of the cube due to its structure. As a consequence, a cube is treated as a whole and not as a multi-camera system as for

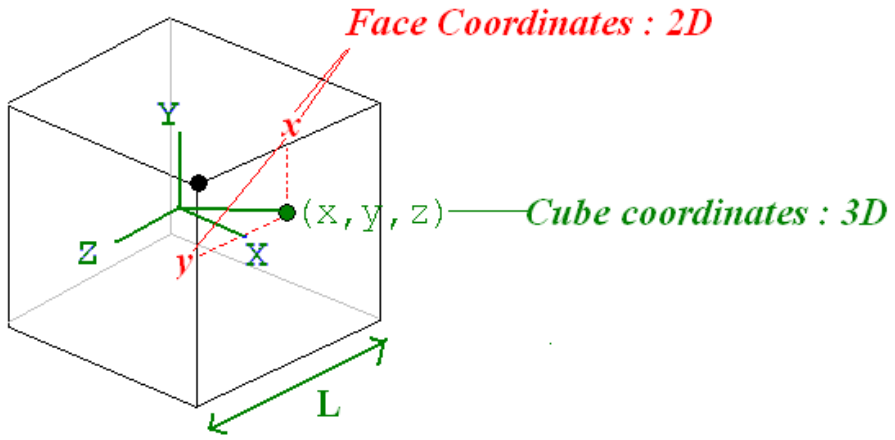


Figure 3.5: Conversion from $2D$ image coordinate to $3D$ cube coordinates.

the case of the fundamental matrices. This mainly results in simpler algorithms and more "global" approaches.

3.6.2 Matches : From 2D to 3D

Similarly to what is done for fundamental matrices, the support of the algorithm consists in matches between a pair of cubes. This is needed before any computation of any kind can take place. Thus, the matches needed for the estimation of E are selected for the sake of simplicity on the cross pattern images of the cubes either manually or using for example the *PVT* tool [29].

As noted earlier, the $2D$ coordinates of a point on a face of the cube allow an easy recovery of the corresponding $3D$ coordinates of that point in respect to the reference frame displayed on figure 3.2(a). This is used during the matches selection and consists in what follows.

Applying the right offset values (coordinates of top left corner of face in the cross pattern image) in the x and y directions converts these coordinates into their equivalent with respect to the faces of interest. Once these "face" coordinates are known the "conversion" to 3D coordinates in the cube frame seen in figure 3.2(a) is a matter of applying a simple transformation noted T_i. If x is a 2D point on face i, $p = T_i\tilde{\mathrm{x}}$. The expressions of the transformations T_i can be found in appendix B. The conversion process is illustrated in Fig.3.5

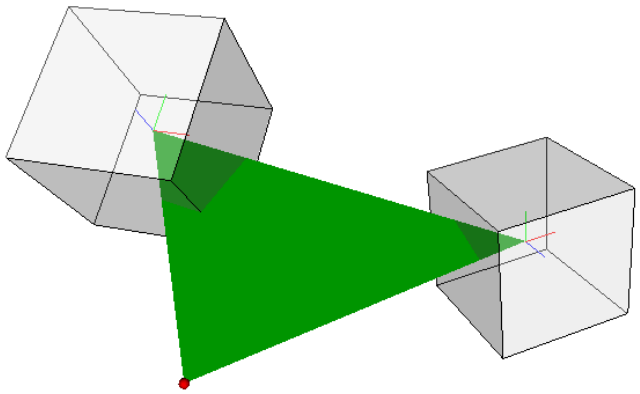


Figure 3.6: Epipolar plane through the camera centers and a $3D$ point.

3.6.3 The epipolar plane

Fig.3.6 illustrates the concept of epipolar plane. By definition, it is simply the plane through both cubes centers and the $3D$ point X[14]. Approaching the problem from the point of view of E is a switch of paradigm that proves to be quite efficient. The dimension of our "working" space is increased by one and this has the advantage of making us work with a plane instead of lines.

Equation (3.27) is in some way the expression of the epipolar plane. As a matter of fact [8] gives the following geometric interpretation of the constraint. Vectors p and p' are coplanar if they correspond to matches over two cubes : (3.27) is the dot product of p' (resp. p)and the normal to the plane in question given by Ep (resp. $p'^T E$). A plane through the origin of a coordinate frame is totally defined by its normal. This why Ep can be referred to as the epipolar plane with respect to the reference frame of cube C' (resp. $p'^T E$ is the epipolar plane with respect to reference frame of cube C).

It the intersection of this plane with both cubes that provide the set of epipolar lines that where computed individually earlier. The epipolar plane therefore gives a closed form of the geometry associated with two cubes once it is totally defined i.e once E is computed from pre-established matches.

3.6.4 Estimate E

The proper form of matches being known at this point ,we apply the 8 point algorithm discussed by [8, 14] among others to solve for E from the known p_i on C and C' : we solve the classical equation $p_i'^T E p_i = 0$. As a matter of fact we use a variant of the *DLT* algorithm used in the first chapter (Chapter 2) to evaluate either a fundamental matrix F or a homography H from pre-selected matches.

Scaling

The first step in the *DLT* algorithm is the normalization of the matches used in the estimation. Recall that this normalization aims at transforming the set of points into a cloud of centroid at 0 and of average distance to the centroid equal to $\sqrt{2}$. This procedure is much simpler in the case of E for cubes by the fact that the points all belong to a cube since all matches are already centered in 0. We therefore just apply a scaling of the 3 coordinates of all p_i by their maximum possible absolute value which is $\frac{L}{2}$ with L the side of the cube.

Forming a new system of equations

Next, we need to re-write (3.27) into an equation of the form :

$$Ah = 0 \tag{3.28}$$

where $h = (e_0, e_1, \cdots, e_8)^T$ stands for the elements of E in a single column vector. Each pair i of matches (p_i, p_i') produces one equation of the form :

$$A_i h = 0 \tag{3.29}$$

With :

$$A_i = \begin{bmatrix} p_{i_x}p'_{i_x} & p_{i_y}p'_{i_x} & p_{i_z}p'_{i_x} & p_{i_x}p'_{i_y} & p_{i_y}p'_{i_y} & p_{i_z}p'_{i_y} & p_{i_x}p'_{i_z} & p_{i_y}p'_{i_z} & p_{i_z}p'_{i_z} \end{bmatrix} \tag{3.30}$$

The resulting matrix A is just a concatenation of all row matrices A_i produced by all available pairs.

First estimate of E

The third step of the algorithm is the resolution of the new equation (3.28). [8] states that the solution to such a system with the right singular vector corresponding the smallest singular value of A just as it was the case for the fundamental matrix and the homography computation. This result in h that provides us with an estimate of E noted $\tilde{\mathrm{E}}$.

Singular values equalization

The the singular values of the best estimate $\tilde{\mathrm{E}}$ of E are equal and typically if a and b are the two singular values of $\tilde{\mathrm{E}}$, we force them to $s = \frac{a+b}{2}$, the third singular value being null [8]. $\tilde{\mathrm{E}}$ is then recomputed by multiplying the new diagonal matrix of singular values (now equal) by the left and right singular decomposition matrices.

Observations

We obtained decent accuracy when using this slightly modified *DLT*. Let us note however that a strong match localization affects the accuracy of E as the latter is precise for those matches but does not respond well to extrapolated points i.e points not in the original cloud of matches. Matches well spread over the surface of the cubes are therefore strongly suggested.

3.6.5 Results

The pair of cubes that has been rectified is displayed in Fig.3.11. Note that they respectively the same as the cubes in Fig.3.3 and Fig.3.7. All the previous tools being in place, we should be able to estimate the essential matrix between two cubes provided a "good" set of matches. We have seen the matches selection and the transformation to cube coordinates. Then we presented the concept of epipolar plane that necessitated the estimation of E. The latter was also described and obviously required the fore mentioned matches. Once the essential matrix is known, it is possible to obtain for any point of the surface of the cube, its associated epipolar plane i.e also the set of associated epipolar lines.

As a matter of fact, for a point $\tilde{\mathrm{x}}$ on a face of a cube C in correspondence with another cube C', its cube coordinates are $p = T_i\tilde{\mathrm{x}}$ with the T_i given in appendix B. To this point p in C corresponds the plane Ep in cube C'. The intersections of Ep with C'

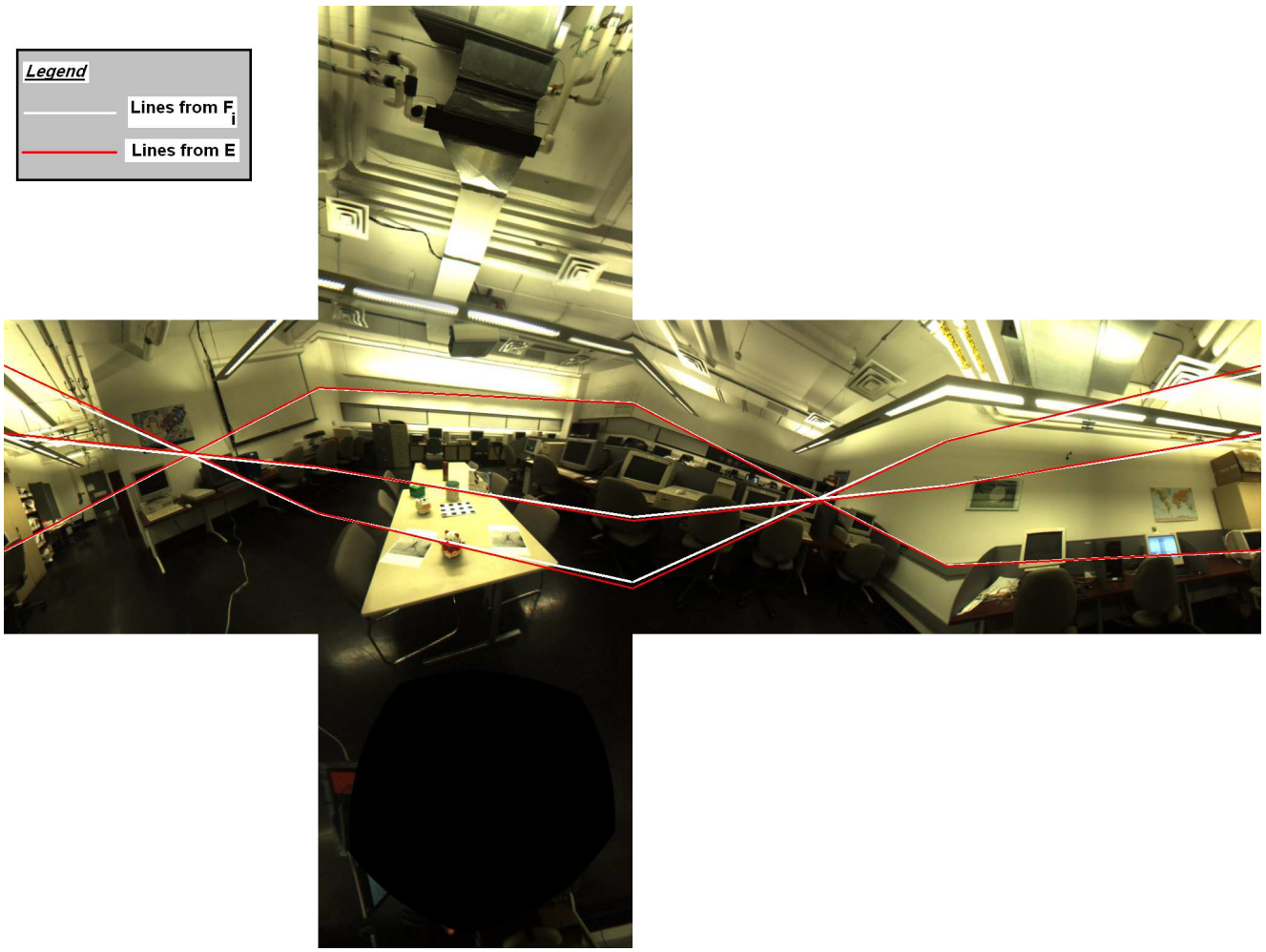


Figure 3.7: Epipolar lines over a cube from the E and the matrices F.

are provided by a set of equations in appendix B : the result is a set of 4 lines that should be identical to the previously computed epipolar lines from fundamental matrices. To give an example of the consistency our approaches, a sample of lines obtained with both methods are displayed on figure 3.7 : white lines are generated using F and red lines are generated with E. The points selected in the other cube of the pair are indicated by small red squares on Fig. 3.3.

As predicted, both solutions fusion to equally describe the geometry between two cubes. Only, the essential matrix is simpler and more intuitive.

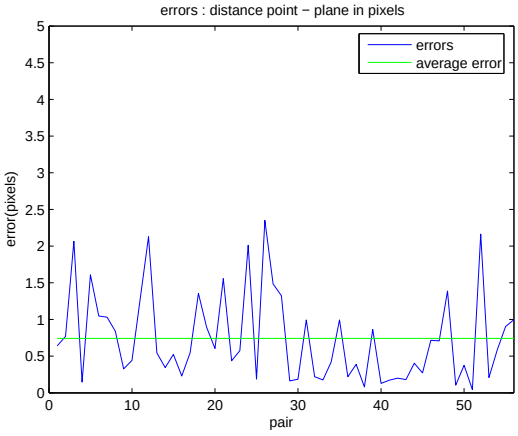


Figure 3.8: Distance point to plane for all matches

Numerical observations

This paragraph is a short numerical example of the estimation of the essential matrix for a sample pair of cubes. The pair used here is the same one that appears in Fig.3.3 and Fig.3.7. Its essential matrix is computed using 56 matches and has the value :

$$\begin{pmatrix} -0.0189908 & -0.0853818 & -0.0272546 \\ -0.143047 & -0.0851576 & -0.307995 \\ -0.0159993 & 0.332007 & -0.0973107 \end{pmatrix} \tag{3.31}$$

It is a priori difficult to evaluate the accuracy of the DLT algorithm used for the estimation. A good measure to achieve this is the distance from a point to its associated epipolar plane. Ideally the distance should be null since the point should lie on the plane so the smaller the better when it comes to this entity. The way to compute it is simple. Given a plane P through 0 of normal $n = (a, b, c)^T$, and a point p, the distance point-plane is given for example by [28]:

$$d(p, P) = \frac{|n.p|}{||n||} = \frac{|ax + by + cz|}{\sqrt{a^2 + b^2 + c^2}} \tag{3.32}$$

Thus we compute this distance for all matches and the resulting graph is shown in Fig.3.8. The average error is $0.7452 pixels$ which is a very satisfactory result despite the fact that there is some pre-processing during the cube generation and that some ideal assumptions were made about the calibration matrix among other things.

3.7 Application: cube rectification

Characterizing the geometry of a pair of cubes was done previously by using an extension to stereo images procedures through multiple fundamental matrices. Moreover, a closed form recovery was also introduced and used the essential matrix. Up to a scale, the latter completely characterizes the relationship between two cubes. The procedure to obtain it consisted in getting matches over the cubes, convert them into cube coordinates, and finally estimate the essential matrix E using a variant of the 8-point algorithm.

The present section will help cement the previously established theory with more than the epipolar lines, the only illustration we have seen so far. As a matter of fact, just as rectification was used in chapter one to demonstrate epipolar geometry for stereo images, we present the equivalent in the cubic panorama case that is not surprisingly named *cube rectification*.

3.7.1 Objective

Cube rectification is to cubic panoramas what epipolar rectification is to planar images. As a matter of fact, for a pair of cubes, the associated essential matrix E carries the information of the epipoles directions and of the rotation between the frames. Using that information, just as it is done with the fundamental matrix in the case of planar images, we aim at finding the rotations that - by analogy with the homographies for images - will be applied to each of the panoramas to obtain a rectified configuration. In such a configuration, both panoramas have all their corresponding faces parallel and coplanar as can be seen in Fig.3.9. This extension to the equivalent planar image concept is particularly useful as a starting point of in-between viewpoint interpolation for cubes.

3.7.2 Principle

For two images, the epipoles were literally sent to infinity in the x direction to solve the rectification problem as it is done for example in [13, 21] : two rectifying homographies H_1 and H_2 were applied to each image to achieve rectification. To be able to rectify two cubes in any configuration, on top of computing the essential matrix E for the pair of cubes, we follow an analog principle by finding two rotations R_1 and R_2 that align the cubes in a preferred configuration. As a matter of fact, if we define $p_i = R_i m_i$ or $m_i = R_i^{-1} p_i$ for $i \in 1, 2$ and the points p_1 and p_2 of cubes C_1 and C_2, we have from (3.27) :

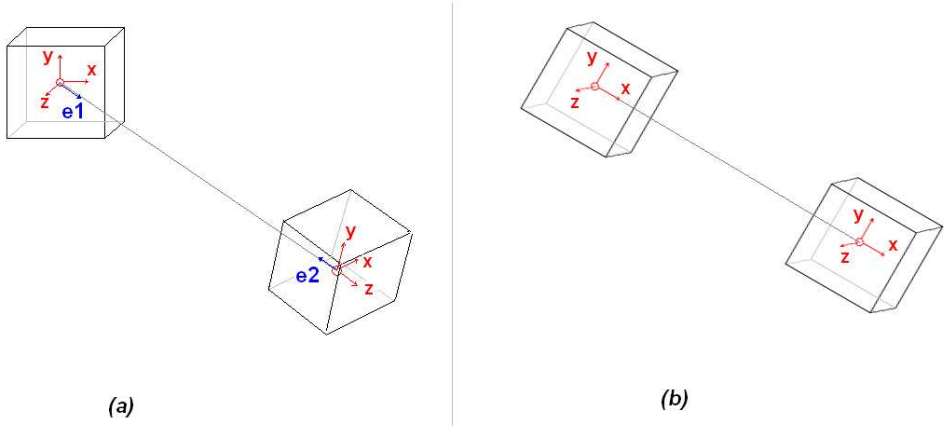


Figure 3.9: (a) Cubes in general configuration (b) Rectified cubes

$$p_2^T E p_1 = 0 \tag{3.33}$$
$$(R_2 m_2)^T E (R_1 m_1) = 0 \tag{3.34}$$
$$m_2^T R_2^T E R_1 m_1 = 0 \tag{3.35}$$
$$m_2^T E_r m_1 = 0 \tag{3.36}$$

With :

$$E_r = R_2^T E R_1 \tag{3.37}$$

After applying the rotations R_i to each cube C_i, the resulting configuration can be seen on figure 3.9. It basically shows that the rotations R_i are such that the x axis of both cubes coordinate systems merge into a common axis between both cubes going through both centers (baseline axis): the difference between the cubes C_1 and C_2 becomes only translational along the x axis, the original rotation being reduced to the identity matrix. In such a configuration [14, 21] shows that, using equation (3.26), the corresponding essential matrix is :

$$E_r = [(1,0,0)]_\times^T I_3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \tag{3.38}$$

This is a lot reminiscent of what is done for example in [13, 21] for the case of stereo images where homographies become rotations and the fundamental matrix is replaced by the essential matrix.

The interest of such a procedure resides in the possible use in other applications. For example such rectification could greatly ease cubic panorama interpolation, disparity estimation just like in the case of stereo rectification in the case of images. It could also help stabilize an almost linear sequence of cubes aligning them one by one along a common direction reducing navigation jitters.

Let us note that [11] also mentions a rectification process. The main difference is that the cubic panoramas are only rectified by a rotation along their vertical axis. This implies that the cubes must lie, in their original configuration, in the same plane. In our case, we cast away this constraint by solving the general configuration problem. Therefore there is no restriction on the capture process of the cubic panoramas of interest.

3.7.3 Extracting Epipoles

Needless to say, computing the essential matrix E(see previous section) is the preliminary step of this procedure. Once E is known, we then need to recover its epipoles. Similarly to their definition in the chapter on epipolar geometry for images, the epipoles in the cubic panorama context are unit vectors encoded in the essential matrix and noted e_1 and e_2. e_1 (resp. e_2) is basically the direction in which C_2 (resp. C_1) is located with respect to the cube C_1's (resp. C_2's) frame.[8, 7] notes as a property of the essential matrix that :

$$Ee_1 = 0 \text{ and } e_2^T E = 0 \tag{3.39}$$

Both vectors e_1 and e_2 are recovered from a *SVD* decomposition of E. They respectively correspond to the right and left singular vectors of least singular value that is ideally 0. Fig.3.9 displays the epipoles of the pair.

3.7.4 Computing rotation R_1

To recover the first rotation R_1, the x axis of the C_1 reference frame has to be aligned onto e_1. In other terms, the rotation R_1 needed here is such that :

$$R_1(1, 0, 0)^T = e_1 \tag{3.40}$$

This can be done in a geometrical manner. In general, if we are looking for a rotation R such that for two non null and non collinear vectors u and v, we have $Ru = v$ we can proceed as follows :

- First look for the axis of rotation w. It is given in this case by the cross product $w = u \times v$.
- Next, we recover the angle θ such that $R = R_w(\theta)$. This is done using the following formulas :

$$u.v = ||u||||v|| \cos(u, v) \tag{3.41}$$
$$u \times v = ||u||||v|| \sin(u, v) \tag{3.42}$$

They allow us to recover :

$$\tan\theta = \frac{u \times v}{u.v} \tag{3.43}$$

Once the axis w and the angle θ are recovered we use equation (3.5) to obtain R. Special attention is needed for collinear vectors, situation in which the rotation is a trivial one : either identity, or a reflection. For orthogonal vectors $u.v = 0$ and the division by 0 implies a $\frac{\pi}{2}$ angle. This can also be "caught" before hand. To recover R_1, we thus need to consider $u = (1, 0, 0)^T$ and $v = e_1$ and those special cases if necessary.

3.7.5 Computation rotation R_2

For R_2, an identical procedure as the one used for R_1 is used here. The only difference is the fact that the x axis of the reference of C_2 is aligned onto $-e_2$ instead of e_2 as it would have been the case for R_1. This insures that both resulting x axis point int the same direction, thus insuring the ideal essential matrix $E_r$for the new configuration defined in (3.38).

Both rotations R_1 and R_2 are now known. Applying the latter of the respective panoramas guarantees a configuration in which both cubes x axis are aligned. However, nothing implies that the cubes are in the proper configuration face wise : the faces of the cubes also need to be aligned similarly as what is done with image rectification. Therefore an additional rotation on one of the cubes, in our case C_2, has to be evaluated to compensate the rotation computed above and finalize the rectification process (see Fig.3.10).

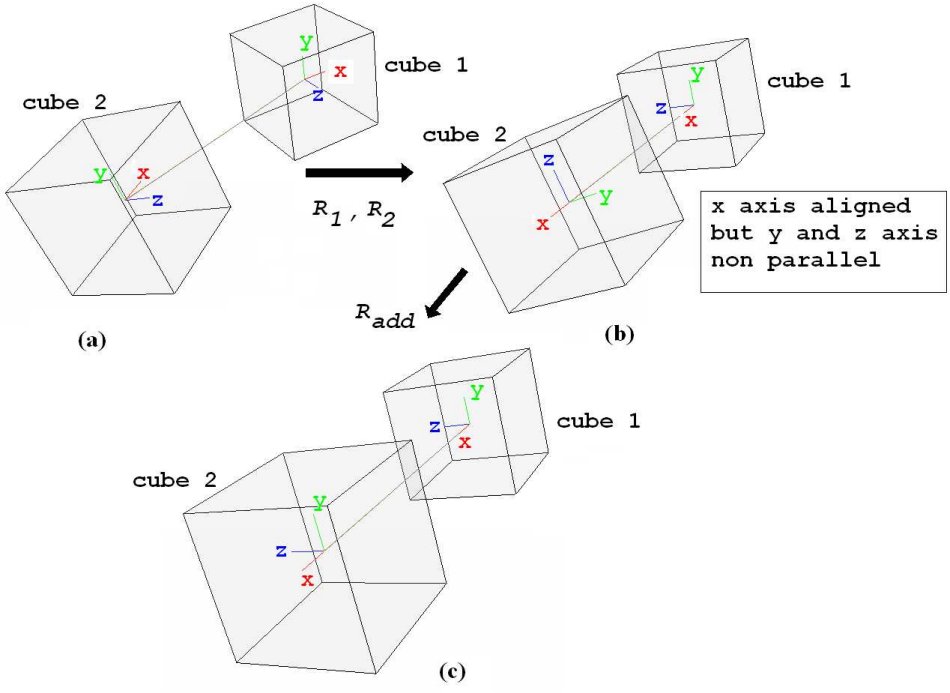


Figure 3.10: Rectification process with compensating rotation

3.7.6 Computing the tweaking rotation R_{add}

We chose to compensate the rotation R_2 applied on cube C_2 with an additional rotation noted R_{add}. To achieve this goal, two methods were designed both trying to solve the same problem of estimating R_{add} such that :

$$(R_{add}R_2)^T E R_1 = E_r \tag{3.44}$$

The rotation R_2 in section 3.7.5 is then pre-multiplied by R_{add} to obtain the final rotation on C_2. To avoid any possible confusion, we will note R_2 the final rotation applied to C_2, by r_2 the rotation computed in section 3.7.5. We therefore have :

$$R_2 = R_{add}r_2 \tag{3.45}$$

The first solution is a "brute force" one. The problem is solved as it is by solving (3.44) as a minimization problem with the elements of R_{add} as variables. The second solution is a direct and geometrical one in which we use our knowledge of the configuration of both cubes to compute the unknown rotation.

Using minimization

The solution presented here follows the template laid out by [21] for its distortion reduction algorithm (see image rectification in chapter 1 for more details). As a consequence the problem is re-written as follows :

$$(R_{add}r_2)^{-T} E_m R_1^{-1} - \alpha E = 0 \tag{3.46}$$

α is a new unknown on top of the elements of the rotation R_{add} and stands simply for a scale factor. As a matter of fact, the equalities in (3.44) or (3.46) are up to a scale factor.

We know that the additional rotation will be of axis x. We also know E_r, R_1, r_2 and E. The fact that we know the axis of rotation reduces the number of unknowns in the equation above to 2 : the rotation angle θ around the x axis and α. The equation (3.46) can then be considered of the form :

$$f(\theta, \alpha) = 0 \tag{3.47}$$

It is a minimization problem that we solved in $C++$ using the simplex algorithm [22]. But really, any state of the art multi-variable function minimization algorithm could be

used (for example the popular *fminsearch* in *Matlab*). Once θ is found, we use (3.5) to obtain $R_{add} = R_x(\theta)$.

Direct geometric solution

The rotation that we are looking for is a rotation of angle θ around the x axis. As a matter of fact, once the rotation R_1 and r_2 are applied, both cubic panoramas are in a configuration where they differ only by a translation on the x axis and by an unknown rotation of angle θ around the x axis (see Fig.3.10). The essential matrix E'_r that corresponds to such a configuration is given by :

$$E'_r = t_\times R_x(\theta) \tag{3.48}$$

but also by :

$$E'_r = r_2^T E R_1 \tag{3.49}$$

Using (3.2) and the general formula $R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}$ allows us to write (3.48) as :

$$E'_r = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -\sin\theta & -\cos\theta \\ 0 & \cos\theta & -\sin\theta \end{pmatrix} \tag{3.50}$$

The fact that all variables on the right hand side of (3.49) allows us to recover the values of $\sin\theta$ and $\cos\theta$ by considering (3.49) equal to (3.50). Using the inverse tangent, θ is computed and as a result, so is $R_{add} = R_x(\theta)$.

3.7.7 Results and Observations

Both methods were implemented on the same pair as the one studied in the previous section. Each cube of the pair is shown respectively in Fig.3.3 and in Fig.3.7, and both panoramas are displayed in Fig.3.11. The essential matrix is computed from 56 matches and has the value given in (3.31). The procedure explained in 3.7.2 is followed to compute the rotations R_1 and R_2.

Minimization approach

The *minimization* based method produces the rotations :

$$R_1 = \begin{pmatrix} -0.9121 & 0.0679 & -0.4042 \\ 0.0742 & 0.9972 & 0.0000 \\ 0.4031 & -0.0300 & -0.9147 \end{pmatrix} \text{ and } R_2 = \begin{pmatrix} -0.9660 & 0.0853 & 0.2442 \\ 0.1517 & 0.9515 & 0.2676 \\ -0.2095 & 0.2955 & -0.9321 \end{pmatrix} \tag{3.51}$$

The product expressed in equation (3.37) should ideally give us the matrix E_r. Computing this product with our estimated rotations is a good measure of how close they are to ideal solutions. For the minimization based method, we have :

$$R_2^T E R_1 = \begin{pmatrix} 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0098 & 0.3541 \\ 0.0000 & -0.3541 & 0.0098 \end{pmatrix} \tag{3.52}$$

The result is quite satisfying since the resulting matrix is really close to the ideal essential matrix up to a scale factor of -0.3541.

Direct geometric solution

For the direct geometric solution, we obtain the following rotations from the same essential matrix and matches as above :

$$R_1 = \begin{pmatrix} -0.9121 & -0.0742 & -0.4031 \\ 0.0742 & 0.9373 & -0.3404 \\ 0.4031 & -0.3404 & -0.8495 \end{pmatrix} \text{ and } R_2 = \begin{pmatrix} -0.9660 & 0.1690 & 0.1958 \\ 0.1517 & 0.9833 & -0.1006 \\ -0.2095 & -0.0675 & -0.9755 \end{pmatrix} \tag{3.53}$$

The rotations obtained here are very close to the ones obtained in the previous case. This shows the consistency in the approach. However the product that is used as a test measure gives us, in this case :

$$R_2^T E R_1 = \begin{pmatrix} 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.3542 \\ 0.0000 & -0.3542 & 0.0000 \end{pmatrix} \tag{3.54}$$

This is what is expected in terms of ideal essential matrix up to a scale factor of -0.3542. (3.52) and (3.54) show that both approaches a quasi identical. The almost

imperceptible advantage goes to the direct geometric solution from a numeric point of view. This decision is much clearer when it comes to simplicity and efficiency as selection criteria

Finally, Fig.3.13 shows a 3D rendition of the situation that we are facing. In the foreground are the cubes in their original configuration. This is reproduced up to a scale as far as the distance between both panoramas. We essentially used t and R extracted from E to be able to recreate the original situation (see method in appendix D). In the background are the rectified cubes. The rectification is easily perceived visually. The rotations used here are the ones obtained by the direct geometric method. Note that the same rendition in $3D$ with the other solution (minimization) has visually no difference with this one, in other words, the difference is really too small to be perceived. To end this results section, Fig.3.12 presents the rectified pair of panoramas corresponding the result obtained from the pair in Fig.3.11.

3.8 Conclusion

Cubic panoramas offer through their structure many advantages that were exploited in the present chapter. Prior knowledge of epipolar geometry for non panoramic cameras was molded to fit the particular case of cubes proving once more the engineering aspect of this whole study : use available tools to develop and reinforce new algorithms and possibly extend existing theory to new grounds.

It is explicitly what was done here by building a transition from a simple camera system to a multi-camera system. The equivalent of important epipolar entities and relationships were found for cubic panoramas. Thus we first introduced the fundamental matrix concept in a stereo cube configuration. Not one but 6 fundamental matrix are necessary to characterize the geometry of a pair of cubes even if we have seen that one matrix could “generate” the others.

This first attempt to conceptualize cubic panorama epipolar geometry was reinforced by the concept of essential matrix. Allowing the consideration of a cube a whole and not as 6 cameras, the essential matrix is a simple and efficient mean of characterization. Both methods produce epipolar lines over the panoramas allowing us to verify their validity and their consistency.

Finally, as an illustration of the established entities, cubic panorama rectification was undertaken. The concept is also based on epipolar rectification for images, although being in $3D$ space implies the use of rotations instead of homographies. Essentially

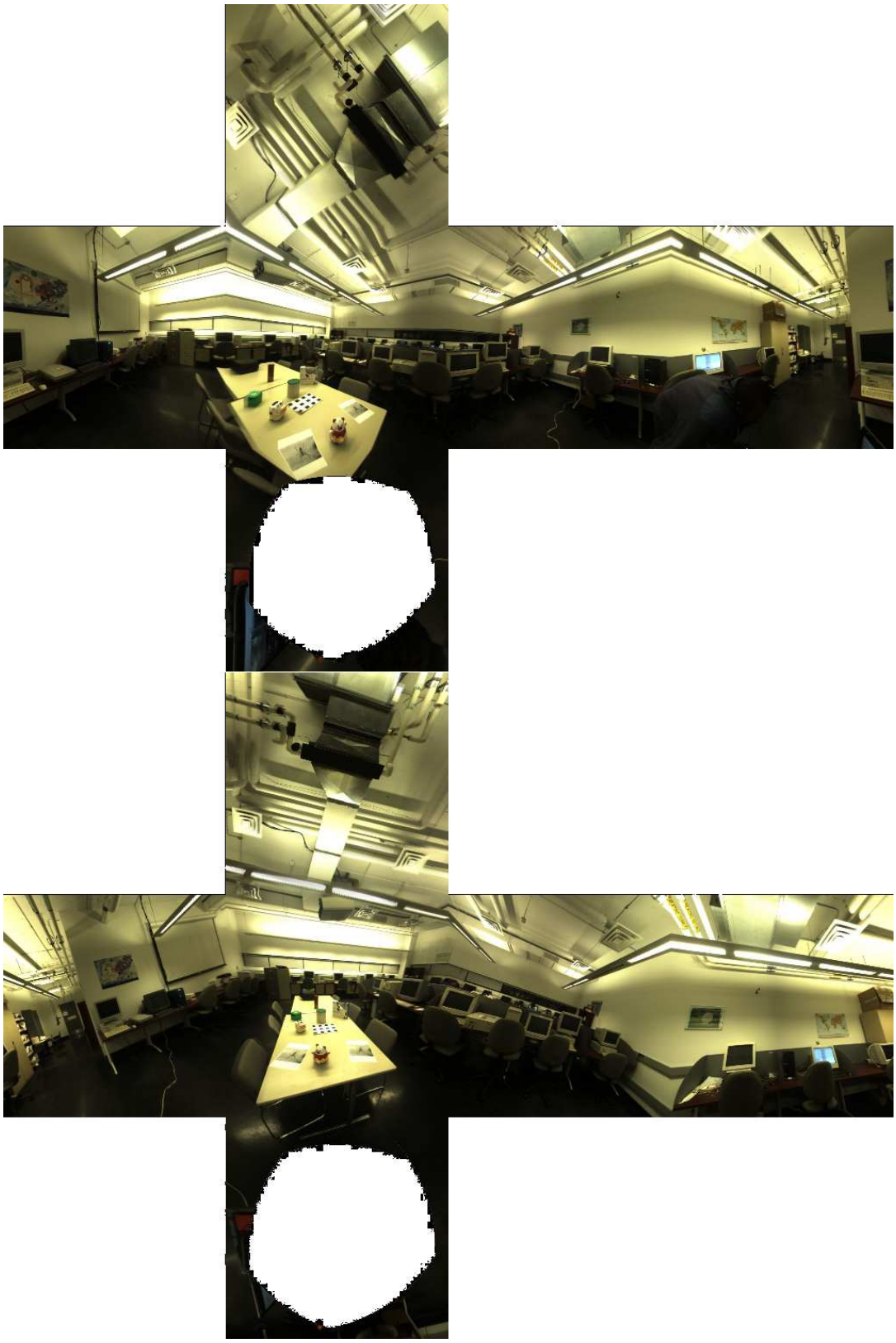


Figure 3.11: Pair of cubic panoramas before rectification.

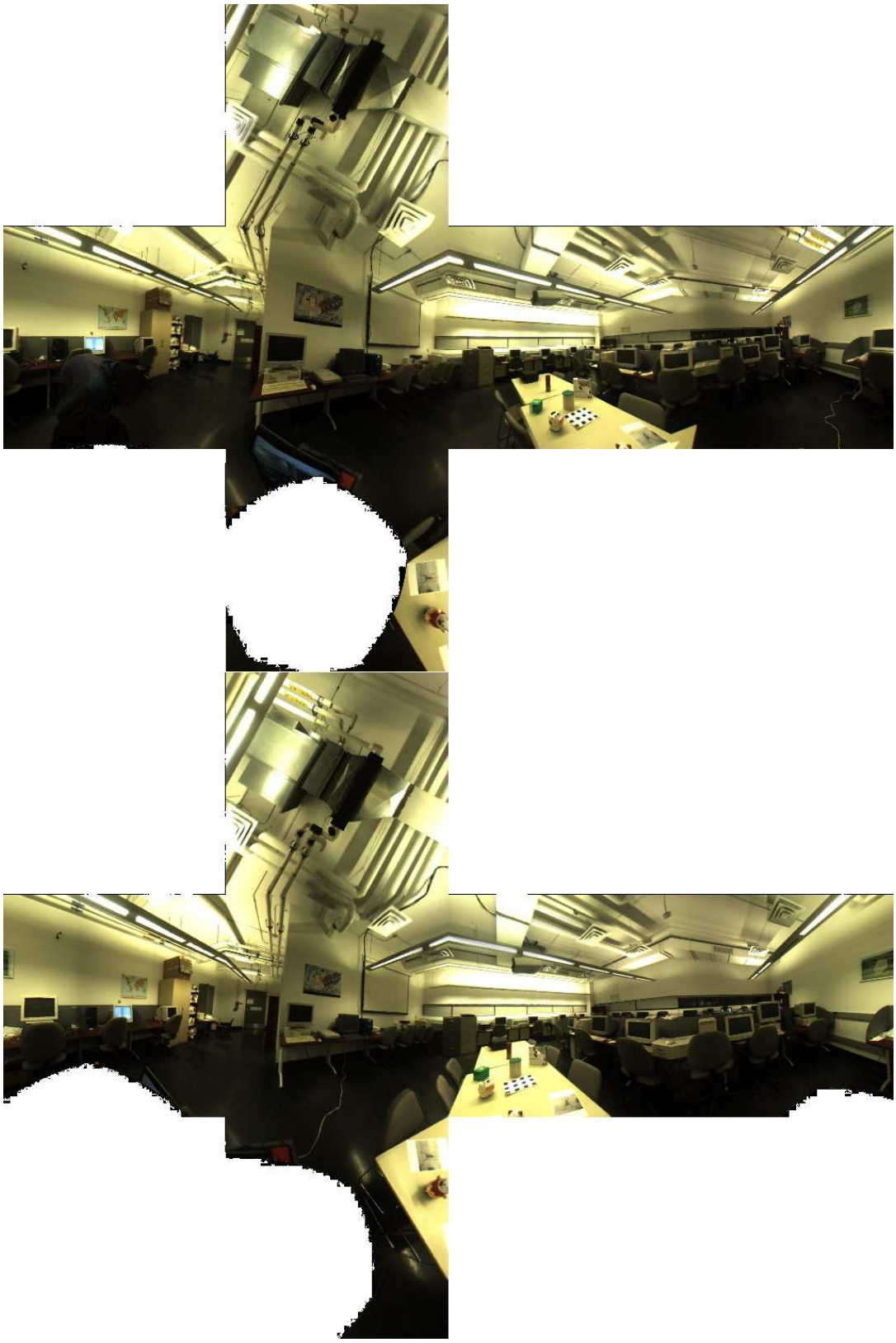


Figure 3.12: Pair of cubic panoramas after rectification.

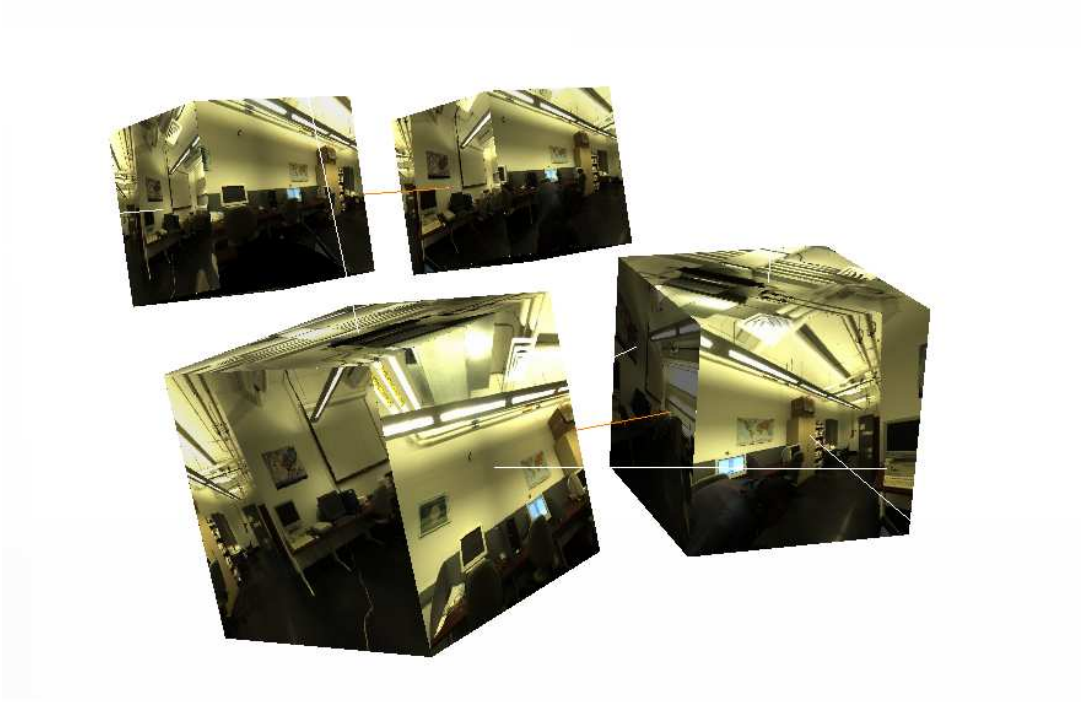


Figure 3.13: Cube Rectification : original configuration (front) , rectified pair (back)

two methods were presented to estimate the rectifying pair of rotations each producing consistent results. However, a direct geometric solution provided a simpler and more elegant approach to the problem.

Adapting existing algorithms to cubes proved to be a success when certain considerations were carefully followed. We surely cannot blindly apply standard stereo image theory to cubes but we can definitely use it as a starting point. The next chapter even goes further in the idea of illustrating the epipolar geometry for cubes with the cube alignment concept as well as the pose recovery from cubes. Once again, it will heavily rely on state of the art processes such as bundle adjustment.

Chapter 4

Pose Recovery Applied to Cubes

4.1 Introduction

Cube rectification was presented in the previous chapter as a possible application of the study of cubic panoramas' epipolar geometry. Another application, namely the camera pose recovery problem in the case of spherical panoramas, is discussed here. It consists essentially in using visual information extracted from a sequence of images to estimate the motion parameters of each of the cameras involved in the capture process. A solution to this problem can be exploited in many ways. One could for example localize objects of the scene from given features or insert a virtual object into the environment at a given position. One could also take advantage of the motion parameters, in a virtual navigation application, to adequately position the images with respect to the environment, to ensure consistent and fluid navigation and to ease the process of image interpolation.

The pose recovery problem has been quite extensively addressed in the literature especially for the case of planar images and essential matrix based methods. For example, Zhang provides in [32] a starting point for any structure from motion procedure based on the essential matrix. His method identifies some constraints used during reconstruction and puts the accent on the refinement of the essential parameters extracted from the essential matrix.

Carceroni et al. present a method that relies on some properties of the Special Orthogonal Group $SO(3)$ [6]. It is a feature-based procedure that uses a GPS to recover camera orientation from known positions of multiple cameras. Constraints on the essential matrices between the images are established resulting in overall rotations estimation. Their work is a special case of the problem discussed in this chapter in which only the

rotation components remain to be estimated. The algorithm presented here applies to spherical panoramas in general unknown positions.

Makadia et al. present in [20] a method to recover the rotation from two spherical images. This method also uses properties of the Special Orthogonal Group $SO(3)$ in addition to "the persistence of image content". It has the advantage of not being based on feature points. One of the images is rotated during the search until it matches the other in a harmonic coefficients space. The matching criterion is correlation-based. Nonetheless, the rotation estimation being the only point of interest of this method no translation information is extracted.

Ramalingam et al. 's work in [23] is very similar to the subject in this chapter. The similarities are found at many levels. Their pose recovery framework applies to spherical images and relies on matches and the essential matrix as well as a bundle adjustment for motion parameters estimation. The motion parameters are refined all at once in the bundle and cross-scenarios with different types of cameras are investigated with a necessary calibration step. The latter are the main differences with what is done here.

Finally, Fiala et al. in [11] present a method based on the essential matrix to recover the relative orientations of panoramas belonging to a sequence. The assumption is made that the rotations are only along the vertical axis which is valid in the case of a capture process that maintains the camera in the same plane. Our work differs in that it does not rely on any assumption on the rotations and that rotation and translation are treated separately.

Thus, this chapter describes a two-stage algorithm to retrieve the camera motion from a set of spherical panoramas represented in a cubic format. The first stage is the estimation of the relative rotations with respect to the world frame. A bundle adjustment approach in conjunction with pair-wise essential matrices is used to recover each rotation in a global solution. Once each rotation is known, one can then obtain a configuration in which all camera frames are aligned with respect to each other. This configuration achieves what will be designated as "cube alignment", an extension of the concept of cube rectification - presented in presented in Chapter3 - to a given number N of cubic panoramas.

The second stage uses the resulting configuration from the first step. Since negligible rotation now exists between each camera frame, the problem becomes a pure translation estimation. The final solution, up to a scale, is obtained using an error minimization approach.

The layout of the chapter is as follows. First, some useful notations and reminders are

presented, followed by the concept of cube alignment, the first stage of the pose recovery algorithm. Next, is discussed the translation estimation from an aligned configuration to complete the recovery process as the second stage. This chapter ends with some results obtained on two sets of cubic panoramas, respectively captured indoor and outdoor.

4.2 Notations

The set of all cubes that are of interest in the bundle adjustment algorithm is noted $\mathcal{C}$. For a given cube c, the set of cubes that share some matches with the latter is noted $\mathcal{M}(c)$. Moreover, $\mathcal{F}(c, c)$ stands for the set of common features for a pair of cubes $(c, \bar{c})$. Any entity related to a given set of cubes $c_1, \cdots, c_N$ and other variables $a_1, \cdots, a_n$ will have $(c_1, \cdots, c_N, a_1, \cdots, a_n)$ appended to its symbol for example a rotation R associated to a cube $\bar{c}$ will be noted $R(\bar{c})$, the translation t between cubes c and $\bar{c}$ is noted $t(c, \bar{c})$; all these notions stand as mentioned here unless otherwise specified.

4.2.1 Tensor notations and useful properties

We will also make use of the tensor notation to manipulate complex equations in a much simpler way. It is important to note that for a given indexable entity such as a vector or a 3 by 3 matrix, its tensor notation is of the type :

$$A_{i_1 \cdots i_N}$$

where N is the order of the tensor. For a more complete introduction on tensor calculus the reader should refer to [15]. Here we are only interested in tensor notations of 3 element vectors and 3 by 3 matrices and we present a short summary of some properties retrieved from the previously mentioned source. For example :

- $p_i,\ i = 1, 2, 3$ represents the components of a 3D vector p
- $R_{ij},\ i = 1, 2, 3$ and $j = 1, 2, 3$ represents the components of a 3 by 3 matrix R

In addition we use some fundamental results of tensor calculus concerning the triple scalar product of vectors as well as the concept of summation or Einstein's summation very nicely described in [15]. The latter case states that if an index, in a tensor notation context, is repeated not more than twice on a side of an equation, this implies a summer

on that index-called *summation index*-, over the range of that index. A possible example of this is the expression of the scalar product of two vectors p and q. We know :

$$p.q = \sum_{i=1}^{N} p_i q_i,$$

if N stands for the dimension of the vector space of interest. The summation property stated above gives us :

$$p.q = p_i q_i \tag{4.1}$$

Where the sum symbol is omitted since implicitly considered. As it can be seen, this property greatly lightens up equations involving multiple summations as long as the indexes are used soundly.

To end this paragraph, we mention the last equation related to the tensor notation that will appear to be quite useful in the next sections. As a matter of fact, the triple scalar product of three vectors $u,\ v$ and w classically given by :

$$[u, v, w] = (u \times v).w = \begin{vmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w_3 \end{vmatrix}$$

can be re-written as follows thanks to the summation property :

$$[u, v, w] = (u \times v).w = \varepsilon_{ijk} u_i v_j w_k \tag{4.2}$$

Assuming $i,\ j$ and k can take any value in $\{1, 2, 3\}$, the symbol ε_{ijk} is known as the permutation tensor and is defined in [15] by :

$$\varepsilon_{ijk} = \begin{cases} 0 & \text{, if } i = j \text{ or } j = k \text{ or } k = i \\ 1 & \text{, if } (i, j, k) \text{ is an even permutation of } (1, 2, 3) \\ -1 & \text{, if } (i, j, k) \text{ is an odd permutation of } (1, 2, 3) \end{cases} \tag{4.3}$$

As a reminder, $\sigma = (i, j, k)$ with $i, j, k \in \{1, 2, 3\}$ is an even (resp. odd) permutation of $(1, 2, 3)$ if the number of permutations applied to (i, j, k) to turn into $(1, 2, 3)$ is even (resp. odd).

4.2.2 Exponential representation of rotations

In the preceding chapter, we mentioned different definitions related to rotations.This is just a reminder of previously established and enounced properties. For a given rotation R of non null angle θ and of axis guided by the unit vector v, its Rodrigues vector ω is given by :

$$\theta = |\omega| \tag{4.4}$$
$$v = \frac{\omega}{\theta} \tag{4.5}$$

the associated matrix to R has the following exponential representation:

$$R = I_3 + sin\theta[v]_x + (1 - cos\theta)([v]_x)^2 \tag{4.6}$$

Where $[v]_\times$ stands from the antisymmetric matrix derived from the unit vector v :

$$[r]_\times = \begin{pmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ v_2 & v_1 & 0 \end{pmatrix}$$

4.3 General pose estimation algorithm

To solve the pose estimation problem applied to spherical images, we have designed an algorithm that is divided into a preliminary stage and the previously mentioned two-stage pose recovery itself.

- The preliminary stage consists in computing the essential matrix of all the pairs of panoramas for which a sufficient number of matches is available.
- The first stage of the pose recovery is to find the rotation to be applied to each cube so that, in the resulting configuration, all panoramas of the set only differ by a translation, all corresponding faces being parallel but not necessarily coplanar as it is the case in the rectification method. This results in what we called *cube alignment*.
- The second and final stage of the pose recovery is to start from the aligned set of panoramas to estimate their relative positions with respect to one another up to a scale. This translation estimation step relies on results of the first stage and allow the recovery of the camera positions.

4.4 Cube alignment

The problem of cube alignment is quite succinctly summed up by Fig. 4.1. In comparison, the rectification process is showed in Fig. 4.2. The difference could seem obvious visually but requires a little more elaboration theoretically. On one hand, cube alignment consists essentially in removing the rotational component that separates different cube frames resulting in cubes all facing the same direction without necessarily being aligned on their pairwise center-joining axis. On the other hand, cube rectification requires that, for a given pair of cubes, the resulting configuration is one in which the cubes are perfectly aligned (for each pair of corresponding faces considered) along the axis joining their centers which is also the epipolar axis.

We observe that the rectification process is a particular case of the cube alignment. As a matter of fact, if cube alignment is applied to a pair of cubes that differ by a translation along one of the base axis x, y or z composed with a rotation, the result is a rectified pair of cubes as shown in figure Fig. 4.3 : the cube in red is obtained from the cube on the left after a random rotation and a translation along the x axis.As a matter of fact, cube rectification is more restrictive than cube alignment. For cube alignment, all that is needed in parallel corresponding faces, which can be achieved by an infinite number of rotations. For cube rectification, the resulting panoramas must not only have parallel corresponding faces but also "look" in the same direction sustained by the axis through their centers. The latter makes it difficult to rectify more than 2 panoramas since these N cubes would have to share a common axis.

4.4.1 Why a bundle adjustment approach ?

We have mentioned previously that the configuration of the problem studied here is one where an arbitrary number N of cubic panoramas are considered all at once (versus cube rectification that involves a pair of panoramas). Fig. 4.1 shows the cubes in random configuration with respect to one another. As said earlier, a very common problem in vision is often to try to extract the structure of a scene from pertinent information retrieved from each sensor. Usually this information consists in points of the scene observable from multiple viewpoints. Each cube frame differs from the other by a translation and a rotation; the latter rotation is what is estimated in this section for each of the involved viewpoints. And for this purpose, we have chosen a bundle adjustment approach.

From [26], "*bundle adjustment is the problem of refining a visual reconstruction to produce jointly optimal 3D structure and viewing parameter(camera pose and/or calibra-*

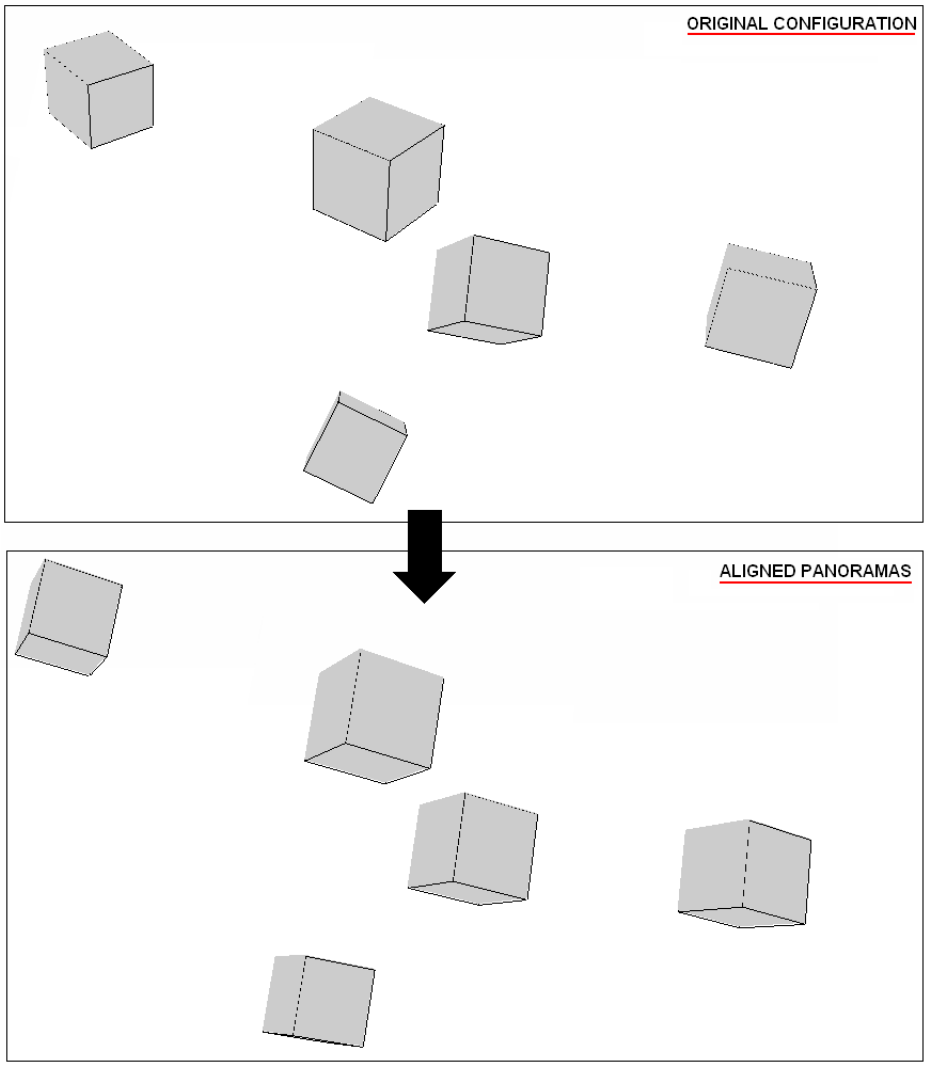


Figure 4.1: Cube alignment.

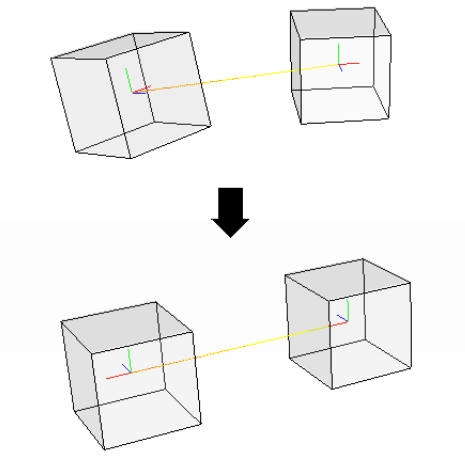


Figure 4.2: Cube rectification.

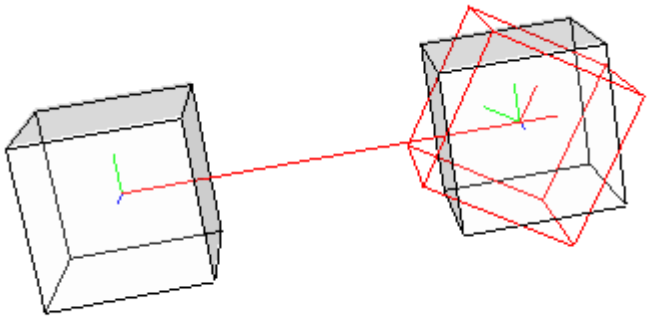


Figure 4.3: Cube rectification as a special case of alignment.

tion) estimates". In our case, the results that are sought after are the camera rotation (The calibration of the cube being implicitly known) for all cubes. The choice of such an approach is most of all guided by the fact that it is a "global" solution where each variable (rotation in this case) has its influence on the final result. This balanced concept is quite popular and has proven its efficiency in many applications [5, 26, 23] where camera poses were computed in order to provide a $3D$ representation of a scene. The extensive documentation and variety of applications as well as the very interesting work on rotations estimation through the optimization over the Special Orthogonal Group $SO(3)$ by [17] are other factors that influenced our choice. Also a naive solution would have been to extract and cumulate the motion from the pairwise essential matrices but the error would increase rapidly.

The core of the concept relies on the error or cost function that is to be minimized so that the parameters are estimated properly. In the usual case of camera pose recovery for example, the cost function is the sum of all re-projection errors for all observed points of the scene. In the case of cubic panoramas alignment, the cost function expresses the re-projection error by evaluating the epipolar constraint for all available pairs of panoramas.

4.4.2 From epipolar constraint to cube constraint

Let f be a feature in a scene (i.e also a 3D point) visible in cube c and $\bar{c}$ with respective cube coordinates $p(c, f)$ and $p(\bar{c}, f)$ according to the notation convention described at the beginning of this chapter. If $R(c)$ and $R(\bar{c})$ are the respective "aligning" rotations of each of these cubes with respect with the world coordinate frame system, and $t(c, \bar{c})$ is the direction unit vector between the panoramas (in the coordinate frame of c), then epipolar constraint suggests that the vectors $R(c)p(c, f)$, $R(\bar{c})p(\bar{c}, f)$ and $t(c, \bar{c})$ be coplanar (see Fig.4.4). This is expressed by the following triple scalar product and *residual*:

$$r(c, \bar{c}, f) = (R(c)p(c, f) \times R(\bar{c})p(\bar{c}, f)) \bullet R(c)t(c, \bar{c}) \tag{4.7}$$

And ideally, the coplanarity implies that :

$$r(c, \bar{c}, f) = 0 \tag{4.8}$$

This gives rise to quite a natural choice for the objective function that is to be minimized to zero. Since the coplanarity constraint has to be verified for all matches between all possible pairs of cubes, the total error is given by :

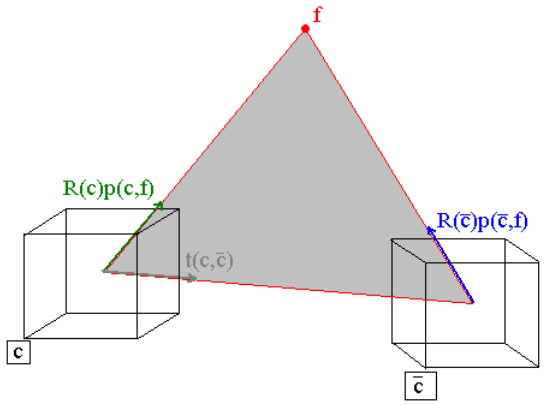


Figure 4.4: Epipolar/Cube constraint

$$e(\Theta) = \sum_{c \in \mathcal{C}} \sum_{\bar{c} \in \mathcal{M}(c,\bar{c})} \sum_{c \in \mathcal{F}(c,\bar{c})} r(c, \bar{c}, f)^2 \tag{4.9}$$

This error function is similar to the one used by [5], but is of course adapted to the case of cubes. It is important to note that the minimization of this cost function results in estimates of the different rotations $R_1, \cdots, R_N$ represented by their Rodrigues vector (see section 4.2.2) all stated under the single parameter $\Theta = (\omega_{1_x}, \omega_{1_y}, \omega_{1_z}, \cdots, \omega_{N_x}, \omega_{N_y}, \omega_{N_z})$. The solution Θ_{min} is such that :

$$e(\Theta_{min}) = 0 \tag{4.10}$$

4.4.3 Solving for all rotations

This section describes the core of the algorithm that is somewhat summarized by (4.9) but explained here for effective implementation. The steps mentioned next follow the framework described in [5] to solve the problem of bundle adjustment for N cubes. Let us, beforehand, note $\mathcal{B}$ the set of cubes for which a rotation has been found or is being optimized : this is what is called the "bundle adjuster" in [5].

For all available pairs of cubes $(c, \bar{c})$ that share matches, it is necessary to first compute the associated essential matrix $E(c, \bar{c})$ then extract the translation unit vector $t(c, \bar{c})$

and the rotation $\rho(c, \bar{c})$ between cube c and $\bar{c}$ using the method mentioned in chapter 3 and appendix D.

While $\mathcal{B}$ does not contain all cubes of $\mathcal{C}$ {

- *Add a cube c in $\mathcal{C}$ that is not in $\mathcal{B}$ and that has the highest total number of matches with the cubes in $\mathcal{B}$ (or $\mathcal{C}$ if it is the first iteration)*
- *Identify the cube $\bar{c}$ of $\mathcal{B}$ that best matches c and initialize the rotation $R(c)$ associated with the current cube as follows :*

$$R(c) = \begin{cases} R(\bar{c})\rho(c, \bar{c}) & , \text{ if } \mathcal{B} \text{ contains at least a cube} \\ I_3 & , \text{ otherwise} \end{cases}$$

  Where $\rho(c, \bar{c})$ is the rotation extracted from the essential matrix linking c and $\bar{c}$ and I_3 the 3 by 3 identity matrix. The initialization above can be thought of as bringing back the cube c to the world frame configuration first then rotating it into the best matching cube position. This causes both involved cubes to be aligned but does not guarantee alignment with all the other cubes of $\mathcal{B}$.

- *Minimize the error function given in (4.9) except the fact that the first sum is over the adjuster $\mathcal{B}$ instead of the whole set of cubes $\mathcal{C}$:*

$$e(\Theta) = \sum_{c \in \mathcal{B}} \sum_{\bar{c} \in \mathcal{M}(c,\bar{c})} \sum_{c \in \mathcal{F}(c,\bar{c})} r(c, \bar{c}, f)^2 = 0$$

  *We used non linear solution available in Matlab under **lsqnonlin** with zero as a goal and the jacobian of the residual defined in the next section.*

- *Process the next cube*

}

4.4.4 Jacobian evaluation for minimization

To be able to specify the variation of the objective function that is minimized in the previous section, it is necessary to evaluate the variation of the residuals that are squared and summed up in the error function. We recall that Θ stood for all possible parameters involved in the process describing all rotations that are being optimized. Considering

that, the goal here is to compute : $\frac{\partial r}{\partial \Theta}$. Equation (4.8) gives us the residual involving 2 cubes c and $\bar{c}$ and a given feature f:

$$r(c, \bar{c}, f) = (R(c)p(c, f) \times R(\bar{c})p(\bar{c}, f)) \bullet R(c)t(c, \bar{c})$$

We exceptionally change slightly the notation for the sake of simplicity. Thus, $R(c)$ will be noted R, $R(\bar{c})$ will be noted $\bar{R}$. Idem for $p(c, f)$ and $p(\bar{c, f})$. Finally $t(c, \bar{c})$ becomes simply t. Therefore the residual attached to a particular feature f becomes:

$$r = (Rp \times \bar{R}\bar{p}) \bullet Rt \tag{4.11}$$

Using (4.2), the triple scalar product (4.11) can be rewritten using the tensor notation introduced in section 4.2.1. Let us first consider the vectors :

$$u = Rp \tag{4.12}$$
$$v = \bar{R}\bar{p} \tag{4.13}$$
$$w = Rt \tag{4.14}$$

They correspond respectively to the following tensors (as a reminder, the summation property states that the repeating indices on a same side of an equation implicitly stand for a sum over the range of 1 to 3) :

$$u_i = R_{im}p_m \tag{4.15}$$
$$v_j = \bar{R}_{jn}\bar{p}_n \tag{4.16}$$
$$w_k = R_{kr}t_r \tag{4.17}$$

The fore-mentioned residual or triple scalar product can be expressed in terms of these tensors as follows :

$$r = (u \times v) \bullet w \tag{4.18}$$
$$= \varepsilon_{ijk}u_i v_j w_k \tag{4.19}$$
$$= \varepsilon_{ijk}R_{im}p_m\bar{R}_{jn}\bar{p}_n R_{kr}t_r \tag{4.20}$$
$$= \varepsilon_{ijk}R_{im}\bar{R}_{jn}R_{kr}p_m\bar{p}_n t_r \tag{4.21}$$

Let us consider the tensor D that depends only on the components of the vectors p, $\bar{p}$ and t :

$$D_{mnr} = p_m \bar{p}_n t_r \tag{4.22}$$

(4.22) in (4.21) gives rise to the following tensorial expression of the residual recalling that ε_{ijk} is given by (4.3) :

$$r = \varepsilon_{ijk} R_{im} \bar{R}_{jn} R_{kr} D_{mnr} \tag{4.23}$$

[17] gives us an expression of the derivative of a given rotation R with respect to the associated Rodrigues vector ω:

$$\frac{\partial R_{\mu\nu}}{\partial \delta\omega_\alpha} = -\varepsilon_{\alpha\mu\rho} R_{\rho\nu} \tag{4.24}$$

This expression is the one used in [17] in the author's approach of the minimization problem over the space of rotations using a gradient method with rotations matrices represented by their exponential form. The reader is referred to the latter article for more mathematical details on the way to derive the expressions used here.

(4.23) and (4.24) thus allow us to write :

$$\begin{aligned} \frac{\partial r}{\partial \delta\omega_\alpha} &= \varepsilon_{ijk} \bar{R}_{jn} D_{mnr} \left[\frac{\partial R_{im}}{\partial \delta\omega_\alpha} R_{kr} + R_{im} \frac{\partial R_{kr}}{\partial \delta\omega_\alpha} \right] & (4.25) \\ &= -\varepsilon_{ijk} \bar{R}_{jn} D_{mnr} \left[\varepsilon_{\alpha i\rho} R_{\rho m} R_{kr} + \varepsilon_{\alpha k\beta} R_{im} R_{\beta r} \right] & (4.26) \end{aligned}$$

(4.26) gives us the expression of partial derivatives of r with respect to the components of the Rodrigues vector associated with the rotation R. For the components $\bar{\omega}_\alpha$ related to rotation $\bar{R}$, we have a similar expression :

$$\frac{\partial r}{\partial \delta\bar{\omega}_\alpha} = -\varepsilon_{ijk} \varepsilon_{\alpha j\rho} \bar{R}_{\rho n} R_{im} R_{kr} D_{mnr} \tag{4.27}$$

At last, we need to mention an implementation detail about the minimization approach. Theoretically, residuals and jacobians are evaluated in a closed form i.e in a "single" operation. Practically, the different sums involved are implemented as loops meaning the latter entities are computed incrementally. This principle sort of applies to the resulting jacobian for each step of the minimization process : it is constructed as a concatenation of the jacobians of all involved pairs of matching cubes in the residuals of expression (4.9). Fig. 4.5 shows the procedure adopted. It is easy to observe that the size

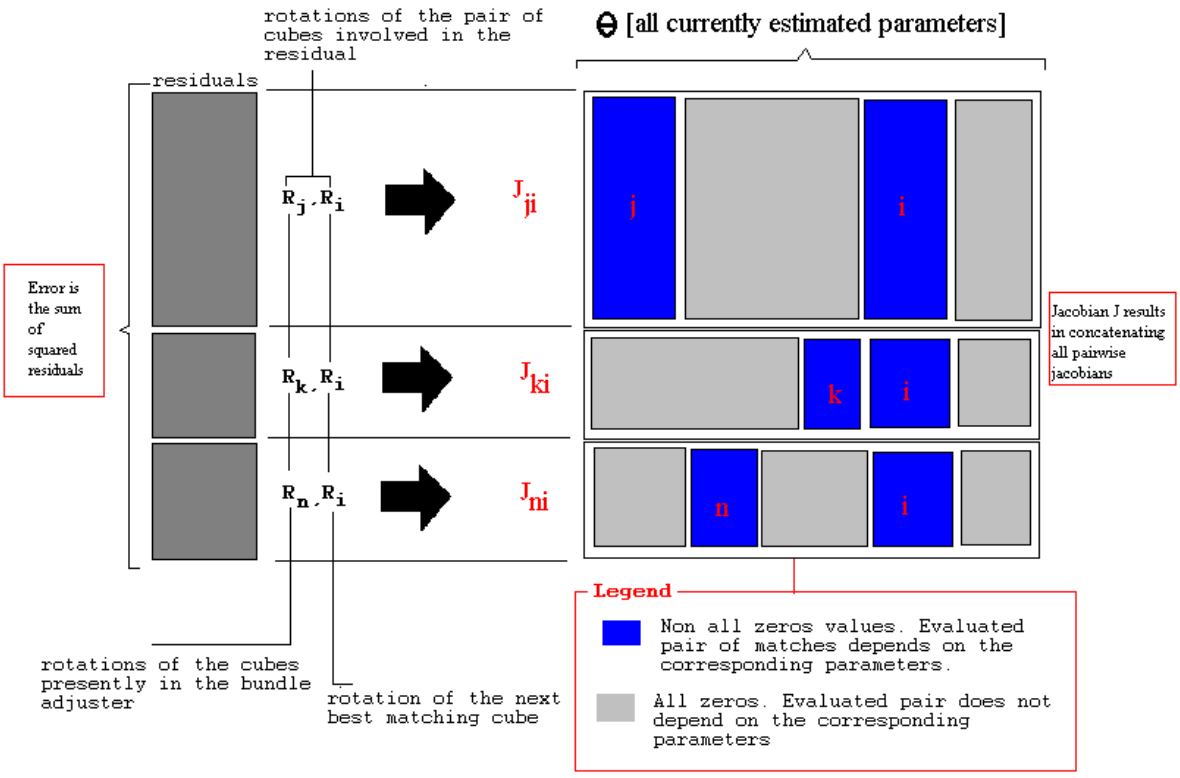


Figure 4.5: Concatenation principle in the residuals and jacobians evaluation. Example with the bundle adjustment algorithm with four cubes, the fourth cube of index i being added to the adjuster that already contains cubes j, k and n.

of the vector of residual vector, for which all elements will be squared and summed up, increases with the number of matches between a given pair of cubes. As a consequence, the computation time depends on the number of matches.

This particular step of the jacobian computation is very demanding from a *CPU* point of view. Our *Matlab* implementation proved to be a bit slow since optimized code was not our goal. For the sake of providing an alternative for performance oriented readers, the speed of the algorithm can be increased by using the *sparse matrix* approach mentioned [8].

As a summary, a bundle adjustment approach with a criterion based on the epipolar constraint and the use of pair-wise essential matrices theoretically ensures to find an optimal set of rotations resulting in *cube alignment*. A typical resulting configuration showed in the example in Fig.4.1 is used as input of the second step of the pose recovery procedure : the translations estimation up to a scale.

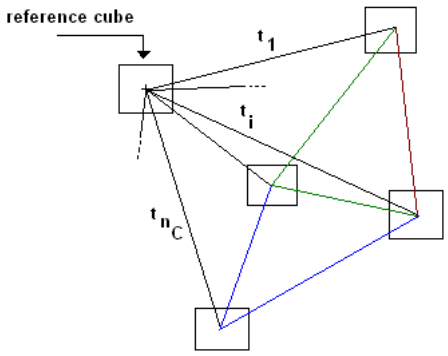


Figure 4.6: Configuration of the translation recovery problem

4.5 Pose estimation and structure recovery : evaluating translations

In the previous section, the cube alignment created a new set of N cubes with negligible rotations with respect to the world coordinate frame. Recall that the resulting set of panoramas after alignment is used as input for the translations estimation step. Thus, to complete the pose recovery, one only has to estimate the position of each of the new camera frames with respect to the world, up to a scale, consequently building a graph with all cameras as nodes and pair-wise translations as vertices (see Fig.4.6).

This stage relies mainly on the principle used in "classic" planar image pose recovery [32, 23] where the $3D$ points corresponding to matches are recovered at the same time as the motion parameters of the camera, both up to a scale and through a single optimization procedure. The criterion used here however is simpler due to the assumption that all rotations are neglected (all rotation matrices are assumed to be equal to the identity matrix, since they have been estimated in the previous step). Thus, we propose in the upcoming sections, an explanation of the criterion chosen and of the optimization procedure are given followed by the initialization step necessary for the optimization procedure. We end this section with some precision about the quantification of the update during the optimization.

4.5.1 Minimization criterion

In the case of images, as mentioned by [32] for example, the final solution results in extrinsic parameters (rotation and relative translation) for all cameras that were used as well as approximate $3D$ points corresponding to processed matches. The criterion that is minimized is the re-projection error illustrating the fact that for all computed $3D$ points, their projection in each camera should ideally correspond to the image points of the original matches that were used; this meaning that in each image planes, the euclidian distance between each pair from both fore-mentioned sets ($3D$ points re-projections and matches)should be null.

The situation dealt with in this section is one for which the rotations are already found and the calibration i.e the intrinsic parameters are already known. The criterion is therefore simplified since only translations or relative positions are involved. Fig.4.7 illustrates the choice of the minimization criterion. Very similar to what is done with regular $2D$ images, this criterion should insure that the re-projection of all $3D$ points corresponding to matches in all cubes should be equal to the original matches in respective cube frames. In other terms, if we insure that, with respect to each cube reference frame, the ray through the center of the cube of concern and the $3D$ point ray_1, and the ray through the corresponding match in the same cube frame ray_2 are collinear and of same direction, then we guarantee a minimal re-projection error. This can be expressed by a simple dot product on normalized direction vectors since for two *unit* vectors v_1 and v_2, their dot product is by definition :

$$v_1 \bullet v_2 = \cos(v_1, v_2) \tag{4.28}$$

Moreover, if v_1 and v_2 are aligned and of same direction $v_1 \bullet v_2 = cos(v_1, v_2) = 1$ since the angle must be $0(2\pi)$. Therefore to verify if two unit vectors v_1 and v_2 are collinear and of same direction, it is sufficient to test for :

$$v_1 \bullet v_2 = 1 \text{ or } 1 - v_1 \bullet v_2 = 0 \tag{4.29}$$

To apply (4.29) to the studied case, let us consider X_i the i^{th} $3D$ point estimated, t_j the relative position of cube C_j with respect to the world reference frame and finally p_{ij} the original match corresponding to the projection of X_i in C_j. The residual for this particular point is given by :

$$r_{ij} = 1 - \tilde{p}_{ij} \bullet \frac{(X_i - t_j)}{N_{ij}} \tag{4.30}$$

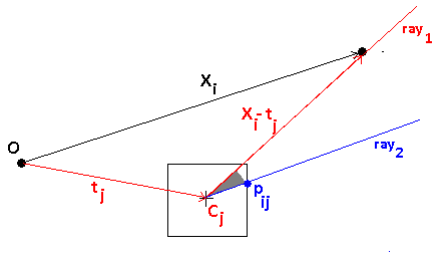


Figure 4.7: Minimization criterion illustration.

With :

$$\tilde{p}_{ij} = \frac{p_{ij}}{\|p_{ij}\|} \quad \text{and} \quad N_{ij} = \|X_i - t_j\| \tag{4.31}$$

Using the tensor notation introduced in section 4.2.1 to express the dot product (see equation (4.1)), one can rewrite (4.30) as :

$$r_{ij} = 1 - \tilde{p}_{ij}^{\alpha} \frac{(X_i^{\alpha} - t_j^{\alpha})}{N_{ij}} \tag{4.32}$$

i and j do not imply implicit summations in (4.32) since they appear on both sides of the equations and help specify what residual is being computing. Thus, for all panoramas and all reconstructed points, the criterion is equivalent to *null residuals* for all values of i and j. And overall, if $i \in [1, \cdots, n_X]$ and $j \in [1, \cdots, n_C]$, with n_X and n_C respectively the number of $3D$ points reconstructed and the number of cubic panoramas involved, the error e that is minimized is given by :

$$e = \sum_{i=1}^{n_X} \sum_{j=1}^{n_C} r_{ij}^2 \tag{4.33}$$

This justifies the term residual used to qualify r_{ij} previously. Minimizing this global error results in the best approximate $3D$ reconstruction of the scene of interest. What follows are the steps of such minimization solution as announced previously.

4.5.2 Initialization step : triangulation

Since a global optimization solution was chosen, a first estimate of the $3D$ structure sought after is necessary and will serve as a seed or the algorithm. This means that

one must obtain approximate positions of the cubic panoramas and of the generated $3D$ points from the available data, mainly matches across all panoramas. It is important to note, as an implementation hint, that if for example N cubes are involved, then each $3D$ to-be reconstructed point is associated to at most a $N-tuple$ of matches on all cubes and at least a pair across two panoramas. Triangulation will be used here for computing the first approximate of all the unknowns. It is a simple and fast way to obtain a rough estimate of the structure.

Principle

Triangulation is described in the literature as a straightforward error prone procedure of pose estimation [27]. However, only an approximate of the structure is needed here. The reconstruction error, as it will be shown in the results section, is handled by the optimization step. The principle of triangulation is basically the search for the intersection of two rays in space. The two rays could be generated from matches' back-projections or given by some specified vectors. Both cases are displayed on Fig.4.8.

Let us consider the general setup where two vectors u and v belong respectively to the frames F_u and F_v. These frames are such that F_u is the reference frame and F_v is obtained by applying the rotation R and translation T to F_u. For such a configuration, [27] mentions that the intersection is estimated by the midpoint of the segment of least length between both rays since in general the rays directed by u and v *do not intersect* exactly. [27] also notes that the segment is composed of points on each ray, $p_u = \alpha u$ and $p_v = T + \beta Rv$ both expressed with respect to F_u, that verify :

$$\alpha u - \beta Rv + \gamma(u \times Rv) = T \tag{4.34}$$

Once p_u and p_v are recovered by solving for α and β in the square system given in (4.34) - for example by using the Kramer method - the intersection point p_m is estimated by $\frac{p_u+p_v}{2}$.

It is important however to note that all cubes have the same reference frame up to translation since the relative rotations with respect to each other are supposed to be non existent as a consequence of the alignment. This means that no rotation appears in the general expression of triangulation given by (4.34) which thus suitably becomes :

$$\alpha u - \beta v + \gamma(u \times v) = T \tag{4.35}$$

The latter equation is the one that will be used with cubic panoramas to recover both the type of first estimates that are mentioned above.

For cube positions

The notations on Fig.4.8 are the ones that are followed here. Cubes positions with respect to one another are estimated from panoramas triplets and always with respect to a particular reference cube. A reference cube C_r has to be chosen and will obviously have a position t_r equal to 0.

Next, the *scale* of the reconstruction needs to be fixed by setting the length between the reference cube and a given cube C_k. This is done by computing the essential matrix E_{rk} from the available matches. The choice could be random if all pairs of panoramas have the same number of matches or based on the best matching panorama. Once E_{rk} is found, the unit direction vector t_{rk} is extracted using the method in appendix D. The position of cube C_k, that is not a priori known since only the direction is known, is then fixed to be $t_k = t_{rk}$. The length between both cubes is therefore equal to 1 and represents the unit of measure for all subsequent estimations even for $3D$ points.

With the previous two initializations in place, it is then possible to recover all other cubes positions estimates from available matches up to a scale factor fixed before hand. The procedure involves triplets of cubes for example (C_k, C_l, C_m). As shown in Fig.4.8, epipoles e_{km} and e_{lm} need to be recovered implying the computation of essential matrices E_{km} and E_{lm} (see Chapter 2). Equation (4.35) is then applied to the given cubes resulting in :

$$\alpha e_{km} - \beta e_{lm} + \gamma(e_{km} \times e_{lm}) = t_{kl} \tag{4.36}$$

α and β are solved for and allow us to recover the position t_{km} of C_m wit respect to C_k as follows :

$$t_k m = \frac{\alpha e_{km} + (t_k l + \beta e_{lm})}{2} \tag{4.37}$$

And overall, the position of cube C_m with respect to the reference cube C_r is given by :

$$t_m = t_k + t_{km} \tag{4.38}$$

Choosing the reference cube and the scale of the reconstruction allows us to recover all estimates of cubes positions by using the suitable triplets. If many positions are found

for one cube as a result of many triplets involving the same panorama, then the average is used as the first estimate.

For $3D$ points

The notations on Fig.4.8 are also valid in the present section. The estimated position of the $3D$ point X_i with respect to the frame of cube C_k will be obtained by first solving a triangulation formula similar to (4.35):

$$\alpha p_{ik} - \beta p_{il} + \gamma(p_{ik} \times p_{il}) = t_{kl} \tag{4.39}$$

Once the values of α and β are recovered, X_i is given with respect to the frame of C_k by :

$$X_{ik} = \frac{\alpha p_{ik} + (t_{kl} + \beta p_{il})}{2} \tag{4.40}$$

If the cube C_k is the reference then X_ik is the $3D$ position X_i sought after. Otherwise, it is more generally given by :

$$X_i = t_k + X_{ik} \tag{4.41}$$

Where t_k stands for the estimated cube C_k's position as described in the previous section. Recall that each $3D$ point is associated to at most a $N - tuple$ of matches if N is the number of cubes. Therefore, for one $3D$ point, there are as many estimates as the number of pairs of matches in a $N - tuple$. The final estimate could be the average of all estimates.

All the steps above are repeated for each of the $3D$ points. This result in a cloud of points in space that sparsely or discreetly describe the scene of interest.

4.5.3 Jacobian estimation : residual derivation

The variables of the optimization algorithm minimizing the error given in (4.33) are the $3D$ points of the scene as well the positions of each one of the panoramas. The initialization procedure explained above described a simple way to get a rough approximate of all variables grouped under $\Theta = \{X_1, \cdots, X_{n_X}, t_1, \cdots, t_{n_C}\}$ - recall n_X and n_C are respectively the number of reconstructed points and the number of panoramas. Each element of Θ has 3 components, its coordinates with respect to the world reference frame

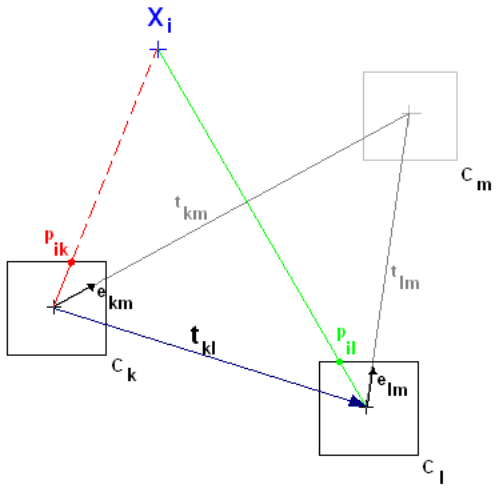


Figure 4.8: Triangulation for $3D$ points and for cubic panoramas.

which coincide with the reference cube's coordinate system. An extended expression of Θ is therefore :

$$\Theta = \{X_1^x, X_1^y, X_1^z, \cdots, X_{n_C}^x, X_{n_C}^y, X_{n_C}^z, t_1^x, t_1^y, t_1^z, \cdots, t_{n_C}^x, t_{n_C}^y, t_{n_C}^z\} \tag{4.42}$$

To express the progression of the minimization through the variation of the error function, the jacobian associated to all residuals has to be computed. As a consequence, it is necessary to establish the derivative of any residual r_{ij} as given in 4.32 with respect to each one of the elements of Θ, the variable vector. We have established the following using derivation properties such as the derivative of a ratio and a square root, and the fore-mentioned tensor notation (section 4.2.1):

$$\begin{cases} \text{If } \Theta_m \in \{X_1, \cdots, X_{n_X}\}, & \frac{\partial r_{ij}}{\partial \Theta_m^\beta} = \delta_{mi}(-\tilde{p}_{ij}\frac{\delta_{\alpha\beta}}{N_{ij}} + \frac{1-r_{ij}}{N_{ij}^2}(X_i^\beta - t_j^\beta)) \\ \text{If } \Theta_m \in \{t_1, \cdots, t_{n_C}\}, & \frac{\partial r_{ij}}{\partial \Theta_m^\beta} = \delta_{mj}(\tilde{p}_{ij}\frac{\delta_{\alpha\beta}}{N_{ij}} - \frac{1-r_{ij}}{N_{ij}^2}(X_i^\beta - t_j^\beta)) \end{cases} \tag{4.43}$$

A total of $n_X n_C$ residuals r_{ij} are computed at each iteration and as a result the jacobian is a $n_X nC$ by $3(n_X + n_C)$ sparse matrix. No additional step has been implemented here but it was mentioned earlier that a more efficient jacobian matrix form was given by the *sparse matrix* solution presented in [8]. As an implementation note, the optimization algorithm was performed in Matlab using the function *lsqnonlin* with activated jacobian estimation.

This concludes the translation estimation step and at the same time the two-stage algorithm for pose recovery. The positions of all panoramas are recovered from the previously computed aligned configuration up to a scale. The $3D$ positions of the features used in the process are also estimated at the same scale given us a preview of the structure of the scene. The pose recovery problem is therefore solved theoretically through these two steps and results obtained on real panoramas sets are given in the following section to illustrate the feasibility and validity of such a procedure.

4.6 Results

This section presents the results obtained for two different sets of panoramas. The first set depicts an indoor laboratory scene and the second one, an outdoor scene. For each set, each step of the two stage algorithm is presented one after the other. The indoor example is extensively commented so that the different steps of each algorithm are easily identifiable; the effects of the algorithms on the panoramas as far as the visual aspect is concerned are also pointed out. The second set is mainly another illustration of the kind of results that can be obtained especially in an outdoor less controllable setup, where features can be difficult to track in all panoramas.

4.6.1 Indoor Scene

Rotations estimation and alignment

To present the results obtained after applying the alignment procedure, we have chosen two different approaches. A visual one where the viewer or the reader can inspect the effect of the algorithm on the cubes and another one a little more formal to have some kind of numerical and theoretical backup of these results.

Our first experiment was conducted on four randomly captured cubic panoramas in a laboratory. An approximative map of the laboratory with the panorama locations is given in Fig.4.9. The scene is rich in features that were easily hand picked, therefore no automated matching process was used here. The cubes that were processed can all be seen on Fig.4.10(a), 4.11(a), 4.12(a), 4.13(a). The cross pattern observed here is one introduced in chapter 3 and it allows us to see the cubes as a whole since it is not possible to use a proper 3D viewer. The aligned cubes are displayed in Fig.4.10(b), 4.11(b), 4.12(b), 4.13(b). These cubes are not at their full resolution and as consequence it is a little difficult to perceive clearly some details of the scene. But in general, we

can observe that for example, the background of all front faces contain identical items in similar positions. The rotations obtained from the algorithm are as follow :

$$R_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.44}$$

$$R_2 = \begin{bmatrix} 0.8231 & -0.1518 & 0.5472 \\ 0.0251 & 0.9724 & 0.2319 \\ -0.5673 & -0.1771 & 0.8043 \end{bmatrix} \tag{4.45}$$

$$R_3 = \begin{bmatrix} -0.4587 & -0.1903 & 0.8680 \\ 0.0902 & 0.9618 & 0.2585 \\ -0.8840 & 0.1968 & -0.4240 \end{bmatrix} \tag{4.46}$$

$$R_4 = \begin{bmatrix} 0.9108 & -0.3989 & -0.1060 \\ 0.4027 & 0.8023 & 0.4406 \\ -0.0907 & -0.4440 & 0.8914 \end{bmatrix} \tag{4.47}$$

It is interesting to note that the first rotation is the identity matrix (recall the initialization step in section 4.4.3) meaning that the cubes 2 to 4 are aligned with respect to the frame of cube 1. Visually and as we said before, we can see for example that the front face of the cubes all point to the same area of the scene particularly the two computer monitors well visible in Fig.4.12(b). The same can be said by examining the top face or all other faces in correspondence.

Fig4.15 and Fig.4.16 also illustrate alignment. As a matter of fact, the "truncated" versions of the cubes that are shown in these images (top and bottom faces were removed) are vertically concatenated to ease the observation of the effect of alignment. Once more, the same scene elements appear in each aligned cube image for a the same viewing direction for instance the windows in the background. With a little more observation, the viewer can also perceive some kind of uniformity in the tilt level of the camera which seems to be equal in all presented images of Fig.4.16.

In a more formal way there are many details that allow us to confidently state the accuracy and the efficiency of our approach. First the value of the final residual at the last iteration of the algorithm numerically shows the accuracy of the solution that was found. The residuals are computed for each of the pairs of cube matches involved in the algorithm.

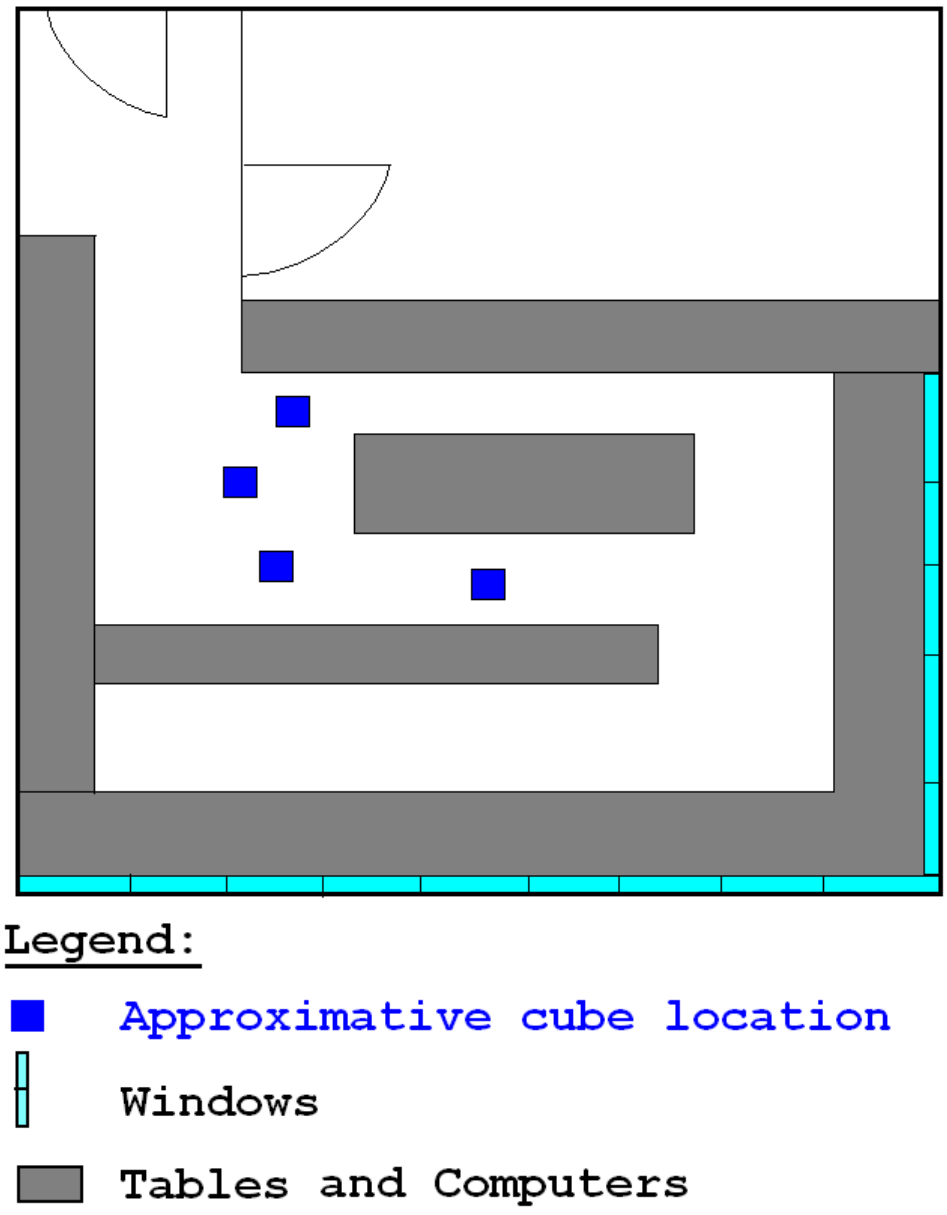
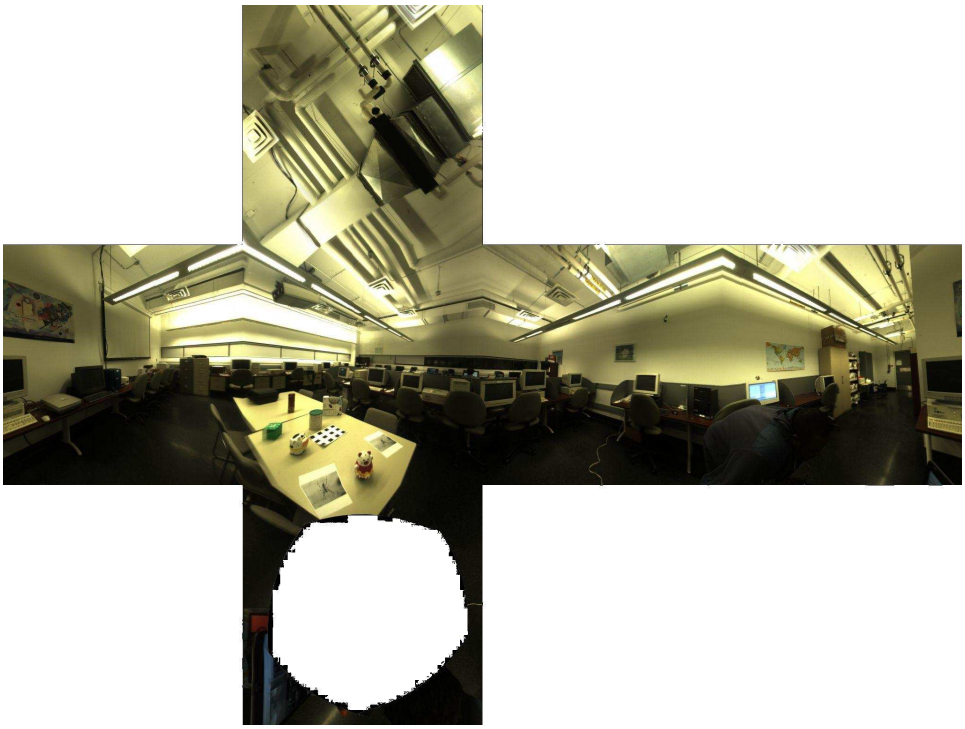
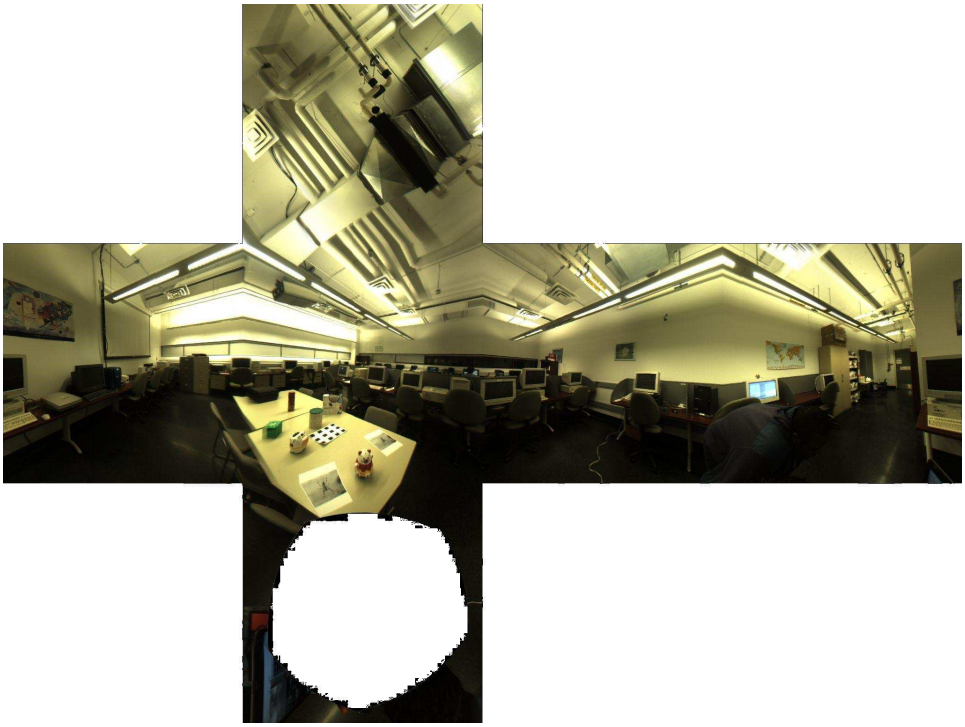


Figure 4.9: Floorplan of the laboratory with approximative cubes locations

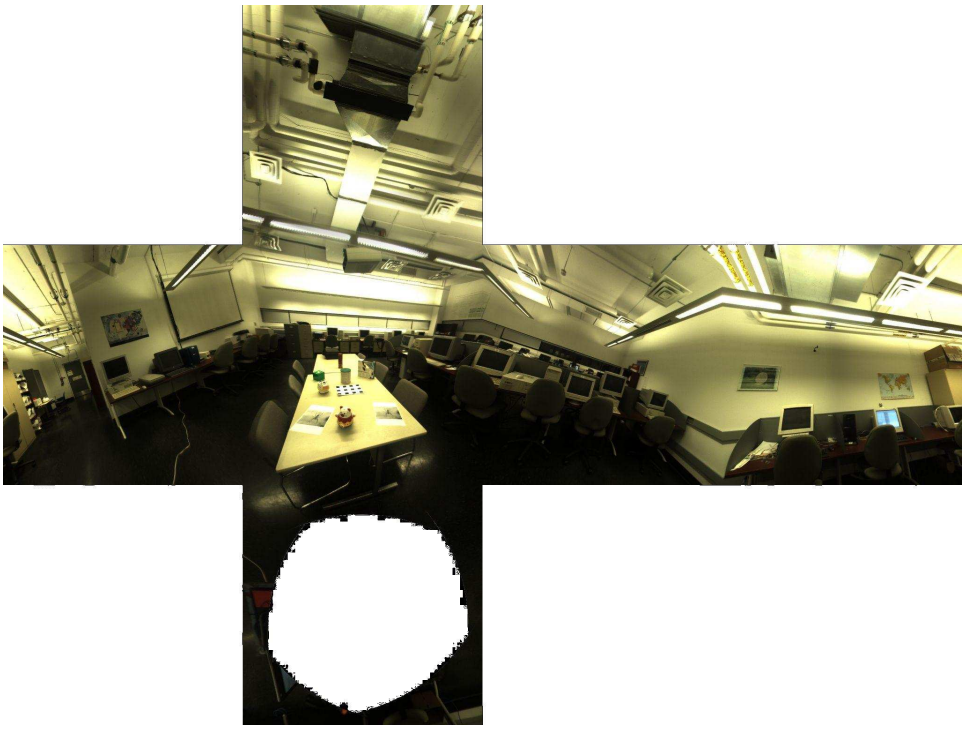


(a) Cube 1 before alignment.



(b) Cube 1 after alignment.

Figure 4.10: Cube 1 before and after alignment



(a) Cube 2 before alignment.

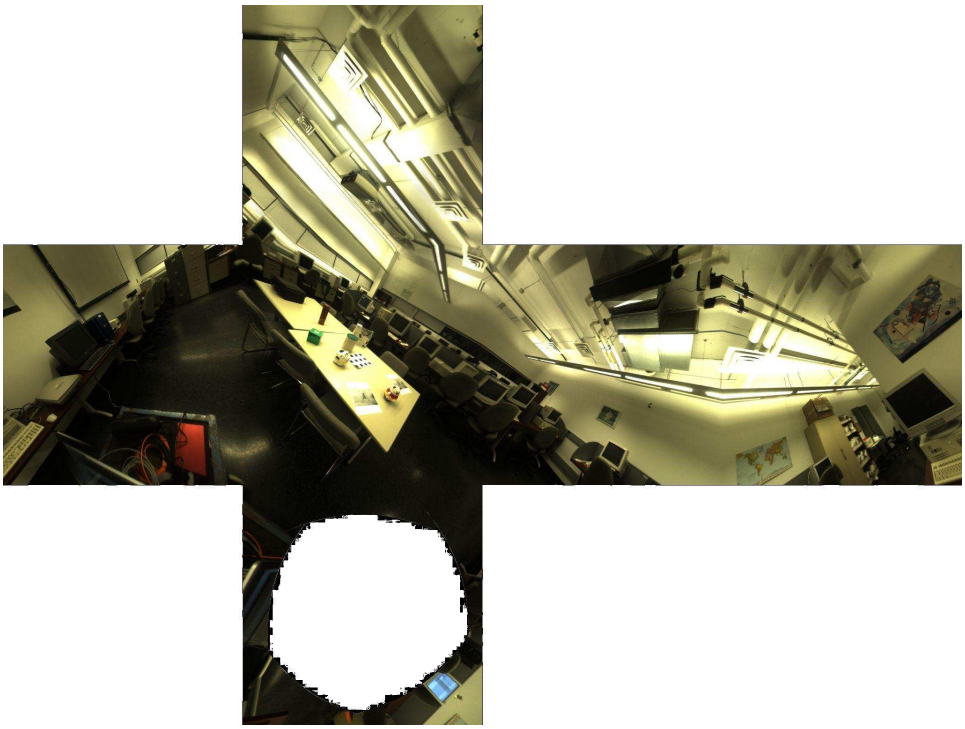
(b) Cube 2 after alignment.

Figure 4.11: Cube 2 before and after alignment

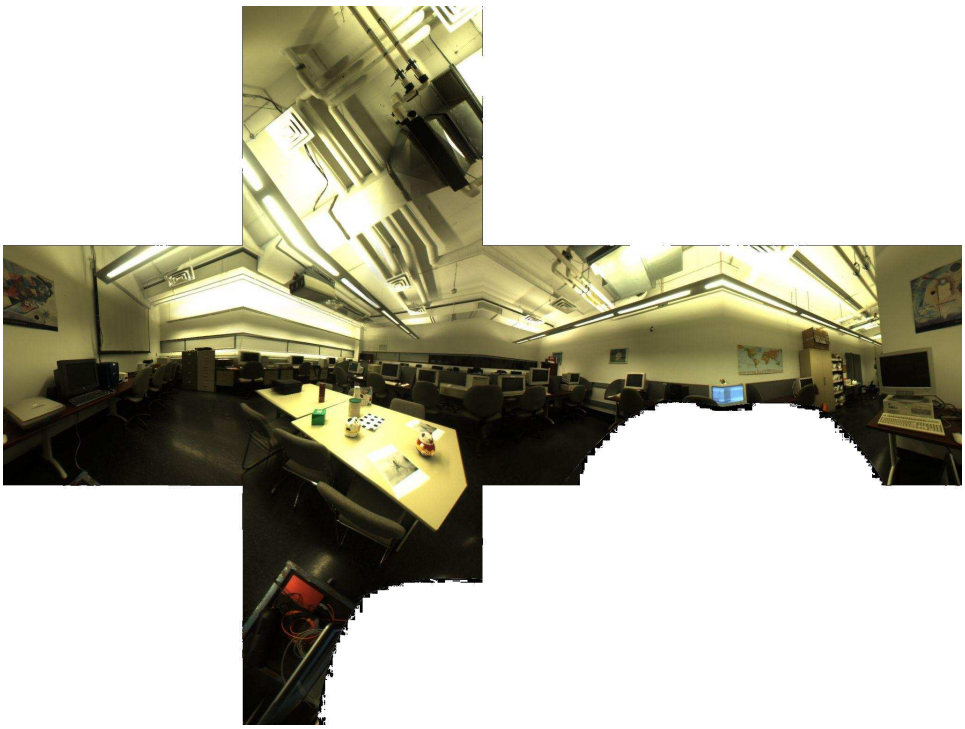
(a) Cube 3 before alignment.

(b) Cube 3 after alignment.

Figure 4.12: Cube 3 before and after alignment



(a) Cube 4 before alignment.



(b) Cube 4 after alignment.

Figure 4.13: Cube 4 before and after alignment

The graph in Fig.4.6.1 shows the residuals obtained once the cube 4 has been added to the bundle containing already cube 1 to 3, this in with respect to each pair of cubes matches as well as for each pair of cubes.

Summing the squared values of these residuals gives us the total error of 2.3685×10^{-4}. This confirms numerically that the result is close to optimum. This is one possible way to quantify the validity of the algorithm that was applied. Another efficient way to validate it is to use our knowledge of epipolar geometry. Thus, we know that aligned cubes present a particular geometric configuration in which the rotation between any pair of cubes should be non existent i.e the identity matrix. As a consequence, if we recall that the essential matrix of a pair of cubes that differ by a rotation R and a translation unit vector t is given by :

$$E = [t]_{\times} R$$

We have that, with $R = I_3$, all final essential matrices linking cubes in the bundle should have a form close to the following :

$$E = [t]_{\times} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \tag{4.48}$$

In other words, the essential matrices obtained with aligned cubes should be as close as possible - ideally equal - to an antisymmetric matrix materializing the sole transformation between both involved cubes frames i.e a translation of known estimated value. This was verified for a few pairs of cubes used in the example. Some results are given in tables 4.1 and 4.2 where for a specific pair of cubes we give the original estimated essential matrix (resp. rotation matrix) , the essential matrix (resp. rotation matrix) after alignment; in both cases the essential matrix is noted $e.m.$

For all matrices in the aligned case in table 4.1, we observe the similarity with an anti-symmetric matrix. The diagonal elements are all close to zero and others terms are almost all symmetric with opposite signs. For table 4.2, all original rotations are almost completely "removed" by the alignment, meaning that they all become almost equal to the 3 by 3 identity matrix. These observations confirm what was established theoretically, obviously up to some numerical errors that mainly originate from the optimization/minimization algorithm. Nonetheless the results are really satisfying both visually and numerically.

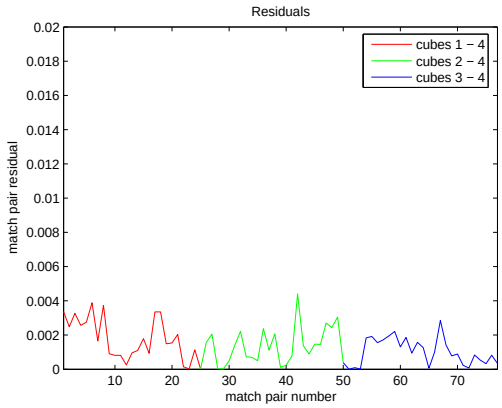


Figure 4.14: Residuals at the last iteration of the bundle adjustment for the rotations estimation.

pair	e.m before	e.m after
$1-2$	$\begin{bmatrix} -0.019 & -0.085 & -0.027 \\ -0.143 & -0.085 & -0.308 \\ -0.016 & 0.332 & -0.097 \end{bmatrix}$	$\begin{bmatrix} 0.003 & -0.127 & 0.035 \\ 0.144 & 0.008 & 0.322 \\ -0.028 & -0.330 & 0.008 \end{bmatrix}$
$2-3$	$\begin{bmatrix} 0.018 & 0.124 & 0.049 \\ 0.281 & 0.013 & 0.211 \\ 0.070 & -0.322 & -0.044 \end{bmatrix}$	$\begin{bmatrix} -0.004 & -0.355 & -0.018 \\ 0.356 & -0.005 & 0.002 \\ 0.010 & 0.030 & 0.002 \end{bmatrix}$
$3-4$	$\begin{bmatrix} -0.138 & -0.260 & 0.019 \\ -0.213 & 0.186 & 0.201 \\ -0.152 & -0.131 & 0.067 \end{bmatrix}$	$\begin{bmatrix} 0.008 & -0.307 & -0.006 \\ 0.310 & 0.013 & 0.177 \\ 0.019 & -0.183 & 0.005 \end{bmatrix}$

Table 4.1: Essential matrices of cube pairs 1-2, 2-3, 3-4.

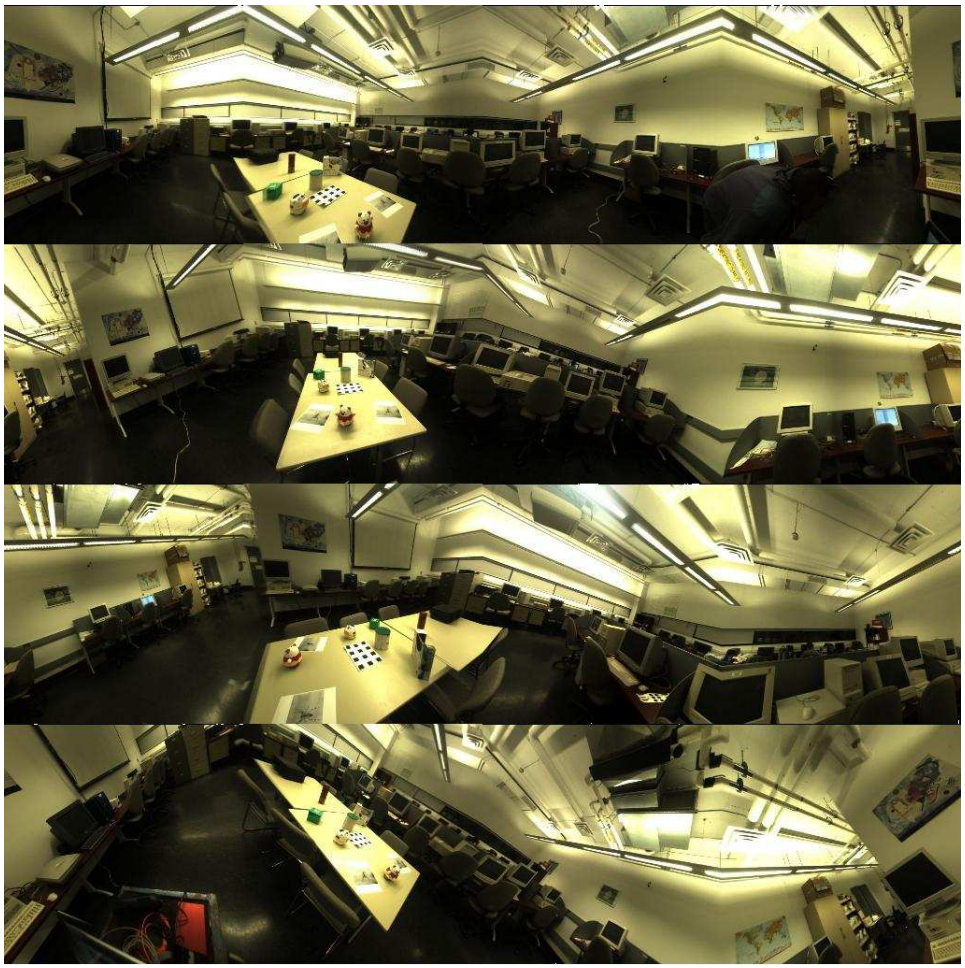


Figure 4.15: Indoor cubic panorama set before alignment (top and bottom faces truncated)

Figure 4.16: Indoor cubic panorama set after alignment (top and bottom faces truncated)

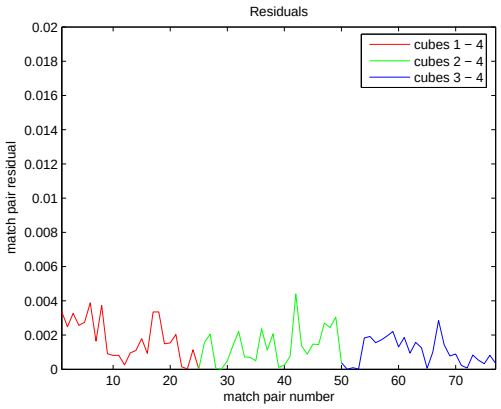


Figure 4.17: Residuals at the last iteration of the bundle adjustment for cubes 1 to 4.

pair	rotation before	rotation after
$1-2$	$\begin{bmatrix} 0.790 & -0.171 & 0.589 \\ 0.020 & 0.967 & 0.253 \\ -0.613 & -0.188 & 0.767 \end{bmatrix}$	$\begin{bmatrix} 0.999 & -0.011 & 0.052 \\ 0.010 & 1.000 & 0.020 \\ -0.053 & -0.020 & 0.998 \end{bmatrix}$
$2-3$	$\begin{bmatrix} 0.229 & -0.248 & 0.941 \\ 0.333 & 0.929 & 0.164 \\ -0.915 & 0.276 & 0.295 \end{bmatrix}$	$\begin{bmatrix} 0.996 & 0.015 & -0.090 \\ -0.017 & 1.000 & -0.023 \\ 0.090 & 0.024 & 0.996 \end{bmatrix}$
$3-4$	$\begin{bmatrix} -0.266 & 0.667 & -0.695 \\ 0.212 & 0.744 & 0.633 \\ 0.940 & 0.021 & -0.339 \end{bmatrix}$	$\begin{bmatrix} 0.999 & -0.026 & 0.019 \\ 0.025 & 0.999 & 0.029 \\ -0.020 & -0.029 & 0.999 \end{bmatrix}$

Table 4.2: Rotations matrices of cube pairs 1-2, 2-3, 3-4.

Pose estimation

The cubic panoramas obtained after alignment are the starting point of the pose estimation. New matches are found across all panoramas (60 for each of the 4 cubes in our case). These matches are used to perform the essential matrix computation, the initialization through multiple triangulations and finally the optimization algorithm.

Fig.4.18 displays the reconstruction error as a set of residuals (240 to be more accurate). It is quite a low error - average residual error of 0.0005 - indicating that the final solution is at least numerically very accurate. The figures 4.19(a), 4.19(b) and 4.19(c) respectively depict a perspective view, side view (along the Z axis) and top view (along the Y axis) of the reconstructed scene with some elements of the scene indicated in Fig.4.6.1 as an indication to what is pointed out in the cubes. The elements are constructed from their corresponding features, using polygonal shapes that are easily recognizable. The size of the cubes in the different views is irrelevant since only their positions is of interest here.

A few comments come to mind especially when observing the perspective view and the elements of the scene shown in Fig.4.6.1. First, on the perspective view, the elements of the scene in gray are all initial positions of points of the scene before optimization. The effect of the pose estimation algorithm can therefore be seen as elements move closer or further to the center of the system coordinate as well as change shape to more recognizable forms. The windows in the background of the scene in dark blue are a very good example; so is the cyan square shape observable on the ceiling of the laboratory.

Next, a closer observation of figures 4.19(b) and 4.6.1 allows us to confirm that the reconstruction is spatially consistent with the visual information : the elements of the scene are where they are expected to be. The view along the Z axis i.e the side view positions the user as looking towards the front face of the cubes.

Finally, Fig.4.21 illustrates the complete pose recovery in $3D$ since the estimated original positions of all cubes in space are approximately reproduced.

4.6.2 Outdoor Scene

Rotations estimation and alignment

For the outdoor set, that contains 6 panoramas instead of 4 as previously, the cube alignment results are shown as it was done earlier with modified versions of the panoramas to ease the observation. Theoretically, we already established that corresponding faces in

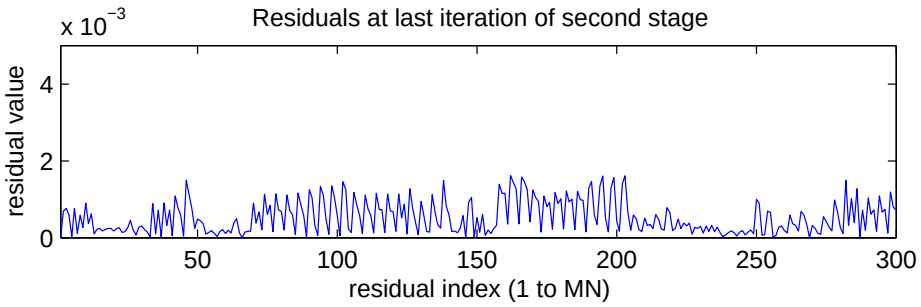


Figure 4.18: Residuals at the last iteration of the second stage for the indoor set : residuals r_{ij}, $i \in \{1, \cdots, 75\}$ for the matches and $j \in \{1, \cdots, 4\}$ for the panoramas.

all cubes should be parallel after alignment and therefore roughly display similar scene elements in similar positions. Fig.4.22 and Fig.4.23 show the panoramas respectively before and after alignment. Note for example the presence of the ramp by the stairs in the direction of the front face of all cubes.

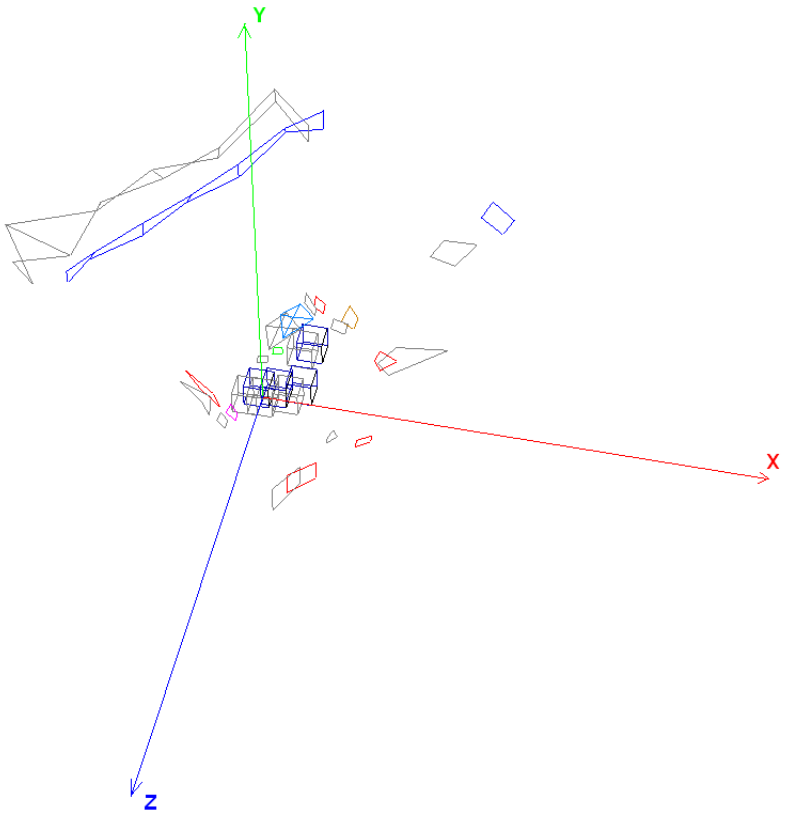
Pose estimation

To complete the study on the outdoor set, the complete pose recovery was performed and the results are displayed as a perspective and top view (Fig.4.25) of the scene with mainly elements of the closest building. Figure 4.24 shows the residual errors after the last iteration of the second stage with an average residual value of 0.001. The higher values of residuals obtained toward the end of the graph correspond to matches that include feature points that were inaccurately localized across the cubes. This is in part due to the considerable change in viewpoints of the camera in terms of translation. Finally Fig.4.26 shows the complete pose recovery from a cube point of view, with panoramas in their estimated original configuration.

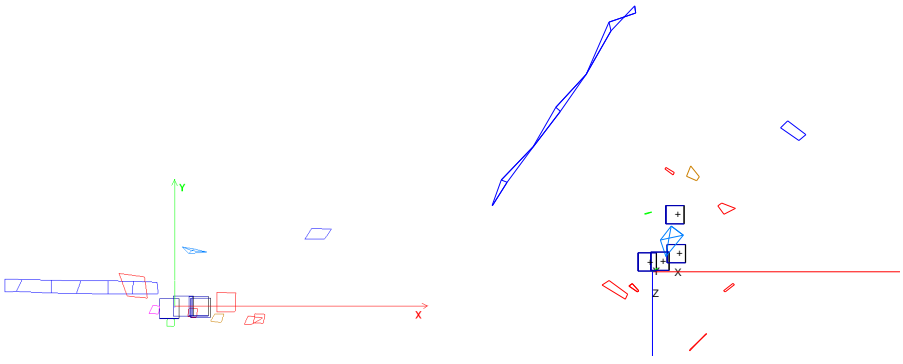
4.7 Conclusion

This chapter presented another possible application of cube epipolar geometry in addition to cube rectification. As a matter of fact a two stage algorithm to achieve pose recovery for a given set of panoramas was presented here.

The first step consisted in what was designated panorama "alignment" i.e rotations recovery. Unlike the cube rectification that was presented in the previous chapter, the



(a) Perspective view of the reconstructed scene with the aligned cubic panoramas (Indoor Scene).



(b) Side view

(c) Top view

Figure 4.19: Views of the reconstructed scene (Indoor Scene)

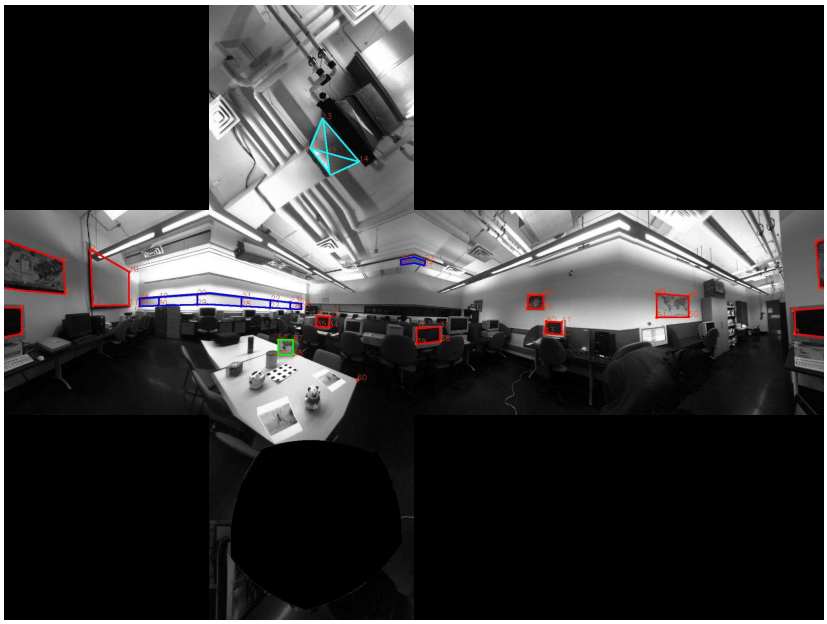


Figure 4.20: Elements of panoramas shown in the reconstructed scene (Indoor Scene)

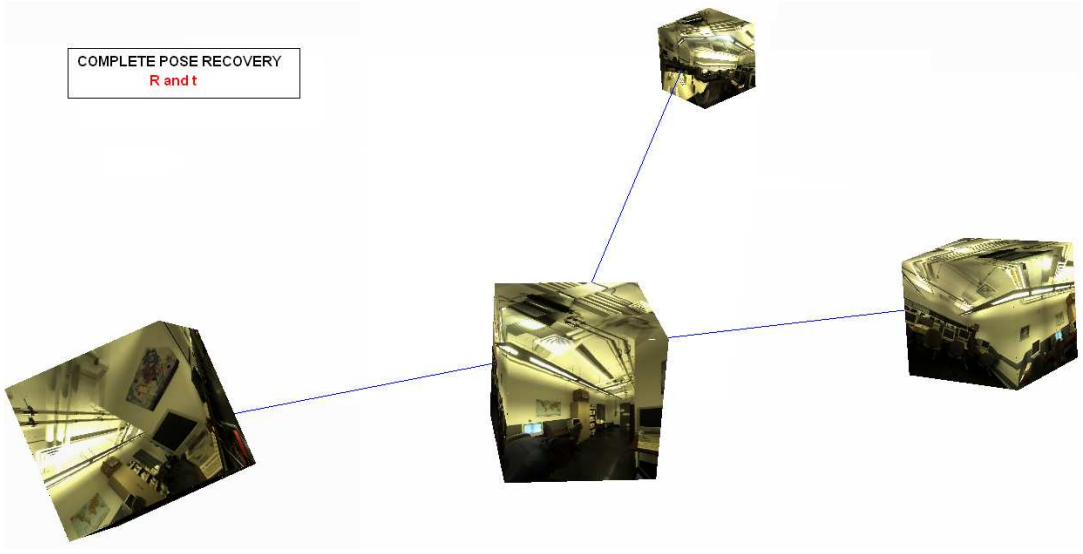


Figure 4.21: Pose recovery for the indoor set of panoramas

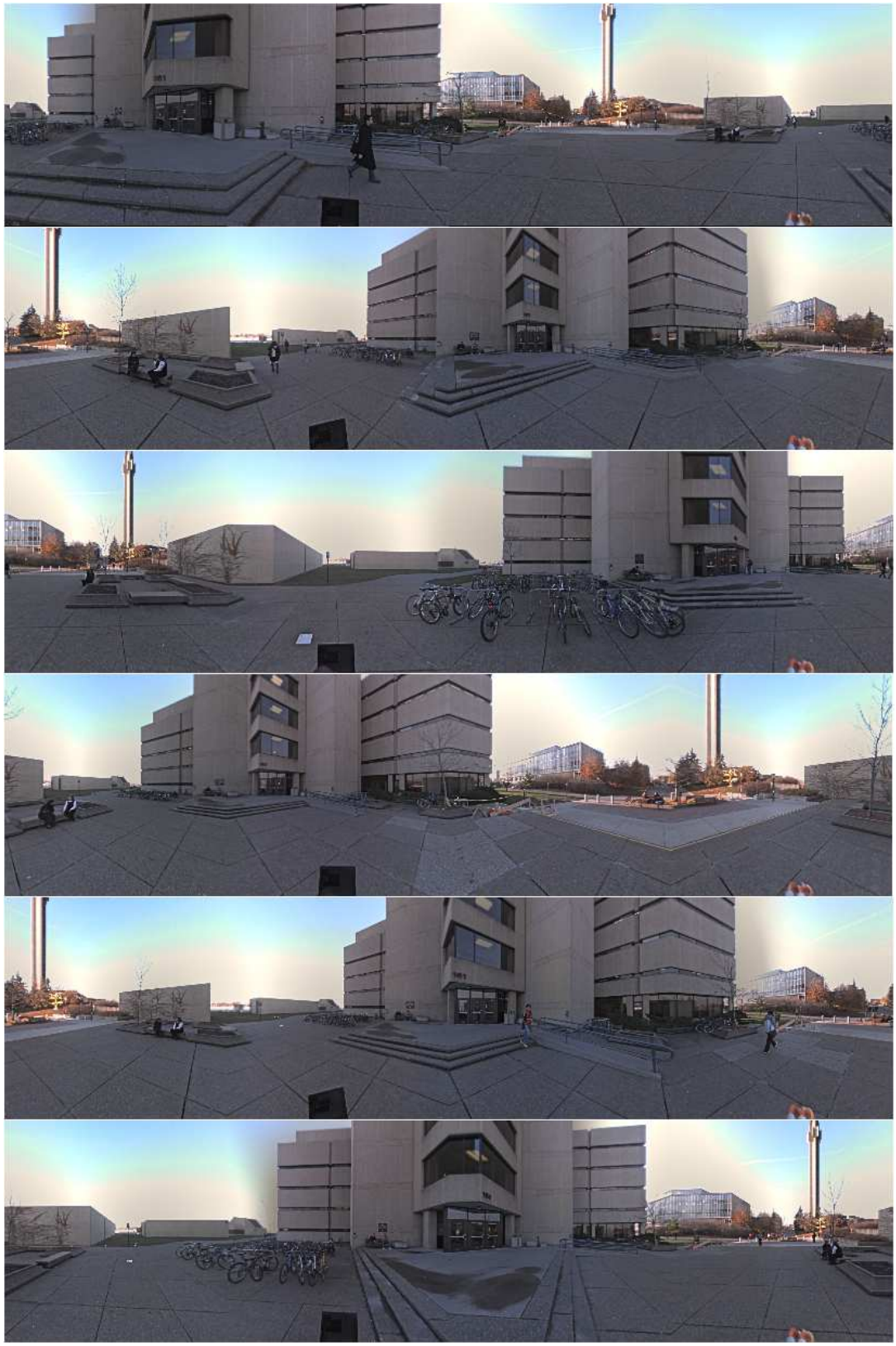


Figure 4.22: Outdoor cubic panorama set before alignment (top and bottom faces truncated)

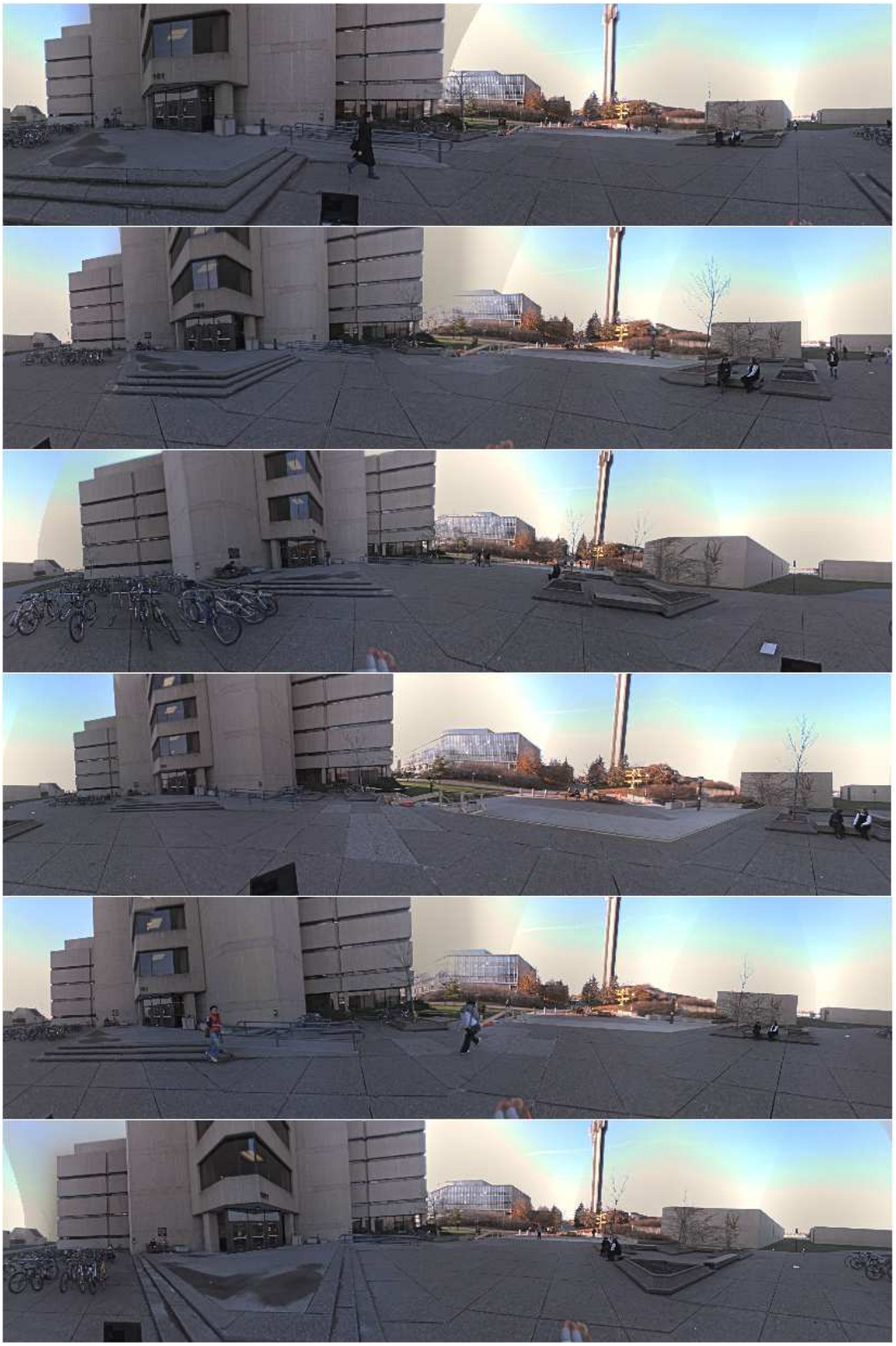


Figure 4.23: Outdoor cubic panorama set after alignment (top and bottom faces truncated)

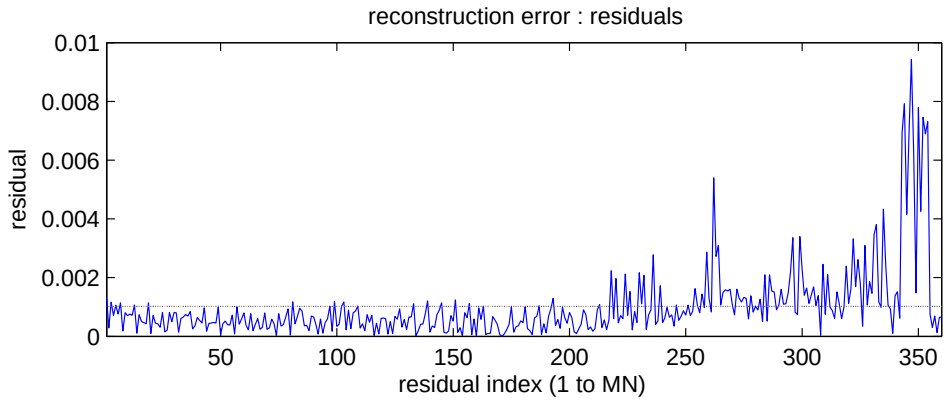


Figure 4.24: Residuals at the last iteration of the second stage for the outdoor set : residuals r_{ij} , $i \in \{1, \cdots, 60\}$ for the matches and $j \in \{1, \cdots 6\}$ for the panoramas

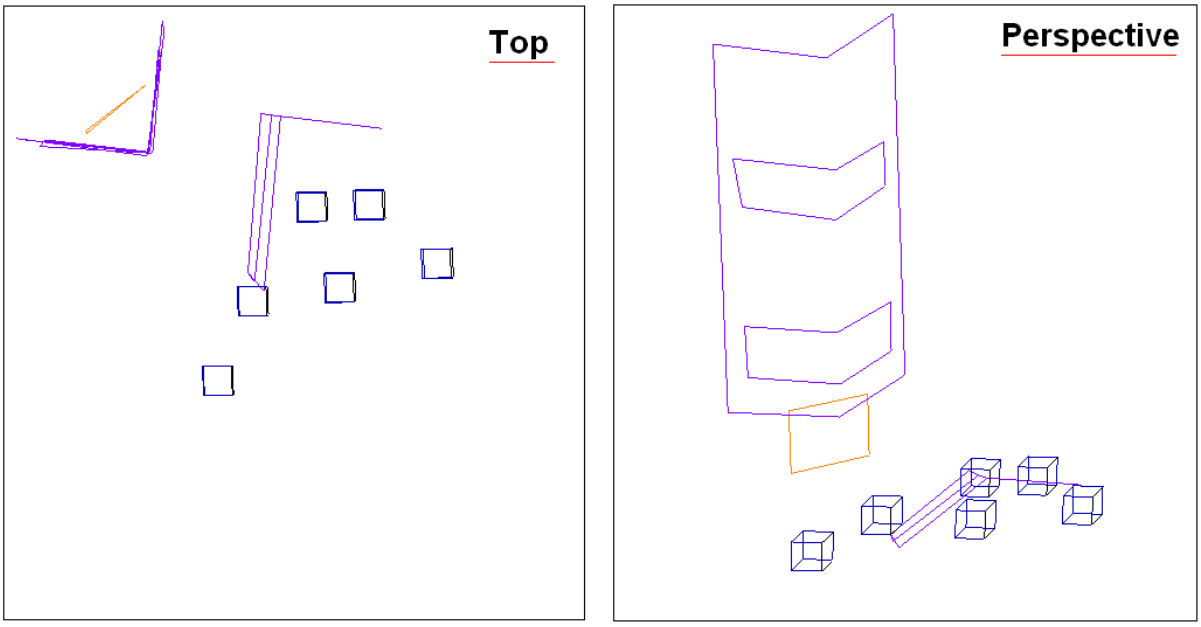


Figure 4.25: Outdoor scene with a few objects and the 6 aligned panoramas (cubes) at their computed locations.

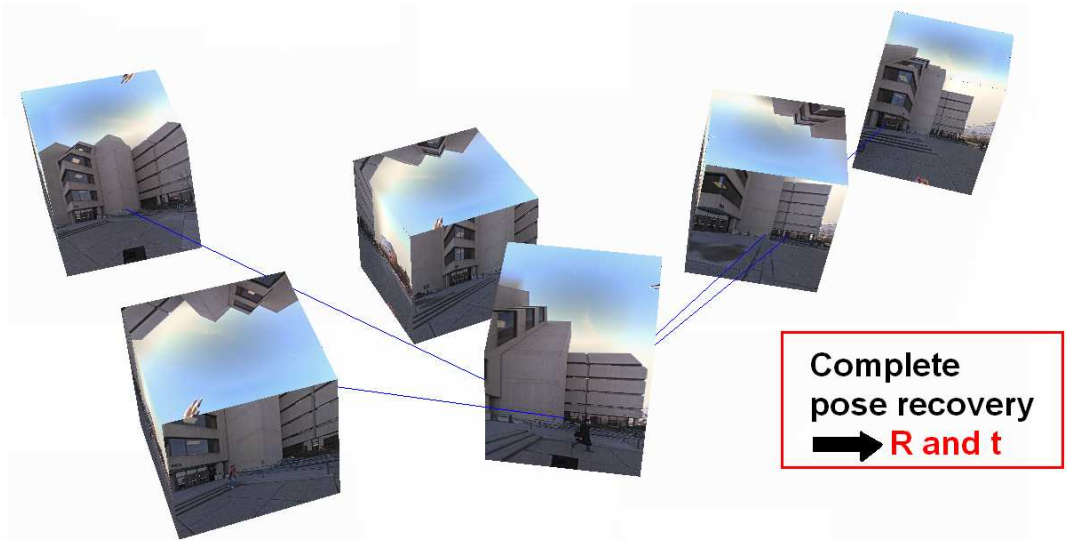


Figure 4.26: Pose recovery for the outdoor set of panoramas

alignment is a multi panorama process that resulted in cubes with corresponding faces all, parallel therefore turned towards a common direction after their rotation with respect to the world frame was found. This configuration was exploited in the second stage of the algorithm.

Thus the second stage used the aligned panoramas to compute the estimated positions of the cubes up to a scale. Basically a minimization of re-projection error criterion was used to refine an initial structure computed through multiple triangulation. Combining this information with the rotations recovered in the first stage allowed us to complete the pose recovery procedure.

Results were given for an indoor and an outdoor sets for both stages of the algorithm. Ultimately, the pose recovery for each one of the sets was illustrated by a $3D$ approximation of the original camera configuration.

Finally, we have noted that the performance of each stage could be improved by using a better approach in the update steps of the optimizations. Moreover, the presented two stage algorithm could greatly benefit from a feature tracking procedure, for a set of cubic panoramas, that would produce enough accurate matches across all of the images and thus improve the overall accuracy of the main process.

Chapter 5

Conclusion

All throughout the study presented in this text, we have gone from implementing the well known solution to stereo image rectification to solving the pose recovery problem in the case of cubic panoramas. All the algorithms presented here relied heavily on state-of-the-art notions and methods of epipolar geometry. In the end they all proved worthy of interest for the results obtained were mostly satisfactory.

As a matter of fact, in chapter 2, a known rectification procedure was extended to suit the trinocular case for particular configuration. A composition of homographies proved to be a simple observation and to provide an efficient solution to the trinocular case as it was shown for horizontal and "L"-shape triplets. We however had to note some limitations of the algorithm for some configurations as well the need to carefully compute the fundamental matrix by among other things selecting "good" matches.

In chapter 3, we re-modeled the already presented image rectification to suit the cubic panorama case. However, we first had to establish the equivalent of the concepts of the fundamental and essential matrices in the case of cubes. This was done successfully by determining the epipolar lines across a panorama after proper computation of the latter entities. Once this step was completed, the final objective was to obtain a rectified pair of cubes; essentially, corresponding faces were to form the usual rectified image pair. Forcing the cubes in a particular configuration by rotating them was sufficient to accomplish our goal. This was convincingly illustrated by $3D$ data.

In chapter 4, pushing the rectification another step further, we aimed at solving the pose recovery problem in the case of cubes. Thus was introduced a two-stage algorithm that proceeded by first estimating the rotations and then the translations of the camera positions much to the delight of "divide and conquer" partisans. Assumptions made

about negligible rotations after the first stage were justified by the result of the second stage and the final reconstruction of the original configuration. Results were obtained for an indoor and outdoor scene respectively for 4 and 6 panoramas to end this chapter and our study.

In each of the chapter we noted a few elements that affected the performance of each of the presented algorithms. For the image triplet rectification, apart from the heavy dependency on the fundamental matrix that was already mentioned a few times, we noted that the capture process was somewhat constrained even before any processing was to be done. As far as the cube rectification is concerned the most important issues were the matches - which were hand selected - across the panoramas and the estimation of the essential matrix. Finally for the pose recovery problem, the use optimization approaches was slowed down by heavy jacobian computations. All this naturally leads us to the possible future work that can be related to our study. Instead of taking the burden of guaranteeing the pursue of all possible aspects as ramifications of this work, we will give possible guidelines and directions for further studies.

First, feature detection was and still is quite an important part of machine vision. Having been confronted to it in the first stage of all presented algorithms, it could be interesting to investigate the possibility of using the strong detectors available nowadays such as SIFT [19] to find well spread - we emphasize the *well spread* character - features across a pair or multiple images. This should also be extended to cubic panoramas. The pose recovery problem for example could be completely automatized if a good tracking algorithm for multiple panoramas is developed.

Moreover, as far image rectification is concerned, algorithms taking into a account really general configurations of camera positions should be explored. This is interesting since the treated pairs of images would not be constrained to contain the epipoles before any valid processing can take place. Besides, the homography solution proposed here is simple and fast and could serve as starting point to more complex, optimization based, procedures such as the ones used in [25].

Finally, for the ones that are performance oriented, one could look at an implementation of our two-stage pose recovery algorithm that would include the sparse matrix method presented in [14] among others. This could speed up the solution and therefore allow a greater number of panoramas to be processed faster.

These suggestions, far from taking away from our study, are supplements that, if completed, could give great range to the preliminary work started here. Our main contribution is essentially the formalization of the cubic panoramas epipolar geometry

and the complete solution to the pose recovery applied to those. Moreover, our study has the particularity of easily being adaptable to any king of panorama be it cylindrical or purely spherical. Notions such as the essential matrix would still be valid therefore so would be the rectification and pose recovery procedures.

This study was conducted as part of the NAVIRE[1] project developed at the University of Ottawa. The objective of the project being the "effective and natural virtual navigation in image-based renditions of real environments", our study mainly answered some issues related to cubic panorama interpolation thanks to the cube rectification and to navigation thanks to the pose recovery solution. Overall, we have successfully met the goals that were set originally and definitely improved our knowledge of spherical panorama represented as cubes.

[1] http://www.site.uottawa.ca/research/viva/projects/ibr/

Appendix A

From 3D coordinates to cube 3D coordinates

The objective here is simple. Given any vector $p = (x, y, z)$ in 3D space, we are looking for its projection $p_c = (x_c, y_c, z_c)$ onto the cube of interest. The projection mentioned here is in respect to the center of the cube which is also considered the center of the world reference frame. This correspondence could be useful in the generation of a new cube, allowing the computation of the new position of a point of the cube after a given rotation is applied to the cube.

The principle is quite simple. The vector p is perceived as a ray's direction and we are looking for the intersection of the latter ray with one of the cube faces : it is a matter of intersecting a line with multiple (portions of) planes and choosing the consistent resulting point.

A.1 Cube Faces : equations

Let us first set the reference frame. The one considered here is such that the front face of the cube is contained in the plane XY in the direction of negative Z. Moreover the Y axis points to the "up" face, and so does the X axis with the right face. For a cube of size s, the different face planes are of the form $ax + by + cz + d = 0$ with $d \neq 0$ and one of (a, b, c) non null for all cases. The equations of these planes are given in the table below :

face	a	b	c	d
up	0	1	0	$-\frac{s}{2}$
left	1	0	0	$\frac{s}{2}$
front	0	0	1	$\frac{s}{2}$
right	1	0	0	$-\frac{s}{2}$
back	0	0	1	$-\frac{s}{2}$
down	0	1	0	$\frac{s}{2}$

A.2 Intersection Line - Plane

Let us consider a vector $p = (x_p, y_p, z_p)$ with its origin at $(0, 0, 0)$. We note p^i one of the coordinates of p with $i \in 1, 2, 3$. We suppose that at least one of the coordinates of p is non null and of index n. If all coordinates are null, the intersection problem is pointless. p intersects a plane Π of equation $ax + by + cz + d = 0$ and of normal $v = (a, b, c)$:

- If p and $v = (a, b, c)$ are orthogonal i.e $p.v = 0$, the intersection is a line and is infinite.
- Else, the ray of equation $\frac{x}{x_p} = \frac{y}{y_p} = \frac{z}{z_p} =$ intersects Π at the point p_c such that:

$$p_c^n = \frac{-d}{v^n + \frac{1}{p^n} \sum_{i \neq n, i=1}^{3} v^i p^i} \quad \text{, with } i, n \in \{1, 2, 3\}$$

  Recall that n stands for the index of a non null coordinates of p. The other coordinates of the intersection point p_c are given by :

$$p_c^i = \frac{p_c^n}{p^n} p^i \quad \text{, with } i \in \{1, 2, 3\} \setminus \{n\}$$

A.3 Intersection Line - Cube

A simple systematic approach allows us to recover the intersection of a line of direction $p = (x_p, y_p, z_p)$ with a cube C at the point p_c. From the 6 possible choices we have by intersecting the line with each of the planes described earlier we only choose the point p_c such that the coordinates that are not exactly equal to $\frac{s}{2}$ in absolute value are in the interval : $\left[-\frac{s}{2}, \frac{s}{2}\right[$. Moreover, the vectors p and p_c must have the same orientation i.e $p.v > 0$. The algorithm can be summarized as follows :

- Loop on each face
- Compute the intersection of the line with the current face
- Check the validity of the intersection : coordinates interval, same orientation
- If found finish loop else continue

Appendix B

$3D$ conversion and plane intersections

In the case of a cube of side L, faces are located at trivial coordinates in respect to the cube reference frame : $x = \pm\frac{L}{2}$ for right and left faces, $y = \pm\frac{L}{2}$ for top and down faces, $z = \pm\frac{L}{2}$ for front and back faces. This allows us to convert easily $2D$ faces coordinates into $3D$ cube reference frame coordinates. In general, if $\tilde{\mathrm{x}} = (x, y, 1)^T$ is the point of concern, an affine transformation is applied to x and y to find 2 of the 3 $3D$ coordinates along the axis x,y or z that form the plane of the face, the third coordinate being a constant as mentioned above.

$$T_{\mathtt{U}} = \begin{pmatrix} 1 & 0 & -\frac{L}{2} \\ 0 & 0 & \frac{L}{2} \\ 0 & -1 & \frac{L}{2} \end{pmatrix}$$

$$T_{\mathtt{L}} = \begin{pmatrix} 0 & 0 & -\frac{L}{2} \\ 0 & -1 & \frac{L}{2} \\ -1 & 0 & \frac{L}{2} \end{pmatrix}$$

$$T_{\mathtt{F}} = \begin{pmatrix} 1 & 0 & -\frac{L}{2} \\ 0 & -1 & \frac{L}{2} \\ 0 & 0 & -\frac{L}{2} \end{pmatrix}$$

$$T_{\mathtt{R}} = \begin{pmatrix} 0 & 0 & \frac{L}{2} \\ 0 & -1 & \frac{L}{2} \\ 1 & 0 & -\frac{L}{2} \end{pmatrix}$$

face	a_i	b_i	c_i
u	a	$-c$	$g_{\mathtt{U}}(b)$
l	$-c$	$-b$	$g_{\mathtt{L}}(-a)$
f	a	$-b$	$g_{\mathtt{F}}(-c)$
r	c	$-b$	$g_{\mathtt{R}}(a)$
b	$-a$	$-b$	$g_{\mathtt{B}}(c)$
d	a	c	$g_{\mathtt{D}}(-b)$

Table B.1: Epipolar lines as intersections of cube and epipolar plane

$$T_{\mathtt{B}} = \begin{pmatrix} -1 & 0 & \frac{L}{2} \\ 0 & -1 & \frac{L}{2} \\ 0 & 0 & \frac{L}{2} \end{pmatrix}$$

$$T_{\mathtt{D}} = \begin{pmatrix} 1 & 0 & -\frac{L}{2} \\ 0 & 0 & -\frac{L}{2} \\ 0 & 1 & -\frac{L}{2} \end{pmatrix}$$

On the other hand, the fact that the faces lie at particular coordinates allows us also to find the intersections of a plane with the cube. The epipolar plane is given by Ep if p is the point of interest. It is a plane that goes through the cube center and therefore is noted (a, b, c) in projective coordinate. We therefore have the results presented in table B.1 for the resulting lines $l_i = (a_i, b_i, c_i)$ on the faces of the cube from the plane (a, b, c). The functions g_i in table B.1 are defined as follows :

$$g_i(m) = \frac{L}{2}(m - (a_i + b_i))$$

Appendix C

Rotation Matrices

The rotations R_i for $i \in \mathtt{U}, \mathtt{L}, \mathtt{F}, \mathtt{R}, \mathtt{B}, \mathtt{D}$ mentioned in the text are obtained by simply observing the frame on figure 3.2(a). To align the frame each face i a trivial rotation needs to be applied. We thus can derive the following expressions for each face :

$$R_{\mathtt{U}} = R_x(\frac{\pi}{2}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}$$

$$R_{\mathtt{L}} = R_y(\frac{\pi}{2}) = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

$$R_{\mathtt{F}} = R_x(0) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R_{\mathtt{R}} = R_y(-\frac{\pi}{2}) = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$R_{\mathtt{B}} = R_y(\pi) = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

$$R_{\mathtt{D}} = R_x(-\frac{\pi}{2}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}$$

Appendix D

Extracting the translation t and rotation R from the essential matrix E

O. Faugeras and R. Hartley respectively discussed this procedure in [8, 7] and [14]. An attempt to summarize the concept is the follows. Based on the equation introduced in *Chapter*2,

$$E = t_{\times}R$$

t and R are the entities sought after knowing E the essential matrix for a pair of cubes C and C'. Recall t is direction in which C is with respect to C' with coordinates expressed in C' and R is the rotation from C to C'. t is the first entity to be computed by solving $E^T t = 0$ with the constraint $||t|| = 1$. The fore-mentioned authors recommend to use the SVD solution approach meaning t is the left singular vector of E^T of least singular value.

Estimating R requires the use of the SVD decomposition of E as $USV^T = E$. [14] goes into more details about this mentioning first the use of two matrices :

$$W = R_z(\frac{pi}{2}) \text{ and } Z = \left[(0,0,1)^T\right]_{\times}$$

Both fore mentioned authors state that there are four solutions possible to the couple (R,t) explicitly given in [14] by:

$$(UWV^T, t)\ ;\ (UWV^T, -t)\ ;\ (UW^TV^T, t)\ ;\ (UW^TV^T, -t)$$

The couple produce the motion parameters from the frame C' to C. However, if the parameters sought after are those describing the motion from C to C', one could use the very simple adjustment :

$$\tilde{R} = R^T \quad \text{and} \quad \tilde{t} = -R^T t \tag{D.1}$$

Where the couple $(\tilde{R}, \tilde{t})$ are the new motion parameters to be used.

In both cases, the choice of the proper solution requires two matches from which the depths have to be accordingly showing that both points are in front of the concerned cameras. In the case of cubic panoramas, the calibration matrix is known and the choice of points in front faces both greatly ease the extraction process. Points can be extracted on other faces but the calibration matrices would then just have to be adjusted by a ± 90 degrees rotation around one of the trivial axis.

Appendix E

Glossary of Terms

DLT Normalized Direct Linear Transform algorithm used in the solution of over-determined linear systems. Useful for the computation of the homography, fundamental matrix and essential matrix from pairs of matches.

"L-triplet" Triplet of images captured following a pattern that resembles the letter "L". One image on top of a regular stereo pair.

Cube Short for cubic panorama. It is one the possible format that is used to represent a spherical image. It is the format of choice in this thesis.

Cube Alignment Procedure by which a set of cubes is transformed into a new set where all cubes have parallel corresponding faces.

Bibliography

[1] H. Abdi. Singular value decomposition (svd) and generalized singular value decomposition (gsvd). In N. J. Salkind, editor, *Encyclopedia of Measurements and Statistics*. Thousand Oaks(CA), 2007.

[2] L. An, Y. Jia, J. Wang, X. Zhang, and M. Li. An efficient rectification method for trinocular stereovision. In *Int. Conf. on Pattern recognition*, pages IV: 56–59, 2004.

[3] M. Baker. Maths - rotations conversions (web).

[4] D. Bradley, A. Brunton, M. Fiala, and G. Roth. Image-based navigation in real environments using panoramas. In *IEEE Int. Workshop on Haptic Audio Visual Environments and their Applications*, pages 103 – 108, October 2005.

[5] M. Brown and D. G. Lowe. Recognising panoramas. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 1218, Washington, DC, USA, 2003. IEEE Computer Society.

[6] Rodrigo Carceroni, Ankita Kumar, and Kostas Daniilidis. Structure from motion with known camera positions. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 477–484, Washington, DC, USA, 2006. IEEE Computer Society.

[7] O. Faugeras. *Three-Dimensional Computer Vision*. The MIT Press, ISBN: 0262061589, first edition, 1993.

[8] O. Faugeras and Q-T. Luong. *The geometry of multiple images*. Cambridge University Press, ISBN: 0262062208, first edition, 2001.

[9] M. Fiala. Immersive panoramic imagery. In *Canadian Conference on Computer and Robot Vision*, pages 386 – 391, 2005.

[10] M. Fiala. Pano-presence for teleoperation. In *Intelligent Robots and Systems (IROS 2005), 2005 IEEE/RSJ International Conference on*, pages 3798–3802, 2005.

[11] M. Fiala and G. Roth. Automatic alignment and graph map building of panoramas. In *IEEE Int. Workshop on Haptic Audio Visual Environments and their Applications*, pages 103 – 108, October 2005.

[12] R. Hartley. In defense of the 8-point algorithm. *IEEE trans. Pattern Analysis and Machine Intelligence*, 19:580–593, 1995.

[13] R. Hartley. Theory and practice of projective rectification. *Int. journal Computer Vision*, 35(2):115–127, 1999.

[14] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[15] J.H Heinbockel. *Introduction to Tensor Calculus and Continuum Mechanics*. Trafford Publishing, ISBN: 1553691334, first edition, 2001.

[16] Florian Kangni and Robert Laganiere. Projective rectification of image triplets from the fundamental matrix. In *IEEE Int. Conf. Acoustics Speech Signal Processing (ICASSP'06)*, 2006.

[17] F.O Kuehnel. On the minization over so(3) manifolds. Technical Report TR.0312, Research Institute for Advanced Computer Science (RIACS), USA, 2003.

[18] C. Loop and Z. Zhang. Computing rectifying homographies for stereo vision. In *in Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 125–131.

[19] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[20] Ameesh Makadia and Kostas Daniilidis. Rotation recovery from spherical images without correspondences. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(7):1170–1175, 2006.

[21] J. Mallon and P.F. Whelan. Projective rectification from the fundamental matrix. *Image and Vision Computing*, 23(7):643–650, July 2005.

[22] J-P. Moreau. Functional approximations in c/c++ : tamoeba.cpp (web).

[23] Srikumar Ramalingam, Suresh K. Lodha, and Peter Sturm. A generic structure-from-motion framework. *Comput. Vis. Image Underst.*, 103(3):218–228, 2006.

[24] Point Grey Research. Ladybug2 Camera.

[25] C. Sun. Uncalibrated three-view image rectification. *Image and Vision Computing*, 21(3):259–269, 2003.

[26] Bill Triggs, Philip McLauchlan, Richard Hartley, and Andrew Fitzgibbon. Bundle adjustment – A modern synthesis. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag, 2000.

[27] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.

[28] Eric Weisstein. Wolfam mathworld (web).

[29] A. Whitehead and G. Roth. The projective vision toolkit. In *Proceedings, Modelling and Simulation.*, pages 204–209, 2000.

[30] H. Zhang, J. Cech, R. Sara, F. Wu, and Z. Hu. A linear trinocular rectification method for accurate stereoscopic matching. In *British Machine Vision Conf.*, pages 281–290, 2003.

[31] Z. Zhang. Determining the epipolar geometry and its uncertainty: A review. Technical Report 2927, Sophia-Antipolis Cedex, France, 1996.

[32] Zhengyou Zhang. Motion and structure from two perspective views: from essential parameters to Euclidean motion through the fundamental matrix. *Journal of the Optical Society of America A*, 14:2938–2950, November 1997.