# THE TRIFOCAL TENSOR AND ITS APPLICATIONS IN AUGMENTED REALITY

Jia Li

A Thesis submitted to the Faculty of Graduate and Postdoctoral Studies in partial fulfillment of the requirements for the degree of Master of Applied Science, Electrical Engineering

September 2005

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario, Canada

# Contents

# List of Figures

# Abstract

The thesis focuses on finding a robust approach to estimating trifocal tensors in video sequences in real-time using a keyframe-based approach. Instead of estimating the tensor for consecutive frames, we compute the tensor associated with each video frame and two reference images. An algebraic minimization method is used for tensor estimation because of its speed and ease of implementation. Additional procedures are explored to compensate its fragility to strong outliers that can not be avoided in practice. The second part of the thesis describes an example application of the proposed framework. By using a method similar to the ARToolkit, our approach enables embedding of virtual objects onto the live video. We demonstrate experimentally that in our AR-system (1) the registration of virtual objects is accomplished without explicitly computing the camera pose; (2) that virtual objects are properly augmented even though they are occluded or partly out of view. Strategies designed for reducing the video 'jitter' or smoothing the object movement are also implemented and described.

# Acknowledgements

Professor Robert Laganière in University of Ottawa and Dr. Roth Gerhard at National Research Council are my co-supervisors. I would like to gratefully thank them for giving me an interesting topic to study. Without their direction, guidance and continuous support this thesis could not be done.

I am grateful to all my friends in the VIVA lab. Xiaoyong Sun, Kehua Jiang and Etienne Vincent are thanked for generous help with advices and assistances.

Finally, special thanks to my parents for their support and encouragement when it was most required.

# Chapter 1

# Introduction

Estimating trifocal geometries of video sequences is an important research topic that has been studied for many applications including include camera pose estimation [36], 3D modelling [4, 15, 46, 44], novel view synthesis [37, 19, 3, 7, 28, 33], augmented reality [46, 45], object-based video compression [38], self-calibration [23] and motion segmentation [39, 14]. Generally it involves feature tracking, matching and tensor estimation. Most existing approaches are implemented in a chaining [36, 46] or hierarchy scheme [29]. RANSAC-based tensor estimation methods are then applied on matched consecutive frames to calculate their trifocal geometry. Experiments show that these approaches work well for post-processing applications that don't require real-time computation. However, even in this case error accumulation and poor estimation of camera geometry in consecutive frames still remain a problem.

The thesis focuses on finding a robust approach to estimating trifocal tensors in video sequences in real-time using a keyframe-based approach. Instead of estimating the tensor for consecutive frames, we compute the tensor associated with each video frame and two reference images. An algebraic minimization method is used for tensor estimation because of its speed and ease of implementation. Additional procedures are explored to compensate its fragility to strong outliers that can not be avoided in practice.

The second part of the thesis describes an example application of the proposed

framework. By using a method similar to the ARToolkit, our approach enables embedding of virtual objects onto the live video. We demonstrate experimentally that in our AR-system (1) the registration of virtual objects is accomplished without explicitly computing the camera pose; (2) that virtual objects are properly augmented even though they are occluded or partly out of view. Strategies designed for reducing the video 'jitter' or smoothing the object movement are also implemented and described. The system has been presented at the ISMAR 2004 conference [16].

## 1.1 Thesis Contributions

This thesis aims to give a perspective view of the trifocal tensor, including its properties and utilization. The main contribution of this thesis is a three-view tracking system to track points over sequences of a moving camera and perform online estimation of the trifocal tensor associated with each frame.

Another important contribution is that we exploit the use of the trifocal tensor for realtime augmented reality in a novel way, which is completely different from existing AR approaches. Experiments show that our AR system works at 10fps.

## 1.2 Thesis Outline

The thesis is organized as follows. Chapter 2 describes the basic theory of the trifocal geometry. It starts by giving notations that will be used in the entire thesis. The trifocal tensor and its important properties are then introduced. Even though attention is given to trilinear transfer, another way of transferring points or lines by using a homography is discussed for the purpose of comparison.

Chapter 3 describes several algorithms to estimate the trifocal tensor from image correspondences; including linear solution, algebraic algorithm and RANSAC methods. Their performances in terms of the transfer error, time efficiency and robustness against mismatches are compared by experiments on real image data.

In chapter 4, a new approach of estimating the trifocal tensor in real-time is described. It uses a three-view framework that is designed for processing of video sequences. Previous methods including using the chained transformation and keyframes are reviewed at the beginning of Chapter 4. After that, the outline of the approach proposed in this thesis is given and its implementation details are described. The scope of this approach and its extension to a camera-matrix framework will be also addressed.

Chapter 5 gives an application of our approach to augmented reality. First, existing augmented reality systems are reviewed and the ARToolkit approach which is used in the new AR system proposed in this thesis is described; After that, implementation details of rendering using the online estimated tensor are provided; In the end, a Kalman filter is used to reduce the jitter of inserted virtual objects and example sequences produced from our AR system are presented.

In the conclusion, Chapter 6, gives a summary of the conclusions and an overview over possible future work.

# Chapter 2

# Basic Theory of The Trifocal Tensor

## 2.1 Trifocal Geometry

Establishing correspondences of image primitives (points or lines) over multiple views is recognized as a fundamental problem in computer vision. Assume there are three images captured by three calibrated cameras with the known extrinsic calibration (camera positions in Euclidean space) and the intrinsic parameters (camera calibration information). Figure 2.1 shows the geometry of three cameras. Their optical centers are denoted by $\mathbf{C}$, $\mathbf{C}'$ and $\mathbf{C}''$. A 3D point $\mathbf{X}$ in Euclidean space has its unique image on the image plane of each camera at where the ray linking the point to the camera optical center intersects the plane. However the opposite is not true since the image point can be the projection of any 3D point on that ray. The nonzero scale factor $\lambda$ in the equation 2.1 explains this ambiguity. The Euclidean camera is defined by a $3 \times 4$ camera matrix $\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$. The $3 \times 3$ camera calibration matrix $\mathbf{K}$ contains the intrinsic parameters of the specific camera. The camera's pose with respect to the world coordinate system is represented by $3 \times 3$ rotation $\mathbf{R}$ and 3D vector translation $\mathbf{t}$. The projection from 3D point $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ to image point $(\mathbf{x}, \mathbf{y})$ is

up to a scale.

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{P} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.1}$$

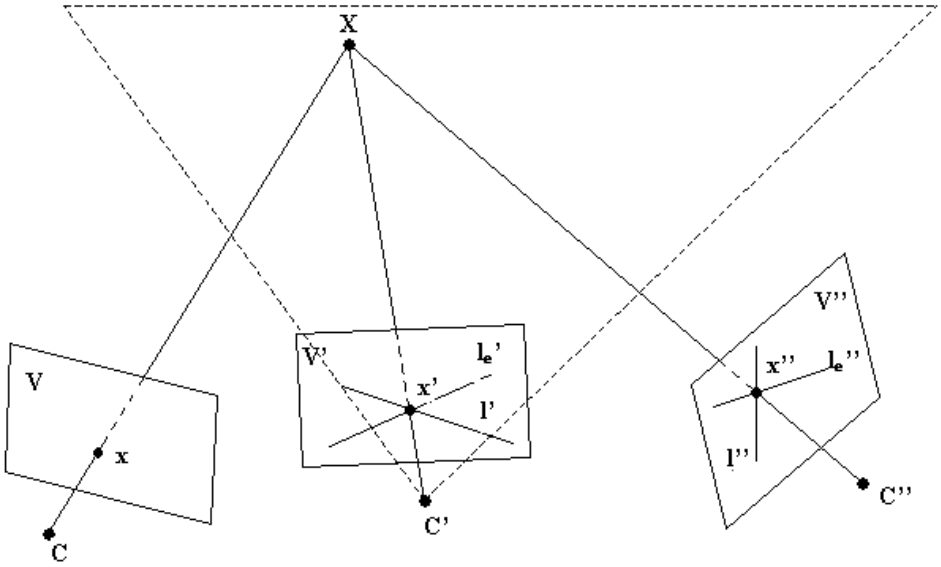



Figure 2.1: Trifocal geometry of three views.

The scale ambiguity can be overcome by having correspondences for a feature in two views. The 3D point $X$ can then be reconstructed by means of triangulation using the point correspondence pair $\mathbf{x}$ and $\mathbf{x}'$ on view $\mathbf{V}$ and $\mathbf{V}'$. Then this point can be re-projected into the third view $\mathbf{V}''$ to find out where the image coordinate of this 3D point should appear on that view. This process $x, x' \rightarrow x''$ is called transfer, and it can be done with at least one calibrated camera and correspondences across two views.

For un-calibrated cameras no transformation between an image point and a ray in Euclidean space is provided because their intrinsic and extrinsic parameters are unknown. In this case the cameras are defined to be projective cameras ($\mathbf{K} = \mathbf{I}$) and any one can be replaced by another. The 3D-to-2D projection $\mathbf{P^P}$ becomes

camera independent. The projective spaces where such projections take place are all equivalent since they can align to each other only by a 3D projective transformation ($4 \times 4$ matrix $\mathbf{H}$)[15].

$$\mathbf{P^p_i} = \mathbf{HP^p_j} \tag{2.2}$$

It is the relative camera position that determines the transformation between two projective spaces. Usually when more than one camera are considered, a normalized camera coordinate system is used and one of cameras is $\mathbf{P} = [\mathbf{I}|\mathbf{0}]$. Consequently the positions of other cameras in this system, and the extrinsic camera parameters are all that is necessary to define the camera projection matrices. Therefore even when the three cameras mentioned above are not calibrated, image correspondences can still be transferred from two views to the third. As will be discussed in the next section, it is possible to define the projective geometry of the three cameras by a trifocal tensor. When the tensor is computed the transfer can be done directly by it without computing an intermediate 3D point. Since this removes the need of camera calibration, the trifocal tensor has been recognized an important tool for processing three views.

It needs to be pointed out that corresponding image points on three views are also related by epipolar geometry. The ray projected from one image point will be viewed as a line in another view (called the epipolar line). The relation between image correspondences of a view pair is described by a $3 \times 3$ fundamental matrix, i.e. $\mathbf{x'^T F_{12} x} = \mathbf{0}$ of view $\mathbf{i}$ and $\mathbf{j}$. The three views can be registered together in a pair-wise manner. Again given a pair of image correspondence $(\mathbf{x}, \mathbf{x}')$, intersecting their epipolar lines on the third view should show the image position of the 3D point. However, if the point $\mathbf{X}$ is in the trifocal plane defined by the optical centers $\mathbf{C}$, $\mathbf{C}'$ and $\mathbf{C}''$ or if the centers are aligned, it is impossible to determine if three image points belong to a single 3D point by epipolar geometry. Such ambiguity can be avoided by using a trifocal tensor since it provides a more accurate and stable description of three views' geometry than the fundamental matrices between each pair of views.

## 2.2 The Trifocal Tensor

The trifocal tensor is expressed by a set of three $3 \times 3$ matrices, $[\mathbf{T_i}]$, $\mathbf{i} = \mathbf{1}, \mathbf{2}, \mathbf{3}$. It describes the projective geometric relations of image triplets taken from three cameras [15]. Considering a view triplet, if the camera matrix of the first view is in canonical form, $\mathbf{P_1} = [\mathbf{I}|\mathbf{0}]$, and the camera matrices of the other two views are expressed as $\mathbf{P_2} = [\mathbf{A}|\mathbf{e}']$, $\mathbf{P_3} = [\mathbf{B}|\mathbf{e}'']$, where $\mathbf{A}$ and $\mathbf{B}$ are $3 \times 3$ matrices, and, $\mathbf{e}'$ and $\mathbf{e}''$ are the epipoles corresponding to the image of the center of the first camera on the image plane of the second and third cameras respectively, then the $3 \times 3 \times 3$ trifocal tensor can be denoted as $\mathbf{T} = [\mathbf{T_1}, \mathbf{T_2}, \mathbf{T_3}]^T$, with:

$$\mathbf{T_i} = \mathbf{a_i}\mathbf{e}''^T - \mathbf{e}'\mathbf{b_i}^T \tag{2.3}$$

The above equation presents a straightforward way to construct the trifocal tensor from the camera matrices. However, the trifocal tensor can also be estimated from image correspondences alone, without knowledge of the camera parameters. This means that no explicit 3D information is required in order to obtain the tensor relation. Several methods have been proposed to compute the tensor from a set of point or line matches. They will be discussed in details in Chapter 3.

Once a tensor is computed the epipoles of the first camera within the second and third view, $\mathbf{e}'$ and $\mathbf{e}''$, are those that satisfy the following equations:

$$\mathbf{e}'^{\mathbf{T}}[\mathbf{u_1}, \mathbf{u_2}, \mathbf{u_3}] = 0 \tag{2.4}$$

$$\mathbf{e}''^{\mathbf{T}}[\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}] = 0 \tag{2.5}$$

where $\mathbf{u_i}$ and $\mathbf{v_i}$ be the left and right null-vectors respectively of $\mathbf{T_i}$, i.e.

$$\mathbf{u_i}^{\mathbf{T}}\mathbf{T_i} = \mathbf{0}^{\mathbf{T}}, \quad \mathbf{T_i}\mathbf{v_i} = \mathbf{0} \tag{2.6}$$

These equations are important because any effort to recover the complete projective geometry from the tensor needs the prior knowledge of the epipoles.

## 2.3 From tensor to projective geometry

When the trifocal tensor $[\mathbf{T_i}]$ is known, the fundamental matrices $\mathbf{F_{21}}$, $\mathbf{F_{31}}$ relating the first camera with the other two cameras are

$$\mathbf{F_{21}} = [\mathbf{e}']_\times[\mathbf{T_1}, \mathbf{T_2}, \mathbf{T_3}]\mathbf{e}'' \tag{2.7}$$

$$\mathbf{F_{31}} = [\mathbf{e}'']_\times[\mathbf{T_1}^\mathbf{T}, \mathbf{T_2}^\mathbf{T}, \mathbf{T_3}^\mathbf{T}]\mathbf{e}' \tag{2.8}$$

As mentioned above, $[\mathbf{T_i}]$ is defined within a normalized canonical system which is aligned with the first camera. So that, the camera matrices ($\mathbf{P_1}$, $\mathbf{P_2}$, $\mathbf{P_3}$) corresponding to the computed tensor may be chosen as

$$\mathbf{P_1} = [\mathbf{I}|\mathbf{0}] \tag{2.9}$$

$$\mathbf{P_2} = [[\mathbf{T_1}, \mathbf{T_2}, \mathbf{T_3}]\mathbf{e}''|\mathbf{e}'] \tag{2.10}$$

$$\mathbf{P_3} = [(\mathbf{e}''\mathbf{e}''^T - I)[\mathbf{T_1}^T, \mathbf{T_2}^T, \mathbf{T_3}^T]\mathbf{e}'|\mathbf{e}''] \tag{2.11}$$

This triplet of camera matrices expresses the projective transformations between three views. In order to upgrade to metric or Euclidean reconstruction, self-calibration is needed to obtain the intrinsic parameters of the camera.

Though the trifocal tensor is usually estimated from point or line-matches as will be discussed in the next chapter, there are two alternatives to estimate the tensor. (1) Assuming that the fundamental matrices of three views, $\mathbf{F_{12}}$ and $\mathbf{F_{23}}$, are known, the tensor can be computed from the entries of the projection matrices obtained from the fundamental matrices; (2) Given a tensor of three views $\mathbf{V_{123}}$, a new tensor related to the first two views and an additional view $\mathbf{V_{124}}$ can be derived by [37]

$$\mathbf{G_i}^{\mathbf{jk}} = \mathbf{d_1}^\mathbf{k} \times \mathbf{T_i}^{\mathbf{j1}} + \mathbf{t}^\mathbf{k} \times \mathbf{a_i}^\mathbf{j} \tag{2.12}$$

$\mathbf{G_i}^{\mathbf{jk}}$ is used to represent the new tensor. Motion from the last view $\mathbf{V_3}$ to the new view $\mathbf{V_4}$ is described by a $3 \times 3$ homography matrix $\mathbf{D}$ and a translation vector $\mathbf{t}$. Here $\mathbf{A}$ is the upper $3 \times 3$ sub-matrix of the projection matrix of the first view. This equation is often used in novel view synthesis methods (see [7, 37] for example).

## 2.4 Trifocal transfer

The most attractive characteristic of the trifocal tensor is the transfer of points and lines. A point/line in one image can be computed from its correspondence in the other two images. If $(\mathbf{l}, \mathbf{l}', \mathbf{l}'')$ is a set of corresponding lines and $(\mathbf{x}, \mathbf{x}', \mathbf{x}'')$ is a set of corresponding points in three images, the transfer operations can be represented by following equations:

$$\mathbf{l^T} = \mathbf{l'^T}[\mathbf{T_1}, \mathbf{T_2}, \mathbf{T_3}]\mathbf{l''} \tag{2.13}$$

$$[\mathbf{x'}]_\times(\sum_{\mathbf{i}} \mathbf{x^i T_i})[\mathbf{x''}]_\times = \mathbf{0} \tag{2.14}$$

where the notation $[\mathbf{x}]_\mathbf{x}$ indicates a skew-symmetric matrix of the vector $\mathbf{x} = (\mathbf{x^1}, \mathbf{x^2}, \mathbf{x^3})$ defined as

$$[\mathbf{x}]_\times = \begin{bmatrix} 0 & -x^3 & x^2 \\ x^3 & 0 & -x^1 \\ -x^2 & x^1 & 0 \end{bmatrix} \tag{2.15}$$

### 2.4.1 Transfer by solving the trilinear equations

Let $\mathbf{T_i}^{\mathbf{ij}}$ denotes the $(\mathbf{i}, \mathbf{j})$ entry of submatrix $T_i$ of tensor, the point-point-point trilinear equation (2.14) can be be expanded as:

$$\mathbf{x^i}[-\mathbf{x'^3}(\mathbf{x''^3}\mathbf{T_i}^{\mathbf{21}} - \mathbf{x''^1}\mathbf{T_i}^{\mathbf{23}}) + \mathbf{x'^2}(\mathbf{x''^3}\mathbf{T_i}^{\mathbf{31}} - \mathbf{x''^1}\mathbf{T_i}^{\mathbf{33}})] = \mathbf{0_{12}} \tag{2.16}$$

This leads to a set of nine equations, of which only four equations are linearly independent. Since the z homogenous coordinates, $\mathbf{x^3}$, $\mathbf{x'^3}$ and $\mathbf{x''^3}$ of image points are always one, the four equations are:

$$\begin{aligned}
&\mathbf{x^i}\mathbf{T_i}^{\mathbf{11}} - \mathbf{x^i}\mathbf{x''^1}\mathbf{T_i}^{\mathbf{13}} - \mathbf{x^i}\mathbf{x'^1}\mathbf{T_i}^{\mathbf{31}} + \mathbf{x^i}\mathbf{x'^1}\mathbf{x''^1}\mathbf{T_i}^{\mathbf{33}} = \mathbf{0} \\
&\mathbf{x^i}\mathbf{T_i}^{\mathbf{21}} - \mathbf{x^i}\mathbf{x''^1}\mathbf{T_i}^{\mathbf{23}} - \mathbf{x^i}\mathbf{x'^2}\mathbf{T_i}^{\mathbf{31}} + \mathbf{x^i}\mathbf{x'^2}\mathbf{x''^1}\mathbf{T_i}^{\mathbf{33}} = \mathbf{0} \\
&\mathbf{x^i}\mathbf{T_i}^{\mathbf{12}} - \mathbf{x^i}\mathbf{x''^2}\mathbf{T_i}^{\mathbf{13}} - \mathbf{x^i}\mathbf{x'^1}\mathbf{T_i}^{\mathbf{32}} + \mathbf{x^i}\mathbf{x'^1}\mathbf{x''^2}\mathbf{T_i}^{\mathbf{33}} = \mathbf{0} \\
&\mathbf{x^i}\mathbf{T_i}^{\mathbf{22}} - \mathbf{x^i}\mathbf{x''^2}\mathbf{T_i}^{\mathbf{23}} - \mathbf{x^i}\mathbf{x'^2}\mathbf{T_i}^{\mathbf{32}} + \mathbf{x^i}\mathbf{x'^2}\mathbf{x''^2}\mathbf{T_i}^{\mathbf{33}} = \mathbf{0}
\end{aligned} \tag{2.17}$$

They can be written in a matrix form as $\mathbf{mx^i} = \mathbf{0}$ where $\mathbf{m}$ is a 4-element vector. The vector $\mathbf{m}$ is the most compact representation of trilinear relations because only

12 instead of all 27 tensor entries appear in each of the four equations. When $\mathbf{i}$ ranges from 1 to 3, the above linear equations contribute a linear system of the homogenous coordinates of the image point $\mathbf{x}$.

$$\mathbf{M} \times \mathbf{x} = \mathbf{0} \tag{2.18}$$

where $\mathbf{M}$ stands for a $4 \times 3$ matrix with entries in terms of the tensor $\mathbf{T}$ and a pair of image point ($\mathbf{x}'$, $\mathbf{x}''$) between two views. By solving this linear system, one may find the corresponding point in the first view as $(\frac{\mathbf{x^1}}{\mathbf{x^3}}, \frac{\mathbf{x^2}}{\mathbf{x^3}})$.

It is possible to write closed-form expressions to perform a mapping from $\mathbf{V}$ to $\mathbf{V}'$ or $\mathbf{V}''$ using the trilinear constraints. For example, by taking the first and the third equation in 2.17, we get:

$$\begin{aligned} \mathbf{x''^1} &= \frac{\mathbf{T_1^{11}x^1 + T_2^{11}x^2 + T_3^{11} - x'^1(T_1^{31}x^1 + T_2^{31}x^2 + T_3^{31})}}{\mathbf{T_1^{13}x^1 + T_2^{13}x^2 + T_3^{13} - x'^1(T_1^{33}x^1 + T_2^{33}x^2 + T_3^{33})}} \\ \mathbf{x''^2} &= \frac{\mathbf{T_1^{12}x^1 + T_2^{12}x^2 + T_3^{12} - x'^1(T_1^{32}x^1 + T_2^{32}x^2 + T_3^{32})}}{\mathbf{T_1^{13}x^1 + T_2^{13}x^2 + T_3^{13} - x'^1(T_1^{33}x^1 + T_2^{33}x^2 + T_3^{33})}} \end{aligned} \tag{2.19}$$

It is important to observe that the equations 2.19 do not contain the $\mathbf{y}$ coordinate of the point $\mathbf{x}'$ so that $\mathbf{x''^1}$ and $\mathbf{x''^2}$ are more heavily influenced by the motion along the axis $x$ from view $\mathbf{V}$ to view $\mathbf{V}'$. To get a better result for generic camera motion, it is desirable to solve for the two unknowns $\mathbf{x''^1}$ and $\mathbf{x''^2}$ from all independent trilinear equations in 2.17 simultaneously.

The linear system for the homogeneous coordinates of point $\mathbf{x}'$ is:

$$\begin{bmatrix} {a_{21}}^1 & 0 & {a_{21}}^3 \\ {a_{22}}^1 & 0 & {a_{22}}^3 \\ 0 & {a_{12}}^2 & {a_{12}}^3 \\ 0 & {a_{11}}^2 & {a_{11}}^3 \end{bmatrix} \mathbf{x}' = \mathbf{0} \tag{2.20}$$

$$
\begin{aligned}
\mathbf{a}_{21}^{1} &= -\mathbf{x}^{1}\mathbf{x}'^{2}\mathbf{T}_{1}{}^{33} - \mathbf{x}^{2}\mathbf{x}'^{2}\mathbf{T}_{2}{}^{33} - \mathbf{x}'^{2}\mathbf{T}_{3}{}^{33} + \mathbf{x}^{1}\mathbf{T}_{1}{}^{32} + \mathbf{x}^{2}\mathbf{T}_{2}{}^{32} + \mathbf{T}_{3}{}^{32} \\
\mathbf{a}_{21}^{3} &= \mathbf{x}^{1}\mathbf{x}'^{2}\mathbf{T}_{1}{}^{13} - \mathbf{x}^{2}\mathbf{x}'^{2}\mathbf{T}_{2}{}^{13} - \mathbf{x}'^{2}\mathbf{T}_{3}{}^{13} - \mathbf{x}^{1}\mathbf{T}_{1}{}^{12} - \mathbf{x}^{2}\mathbf{T}_{2}{}^{12} - \mathbf{T}_{3}{}^{12} \\
\mathbf{a}_{22}^{1} &= \mathbf{x}^{1}\mathbf{x}'^{1}\mathbf{T}_{1}{}^{33} - \mathbf{x}^{2}\mathbf{x}'^{1}\mathbf{T}_{2}{}^{33} - \mathbf{x}'^{1}\mathbf{T}_{3}{}^{33} - \mathbf{x}^{1}\mathbf{T}_{1}{}^{31} - \mathbf{x}^{2}\mathbf{T}_{2}{}^{31} - \mathbf{T}_{3}{}^{31} \\
\mathbf{a}_{22}^{3} &= -\mathbf{x}^{1}\mathbf{x}'^{1}\mathbf{T}_{1}{}^{13} - \mathbf{x}^{2}\mathbf{x}'^{1}\mathbf{T}_{2}{}^{13} - \mathbf{x}'^{1}\mathbf{T}_{3}{}^{13} + \mathbf{x}^{1}\mathbf{T}_{1}{}^{11} + \mathbf{x}^{2}\mathbf{T}_{2}{}^{11} - \mathbf{T}_{3}{}^{11} \\
\mathbf{a}_{12}^{2} &= -\mathbf{x}^{1}\mathbf{x}'^{1}\mathbf{T}_{1}{}^{33} - \mathbf{x}^{2}\mathbf{x}'^{1}\mathbf{T}_{2}{}^{33} - \mathbf{x}'^{1}\mathbf{T}_{3}{}^{33} + \mathbf{x}^{1}\mathbf{T}_{1}{}^{31} + \mathbf{x}^{2}\mathbf{T}_{2}{}^{31} + \mathbf{T}_{3}{}^{31} \\
\mathbf{a}_{12}^{3} &= \mathbf{x}^{1}\mathbf{x}'^{1}\mathbf{T}_{1}{}^{23} - \mathbf{x}^{2}\mathbf{x}'^{1}\mathbf{T}_{2}{}^{23} - \mathbf{x}'^{1}\mathbf{T}_{3}{}^{23} - \mathbf{x}^{1}\mathbf{T}_{1}{}^{21} - \mathbf{x}^{2}\mathbf{T}_{2}{}^{21} - \mathbf{T}_{3}{}^{21} \\
\mathbf{a}_{11}^{2} &= \mathbf{x}^{1}\mathbf{x}'^{2}\mathbf{T}_{1}{}^{33} + \mathbf{x}^{2}\mathbf{x}'^{2}\mathbf{T}_{2}{}^{33} + \mathbf{x}'^{2}\mathbf{T}_{3}{}^{33} - \mathbf{x}^{1}\mathbf{T}_{1}{}^{32} - \mathbf{x}^{2}\mathbf{T}_{2}{}^{32} - \mathbf{T}_{3}{}^{32} \\
\mathbf{a}_{12}^{3} &= -\mathbf{x}^{1}\mathbf{x}'^{2}\mathbf{T}_{1}{}^{23} - \mathbf{x}^{2}\mathbf{x}'^{2}\mathbf{T}_{2}{}^{23} - \mathbf{x}'^{2}\mathbf{T}_{3}{}^{23} + \mathbf{x}^{1}\mathbf{T}_{1}{}^{22} + \mathbf{x}^{2}\mathbf{T}_{2}{}^{22} + \mathbf{T}_{3}{}^{22}
\end{aligned}
\tag{2.21}
$$

The linear system for the homogeneous coordinates of point $\mathbf{x}''$ is given as:

$$
\begin{bmatrix}
0 & a_{21}{}^{2} & a_{21}{}^{3} \\
a_{22}{}^{1} & 0 & a_{22}{}^{3} \\
a_{12}{}^{1} & 0 & a_{12}{}^{3} \\
0 & a_{11}{}^{2} & a_{11}{}^{3}
\end{bmatrix} \times \mathbf{x}'' = \mathbf{0}
\tag{2.22}
$$

where

$$
\begin{aligned}
\mathbf{a}_{21}^{2} &= -\mathbf{x}^{1}\mathbf{x}'^{1}\mathbf{T}_{1}{}^{33} - \mathbf{x}^{2}\mathbf{x}'^{1}\mathbf{T}_{2}{}^{33} - \mathbf{x}'^{1}\mathbf{T}_{3}{}^{33} + \mathbf{x}^{1}\mathbf{T}_{1}{}^{13} + \mathbf{x}^{2}\mathbf{T}_{2}{}^{13} + \mathbf{T}_{3}{}^{13} \\
\mathbf{a}_{21}^{3} &= \mathbf{x}^{1}\mathbf{x}'^{1}\mathbf{T}_{1}{}^{32} + \mathbf{x}^{2}\mathbf{x}'^{1}\mathbf{T}_{2}{}^{32} + \mathbf{x}'^{1}\mathbf{T}_{3}{}^{32} - \mathbf{x}^{1}\mathbf{T}_{1}{}^{12} - \mathbf{x}^{2}\mathbf{T}_{2}{}^{12} - \mathbf{T}_{3}{}^{12} \\
\mathbf{a}_{22}^{1} &= \mathbf{x}^{1}\mathbf{x}'^{1}\mathbf{T}_{1}{}^{33} + \mathbf{x}^{2}\mathbf{x}'^{1}\mathbf{T}_{2}{}^{33} + \mathbf{x}'^{1}\mathbf{T}_{3}{}^{33} - \mathbf{x}^{1}\mathbf{T}_{1}{}^{13} - \mathbf{x}^{2}\mathbf{T}_{2}{}^{13} - \mathbf{T}_{3}{}^{13} \\
\mathbf{a}_{22}^{3} &= -\mathbf{x}^{1}\mathbf{x}'^{1}\mathbf{T}_{1}{}^{31} - \mathbf{x}^{2}\mathbf{x}'^{1}\mathbf{T}_{2}{}^{31} - \mathbf{x}'^{1}\mathbf{T}_{3}{}^{31} + \mathbf{x}^{1}\mathbf{T}_{1}{}^{11} + \mathbf{x}^{2}\mathbf{T}_{2}{}^{11} + \mathbf{T}_{3}{}^{11} \\
\mathbf{a}_{12}^{1} &= -\mathbf{x}^{1}\mathbf{x}'^{2}\mathbf{T}_{1}{}^{33} - \mathbf{x}^{2}\mathbf{x}'^{2}\mathbf{T}_{2}{}^{33} - \mathbf{x}'^{2}\mathbf{T}_{3}{}^{33} + \mathbf{x}^{1}\mathbf{T}_{1}{}^{23} + \mathbf{x}^{2}\mathbf{T}_{2}{}^{23} + \mathbf{T}_{3}{}^{23} \\
\mathbf{a}_{12}^{3} &= \mathbf{x}^{1}\mathbf{x}'^{2}\mathbf{T}_{1}{}^{31} + \mathbf{x}^{2}\mathbf{x}'^{2}\mathbf{T}_{2}{}^{31} + \mathbf{x}'^{2}\mathbf{T}_{3}{}^{31} - \mathbf{x}^{1}\mathbf{T}_{1}{}^{21} - \mathbf{x}^{2}\mathbf{T}_{2}{}^{21} - \mathbf{T}_{3}{}^{21} \\
\mathbf{a}_{11}^{2} &= \mathbf{x}^{1}\mathbf{x}'^{2}\mathbf{T}_{1}{}^{33} + \mathbf{x}^{2}\mathbf{x}'^{2}\mathbf{T}_{2}{}^{33} + \mathbf{x}'^{2}\mathbf{T}_{3}{}^{33} - \mathbf{x}^{1}\mathbf{T}_{1}{}^{23} - \mathbf{x}^{2}\mathbf{T}_{2}{}^{23} - \mathbf{T}_{3}{}^{23} \\
\mathbf{a}_{11}^{3} &= -\mathbf{x}^{1}\mathbf{x}'^{2}\mathbf{T}_{1}{}^{32} - \mathbf{x}^{2}\mathbf{x}'^{2}\mathbf{T}_{2}{}^{32} - \mathbf{x}'^{2}\mathbf{T}_{3}{}^{32} + \mathbf{x}^{1}\mathbf{T}_{1}{}^{22} + \mathbf{x}^{2}\mathbf{T}_{2}{}^{22} + \mathbf{T}_{3}{}^{22}
\end{aligned}
\tag{2.23}
$$

### 2.4.2 Transfer by a homography

Beside solving the trilinear equations, there is another method to transfer a pair of point to a third view [15]. It is based on the tensor's constraint for point-line-point correspondences.

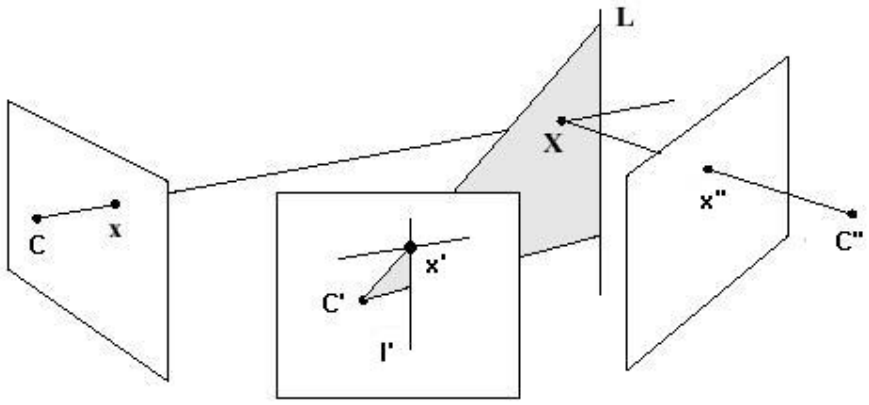



Figure 2.2: The point transfer

The key idea, as shown in Figure 2.2, is that given a pair of matches $(\mathbf{x}, \mathbf{x}')$ between views $(\mathbf{V}, \mathbf{V}')$, the corresponding point on the third view is

$$\mathbf{x''^k} = \mathbf{x^i l_j' T_i^{jk}} \tag{2.24}$$

where $\mathbf{l}'$ indicates a line passing through the point $\mathbf{x}'$ in the image plane of the second view. Together with the camera center $\mathbf{C}'$, it defines a plane which interacts with the ray of $\mathbf{x}$ in the 3D point $\mathbf{X}$. Then the point $\mathbf{x}''$ is given by the projection of $\mathbf{X}$ on the third view. The above equation represents a homography between views $\mathbf{V}$ and $\mathbf{V}''$ induced by a line in the second view $\mathbf{V}'$. we can write $\mathbf{x}'' = \mathbf{H_{13}x}$. The homography is given by

$$\mathbf{H_{13}}(\mathbf{l}') = \mathbf{H_i^k} = \mathbf{l_j' T_i^{jk}} = [\mathbf{T_1}^\mathbf{T}, \mathbf{T_2}^\mathbf{T}, \mathbf{T_3}^\mathbf{T}]\mathbf{l}' \tag{2.25}$$

For the sake of stability, the line $\mathbf{l}'$ is chosen to be perpendicular to the epipole line of $\mathbf{x}'$. The method is summarized as following steps:

1. Given a correspondence $\mathbf{x} \leftrightarrow \mathbf{x\prime}$, correct the points using the fundamental matrix between the first and second view $\mathbf{F_{21}}$ to get an exact match $(\hat{\mathbf{x}} \leftrightarrow \hat{\mathbf{x}}')$.

2. The epipole line of $\mathbf{x}'$ is $\mathbf{l_e}' = \mathbf{F_{21}}\hat{\mathbf{x}}$. Assume coordinates of $\mathbf{l_e}'$ are $(\mathbf{l_e^1}, \mathbf{l_e^2}, \mathbf{l_e^3})$, compute the perpendicular line $\mathbf{l}' = (\mathbf{l_e^2}, -\mathbf{l_e^1}, -\hat{\mathbf{x}}^\mathbf{1}\mathbf{l_e^2} + \hat{\mathbf{x}}^\mathbf{2}\mathbf{l_e^1})$.

3. Then calculate the corresponding point $\mathbf{x''}$ with the homography $\mathbf{H_{13}}$.

The first step of correcting point correspondences is important especially when the observed point sets are not free of noise. The fundamental matrices w.r.t the first view, $\mathbf{F_{21}}$ and $\mathbf{F_{31}}$ would be calculated from the tensor as defined in equation 2.8 if they are not provided beforehand. To compute the fundamental matrix $\mathbf{F_{23}}$ required in transfer to view $\mathbf{V}$, the conventional 8-point method is sufficient.

The formulas for point transfer into each view are:

$$\begin{aligned}
\mathbf{l_e}' &= \mathbf{F_{23}}\hat{\mathbf{x}}'', \quad \mathbf{l}' = (\mathbf{l_e}^{\mathbf{2}}, -\mathbf{l_e}^{\mathbf{1}}, -\hat{\mathbf{x}''}^{\mathbf{1}}\mathbf{l_e}^{\mathbf{2}} + \hat{\mathbf{x}''}^{\mathbf{2}}\mathbf{l_e}^{\mathbf{1}}), \quad \mathbf{H_{31}} = \mathbf{l'^j T_i^{jk}}, \quad \mathbf{x} = \mathbf{H_{31}}\hat{\mathbf{x}}'' \\
\mathbf{l_e}'' &= \mathbf{F_{31}}\hat{\mathbf{x}}, \quad \mathbf{l}'' = (\mathbf{l_e}^{\mathbf{2}}, -\mathbf{l_e}^{\mathbf{1}}, -\hat{\mathbf{x}}^{\mathbf{1}}\mathbf{l_e}^{\mathbf{2}} + \hat{\mathbf{x}}^{\mathbf{2}}\mathbf{l_e}^{\mathbf{1}}), \quad \mathbf{H_{12}} = \mathbf{l''^k T_i^{jk}}, \quad \mathbf{x}' = \mathbf{H_{12}}\hat{\mathbf{x}} \\
\mathbf{l_e}' &= \mathbf{F_{21}}\hat{\mathbf{x}}, \quad \mathbf{l}' = (\mathbf{l_e}^{\mathbf{2}}, -\mathbf{l_e}^{\mathbf{1}}, -\hat{\mathbf{x}}^{\mathbf{1}}\mathbf{l_e}^{\mathbf{2}} + \hat{\mathbf{x}}^{\mathbf{2}}\mathbf{l_e}^{\mathbf{1}}), \quad \mathbf{H_{13}} = \mathbf{l'^j T_i^{jk}}, \quad \mathbf{x}'' = \mathbf{H_{13}}\hat{\mathbf{x}}
\end{aligned} \tag{2.26}$$

The advantage of this method is that image points can be transferred into view $\mathbf{V}'$ and $\mathbf{V}''$ without solving an over-determined linear systems. But on the negative side, it requires good estimations of the fundamental matrices.

# Chapter 3

# Overview of Estimation of The Trifocal Tensor

Since the trifocal tensor is uniquely defined by the projection matrices of the cameras, it can be computed directly from the knowledge of the relative motions between views and the internal camera parameters. However, this method can only be applied on calibrated views. For uncalibrated views, one can compute the tensor from image correspondences alone. The image correspondences can be points or lines or a mix of both; Table 3.1 shows the feature combinations and the number of equations that are necessary to compute the trilinear tensor.

| Image Features | equations |
|---|---|
| 3 points | 4 |
| 2 points, 1 line | 2 |
| 1 point, 2 lines | 1 |
| 3 lines | 2 |

Table 3.1: Trilinear equations provided from image correspondences

Computation of tensor using correspondences of points or lines has been studied extensively. This thesis is concerned mainly with points in images. The existing methods could be roughly divided into two classes. The first class, which are called the over-parameterized approaches includes a linear least-square solution, algebraic

minimization and geometric distance minimization method. With these methods, the tensor is parameterized in ways that ignore the fact that it has only 18 independent degrees of freedom. The main advantage of these approaches is that they are easily implemented.

The second class is the one that uses random sampling in the estimation of the tensor. Tensors are computed from points that are randomly selected from a set of available correspondences. The obtained tensors are scored against the whole point data so that the desired tensor is the one that has the most support data. The use of *RANSAC* (RANdom SAmple Consensus) enable the tensor estimation to be robust to false matches ("outliers") that usually cause the aforementioned methods prone to fail.

In the thesis work, the linear-square method, the algebraic minimization method and the RANSAC methods are implemented and tested on several sets of synthetic and real data. The computed tensors are evaluated in terms of residual error between predicated point positions obtained through tensor transfer and their actual positions. Comparison of their accuracy and computational efficiency gives a clear picture of the conditions suitable for each method.

## 3.1 The linear algorithm

According to equation 2.18, each point-point-point correspondence provides four linearly independent equations. It follows that seven corresponding points across the three views uniquely determine the 27 entries of the tensor matrix. Given $n$ point correspondences, let $\mathbf{A}$ denote a matrix of size $4n \times 27$ and $\mathbf{t}$ a vector containing all entries of the tensor, we can write

$$\mathbf{At} = \mathbf{0} \tag{3.1}$$

Then the tensor can be obtained by the least square or the SVD solution to this linear system.

Hartley has shown that this kind of algorithm does not do well if all points are of the form $(x_1, x_2, 1)$ in homogeneous coordinates with $x_1$ and $x_2$ very much larger than 1 [15]. Therefore, it is necessary to normalize the points of each image separately before computing the tensor. The points are translated in each image so that the centroid of all measured points is at the origin of the image coordinates, and then scaled so that the average distance of a point from the origin is $\sqrt{2}$ units. This way the average point will be something like $(1, 1, 1)$ in homogeneous coordinates, and each of the homogeneous coordinates will be approximately of equal weight. This transformation improves the condition of the matrix of equations, and leads to a much better solution.

As a consequence, it is necessary to de-normalize the computed tensor in order to work with original image coordinates. The overall process is as follows:

1. Normalize the set of point triplets by performing transformations $\mathbf{H}$, $\mathbf{H}'$ and $\mathbf{H}''$. Here $\hat{\mathbf{x}} = \mathbf{H}\mathbf{x}$, $\hat{\mathbf{x}}' = \mathbf{H}'\mathbf{x}'$, $\hat{\mathbf{x}}'' = \mathbf{H}''\mathbf{x}''$.

2. Compute the tensor linearly by solving a set of equations of the form $\mathbf{A}\mathbf{t} = \mathbf{0}$, where $\mathbf{A}$ expresses the equation (2.14) and $\mathbf{t}$ is the vector of entries of tensor.

3. De-normalize the tensor by $\mathbf{T_i} = \mathbf{H'^{-1}} \sum_{\mathbf{j}} (\mathbf{H^T}(\mathbf{i}, \mathbf{j})\mathbf{T_j})\mathbf{H''^{-T}}$

The linear solution is the easiest method, but also the most unreliable because of two reasons: (1) the tensor is parameterized by all its entries and this does not take into account the tensor geometrical constraints i.e the equation 2.3. Therefore there is always a risk of obtaining an invalid tensor; (2) Using all available point correspondences causes the tensor to be significantly affected by the presence of strong outliers.

## 3.2 The algebraic minimization algorithm

The principle of algebraic minimization is to use the linear solution as an initial estimate and re-parameterize it by the 24 entries of the projection matrices $\mathbf{P}'$ and

$\mathbf{P}''$. The desired tensor is then found by minimizing the residual error. The standard algorithm of algebraic minimization is briefly given here [15].

1. From the set of point triplets, compute the tensor linearly by solving a set of equations of the form $\mathbf{At} = \mathbf{0}$, where $\mathbf{A}$ expresses the equation (3.1) and $\mathbf{t}$ is the vector of entries of tensor.

2. Find the two epipoles $\mathbf{e}'$ and $\mathbf{e}''$ from the tensor.

3. According to Equation (2.3), construct the $28 \times 12$ matrix $\mathbf{E}$ such that $\mathbf{t} = \mathbf{Ea}$, where $\mathbf{a}$ is the vector representing entries of $\mathbf{a_i}$ and $\mathbf{b_i}$.

4. Compute the tensor by minimizing the algebraic error $\|\mathbf{AEa}\|$ subject to $\|\mathbf{Ea}\| = \mathbf{1}$.

Again, all points should be normalized before calculating the initial estimation of the tensor, and de-normalization should be applied on the resulting tensor. The algorithm is executed in an iterative manner by repeating the last two steps with varying $\mathbf{e}'$ and $\mathbf{e}''$ from the currently computed tensor. The Levenberg-Marquardt algorithm, a widely-used algorithm for solving the nonlinear optimization problem, is used to find the optimal $\mathbf{e}'$ and $\mathbf{e}''$. Since the epipoles have 6 entries, the problem is of modest size.

The above algorithm leads to a geometrically valid tensor because its entries satisfies the equation 2.3. However, the main weakness of the algebraic minimization method is that all point correspondences are involved in the tensor computation in the same way as for the linear method, which consequently makes it to outliers (invalid correspondences) that are unavoidable in practice.

## 3.3 The 6-point algorithm

Quan proposed a method to compute the trifocal tensor from 6 point correspondences in [21]. A set of 6 point non-coplanar correspondences from three uncalibrated images can lead to at most 3 solutions of the projection matrices of the three camera. From

the best solution the desired trifocal tensor can be produced. In this thesis work, Torr's modified 6-point algorithm [40] has been implemented, which is summarized as follows:

1. Given 6 point correspondences, in each image transform four points to four basis points in a canonical frame.

$$\begin{aligned} \lambda_1{}^{\mathbf{i}}\mathbf{H}^{\mathbf{i}}\mathbf{x_1}{}^{\mathbf{i}} = (\mathbf{1},\mathbf{0},\mathbf{0}), \quad \lambda_2{}^{\mathbf{i}}\mathbf{H}^{\mathbf{i}}\mathbf{x_2}{}^{\mathbf{i}} = (\mathbf{0},\mathbf{1},\mathbf{0}), \\ \lambda_3{}^{\mathbf{i}}\mathbf{H}^{\mathbf{i}}\mathbf{x_3}{}^{\mathbf{i}} = (\mathbf{0},\mathbf{0},\mathbf{1}), \quad \lambda_4{}^{\mathbf{i}}\mathbf{H}^{\mathbf{i}}\mathbf{x_4}{}^{\mathbf{i}} = (\mathbf{1},\mathbf{1},\mathbf{1}) \end{aligned} \tag{3.2}$$

   where the transformation in the $i$th image is denoted by $\mathbf{H^i}$ for each $i = 1, 2, 3$ and $\lambda_\mathbf{j}, j = 1, 2, 3, 4$ are the scale factors for the four points. Since $\mathbf{H^i}$ can be defined up to a global scale, we set $\lambda_\mathbf{4} = \mathbf{1}$. Combining the first three equations result in

$$\begin{gathered} \mathbf{H^{-i}} = [\mathbf{x_1}{}^{\mathbf{i}} \;\; \mathbf{x_2}{}^{\mathbf{i}} \;\; \mathbf{x_3}{}^{\mathbf{i}}](\lambda_1{}^{\mathbf{i}} \;\; \lambda_2{}^{\mathbf{i}} \;\; \lambda_3{}^{\mathbf{i}})^{\mathbf{T}} \\ or \\ \mathbf{H^i} = [\lambda_1{}^{\mathbf{i}}\mathbf{x_1}{}^{\mathbf{i}} \;\; \lambda_2{}^{\mathbf{i}}\mathbf{x_2}{}^{\mathbf{i}} \;\; \lambda_3{}^{\mathbf{i}}\mathbf{x_3}{}^{\mathbf{i}}]^{-\mathbf{1}} \end{gathered} \tag{3.3}$$

   Substituting the $\mathbf{H^i}$ into the equation of the fourth point, we have

$$(\lambda_1{}^{\mathbf{i}} \;\; \lambda_2{}^{\mathbf{i}} \;\; \lambda_3{}^{\mathbf{i}})^{\mathbf{T}} = [\mathbf{x_1}{}^{\mathbf{i}} \;\; \mathbf{x_2}{}^{\mathbf{i}} \;\; \mathbf{x_3}{}^{\mathbf{i}}]^{-\mathbf{1}}\mathbf{x_4}{}^{\mathbf{i}} \tag{3.4}$$

   which gives a set of three equations for each image to compute $\lambda_1{}^{\mathbf{i}}$, $\lambda_2{}^{\mathbf{i}}$ and $\lambda_3{}^{\mathbf{i}}$.

2. Apply the transformation $\mathbf{H^i}$ on the coordinates of the other two points $\mathbf{x_5}{}^{\mathbf{i}}$ and $\mathbf{x_6}{}^{\mathbf{i}}$. The transformed points have homogenous coordinates $(\mathbf{x_5}{}^{\mathbf{i}}, \mathbf{y_5}{}^{\mathbf{i}}, \mathbf{w_5}{}^{\mathbf{i}})$ and $(\mathbf{x_5}{}^{\mathbf{i}}, \mathbf{y_5}{}^{\mathbf{i}}, \mathbf{w_5}{}^{\mathbf{i}})$.

3. In the canonical frame, the five point correspondences across three views $\mathbf{x_j}{}^{\mathbf{i}}, (j = 1, .., 5)$ are images of space points $(1, 0, 0, 0)$, $(0, 1, 0, 0)$, $(0, 0, 1, 0)$, $(0, 0, 0, 1)$, and$(1, 1, 1, 1)$ respectively. To recover the unknown coordinates $(\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{W})$ of the sixth point corresponding to the triplet $\mathbf{x_6}{}^{\mathbf{i}}$, we have

$$\begin{bmatrix} 1 & 0 & 0 & 1 & {x_5}^i & {x_6}^i \\ 0 & 1 & 0 & 1 & {y_5}^i & {x_6}^i \\ 0 & 0 & 1 & 1 & {w_5}^i & {w_6}^i \end{bmatrix} = \begin{bmatrix} \alpha^i & 0 & 0 & \delta^i \\ 0 & \beta^i & 0 & \delta^i \\ 0 & 0 & \gamma^i & \delta^i \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & X^i \\ 0 & 1 & 0 & 0 & 1 & Y^i \\ 0 & 0 & 1 & 0 & 1 & Z^i \\ 0 & 0 & 0 & 1 & 1 & W^i \end{bmatrix} \tag{3.5}$$

The parameters $\alpha^{\mathbf{i}}$,$\beta^{\mathbf{i}}$,$\gamma^{\mathbf{i}}$ and $\delta^{\mathbf{i}}$ are of the camera matrix for each $i = 1, 2, 3$. From this equation, the values of the sixth space point and camera parameters may be obtained in terms of the fifth and sixth image coordinates as follows:

$$\begin{bmatrix} {w_5}^i & 0 & -{x_5}^i & {w_5}^i - {x_5}^i \\ 0 & {w_5}^i & -{y_5}^i & {w_5}^i - {y_5}^i \\ {w_6}^i X & 0 & -{x_6}^i Z & {w_6}^i W - {x_6}^i W \\ 0 & {w_6}^i X & -{y_6}^i Z & {w_6}^i W - {y_6}^i W \end{bmatrix} \begin{bmatrix} \alpha^i \\ \beta^i \\ \gamma^i \\ \delta^i \end{bmatrix} = 0 \tag{3.6}$$

The $4 \times 4$ matrix on the left has rank 3 so that its determinant must be equal to zero. Expending this determinant, we find

$$\begin{gathered} (-\mathbf{x_5}^{\mathbf{i}}\mathbf{y_6}^{\mathbf{i}} + \mathbf{x_5}^{\mathbf{i}}\mathbf{w_6}^{\mathbf{i}})(\mathbf{WX} - \mathbf{YZ}) + (\mathbf{x_6}^{\mathbf{i}}\mathbf{y_5}^{\mathbf{i}} - \mathbf{y_5}^{\mathbf{i}}\mathbf{w_6}^{\mathbf{i}})(\mathbf{WY} - \mathbf{YZ}) \\ +(-\mathbf{x_6}^{\mathbf{i}}\mathbf{w_5}^{\mathbf{i}} + \mathbf{y_6}^{\mathbf{i}}\mathbf{w_5}^{\mathbf{i}})(\mathbf{WZ} - \mathbf{YZ}) + (-\mathbf{x_5}^{\mathbf{i}}\mathbf{w_6}^{\mathbf{i}} + \mathbf{y_5}^{\mathbf{i}}\mathbf{w_6}^{\mathbf{i}})(\mathbf{XY} - \mathbf{YZ}) + \\ (\mathbf{x_5}^{\mathbf{i}}\mathbf{y_6}^{\mathbf{i}} - \mathbf{y_6}^{\mathbf{i}}\mathbf{w_5}^{\mathbf{i}})(\mathbf{XZ} - \mathbf{YZ}) = \mathbf{0} \end{gathered} \tag{3.7}$$

This is true in each of the three images. Therefore, we have three equations, which can expressed as $\mathbf{Mt} = \mathbf{0}$, while $\mathbf{M}$ is a $3 \times 5$ matrix with image coordinates of $\mathbf{x_5}^{\mathbf{i}}$ and $\mathbf{x_6}^{\mathbf{i}}$ and

$$\mathbf{t} = (\mathbf{WX} - \mathbf{YZ}, \mathbf{WY} - \mathbf{YZ}, \mathbf{WZ} - \mathbf{YZ}, \mathbf{XY} - \mathbf{YZ}, \mathbf{XZ} - \mathbf{YZ}) \tag{3.8}$$

A general solution of $\mathbf{t}$ is $\mathbf{t_1} + \mu_\mathbf{2}\mathbf{t_2}$. $\mathbf{t_1}$ and $\mathbf{t_2}$ are two null space (or eigenvectors of 2 smallest eigenvalues) of the matrix $\mathbf{M}$ and $\mu_\mathbf{2}$ is a scale factor. The constraints among elements of $t$ leads to a cubic equation in $\mu$, which has one or three real solutions for $\mu$ and therefore will be one or three solutions for $\mathbf{t}$.

4. For each $\mathbf{t}$ obtained from the last step, recover the space point $(\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{W})$ and three projection matrices $(\mathbf{P^1}, \mathbf{P^2}, \mathbf{P^3})$ (refer to equation 3.6), and calculate the reprojection error of the six points.

5. Select the triplet of projection matrices that correspond to the smallest reprojection error as the best solution.

6. Transform the camera matrix of the first image to be $[\mathbf{I}|\mathbf{0}]$ by applying a non-singular $4 \times 4$ matrix $\mathbf{R}$ such that $\mathbf{P^1R} = [\mathbf{I}|\mathbf{0}]$, then there exists a pair of reduced camera matrices $([\mathbf{I}|\mathbf{0}], \mathbf{P^2R}, \mathbf{P^3R})$.

7. Undo the normalization that was carried out in the first step on the camera matrices. Produce the tensor with coefficients of the the three de-normalized camera matrices.

It is also important to normalize the coordinates of the 6 points on each image before execution of the above algorithm.

## 3.4 The RANSAC methods

Tensor estimation has proved to be difficult in practice because the image matches are not likely to all be correct. To overcome this difficulty, it is possible to apply solutions based on the principle of random sampling. Using such an approach, the tensor can be estimated from the valid inlier matches identified using an iterative procedure. The standard RANSAC and improved schemes are discussed in this section.

### 3.4.1 Standard RANSAC

Many computer vision algorithms include a robust estimation step where model parameters are computed from a data set containing a significant proportion of outliers. RANSAC is an algorithm for robust fitting of models in the presence of many data outliers [9].

The algorithm assumes that the model parameters can be estimated from $\mathbf{m}$ data items (where $\mathbf{m} \ll \mathbf{N}$, with $\mathbf{N}$ being the total number of available data items). If the fraction of outliers in the data set is $\mathbf{e}$, then the probability of a randomly selected item being part of the actual model is $(\mathbf{1} - \mathbf{e})$.

The standard RANSAC algorithm proceeds as follows:

1. Randomly select $\mathbf{m}$ data items

2. Estimates the model parameters

3. Find how many data items (of $\mathbf{N}$) fit the model with the computed parameters within a user given tolerance.

4. If the number of supporting data, denoted by $\mathbf{k}$, is large enough, accept a candidate model and exit with success.

5. Repeat the last steps from 1 to 4 until a maximum of $\mathbf{L}$ iterations have been performed.

When the $\mathbf{m}$ selected points are all inliers, then all other inliers in the data set should also agree with the computed model. This means that the number of supporting data items (also called as the score of the model) should be equal to $\mathbf{k} = (\mathbf{1} - \mathbf{e}) \times \mathbf{N}$. However, this is rarely the case because inliers are also corrupted by (Gaussian) noise. Consequently, the acceptable value for $\mathbf{k}$ is set to be a smaller value.

The value of $\mathbf{L}$ is determined by the probability of selecting $\mathbf{m}$ inliers in a set of $\mathbf{N}$ values with $\mathbf{e} \times \mathbf{N}$ outliers. Consequently, a good value for $\mathbf{L}$ would be the number of trials required for the probability of selecting $\mathbf{m}$ inliers to be equal to $\mathbf{p}$. The value of $\mathbf{L}$ is therefore given by $\mathbf{L} = \frac{\lg(\mathbf{1}-\mathbf{p})}{\lg(\mathbf{1}-(\mathbf{1}-\mathbf{e})^{\mathbf{s}})}$. Table 3.2 gives the number of trials that would be required for different values of $\mathbf{m}$ in the case of $\mathbf{p} = \mathbf{0.99}$.

| size of subset | portion of outliers | | | | | | |
|---|---|---|---|---|---|---|---|
| s | $e$=5% | 10% | 15% | 20% | 25% | 30% | 40% |
| 6 points | 4 | 7 | 16 | 24 | 37 | 97 | 293 |
| 7 points | 4 | 8 | 20 | 33 | 54 | 163 | 588 |
| 8 points | 5 | 9 | 26 | 44 | 78 | 272 | 1177 |

Table 3.2: Standard RANSAC trail numbers

The RANSAC algorithm has been widely used in the estimation of the homography, the fundamental matrix or the trifocal tensor. A homography is computed from a minimum of 4 pairs of point correspondences, while at least 7 pairs of correspondences are required to estimate a fundamental matrix. Using Quan's algorithm

a tensor is estimated using 6 points. Note that compared with homography and fundamental matrix estimation, using RANSAC to compute tensors can be computationally expensive. The reason is that tensors must be computed from numerous triplets of point correspondences, and each of these must be evaluated against all data points. By their nature, three random correspondence triplets are more likely to contain outliers than a pair of matches. This means that more random samples are necessary to compute a valid tensor than a fundamental matrix.

The main drawback of RANSAC is its computational complexity, which increases rapidly with the number of matches and the proportion of outliers. Ways to improve the speed of RANSAC schemes are proposed in [31, 32] and [30]. A detailed discussion of this issue will be given in the next section.

Another potential problem with RANSAC for the estimation of the homography, fundamental matrix and trifocal tensor is that it does not always work well in a multi-planar scene. This problem has already been addressed in some papers [11, 42]. If the selected points are accidently collinear or coplanar, the resulting tensor can only provide correct geometry for one plane.

The standard RANSAC is implemented in this thesis. Tensors are computed from randomly selected 7 correspondences by using linear method or algebraic minimization, or 6 correspondences by 6-point method of Quan. Our online tensor estimation method is presented in the next chapter. In our case, the iteration process stops when one of the following conditions is true:

1. The number of random trials exceeds a preset maximum number.

2. The accumulated computation time is over the time budget.

3. The number of the supporting matches of the tensor is larger than an expected number.

The default setting used in experiments is: the maximum number of trials is 1000, the time limit is 10000 ms and the expected size of supporting matches is $0.8 \times N$. The "inliers" are defined as points whose transfer error is less than 5-pixels.

### 3.4.2 The accelerated RANSAC

With RANSAC, the two most time consuming operations are: model parameter estimation and support set evaluation. If $\mathbf{L}$ random trials are necessary before the algorithm finds a good model, the total processing time is equal to $\mathbf{L} \times (\mathbf{t_h} + \mathbf{t_e})$, where

1. The time $\mathbf{t_h}$ spent producing a hypothesis model, which does not vary because of the fixed size of each sample.

2. The level of contamination determines the number $\mathbf{L}$ of random samples that have to be taken to guarantee a certain confidence in the optimality of the solution.

3. $\mathbf{t_e}$ is the time spent evaluating the quality of each of the hypothesized model parameters. It is proportional to the size $N$ of the data set.

A new randomized (hypothesis evaluation) version of the RANSAC algorithm, *R-RANSAC*, is introduced in [31]. During the support set evaluation step, computational savings are achieved by considering only a fraction of the available data points. The idea comes from the observation that the erroneous models obtained from contaminated samples are consistent with only a small fraction of the data. This implies that it is not necessary to evaluate such models over all data points. Only a small number of data points need to be tested to conclude, with high confidence, that a given model does not correspond to the desired solution.

In the new algorithm, the evaluation on all $\mathbf{N}$ data points is carried out only if the tensor passes a preliminary test on $\mathbf{d}$, $(\mathbf{d} < \mathbf{N})$ points. Even though the pre-test is, with a high probability, effective to reject the erroneous models quickly, two types of errors would also be made in this step: (1) rejection of a good model; (2) acceptance of a contaminated model. The occurrence of errors of these two types would slow down the RANSAC procedure. Therefore the increase in the speed of the modified RANSAC method depends on the likelihoods of these two types of errors.

Chum [31] demonstrated experimentally on synthetic and real data that R-RANSAC for computing fundamental matrix shortens the computation time by about 40%. However no experiments for estimating the trifocal tensor is given.

Locally optimized RANSAC (*LO-RANSAC*) is another improved RANSAC method proposed by Chum [32]. Instead of keeping selecting a random sample to compute the model parameters until the best estimation is found, LO-RANSAC tries to optimize the best samples obtained so far to get the best solution. A number of experiments on two-image estimation of fundamental matrix and homography shows that compared with the standard RANSAC, LO-RANSAC offers a two to three fold speed-up and an increase in the quality of estimations (measured by the number of inliers) by $10 - 20\%$.

Nister proposed the method of Preemptive RANSAC (*P-RANSAC*) and showed its promising usage in computing relative camera motion between two calibrated views given five corresponding points [30]. The algorithm consists of repeatedly evaluating a number of computed models at $\mathbf{N}$ levels by different sets of data and only keeping some of the models for the next level. How many models remain at each level is indicated by a decreasing function of levels $\mathbf{f(i)i} = \mathbf{1}, ..., \mathbf{N}$. The total number of models is equal to $\mathbf{f(1)}$.

The P-RANSAC algorithm is summarized in Table 3.3 by the following steps:

P-RANSAC is significantly fastened because (1) only $\mathbf{f(1)}$ hypotheses produced at the $1^{st}$ level are evaluated; (2)the hypotheses are scored on single observation instead of all observations. This is the novelty of this algorithm, but also the weakness. There are two assumptions that this algorithm is built on: the desired solution must exist within the hypotheses of number $\mathbf{f(1)}$; the observations randomly selected in the scorings have high probability of being inliers. However, the assumptions can not be guaranteed in practice.

However, P-RANSAC has two important advantages: first the processing time is invariant to the size of the correspondence set; second, scoring of tensors over correspondences is not required anymore. Instead, the tensors are compared against

1. Randomly permute the available data.

2. At the first level($\mathbf{i} = \mathbf{1}$), generate the models $\mathbf{M_h}$, when $h = 1, ..., f(1)$.

3. Evaluate all models using the $i^{th}$ point in the data set and accumulate their error.

4. Reorder the models in such order that the best model obtained so far is the first. This means only keeping the first $\mathbf{f(i)}$ models.

5. Quit at the last level (when $\mathbf{i} = \mathbf{N}$) with the remaining models. Otherwise, increase $i$ by 1 and go to step 3.

Table 3.3: P-RANSAC algorithm summary in [30]

each other based on their accumulated transfer error.

As a part of the thesis work, the algorithm of R-RANSAC and Preemptive RANSAC are implemented for tensor estimation. Some experimental results will be given in the next section.

## 3.5 Experiments

This section presents an experimental comparison of the methods discussed in the previous section. Experiments are conducted to examine the accuracy of each method and test their stability against noise or outliers.

### 3.5.1 Error Definition

Two criterions exist to measure the quality of the estimated tensor. They are transfer error and residual error (also called reprojection error).

The residual error is defined by the distance between the observed image points

and the reprojected points. Each point correspondence $(x, x', x'')$ is triangulated to compute the corresponding space point $X$. Then $X$ is projected on each image plane using the projection matrices recovered from the given tensor. This gives a reprojected triplet $(\hat{x}, \hat{x'}, \hat{x''})$. The residual error of three views is then:

$$\mathbf{e_{res}} = \frac{\mathbf{1}}{\mathbf{6n}} \sum (\mathbf{dist}(\mathbf{x}, \hat{\mathbf{x}}) + \mathbf{dist}(\mathbf{x'}, \hat{\mathbf{x}'}) + \mathbf{dist}(\mathbf{x''}, \hat{\mathbf{x}''})) \tag{3.9}$$

In theory this is a reliable measurement of the quality of the tensor because the projective constraints enclosed in the tensor are evaluated.

The transfer error, based on the transfer property of the trifocal tensor, is easier to implement. It is defined by the distance between the transferred points and the measurements.

$$\mathbf{e_{trans}} = \frac{\mathbf{1}}{\mathbf{6n}} \sum (\mathbf{dist}(\mathbf{x}, \hat{\mathbf{x}}) + \mathbf{dist}(\mathbf{x'}, \hat{\mathbf{x}'}) + \mathbf{dist}(\mathbf{x''}, \hat{\mathbf{x}''})) \tag{3.10}$$

The points $(\hat{x}, \hat{x'}, \hat{x''})$ are the transferred points.

In general, there are two reasons why a given point would give a large residual or transfer error: (1)The computed tensor is not accurate; (2)The point is an outlier. In the following sections, the tensor computed by different methods from the same point set is evaluated in terms of these two types of error.

### 3.5.2 Experiments on calibrated images

The purpose of the experiments in this section is to compare the quality of the tensors directly constructed from projection matrices and those computed from the image points by different algorithms.

Figure 3.1, 3.2 and 3.3 show three image triplets. The corner matches across three images and the projection matrices of images are also known and they will serve as as ground truth here.

Experimental Results using the above image triplets are shown in Table 3.4. Three estimation methods are compared here against the tensor constructed from projective matrices using equation 2.3. The column "Matches" displays how many point matches

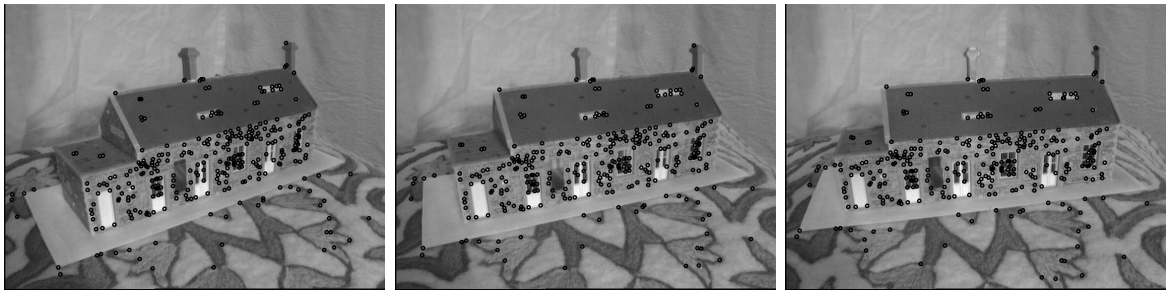


Figure 3.1: Test images from the house sequence (298 point matches)

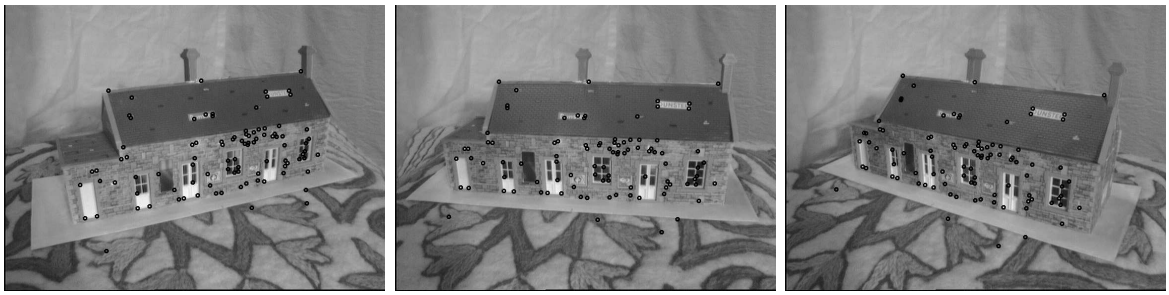


Figure 3.2: Another three widely separated images (95 matches) are selected from the house sequence. They are used as an example of wide-baseline images

are used in the estimation. The transfer error and residual error of each computed tensor are also listed in the table.

The 6-point RANSAC's setting is: time budget = 100000ms, max inlier distance = 3, max samples = 1000, stop precision = 1.5. The final tensor is the one having the best score computed with the RANSAC procedure.

| Images | Matches | Constructed tensor | Linear Method | Algebraic Minimization | 6-point RANSAC |
|---|---|---|---|---|---|
| house (0,1,2) | 298 | 0.54, 24.59 | 0.54, 2494.35 | 0.53, 255.05 | 3.85, 36.67 |
| house(1,4,6) | 95 | 1.230, 7.77 | 1.15, 2533.81 | 1.170, 1019.98 | 1.54, 12.60 |
| corridor(0,3,6) | 199 | 3.12, 10.49 | 0.56, 149.7 | 0.53, 53.22 | 0.85, 14.58 |

Table 3.4: Tensors computed with exact correspondences on three image triplets. In each column, transfer error is on the left while reprojection error is on the right.

Note that the large reprojection errors are due to inaccuracy of the 3D point reconstruction.

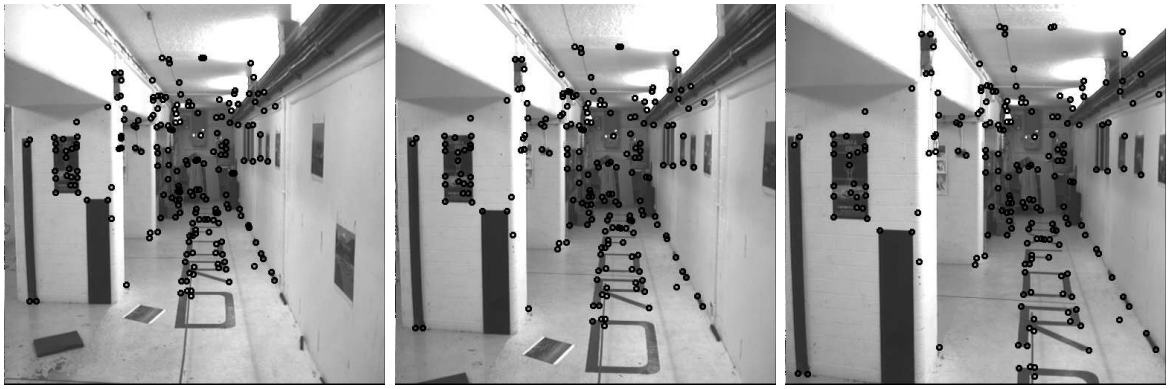


Figure 3.3: Three images from the corridor sequence (199 matches)

### 3.5.3 Experiments on noisy image points

The goal of the following experiments is to study the influence of image noise or outliers on different tensor estimation methods. Understanding the strength and weakness of each method is very helpful when it comes to selecting one for a particular application.

Three image triplets as shown in figure 3.1, 3.2 3.3 and their exact point correspondences will be used as experiment data here. New tensors will be computed from new sets of matches after they have been contaminated by different types of noise.

In the first test, images points are perturbed by a uniformly distributed noise of $\pm s$ pixels. Tensors will be computed at different levels or amplitudes of noises. In Table 3.5, the tensor computed from the contaminated data is evaluated using the exact correspondences. The impact of the added white noises is measured by comparison with the tensor constructed from the exact projection matrices. Since RANSAC is a random procedure, it is executed 20 times in the experiments and the tensor with the highest score is selected as the final result.

The results show that the linear method and algebraic minimization method are not too affected by low level of noise. However, the linear method is the most unreliable method when the accuracy of projective matrices recovered from tensors are considered. 6-point RANSAC doesn't present any advantage in handling image data

| Images | Matches | Noise $\pm s$ pixels | Constructed tensors | Linear Method | Algebraic Minimization | RANSAC 6-point |
|---|---|---|---|---|---|---|
| house (0,1,2) | 298 | 0.1 | 0.55, 25 | 0.53, 7707 | 0.53, 2798 | 5.55, 118 |
| | | 1 | 1.42, 82 | 0.56, 5521 | 0.56, 2385 | 5.84, 1251 |
| | | 2.5 | 3.27, 446 | 30.92, 125135 | 0.76, 12242 | 3.19, 148 |
| | | 5 | 6.27, 1707 | 0.72, 16891 | 0.70, 373 | 5.93, 206 |
| | | 7.5 | 9.93, 1744 | 1.51, 2290 | 1.52, 5216 | 8.27, 103 |
| | | 10 | 12.96, 2551 | 2.37, 1885 | 2.26, 761 | 10.78, 233 |
| corridor (0,3,6) | 199 | 0.1 | 1.63, 9 | 0.56, 152 | 0.53, 24 | 0.53, 261 |
| | | 1 | 1.82, 23 | 1.57, 887 | 1.56, 60 | 2.38, 219 |
| | | 2.5 | 3.62, 43 | 3.59, 897 | 3.57, 405 | 3.59, 350 |
| | | 5 | 4.64, 376 | 2.72, 1410 | 2.74, 79 | 4.97, 291 |
| | | 7.5 | 5.49, 514 | 21.251, 353 | 15.948, 475 | 7.57, 123 |
| | | 10 | 7.71, 874 | 8.79, 1280 | 9.52, 942 | 14.07, 404 |

Table 3.5: Results of different algorithms on two image triplets. The exact point matches are distorted by white Gaussian noise at different levels

globally disturbed by white noise. This can be explained by the fact that RANSAC expects at least a minimum number of correct data to obtain an acceptable estimation.

The second tests are carried out on point sets that contains different number of false matches, which are randomly inserted into the correspondence set. Tensors computed by the four algorithms are compared again in terms of transfer error and support point number in the table 3.6. The column 'outliers' show the percentage of false matches in test data.

This experiment demonstrates that 6-point RANSAC is effective in classifying exact matches and outliers. In the iterative AM method, an optimized tensor is obtained by searching the local minimum around an initial tensor. As shown in the above table, the optimization may not succeed at all time.

The next experiment compares the performance of two accelerated RANSAC approaches against the standard RANSAC method. Since we are more interested in their usage in real-time applications, the time budget is set to 1500ms and the maximum sample number is 1000. The RANSAC program was executed 50 times and

| Images | Matches | Outliers % | Linear Method | Algebraic Minimization | Iterative AM | 6-point RANSAC |
|---|---|---|---|---|---|---|
| house (0,1,2) | 298 | 0.1 | 9.05, 107 | 9.16, 106 | 9.39, 103 | 1.63, 282 |
| | | 0.15 | 7.08, 149 | 7.42, 143 | 7.19, 144 | 1.06, 296 |
| | | 0.2 | 6.88, 155 | 7.07, 148 | 6.94, 156 | 1.05, 298 |
| | | 0.25 | 9.83, 85 | 9.59, 97 | 9.24, 101 | 1.19, 286 |
| house (1,4,6) | 95 | 0.1 | 10.52, 24 | 10.06, 27 | 9.65, 26 | 1.28, 94 |
| | | 0.15 | 8.46, 35 | 8.54, 33 | 8.54, 33 | 1.81, 91 |
| | | 0.2 | 17.02, 8 | 17.32, 8 | 17.12, 9 | 1.43, 94 |
| | | 0.25 | 17.12, 23 | 17.43, 20 | 17.07, 21 | 2.19, 89 |
| corridor (0,3,6) | 199 | 0.1 | 22.39, 8 | 17.64, 8 | 16.12, 9 | 0.85, 199 |
| | | 0.15 | 10.67, 30 | 10.32, 31 | 10.41, 31 | 0.75, 199 |
| | | 0.2 | 46.25, 3 | 86.32, 1 | 41.83, 5 | 0.93, 197 |
| | | 0.25 | 104.67, 0 | 38.82, 3 | 38.82, 3 | 0.99, 197 |

Table 3.6: Results of different algorithms tested on three sets of point triplets that contain randomly generated false matches.

average process time and transfer error were taken. Different number of exact matches that are randomly selected are disturbed by ±20-pixel gaussian noises.

Here is a list of the most important parameters used in R-RANSAC and P-RANSAC program.

1. R-RANSAC: new tensors won't be scored over the whole data unless it fits a subset of randomly selected data. The size of the subset is equal to 30% of the total data number. And the tensor is required to fit at least half data in the subset.

2. P-RANSAC: to investigate the impacts of the number of hypothesis and evaluation levels on this method. P-RANSACs with two configurations are tested. One is to extract the best tensor estimation from 20 hypothesis through 6 levels of evaluation. The decreasing function $f(i) = 20, 16, 12, 8, 4, 2$, $i = 1..6$, which means the computation of the transfer error will be executed 62 times. Another P-RANSAC program uses the decreasing function $f(i)$=$200, 190, 180, ..., 20, 10, 4$, $i = 1..21$. Transfer error computation need to be carried out 2104 times to obtain the best tensor from 200 hypothesis.

The results of three RANSAC methods are summarized in the table 3.7 and 3.8. Each table shows the average transfer error of computed tensors, average computational time(in milliseconds), average number of samples evaluated and the number of "inliers" in the exact matches sets that are supported by the tensor. The column "Outliers" shows the percent of contaminated matches.

| Methods | Outliers | Average transfer error | Time (ms) | Number of samples | Number of supporting points |
|---|---|---|---|---|---|
| Std RANSAC | 0.2 | 3.27 | 1503.2 | 565.1 | 83.3 |
| | 0.4 | 6.12 | 1502.2 | 822.4 | 73.85 |
| | 0.6 | 9.05 | 896.8 | 1000 | 65.4 |
| R-RANSAC | 0.2 | 1.85 | 1511.7 | 614.25 | 89.45 |
| | 0.4 | 5.80 | 1511.7 | 1039.75 | 79.35 |
| | 0.6 | 5.55 | 1508.1 | 1163.1 | 77.3 |
| P-RANSAC (200 models) | 0.2 | 6.92 | 771.15 | 200 | 75.05 |
| | 0.4 | 25.41 | 831.15 | 200 | 48.7 |
| | 0.6 | 22.99 | 816.96 | 200 | 40.28 |
| P-RANSAC (20 models) | 0.2 | 8.63 | 76.15 | 20 | 63.85 |
| | 0.4 | 29.92 | 80.15 | 20 | 48.35 |
| | 0.6 | 70.24 | 88.92 | 20 | 32.66 |

Table 3.7: Experiment on wide baseline house images (95 correspondences) using standard and two accelerated RANSAC

The above tables show that R-RANSAC achieved the best results in the number of support points and average transfer error. The reason is that more candidate tensors are computed and tested compared to standard RANSAC before the time budget is reached. On average only about 10% of the computed tensors pass the first test when the threshold of 0.5 is applied on the scores over the subset correspondences.

However, this threshold may not be proper for all test data that have different percentage of outliers. It is actually a problem in practice since the outlier information can not be known beforehand. High thresholds will erroneously block some hypotheses while small thresholds may have the consequence of increasing RANSAC processing time.

We see that although P-RANSAC is the fastest method, its performance is quite

| Methods | Outliers | Average transfer error | Time (ms) | Sample number | Number of supporting points |
|---|---|---|---|---|---|
| Std RANSAC | 0.2 | 1.34 | 1516.7 | 280.1 | 194.85 |
| | 0.4 | 2.27 | 1502.15 | 477.6 | 175.35 |
| | 0.6 | 4.80 | 1414.44 | 992.02 | 152.98 |
| R-RANSAC | 0.2 | 0.56 | 939.8 | 359.1 | 199 |
| | 0.4 | 1.33 | 1478.1 | 509 | 190.9 |
| | 0.6 | 3.22 | 1524.4 | 1141.78 | 169.84 |
| P-RANSAC (200 models) | 0.2 | 3.79 | 716.8 | 200 | 160.72 |
| | 0.4 | 5.44 | 764.1 | 200 | 139.92 |
| | 0.6 | 17.95 | 792.9 | 200 | 108.28 |
| P-RANSAC (20 models) | 0.2 | 4.01 | 91.6 | 20 | 163.5 |
| | 0.4 | 26.18 | 98.15 | 20 | 129.35 |
| | 0.6 | 20.62 | 91.74 | 20 | 92.28 |

Table 3.8: Experiment on corridor images (199 correspondences) using standard and two accelerated RANSAC

unstable. Increasing the number of hypothesis can improve its accuracy at a certain level, but also reduce its advantages in realtime applications.

The comparative efficiency of the three methods is illustrated in Figure 3.4 where the number of points in the support set is shown for the case of a match set containing 60% outliers.

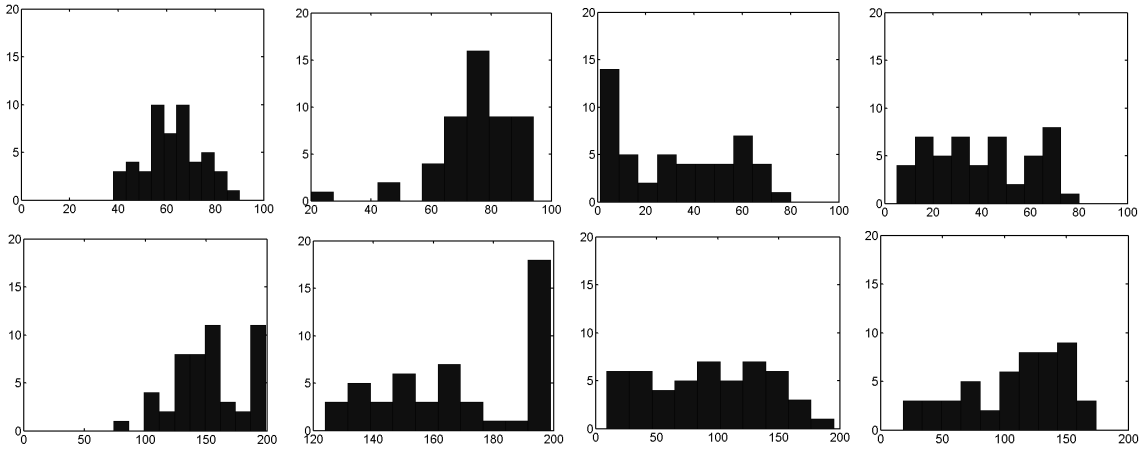



Figure 3.4: Histograms of the number of supporting points. Results of house images and corridor images are shown in rows. The four methods (Standard RANSAC, R-RANSAC, 6-level and 21-level P-RANSACs) are stored in columns. X-axis an Y-axis of each graph indicate the number of supporting points and how many times this number was reached within 50 iterations.

# Chapter 4

# Online tensor estimation

Estimating multiple view entities in video sequences is the basic building block of structure-and-motion (SaM) systems to recover 3D structure and camera motions. In most SaM systems these view entities are either relative to adjacent frames or to some selected keyframes. In this chapter, we will propose an approach that uses three reference frames to compute trifocal tensors in video sequences at realtime. Known image correspondences of three reference frames are tracked on the images of the moving camera to produce triplets, from which accurate tensors can be obtained by a combination of several tensor estimation methods. This approach provides a solution not only for realtime camera pose estimation (when the camera is well calibrated beforehand), but also for applications like augmented reality that will be discussed in the next chapter.

## 4.1 Literature review

Structure-and-motion (SaM) is the technology to recover camera motion and 3D scene structure from video sequences captured by the camera. The multiple view entities of video frames, including homography, fundamental matrices and trifocal tensors, are very important in SaM because they contain information about camera's relative motion throughout the frames and they can be obtained from images only.

A basic SaM system generally proceeds as follows: (1)first features are matched across multiple frames into pairs or triplets and multiple view entities are computed; (2)pairs or triplets of projective matrices from view entities are integrated into a coherent projective coordinate frame;(3)the resulting projective matrices must then be upgraded into Euclidean coordinates by the use of auto-calibration or calibration patterns;(4)finally 3D models of the scene are built using the computed camera matrices if required.

The second step where multiple view entities are calculated and registered into the same projective coordinate system is the most important. Different methods used at this step make SaM systems distinct from each other. They can be divided into two categories. An overview of each category will be given.

### 4.1.1 Approaches using chained transformations

Approaches of the first category are devoted to calculating fundamental matrices or trifocal tensors of successive frames and chaining camera matrices along the sequences. The procedure is illustrated by figure 4.1.

First cross-correlation and guided matching based on fundamental matrices are used to find matches between two adjacent frames. The resulting overlapping match sets are then used to build putative triplet sets of correspondences between three selected images. By applying RANSAC on triplets of matches, tensors and fundamental matrices can be calculated along with supporting correspondences. This method to match consecutive pairs or triplets of video frames is called "F-based tracker" ("F" stands for fundamental matrix) or "T-based tracker" ("T" for trifocal tensor).

When T-based tracker is employed, each consecutive image triplet $(\mathbf{i-1}, \mathbf{i}, \mathbf{i+1})$ can have its trifocal tensor $\mathbf{T_{i-1,i,i+1}}$ computed independently. From the tensor, the projection matrices of the image triplet, denoted by $\mathbf{P^{i-1}}_{\mathbf{i-1}}, \mathbf{P^{i-1}}_{\mathbf{i}}, \mathbf{P^{i-1}}_{\mathbf{i+1}}$, can be determined by equation 2.11. They are subjected to $\mathbf{P^{i-1}}_{\mathbf{i-1}} = [\mathbf{I}|\mathbf{0}]$.

Adjacent image triplets are then registered into a long chain by the homographies $\mathbf{H^{i-1,i}}$ linking projection matrices of the overlapping views $\mathbf{P^{i-1}}_{\mathbf{i}} = \mathbf{H^{i-1,i}} \times \mathbf{P^{i}}_{\mathbf{i}}$. In

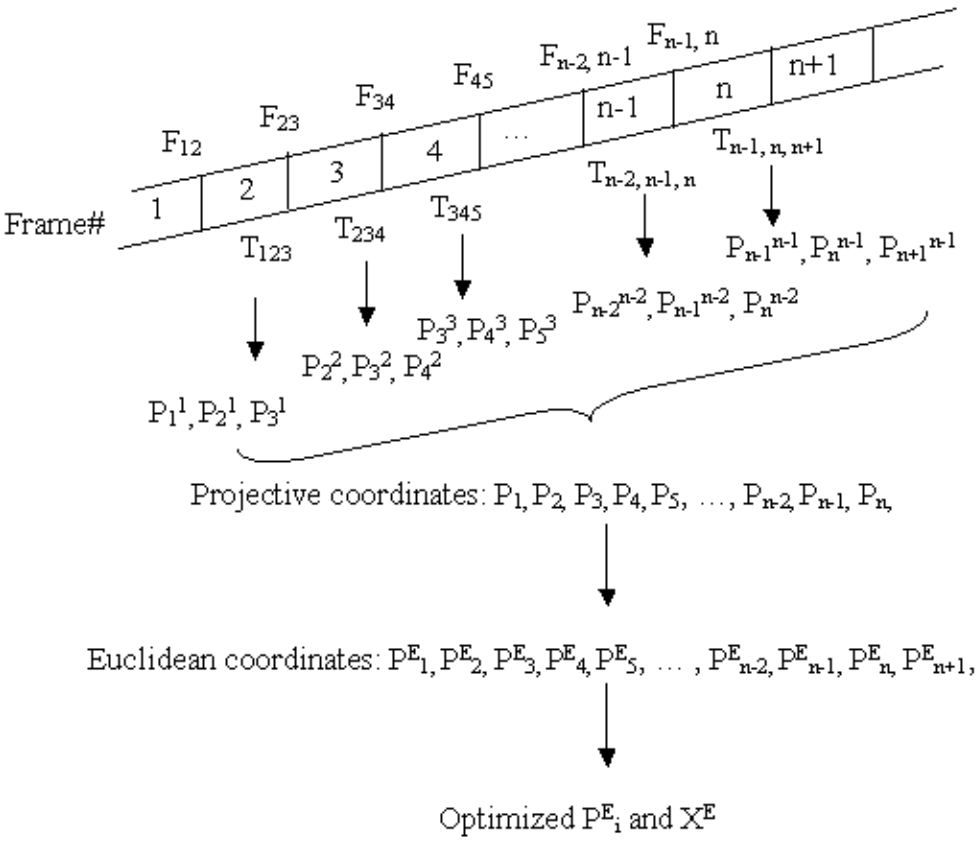



Figure 4.1: Scheme of the approaches based on chaining camera matrices along the video sequence

this way, the projection matrices of all images are transformed into a common projective coordinate systems. Bundle adjustment is then applied on them to obtain optimized camera matrices and 3D scene structure simultaneously.

This mechanism has been used in many applications, ex: camera pose estimation [36], object-based video compression [38], 3D modelling [4] and augmented reality [46]. It has two major problems: (1)the error accumulated during the chaining operation can result in inaccurate reconstruction of long sequences; (2) the accuracy of the resulting tensor is doubtful because of the small motions involved in consecutive three frames.

A hierarchical scheme of registration proposed in [10, 29] attempts to reduce the error accumulation problem. The core of this scheme is to divide the whole sequence into sub-sequences, register triplets into consistent sub-sequence using homographies and register sub-sequences, again using homographies, to obtain cameras and structure for the complete sequence. Bundle adjustment can be applied in each step of registration throughout the hierarchical structure to increase accuracy.

In order to achieve reliable tensors, frames of every view triplet have to be reasonably separated. The number of frames needed to create a reasonable separation varies from one video sequence to another. Sometimes it is even necessary to keep changing this number over a single sequence.

In sum, the above approaches are designed for offline processing of recorded videos because (1) they process images in a batch mode. Both past and future frames are required in estimating camera pose of the current frame; (2) Because of using bundle adjustment and other non-linear optimization methods, these approaches can not meet speed requirement for realtime processing.

### 4.1.2 Approaches using keyframes

Approaches based on key frames or reference images of the scene that are captured during an initialization procedure compose another category of SaM systems. They consist in registering every video frame with keyframes or reference images. Therefore

the corresponding view entities are free from error accumulation and more geometrically valid because of the wider baseline that exists between the video frames and the key frames, which however, on the other hand, can result in fewer matches available for tensor or other entity estimation.

Because they avoid the computation of long chains of transformations, keyframe-based methods provide a promising solution for live video applications. Providing a few of images of the scene, the camera can freely move inside the field of view spanned by these reference images.

Chia [6] used two reference images to register video frames of a calibrated camera for an augmented reality application. Optimal camera pose is obtained by minimizing the epipolar constraints of two pairs of views composed by the current frame and two keyframes. To avoid the wide baseline matching problem the video frames are matched with the reference images through their previous frames.

Boufama [5] presents another augmented reality system using two reference images. The current video frame and the reference images build up one image triplet instead of two image pairs. The corresponding trifocal tensor is computed for registration of the current frame. Its idea of envisaging trifocal geometry is similar to the approach proposed in this thesis. We will demonstrate later that our approach is different and more effective in tracking and tensor estimation.

The papers described in [35] and [41]use multiple key frames for the tracking of a known model and the recovery of camera pose. Both model registration systems require the 3D model of the target object and a number of keyframes captured from distinct viewpoints that can supply a set of model-image correspondences. The 2D-3D matches produced by tracking of known points are feed into an iterative procedure starting from the closest known viewpoint to obtain an optimal camera pose. The first model registration system uses an aspect table generated from the keyframes and the visible points in these frames to predict the appearing points during tracking. The matching problem is simplified by only searching the predicated points on the moving views. The second system improved this by not only using off-line keyframes but also

online generated keyframes to achieve a wider span of views. Also a method combining consecutive frame processing and keyframe technique is used in camera pose estimation. In pose estimation RANSAC is applied on matches between consecutive frames, while only the pose models supporting matches between the moving view and the chosen keyframe are evaluated.

## 4.2 Online tensor estimation from reference images

This thesis presents an approach for the online estimation of the tensor in live video. It works with a single camera moving freely inside a scene of interest. Image features taken from an initial triplet set, detected on three reference images, are tracked across a video sequence. Then, as the camera moves the tensor associated with each frame and the two reference images is estimated. The method proposed in this thesis utilizes a 3-step paradigm where i) the feature points are tracked, ii) a tensor is estimated and iii) the triplet match set is updated through trifocal transfer. An important aspect of this work is the fact that the updated tensors associated with every moving frame and the two fixed reference views are computed based on a common set of detected features. As a consequence, there is no drift problem. At the same time, the points of view will always remain sufficiently wide to ensure an accurate estimation of the tensor. Also, another important point is that because two fixed reference images are used, all the projection matrices of a video sequence are in the same projective framework. This simplifies the computation of the camera matrix for each frame with respect to a global coordinate system.

This method is also distinctive in three aspects: (1) We try to avoid the use of random sampling techniques during the process of tensor estimation because of their complexity. Instead, the tensor is estimated by using the algebraic minimization method and its accuracy is improved after a quick outlier removal step; (2) A simple tracker is used to provide an evolving point set for the purpose of establishing a

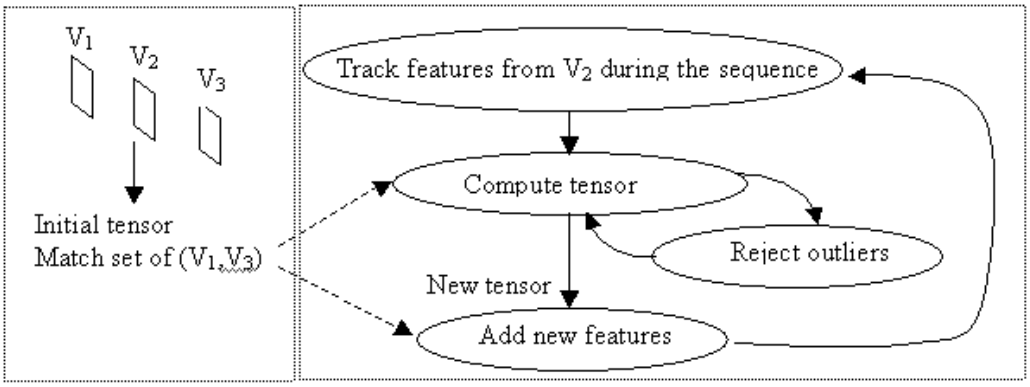



Figure 4.2: Flow chart of the proposed approach

new set of point triplets. This point set is also updated by recovering lost points and correcting mismatches using the trifocal transfer. Stable performance of tracking over a long video sequence is also ensured by automatically resetting the tracker when the size of the tracked point set becomes too small. (3) The method is flexible in the sense that the three reference images can be taken from three distinct cameras or by using a single camera moving to three locations. It is also important to note that the proposed 3-view system can easily be extended to the case of multiple views; An additional first step would be to identify the two closest reference views with which tensor estimation is to be performed. This way the scope of this vision system could be scaled to a size that a given application would require.

### 4.2.1 Approach outline

Our proposed approach is illustrated in Figure 4.2. The system has, as input, three camera views, denoted by $\mathbf{V}_1$, $\mathbf{V}_2$, $\mathbf{V}_3$ respectively. The initialization step consists of obtaining both an initial estimate of the tensor and a large set of matched triplets. Several alternatives can be envisaged in order to achieve this goal, including a tensor-based guided-matching [43] and the PVT tool described in [36]. The feature points of the obtained triplet set that belong to one reference view will constitute the initial set of point to be tracked. The match pairs between the other fixed views will serve

as a match pool that will be used, during the process, to update the list of points to be tracked.

Once the initialization process is completed, the online tensor estimation process can start. The detected points in one reference view are tracked from one frame to another. This leads to new positions of the points for which we still have the correspondences in the two fixed frames. Section 4.3 describes the methods used to produce tentative point triplets from the tracking of the feature points. Using this triplet set, robust and fast estimation of the tensor is achieved; this aspect is discussed in detail in Section 4.4.

Obviously, when points are tracked over time, more and more features are unavoidably lost. If nothing is done, the tracked set will eventually vanish. To overcome this problem, the match set is updated after each tensor estimation. Indeed, using the pool of match pair available in the two fixed views, it become possible to transfer new points on the image using the newly estimated tensor. This last step ensures the long term viability of the estimation process. A detail description of this step is given in Section 4.5.

In a multi-camera implementation, points from the view close to the current reference views would also be transferred, thus allowing us to identify to which view the moving camera is transiting.

## 4.3 Tracking feature points

The task of tracking matched points belonging to one reference view is done using the widely used Lucas-Kanade tracker [22]. During this tracking process, it is unavoidable that the tracker will lose some features and will introduce some wrong traces. This is especially true in the case of handheld cameras where quick and saccadic motion are often introduced. Therefore, additional efforts are required in order to monitor the quality of tracked points and get rid of wrongly tracked points. It would be, however,

too costly to measure the similarity between tracked points and their possible correspondences in the two reference images through cross-correlation. An alternative solution is to apply a disparity-gradient constraint on the candidate matches composed by the tracked points and their correspondences on the reference frame that is used to initialize the tracker.

Disparity gradient is used to measure the similarity of disparities of two pairs of points [34]. If the disparity of two pairs $(p_1, p_2)$ and $(q_1, q_2)$ are $d_1$ and $d_2$ respectively, their disparity gradient is equal to $\mathbf{d} = \frac{|\mathbf{d_1} - \mathbf{d_2}|}{\mathbf{l_{12}}}$. $l_{12}$ is the distance between the midpoints of lines $\overline{p_1 p_2}$ and $\overline{q_1 q_2}$. Two pairs having the same disparity will have a zero disparity gradient. Based on a reasonable assumption that a match has a similar disparity with its neighbors, a pair of points will be identified as a mismatch if its disparity gradients with less than 2 neighbor points are below a threshold (say 0.4). Applying such constraint of disparity gradient on a set of match candidates can remove many mismatches while eliminating few of the good matches.

## 4.4 Trifocal Tensor estimation

The tensor estimation part uses the set of point triplets produced from tracking as their input. As has been mentioned previously in Section 3.5, RANSAC methods are computationally too demanding to be used at frame rate. To achieve the expected performances in both time and precision, we select an algebraic minimization to estimate the tensor from all the correspondences. The average value of the transfer errors is used to assess the quality of the resulting transfer. If its value is smaller than a given threshold (we used 3 pixels), then the tensor is judged to be of good quality and can be used as is. Otherwise, additional steps to identify potential outliers are required.

### 4.4.1 Estimation of tensors with fixed projection matrices

In our framework the tensor is always computed from two fixed reference frames and one moving frame. The result is that two epipoles and two projection matrices of this view triplet are actually fixed. Assuming that in equation( 2.3), $\mathbf{P_1}$, $\mathbf{P_3}$ correspond to the projection matrices of the two reference frames and $\mathbf{P_2}$ of the moving camera, then it follows that $\mathbf{P_3}$ is known from the initial tensor of the three reference frames and so is $\mathbf{e''}$. In Algebraic Minimization, the matrix $\mathbf{E}$ is constructed in terms of the 6 entries of the two epipoles, $\mathbf{e'}$ and $\mathbf{e''}$, which are retrieved from the tensor computed by the linear least-square solution. And the tensor is expressed linearly as $\mathbf{t} = \mathbf{Ea}$ where $\mathbf{a}$ is a vector of the entries of the left $3 \times 3$ matrices in $\mathbf{P_2}$ and $\mathbf{P_3}$.

Now, to force the tensor to be consistent with the fixed $\mathbf{P_3}$ and $\mathbf{e''}$, two options are available: (1) construct matrix $\mathbf{E}$ with the fixed $\mathbf{e''}$ instead of the one computed from the initial tensor of linear solution. Then solve the equation $\mathbf{AEa} = \mathbf{0}$ to obtain the tensor; (2) skip the step to solve the linear tensor and construct a new linear system directly with the entries of $\mathbf{P_3}$. Equation 2.3 can be rearranged to

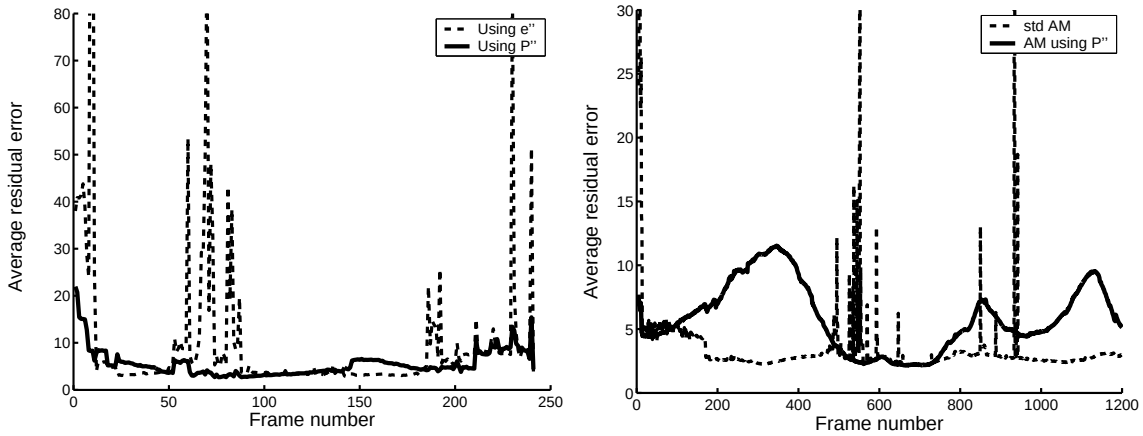
$$\mathbf{t} = \mathbf{H}(\mathbf{b_i}^\mathbf{j})\mathbf{D}(\mathbf{a_i}^\mathbf{j}) \tag{4.1}$$

$\mathbf{27 \times 12}$ matrix $\mathbf{H}$ is composed of known entries of $\mathbf{P_3}$ and $\mathbf{12 \times 1}$ column vector $\mathbf{D}$ contains the entries of $\mathbf{P_2}$. Substituting the vector $\mathbf{t}$ in $\mathbf{At} = \mathbf{0}$ yields:

$$\mathbf{AHD} = \mathbf{0} \tag{4.2}$$

The linear solution to the above equation gives the entries of the second projection matrix and the trifocal tensor.

In Figure 4.3(a), we compare the performance of these two alternative approaches for a test sequence of 250 frames. The tensor computed using $\mathbf{e''}$ is drawn with a dash line. From the experiment, it appears that the use of a fixed $\mathbf{e''}$ in the tensor computation is not very stable. In contrast, the tensor subject to the fixed $\mathbf{P_3}$ exhibits reliable performance over the entire sequence. It has capability to counter the effect of false matches and update the solution where few features are being tracked. However,




(a) (b)

Figure 4.3: Comparison of Algebraic Minimization method with the two modified methods with fixed epipoles or projective matrices

its overall accuracy remains inferior to the one obtained with the tensor computed by the standard Algebraic Minimization, as illustrated in Figure 4.3(b). Both methods are applied on a long sequence. The solid line represents the tensor subject to $\mathbf{P_3}$. The standard AM is fragile when few features are being tracked. Its instability results in peaks of error in the tensors estimated over the sequence while the fixed $\mathbf{P_3}$ approach produces a much smoother curve. In consequence, the use of a fixed $\mathbf{P_3}$ will be restricted to the case where few features are available or a large portion of false matches exists or when tracked features are not well distributed across the whole scene.

### 4.4.2 Removal of outliers

In the case where the quality of the tensor computed from all putative triplets using standard algebraic minimization is not acceptable, the tensor must be re-estimated. First a new tensor will be computed using the fixed projection matrix $\mathbf{P_3}$. It will be discarded if it has a even larger transfer error. When the error is over an upper limit, the tensor estimation procedure will stop without a tensor output. Otherwise, a step of outlier classification has to take place.

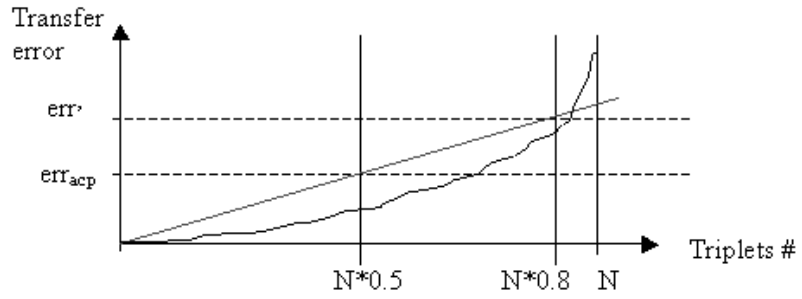



Figure 4.4: Analysis of transfer errors before implementing x84

The method we adopted requires that the obtained tensor supports a large number of triplets in the set of points. The quality of the tensor is not good mainly because of the presence of a few strong outliers. In this case, a statistical method based on the so-called x84 rule [12] is implemented. Absolute deviations of all triplets' transfer error are calculated, from which a threshold is automatically set as the 5.2×MAD(Median Absolute Deviation). The points having larger deviations are considered outliers and must then be eliminated. Once the outliers are rejected, the tensor has to be re-estimated with all remaining triplets and its quality needs again needs to be re-evaluated.

Implementing the x84 rule in outlier detection has an advantage over using a fixed threshold because the threshold actually varies from experiment to experiment and from sample to sample. However, since the rejecting threshold is determined by the distribution of the majority of the data, x84 is only effective when the tensor is disturbed by a small number of outliers. In order to determine if this condition holds for a given tensor a coarse analysis of the distribution of the transfer errors is necessary before running rule x84. This is done by sorting the transfer errors in ascending order. In figure 4.4 the error that is larger than the first 80% errors is denoted by $\mathbf{err'}$ and the average transfer error of an accepted tensor is $\mathbf{err_{acp}}$. The result is that the x84 is implemented only when the value of $\mathbf{err'}$ is smaller than $\frac{\mathbf{0.8}}{\mathbf{0.5}} \times \mathbf{err_{acp}}$.

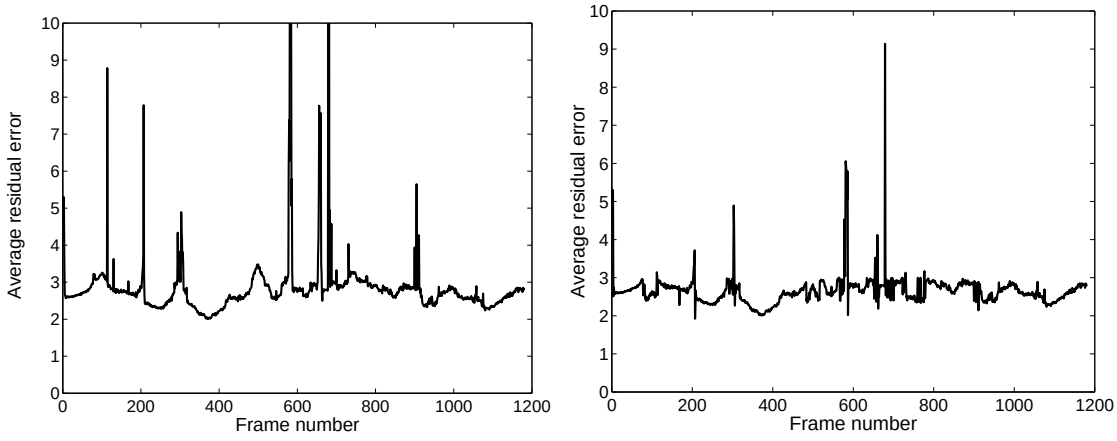



Figure 4.5: Computed tensors of a sequence of 1170 frames. The average transfer errors of the tensors computed before and after using the fixed projection matrices and x84 rules are given in the left and right plot respectively.

According to the quality of the resultant tensor, the tracked point set will be updated in different ways that will be discussed in the next section.

## 4.5 Updating of the tracked point set

The estimated tensor will be used to remove wrongly tracked points so that they won't affect the tensor estimation of the next frame. When the number of supporting triplets is relatively high, then the current tensor is able to guide the identification of outliers. Points with large transfer errors are removed from the tracking point set.

Another important use of an accurately estimated tensor is that in the addition of new points into the current set of tracked points. During tracking, the lifetime of a given tracked point largely depends on the magnitude of the camera motion. Over time, more and more points will thus be lost, mainly because they go out of the field of view or because they are occluded by some scene object. However, at the same time, other points, that are included in the initial pool of matches will appear in the view. It is therefore important, for the viability of the procedure, to identify those points and to incorporate them into the tracking process. Using the current estimation of the

tensor, these points could be identified by transferring all the reference matches (in the two fixed views) onto the current moving view. The presence of a correspondence is verified by searching in a small area around the transferred point for a point that correlates well with one of the two reference matched points. This way, the method can even recover points that have been badly tracked in the current frame.

In order to illustrate the importance of maintaining a large pool of tracked points, a long sequence(1183 frames) is recorded when a black-white square was placed on the desk. Figure 4.5 shows the reference images of this sequence and in these two views the camera's viewpoints have dramatically changed. Its four corners on every frame are detected and compared with the predicted corner positions computed from the estimated tensor. The number of tracked features across the sequence and the offset between tensor-predicted corners and their true detected positions before and after recovering lost and inaccurate features are shown respectively in the left and right plot in Figure4.5. Note that in the without-adding-features case, the tracker is reset automatically when too few features remain. In this experiment, this happens at the $624th$ frame.

## 4.6 Additional experiments

Our system runs on a desktop PC with a web camera of image resolution $320 \times 240$ pixels. Experiments have been performed on various sequences composed of thousand of frames. Undoubtedly, the performance varies according to different conditions appearing in the videos. The number of points in the match pool as well as the number of points actually tracked have also an important impact on the performance. On average, the median residual error of the tensor is around 3 pixels and the processing is carried out at a speed of 14fps.

An analysis of processing time is given in Figure 4.6. For each frame, about $8.32ms$, that is 12% of the time, is spent on tracking features along the sequence. Another 32% of the time is consumed on estimating the tensor. Establishing triplets

Figure 4.6: Reference images of the 1183-frame sequence Black-white Square. Corners of the square are not in the loop of tracking.

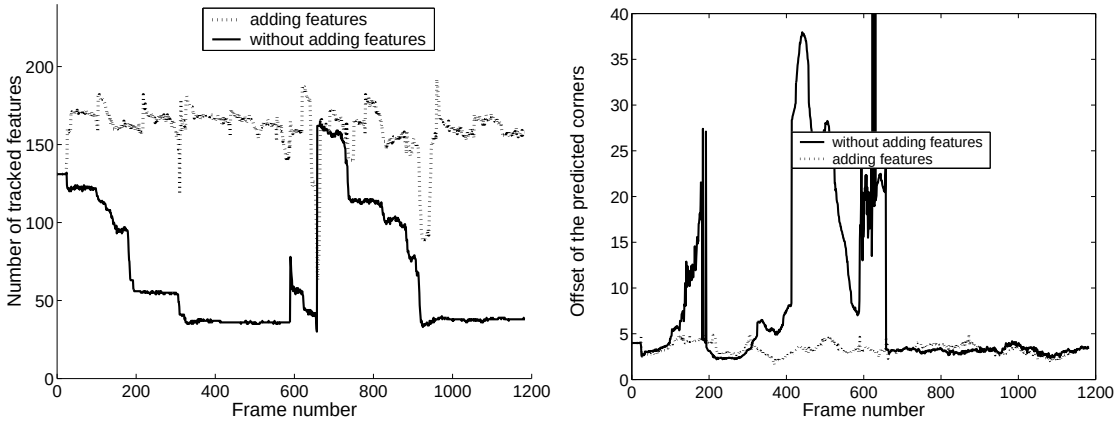



Figure 4.7: Results of the sequence Black-white Square

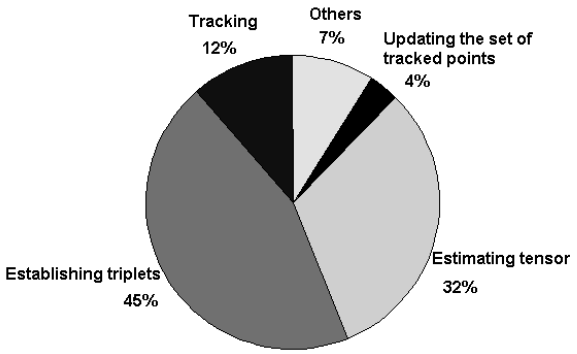



Figure 4.8: Timing chart

for every video frame consisting of point-wise disparity-gradient test takes the largest portion of time. It varies with the number of features tracked along the sequence. Typically in our experiments the initial set of triplets over three reference images contains around 150 features.

Figures 4.6 to 4.6 presents the tensor estimation results on two long video sequences. These sequences include large camera motions resulting in viewpoints that differ significantly from the reference images. At certain instants, only a few points are currently visible, which increases the difficulty of the tracking and tensor estimation tasks. The experiment results demonstrate that our system has the capability of achieving good tensor estimation throughout long sequences. Images given in Figure 4.6 show that our approach is able to transfer scene points (here the corners of a CD envelop) correctly from the reference views to the current view.

The motion blur problem occurring in the sequences captured by the web camera used is very significant, which often results in heavily contaminated putative triplets. The tensors computed from these set of triplets were difficult to correct. Most of the bad tensors obtained in our experiments due to this reason.

Our system also encounters the problem in resuming the tracking after it has stopped for some reason. The user then has to move the camera to a viewpoint close to the reference image that is used to initialize the tracker. No tensor is computed during this period of time. One way to shorten this waiting time is to match the

current frame with the closest reference image. How to choose that reference image remains a problem. Most existed methods [41] need the information about the current camera pose and those of the reference images. The paper described in [35] uses points with a different appearance on the reference frames to distinguish them.

Figure 4.9: Reference images prepared for video sequence Table (2026 frames). The initial set of 154 matched corners shown superimposed on each image.

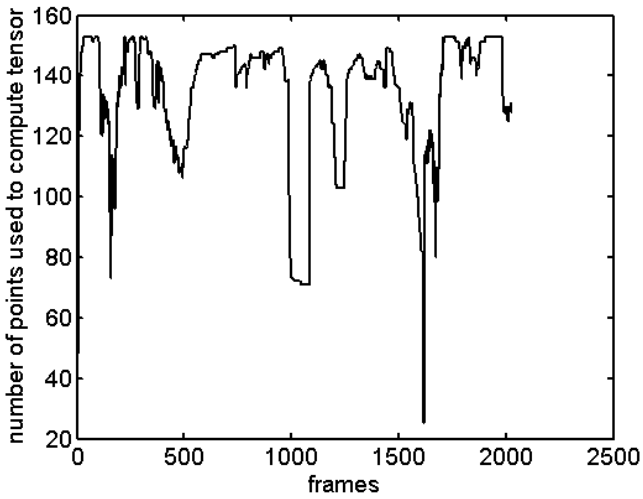



Figure 4.10: Plot of the number of tracked corners in the sequence Table. The tracker was re-initialized twice during tracking.

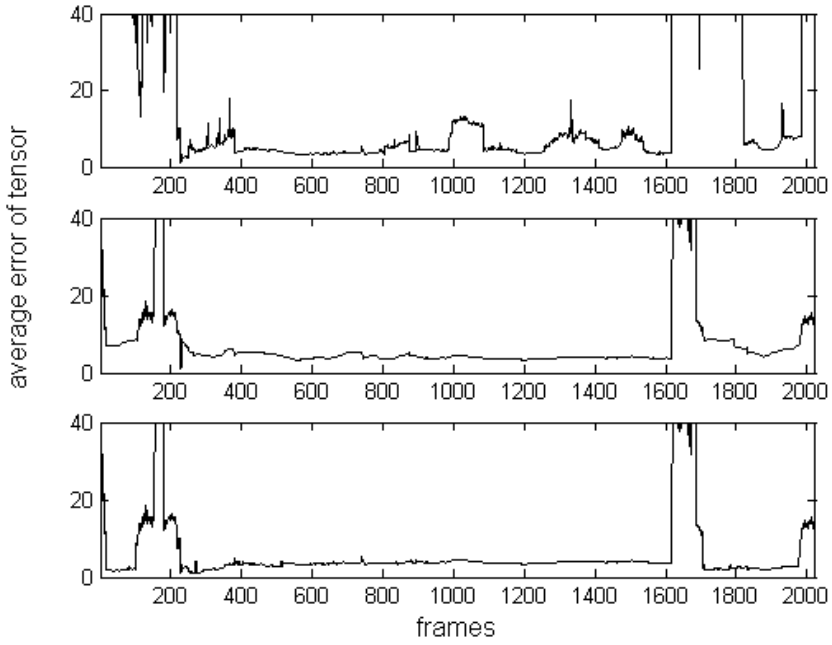



Figure 4.11: Tensors of 1749 frames in the sequence Table were obtained. Top: the tensors computed from all putative triplets; Middle: the improvements achieved by using the fixed $\mathbf{P_3}$; Bottom: the resulting tensors refined by x84


Figure 4.12: Resulting tensors were used to estimate the location of the CD envelop in the sequence Table.


Figure 4.13: Reference images prepared for video sequence Magazine (1359 frames). 123 points were matched across the three images.

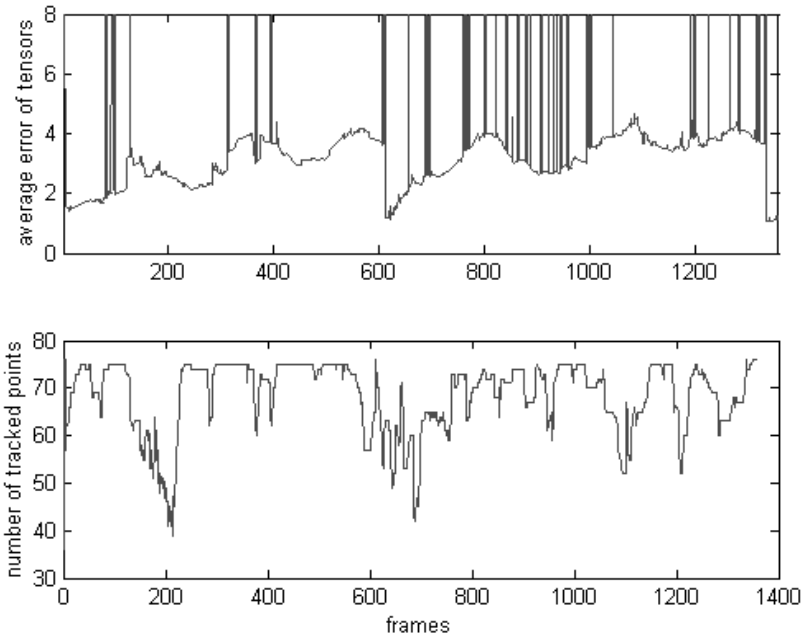



Figure 4.14: Results of the sequence Magazine. Top: tensors of 1302 frames were obtained. Bottom: the number of tracked points

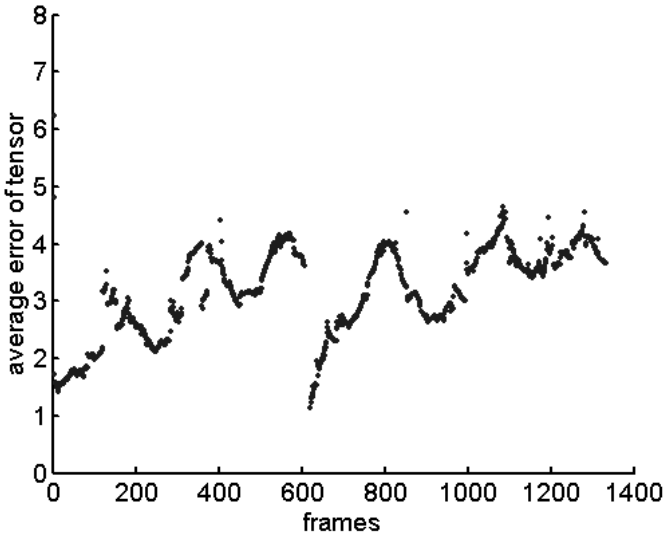



Figure 4.15: The resulting tensors in the sequence Magazine

# Chapter 5

# Applications to Augmented Reality

This chapter presents an application to Augmented Reality (AR) of the online tensor estimation approach proposed in Chapter 4. Augmented Reality is a technology to register virtual objects into the real world. Computer vision methods have proved to be a promising way to solve the registration problem. The AR system we propose computes the trifocal geometry of video frames and two reference images. It is able to augment live video captured by a camera moving in an unprepared scene without requiring camera calibration, without special markers and with no explicit 3D information.

## 5.1 Introduction

When the term of Augmented reality was created by Milgram in 1994, it was used to identify the systems that augment the real world with computer generated data. Now this technology has evolved into the "middle ground" between Virtual environment (completely synthetic) and telepresence (completely real) [2]. In the same way, Virtual Reality(VR) immerses a human into a processed environment. They are different in that VR let human enter a completely synthetic environment while AR combines the real world with a virtual one so that we still can feel our presence in the real world. The things added to the real world would be computer generated objects that don't

exist in the captured scene.

Expeditious improvement of the capability of realtime image processing, computer graphic systems and display technology enable Augmented reality achieve great progresses in many promising applications in recent years. The applications include image-guided surgery and ultrasound imaging, manufacturing and repair, annotation and visualization, robot path planning, entertainment and military training.

In theory, an AR system is required to have following characteristics: correctly merge virtual and real views of the world in real-time. In order to create a realistic looking new environment all virtual objects, including planar ones, must align with the physical world and stand firmly at their assigned positions as the user changes viewpoint. This requires that both virtual objects in the graphic coordinates system and the 3D observed scene in the figure 5.1 are transform into the world coordinate frame where they are blended together. The new views are then projected onto the image plane in the same way as real views. Accurate tracking of the viewing pose, relative to either the environment or the observed objects, is recognized as the key challenge for developing a successful AR system.

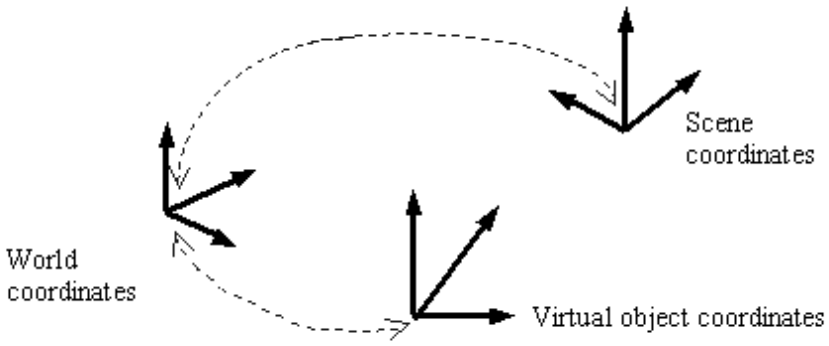



Figure 5.1: the related coordinates systems of AR

An AR system is usually an integration of an image acquisition component that can capture the real views, a computer graphic workstation, which is responsible for rendering virtual objects with real views and display equipment that show us the

mixed view. Figure 5.2 shows three typical AR systems, whose principle components, advantages and drawbacks are listed in table 5.1.

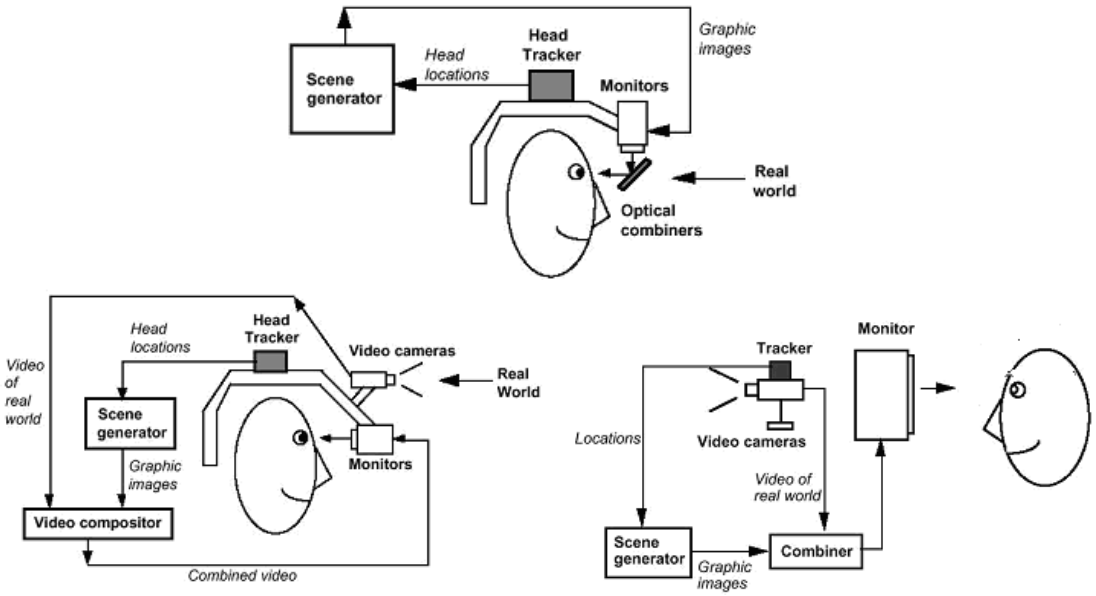



Figure 5.2: Conceptual diagrams of AR (from [2])Top figure: optical see-through AR system, Bottom: video-based HMD AR system and video-based monitor-display AR system

An optical see-through HMD AR is suitable for applications where realtime augmentations of the scene from user's point of view are required and the rendering accuracy is not highly demanding; such as annotation of a real scene, and guided diagnosis. When the difficulty of setting up emitters or sensors over the entire environment increases, video-based AR systems, using either HMD or monitor are necessary to embed virtual objects by the processing of the live video. They are more flexible for applications such as remote manipulation of robots, post-processing of a recorded video, CT/MRI images analysis. The rest of the chapter will discuss the application of our online tensor estimation in video-based AR systems.

| | optical see-through HMD AR | video-based HMD AR | video-based monitor-display AR |
|---|---|---|---|
| 3D Register | head tracker | head tracker | tracker or video processing |
| Real View Capture | human's eye | camera mounted on HMD | camera held by user or attached on other devices |
| Augmented View Displayer | displayed on monitor then reflected to the HMD glasses which is a partially transmissive optical combiner | monitor in HMD | monitor in front of user's eyes |
| Advantages | easiest to implement because of the optical combination. | more realistic views because the real and virtual views are blended in the graphic system | 1.no need to wear a HMD or any other display devices. 2.can be configured to see 3D vision of the environment with user wearing a pair of stereo glasses. |
| Drawbacks | 1.loss of light in the optical transmission or reflection. 2.images of virtual objects are overlaid on real views. | 1.cumbersome 2.the composite images are not stereographic | not suitable for some applications because the points of cameras'views are not agreed to the user's nature view |

Table 5.1: Comparison of three typical AR systems

## 5.2 Registration in video-based AR

In video-based AR, the real scene is seen by a video camera, whose (intrinsic) parameters and pose determine how the scene is perspectively projected onto a 2D image plane. The graphic processor needs this information so that it can project 3D virtual objects onto the image plane. In the area of computer graphics, this is called rendering. In popular graphic rendering tools, ex: OpenGL, OpenInventor and VTK, virtual objects are modelled in an object coordinate frame. A virtual camera needs to be created and manipulated in the graphic world frame.

The correct projection of virtual objects by the virtual camera requires: 1. camera calibration to obtain camera's intrinsic parameters (camera-to-image transformation); 2. estimation of camera's motion(world-to-camera transformation); 3. initialization of the virtual objects' positions in the real environment(virtual objects-to-world transformation). They are the key technical issues in the registration of video-based AR systems.

There are many way to achieve camera calibration including using calibration patterns and auto-calibration methods based on image observations. Tracking camera pose has been proven to be the most difficult problem in designing a AR system, because the camera pose must be estimated from 2D images.

Many computer vision approaches have been proposed for registration in video-based AR. They can be roughly classified into two catalogs: registration by tracking markers and registration by using the technology of structure and motion (SaM). The next subsection reviews these approaches.

### 5.2.1 Registration by markers tracking

Existing augmented reality approaches relying on special markers have achieved impressive results in video-based applications. The markers can be patterns or landmarks that are introduced in the scene, or naturally occurring features selected in the scene. Their images on every video frame define the transformation between the

observed environment and the current image. By applying the same transformation to arbitrary virtual object, the video sequence can be augmented with such object.

One of the most popular tool to perform scene augmentation is the ARToolkit of Human Interface Technology Lab at the University of Washington. It is a pattern-based technology, which can augment virtual objects onto predetermined black and white square markers [13, 1]. It has been used in [17, 27, 24, 25]. Similar methods can be found in [26]. The square pattern is assumed at X-Y plane in the graphic world coordinate frame so that the virtual camera's $\mathbf{3} \times \mathbf{4}$ projection matrix is degraded to a $\mathbf{3} \times \mathbf{3}$ matrix. The projection equation of a 3D point $\mathbf{X}$ on the pattern is

$$\lambda \mathbf{x} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \\ P_{41} & P_{42} & P_{43} & P_{44} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} |_{\mathbf{Z=0}} = \begin{bmatrix} P_{11} & P_{12} & P_{14} \\ P_{21} & P_{22} & P_{24} \\ P_{41} & P_{42} & P_{44} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{5.1}$$

It can also be expressed in terms of a 2D plane-to-plane homography $\mathbf{H}$.

$$\mathbf{x} = \mathbf{H}\mathbf{X} \tag{5.2}$$

Given the images of the four corners on the pattern, $\mathbf{H}$ can be obtained by solving a linear least-square equations of the corners.

$$\mathbf{H} = \lambda \mathbf{P_{3x3}} = \lambda \begin{bmatrix} f_u r_{11} & f_u r_{12} & f_u t_1 \\ f_v r_{21} & f_v r_{22} & f_v t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} \tag{5.3}$$

while $\mathbf{r_{ij}}$ and $\mathbf{t_i}$ are entries of the camera rotation matrix $\mathbf{R}$ and the translation matrix $\mathbf{T}$. Assume the virtual camera is a zero-skew camera, its intrinsic matrix $\mathbf{K}$ is composed by the focal length $\mathbf{f_u}$ and $\mathbf{f_v}$. Decomposing the homography gives $\mathbf{R}$, $\mathbf{T}$ and $\mathbf{K}$, which are used to set the virtual camera to embed virtual objects.

In Kutulakos' calibration-free system [18], four or more non-coplanar points are tracked along the video and an affine object representation is used to overlay virtual objects on an orthographic video stream without camera calibration. The basic idea

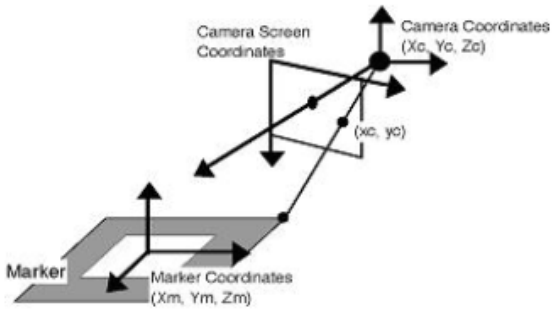



Figure 5.3: Illustration of marker coordinate frame

behind this approach is that real world and virtual objects are specified in an affine representation. The user is asked to select at least four non-coplanar points in the scene during system initialization. These points define an affine frame and its relationship with four non-coplanar points on the virtual objects. This approach's using the affine transformation to replace projective transformation has some drawbacks. For example, it is not able to handle quick or big motions of the camera. And the approximation results in increasing errors when the camera is moving close to the object. Thus this method is limited to the large object-to-camera distance applications.

The approaches that use markers or control points are impressive because (1) no offline camera calibration is required; (2) they have the capability of operating at frame rate. The common problems with them are (1) the scene has to be prepared by inserting markers in it, which can not be always feasible; (2) a minimum number of markers or control points are required being visible in every frame, otherwise the registration can not be performed; (3) The virtual object is fixed relative to the pattern or control points being tracked; (4) These methods are suitable to augment simple planar scenes since occlusion of virtual and real objects can not be handled without knowledge of the scene's 3D structure.

### 5.2.2 Registration using Structure-and-Motion

In comparison to marker-based approaches, registration based on the tracking of natural features provides a general solution to video augmentation, especially when a scene can not be prepared. The technology of structure-and-motion (SaM), which has been discussed in Section 4.1, is able to recover camera motion and the scene's 3D structure from video sequences. Minimal prerequisites to developing a AR system using SaM are: (1)registering the virtual object in one frame of the sequence, which usually needs a lot of user interaction; (2)calibrating the camera in order to obtain scene structure and camera motion in real world coordinate system.

Sequential processing approaches consist of the batch processing of the whole image sequence. Pairs or triplets of consecutive frames are matched and their corresponding view entities are computed in a chaining operation to find the projection matrices of the camera. Bundle adjustment or nonlinear optimization methods are applied on the whole sequence or subsequences to simultaneously obtain optimal camera matrices and 3D scene reconstruction. Using these approaches in augmented reality has the advantage that occlusions of real and virtual objects can be properly handled. However, processing sequences in this batch mode can not deal with live video.

An excellent AR system that consists of computing projective matrices of the camera from fundamental matrices or trifocal tensors of consecutive frames can be found in [46]. Another AR system using sequential processing technique is introduced in [20]. With the homographies induced by a 3D virtual plane being tracked in the image sequence, projective matrices of the camera are obtained in the same coordinate system without the need of scene reconstruction. In the end the scene is reconstructed separately from the AR system in order to speed up the processing for realtime augmentation.

Unlike approaches using batch techniques, keyframe-based approaches can estimate the camera pose of the current frame without using both the past and future

frames. An early approach to using keyframes for augmented reality consists in computing the 3D structure of features matched in the keyframes, then tracking these features in every other frame in order to obtain 2D-3D matches to compute projection matrices [8]. Unfortunately, the computational complexity of such reconstruction and auto-calibration approaches hinders their application to online processing.

In recent years, other approaches have been proposed to solve the camera pose problem without the need of scene construction. The time efficiency achieved by such approaches should make them feasible to develop a realtime AR system. For example, the AR system proposed in [6] can run at 10 fps on $320 \times 240$ images. The camera pose is obtained by using the epipolar constraints that exists between every video frame and two keyframes. An alternative method consists of linking the video frames with two keyframes into image triplets whose trifocal tensors are computed to transfer the location of the virtual object from the keyframes [5]. However, in these experiments virtual objects are attached to a calibration pattern, which simplified the problems of feature matching and tensor estimation. A recent system that performs very well in real time scene augmentation is the one proposed in [41]. In addition to keyframes, the system also needs a 3D model of the target object. It combines the strength of sequential processing in tracking and the keyframe technique to obtain a stable performance. The processing rate is about 15 fps for $320 \times 200$ images.

All these approaches share the limitation that virtual objects need to be registered in chosen frames or keyframes beforehand. Not only is the knowledge of the corresponding camera matrices of these images necessary, but also projections of the virtual objects on these images.

## 5.3 Proposed AR system

In this thesis we propose an original system to augment live videos or recorded sequences. It works in the context of a three-view system, which consists of a moving camera and three reference images of the scene. It combines the strengths of both

keyframe-based techniques and the ARToolkit technique to achieve a good performance in terms of robustness and speed. Augmentation is achieved through the use of a virtual square pattern that is transferred to the video. Every frame is registered to the graphic coordinate system individually. Therefore, the need for computation of camera pose and scene reconstruction is avoided, and camera calibration is no longer required a-priori. The online estimation of the trifocal tensor, proposed in Chapter 4, produces the trinocular geometry relating every video frame to two of the reference images. The tensors updated online over the video stream provides all the necessary information for embedding the virtual objects. In this manner an online processing system can be easily built and provide a stable performance.

In the off-line initialization stage, three reference images are captured from different viewpoints with a square pattern temporarily placed in the scene. Then, the pattern is withdrawn, and as the camera is moving freely inside the scene, image features taken from an initial set of corresponding triplets detected on the three reference images are tracked across the video sequence. The trifocal tensor associated with each frame and two of the reference images is then estimated in realtime. Using these computed tensors, the square pattern, which was visible in the reference images, can be transferred to the moving frames. Virtual objects are then placed onto the video, by feeding this virtual pattern to the ARToolkit.

Figure 5.4 shows a schematic overview of the whole procedure. Three reference images,$\mathbf{V}_1$, $\mathbf{V}_2$, $\mathbf{V}_3$, are shown in the top three figures. Matched corners from the initial set of triplets are shown superimposed on each image. A blank paper was selected on the reference images by manually selecting its four corners at system initialization time. For each video frame, the bottom-left image, the trifocal tensor is estimated from tracked corners (blank circles). All matches of the view pair ($\mathbf{V}_1$,$\mathbf{V}_3$) can be transferred to this frame. The transferred points (white dots) are used for updating the tracker's point set. Transfer of the four corners on the blank paper gives a virtual plane location, shown enclosed by the re-projected black rectangle. The transferred plane, upon which a virtual object (teapot) can be added, agrees


Figure 5.4: Embedding and rendering procedure

with the real paper that remained on purpose when the video started.

The advantage of the system we developed over the ARToolkit systems is that once the user completes the initialization step the square marker is not needed any more. There are two benefits brought by the use of an *invisible* pattern. First, no artificial object other than the embedded virtual objects appears in the processed videos since the augmentation completely relies on the nature features in the scene. Second, virtual objects can be added anywhere, including on untextured surfaces (see Figure 5.4), as long as the surrounding regions have sufficient features.

Our system is competitive with the existing keyframe-based online AR systems because camera pose estimation, the most time consuming part, is not required. In addition, the development of our system is simplified by the fact that it does not require camera calibration and 3D model of the target object. It is important to point out that our system is an improvement over [6] that takes view pairs and fundamental matrices. The reason is that the trinocular geometry is exploited to provide a more powerful disambiguation constraint than the epipolar constraints allow. This is because in a view triplet, image coordinates in a third view are completely determined, given matches in the other two views, whereas image positions are only restricted to a line by the epipolar geometry of image pairs. The AR system in [5] has much in common with ours, computing trifocal tensors of video frames and using the property of trifocal transfer in embedding. However there are several important differences: first a semi-automatical camera calibration using two calibration patterns is required to place virtual objects in the first two images in the sequence so that the registration plane of the virtual objects can be anywhere in the scene. In our case the same objective is achieve by a simpler method without camera calibration. Second, the problem of feature tracking and tensor estimation is simplified by keeping a calibration pattern in the scene. Our system based on a complete framework is a general solution to the processing of long image sequences.

The main requirement of this system is a consequence of the feature-based approach we have taken: the scene must contain a sufficient number of features, and

that these features must be well distributed. The proposed methodology works effectively as long as the moving camera captures views that remain within the visual hull spanned by the reference images. The performance of the system can be improved by adding more reference views.

Our system is also flexible in the sense that for simple applications, such as video labelling or notation, the line segments and vertices of the rendered virtual objects may be transferred from the fixed reference views to the moving camera view using the trifocal tensor that we compute at each frame. For a rudimentary polyhedral object, this transfer process is quick and direct. Even for a complex object, the transfer of its rendered triangulation is still straightforward, as long as its rendering in the two fixed reference images is available.

Our AR system consists of two parts: embedding and rendering. The steps of initializing the virtual objects in the reference images and determining their registration plane (the square pattern) in video frames are included in the embedding part. The details about embedding are given in section 5.3.1. Rendering of video frames will be discussed in section 5.3.2. Section 5.4 shows the experimental results of our system.

### 5.3.1 Embedding of virtual objects

First, our AR system needs an initialization procedure to prepare for live video augmentation. It is carried out in following steps:

1. Capture three images of the scene of interest. These will be the reference images and the associated match set.

2. Choose one reference image onto which the virtual object will be inserted. Specify the locations $(\mathbf{x^b_i}, \mathbf{y^b_i}), \mathbf{i} = \mathbf{1}, ..., \mathbf{4}$ of the four corners of the virtual square pattern that will define the reference frame for rendering.

3. Apply the ARToolkit method to insert the virtual object on the chosen image.

This means that $(\mathbf{x^b_i}, \mathbf{y^b_i}), \mathbf{i} = \mathbf{1}, ..., \mathbf{4}$ are feed into Projection equations 5.1 to compute the homography. Set the virtual camera and project virtual objects on the image. Adjust the pose and scale of the virtual objects to achieve the desired visual effect.

4. Specify the corresponding locations $(\mathbf{x'^b_i}, \mathbf{y'^b_i})$ and $(\mathbf{x''^b_i}, \mathbf{y''^b_i}), \mathbf{i} = \mathbf{1}, ..., \mathbf{4}$ of the square pattern in one of the other two reference images with the help of epipole geometry.

Once the initialization is completed, the camera starts moving from a location close to one reference image. Our online tensor estimation approach is then applied as explained in the previous chapter. Using the online estimated tensors the virtual square pattern is located in the current frame by transferring its four corners from reference images $(\mathbf{x'^b_i}, \mathbf{y'^b_i}), (\mathbf{x''^b_i}, \mathbf{y''^b_i}) \rightarrow (\widetilde{\mathbf{x}}^\mathbf{b}_\mathbf{i}, \widetilde{\mathbf{y}}^\mathbf{b}_\mathbf{i}), \mathbf{i} = \mathbf{1}, ..., \mathbf{4}$. The transferred corners, $(\widetilde{\mathbf{x}}^\mathbf{b}_\mathbf{i}, \widetilde{\mathbf{y}}^\mathbf{b}_\mathbf{i})$, on each frame yields a new homography.

### 5.3.2 Rendering

Once the plane on the reference images is transferred on the $i^{th}$ video frame using the computed trifocal tensor, rendering of this frame is accomplished by the ARToolkit [27]. The homography from the virtual square pattern to the $XY$ plane of the virtual objects is computed. It gives the camera matrix $\mathbf{K}[\mathbf{R}|\mathbf{T}]$ of the virtual camera that projects virtual objects onto the image plane where video frames reside (refer to equation 5.3). Assuming a constant focal length for the virtual camera, a transformation matrix $[\mathbf{R}|\mathbf{T}]$ is then obtained and sent to an openGL graphic server.

Though the online tensor estimation is free of error accumulation, the resulting tensors may produce 'jitter' because they are computed from different sets of matches appearing in images during the tracking. The 'jittering' of the virtual objects is generally small but sometimes noticeable, especially when the camera looks at the

edges of a scene containing multiple planes. One way to reduce this jittering effect would be to use the previous frames to achieve smoother motion. In our system under the assumption that the camera is moving slowly, a third-order Kalman filter is applied on three translations of the transformation matrices.

The camera translations are smoothed by updating estimate state vector $(\mathbf{x}, \mathbf{x}', \mathbf{x}'')$ and predicate error vector $(\mathbf{P}, \mathbf{P}', \mathbf{P}'')$ at every frame. Since estimating the general noise characteristics of the feature tracker and the camera motion is not trivial, the state vector is simply updated by

$$\widehat{\mathbf{x_k}^-} = \mathbf{A}\widehat{\mathbf{x_{k-1}}} + \mathbf{B}\mu_{\mathbf{k-1}}, \quad \widehat{\mathbf{x_k}} = \widehat{\mathbf{x_k}^-} + \mathbf{K}\mathbf{P_k} \tag{5.4}$$

The constant matrix $\mathbf{A}$ relates the state at the previous time $\mathbf{k-1}$ to the state at the current time step $\mathbf{k}$. The matrix $\mathbf{B}$ relates the optional control input $\mu$ to the state $\mathbf{x}$. The first equation gives the prior state estimate $\widehat{\mathbf{x_k}^-}$ as an prediction of the current state. Its difference to the observation is the estimate error covariance $\mathbf{P_k}$, according to which the Kalman filter gain $\mathbf{K}$ the posterior state $\widehat{\mathbf{x_k}}$ is obtained as new camera translations for rendering.

The result of tensor estimation for each frame is classified according to whether a tensor can be obtained to correctly represent the trifocal geometry of the current frame. With good tensors the pattern corners are transferred and then used in rendering. The translations extract from the homographies are then smoothed by the Kalman filter. When there is a lack of sufficient matches or large number of strong mismatches the result is that no tensor is estimated or a tensor with poor quality is produced. The frames where no patterns are transferred to are "holes" left in the augmented videos. To give user a real-life impression of the augmented results, it is important to fill these "holes" where the virtual objects looks suddenly disappear. By using a Kalman filter they can be filled with predicated translations based on previous motions. This leads to a smoother motion of the virtual objects without abrupt jumps in the augmented video and can therefore give more appealing results. Figure 5.5 and5.6 illustrate that smooth motions of the virtual object can be achieved by Kalman filter in the sequence Magazine (its three reference images are given in

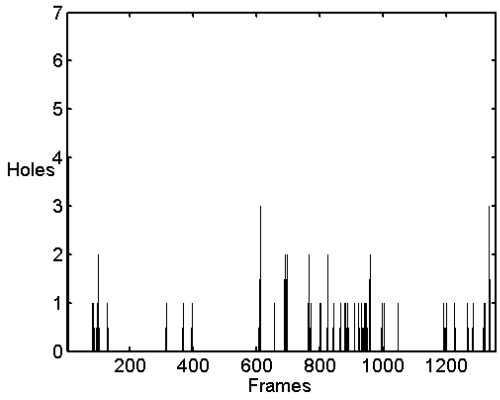



Figure 5.5: Plot of 'holes' that were filled with predicated motions in sequence Magazine.

Figure 4.6). Some frames from the augmented sequence are shown in Figure 5.7.

## 5.4 Experiments

Our system runs on a Pentium 2.0$G$ desktop PC with a web camera of image resolution $320 \times 240$ pixels. The online tensor estimation can carry out at the speed of 14fps and the entire augmentation system can perform at about 10fps.

As observed from Figure 5.7 the augmentation was not affected by the changes of the viewpoint and position of the camera, even though the invisible pattern was sometimes partially out of images. In our system the transferred pattern always leads to a homography regardless of its image position. This is another capability of our system that ARToolkit doesn't possess, since ARToolkit fails when any corner of the physically existed pattern is absent.

Figure 5.8and 5.9 show reference images and rendering results of an example sequence Poster. A square pattern was pasted on the wall when three reference frames were captured. The transferred patterns are shown superimposed on the video frames. From these patterns, the homographies are computed which map a logo image on the wall. The logo is augmented on the wall and is not lost, even though on some frames, the transferred pattern is almost out of view. Some of the

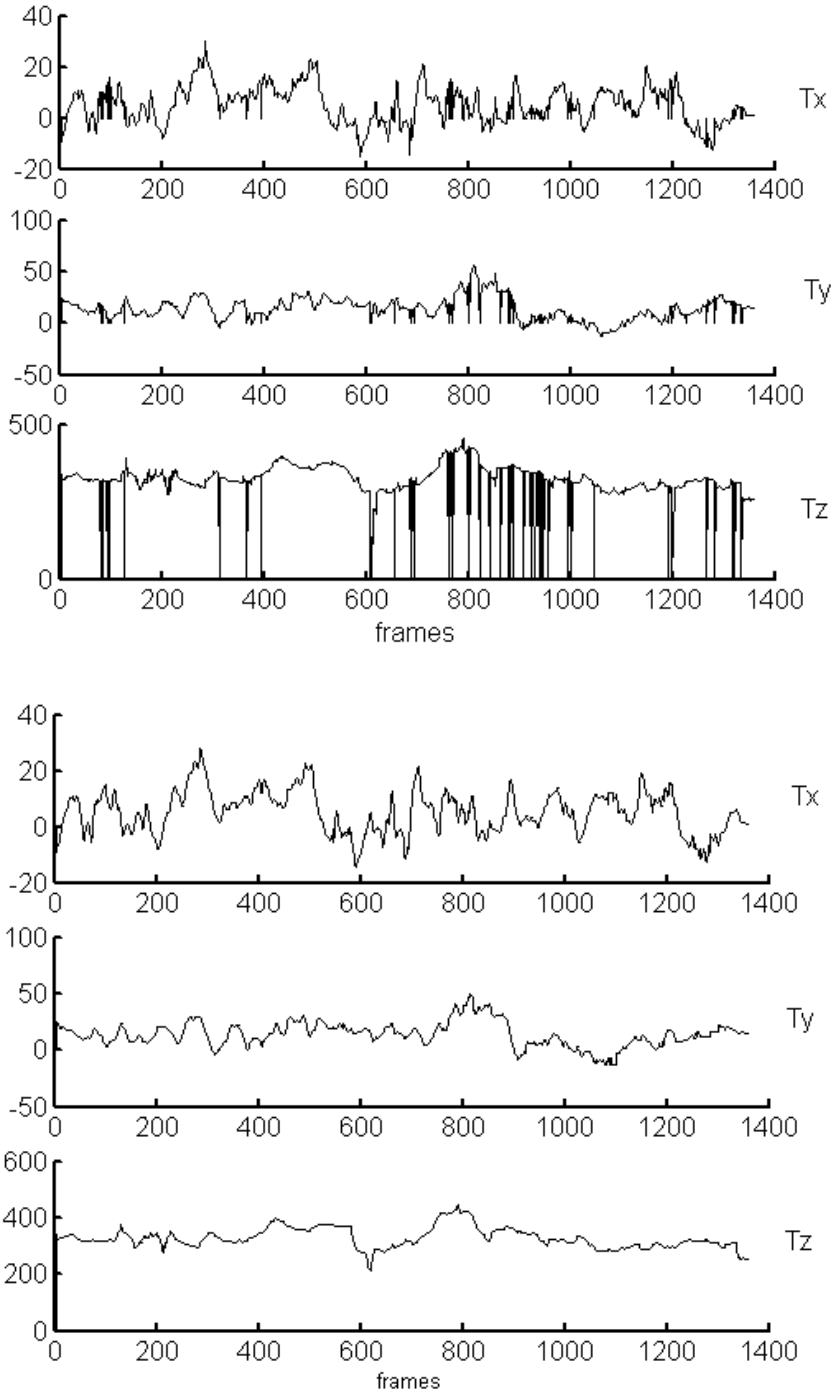



Figure 5.6: The computed X, Y, Z-translations through the sequence Magazine before and after smoothed by Kalman filter.


Figure 5.7: Sequence Magazine was augmented by a virtual teapot.

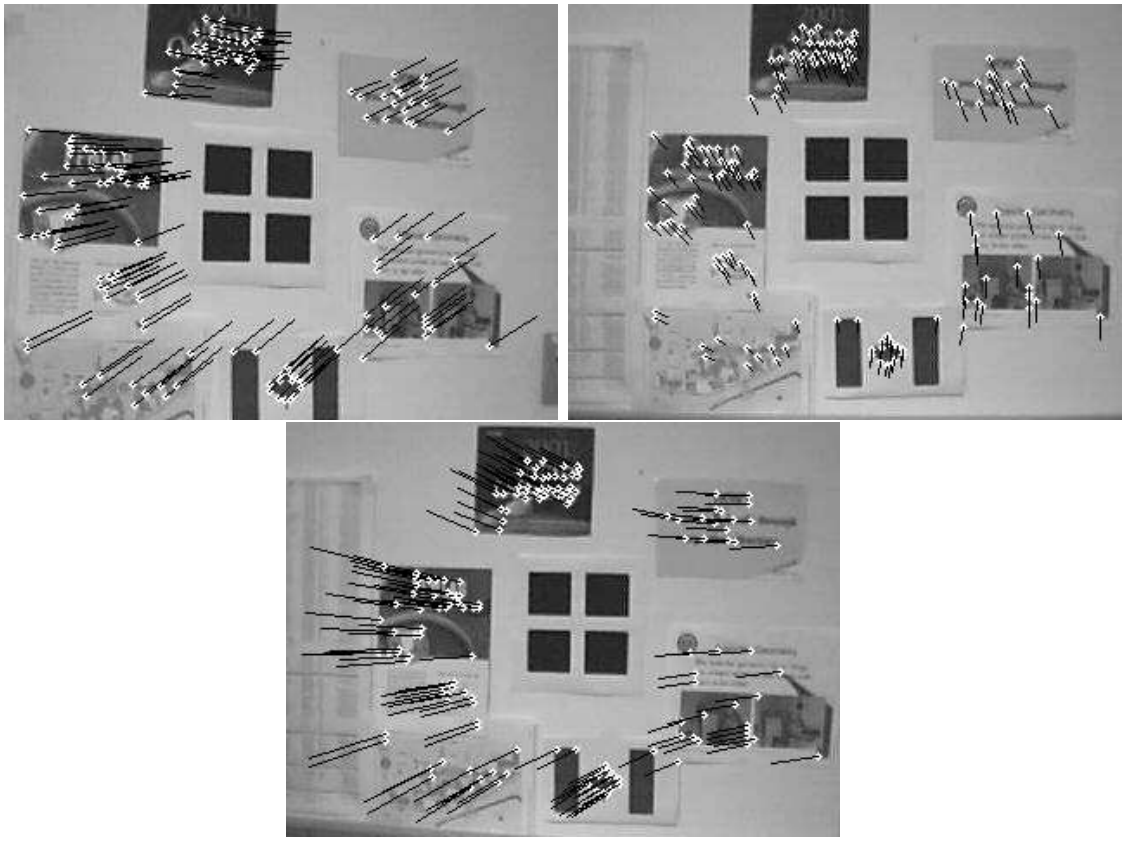


Figure 5.8: Reference images of the sequence Poster

video sequences are available at: http://site.uottawa.ca/ jiali/work.html.


Figure 5.9: Augmentation results of the video sequence Poster. The frames on which transferred patterns are superimposed are shown in the left column. They are augmented by a DirectX logo as shown in the right column.

# Chapter 6

# Summary

## 6.1 Conclusions

The goal of this thesis is to find robust estimators for the trifocal tensor that can be used in realtime video processing. First different tensor estimation methods were implemented and tested on real and synthetic triplets of points. Though RANSAC-based methods are more robust against mismatches than linear and algebraic minimization method, they are not suitable for online processing because of the computational cost.

The first contribution of this thesis, a three-view framework, utilizes two reference images to continuously construct correspondence triplets of the moving camera frames, from which trifocal tensors are estimated. Efforts are invested into eliminating outliers in putative triplet sets and estimating tensors efficiently. The experiment results demonstrated that the framework has the capability to compute the trifocal geometry of the moving camera at about 14 fps without any requirement other than three reference images. When the camera is inside the view spanned by the reference images, the estimated tensors are accurate enough to have a transfer error of under three pixels.

Our second contribution is an augmented reality application built upon this framework. By employing the online estimated trifocal geometry, videos can be augmented

without computing the camera pose, which has been recognized as the biggest problem in natural feature based AR systems. Its stable performance under difference camera motions were proven by augmentation results on several long sequences.

## 6.2 Future work

There are several issues we plan to address in future work. In online tensor estimation the procedure of identifying outliers takes the most processing time and we need to find ways to reduce this time. Another critical problem is how to obtain a good trifocal tensor even when a higher fraction of mismatches exists. To prevent failures some other methods have to be studied.

Trifocal tensors are evaluated on identified good matches in terms of transfer error. However, when only a small number of inliers exist the trifocal geometry of the whole scene may not be correctly represented by the computed tensor. How to exam the quality of tensors under such conditions is also an important unsolved question.

As mentioned in Section 4.2 our three-view framework for online tensor estimation can be extended using more reference images to offer further freedom of the moving camera. How to register all available reference images and correctly determine the two closest to the current video frame is another key problem that needs to be solved.

In the long term, we also plan to utilize the proposed online tensor estimation in applications of camera pose estimation and 3D modelling from videos.

# Bibliography


[1] http://www.hitl.washington.edu/artoolkit/.

[2] R. Azuma. A survey of augmented reality. *Teleoperators and Virtual Environments*, 4:355–385, 1997.

[3] S. A. B. Rousso, A. Shashua, and S. Peleg. Robust recovery of camera rotation from three frames. pages 851–856, 1996.

[4] P. Beardsley, P. Torr, and A. Zisserman. 3d model acquisition from extended image sequences. In *European Conference on Computer Vision*, pages 683–695, 1996.

[5] B. Boufama and A. Habed. Registration and tracking in the context of augmented reality. *ICGST International Journal on Graphics, Vision and Image Processing*, 2, 2005.

[6] K. Chia, A. Cheok, and S. Prince. Online 6 dof augmented reality registration from natural features. In *Proc. International Symposium on Mixed and Augmented Reality(ISMAR)*, 2002.

[7] K. Connor and I. Reid. Novel view specification and synthesis.

[8] K. Cornelis, M. Pollefeys, M. Vergauwen, and L. V. Gool. Augmented reality from uncalibrated video sequences. In *3D Structure from Images - SMILE 2000, Lecture Notes in Computer Science*, volume 2018, pages 144–160, 2001.

[9] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM*, 24:381–395, 1981.

[10] A. Fitzgibbon and A. Zisserman. Automatic camera recovery for closed or open image sequences. In *ECCV 1998*, pages 1261–1269.

[11] F. Fraundorfer. Robust estimation of the trifocal tensor and it's application to image matching. Master's thesis, Graz University of Technology, 2001.

[12] A. Fusiello, E. Trucco, T. Tommasini, and V. Roberto. Improving feature tracking with robust statistics. In *Pattern Analysis and Applications*, volume 2, pages 312–320, 1999.

[13] M. B. H. Kato and I. Poupyrev. *ARToolKit User Manual*. Human Interface Technology Lab, University of Washington, 2.33 edition, November 2000.

[14] R. Hartley and R. Vidal. The multibody trifocal tensor: Motion segmentation from 3 perspective views. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[15] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[16] R. L. J. Li and G. Roth. Online estimation of trifocal tensors for augmenting live video. In *The Third IEEE and ACM international Symposium on Mixed and Augmented Reality(ISMAR)2004*, pages 182–190, 2004.

[17] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR)*, pages 85–94, 1999.

[18] K. Kutulakos and J. Vallino. Calibration-free augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 4:1–20, 1998.

[19] S. Laveau and O. Faugeras. scene representation as a collection of images. pages 689–691, 1994.

[20] M. Lourakis and A. Argyros. Vision-based camera motion recovery for augmented reality. In *Computer Graphics International Conference*, pages 569–576, 2004.

[21] L.Quan. Invariants of six points and projective reconstruction from three uncalibrated images. *IEEE Transactions on Pattern Analysis and Machine Intelligence(TPAMI)*, 17:34–46, 1995.

[22] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *Proc. of 7th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 674–679, 1981.

[23] A. Z. M. Armstrong and R. Hartley. Self-calibration from image triplets. In *ECCV 1996*, pages 3–16.

[24] H. K. M. Billinghurst and I. Poupyrev. The magicbook moving seamlessly between reality and virtuality. *IEEE Computer Graphics and Applications*, 2001.

[25] H. K. M. Billinghurst and I. Poupyrev. The magicbook: A transitional ar interface. *Computer & Graphics*, pages 745–753, 2001.

[26] A. Z. M. Jethwa and A. Fitzgibbon. Real-time panoramic mosaics and augmented reality. In *BMVC 1998*, pages 852–862.

[27] S. Malik, C. McDonald, and G. Roth. Hand tracking for interactive pattern-based augmented reality. In *Proceedings of IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2002.

[28] H. Mayer. Estimation and view synthesis with the trifocal tensor. page A: 211, 2002.

[29] D. Nister. Reconstruction from uncalibrated sequences with a hierarchy of trifocal tensors. In *ECCV00*, volume 1, pages 649–663, 2000.

[30] D. Nister. Preemptive ransac for live structure and motion estimation. In *Proc. ICCV03*, pages 199–206, 2003.

[31] O.Chum and J.Matas. Randomized ransac with $t_{d,d}$ test. In *Proc. BMVC'02*, pages 448–457, 2002.

[32] J. O.Chum and J. Kittler. Locally optimized ransac. In *DAGM03*, pages 236–243, 2003.

[33] S. Pollard and M. Pilu. View synthesis by trinocular edge matching and transfer. 2:770–779, 1998.

[34] K. R. Klette and A.Koschan. *Computer Vision: three-dimensional data from images*. Sprigner, 1996.

[35] S. Ravela, B. Draper, J. Lim, and R. Weiss. Adaptive tracking and model registration across distinct aspects. In *IEEE International Conference on Intelligent Robots and Systems(IROS)*, 1995.

[36] G. Roth and A. Whitehead. Using projective vision to find camera positions in an image sequence. In *Vision Interface*, 2000.

[37] A. S. S. Avidan. Novel view synthesis by cascading trilinear tensors. *IEEE Transactions on Visualization and Computer Graphics(TVCG)*, 4(4):293–306, 1998.

[38] Z. Sun and A. Tekalp. Trifocal motion modeling for object-based video compression and manipulation. *IEEE Transactions on circuits and system for video technology*, 8, 1998.

[39] P. Torr. *Motion segmentation and outlier detection*. PhD thesis, Dept. of Engineering Science, University of Oxford, 1995.

[40] P. Torr and A. Zimmerman. Robust parametrization and computation of the trifocal tensor. *Image and Vision Computing*, 15, 1997.

[41] L. Vacchetti, V. Lepetit, and P. Fua. Fusing online and offline information for stable 3d tracking in real-time. In *Proc. International Conference on Computer Vision and Pattern Recognition*, 2003.

[42] J. Vigueras-Gomez, M. Berger, and G. Simon. Iterative multi-planar camera calibration : Improving stability using model selection. In *Vision, Video and Graphics(VVG)'03*, 2003.

[43] E. Vincent and R. Laganière. Matching feature points for telerobotics. In *Proc. 1st International Workshop on Haptic Audio Video Environments and their Applications*, pages 13–18, 2002.

[44] K. S. W. Lin and R. Sharma. Augmented reality with occlusion rendering using background foreground segmentation and trifocal tensors. In *Proceedings of the IEEE Conference on Multimedia and Exposition*, volume 2.

[45] J. Xiao and M. Shah. From images to video: View morphing of three images.

[46] A. Zisserman, A. Fitzgibbon, and G. Cross. Vhs to vrml: 3d graphical models from video sequences. In *IEEE International Conference on Multimedia and Systems*, volume 1, pages 51–57, 1999.