Applying Image Processing Techniques to Pose Estimation and View Synthesis

FUNG Yiu-fai Phineas

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Computer Science and Engineering

June 1999

Acknowledgements

This thesis is dedicated to my parents, Hon-ming and Alision Fung, who inspired me to pursue a master degree, and for their continuous support throughout my studies.

I wish to express my deepest gratitude to Kin-hong Wong, my project supervisor, and Siu-hang Or, for their guidance and support. Their enlighting discussions and advice have been an invaluable resource for me. I would also like to thank other members of my thesis committee, Isaac Chan, Laiwan Chan and Hanqiu Sun for their comments and insights on this work.

In the course of my graduate studies, I have benefited from the interactions with many warm people around me. I would especially like to thank Sam Lee, Jackie Lao and Po-shan Kam for their endless encouragement and enthusiasm, and for making these several years truly enjoyable.

Abstract

Computer vision has been proved to be a very difficult problem after some thirty years of continuous research. An image or video sequence usually contains more information than necessary, but computer vision researchers are more interested in finding the relative positions and orientations of an object in the three-dimensional (3D) space during a specific period of time. This is known as the pose estimation problem, and is applicable to many areas like robot navigation or camera calibration. In this report, we propose an iterative algorithm for the Perspective-3-Point (P3P) problems, together with a novel technique for panoramic walkthrough based on the P3P calculations.

Pose estimation is difficult as spatial data is encoded inside two-dimensional (2D) images during the nonlinear camera projection. Complex formulae are often involved to recover the 3D pose information, leading inevitably to numerical instability and multiple solutions. Much research work on this problem has been reported in the literature. Among the many iterative solutions to P3P problems, most of which make use of weak-perspective projection to reduce the three unknown depths into image plane coordinates. Then various mathematical techniques are used to solve the quartic or cubic equations derived. On the other hand, our algorithm recovers the motion parameters (rotation $\mathbf{R}$ and translation $\mathbf{t}$) of an object, given its 3D dimensions and subsequent image point correspondences under full perspective projection. It is broken down into two stages: (1) depth recovery by the Gauss-Newton method, in which depth estimates are iteratively refined under a least-square minimization paradigm; and (2) 3D-to-3D pose calculation, in which two 3D point sets at time t and $t+1$ respectively are used to find $\mathbf{R}$ and $\mathbf{t}$. Additional geometrical constraints are also incorporated to reduce multiple solutions inherent in P3P problems, and improve the stability of the estimation process when measurements are noisy.

On the other hand, image-based rendering is becoming a popular way in the construction of virtual environments in computer graphics application. Instead of build-

ing a complete 3D environment model, a collection of images is used to synthesize the scene while supporting virtual camera motions. However, most existing methods navigate by jumping from one hot-spot to another instantaneously, and provide users with little sense of sense immersion. Our method generates realistic intermediate frames to simulate a smooth transition from one mosaic node to another, with minimal sacrifice in image quality and execution speed. Specifically, given a pair of images about an object from two adjacent panoramas, and their pose estimates from the algorithm above, we show that it is possible to uniquely synthesize perceptually correct intermediate views, along the line segment between the panoramic optical centers. This greatly simplifies the efforts in locating control points for view synthesis, and helps to create realistic walkthrough video sequences with sparse correspondences and uncalibrated cameras.

Our pose estimation algorithm has a distinctive property of preserving object rigidity, and its performance advantage is verified empirically through detailed comparisons with the closed-form Fischler and Bolles' algorithm. Both real panoramic images and synthetic data have been used to verify our method, showing that our proposed algorithms can effectively reduce multiple solutions inherent in P3P solutions, and synthesize perceptually correct intermediate views. The computation time of our P3P algorithm in ANSI C implementation for 1000 synthetic motion instants on a UltraSparc 1/170 workstation is 0.11 sec. Knowing that the Fischler and Bolles' approach is among the most efficient algorithm for P3P problems, our algorithm is stable and efficient in view of its iterative nature (only about six iterations are required for convergency in most of the cases). Our P3P method also has higher flexibility in tracking complicated object like articulated ones, because it requires only three point correspondences for every motion instant. In addition, our panoramic viewer in Visual C++ implementation takes about 15 seconds to synthesize a 10-frame movie sequence at resolution 500×300 pixels on a Pentium II 300 PC. Although there are noticeable delays in generating a sequence on-the-fly, the resulting image quality is much superior to the case of digital zooming (pixel enlargement). This demonstrates the value of our algorithms and justifies them for the cost. Our next target is to relax the requirements of *apriori* 3D feature dimensions, and investigate if other computer vision tools, e.g. epipolar geometry, can be used to make the walkthrough process faster and more robust.

論 文 摘 要

計算機視覺發展至今，大部份的研究均集中討論物件位置測定方法 (pose estimation)，即如何從二維影像中得出物件於三維空間的位置及姿態。此技術可被廣泛應用於機器人空間瀏覽或攝影機變數測定等方面。

當物件的三維空間資料被投射爲二維影像時，涉及非線性的攝影機投射效應。因此，若要從二維影像推算三維空間資料，我們便需要進行異常繁複的運算，及處理數態不穩 (numerical instability) 或多重根 (multiple solutions) 的問題。一直以來，不少研究工作皆針對這方面作改善，其中三點透視法 (Perspective-3-Point (P3P)) 的研究更普遍採用弱透視投射法 (weak-perspective projection) 來評估物件大約深度，接著再以不同方法化解那些代數算式。在這報告中，我們將集中探討 P3P 問題，並提出新的計算方法以改良現有技術。我們會進一步把這新計算法套用於全景圖瀏覽 (panoramic walkthrough) 的問題上。

然而，在計算物件的外在運動變數 (external motion parameters) 時，除了已知的物件幾何特性及其對應點 (point correspondences) 外，我們首先使用全透視投射法 (full perspective projection) 及 Gauss-Newton 反覆計算法，求物件大約深度。接著利用三維位置還原法 (3D-3D pose recovery) 推算物件的三維空間資料。再套用物件的幾何特性以篩選及摒棄在運算過程中可能遇到的不合理數據或多重根。

另一方面，影像爲本演示法 (image-based rendering) 在計算機圖像處理上日趨普及。研究員們大多以結集一連串的二維影像組成全景圖，再配合虛擬攝影機運動以製造虛擬現實效果。但大部份現有的技術往往因影像間變化太大而予人不實在的感覺。故此我們提出以影像變形來模擬物件移動過程，配合先前提及的 P3P 物件位置測定方法，製造出準確且自然的虛擬運動。

爲了驗證我們提出的方案，我們利用真實影像及人造數據，並以 Fischler and Bolles 的方法爲對照。結果發現我們的方案較穩定，且能準確地模擬物件移動過程，效果遠較數碼變焦或傳統計算機圖像方案爲佳。將來我們希望能放寬對物件幾何特性及其對應點的要求，及研究引用其他計算機視覺工具到全景圖瀏覽的問題上。

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since 1970s, scientists have been working on the possibility of bringing intelligence into computers so that they can visualize as well as understand the environment around them. Although it seems to be easy at first glance, it has been proved to be a very difficult problem after some thirty years of research. An image or image sequence usually contains more information than necessary. We would like to extract as much information as possible at minimal effort. The basic tokens for consideration are *points*, *lines* or *planes*. However, as both lines and planes from natural scenes can hardly be expressed in simple algebraic forms, and indeed they are not always available in real-life configurations, more research efforts have been focused on the analysis of points.

In addition, the computer vision community is also very much interested in finding the *pose* of an object from its subsequent two-dimensional (2D) images. To be exact, the pose of an object refers to its relative position and orientation in the three-dimensional (3D) space at a particular time instant, with respect to the camera reference frame. Theoretically speaking, any motion of a rigid body can be represented as a concatenation of rotation followed by translation. Thus by estimating the poses of an object in consecutive image frames, one could recover the extrinsic motion parameters (rotation and translation) of that object during that specific period of time. However, very often there are too many possible combinations in real-life situations that it is hard to locate the matches of a particular feature point throughout a sequence of consecutive image frames. This is known as the *correspondence problem* in computer vision literature [56], and is already an exceedingly complicated problem by itself. Furthermore, as spatial data is encoded inside 2D arrays of picture ele-

ments (pixels) during the nonlinear camera projection, depth information is usually lost, and complex formulae are often involved to recover the 3D pose information. On the other hand, efficiency is another crucial factor in computer vision application. For example, many automatic assembly or traffic monitoring systems require a refresh rate of 30 frames per second to capture velocity or positional changes. The time allowed between successive updates thus places a stringent requirement on the estimation process.

A number of different schemes have been proposed, but regardless of the method chosen, the vision problem is generally divided into three processes: *acquisition*, *analysis* and *recognition*. In the image acquisition stage, spatial light intensity information is converted by video cameras or other imaging devices into machine formats. A beam of laser or polarized light may be directed towards an object under investigation to find out its exact 3D positions, or that a camera quickly saccades to follow an object of interest during a certain time interval. The analysis stage extracts useful information from 2D or 3D images. In most application the first step is to recover depth information from 2D image data to form a $2\frac{1}{2}D$ sketch [56]. Subsequent analysis may include object segmentation, motion analysis, etc. The final stage makes use of the information in a knowledge database (which stores the image characteristics of many objects) and the extracted information to find the semantics of a scene, e.g. the number of objects and their properties. This step often involves target localization and identification, as well as a rough initial estimation of poses, velocities and possibly some other state-variables of interest. This phase is particularly hard in active vision systems, owing to the need to saccade towards the moving target.

In this dissertation, we wish to develop motion-analysis and view synthesis methodologies that address a number of goals:

1. Measurability: Sufficient information to compute the transformation must be automatically or semi-automatically extracted from the basis images.

2. Correctness: Each synthesized image should be perceptually correct, i.e., it should correspond to what the real scene would look like as a result of the specified scene transformation, and is sophisticated enough to precisely model viewpoint changes and other 3D operations.

3. Robustness: The techniques developed for image-based scene transformations should be robust and general enough to handle complex real-world scenes and

photographs.

To make life easier, we focus on the 3D-to-2D pose estimation problem, with given point correspondences (established in advance with substantial manual efforts). A nonlinear procedure to recover depth values from image correspondences and camera parameters is derived, with several geometrical constraints to further improve the quality of depth estimates. A hierarchical organization of depth values is then used to calculate the depth map of a novel view, and by reprojecting onto original views gives the intensity values of each pixel in the novel view. In addition, we propose the use of image-based operations as a general framework for visualizing and manipulating 3D scene transformations or viewpoint changes. They have a key advantage that they operate on a set of basis images and do not require precise 3D modeling of the scene. An alternative approach is discussed in later chapters to determine how an image would change in response to a particular physical movement of the object or camera in the 3D space. In such a case, these changes can be modeled as a mapping or a warp from pixels in one or more basis views to pixels in a new synthetic view of the same scene. By categorizing all such mappings, we can devise a mechanism for synthesizing novel views to simulate transformations by warping a set of basis images.

1.1 Model-based Pose Estimation

Model-based motion analysis refers to the recovery of pose information of a rigid body (which 3D description is known *apriori*) from subsequent image projections of its corresponding feature points. Typical techniques under this category exploit known geometrical relationships between model features to derive a set of constraints that can be used to reverse the camera 3D-to-2D projective transformation. This problem is important in computer vision, and in many areas like photogrammetry, robot navigation, motion tracking, object recognition and camera calibration. Solving this problem enables computers to understand human motion, as well as navigate through unknown environments. However, even though the object 3D dimensions and image point correspondences are assumed to be known in advance, it is difficult to solve as nonlinear techniques are usually involved, leading inevitably to numerical instability and multiple solutions. To further illustrate the concept of pose estimation, we begin with an example about 3D motion tracking.

1.1.1 Application - 3D Motion Tracking

The task of tracking an object can be divided into two parts: acquisition and tracking. As discussed in previous paragraphs, acquisition involves a rough estimation of the pose, velocities and some other state-variables of interest. In the first stage, a tracker is initialized to maintain a given multi-value time series of some possibly noisy measurements about the object. When new measurements are available, these variables are extrapolated to predict their values at the next sampling instant. Using this predicted object pose and a 3D-to-2D camera projection model, the tracker determines the target appearance in the scene, its positions, orientations, etc. These help to restrict the search range of object features to a relatively small part of the image. The next step is to compare the true image from camera with the predicted image, and search the true image for expected object features by correspondence, i.e. identifying which image feature correspond to each model feature. The 2D image is then back-projected to the 3D scene space to compute the discrepancy between them, and determine how this discrepancy affects the current state estimates. Finally, the whole process is repeated as soon as another set of measurements is available [24] (Fig. 1.1).

1.2 Image-based View Synthesis

In traditional computer graphics or model-based approaches, each object in a scene is represented by a geometrical formulation like a polygon mesh [10][43][78] using prior knowledge about the object. These geometrical entities are then rendered onto the screen by specifying their 3D positions, orientations and other parameters such as surface reflectance, illumination, etc. Applying this approach to simulate real world scenes can be difficult, however, for the possibly large number of objects involved, and the artistic distortions or exaggerations inherent in synthesizing. There is also an important trade-off among rendering time, scene complexity, and the huge amount of storage required for pre-rendered image segments. On this aspect, image-based transformation techniques offer a completely different approach [83][8]. Images produced by this method can appear strikingly lifelike and visually convincing without explicit modeling or reconstruction. By exploiting the geometric relations, e.g. pixel intensities, 2D positions, etc. inside a collection of images, these techniques combine 2D interpolations of shape and color to produce illusions of smooth transitions or trans-

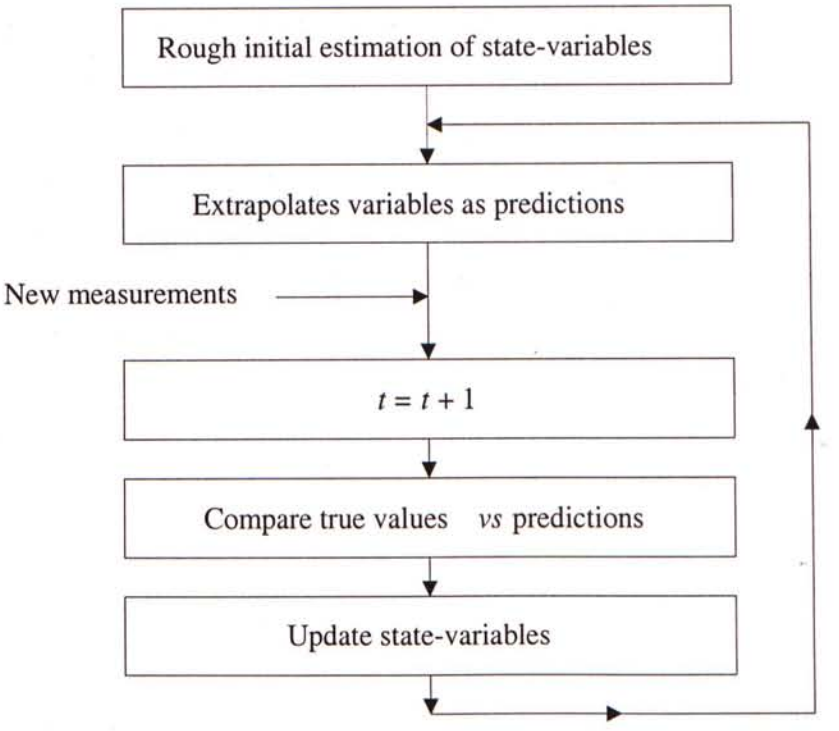

Figure 1.1: An example of 3D motion tracking application.

formation. Consequently, these techniques are simpler and more efficient than those by 3D reconstructions or modeling, because only a small amount of 3D information has to be recovered and the burden in representing scene information is reduced.

One interesting 3D movement to model in particular is viewpoint transformation (as in camera panning, zooming, and translation). This is critical for interactive graphics application like *virtual reality* (VR) navigation, in which the camera perspective changes continuously under user control. In the early part of this century, Walt Disney and other pioneers developed the *cel animation* technique that was used to generate cartoons and films like the *Three Little Pigs*, *Fantasia* and *Aladdin*. While sufficient to give an impression of camera movement, this technique does not model depth effects like parallax and occlusion, and therefore is somewhat unrealistic. A more compelling effect can be achieved by designing the background image with a specific camera path in mind. This technique was used in Walt Disney's 1940 animated film, *Pinocchio*, in which a window was moved in a fixed path along a special background image. This approach began to receive much interests when Chen and Williams [10] showed that virtual navigation among a given collection of images is possible by view interpolation. However, substantial efforts are required to locate

corresponding feature pairs, perform dense sampling of the scene, and build up image depth maps.

Perhaps Laveau and Faugeras [43] was the first group in attempting to generate novel views based on a collection of real world photographs, using fundamental matrices [48] together with pixel correspondences between different views. They demonstrated that view synthesis can be performed without depth map reconstruction. However, full correspondences have to be established prior to the synthesis. Later, McMillan [57] presented a plenoptic modeling system which made use of computer vision techniques to generate panoramas. Epipolar relationships between neighboring panoramas were established to indicate the possibility of linear view interpolation between different positions. Chen [11] also proposed another view generation system (QuickTime VR) that could synthesize arbitrary views about a point using a pre-rendered panorama at that point. Virtual navigation became possible at a much lower cost with these mosaicing techniques. However, their work was limited to cylindrical panoramas which have singularities near the top and bottom. Szeliski and Shum [78] reformulated cylindrical panorama representation into spherical ones to overcome these singularities. They also proposed an alignment process to rectify the accumulated errors in sampling, as well as a deghosting process to correct the intensity differences of a specific patch under different viewing conditions. To further simplify the complex relationship in generating novel views, Seitz [71] proposed another approach to image editing based on volume-elements (*voxel*). With *apriori* voxel correspondences, editing on one image would be automatically propagated to other images through voxel calculations. This approach is very interesting, but the accuracy of editing operations depends heavily on the voxel resolution [31]. He further argued that while image morphs enable the synthesis of 3D effects, they provide no direct control of the 3D transformation that is being synthesized. As a result, simple 3D transformation such as viewpoint changes are often difficult to convey accurately with image morphing methods. Another way of simulating depth effects in animation is through *layering*, which refers to composing a series of background images and windows that move at different speeds. This technique can model relative motion and occlusion between objects by representing the scene as a set of independently moving 2D layers rendered in back-to-front order. Researchers in computer vision and computer graphics have advocated this layering paradigm as an effective way for representing and rendering more complicated 3D scenes. Wood *et al.* [87] have

recently developed computer algorithms for partially automating the creation and segmentation of these warped background, which they call *multi-perspective panoramas.*

The techniques in previous paragraphs are examples where 2D representations and operations may be used to convey a strong impression of three-dimensionality. However they often involve a large number of real images, and require substantial efforts in locating corresponding feature pairs, which sometimes out-balance the advantages.

1.3 Thesis Contribution

This thesis contains a combination of theoretical results and practical algorithms. Its main contribution is:

- We develop an iterative algorithm that gives unique results for the *Perspective-3-Point* (P3P) problems. This is nontrivial because multiple solutions are inherent in such model-based pose estimation algorithms, if less than eight point correspondences are available. The efficiency and accuracy of our algorithm is demonstrated through comparisons with established approaches like Fischler and Bolles' [19]. In addition, as our algorithm requires as few as three point correspondences for every motion instant, it has higher flexibility in tracking complicated object such as articulated ones.

- Another novelty is that our panoramic walkthrough algorithm demonstrates the feasibility of view synthesis without dense correspondence information or detailed camera parameters. Specifically, given a pair of uncalibrated images about an object, we show that it is possible to uniquely synthesize perceptually correct in-between views along the line segment between optical centers. This is important because it paves the way for panoramic walkthrough algorithms that represent scene appearance with a collection of basis images.

A crucial property of our algorithms is that they are robust and produce realistic results even for significant violations of the assumptions. Our experimental results with various real image sequences clearly suggest that the algorithms are in fact more widely applicable than the theory predicts.

1.4 Thesis Outline

This report begins with our notations, terminologies, camera projection models, as well as a brief introduction on model-based motion analysis and panoramic view analysis (Chapter 2).

Chapter 3 investigates in greater depth the problem of model-based pose estimation. After an overview of related work, the first part of the chapter is about some theoretical issues of the P3P problems, and describes their classic solution by Fischler and Bolles. The remaining sections present our proposed iterative P3P solution, together with experiment results as well as comparisons with some other established approaches.

Chapter 4 considers the problem of view synthesis and panoramic construction. We would review existing approaches, explain the theory behind, and discuss their advantages and short-comings with adequate examples.

Motivated by limitations in the state of the art, we introduce in Chapter 5 our approach to integrate model-based pose recovery and panoramic view analysis. Given a set of images of a scene under different viewing conditions, our method recovers the 3D motion of an object in the scene from its subsequent image correspondences, and synthesizes 3D transformations of real scenes from these basis images. The chapter concludes by evaluating the performance and robustness of our proposed approach, when applied to real and synthetic image sequences.

Chapter 6 addresses some insight in the future directions of panoramic walk-through, and summarizes our work in this thesis.

Formula derivations are given in the appendices.

Chapter 2

General Background

Most of the techniques developed for motion estimation assume the use of an image as the primary input. However, the imaging process is a nonlinear process, which leads to increased difficulties in finding solutions. This chapter first examines the fundamental issues on camera models (section 2.2). Then we will give a brief introduction to the problems we are going to handle in this thesis: model-based motion analysis (section 2.3), panoramic representation (section 2.4), and image pre-processing (section 2.5).

2.1 Notations

Consider an object as shown in Fig. 2.1, **O** represents the optical center of a pin-hole camera, and feature points from a non-deformable object are represented by $\mathbf{p}_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^T$ at time t and $\mathbf{p}'_i = \begin{bmatrix} x'_i & y'_i & z'_i \end{bmatrix}^T$ at time $t+1$ respectively, where x_i, y_i and z_i are the coordinates in 3D space. In this thesis, bolded symbols denote vectors or matrices, ^{T} denotes matrix transpose, primed symbols denote measurements at time $t+1$, and $i = 1, ..., N$ is an index to the feature points. These points are projected onto an image plane at $z = f$ giving $\mathbf{q}_i = \begin{bmatrix} u_i & v_i \end{bmatrix}^T$ and $\mathbf{q}'_i = \begin{bmatrix} u'_i & v'_i \end{bmatrix}^T$, where $u_i = f\frac{x_i}{z_i}$, $v_i = f\frac{y_i}{z_i}$ and f is the camera focal length known precisely through careful calibration procedures [81]. The optical center of the camera is assumed to coincide with the origin of a Cartesian coordinate system.

As a convention, all objects are non-deformable so that their shapes will not change during any movement. We further assume that motions are rigid in the sense that all feature points on an object undergo the same motion.

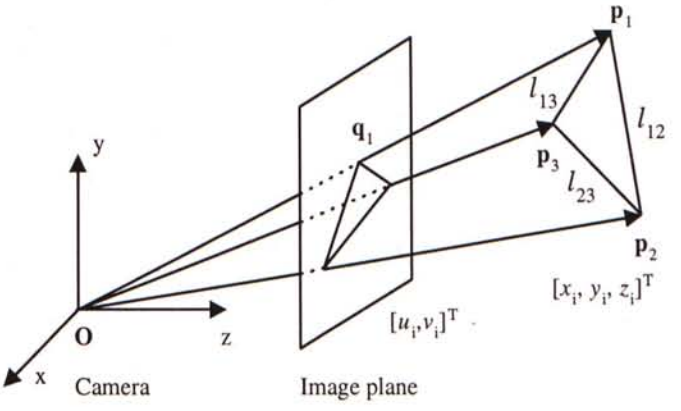

Figure 2.1: Our notation.

2.2 Camera Models

The fundamental problem in analytical solutions for perspective pose recovery is the intrinsic non-linearity of the geometrical constraints that arise when the imaging process is modeled as a perspective transformation. Fortunately, under certain special conditions, an imaging transformation that is actually perspective (or even more complex, for instance, if we take into account lens distortion), can be reasonably approximated with much simpler models in terms of linear constraints.

2.2.1 Generic Camera Model

A generic camera model can be defined as a 3×4 transformation matrix $\mathbf{M}$ that specifies the mapping between a point (represented as $\mathbf{p}_i = \begin{bmatrix} x_i & y_i & z_i & 1 \end{bmatrix}^T$ in homogeneous coordinates notation) in the 3D object space with its corresponding image point (represented as $\mathbf{q}_i = \begin{bmatrix} u_i & v_i & 1 \end{bmatrix}^T$) on the 2D image plane.

$$\lambda \mathbf{q}_i = \mathbf{M}\mathbf{p}_i$$

where λ is an arbitrary scalar constant. In particular, the matrix $\mathbf{M}$ can be decomposed as:

$$\mathbf{M} = \mathbf{K}\mathbf{I}p\mathbf{G} = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ 0 & 0 & k_{33} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} g_{11} & g_{12} & g_{13} & g_{14} \\ g_{21} & g_{22} & g_{23} & g_{24} \\ g_{31} & g_{32} & g_{33} & g_{34} \\ g_{41} & g_{42} & g_{43} & g_{44} \end{bmatrix} \tag{2.1}$$

where $\mathbf{K}$ (also called the intrinsic matrix) encodes the intrinsic parameters of a camera, such as the origin of its image plane, pixel aspect ratio, focal length, etc., and $\mathbf{G}$ (also called the extrinsic matrix) maps points from a 3D object-centered coordinate system to a 3D camera-centered coordinate system. Without any further constraints, this model has eleven *degree of freedom* (DOF) (because in homogeneous coordinates the overall scaling factor is irrelevant), and is denoted as the *projective camera model.* Readers are referred to [89] for more details.

2.2.2 Full-perspective Camera Model

In many situations, however, it is reasonable to assume that the calibrated version of $\mathbf{K}$ can be written as:

$$\mathbf{K}_{calib} = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

where f_x and f_y encode both the aspect ratio and the camera focal length, and $\begin{bmatrix} o_x & o_y \end{bmatrix}^T$ is the origin of the image plane. These four parameters can usually be determined through some *apriori* camera calibration [81]. Finally, the perspective mapping of $\mathbf{G}$ ($\mathbf{G}_{persp}$) between the model and the camera frames can be thought of as an Euclidean transformation, i.e. a composition of rotation and translation. The result of all these assumptions is known as a perspective camera model, which extrinsic matrix can be written as:

$$\mathbf{G}_{persp} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$$

where $\mathbf{R} = \begin{bmatrix} \mathbf{r}_x & \mathbf{r}_y & \mathbf{r}_z \end{bmatrix}^T$ is a 3×3 orthonormal matrix representing the relative rotation between the model and the camera frames, and $\mathbf{t} = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T$ is a column vector representing the relative translation.

Assuming $\mathbf{K}_{calib}$ to be an identity matrix, the 2D image projection $\begin{bmatrix} u_{persp} & v_{persp} \end{bmatrix}^T$ of a 3D point $\mathbf{p}$ is given by:

$$\begin{aligned} u_{persp} &= \frac{(\mathbf{p} \cdot \mathbf{r}_x + t_x)}{(\mathbf{p} \cdot \mathbf{r}_z + t_z)} \\ v_{persp} &= \frac{(\mathbf{p} \cdot \mathbf{r}_y + t_y)}{(\mathbf{p} \cdot \mathbf{r}_z + t_z)} \end{aligned} \tag{2.3}$$

A major problem with this model is its nonlinear formulation. The denominators in the above equations depend not only on t_z, but also on the 3D point $\mathbf{p}$ and the rotation matrix $\mathbf{r}_z$. So if the scene to be analyzed allows the use of approximate linear models, all the problems generated by the nonlinear nature of analytical perspective solutions can be avoided.

2.2.3 Affine Camera Model

To start with, the transformation matrix $\mathbf{M}$ in Eq. (2.1) may be assumed to take the following form to give an affine camera model:

$$\mathbf{M}_{aff} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

If a model of this kind is used, the coordinates of an image point $\begin{bmatrix} u & v \end{bmatrix}^T$ can be expressed as a linear function of the corresponding model coordinates $\begin{bmatrix} x & y & z \end{bmatrix}^T$. However, this linearization is a rather strong assumption that is in general valid only if the object dimensions are relatively small compared with the object-camera distance, and will break down when the object is close to the camera, e.g. during a video conference session. In addition, the number of unknowns in the above camera model is still large in the sense that it may be too complicated to be used for real life application. Some reasonable restrictive assumptions can be used to obtain more tractable specializations. Assuming the four intrinsic parameters in Eq. (2.2) are known *apriori*, and the projective transformation is Euclidean, there are two different linearized approximations to the perspective camera model. The difference is that one of them, known as the *weak-perspective* camera model, is an approximation of order

zero, while the other, known as the *paraperspective* camera model, is a first-order approximation.

2.2.4 Weak-perspective Camera Model

In order to understand more clearly the relationship between the perspective camera model and its two affine approximations, we factorize t_z, that is constant for all points of interest in the object space, out of the denominators in Eq. (2.3). Then u_{persp} and v_{persp} can be expressed as a function of $\mathbf{p}$ by:

$$\begin{aligned} u_{persp} &= \frac{\mathbf{p} \cdot \mathbf{r}_x + t_x}{t_z\,(1+\epsilon)} \\ v_{persp} &= \frac{\mathbf{p} \cdot \mathbf{r}_y + t_y}{t_z\,(1+\epsilon)} \end{aligned} \tag{2.4}$$

where

$$\epsilon = \frac{\mathbf{p} \cdot \mathbf{r}_z}{t_z}$$

The weak-perspective camera model basically approximates a perspective projection by assuming that all the points on a 3D object are roughly at the same distance from the image plane of the camera, so that the nonlinear factor $\frac{1}{1+\epsilon}$ can be replaced by the constant 1 to give a linear relation between $\mathbf{p}$, u, and v:

$$\begin{aligned} u_{weak} &= \frac{\mathbf{p} \cdot \mathbf{r}_x + t_x}{t_z} \\ v_{weak} &= \frac{\mathbf{p} \cdot \mathbf{r}_y + t_y}{t_z} \end{aligned} \tag{2.5}$$

Another interpretation for this model is to view it as a two-step process as shown in Fig. 2.2.

The first stage involves an orthographic projection of an object onto a plane parallel to the image plane and at a distance z_c, the z-coordinate of the object centroid. Then a perspective projection, which is an uniform scaling of the resulting image by the factor $\frac{1}{t_z}$, is applied to get the image coordinates. This type of transformation is also known as scaled orthography. However, this zero-order approximation to the full perspective projection model would have substantial error when the object is far away from the optical axis, and is valid only when the object-camera distance is much larger than the object dimensions.

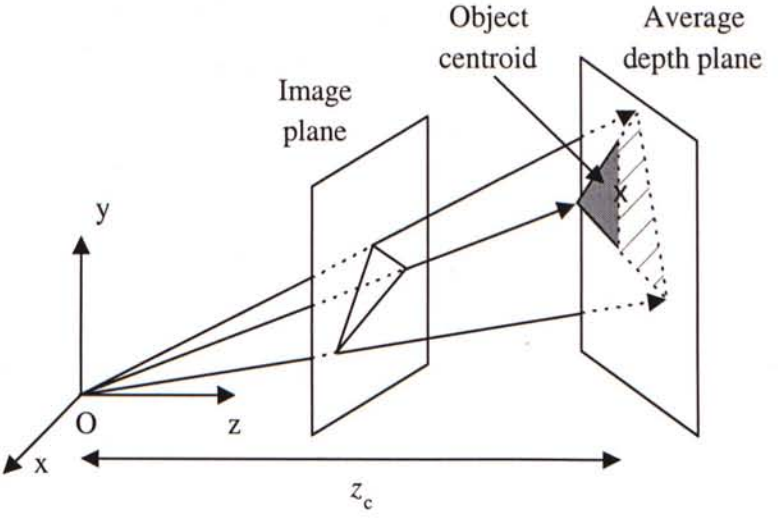

Figure 2.2: Weak-perspective camera projection model.

2.2.5 Paraperspective Camera Model

The paraperspective camera model, on the other hand, uses the first-order term of the expansion of $\frac{1}{1+\epsilon}$ around the point $\epsilon = 0$, in addition to the term of order zero. So, it assumes that $\frac{1}{1+\epsilon} \approx 1 - \epsilon$. The expressions for u and v obtained after this linearization are then approximated again by neglecting the terms that involve $\frac{p^2}{t_z^2}$, as shown by Christy and Horaud [13]:

$$u_{para} = \frac{\mathbf{p} \cdot \left(\mathbf{r}_x - \frac{t_x}{t_z}\mathbf{r}_z\right) + t_x}{t_z}$$

$$v_{para} = \frac{\mathbf{p} \cdot \left(\mathbf{r}_y - \frac{t_y}{t_z}\mathbf{r}_z\right) + t_y}{t_z}$$

The geometrical semantics of this camera model is illustrated and compared with the perspective and weak-perspective models in Fig. 2.3. In particular, the paraperspective camera model corresponds to projecting each model point $\mathbf{p}$ along the line parallel to $\mathbf{Op}_c$, where $\mathbf{p}_c$ is the object centroid. The resulting virtual image is then uniformly scaled by the factor $\frac{1}{t_z}$ onto the image plane.

These two camera models (weak-perspective and paraperspective) are the simplest ones that can be used in application in which the pose of an observed objects with respect to the camera is unconstrained. So, in situations where the object-camera distance is relatively big when compared with the object dimensions, these models

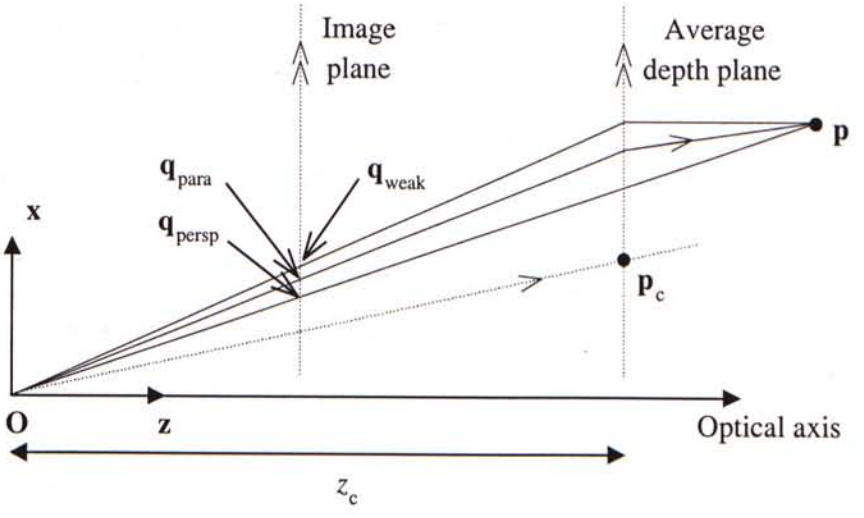

Figure 2.3: Geometrical configurations of full-perspective, weak-perspective and para-perspective camera model.

can yield precise pose estimates at a low computational cost. Furthermore, solutions based on them are in general much simpler to analyze and much easier to implement than their perspective counterparts, and their linearity eliminates the necessity of calibration of certain intrinsic camera parameters. Due to these advantages, they have been widely used for object recognition, structure and motion estimation, and augmented reality. On the other hand, in spite of their linearity, both camera models are still too general for application in which the positions or orientations of objects with respect to the camera are restricted by some fundamental constraints. Wiles and Brady [85], for instance, proposed some even simpler camera models for the important problem of smart vehicle convoying on highways. In their analysis, a camera rigidly attached to a certain trailing vehicle was used to estimate the structure of a leading vehicle, in such a way that the paths traversed by these two vehicles are composed exclusively by a series of translation and rotation parallel to a unique ground plane. Many of their observations and suggestions can be generalized to the analogous problem of pose recovery.

2.3 Model-based Motion Analysis

Much research work on this problem has been reported in the literature [34]. The general idea of the analytical solutions is to work with a fixed number of correspon-

dences and then to express image properties such as feature positions, lines, planes, orientations or apparent angles as a function of a predefined set of pose parameters. Since the shape and size of a rigid object remain unchanged irrespective of its motion, all feature points would have the same transformation. By matching the resulting expressions against the corresponding actual image measurements, one can derive a set of polynomial equations or constraints involving the pose parameters. Finally, if the number of correspondences is big enough, these equations can be combined algebraically, yielding the desired pose. Analytical solutions are traditionally classified according to the nature of the geometrical constraints used, or prior knowledge about objects. Three major categories have been identified in the literature: point correspondences, line correspondences, and angle correspondences.

2.3.1 Point Correspondences

Many solutions for point-based motion analysis have been proposed. According to the level of difficulty, they can be classified roughly into three categories [34]:

With 3D-to-3D Point Correspondences (Fitting of two 3D point sets)

Given 3D coordinates $\mathbf{p}_i$ and $\mathbf{p}'_i$, our job is to find an optimal transformation (rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$) between them to account for their difference in position and orientation such that:

$$\sum_{i=1}^{N} |\mathbf{p}'_i - (\mathbf{R}\mathbf{p}_i + \mathbf{t})|^2$$

is minimized for all i [5][32].

Basically, this is a solved problem and various solutions have been proposed with satisfactory results. In particular, we will discuss two solutions proposed by Arun *et al.* [5] and Horn [32] in Chapter 3. A special point worth noting is that they do not require complicated matrix operations, which computation complexity increases exponentially with the number of feature points. This observation will prove to be very useful when we explore application on image analysis.

With 2D-to-3D Point Correspondences (Model-based Pose Estimation)

Analysis with 2D-to-3D point correspondences refers to the recovery of 3D motion parameters (positions and orientations) from subsequent 2D image points (obtained through a 3D-to-2D projection transformation) provided that the original 3D descriptions, e.g. depth or 3D coordinates, of an object or the scene are given. Although this problem is simpler than the structure from motion problem (to be discussed in the next paragraph), solving it is by no means easy for the variety of camera projection models that can be used to form the 2D projections, and the large number of possible positions and orientations that can be deduced from an object projection. In this thesis, we are particularly interested in this class of problems with the following simplifications. The first one being that any motion can be broken down into a sequence of consecutive poses, so that the movement undergone by an object between two consecutive poses is small. In this way, mathematical tools are available to solve the problem in a straightforward way. Second, we assume that the object 3D description is available to provide more cues about object orientations, by re-projecting the model onto image plane. However, the resulting problem is still difficult to solve if we choose to use the perspective projection model. In perspective projection, the depth information of the object are transformed by a nonlinear mapping which yields a foreshortening effect in the captured images. In addition, the rotation of the object in 3D space often leads to a nonlinear formulation which increases dramatically the computational requirement. Nevertheless, the solution to this problem is useful in many application like photogrammetry, passive navigation, industry inspection and human-computer interfaces.

With 2D-to-2D Point Correspondences (Structure from Motion)

This third class of problems is about motion analysis with only 2D point correspondences $\mathbf{q}_i$ and $\mathbf{q}'_i$. As the 3D structure of an object is not given, it is generally agreed that this extraction of 3D information is very difficult due to the large number of unknown variables to be recovered from just two 2D images. We may have to determine both the motion and structure of an object before any meaningful interpretation about the images can be taken. Even if the depth information and translational parameters can be found, they can only be determined up to a scale factor for the non-linearity of this problem. Mathematically, it can be written as the minimization of:

$$\sum_j |\mathbf{q}_{ij} - \rho(\mathbf{R}_j \mathbf{p}_i + \mathbf{t}_j)|^2$$

for all i-th points and j-th frames, where ρ is a projection function which projects the 3D points $\mathbf{p}_i$ onto the image plane.

Under this paradigm, the estimated motion is the best solution in the least square sense. However, most of the mathematical tools available in the literature involve complex mathematical tools that are only suitable for off-line computation. For example, the Levenberg-Marquardt algorithm [67] is the mostly quoted method to recover both the structure and motion from an image sequence [84]. Tomasi and Kanade [80] proposed a factorization method to solve the above problem when the camera model is an orthographic one. Later, an extension of the factorization method to paraperspective projection model [65] was proposed to take better account of the perspectivity effect. Another major direction was based on an essential matrix approach [34][79].

2.3.2 Line Correspondences

Line correspondences are constraints that are weaker than point correspondences. If N point correspondences are known, the lines obtained by grouping them in pairs can be used as input to any method based on line correspondences. But if an arbitrary set of line correspondences is available, it may be impossible to cast the problem of recovering the pose based on them into an equivalent problem involving only point correspondences, because these lines may not have pair-wise intersections in 3D space. On the other hand, lines are in general easier to be located in digitized images than points. The extraction of point features in general can only be performed reliably for distinctive features that can be correlated with predefined templates (but this type of feature is not always present in real scene). The identification of linear features, on the other hand, is based on the extraction of intensity gradients, which can be done in much more generic situations.

Horaud *et al.* [27] introduced a method to solve the pose recovery problem when the image correspondences for three non-coplanar lines with a common intersection point are given, commonly known as a restricted form of the Perspective-3-Line (P3L) problem in the literature. The fundamental idea of this solution, as well as of most solutions based on line correspondences, is to exploit a geometrical entity known as the interpretation plane. Simply speaking, if the imaging process is modeled as a

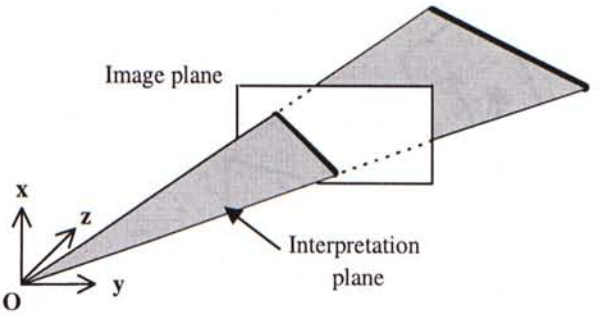

Figure 2.4: The interpretation plane of a generic image edge in line correspondence problems.

perspective transformation, then a model edge can be the right correspondence for a given image edge if and only if it lies on the plane defined by the image edge and the optical center, or alternatively the plane can be considered as the geometrical locus of all possible model edge positions that result in a certain image edge. This plane, represented as a dark surface in Fig. 2.4, is called the interpretation plane.

This solution by Horaud *et al.* could eventually be reduced to a single fourth-degree polynomial equation, and analytically methods can then be used. A more general solution to the P3L problem, which can be used even in cases where the lines with known correspondences do not intersect each other, was presented by Dhome *et al.* [18]. Actually, this solution is very similar to the restricted solution presented above. However, due to the absence of a common intersection point between the edges, the analytical resolution of this system results in an eighth-degree polynomial equation. Dhome *et al.* also showed that the eighth-degree polynomial equation involved in the general solution can be simplified to degree four, if either the three edges with known correspondences are coplanar or they have a common intersection point. However the method proposed by Horaud *et al.* is often used for its greater simplicity, in spite of being less general. Readers are referred to [18][27][61] for more information about line correspondences.

2.3.3 Angle Correspondences

Another type of geometrical constraint that can be used in model-based pose recovery are matches involving angles between model edges in 3D space and the corresponding

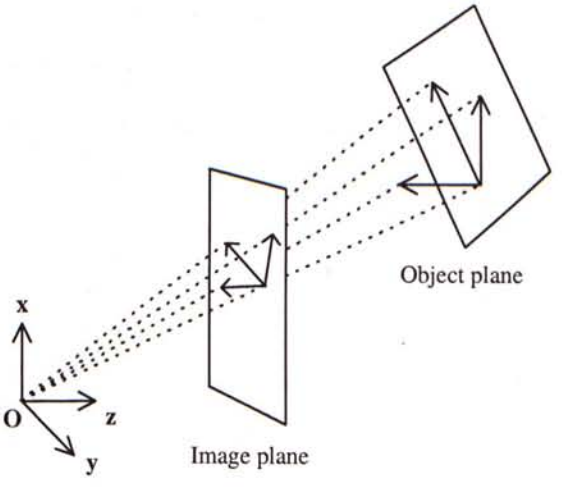

Figure 2.5: Definition of intermediate coordinate frame in the restricted P3A problem.

apparent angles in the image plane. Shakunaga and Kaneko [72] presented a solution to a restricted form of the Perspective-3-Angle (P3A) problem. More specifically, this solution requires at least two of the model angles involved to be right angles, and that the three edges involved are part of a unique trihedral vertex (or in other words, they share a common intersection point). As shown in Fig. 2.5, the geometrical insight that leads to the solution is to notice that, in this case, one of the model edges can be thought of as the image of the vector normal to a plane defined by the other two edges. The image of the common vertex and the image of this normal edge can then be used to build an intermediate coordinate system between the camera and the object frame, which is neither viewer-centered nor object-centered and thus makes the problem of recovering relative orientation easier.

A solution for the P3A problem with arbitrary angles in a trihedral vertex was later presented by Wu *et al.* [88]. Again, the central idea is to create a virtual image in which the shared vertex is foveated. However in Wu's solution, one particular transformation to obtain the intermediate image frame is arbitrary chosen. Readers are referred to [72][88] for more information about angle correspondences.

2.4 Panoramic Representation

There has been a growing interest in the use of panoramas as a visualization device, a tool for video compression [39], interactive video analysis, and enhancements to

the field of view [75][55] and resolution [12] of cameras. Moreover, they play an important role in image-based rendering to synthesize photo-realistic novel views from collections of real (or pre-rendered) images [10], for the construction and navigation of virtual environments [57][11][77]. For such application, it is often desirable to have full-view panoramas which cover the whole viewing sphere and allow visualization in any direction. Unfortunately, most of the results to date have been limited to cylindrical ones obtained with cameras rotating on leveled tripods adjusted to minimize motion parallax [40]. This has limited the users of mosaic building to researchers and professional photographers who can afford such specialized equipment.

Although the idea of mosaicing is simple and clear, a number of subtle variations arise when considering how a complete mosaic representation may be developed for the different types of application. For example, regarding the geometric transformation models used for aligning images to each other, we describe two types of mosaics (*static*, and *dynamic*) that are suitable for different scenarios. They are then unified and generalized in a mosaic representation called *temporal pyramid.* These topics, together with some other extensions, would be described thoroughly in the following paragraphs.

2.4.1 Static Mosaic

Static mosaic is the most common panoramic representation [55][41][75]. It exploits large spatial correlation (over large portions of image frames), and is therefore an efficient scene representation. An example of a static mosaic image is shown in Fig. 2.6.

This is done in batch mode, by aligning all frames to a fixed coordinate system (which can be either user-defined or chosen automatically according to some criteria). The aligned images are then integrated using different types of temporal filters into a mosaic image, and the significant *residues* (information that is not represented in the mosaic) are computed for each frame relative to the mosaic. It is ideal for video storage and retrieval, and for content-based indexing into large digital libraries. It also relieves the tedium associated with video manipulation and analysis by supporting efficient access to individual frames of interest. In addition, it can be used as a visualization tool to enhance image content. This application will be described in greater details in section 4.4.

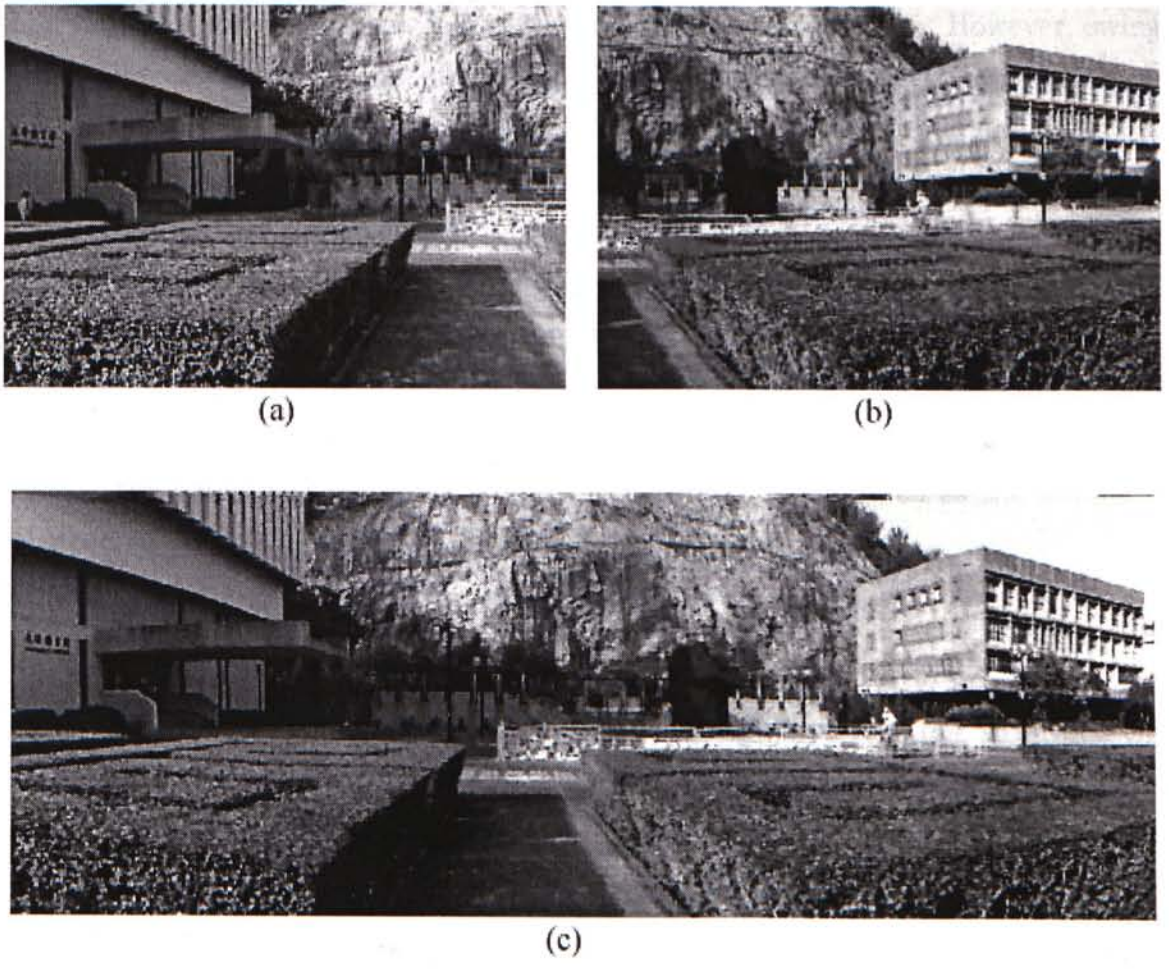

Figure 2.6: A static mosaic image: (a), (b) two source frames, (c) part of the resulting panorama.

2.4.2 Dynamic Mosaic

Since a static mosaic is constructed in batch mode, it cannot depict the dynamic aspects of a video sequence. This requires a dynamic mosaic (a sequence of evolving mosaic frames) where the content of each new mosaic frame is updated with the most recent available information from an input video sequence. In this case, the residues reflect only changes in the scene that occur in the time elapsed between successive frames, as well as additional parts of the scene that were revealed to the camera for the first time. They are different to those in the static case, which represent any movements in the video sequence since a static mosaic was built. As a result, the amount of residual information in a dynamic mosaic is smaller and more efficient

than that in a static one. This is ideal for low bit-rate transmission. However, owing to their incremental and sequential nature of frame reconstruction, dynamic mosaics lack the important capability of random and immediate access to individual frames, making them unsuitable for video manipulation and editing.

In addition, the choice of coordinate systems for constructing and visualizing a dynamic mosaic depends on its application. For example, in remote surveillance over a fixed scene, it is beneficial to construct a dynamic mosaic with respect to a stationary background. On the other hand, in flight surveillance, it makes more sense to keep a dynamically updating coordinate system that matches that of the view as seen by the pilot (with a gradually growing field of view obtained as the mosaic is constructed).

2.4.3 Temporal Pyramid

Static and dynamic mosaics are extremes of a continuum. In order to bridge the gap between these two and benefit from the advantages of both representations (i.e. efficiency versus random-access to frames), a static mosaic can be extended to use a hierarchy of mosaics which levels corresponds to different amounts of temporal integration. In this temporal pyramid, the finest level contains the set of original images, one for each frame in the input sequence. The temporal sampling decreases successively as we go from fine to coarse resolution levels of the pyramid. The coarsest level will consist of a pure static mosaic (Fig. 2.7). Reconstruction can be achieved by hierarchically combining the static mosaic with residual information in logarithmic time.

2.4.4 Spatial Pyramid

On the other hand, when the initial mis-registration between frames is more than a few pixels, a useful heuristic is to use a hierarchical or coarse-to-fine optimization scheme proposed by Bergen *et al.* [6]. Estimates from coarser levels of the pyramid are used to initialize the registration at finer levels [3][6]. This is a remarkably efficient technique, and we typically use three or four pyramid levels in our experiments. However, it may sometimes fail because image details may not exist or may be strongly aliased at coarse resolution levels. The benefits of such approaches are discussed in more details in [37][38].

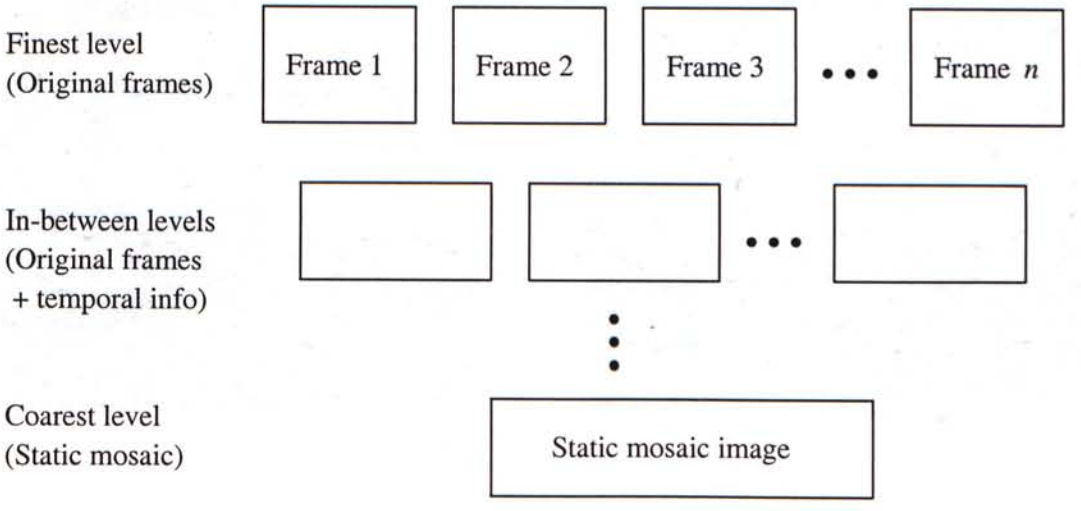

Figure 2.7: A temporal mosaic pyramid.

2.5 Image Pre-processing

Even with the best camera, component images in a sequence are bound to have variations in color hue, intensity, contrast, etc., because of the vast number of variable factors in the surrounding environments that can affect resulting intensity values. In particular, differences in color or intensity profiles among two adjacent images would become very obvious in image stitching, as can be seen in the bottom image of Fig. 2.8. Thus, in order to avoid sudden changes in image details along seam regions, image pre-processing like feature touch-up or histogram equalization are necessary. In our experiments, image pre-processing consists of three parts: feature extraction, local enhancements, and dynamic range stretching.

2.5.1 Feature Extraction

Moravec Interest Operator

Directional variance is measured over small square windows (Fig. 2.9). *Sum square differences* (SSD) of adjacent pixels in each of four directions (horizontal, vertical, and two diagonals) over each window are calculated, and the interest measure of this window is defined as the minimum of these four sums. This measure is evaluated on windows spaced half a window width apart over the entire image. Features are chosen at the center points of windows which interest measures are local maxima, i.e. if a

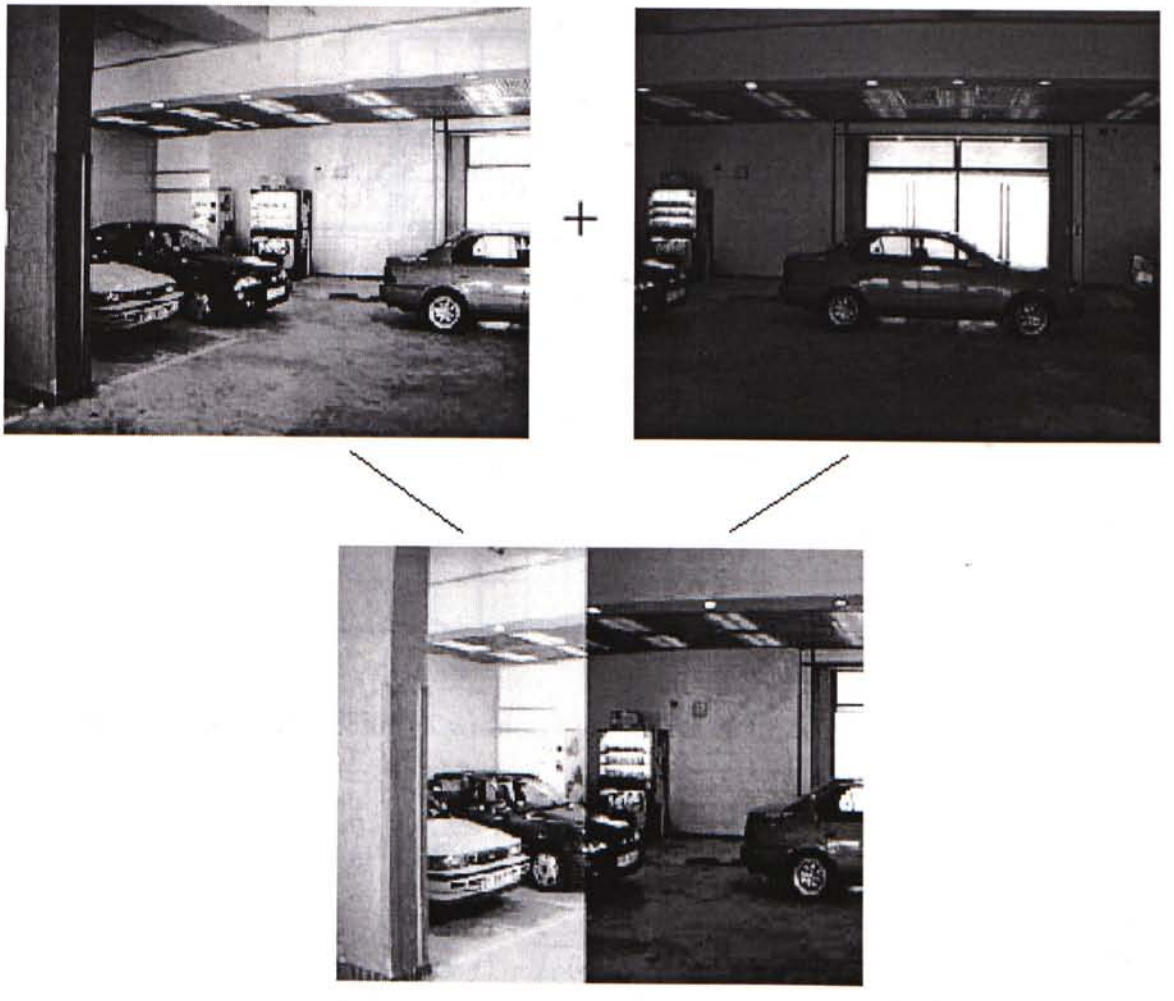

Figure 2.8: Adjacent source images of high intensity difference, and their stitched result.

window has the largest value among 25 windows which overlap or contact it [60].

Matching through Correlation

After feature point extraction, correspondences of these feature points are found by calculating their matching scores. This algorithm is known as *matching through correlation* [89] (Fig. 2.10).

Briefly speaking, with N_1 and N_2 feature points in image 1 (I_1) and image 2 (I_2) respectively. Then with respect to every single feature point in I_1, the correlation routine is executed once for each of the N_2 feature points in I_2. The resulting time complexity is of $O(N_1 \times N_2)$. The matching score of two image patches $I_1(u_1, v_1)$ and $I_2(u_2, v_2)$ of size $(2m_1 + 1) \times (2m_2 + 1)$ (in our experiments, a window size of

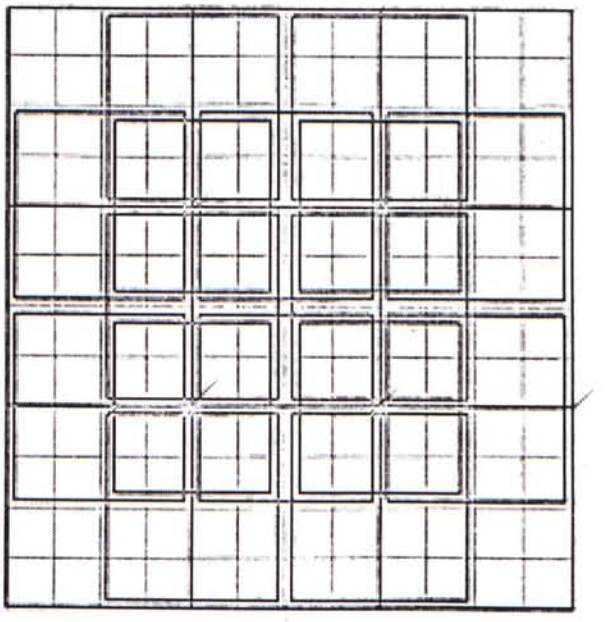

Figure 2.9: The 25 windows for Moravec interest operator.

15×15 pixels is used) is given by:

$$\frac{\sum_{i=-m_1}^{m_1} \sum_{j=-m_2}^{m_2} \left[I_1(u_1+i, v_1+j) - \overline{I_1(u_1, v_1)} \right] \times \left[I_2(u_2+i, v_2+j) - \overline{I_2(u_2, v_2)} \right]}{(2m_1+1)(2m_2+1)\sqrt{\sigma^2(I_1) \times \sigma^2(I_2)}}$$

where

$$\overline{I_k(u,v)} = \sum_{i=-m_1}^{m_1} \sum_{j=-m_2}^{m_2} \frac{I_k(u+i, v+j)}{(2m_1+1)(2m_2+1)}$$

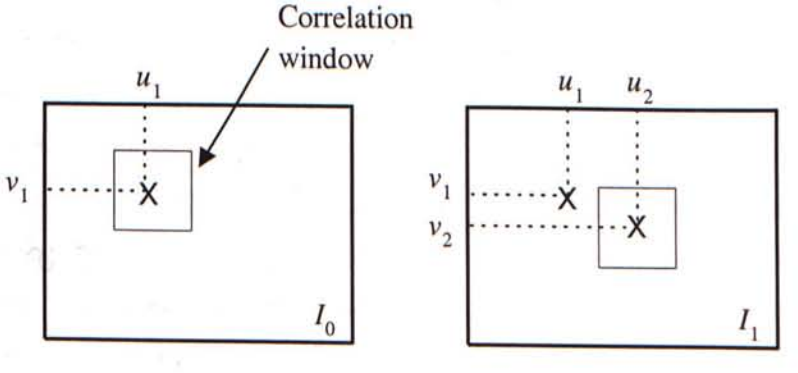

Figure 2.10: Matching through correlation.

is the mean intensity level at point $\begin{bmatrix} u & v \end{bmatrix}^T$ of I_k ($k = 1$ or 2), and

$$\sigma(I_k) = \sqrt{\frac{\sum_{i=-m_1}^{m_1} \sum_{j=-m_2}^{m_2} I_k^2(u,v)}{(2m_1+1)(2m_2+1)} - \overline{I_k(u,v)}^2}$$

is the standard deviation of the image I_k in the neighborhood $(2m_1+1) \times (2m_2+1)$ of $\begin{bmatrix} u & v \end{bmatrix}^T$. With this definition, matching scores are in the range $[-1, 1]$. If the matching score of a pair of feature points in I_1 and I_2 is larger than a threshold (0.8 in our current implementation), they are considered a pair of corresponding points. On the other hand, for a matching score about -1, the two correlation windows are not similar at all.

However, it should be noted that both of these processes are computational-intensive that the processing time required may grow exponentially with image sizes. This is an important trade-off in implementing a fully automated system. Moreover, the image content may differ by a wide margin, a feature point in one image may not exist at all in other images, and the one-to-one mapping that correlation is designed for would hardly work in this situation. In addition, the variance measure in Moravec interest operator depends on adjacent pixel differences and responds to high frequency noise in the image.

2.5.2 Spatial Filtering

The use of spatial masks for image processing is called spatial filtering (as opposed to frequency domain filtering using the Fourier transform), and the masks themselves are called spatial filters. In this section, we consider linear and nonlinear spatial filters for image enhancement.

Lowpass filters attenuate or eliminate high-frequency components in the Fourier domain while leaving low frequencies untouched (i.e. the filter preserves low frequencies). High-frequency components characterize edges and other sharp details in an image, so the net effect of lowpass filtering is image blurring. Similarly, highpass filters attenuate or eliminate low-frequency components. Because these components are responsible for the slowly varying characteristics of an image, such as overall contrast and average intensity, the net result of highpass filtering is a reduction of these features and a correspondingly apparent sharpening of edges and other sharp details.

Regardlessly of the types of linear filters used, the basic approach is to sum up the products of mask coefficients time pixel intensity values under the mask at a specific

location in the image. Eq. (2.6) shows a general 3×3 mask and Eq. (2.7) shows a typical 3×3 pixel image region (only intensity values are considered):

$$\begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} \tag{2.6}$$

$$\begin{bmatrix} z_1 & z_2 & z_3 \\ z_4 & z_5 & z_6 \\ z_7 & z_8 & z_9 \end{bmatrix} \tag{2.7}$$

The response of a linear mask is

$$s = \sum w_i z_i \tag{2.8}$$

If the center of the mask is at location $\begin{bmatrix} u & v \end{bmatrix}^T$ in the image, the intensity value of the pixel located at $\begin{bmatrix} u & v \end{bmatrix}^T$ is replaced by s. The mask is then moved to the next pixel location in the image and the process is repeated. In the following discussion, we would briefly introduce median filtering, highpass filtering, high-boost filtering, derivative filtering and local enhancement.

Median Filtering

One of the principal difficulties of the smoothing method discussed in preceding paragraphs is that it blurs edges and other sharp details. If the objective is to achieve noise reduction rather than blurring, an alternative approach is to use median filters. That is, the intensity value of each pixel is replaced by the median of the intensity values in a neighborhood of that pixel. This method is particularly effective when the noise pattern consists of strong, spiky components and the characteristic to be preserved is edge sharpness. Median filters are nonlinear.

Highpass Filtering

The principle objective of sharpening filters is to highlight fine details in an image or to enhance details that have been blurred, either in error or as a natural effect of a

particular method of image acquisition. A classic implementation for a 3×3 highpass mask is shown in Eq. (2.9).

$$\frac{1}{9} \times \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{2.9}$$

Note that this filter has a positive value in the center and negative coefficients in the outer periphery. As the sum of coefficients is zero, when the mask is over an area of constant or slowly varying gray level, the mask response according to Eq. (2.8) is zero or very small. This implies that the average intensity value in the image is reduced to zero, and the resulting image must have some negative gray levels. As we deal only with positive intensity levels, some forms of scaling and/or clipping are applied so that the intensity values of the final result span the range $[0, L-1]$. Significantly better results can be obtained by using high-boost filters to be described in the next paragraph.

High-boost Filtering

A highpass filtered image may be computed as the difference between the original image and a lowpass filtered version of that image, i.e.

$$\text{Highpass} = \text{Original - Lowpass}$$

Multiplying the original image by an amplification factor A yields the definition of a high-boost filter:

$$\begin{aligned} \text{Highpass} &= (A)(\text{Original}) - \text{Lowpass} \\ &= (A - 1)(\text{Original}) + \text{Original} - \text{Lowpass} \\ &= (A - 1)(\text{Original}) + \text{Highpass} \end{aligned}$$

An $A = 1$ value yields the standard highpass result. When $A > 1$, part of the original is added back to the highpass result, which restores partially the low-frequency components lost in the highpass filtering operation. As A increases, the background of the high-boosted image becomes brighter. The consequence is that the final result

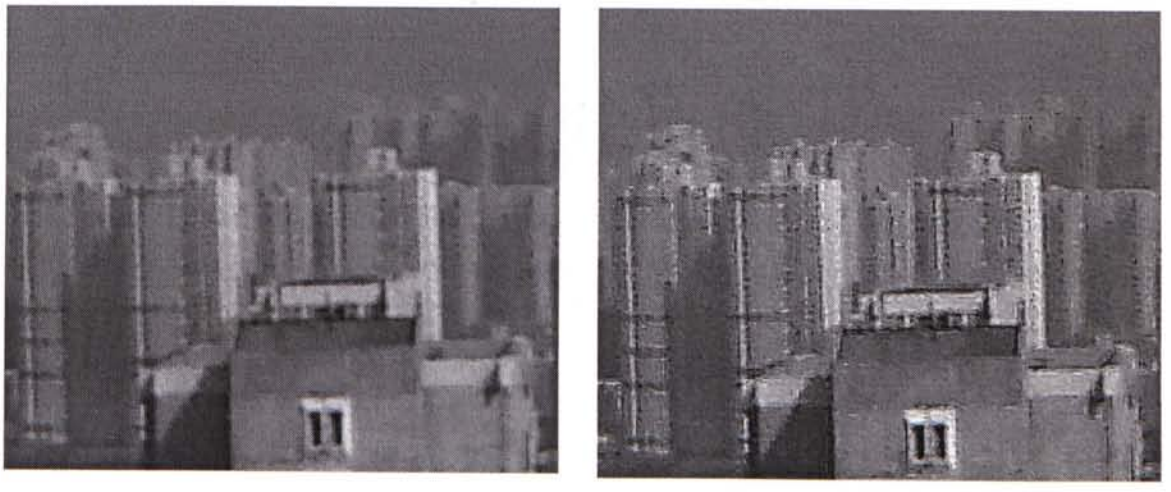

Figure 2.11: (Left) Original image. (Right) Image after high-boost filtering.

looks more like the original image, with a relative degree of edge enhancement that depends on the value of A. In terms of implementation, the preceding results can be combined by using a mask as:

$$\frac{1}{9} \times \begin{bmatrix} -1 & -1 & -1 \\ -1 & w & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

where $w = 9A - 1$ with $A \geq 1$ determines the nature of the filter. The advantage of such filtering is vividly demonstrated in Fig. 2.11. Note that noise plays a significant role in the visual appearance of the image that has been high-boost filtered.

Derivative Filtering

Averaging of pixels over a region tends to blur details in an image. As averaging is analogous to integration, differentiation can be expected to have the opposite effect and thus sharpen an image. The most common method of differentiation in image processing application is the gradient. For a function $f(u, v)$, the gradient of f at $\begin{bmatrix} u & v \end{bmatrix}^T$ is defined as the vector:

$$\nabla \mathbf{f} = \begin{bmatrix} \frac{\partial f}{\partial u} & \frac{\partial f}{\partial v} \end{bmatrix}^T$$

The magnitude of this vector,

$$\nabla f = \text{mag}(\nabla \mathbf{f}) = \left[\left(\frac{\partial f}{\partial u} \right)^2 + \left(\frac{\partial f}{\partial v} \right)^2 \right]^{1/2} \tag{2.10}$$

is the basis for various approaches to image differentiation. For an image region shown in Eq. (2.7) where z_i denotes the intensity value, Eq. (2.10) can be approximate at point z_5 by

$$\nabla f \approx |(z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)| + |(z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)| \tag{2.11}$$

The difference between the third and first row of the 3×3 region approximates the derivative in the x-direction, and the difference between the third and first column approximates the derivative in the y-direction. The following masks in Eq. (2.12) (called the *Prewitt operators*) can be used to implement Eq. (2.11).

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.12}$$

Eq. (2.13) shows another pair of masks (called the *Sobel operators*) for approximating the magnitude of the gradient.

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.13}$$

2.5.3 Local Enhancement

Most processing methods discussed in the previous sections are global in the sense that pixels are modified by a transformation function based on the gray-level distribution over an entire image. Although this approach is suitable for overall enhancement, it is often necessary to enhance details over small areas. The number of pixels in these areas may have negligible influence on the computation of a global transformation, so the use of this type of transformation does not necessarily guarantee the desired enhancement. The solution is to use transformation functions based on local gray-level distribution in the neighborhood of every pixel in the image. The intensity mean

and variance (or standard deviation) are two such properties frequently used because of their relevance to the appearance of an image: the mean is a measure of average brightness and variance is a measure of contrast.

A typical local transformation based on this concept maps the intensity of an input image $f(u,v)$ into a new image $g(u,v)$ by performing the following transformation at each pixel location $\begin{bmatrix} u & v \end{bmatrix}^T$:

$$g(u,v) = \Gamma\{\Gamma\{A(u,v)\cdot[f(u,v) - m(u,v)] + m(u,v)\} + k_1 \cdot \Gamma\{f(u,v)\}\} \quad (2.14)$$

where

$$A(u,v) = k_2 \cdot \frac{M}{\delta(u,v)}$$

In this formulation $\Gamma\{.\}$ is a function to normalize intensity levels to the range $[0, L-1]$, $m(u,v)$, $A(u,v)$, and $\delta(u,v)$ are the gray-level mean, gain factor, and standard deviation computed locally with respect to a small $n \times n$ window centered at $\begin{bmatrix} u & v \end{bmatrix}^T$. M is the global mean of $f(u,v)$, and k_1 and k_2 are constants in the range $[0,1]$ inclusively. In our current implementation, we used $n = 15$, $k_1 = 1.0$, $k_2 = 0.7$ and restrict $A \in [1,20]$ to balance large excursions of intensity in isolated region.

The values of variable quantities A, m, and δ depend on a predefined neighborhood of $\begin{bmatrix} u & v \end{bmatrix}^T$. Application of the local gain factor to the difference between $f(u,v)$ and the local mean amplifies local variations. Because A is inversely proportional to the standard deviation of the intensity, areas with low contrast receive larger gain. The mean is added back in Eq. (2.14) to restore the average intensity level of the image in the local region (Fig. 2.12).

2.5.4 Dynamic Range Stretching or Compression

Low-contrast images can result from poor illumination, lack of dynamic range in the imaging sensor, or even wrong setting of a lens aperture during image acquisition. The idea behind contrast stretching or compression is to reallocate the dynamic range of the gray levels in the image being processed. Fig. 2.13 is a typical transformation function used in this application, and Fig. 2.14 shows a sample result of dynamic range stretching.

Figure 2.12: (Left) Original image. (Right) Image after local enhancement.

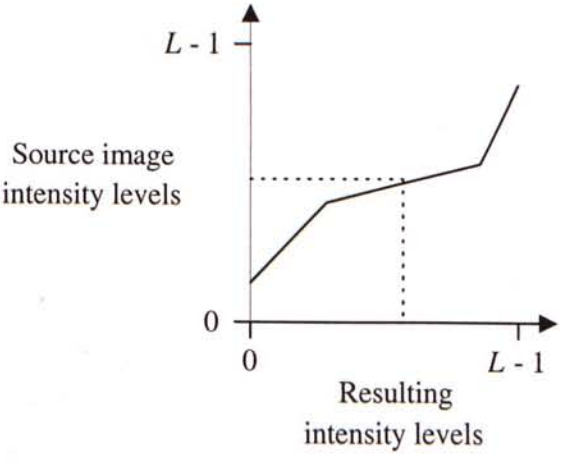

Figure 2.13: A typical transformation function for dynamic range stretching or compression.

2.5.5 YIQ Color Model

Basically, the YIQ (luminance, inphase, quadrature) system is a recoding of the RGB (red, green, blue) scheme for transmission efficiency. It was designed to take advantage of the greater sensitivity in human visual system to changes in luminance than to changes in hue or saturation. The principal advantage of this model in image processing is that the luminance (Y) and color information (I and Q) are decoupled. The Y-component provides all the video information required by a monochrome television set, and can be processed without affecting its color content. For example, we can apply histogram equalization to a color image represented in YIQ format simply

Figure 2.14: (Left) Original image. (Right) Image after dynamic range stretching.

by applying histogram equalization to its Y-component. The relative colors in the images are not affected by this process.

The RGB-to-YIQ conversion is defined as:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

In order to obtain the RGB values from a set of YIQ values, we simply perform the inverse matrix operation.

Chapter 3

Model-based Pose Estimation

3.1 Previous Work

In motion analysis, a sequence of images about an object is given as input, and one would try to recover the motion of the object or the camera throughout the sequence, depending on which quantity we are interested in. Numerous application, e.g. cartography, object recognition, etc. make use of the motion information extracted. In general, two kinds of motion can be recovered. The first one is the movement of individual pixels on the image, which is a planar (2D) one. The other one tells how a point in the scene, which corresponds to a pixel in the image, moves in the 3D space. Much research work on this problem has been reported in the literature [34], especially on the recovery of camera orientation and position (known as extrinsic camera calibration). The first effort on solving model-based motion estimation problems by computer vision researchers is probably due to Fischler and Bolles [19]. They also coined the term Perspective-n-Point (PnP) to designate the problem of determining the pose of a rigid object with respect to the camera, given N pairs of correspondences between model and image features.

Existing approaches can be categorized into two main areas (Fig. 3.1): 1) Estimation from established correspondences between model features like vertices, edges and angles (section 3.1.1), and 2) Direct estimation from image intensities (section 3.1.2). Section 3.2 describes our proposed algorithm to P3P problems, solution constraints, and the way to recover motion parameters. Experimental results and discussions based on real images and synthetic data are given in section 3.3.

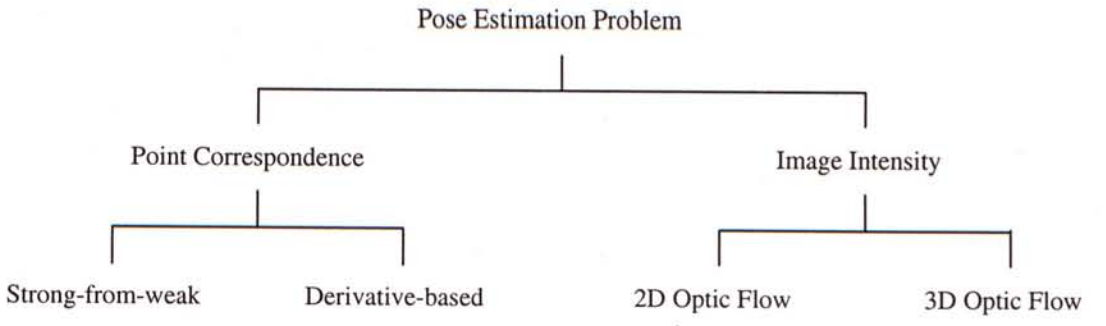

Figure 3.1: Classification of existing pose estimation approaches.

3.1.1 Estimation from Established Correspondences

As discussed in the last chapter, using linearized camera models is an effective way of ameliorating problems that arise with closed-form perspective solutions (such as ambiguity and ill-conditioned error propagation). Furthermore, these models allow one to deal with over-constrainted problems, in which the number of restrictions is possibly much bigger than that of free parameters in the scene, to average out measurement errors and quantization noise in individual features. So when a big number of feature correspondences can be reliably established, solutions based on these models tend to be more robust. Unfortunately, these models are approximations which validity is sometimes questionable in practice. An ideal pose recovery algorithm should combine the generality of a perspective camera model with the robustness of affine approximations. Indeed, it is possible to satisfy this requirement in practice by casting the problem of pose recovery into an equivalent multivariate numerical optimization problem. There are at least two distinct ways in which this can be done.

The most traditional, straightforward, and widely-used approach is to define a global measure for the discrepancy between actual images and predicted images from a perspective camera model and estimates for the unknown poses. Then, by replacing the chosen error measure (which may be a nonlinear function of the pose parameters) with a local linear or low-degree-polynomial approximation, one can compute a correction that in general yields a better pose estimate. This process can be iterated until the error function is locally minimized or the pose estimate converges with a desired precision. As this method involves evaluating first and possibly higher-order derivatives of the error function, we call this approach *derivative-based*.

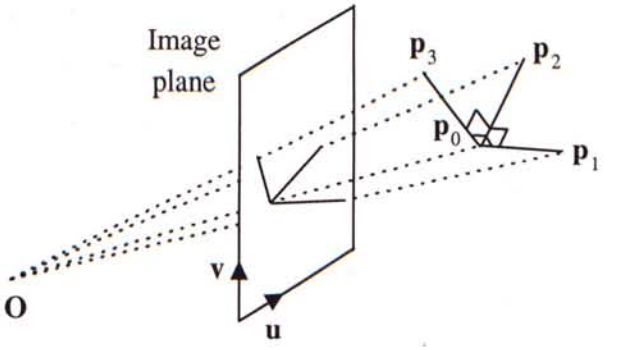

Figure 3.2: Pre-engineered target, with three right angles between edges of unit length.

Another alternative consists of rewriting the perspective projection equations (Eq. (2.3)) as a function of a set of parameters (one for each feature) that explicitly denote the discrepancy between the image predicted by the perspective model and that predicted by some affine approximations. Then, by artificially setting all these parameters to zero, one can compute the pose based on the linearized model. The result is used to compute better values for the discrepancy parameters, which can be in turn used to compute a new pose and so on. Again, after some iterations of this process, the estimated pose will theoretically converge to its true value, within a certain tolerance margin. We refer them as *strong-from-weak* pose methods.

An Example of Weak-Perspective Pose Recovery

Before we discuss the techniques designed to work with arbitrary objects, let us consider a very simple and efficient trick that involves some special arrangements on the target object. Suppose the four vertices $\mathbf{p}_i$, $i = 0, ..., 3$, of a trihedral in the 3D space can be established with three right angles and unit edges ($|\mathbf{p}_i\mathbf{p}_j| = 1$, $i \neq j$) (Fig. 3.2), their corresponding image points $\mathbf{q}_i = \begin{bmatrix} u_i & v_i \end{bmatrix}^T$ can be thought of as projections of the unit vectors that define the coordinate system associated with the model space.

The key idea is that the weak-perspective camera projection is equivalent to a composition of rotation and translation, that preserve the orthonormality of the trihedral vertices; followed by an scaling operation that changes the overall orthonormality by a single multiplicative factor; and the elimination of z coordinates. Denoting the image of the edge intersection point as $\mathbf{q}_0$, if the object coordinate system is identified with

the edges of the trihedral, two of the three rows in the rotation matrix between this system and the camera frame can be recovered up to a scale factor with subtractions only, as shown in Eq. (3.1). Then according to the property of orthonormal matrices, the third row can be determined by normalizing $\mathbf{r}'_x$ and $\mathbf{r}'_y$, together with a cross product operation on the resulting vectors, giving $\mathbf{R}_{weak} = \begin{bmatrix} \mathbf{r}_x & \mathbf{r}_y & \mathbf{r}_z \end{bmatrix}^T$.

$$\begin{bmatrix} \mathbf{r}'_x \\ \mathbf{r}'_y \end{bmatrix} = \begin{bmatrix} u_1 - u_0 & u_2 - u_0 & u_3 - u_0 \\ v_1 - v_0 & v_2 - v_0 & v_3 - v_0 \end{bmatrix} \tag{3.1}$$

$$\begin{aligned} \mathbf{r}_x &= \frac{\mathbf{r}'_x}{|\mathbf{r}'_x|} \\ \mathbf{r}_z &= \mathbf{r}_x \times \frac{\mathbf{r}'_y}{|\mathbf{r}'_y|} \\ \mathbf{r}_y &= \mathbf{r}_z \times \mathbf{r}_x \end{aligned} \tag{3.2}$$

However, this type of target pre-engineering is impractical for most application because it requires precise positioning of feature points on the object (three right angles and unit edges). Other interesting approaches were also proposed to solve the pose estimation problem. Gee and Cipolla [22] suggested a weak perspective solution for the pose estimation problem of a human face. By taking a few assumptions about the facial parameters, they showed that the gaze of a human can be readily resolved by only little calculations. This method, though limited in application, highlights the usefulness of the weak perspective projection model in pose estimation.

Strong-from-weak Pose Methods

The idea of applying a coarse-to-fine analysis to pose estimation problems was ffrst proposed by DeMenthon and Davis [17]. Its main theme is that a rough solution to the problem is first estimated with minimal effort, which is then refined in subsequent steps consisting of more computational efforts. Specifically, their scheme starts with an initial guess obtained from a weak perspective projection model, and in later steps, the estimate is iteratively refined by a full perspective projection model.

This method has several advantages when compared with other approaches: It solves the problem of initial guesses in pose estimation problems, and at the same time reduces the computation to a minimum. Significant efficiency is gained which

saves the algorithm from iterating in slow steps near the solution. Another benefit is that the resulting algorithm is more stable against noise inherent in measurement processes. The iteration will not easily get stuck with local minima.

Mathematically, as shown in Eq. (2.4), the perspective equations can be manipulated so as to isolate the non-linearity of the model in terms of $\frac{1}{1+\epsilon_i}$, where $\epsilon_i = \frac{\mathbf{p}_i \cdot \mathbf{r}_z}{t_z}$. As we are interested in the images of the edges $\mathbf{e}_i = \mathbf{p}_0\mathbf{p}_i$, rather than those of individual points, Eq. (2.4) is rewritten as

$$\begin{aligned} \mathbf{e}_i \cdot \mathbf{r}'_x &= u_i(1+\epsilon_i) - u_0 \quad (1 \le i \le N) \\ \mathbf{e}_i \cdot \mathbf{r}'_y &= v_i(1+\epsilon_i) - v_0 \quad (1 \le i \le N) \end{aligned} \tag{3.3}$$

where

$$\epsilon_i = \mathbf{e}_i \cdot \mathbf{r}'_z \tag{3.4}$$

$$\begin{bmatrix} \mathbf{r}'_x & \mathbf{r}'_y & \mathbf{r}'_z \end{bmatrix}^T = \frac{\begin{bmatrix} \mathbf{r}_x & \mathbf{r}_y & \mathbf{r}_z \end{bmatrix}^T}{t_z} = \frac{\mathbf{R}}{t_z} \tag{3.5}$$

$$\mathbf{t} = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T$$

and N is the number of model points.

Notice that this formulation allows one to separate completely the recovery of $\mathbf{R}$ and t_z from that of the translation components parallel to image plane (t_x and t_y). But the crucial property is that, if the actual imaging process can at least be reasonably approximated by a weak-perspective transformation (so that the values of all parameters ϵ_i are assumed to be fixed), Eq. (3.3) will become a linear system involving only the six unknown values for the elements of $\mathbf{r}'_x$ and $\mathbf{r}'_y$. Then $\mathbf{r}'_x$ and $\mathbf{r}'_y$ can be recovered up to a scale factor from Eq. (3.3), and $\mathbf{r}_x$, $\mathbf{r}_y$ and $\mathbf{r}_z$ can be solved according to Eq. (3.2). Although this weak-perspective solution is usually unacceptable as a final answer, it is good enough to serve as an initial estimate for the full-perspective pose.

In their algorithm, DeMenthon and Davis [17] started by assuming that $\forall i \ \epsilon_i = 0$ (so that $\frac{1}{1+\epsilon_i} \approx 1$) to convert Eq. (3.3) into the imaging equations of a weak-perspective camera model:

$$\mathbf{e}_i \cdot \mathbf{r}'_x = u_i - u_0 \quad (1 \leq i \leq N)$$
$$\mathbf{e}_i \cdot \mathbf{r}'_y = v_i - v_0 \quad (1 \leq i \leq N) \tag{3.6}$$

and solve for a rough solution as described above. Then, they substitute the estimated $\mathbf{r}_z$ and t_z into Eq. (3.4) and Eq. (3.5), in order to recover the values of parameters in ϵ_i. Finally, a new pose estimate is computed from Eq. (3.3) using the presumably improved parameters ϵ_i. This cycle is repeated until either the pose estimate converges, or a predefined number of iterations has exceeded (in which case it is assumed that the method converges). After $\mathbf{R}$ and t_z have been found in this way, the computation of t_x and t_y is trivial. However, this formulation does not enforce the orthonormality of the recovered rotation matrix. Violation of such constraint would result in slight deformation of the transformed point set, which would propagate to later iterations and cause erroneous estimation.

On the other hand, the system of equations in Eq. (3.3) can be rewritten as:

$$\mathbf{E}\mathbf{r}'_x = \mathbf{d}_u$$
$$\mathbf{E}\mathbf{r}'_y = \mathbf{d}_v \tag{3.7}$$

where

$$\mathbf{E} = [\mathbf{e}_1 ... \mathbf{e}_N]^T$$
$$\mathbf{d}_u = [u_1 - u_0 ... u_N - u_0]^T$$
$$\mathbf{d}_v = [v_1 - v_0 ... v_N - v_0]^T$$

If the pose recovery problem is over-determined ($N > 3$) and at least three model points $\mathbf{p}_i$ are non-coplanar, the rank of $\mathbf{E}$ is three, and the system in Eq. (3.7) can be solved in a least-square sense through the computation of matrix pseudo-inverse (denoted $\mathbf{E}^\dagger$). However, as $\mathbf{E}$ depends only on the model for a specific target but not on any pose estimates or ϵ_i parameters, $\mathbf{E}^\dagger$ can actually be computed off-line for each possible target used. This results in a very fast and efficient inner loop in the numerical routine, although the *apriori* computation of pseudo-inverse is relatively expensive. In fact, this preprocessing step can be carried out even if the problem is

not over-constrainted. This seems to be the main advantage of the methods based on the refinement of weaker models.

A problem with such methods is that, when $\mathbf{p}_i$ are coplanar, the rank of $\mathbf{E}$ drops to two and it is no longer possible to compute a unique solution for Eq. (3.7). Actually, even if $\mathbf{p}_i$ are only nearly coplanar, the proximity of this singularity can lead to problems of numerical instability, resulting either in divergence or a very slow convergence of the pose recovery algorithm. This kind of situation arises very frequently in cartography, in which the variance on the height of a set of landmarks is much smaller than the average pair-wise distance between these landmarks.

Oberkampf *et al.* [62] later proposed an extension to the basic full-from-weak-perspective pose recovery algorithm to handle the above singularity. They notice that, in those cases, the pseudo-inverse solution for Eq. (3.7) is such that $\mathbf{r}'_x$ is parallel to the plane that contains the target, and its head is allocated on the point $\mathbf{p}_{min}$ with minimal distances (in a least-square sense) from the restrictive planes defined by the N instances of Eq. (3.3) (again, assuming that the tail of $\mathbf{r}'_x$ is located at $\mathbf{p}_0$). Of course, this solution is not unique, since any point located on the line that contains $\mathbf{p}_{min}$ and is normal to the object plane will also be at a minimal distance from the restrictive planes, as illustrated in Fig. 3.3.

Analogous reasoning is valid for $\mathbf{r}'_y$. So, a generic solution to the undetermined system defined by Eqs. (3.7) and (3.3) can be parameterized with two unknowns λ and μ as follows:

$$\mathbf{r}'_x = \mathbf{r}'_{0x} + \lambda \mathbf{n}$$
$$\mathbf{r}'_y = \mathbf{r}'_{0y} + \mu \mathbf{n}$$

where $\mathbf{r}'_{0x} = \mathbf{E}^{\dagger}\mathbf{d}_u$ and $\mathbf{r}'_{0y} = \mathbf{E}^{\dagger}\mathbf{d}_v$ as a result of Eq. (3.7), and $\mathbf{n}$ is the unit vector normal to the object plane. Actually, in order to extend this analysis to the cases where the target is only nearly planar, $\mathbf{n}$ can be defined as the unit vector of the null space of the matrix $\mathbf{E}$, which can be computed from an off-line *singular value decomposition* (SVD) of $\mathbf{E}$.

Then, one can use the restriction that $\mathbf{r}'_x$ and $\mathbf{r}'_y$ must be perpendicular and must have the same norm, in order to derive the following system:

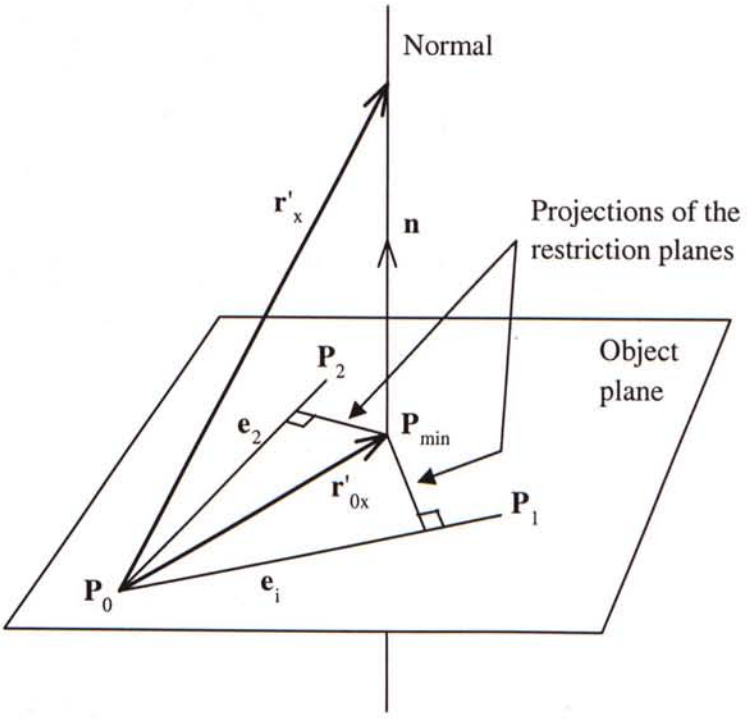

Figure 3.3: Geometry of Oberkampf's solution to the perspective pose recovery problem with planar targets.

$$\lambda\mu = -\mathbf{r}'_x \cdot \mathbf{r}'_y$$
$$\lambda^2 + \mu^2 = \mathbf{r}'^2_x - \mathbf{r}'^2_y$$

An algebraic solution of this system yields two pairs of values for λ and μ (and thus to $\mathbf{r}'_x$ and $\mathbf{r}'_y$). Oberkampf *et al.* suggest that this ambiguity is actually desirable because it is present when a weak-perspective camera is used, and thus it should be taken in account whenever weak-perspective is a good approximation for full perspective. However, the exploration of both solutions in an iterative procedure would result in an exponential growth of the execution time with respect to the number of iterations.

Another problem with the original solution proposed by DeMenthon and Davis is that it tends to have convergence problems if the angle between the optical axis and the line-of-sight of $\mathbf{p}_0$ is relatively large. The problem is that in this situation, the weak-perspective model is in general not a good approximation to full perspective. In order to ameliorate this, Horaud *et al.* [28] suggested an alternative technique that starts with a paraperspective (rather than weak-perspective) camera model that

is then numerically refined until it approximates full perspective with the desired precision.

Recall from the previous chapter that using a paraperspective camera model is equivalent to replacing the factor $\frac{1}{1+\epsilon_i}$ in the full-perspective projection equations with its first order approximation $1-\epsilon_i$, and then neglecting the resulting terms that depend only on $\frac{\mathbf{p}^2}{t_z^2}$. If we apply this transformation directly to Eq. (3.3), we obtain the following equations:

$$\begin{aligned}
\mathbf{e}_i \cdot \mathbf{r}'_x - u_0\epsilon_i &= u_i - u_0 \quad (1 \leq i \leq N) \\
\mathbf{e}_i \cdot \mathbf{r}'_y - v_0\epsilon_i &= v_i - v_0 \quad (1 \leq i \leq N)
\end{aligned} \tag{3.8}$$

Comparing the expressions above with the weak-perspective model defined in Eq. (3.6), the only difference is that the paraperspective model introduces the extra term $-u_0\epsilon_i$ on the left-hand-side of the projection equations. If we add this term to both sides of the perspective Eq. (3.3), the result will still be a perspective camera model:

$$\begin{aligned}
\mathbf{e}_i \cdot \mathbf{r}'_x - u_0\epsilon_i &= u_i(1+\epsilon_i) - u_0 - u_0\epsilon_i \quad (1 \leq i \leq N) \\
\mathbf{e}_i \cdot \mathbf{r}'_y - v_0\epsilon_i &= v_i(1+\epsilon_i) - v_0 - v_0\epsilon_i \quad (1 \leq i \leq N)
\end{aligned}$$

Now, we substitute the expression that dictates the correct value for ϵ_i (Eq. (3.4)) only on the left-hand-side in the above equations, and rearrange the resulting formula as:

$$\begin{aligned}
\mathbf{e}_i \cdot \mathbf{r}''_x &= (u_i - u_0)(1+\epsilon_i) \quad (1 \leq i \leq N) \\
\mathbf{e}_i \cdot \mathbf{r}''_y &= (v_i - v_0)(1+\epsilon_i) \quad (1 \leq i \leq N)
\end{aligned} \tag{3.9}$$

where

$$\begin{aligned}
\mathbf{r}''_x &= \frac{\mathbf{r}'_x - u_0\mathbf{r}_z}{t_z} \\
\mathbf{r}''_y &= \frac{\mathbf{r}'_y - v_0\mathbf{r}_z}{t_z}
\end{aligned}$$

Notice that if the correct values for ϵ_i are used on the right-hand-side, Eq. (3.9) still define a full-perspective camera model. But, on the other hand, if we artificially

impose that $\forall i \ \epsilon_i = 0$ on the right-hand-side, the system above will be equivalent to Eq. (3.8), which define a paraperspective model. So, we can apply the same basic iterative procedure suggested by DeMenthon and Davis, using Eq. (3.9) instead of Eq. (3.3). Horaud *et al.* [28] also suggested a very elegant way of ensuring the orthonormality of $\mathbf{R}$, and their solution is easily generalizable to over and under-constrainted instances.

Derivative-based Methods

It is difficult to determine where did the idea of derivative-based numerical optimization first appear in pose recovery literature, but the most influential technique under this category seems to be the one proposed by Lowe [50][51][52].

Lowe's Algorithm Conceptually, Lowe's method is quite simple, yet extremely elegant and general. Contrary to the techniques discussed in the previous section, this iterative algorithm assumes from the beginning that the imaging transformation is a perspective projection. The key idea to overcome the non-linearity of this camera model is that, given an initial guess for some unknown scene parameters $\mathbf{s} = \begin{bmatrix} s_0 & s_1 & \dots \end{bmatrix}^T$ - including the six pose parameters, some intrinsic camera parameters (e.g. the focal length f), and some internal DOF in the scene - one can reproject the known 3D scene model onto the image plane, using the current parameter estimates and perspective equations in Eq. (2.3), and then compute a vector $\mathbf{d} = \begin{bmatrix} d_0 & d_1 & \dots \end{bmatrix}^T$ of errors between the transformed image points (their positions, orientations and apparent angles) and the corresponding actual measurements.

Rather than solving directly for the vector of parameters $\mathbf{s}$ in a nonlinear system, Lowe applied Newton's method to iteratively compute a series of parameter correction vectors $\mathbf{s}'$ that are successively subtracted from the current estimate $\mathbf{s}$, until either this estimate converges to a point that minimizes $\mathbf{d}$ locally, or the maximum number of iterations allowed is exceeded. Assuming $\mathbf{d}$ is locally linear in the elements of $\mathbf{s}$, the effect of each parameter correction on an error measurement can be determined by solving the following equation from Taylor series expansion about the parameter vector:

$$\begin{bmatrix} \mathbf{J} \\ \mathbf{W} \end{bmatrix} \mathbf{s}' = \begin{bmatrix} -\mathbf{d} \\ 0 \end{bmatrix} \tag{3.10}$$

where $\mathbf{J}$ is an $N \times M$ Jacobian matrix (N being the number of points considered and M is the number of parameters when solving the above equation) of the error d_i with respect to parameter s_j

$$J_{ij} = \frac{\partial d_i}{\partial s_j}$$

and $\mathbf{W}$ is a diagonal matrix which stabilizes the solution. If $\mathbf{s}^{(i)}$ is the pose estimate on iteration i, then $\mathbf{s}^{(i+1)}$ is obtained by:

$$\mathbf{s}^{(i+1)} = \mathbf{s}^{(i)} - \mathbf{s}'^{(i)}$$

When the system in Eq. (3.10) is over-determined, a pseudo-inverse solution that minimizes $\mathbf{d}$ in a least-square sense can be computed [52], in a similar way to the techniques presented in the last section. However, in the case of gradient-based methods, $\mathbf{J}$ depends on the current value of $\mathbf{s}$, and is not constant along different iterations of the numerical optimization procedure. So in this case, it is not possible to compute the pseudo-inverse matrix off-line, which is a major performance drawback compared with the strong-from-weak pose estimation techniques. But on the other hand, the strong-from-weak techniques cannot be generalized to deal with intrinsic camera parameters of internal DOF in the target. There is a trade-off between efficiency and generality involved in the choice between these two paradigms.

Lowe suggested that, in order to achieve greater efficiency, one should reformulate the translational components of the pose, so as to express them directly in image plane coordinates, rather than in a 3D Euclidean space. More specifically, in his simplified projective model, the image coordinates of an arbitrary feature $\begin{bmatrix} u & v \end{bmatrix}^T$ are expressed as a function of the corresponding model-space coordinates $\mathbf{p}$ by the following equations:

$$\begin{bmatrix} x' & y' & z' \end{bmatrix}^T = \mathbf{R}\mathbf{p}$$

$$\begin{bmatrix} u & v \end{bmatrix} = \begin{bmatrix} \frac{fx'}{z'+t'_z} + t'_x & \frac{fy'}{z'+t'_z} + t'_y \end{bmatrix}$$

where the unknown vector $\mathbf{t}$ in Eq. (2.3) (defined in the camera coordinate frame) has been replaced by the parameters t'_x, t'_y and t'_z. In this new parameterization, t'_x

	x	y	z
ϕ_x	0	$-z'$	y'
ϕ_y	z'	0	$-x'$
ϕ_z	$-y'$	x'	0

Table 3.1: Partial derivatives of x, y and z with respect to anti-clockwise rotation ϕ (in radians) about the coordinate axes of the camera reference system.

and t'_y specify the location of the object on image plane and t'_z specifies the distance of the object from camera. They are equivalent when:

$$\mathbf{t} = \mathbf{R}^{-1} \begin{bmatrix} -\frac{t'_x(z'+t'_z)}{f} & -\frac{t'_y(z'+t'_z)}{f} & -t'_z \end{bmatrix}^T \tag{3.11}$$

To compute the partial derivatives of the error with respect to the rotation angles (ϕ_x, ϕ_y and ϕ_z are the rotation angles about the fixed x, y and z-axes, respectively, in the camera frame) it is necessary to calculate the partial derivatives of x, y and z with respect to these angles. Table 3.1 gives these derivatives for all combinations of variables.

Newton's method is carried out by calculating the optimum correction rotation $\Delta\phi_x$, $\Delta\phi_y$ and $\Delta\phi_z$ to be made about the camera-centered axes. Given Lowe's parameterization, the partial derivatives of u and v with respect to each of the seven parameters of the imaging model (including f) are given in Table 3.2.

where

$$c = \frac{1}{z' + t'_z}$$

Lowe then noted that each iteration of the multi-dimensional Newton's method solves for a vector of corrections:

$$\mathbf{s}' = \begin{bmatrix} \Delta t'_x & \Delta t'_y & \Delta t'_z & \Delta\phi_x & \Delta\phi_y & \Delta\phi_z \end{bmatrix}^T$$

The u and v-components of the error $\mathbf{d}$ can be used independently to create separate linearized constraints. For example, we can create an equation that expresses d_{iu}

	u	v
t'_x	1	0
t'_y	0	1
t'_z	$-fc^2x'$	$-fc^2y'$
ϕ_x	$-fc^2x'y'$	$-fc(z'+cy'^2)$
ϕ_y	$fc(z'+cx'^2)$	$fc^2x'y'$
ϕ_z	$-fcy'$	fcx'
f	cx'	cy'

Table 3.2: Partial derivatives of u and v with respect to each of the camera viewpoint parameters and the focal length, according to Lowe's original approximation.

(the u-component of d_i) as the sum of products of its partial derivatives times the unknown error-correcting values:

$$\frac{\partial u}{\partial t'_x}\Delta t'_x + \frac{\partial u}{\partial t'_y}\Delta t'_y + \frac{\partial u}{\partial t'_z}\Delta t'_z + \frac{\partial u}{\partial \phi_x}\Delta\phi_x + \frac{\partial u}{\partial \phi_y}\Delta\phi_y + \frac{\partial u}{\partial \phi_z}\Delta\phi_z = d_{iu}$$

In this way, we can derive six equations from three point correspondences and produce a complete linear system.

A problem with this formulation is that t'_x, t'_y and t'_z (assumed to be constants that are determined by the iterative procedure) are in fact dependent variables. One can easily arrive at this conclusion by substituting $\mathbf{R} = \begin{bmatrix} \mathbf{r}_x & \mathbf{r}_y & \mathbf{r}_z \end{bmatrix}^T$ into Eq. (3.11) to obtain

$$\begin{aligned} t'_x &= -f\frac{\mathbf{t}\cdot\mathbf{r}_x}{\mathbf{p}\cdot\mathbf{r}_z + t'_z} \\ t'_y &= -f\frac{\mathbf{t}\cdot\mathbf{r}_y}{\mathbf{p}\cdot\mathbf{r}_z + t'_z} \\ t'_z &= -\mathbf{t}\cdot\mathbf{r}_z \end{aligned} \tag{3.12}$$

This shows that while t'_z only depends on the pose parameters, t'_x and t'_y are also a function of $\mathbf{p}$ (coordinates of points in the object coordinate frame). If no restrictions are imposed, $\mathbf{p}$ can have a large variance. Even if they do not, the term $\mathbf{p}\cdot\mathbf{r}_z$ may change significantly (due to the object geometry) and affect the estimation process.

	u	v
t''_x	fc	0
t''_y	0	fc
t''_z	$-fac^2$	$-fbc^2$
ϕ_x	$-fac^2y'$	$-fc(z'+bcy')$
ϕ_y	$fc(z'+acx')$	fbc^2x'
ϕ_z	$-fcy'$	fcx'
f	ac	bc

Table 3.3: Partial derivatives of u and v with respect to each of the camera viewpoint parameters and the focal length according to Araujo's full projective solution.

Therefore it is in general impossible to find a single consistent value for either t'_x or t'_y, and we cannot use t'_x and t'_y as defined in Eq. (3.12). Another concern is that the results of this algorithm depend heavily on the quality of initial guesses supplied.

Improvements to Lowe's Algorithm Hoping to improve the accuracy of those approximations, Araujo *et al.* [9] proposed a reformulation for Lowe's original algorithm by eliminating the approximations with

$$\begin{bmatrix} t''_x & t''_y & t''_z \end{bmatrix} = \begin{bmatrix} \mathbf{t}\cdot\mathbf{r}_x & \mathbf{t}\cdot\mathbf{r}_y & \mathbf{t}\cdot\mathbf{r}_z \end{bmatrix} \tag{3.13}$$

$$\begin{bmatrix} u & v \end{bmatrix} = \begin{bmatrix} f\frac{x'+t''_x}{z'+t''_z} & f\frac{y'+t''_y}{z'+t''_z} \end{bmatrix} \tag{3.14}$$

The partial derivatives of u and v with respect to each of the six pose parameters and f are then given by Table 3.3, where $\begin{bmatrix} a & b & c \end{bmatrix} = \begin{bmatrix} x'+t''_x & y'+t''_y & \frac{1}{z'+t''_z} \end{bmatrix}$. As in Lowe's formulation, the translation vector is computed in the object coordinate frame using Eq. (3.11), with t''_x, t''_y and t''_z as defined in Eq. (3.13). The minimization process yields estimates of t''_x, t''_y and t''_z, which are the result of the product of the rotation matrix by the translation vector.

On the other hand, they also proposed a numerically equivalent but conceptually more elegant way of looking at this solution. Through a redefinition of the image formation process,

$$\begin{bmatrix} x & y & z \end{bmatrix}^T = \mathbf{R}\mathbf{p} + \mathbf{t}$$

Eqs. (3.13) and (3.14) are substituted by:

$$\begin{bmatrix} t''_x & t''_y & t''_z \end{bmatrix}^T = \mathbf{t}$$

$$\begin{bmatrix} u & v \end{bmatrix}^T = \begin{bmatrix} f\frac{x'+t_x}{z'+t_z} & f\frac{y'+t_y}{z'+t_z} \end{bmatrix}^T$$

In this case, $\mathbf{R}$ and $\mathbf{t}$ are explicitly decoupled, and the least-square minimization procedure gives the estimates of $\mathbf{t}$ directly. They claimed that the use of a full-perspective imaging model and the relaxation of mathematical approximations in Lowe's algorithm yield a much more accurate numerical technique, with super-exponential convergence for a wide range of initial conditions, but with no significant increase in the computational cost.

3.1.2 Direct Estimation from Image Intensities

Estimation of motion information directly from image intensity values has also been a well researched topic in photogrammetry for years. It relies on the assumption that the intensity value at a feature point should be relatively constant, so that after some object movements, we can still find the point (with nearly the same intensity level) but probably at another location. Broadly speaking, it can be divided into two classes of problems: 2D and 3D.

Finding 2D Motion by Optic Flow

The first class of problem estimates the motion of a particular pixel as viewed on an image plane (this 2D pixel motion on image plane is the result of camera projection of the 3D object movement), and is also known as image matching or image registration. Numerous literature is available, and most of them are based on the intensity comparison between a small patch (e.g. 7×7 pixels) in one image with another patch at the corresponding position in another image. Mathematically, this can be written as:

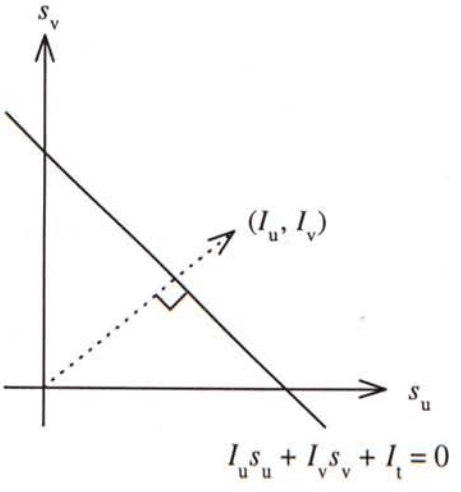

Figure 3.4: The aperture problem in optical flow estimation.

$$\begin{aligned} I(u,v,t) &= I(u+\delta u, v+\delta v, t+\delta t) \\ &= I(u+s_u\delta t, v+s_v\delta t, t+\delta t) \end{aligned}$$

where $\mathbf{s} = \begin{bmatrix} s_u & s_v \end{bmatrix}^T$ is the horizontal and vertical image velocity at an image point $\mathbf{q} = \begin{bmatrix} u & v \end{bmatrix}^T$ at time t, and δt is infinitesimal increment in time. By taking Taylor series expansion of the right hand side and simplifying, one would get the standard optical flow constraint proposed by Horn [32]:

$$I_u s_u + I_v s_v + I_t = 0$$

However, when the above equation is plotted on the optical flow $s_u - s_v$ plane (Fig. 3.4), the optical flow restored are restricted to lie along the line of optic flow constraint and the components perpendicular to this line cannot be recovered without additional constraint. This phenomenon is called the *aperture problem* in computer vision.

To handle this problem, Horn [31] used a Lagrange multiplier approach to regularize the ambiguity in the solution. In addition, Lucas and Kanade [54] discussed the SSD measure for estimating image points movements. In this approach, a small region about the interested point is sampled and a 2D translation of the region is found to minimize the intensity difference:

$$E(s_u, s_v) = \sum_i [I(u, v, t) - I(u + s_u\delta t, v + s_v\delta t, t + \delta t)]^2$$

However, different types of erroneous solutions may occur when the movement of individual pixel is large, the scene has significant depth variation, or too large a correlation window is used. Shi and Tomasi [73] later proposed a solution to this problem which can switch between solving the affine transform and simple translational displacement of features under different window sizes. On the other hand, in order to accommodate larger motions in the image plane, a coarse-to-fine strategy is often used [7]. In that representation, a spatial pyramid of three to four levels is constructed with the coarsest resolution at top level, and highest one at the bottom. Separate processing is carried out at each of these levels and the results are propagated to the lower ones by projective interpolation. However, such a simple approach lacks the ability to revert from wrong estimates at higher levels. Any erroneous estimate at the coarse level just propagates down to the finest scales with no way for correction.

Finding 3D Motion by Optic Flow

Inferring 3D motion from intensity values is much more difficult as 3D information is projected nonlinearly on an image plane, and depth is not available in general. In their classic paper [49], Longuet-Higgins and Prazdny introduced an equation for the 2D image motion of a point from its 3D rotational and translational components. This equation was used by a number of researchers. Horn and Weldon [33] proposed another technique which can directly recover the 3D motion of a camera as well as the surface from intensity images by the brightness constancy constraint. To overcome the restriction that little information is available in only a pair of images, Hanna [25] suggested a method which can estimate the surface description from the motion and stereo data from one or more cameras. It is a consensus now that more frames are needed to reliably recover the motion as well as the structure from past experiences [63].

3.1.3 Perspective-3-Point Problem

Pose estimation is important in many fields like robot navigation, motion tracking, object recognition and camera calibration. Some researchers have considered the P3P

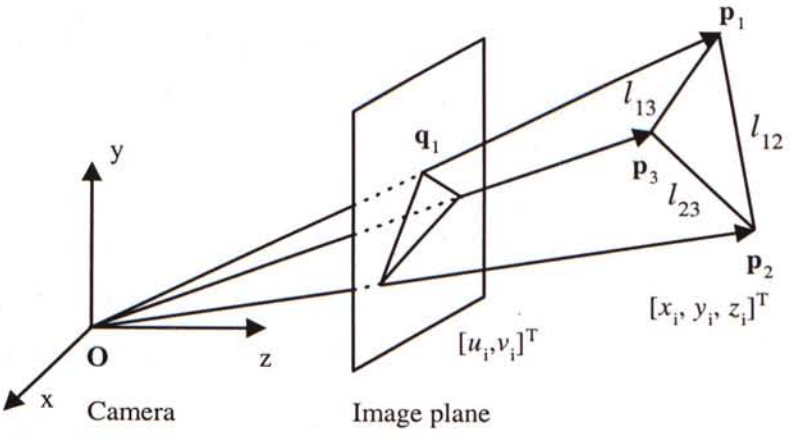

Figure 3.5: A typical configuration of P3P problem.

problem [19] because three point correspondences is the minimal information leading to a finite number of solutions. A P3P problem refers to the task of finding the position and orientation of a rigid body given the perspective projections of three corresponding points on the object. As discussed in Chapter 2, the vertices of an object in Fig. 3.5 are given by $\mathbf{p}_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^T$, where $i = 1, ..., 3$ is an index to the three feature points. The non-zero Euclidean norms of $\mathbf{p}_1\mathbf{p}_2$, $\mathbf{p}_1\mathbf{p}_3$ and $\mathbf{p}_2\mathbf{p}_3$ are known in advance, and to further simplify our notations, they are denoted by l_{12}, l_{13} and l_{23} respectively so that

$$\begin{aligned} l_{12}^2 &= (x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2 \\ l_{13}^2 &= (x_1 - x_3)^2 + (y_1 - y_3)^2 + (z_1 - z_3)^2 \\ l_{23}^2 &= (x_2 - x_3)^2 + (y_2 - y_3)^2 + (z_2 - z_3)^2 \end{aligned} \tag{3.15}$$

Using the full perspective projection model, these points are projected onto an image plane at $z = f$ giving $\mathbf{q}_i = \begin{bmatrix} u_i & v_i \end{bmatrix}^T$, so that each image point constraints the position of the corresponding model point to lie in a line-of-sight with respect to the camera, e.g. $\mathbf{Op}_1$. Furthermore, the unit vectors along these lines-of-sight, with respect to the camera frame, can be determined directly from the image coordinates of each point, and the angles between each pair of these unit vectors can thus be obtained with a single extra dot-product operation.

With all these definitions, the problem can be stated formally as:

> Given camera focal length f, Euclidean norms among three arbitrary object points in the scene $|\mathbf{p}_i\mathbf{p}_j|$ $(i \neq j, |\mathbf{p}_i\mathbf{p}_j| \neq 0)$, and corresponding image points $\mathbf{q}_i$ at time t and $\mathbf{q}'_i$ at time $t+1$ under the full perspective projection model, estimate the depths for $\mathbf{p}_i$ and $\mathbf{p}'_i$; and based on these approximates, infer the positions and orientations of the object in the 3D space, and hence the extrinsic motion parameters, i.e. rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$, of this object with respect to the imaging system during the time interval t and $t+1$.

Actually, the P3P problem was the subject of several studies published mostly in Germany between 1841 and 1949. According to a survey paper by Haralick *et al.* [26] on the empirical comparison of all major solutions known for this problem, the solution with the best error propagation properties in general is the one due to Finsterwalder (1903). They further commented that as Finsterwalder's solution amounts to solving a cubic and two quadratic equations rather than a quartic one (to be described in greater details below), it seems to be more stable than the solutions found in the computer vision literature of the 1980s.

Finsterwalder used cosine rule to get a system of equations, and reduced the 3×3 system to an equivalent 2×2 nonlinear system by means of a change of variables. He multiplied one of the resulting equations by an additional parameter λ and then expressed one of the transformed variables as a function of the other transformed variable and λ. Then, he tried to determine a value for λ that made the resulting relation between the two transformed variables linear. This amounted to solving a cubic equation, and substitution of the resulting linear relation in the 2×2 system returned a quadratic equation involving only one of the transformed variables. This formulation has a total of four possible solutions, because for a particular value of λ one would get two quadratic equations, each of which yields two possibly different and valid solutions.

Being unaware of the previous work by Finsterwalder *et al.*, Fischler and Bolles [19] proposed their own solutions in 1981. Geometrically, the idea is similar to the previous approach: to use cosine rule to get a system of equations. The difference lies on the algebraic manipulation that follows. Rather than making a cubic and two quadratic equations out of the 3×3 system, they tried to derive a single polynomial equation for one of the unknowns. To combat noise, they extended the above approach by using statistical methods to reach a consensus of sampling within the estimated point

set. Their approach, called *random sample consensus* (RANSAC), is robust enough to reject outliers so that the final solution is much more reliable, at the expense of intensive computation that prevents it from being used in real time application. Linnainmaa *et al.* [44] later showed that two of the three unknowns in the system above can be eliminated yielding an eighth-degree polynomial equation with terms of even order only. By solving this new system of equations and substituting back for the other two unknowns, one can recover the positions of the three vertices in the 3D camera frame.

Among the many iterative solutions to this problem, most of which make use of the weak-perspective projection to reduce the three unknown depths z_i into image plane coordinates u_i and v_i. Then one of these two variables is further eliminated by expressing it in terms of the other to give a quartic equation. Various mathematical techniques are then used to solve these equations [26]. Huttenlocher and Ullman [36] showed that with three point correspondences in a general position, it is possible to recover a unique weak-perspective transformation corresponding to the pose, up to a reflection. Alter [2] also achieved similar results but in a simpler and more elegant way. However, it is clear that the weak perspective assumption is valid only when the distance between the object and the camera is much larger than the relative distances between feature points on the object. DeMenthon and Davis [16] further showed that orthographic or paraperspective projection models produce smaller approximation errors than weak perspective, especially for off-centered triangles, and that 2D lookup tables can be precomputed to reduce the number of runtime floating point operations. In addition, Haralick *et al.* [26] pointed out that the precision of P3P methods can vary up to three orders of magnitude depending on the order in which the correspondences are used. So, they suggested some heuristics to determine the best order, that seem to work very well in most cases. However, their comparison of P3P algorithms took into account only the loss of accuracy caused by rounding errors in floating point arithmetic. They did not examine the effect of quantization noise in the images, nor did they consider cases where the matching between model and image edges was not properly done.

Huttenlocher and Ullman/Alter's Weak-perspective Pose Recovery

Indeed, Huttenlocher and Ullman [36] suggested what seems to be the most standard solution to this problem with the weak-perspective camera model. They noticed that

with this projection model, the resulting image is the same regardless of the order in which the two components (an Euclidean transformation and a scaling operation) are applied to the object. They also proved that this model yields very efficient algorithms, and a unique transformation (up to a reflection) corresponding to the pose even in the general case of arbitrary spatial positions with three point correspondences. So, they assume that the scaling, parameterized by an unknown s, is performed first. After that, the partially transformed model can be aligned with the image through the composition of a translation and a series of rotation about known axes. The z transformation between the object and the camera frames is immaterial, because the projection is parallel to the optical axis of the camera. On the other hand, the two-DOF translation parallel to the image plane that properly aligns one of the three model points can be computed directly from the coordinates of the point's image. The same thing is true for the first rotation, performed about the optical axis, that aligns one of three edges with its image. The two final rotation needed to complete the overall alignment can then be parameterized by unknown angles ϕ and θ. Fortunately, after the first rotation, one is left with exactly three geometrical constraints that can be used to derive a 3×3 system involving s, ϕ and θ. One of the two non-aligned points already lies on an aligned edge, and thus its complete alignment constrains one DOF only. The other point is totally unaligned, and so the condition that it must match its image constrains two additional DOF. After some algebraic manipulation, the system generated by these constraints can be solved for s, yielding a fourth-degree polynomial equation with terms of even order only. Obviously, scaling by a negative factor has no feasible geometrical interpretation. So, one is left with at most two solutions, which is already better than the bound of four possible solutions for P3P problems.

On the other hand, Alter [2] also derived the same biquadratic equation proposed by Huttenlocher and Ullman from a completely different analysis of the problem, but he managed to achieve a stronger result in a simpler and more elegant way. He interpreted the two possible solutions for the scaling factor s, in order to demonstrate that only one of them is algebraically and geometrically feasible. So, contrary to the P3P problem, there is actually no ambiguity in pose recovery from three correspondences, if a weak-perspective camera is assumed. The geometry of Alter's solution is illustrated in Fig. 3.6.

Alter initially noticed that the pose recovery problem as a whole can be reduced

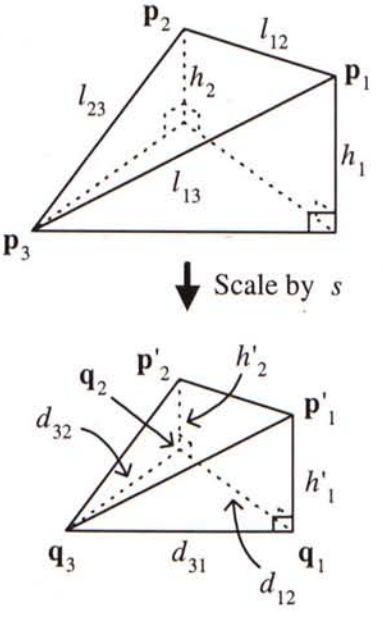

Figure 3.6: Geometry of Alter's solution to the problem of weak-perspective pose recovery with three point correspondences.

to determining the scale factor s, and the distances between any two of the three model points and the plane parallel to the image plane that contains the third point (h_1 and h_2). This can be done very easily if one observes that the scaled tetrahedron at the bottom of Fig. 3.6 contains two right-angle triangles ($\mathbf{q}_0\mathbf{q}_1\mathbf{p}'_1$ and $\mathbf{q}_0\mathbf{q}_2\mathbf{p}'_2$). A third right-angle triangle can be obtained by considering the difference in the heights h_1 and h_2, with respect to the image plane. These three triangles yield the following set of polynomial equations:

$$
\begin{aligned}
h_1'^2 + d_{13}^2 &= (s\ l_{13})^2 \\
h_2'^2 + d_{23}^2 &= (s\ l_{23})^2 \\
(h_1' - h_2')^2 + d_{12}^2 &= (s\ l_{12})^2
\end{aligned}
$$

After some algebraic manipulation one arrives at a biquadratic equation for s. So, the possible solutions for s^2 have the general form $\frac{b \pm \sqrt{\Delta}}{a}$. After some careful analysis, Alter showed that only the solution with the positive $\sqrt{\Delta}$ is geometrically feasible. The other solution corresponds to inverting the problem, or in other words, scaling the image and then projecting it orthogonally to the plane defined by the

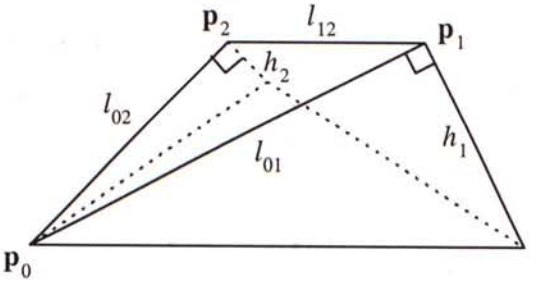

Figure 3.7: Geometrically infeasible solution to the problem of weak-persepective pose recovery with three point correspondences.

three model points, as illustrated in Fig. 3.7. After the correct values of s, h_1 and h_2 have been recovered, the computation of the transformation matrix can be performed by expressing the axes of the object frame as a parametric function of the two model edges and their dot product, as usual.

Fischler and Bolles' Closed-form P3P Formulation

The first solution to the problem of model-based pose recovery with full perspective camera model in modern computer vision literature is probably due to Fischler and Bolles [19]. In their classic work, given the three vectors $\mathbf{p}_1\mathbf{p}_2$, $\mathbf{p}_1\mathbf{p}_3$ and $\mathbf{p}_2\mathbf{p}_3$ (so that their lengths l_{12}, l_{13} and l_{23}, and in-between facing angles with the center of projection $\mathbf{O}$ can be found) at the base of a tetrahedron $\mathbf{O}\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$, cosines rule is applied for each pair of lines-of-sight to get a set of quadratic equations involving the unknowns $|\mathbf{O}\mathbf{p}_i|$, $i = 1, ..., 3$:

$$
\begin{aligned}
|\mathbf{O}\mathbf{p}_1|^2 + |\mathbf{O}\mathbf{p}_2|^2 - 2\,|\mathbf{O}\mathbf{p}_1|\,|\mathbf{O}\mathbf{p}_2| \cos(\angle \mathbf{p}_1\mathbf{O}\mathbf{p}_2) &= l_{12}^2 \\
|\mathbf{O}\mathbf{p}_1|^2 + |\mathbf{O}\mathbf{p}_3|^2 - 2\,|\mathbf{O}\mathbf{p}_1|\,|\mathbf{O}\mathbf{p}_3| \cos(\angle \mathbf{p}_1\mathbf{O}\mathbf{p}_3) &= l_{13}^2 \\
|\mathbf{O}\mathbf{p}_2|^2 + |\mathbf{O}\mathbf{p}_3|^2 - 2\,|\mathbf{O}\mathbf{p}_2|\,|\mathbf{O}\mathbf{p}_3| \cos(\angle \mathbf{p}_2\mathbf{O}\mathbf{p}_3) &= l_{23}^2
\end{aligned}
$$

Knowing the fact that n polynomial equations in n unknowns can have no more solutions than the product of their respective degrees, the system above can have at most eight solutions. Furthermore, as all non-constant terms of the system are of second-degree, for every real positive solution, there is a geometrically isomorphic negative solution. Obviously, the negative solutions are of no relevance for the problem at hand and can be discarded. So, the upper bound on the number of possible

solutions for any instance of the P3P problem with the points in general position and distinct projections in the image plane is four.

Indeed, after some algebraic manipulation, the lengths of the three remaining sides of the tetrahedron $|\mathbf{Op}_1|$, $|\mathbf{Op}_2|$ and $|\mathbf{Op}_3|$ can be found by

$$
\begin{aligned}
|\mathbf{Op}_1| &= l_{23} \left(\omega_1^2 - 2\omega_1 \rho_{12} + 1\right)^{-1/2} \\
|\mathbf{Op}_2| &= \omega_1 |\mathbf{Op}_1| \\
|\mathbf{Op}_3| &= \omega_2 |\mathbf{Op}_1|
\end{aligned}
$$

where ω_1 and ω_2 are variables related to the solution to the following quartic equation:

$$
G_4 \omega_1^4 + G_3 \omega_1^3 + G_2 \omega_1^2 + G_1 \omega_1 + G_0 = 0
$$

and ρ_{12} is the cosine of tetrahedron face angle $\mathbf{p}_1\mathbf{Op}_2$ for some constants G_0 to G_4. With this formulation of $|\mathbf{Op}_i|$, z_i are given by:

$$
z_i = \left[|\mathbf{Op}_i|^2 f^2 / \left(u_i^2 + v_i^2 + f^2\right) \right]^{1/2}
$$

(cf. Appendix for detailed formulation of the Fischler and Bolles' method)

3.2 Our Iterative P3P Algorithm

On the other hand, we [21] proposed another iterative algorithm for the P3P problem under full perspective projection. We showed that, given the 3D dimensions of a rigid object and its subsequent corresponding image points, its depth information z_i can be found with a least-squares minimization process. Instead of iterating over the full range of z_i, depth estimates in our method are continuously refined by the Gauss-Newton method until the global error measurement is less than some thresholds.

Our algorithm is broken down into two separate stages: (1) depth estimation by the Gauss-Newton method, in which depth information is iteratively refined under a least-square minimization paradigm; and (2) pose calculation with a 3D-to-3D pose-recovery algorithm [32], in which 3D point sets $\mathbf{p}_i$ at time t and $\mathbf{p}'_i$ at time $t+1$ are used to find $\mathbf{R}$ and $\mathbf{t}$ such that

$$\mathbf{p}'_i = \mathbf{R}\mathbf{p}_i + \mathbf{t} \tag{3.17}$$

Additional geometrical constraints are also incorporated to improve the stability and reliability of the estimation process when the observations are noisy [58]. Under the full perspective projection model,

$$\begin{aligned} x_i &= u_i \frac{z_i}{f} \\ y_i &= v_i \frac{z_i}{f} \end{aligned} \tag{3.18}$$

and defining

$$\begin{aligned} \lambda_1 &= u_1^2 + v_1^2 + f^2 \\ \lambda_2 &= u_2^2 + v_2^2 + f^2 \\ \lambda_3 &= u_3^2 + v_3^2 + f^2 \\ \lambda_4 &= u_1 u_2 + v_1 v_2 + f^2 \\ \lambda_5 &= u_1 u_3 + v_1 v_3 + f^2 \\ \lambda_6 &= u_2 u_3 + v_2 v_3 + f^2 \end{aligned} \tag{3.19}$$

z_1 and z_3 can be expressed in terms of z_2 as (cf. Appendix for derivation)

$$z_1 = \frac{1}{\lambda_1}\left[\pm\sqrt{\Delta_1} + \lambda_4 z_2\right] \tag{3.20a}$$

$$z_3 = \frac{1}{\lambda_3}\left[\pm\sqrt{\Delta_3} + \lambda_6 z_2\right] \tag{3.20b}$$

where

$$\Delta_1 = l_{12}^2 f^2 \lambda_1 - \left(\lambda_1\lambda_2 - \lambda_4^2\right) z_2^2 \geq 0 \tag{3.21a}$$

$$\Delta_3 = l_{23}^2 f^2 \lambda_3 - \left(\lambda_2\lambda_3 - \lambda_6^2\right) z_2^2 \geq 0 \tag{3.21b}$$

so that our goal is to find a z_2 which minimizes the following error functions e_1 and e_2 to below a predefined error threshold ξ (cf. Appendix for derivation).

$$e_1 = \left(\lambda_1 z_1^2 - 2\lambda_5 z_1 z_3 + \lambda_3 z_3^2 - l_{13}^2 f^2\right)^2 \tag{3.22a}$$

$$e_2 = \max\left(\left(\frac{\sum u_i z_i}{\beta_3} - \frac{\beta_1}{3}\right)^2, \left(\frac{\sum v_i z_i}{\beta_3} - \frac{\beta_2}{3}\right)^2\right) \tag{3.22b}$$

where

$$\beta_1 = u_1 + u_2 + u_3 \tag{3.23a}$$

$$\beta_2 = v_1 + v_2 + v_3 \tag{3.23b}$$

$$\beta_3 = z_1 + z_2 + z_3 \tag{3.23c}$$

Because of the smoothness and well-behaved properties of the full perspective projection model, the Gauss-Newton method may be used for the minimization of the above error functions.

3.2.1 Gauss-Newton Method

Suppose the true value of z_2 falls within the interval $[\delta_1, \delta_2]$ where δ_1 and δ_2 are positive real numbers evaluated from certain constraints to be explained in the next paragraph, we have the following pseudo-code for finding $\mathbf{p}_i$ having $e_1 < \xi$ or $e_2 < \xi$:

```
generate a random initial guess z2 among [δ1, δ2]
do
    find the values of z1 and z3 from Eq. (3.20a) and Eq. (3.20b),
    pick a z1 and a z3 leading to minimal e1 and e2
    if e1 < ξ or e2 < ξ
        solution found,
    else
        calculate ∂e1/∂z2 and ∂e2/∂z2
        update estimate of z2 by ∂e1/∂z2 and ∂e2/∂z2
while loop-terminating conditions are not satisfied.
```

The algorithm iterates until $e_1 < \xi$ or $e_2 < \xi$, or the number of iterations exceeds ten.

Although this method requires initial guesses and there is a risk of converging to local minima, ample experimental results suggest that the Gauss-Newton method will

usually converge in a stable manner from a wide range of starting positions, as long as there are more constraints than unknowns, and by incorporating solution stabilizing procedures.

3.2.2 Dealing with Ambiguity

In fact, the above problem and their solutions are more for academic interest than for practical use. The primary reason is that in real world application it is almost unavoidable that noise is introduced in various stages, e.g. inaccurate measurements as well as quantization noise in imaging. In problems involving too little point samples such as the P3P problem above, it is often the case that noisy quantities can lead to ill-conditioned solutions even when the equations are over-constrainted. Hence one prefers to work with more point correspondences together with model fitting methods like RANSAC to drop out data containing a significant percentage of gross errors. In this case, $\mathbf{R}$ and $\mathbf{t}$ can be obtained as a solution to the following least-squares problem:

$$\text{minimize w.r.t. } \mathbf{R}, \mathbf{t} \quad \left\{ \sum |\mathbf{p}'_i - (\mathbf{R}\mathbf{p}_i + \mathbf{t})|^2 \right\}$$

Such a constrainted problem can be solved by linear procedures like quaternions, SVD or other iterative methods [34].

Another infelicity is that model points in general positions may admit more than one solution and lead to ambiguity. Some alternatives to deal with it have been proposed, but the traditional consistency checks based on the reprojection of the model features, for instance, are in many cases not restrictive enough to yield a unique solution. On the other hand, the techniques based on Hough transforms (see below) are in general so computation intensive that their application under real-time constraints is at least questionable.

Furthermore, the analytical methods discussed so far have very poor error propagation properties. For instance, in the comparison between P3P algorithms performed by Haralick *et al.* [26], all the techniques tested were found to produce relative errors of at least 0.1% in feature positions, just as a result of the propagation of rounding errors with single precision arithmetic. Of course, this problem becomes much more serious if we take into account the effect of quantization noise in the imaging process, for instance. Furthermore, these methods are based only on a very small number of correspondences and thus they can produce completely wrong results if

some incorrectly matched features are eventually used.

Constraints in Orientation Representation Schemes

Rotation in 3D space has been represented by 3×3 matrices with nine parameters. However, as pure rotation has only three DOF, redundant information exists within these nine parameters. Alternatively, the relative orientation between the camera and the target may be denoted by three angles measured about fixed coordinate axes with respect to the camera. This non-redundant notation is known as *roll*, *pitch* and *yaw* (RPY) angles. It is very desirable if one is using a numerical technique that searches the space of possible poses for an acceptable solution. The fewer number of parameters used in encoding, the smaller tends to be the computational cost of exploring it.

Unfortunately, this representation has some serious shortcomings. To start with, it contains a singularity: when the pitch angle is equal to $\frac{\pi}{2}$ radians, roll and yaw rotation combined span only a 1D space and cannot be separated from each other [14]. RPY angles are also ambiguous: there is more than one way to represent certain orientations. Finally, they complicate the composition of two rotation, and the physical meaning of the orientations represented with them is not intuitive. The same observations are valid for Euler angles, another non-redundant scheme that represents rotation as the sequential composition of three rotation about the axes of a frame that moves in space with each rotation.

Another possibility for representing orientations of 3D objects with just three parameters is through a "3-vector" ω, which orientation specifies the rotation axis (with respect to a reference frame), and which norm specifies the angle about this axis. A problem with this representation is that, if the angle is a multiple of 2π radians, then the direction of ω is undefined. The only way to avoid this is to restrict the angle to the interval $[-\pi, -\pi]$, but in this case a discontinuity is introduced. Furthermore, in the general case ω is not a physical vector, and the analytical computation of derivatives with respect to its components is quite complex.

Gennery [24] noticed these shortcomings and suggested that an ideal system for representing orientations should have the following properties:

1. **Non-redundancy**: the number of free parameters should be exactly equal to the number of DOF in the space of possible orientations;

2. **Continuity**: a continuous motion of the object which orientation is represented should always result in continuous changes in all parameters;

3. **Absence of singularities**: the partial derivatives of all parameters with respect to any differential rotation, at any orientation, should be always finite.

Multiple Solutions in P3P

A problem that is common to all the perspective pose recovery techniques discussed so far is that the nonlinear nature of the constraints employed usually results in numerical instability and multiple solutions [34]. Indeed, Fischler and Bolles [19] showed that P3P problems can have as many as four real solutions in front of the camera and another four physically-infeasible ones behind. They also described specific geometric configurations in which four triangles of the same shape project to a single image. One such instance happens when the three image points are the vertices of an equilateral triangle centered at the optical axis, and the 3D model is also an equilateral triangle. In this case, there are four possible poses for the object. It may be on a plane orthogonal to the optical axis, so that the distances between the optical center and the three vertices are all equal, or any one of the vertices may be at a distance four times smaller than the other two (with respect to the optical center). Wolfe *et al.* [86] later showed that while there are cases that yield four solutions, there would only be two for most of the time (Fig. 3.8). Of course, only one such solution corresponds to the actual pose of object in the scene, choosing the right solution when more than one is found is still a problem faced by the computer vision community nowadays.

Solution Verification

The most straightforward type of verification in identifying a unique solution consists of reprojecting object features back onto the image plane to build a synthetic image of the scene using each candidate pose. Then, the discrepancy between reprojected image and the actual one can be used to determine whether the candidate pose is a desired solution or not. For instance, Dhome *et al.* [18] proposed this type of verification in their solution to the Perspective-3-Line problem. The constraints that they use to recover the relative orientation between an object and the camera frame theoretically guarantee that the reprojected edge images will be parallel to the corresponding actual edge images, but they do not guarantee that these images will

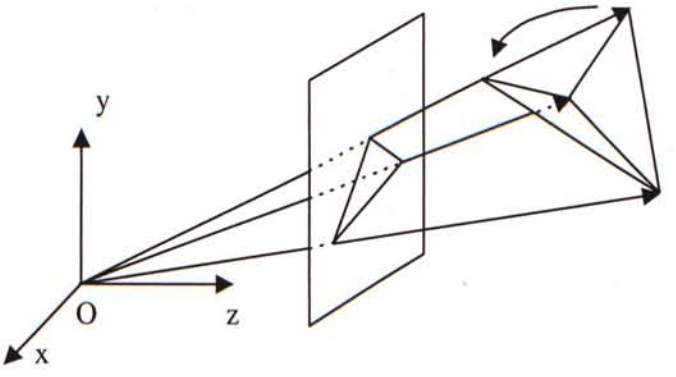

Figure 3.8: An example of multiple solutions for P3P problem.

be coincident. So, the perpendicular distance between matched pairs of reprojected and actual edge images can be used to determine the feasibility of any recovered pose estimate.

This type of verification can be strengthened with the use of visibility constraints to eliminate obviously infeasible candidates. A trivial example is the elimination of poses that result in negative z-component or $z_i < f$ (because in such cases, the object goes behind the camera). Furthermore, poses that would result in the occlusion of features that were actually detected in the real image can be discarded. For example, if the position of a model point in the camera frame is given by a vector $\mathbf{p}$ and the outward normal to the object surface at that point is $\mathbf{n}$ (still in the camera coordinates), then, assuming that the object is convex and it is not occluded by anything else in the scene, the point is visible if and only if:

$$\mathbf{p} \cdot \mathbf{n} < 0$$

In some cases, however, simple consistency checks may not be enough to eliminates all the ambiguity created by nonlinear constraints. Fortunately, more elaborate schemes for disambiguation can be found in the literature. For instance, if one is using a method that requires N correspondences but the actual number of features matched in the image is $N > M$, then a Hough transform can be employed to determine the most likely solution. The idea is to group all the available matches between model and edge features into (possibly non-disjoint) subsets that contain just the minimal number of constraints required by the pose recovery scheme selected (three points in

the case of a P3P technique, for example). Then the analytical pose recovery algorithm of choice can be applied independently for each individual subset, yielding a finite number of possible pose estimates. Each such pose estimate can be thought of as a point in a multi-dimensional space composed by the pose parameters. So, the pose estimates generated by different subsets can be grouped into clusters (for instance through a quantization of the pose space), and the cluster with the biggest number of elements can be used to determine the most likely solution. Linnainmaa *et al.* [44] made use of this approach to disambiguate their solution to the generic P3P problem. But they noticed that the quantization of a 6D space, such as the one formed by all the free parameters in a generic pose recovery problem, can be a quite expensive step both in terms of time and memory space. So, they initially worked only with the 3D space composed by the translational parameters. Then the best intermediate solution was used in another Hough transform to get the desired rotation parameters. Actually, Linnainmaa's disambiguation scheme worked very well for a few test cases, but a more systematic evaluation is necessary.

On the other hand, several additional geometrical constraints on the possible values of z_2 under different circumstances could be derived for our algorithm. They impose an upper bound and a lower bound on every depth estimate to greatly reduce unnecessary root-probing. For example, rearranging Eqs. (3.21a) and (3.21b) gives

$$min\,(\zeta_1, \zeta_2) \geq z_2 > f$$

where

$$\zeta_1 = l_{12} f \sqrt{\frac{\lambda_1}{\lambda_1 \lambda_2 - \lambda_4^2}} \tag{3.24a}$$

$$\zeta_2 = l_{23} f \sqrt{\frac{\lambda_3}{\lambda_2 \lambda_3 - \lambda_6^2}} \tag{3.24b}$$

Furthermore, Eq. (3.22a) indicates the SSD of the calculated values of l_{12} from their true values, while Eq. (3.22b) measures the difference among centroid coordinates $\begin{bmatrix} u_{\text{centroid}} & v_{\text{centroid}} \end{bmatrix}^T$ calculated by the given 2D image coordinates $\begin{bmatrix} u_i & v_i \end{bmatrix}^T$ and the calculated 3D object coordinates $\begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^T$ after the full-perspective projection. Both of them also play a crucial role in eliminating multiple solutions inherent in P3P problems.

3.2.3 3D-to-3D Motion Estimation

Various non-iterative methods, e.g. SVD [34] and quaternions [5], have been suggested for solving the least-square fitting problem of two 3D point sets $\mathbf{p}_i$ and $\mathbf{p}'_i$, $i = 1, 2, ...N$, which tries to find $\mathbf{R}$, $\mathbf{t}$ and a scalar value c such that the value of

$$\sum_{i=1}^{N} |\mathbf{p}'_i - (c\mathbf{R}\mathbf{p}_i + \mathbf{t})|^2$$

is minimized. However, they are often computational expensive for the large amount of matrix operations. Alternatively, Horn [32] discussed a direct method for recovering $\mathbf{R}$ and $\mathbf{t}$ in the three-point case. A very favorable property of this method is that it involves only simple vector multiplications, and the resulting rotation matrix is orthonormal by itself so that there is no need for further orthonormality enforcement or approximation. Later, Umeyama [82] further refined the solution proposed by Arun *et al.* [5] to correct the reflection phenomenon. Basically, all these algorithms break down the estimation process into two stages: (i) estimate $\mathbf{R}$ first, and (ii) recover $\mathbf{t}$ by calculating the 3D positional difference between the centroid of transformed point set at time $t+1$ (i.e. centroid of $\mathbf{p}'_i$) and that of the rotated point set at time t (i.e. centroid of $\mathbf{R}\mathbf{p}_i$). This results from the fact that each correspondence available yields a linear relation that constraints one DOF in $\mathbf{t}$. Using three correspondences, one gets a 3×3 system that can be readily solved for all the three translational components. These methods are briefly described below, readers are referred to [82] and [32] for details.

Arun/Umeyama's Method

In this method, motion estimation is reduced to the decomposition of a 3×3 covariance matrix $\mathbf{H}$, which embeds all the information from point coordinates. The computation is efficient when many points are available, as can be achieved with the following steps:

1. Compute the following

$$\mu_P = \frac{1}{N}\sum_{i=1}^{N} \mathbf{p}_i$$

$$\mu_{P'} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{p}'_i$$

$$\sigma_P^2 = \frac{1}{N}\sum_{i=1}^{N} |\mathbf{p}_i - \mu_P|^2$$

$$\sigma_{P'}^2 = \frac{1}{N}\sum_{i=1}^{N} |\mathbf{p}'_i - \mu_{P'}|^2$$

$$\mathbf{H} = \frac{1}{N}\sum_{i=1}^{N} (\mathbf{p}'_i - \mu_{P'})(\mathbf{p}_i - \mu_P)^T$$

1. Find the SVD of $\mathbf{H}$:

$$\mathbf{H} = \mathbf{U}\mathbf{W}\mathbf{V}^T$$

where $\mathbf{U}$ and $\mathbf{V}$ are 3×3 orthonormal matrices, and $\mathbf{W}$ is a 3×3 diagonal matrix which diagonal elements contain the eigenvalues of $\mathbf{H}$.

1. If the rank of $\mathbf{H} > \mathbf{2}$,

$$\mathbf{S} = \begin{array}{ll} \mathbf{I} & \text{if } \det(\mathbf{H}) \geq 0 \\ \text{diag}(1,1,-1) & \text{if } \det(\mathbf{H}) < 0 \end{array}$$

else if the rank of $\mathbf{H} = 2$,

$$\mathbf{S} = \begin{array}{ll} \mathbf{I} & \text{if } \det(\mathbf{U})\det(\mathbf{V}) = 1 \\ \text{diag}(1,1,-1) & \text{if } \det(\mathbf{U})\det(\mathbf{V}) = -1 \end{array}$$

1. Then the optimal transformation is given by

$$\begin{aligned} \mathbf{R} &= \mathbf{U}\mathbf{S}\mathbf{V}^T \\ c &= \frac{1}{\sigma_P^2}\text{tr}(\mathbf{W}\mathbf{S}) \\ \mathbf{t} &= \mu_{P'} - c\mathbf{R}\mu_P \end{aligned} \tag{3.25}$$

where $\det(\mathbf{X})$ is the determinant of a matrix $\mathbf{X}$ and $\text{tr}(\mathbf{X})$ is the trace $\mathbf{X}$.

Horn's Method

Constructing column unit vectors $\hat{\mathbf{x}}_1, \hat{\mathbf{y}}_1$ and $\hat{\mathbf{z}}_1$ from $\mathbf{p}_1$,$\mathbf{p}_2$ and $\mathbf{p}_3$,

$$\hat{\mathbf{x}}_1 = \frac{\mathbf{p}_2 - \mathbf{p}_1}{|\mathbf{p}_2 - \mathbf{p}_1|}$$
$$\hat{\mathbf{y}}_1 = \frac{(\mathbf{p}_3 - \mathbf{p}_1) - [(\mathbf{p}_3 - \mathbf{p}_1) \bullet \hat{\mathbf{x}}_1]\hat{\mathbf{x}}_1}{|(\mathbf{p}_3 - \mathbf{p}_1) - [(\mathbf{p}_3 - \mathbf{p}_1) \bullet \hat{\mathbf{x}}_1]\hat{\mathbf{x}}_1|}$$
$$\hat{\mathbf{z}}_1 = \hat{\mathbf{x}}_1 \times \hat{\mathbf{y}}_1$$

Similarly unit vectors $\hat{\mathbf{x}}_2, \hat{\mathbf{y}}_2$ and $\hat{\mathbf{z}}_2$ are constructed from $\mathbf{p}'_1$, $\mathbf{p}'_2$ and $\mathbf{p}'_3$.

The rotation matrix is given by

$$\mathbf{R} = \mathbf{M}_2 \mathbf{M}_1^T$$

where

$$\mathbf{M}_1 = \begin{bmatrix} \hat{\mathbf{x}}_1 & \hat{\mathbf{y}}_1 & \hat{\mathbf{z}}_1 \end{bmatrix}$$
$$\mathbf{M}_2 = \begin{bmatrix} \hat{\mathbf{x}}_2 & \hat{\mathbf{y}}_2 & \hat{\mathbf{z}}_2 \end{bmatrix}$$

Once rotation has been found, $\mathbf{t}$ is given by Eq. (3.25) as before (where $N = 3$ and $c = 1$ for this case).

3.3 Experimental Results

We have tested our algorithm on both synthetic data and real images.

3.3.1 Synthetic Data

In the first part, in order to evaluate the effectiveness of our proposed constraints on the reduction of multiple solutions in P3P problems, and to verify the robustness of our algorithm, a thousand sets of 3D points $\mathbf{p}_i$, $i = 1, ..., 3$ (each of them contains three points) were generated by a uniform random number generator. These points lay within a rectangular block of depth 50 centered at $\begin{bmatrix} 0 & 0 & 100f \end{bmatrix}^T$, so that their projections fall within a window of size 128×128 pixels on the image plane. Then the motion is generated randomly such that each Euler angle may take any value between

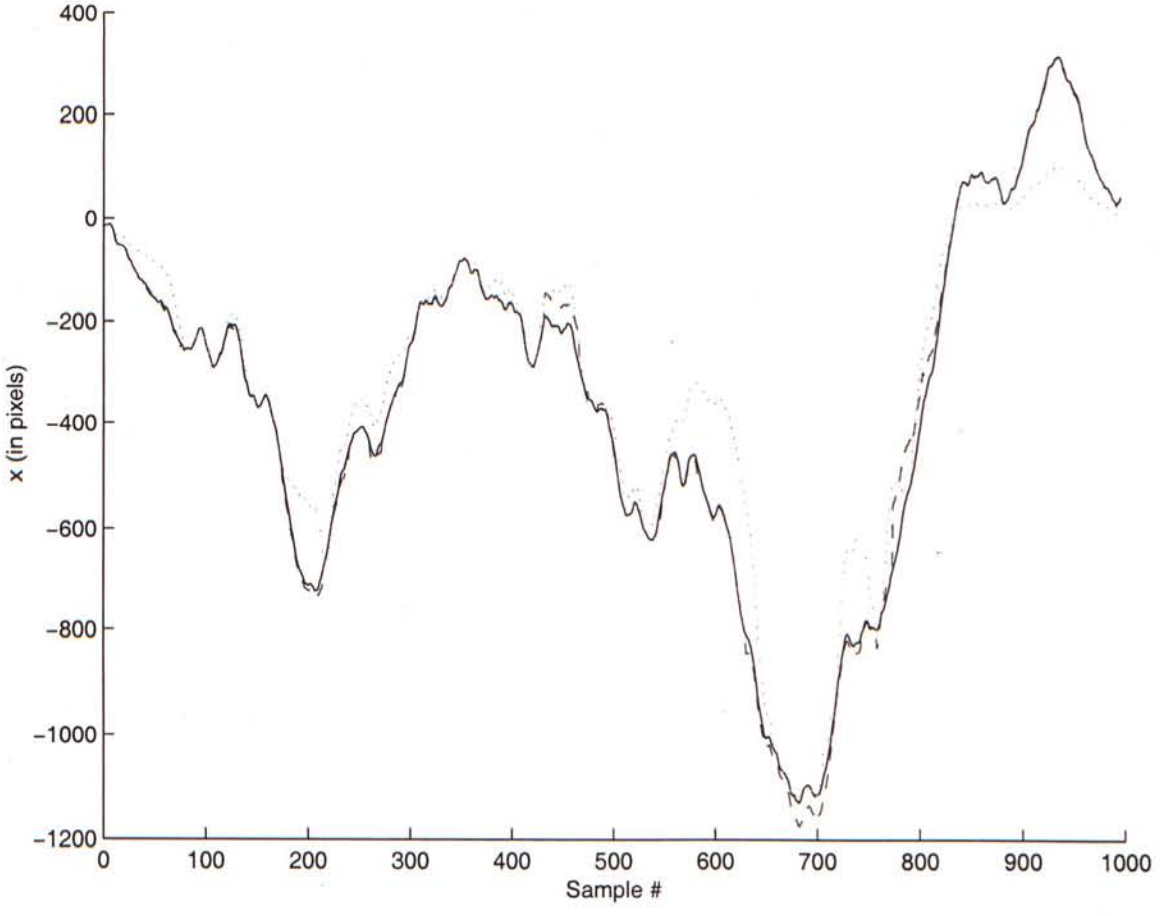

Figure 3.9: x-coordinates of the P3P solutions (solid line: true solutions, dashed line: our iterative algo, dotted line: Fischler and Bolles algo).

$\pm 45°$, while translation vectors may produce a -50 to $+50$ pixels lateral shift on the image plane. Then $\mathbf{p}_i'$ are computed from Eq. (3.17). Image noise was simulated by adding Gaussian random values to the exact image coordinates with a zero mean and a variance of 1, 2 and 3 pixels. We assumed that there is at least one real solution for every motion instant.

Figs. 3.9 to 3.11 show the results of the Fischler and Bolles's method (in dotted lines) and those of our iterative method (in dashed lines), together with the true values in solid lines. In our experiments with the Fischler and Bolles' algorithm, only the single solution (out of a total of four possible roots) leading to minimal motion is used, as the motion within a sampling period (e.g. one-thirtith of a second) is assumed to be very small. However, it is still clear that there are many incorrect solutions made by the Fischler and Bolles' method. To confirm their validity, we

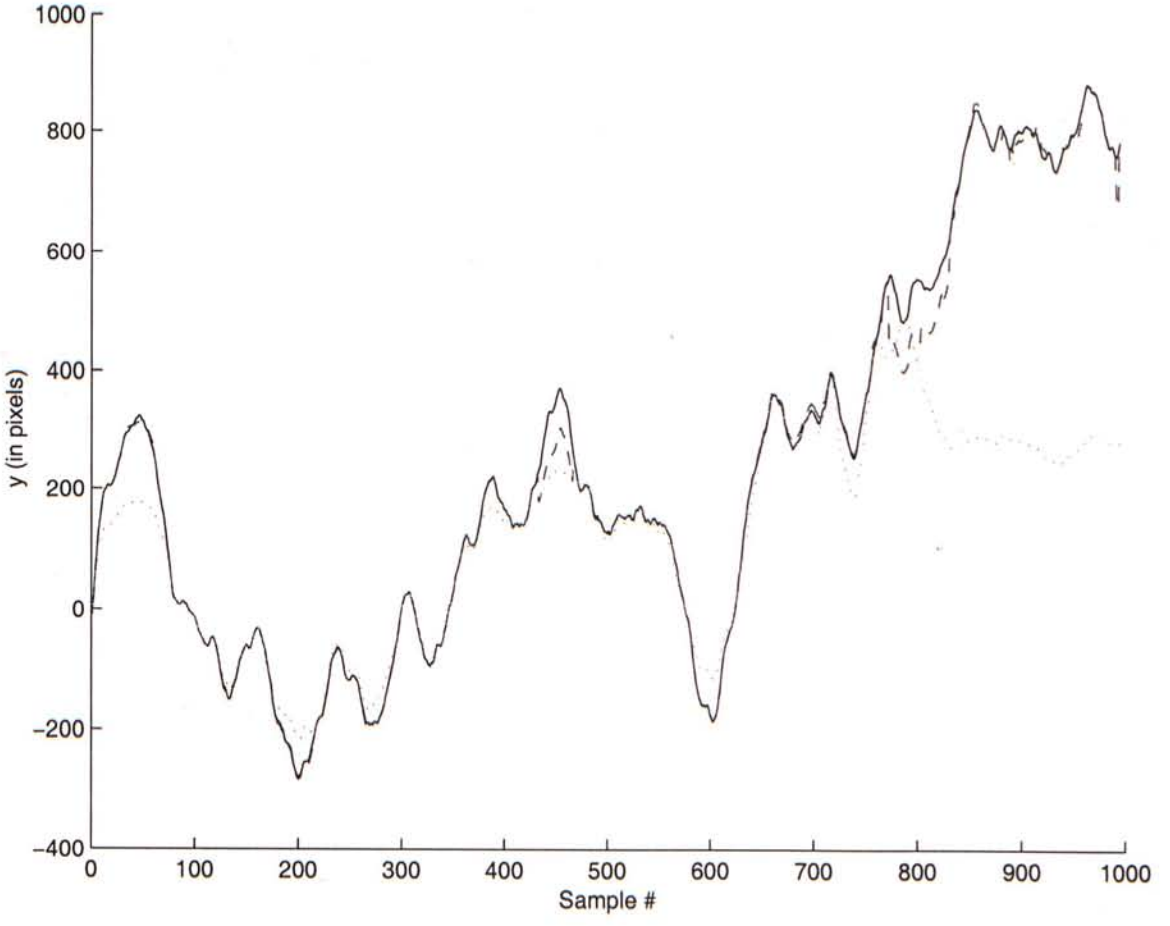

Figure 3.10: y-coordinates of the P3P solutions (solid line: true solutions, dashed line: our iterative algo, dotted line: Fischler and Bolles algo).

evaluated the Euclidean distances among them and back-projected them onto the image plane, finding that they matched closely with the true image data. We can thus conclude that this discrepancy was due to the problem of multiple solutions, and we found that 483 of our samples exhibited this phenomenon. These graphs also show that our proposed solution constraints [21] are robust to noise, and can lead to more accurate and stable results.

In addition, Table 3.4 shows the number of floating point operations (flops) required for finding the depths of feature points with different P3P algorithms on Matlab 5.1 platform. It clearly suggests that applying our solution constraints only induces a marginal increase in the computational cost of the P3P formulations. Knowing that the closed-form Fischler and Bolles' approach is among the most efficient algorithm for P3P problems, we could claim that our improved iterative algorithm is efficient

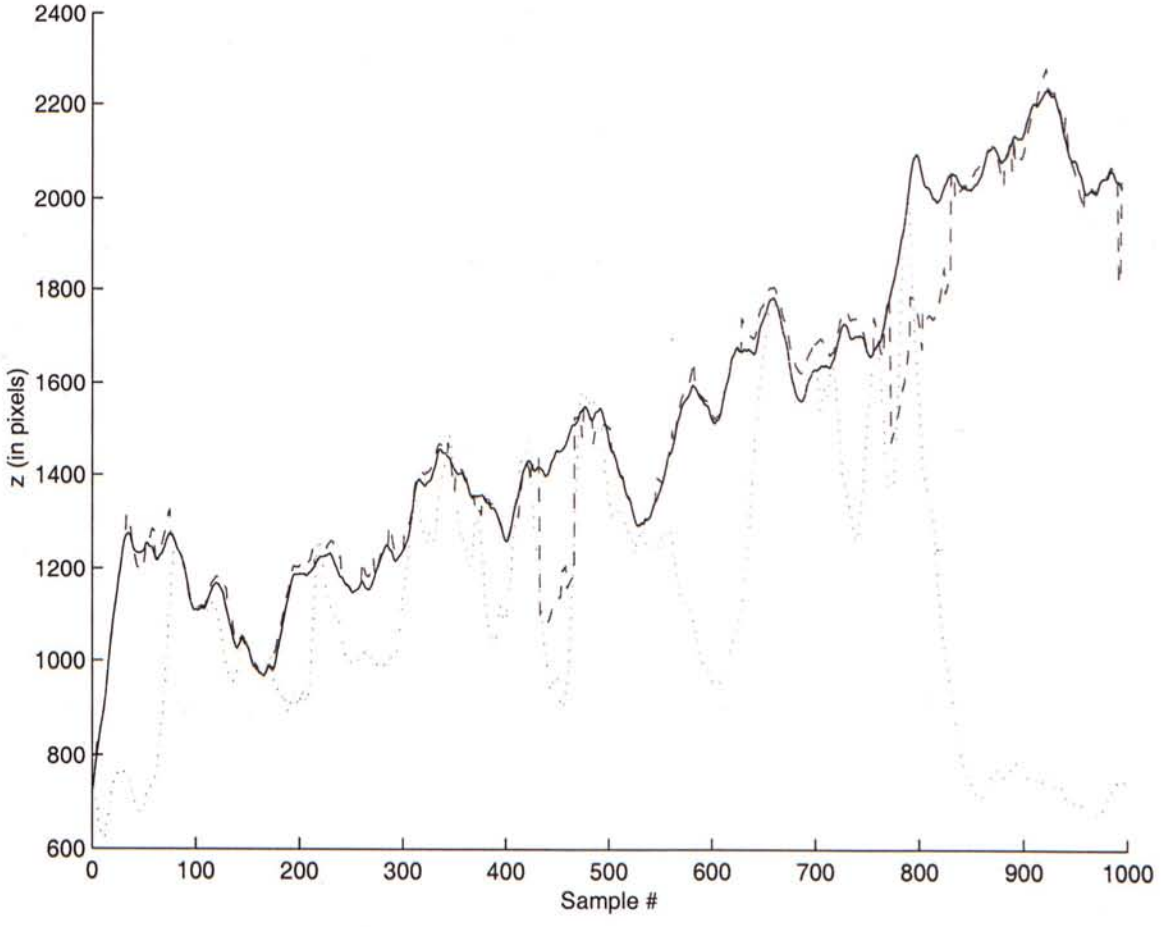

Figure 3.11: z-coordinates of the P3P solutions (solid line: true solutions, dashed line: our iterative algo, dotted line: Fischler and Bolles algo).

(only about six iterations are required for convergency for most of the cases) in view of its iterative nature. On the other hand, we have also implemented our iterative algorithm in ANSI C using double-precision floating-point operations and 10^{-7} as the error threshold. The computation time on a UltraSparc 1/170 workstation running Solaris 2.5.1 with 512MB RAM is 0.11 sec.

To further test the numerical stability of our method, we permute the order of the three object points $\mathbf{p}_i\mathbf{p}_j\mathbf{p}_k$, where $i \neq j \neq k$, to give a total of six combinations. This effectively means a systematic permutation of variable substitution. Charts similar to Fig. 3.11 are obtained for the other five permutations, showing that our algorithm is robust to the order of substitution and numerical calculations.

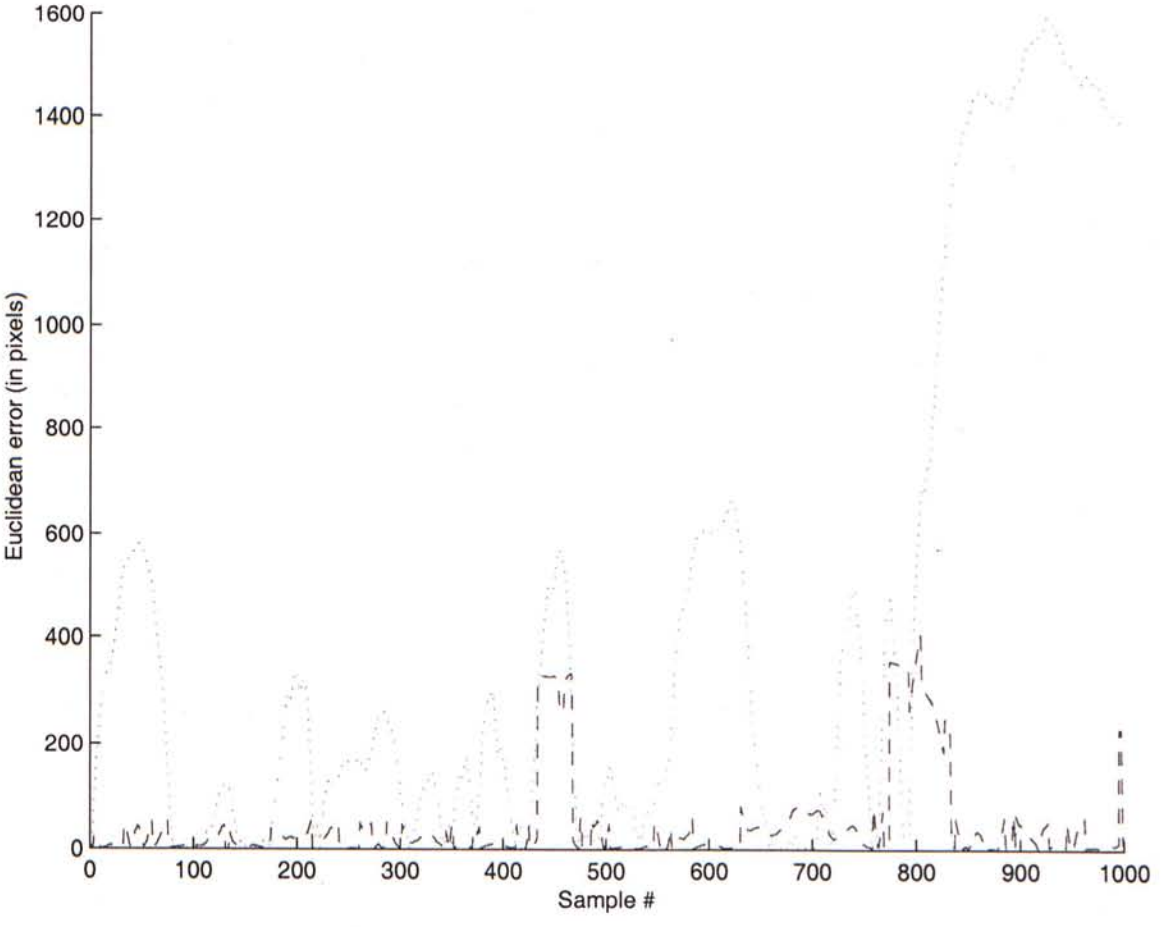

Figure 3.12: Euclidean error (in pixel) of the P3P solutions (dashed line: our iterative algo, dotted line: Fischler and Bolles algo).

3.3.2 Real Images

The algorithm was also tested on real images. A 15-frame sequence of a black cardboard on a table was taken by a digital camera at a resolution of 640 $\times$ 512 pixels and focal length of 28 mm. The object center was approximately 1000 mm from the camera projection center. The total motion between the sequence of image is a rotation of 33° about the y-axis. Fig. 3.15 shows two of the frames marked with three manually selected feature points. Note that the white strips are added solely for camera auto-focusing.

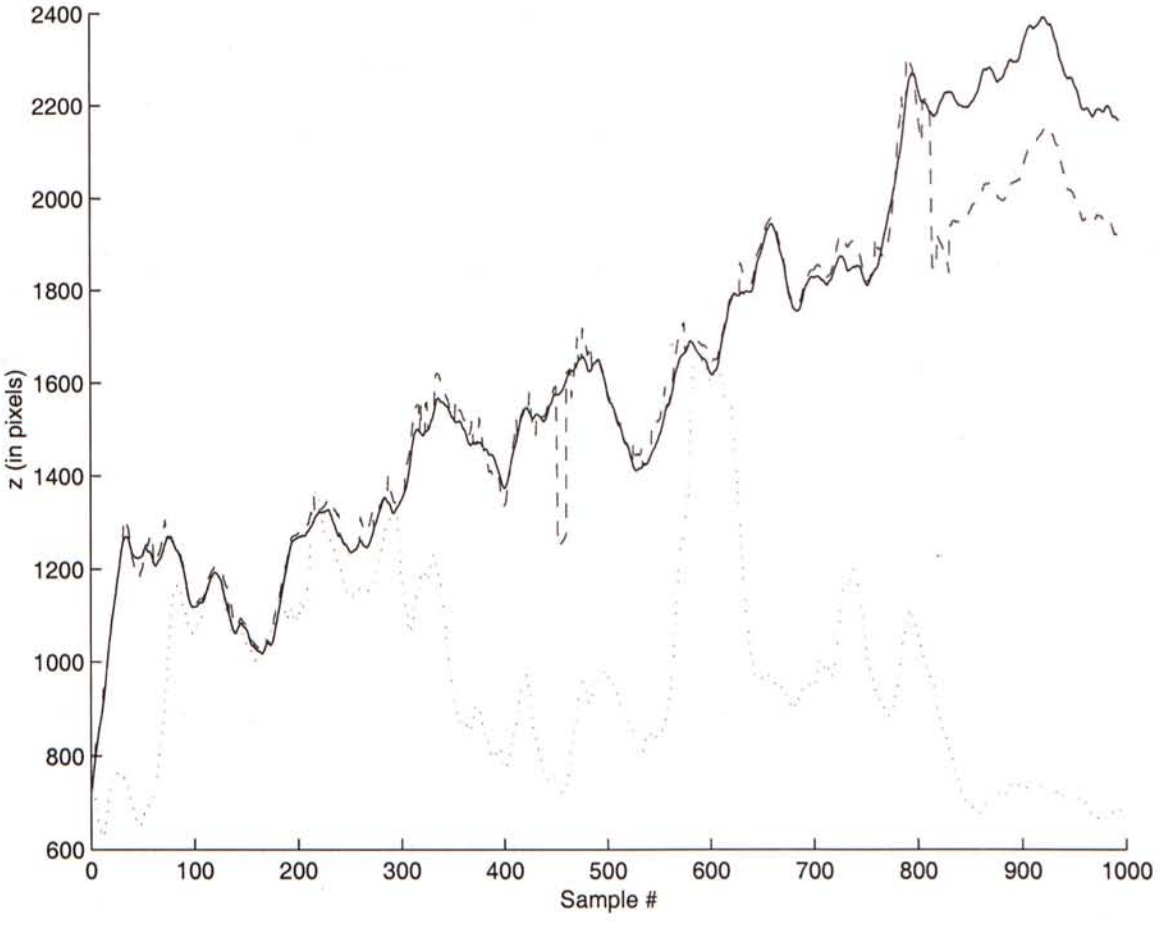

Figure 3.13: z-coordinates of the P3P solutions with no rotation (solid line: true solutions, dashed line: our iterative algo, dotted line: Fischler and Bolles algo).

3.4 Discussions

An iterative algorithm based on the Gauss-Newton method for the Perspective-3-Point (P3P) problem is derived. We are currently investigating the following possible enhancements:

1. Our current implementation relies on noisy point correspondences and 3D feature dimensions. While these measurements may not be readily accessible under all circumstances, excessive noise may lead to wrong calculations and unrealistic view synthesis. Our next target is to relax the dependency on point correspondences and feature dimensions, or devise an update rule so that this information can be updated adaptively on the fly.

2. As numerical errors accumulate with increasing amounts of calculations and

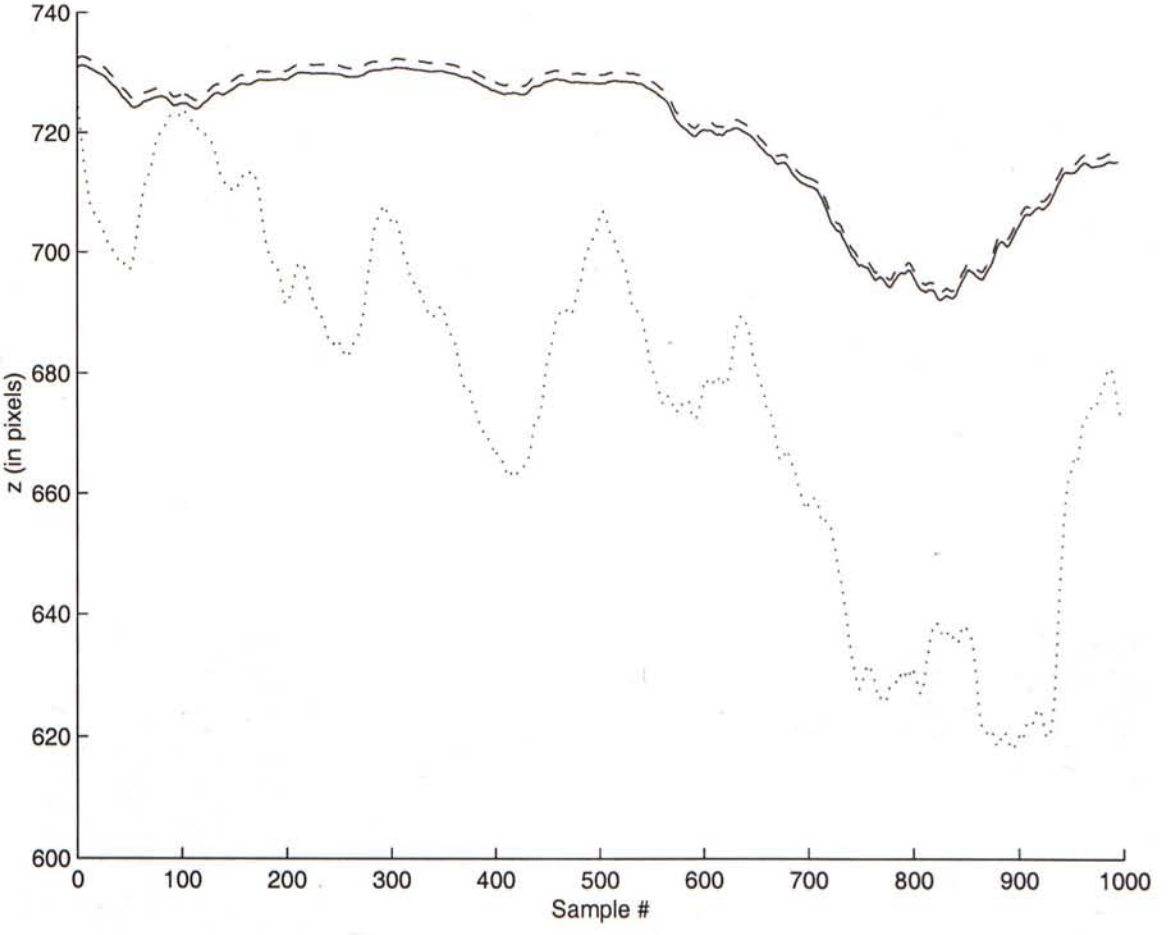

Figure 3.14: z-coordinates of the P3P solutions with no translation (solid line: true solutions, dashed line: our iterative algo, dotted line: Fischler and Bolles algo).

significantly magnify in some mathematical operations, the error would be very serious when a large number of images have to be processed. Intermediate error rectification mechanism has to be incorporated to limit the error to below an acceptable limit.

	Fischler and Bolles's Method	Our Iterative Method
Before	1733	2068
After	1908	2245

Table 3.4: Number of floating point operations required for different P3P algorithms (before and after solution constraints are applied).

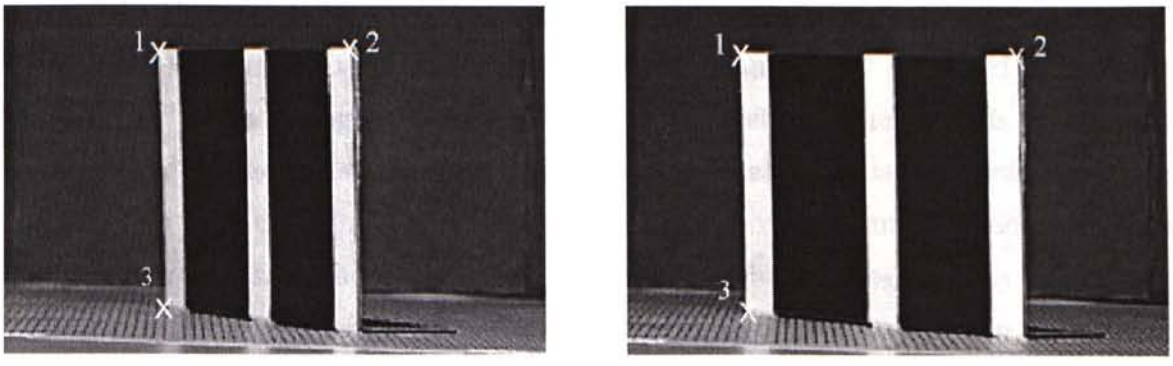

Figure 3.15: Two consecutive frames out of a 15-frame real testing image sequence.

	Fischler and Bolles	Our iterative
Translation ($\mathbf{t}$) only	36.0823	1.5056
Rotation ($\mathbf{R}$) only	6.0131	0.7457
Both $\mathbf{R}$ and $\mathbf{t}$	22.9718	2.6913

Table 3.5: Mean percentage error between true and calculated feature points for different P3P algorithms.

Chapter 4

Panoramic View Analysis

A complete mosaic representation of visual scenes often includes a full-view panoramic image, depth or parallax information, and the incremental alignment parameters or residual differences that represent any scene changes not captured in mosaics [41][69][76]. Even though the idea is easy to understand, the process of building a panorama (image registration, stitching, and residual analysis) involves much theories and is still under active research in computer vision, image processing, and computer graphics. In addition, while mosaics have been recognized as an efficient way of providing snapshot views of scenes, the issue of how to develop a complete representation of scenes based on mosaics has not been adequately studied in the literature.

This chapter explores various advanced issues in mosaic representation (section 4.1), techniques for automatic mosaic construction (section 4.2), and describes a series of increasingly complex image alignment transformations (section 4.3). Different application of such representations would also be illustrated with real examples (section 4.4). Some of our experimental results are presented in section 4.5.

4.1 Advanced Mosaic Representation

As source images in a panorama spatially overlap with each other but are taken at different time instances and viewing directions (Fig. 2.6), there is a choice of *frame alignment policy* about how the different gray values available for the same pixel are combined. Similarly, the variations in pixel resolution between source images leads to the consideration of a *multi-resolution mosaic*. These issues, together with some

other extensions, would be described thoroughly in the following paragraphs.

4.1.1 Frame Alignment Policy

Various parametric motion models [77] have been suggested to cancel out or minimize the effects of camera motion, and approximate the motion of a dominant surface (usually the background) in the scene. In particular, the registration of individual images can be performed in one of the following three ways:

1. **Frame-to-frame**: The alignment parameters between successive frames for an entire image sequence are first computed. They can then be composed to obtain the alignment parameters between any two frames in the sequence. If a virtual coordinate system is adopted, an additional transformation between the virtual coordinate system and the reference frame (to which all images are aligned) needs to be given.

2. **Frame-to-mosaic**: One problem with the above alignment approach is that errors may accumulate during the repeated composition of alignment parameters. A simple solution is to directly use the transformations between each frame and the mosaic image as alignment parameters. Most of the examples in this report rely on this method.

3. **Mosaic-to-frame**: The frame-to-mosaic alignment is appropriate when a panorama is constructed with respect to a static coordinate system. However, in some dynamic application such as real-time video transmission, it is important to maintain the images in their input coordinate systems. In this case, it is more useful to align the mosaic to the current frame, so that the transformation between the mosaic and the current frame is identical to that between the previous frame and the current one.

4.1.2 Multi-resolution Representation

In dynamic mosaics, there may be unpredictable variations in image resolution within the sequence as a result of camera zooming. This can be handled by a multi-resolution mosaic data structure, which captures new information at its closest corresponding resolution level in a mosaic pyramid. When a frame is constructed from the mosaic pyramid, the highest existing resolution data in the mosaic which corresponds to the

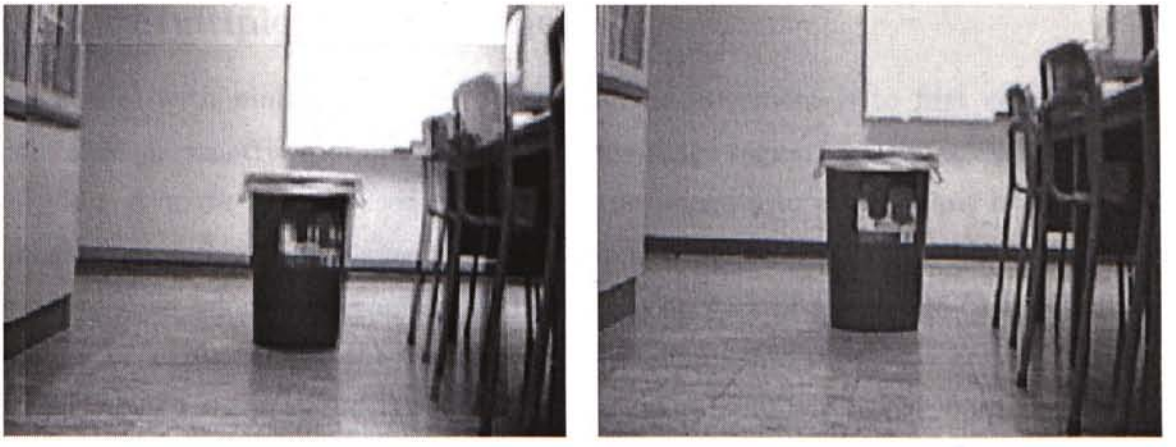

Figure 4.1: Synthesized frames: (left) without depth information, (right) with depth information for parallax handling.

frame is projected onto that frame. This data structure can be applied to static, dynamic, and temporal pyramid mosaic representations. It is different from temporal pyramid in that, the unit elements in the former representation are pixels, while those in the later pyramid are mosaic images.

4.1.3 Parallax-based Representation

For scenes with no independent object motion, their 3D compositions are usually invariant over short periods of time, and can be used to predict any parallax-induced motion over these intervals. Instead of using a *range map* (depths relative to the camera) to represent this information, it may be more efficiently described as a *height map* (depths with respect to a dominant surface in the scene) in a mosaic representation. This increased efficiency is due to the fact that values in a typical height map are significantly smaller than those of a depth map, and can therefore be much more compactly encoded.

An example of such a representation is shown in Fig. 4.1. Note particularly that objects at different depths exhibit parallax, and are not correctly registered by the 2D alignment transformations, leading to the many ghosts in the left image. Improvements are obvious after incorporating parallax motion information. More details about parallax can be found in [41][74][42].

4.1.4 Multiple Moving Objects

For scenes with multiple moving objects, an improvement is to first compute the dominant parametric motion, where all other image regions are detected as outliers [37][38]. A mask is then used to segment the image into a dominant layer (which image motion can be explained by the computed parametric transformation) and a residual layer which corresponds to the remaining part of the image (which motion cannot be explained by the transformation). The same technique can be applied recursively to the residual layer to find the next dominant transformation and its region within the image, etc. This pyramid-based approach locks on to the dominant image motion in the scene, and minimize sensitivity to noise.

4.1.5 Layers and Tiles

In principle, the 2D alignment model augmented with 3D parallax information is adequate for all scenes in which there is no independent object motion in the scene. However, when the scene begins to be cluttered with objects at widely varying depths, or when "fence-like" transparency is present, the parallax-based representation is highly inefficient. A natural extension to the 2D mosaic is to use multiple layers of 2D mosaics as suggested by Adelson [1], in which each layer can either embed a single moving object or a surface at a particular depth. In this way, the representation becomes somewhat more complex, but image distortions are reduced to minimal while making it more of a complete and efficient scene representation. This would be discussed in greater depths in later paragraphs.

4.2 Panorama Construction

In recent years, a number of techniques and software systems, e.g. QuickTime VR^{TM}, PhotoVista^{TM}, etc., have been developed for creating realistic mosaic images with little hardware support. With only regular photographic frames over a horizontal viewing space, these methods align component images and composite them into a complete panorama [57]. The resulting image can then be displayed with special purpose viewers, or alternatively be warped onto cylinders or spheres using texture-mapping. An example of such panorama is shown previously in Fig. 2.6.

At least four steps can be identified in this process: acquisition of source images,

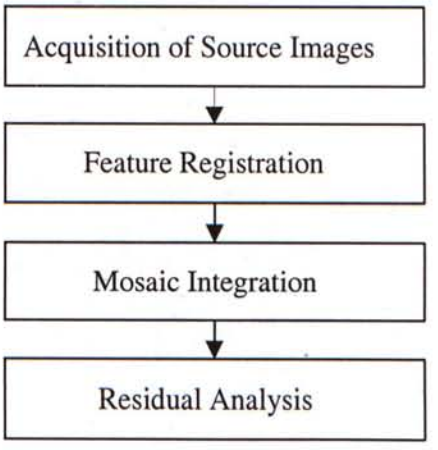

Figure 4.2: Steps in creating a panorama.

feature registration (alignment), integration into a mosaic image, and computation of significant residues between mosaic and individual frames (Fig. 4.2).

4.2.1 Image Acquisition

The first step in building a cylindrical panorama is to take a sequence of images with a camera mounted on a leveled tripod. In order to get good stitching results, each image should overlap its adjacent ones by around 25% to 50%, which amounts to about 16 photos for a full-view (360°) panorama. In our experiments, a leveled electronic tripod is used to rotate the camera at equal-angle increments of 22.5° so that a total of 16 images (with over 30% overlap between neighboring images) are taken to represent the complete field of view. An important assumption here is that there should be no significant movement perpendicular to the camera principal axis during image acquisition.

After some raw image enhancement processes (Chapter 2), each perspective image is warped onto a cylinder by mapping 3D world coordinates $\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ onto 2D cylindrical screen coordinates $\mathbf{q} = \begin{bmatrix} \theta & v \end{bmatrix}^T$ (provided that the camera focal length or field of view is known):

$$\theta = \tan^{-1}\left(\frac{x}{z}\right)$$
$$v = \frac{y}{\sqrt{x^2 + z^2}}$$

Figure 4.3: Camera setup for taking source images at a hot-spot.

where θ is the panning angle and v is the scan-line [77]. Similarly, we can map world coordinates into 2D spherical coordinates $\mathbf{q} = \begin{bmatrix} \theta & \phi \end{bmatrix}^T$ using

$$\theta = \tan^{-1}\left(\frac{x}{z}\right)$$
$$\phi = \tan^{-1}\left(\frac{y}{\sqrt{x^2 + z^2}}\right)$$

Indeed, creating panoramas in cylindrical or spherical coordinates has several limitations. First of all, it can only handle the simple case of pure panning motion. Second, even though it is possible to convert an image to 2D spherical or cylindrical coordinates for a known tilting angle, ill-sampling at north pole and south pole causes big registration errors, and leads to singularities in looking straight up or down the scene. Third, it requires knowing the camera focal length (or equivalently, field of view). While focal lengths can be carefully calibrated in lab [81], estimating them by registering two or more images is not very accurate. However, given the limited availability of sophisticated 3D imaging equipment, cylindrical maps are often easier

Figure 4.4: A sample panorama segment - notice how horizontal lines become curved.

to capture and more efficient in image warping than spherical or hyperbolic ones [11]. This explains the fact that cylindrical mosaics and their formulations are the usual choice most computer vision researchers prefer.

Fig 4.4 shows a sample panorama segment - notice how horizontal lines become curved. Once we have warped all input images, constructing mosaic images becomes a pure alignment problem, with compensations for minor vertical jitter and optical twist.

4.2.2 Image Alignment

Even with the best acquisition and pre-processing procedure, features on one image would not align with the corresponding features on another image without a good stitching algorithm. These algorithms vary greatly from one to another: some require carefully controlled camera motion (e.g. pure horizontal camera rotation) and calibrated intrinsic camera parameters (e.g. focal length), while others impose fewer restrictions or have higher error tolerance [57]. Despite their differences, there are three main approaches in use nowadays. The most straightforward one handles only pure panning motion, and uses feature point extraction methods (e.g. Moravec interest operator in Chapter 2) together with correlation to locate corresponding points among images. The second method (6-parameter affine transformations or 8-parameter planar perspective motion model) handles the case of general camera rotation. It makes use of camera projection models to warp source images onto cylindrical surfaces and

recover the translational motion between corresponding pixels through error minimization [77][78]. This mathematical approach is robust as long as the underlying physical interpretations are not violated. However, iterative algorithms or complex matrix transformations have to be used to minimize intensity errors and find the eight unknown coefficients in the camera projection model. As result, such algorithms require good initial guesses and may suffer from slow convergence. The third approach (3-parameter rotational motion model) is similar to the previous one, but with only three parameters to be estimated, it is faster and more robust [78]. In addition, we would describe a few extensions to alignment policies, e.g. frame alignment, patch-based alignment, as proposed by some researchers.

Simple 2D Image Alignment

To register two images I_0 and I_1, we need to estimate an incremental translation $\delta\mathbf{t} = \begin{bmatrix} \delta t_u & \delta t_v \end{bmatrix}^T$ by minimizing the intensity error $E(\delta\mathbf{t})$ between these images,

$$E(\delta\mathbf{t}) = \sum_i \left[I_1(\mathbf{q}'_i + \delta\mathbf{t}) - I_0(\mathbf{q}_i)\right]^2$$

where $\mathbf{q}_i = \begin{bmatrix} u_i & v_i \end{bmatrix}^T$ and $\mathbf{q}'_i = \begin{bmatrix} u'_i & v'_i \end{bmatrix}^T = \begin{bmatrix} u_i + t_u & v_i + t_v \end{bmatrix}^T$ are corresponding points in the two images, and $\mathbf{t} = \begin{bmatrix} t_u & t_v \end{bmatrix}^T$ is the global motion field (a horizontal translation t_u and a vertical translation t_v) for all pixels [6].

After a first order Taylor series expansion, the above equation becomes

$$E(\delta\mathbf{t}) \approx \sum_i \left[\mathbf{g}_i^T \delta\mathbf{t} + e_i\right]^2$$

where $e_i = I_1(\mathbf{q}'_i) - I_0(\mathbf{q}_i)$ is the intensity or color error, and $\mathbf{g}_i^T = \nabla I_1(\mathbf{q}'_i)$ is the image gradient of I_1 at $\mathbf{q}'_i$. This minimization problem has a simple least-square solution,

$$\left(\sum_i \mathbf{g}_i \mathbf{g}_i^T\right) \delta\mathbf{t} = -\left(\sum_i e_i \mathbf{g}_i\right)$$

Fig. 4.5 shows a portion of a cylindrical panoramic mosaic built using this simple translational alignment technique.

Figure 4.5: A segment of panorama built by simple 2D alignment.

Eight-parameter Planar Projective Transformations (Perspective Homography)

An important element of this framework is to associate each input image with an initial transformation estimate, by finding the best alignment between an image and a warped (resampled) mosaic image constructed from all previous images. Defining a warped image $\tilde{I}_1(\mathbf{q})$ of $I_1(\mathbf{q}')$ using some parametric motion model $\mathbf{q}' = f(\mathbf{q};\mathbf{t})$:

$$\tilde{I}_1(\mathbf{q}) = I_1(f(\mathbf{q};\mathbf{t}))$$

the trick is then to find incremental deformations of $\tilde{I}_1(\mathbf{q})$ which bring it into closer registration with $I_0(\mathbf{q})$. This reduces the problem to that of a parametric motion estimation, and greatly simplifies the computation of the gradients and Hessians required in gradient descent algorithms.

Mathematically, the relationship between two overlapping images (taken from the same viewpoint (optical center) but potentially at different directions and/or with different intrinsic parameters) can be described by a planar perspective transformation [55][39][77], which warps an image into another using

$$\mathbf{q}' \sim \mathbf{M}\mathbf{q} = \begin{bmatrix} m_0 & m_1 & m_2 \\ m_3 & m_4 & m_5 \\ m_6 & m_7 & m_8 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

where $\mathbf{q} = \begin{bmatrix} u & v & 1 \end{bmatrix}^T$ and $\mathbf{q}' = \begin{bmatrix} u' & v' & 1 \end{bmatrix}^T$ are homogeneous or projective coordinates, and $\sim$ indicates equality up to a scale factor. This equation can be rewritten as

$$u' = \frac{m_0 u + m_1 v + m_2}{m_6 u + m_7 v + m_8}$$
$$v' = \frac{m_3 u + m_4 v + m_5}{m_6 u + m_7 v + m_8}$$

To recover the parameters, we iteratively update the transformation matrix using

$$\mathbf{M} \leftarrow (\mathbf{I} + \mathbf{D})\,\mathbf{M} \tag{4.1}$$

or

$$\mathbf{M} \leftarrow \mathbf{T}\,(\mathbf{I} + \mathbf{D})\,\mathbf{T}^{-1}\mathbf{M}$$

where

$$\mathbf{D} = \begin{bmatrix} d_0 & d_1 & d_2 \\ d_3 & d_4 & d_5 \\ d_6 & d_7 & d_8 \end{bmatrix}$$

and $\mathbf{T}$ translates the origin from the top left corner of image plane to the center to improve conditioning of linear systems and speed up convergence.

Resampling image $I_1(\mathbf{q})$ with the new transformation $\mathbf{q}' \sim (\mathbf{I} + \mathbf{D})\,\mathbf{M}\mathbf{q}$ is the same as warping the resampled image $\tilde{I}_1(\mathbf{q})$ by $\mathbf{q}'' \sim (\mathbf{I} + \mathbf{D})\,\mathbf{q}$ (ignoring errors introduced by the double bilinear pixel resampling operation), i.e.

$$u'' = \frac{(1 + d_0)\,u + d_1 v + d_2}{d_6 u + d_7 v + (1 + d_8)}$$
$$v'' = \frac{d_3 u + (1 + d_4)\,v + d_5}{d_6 u + d_7 v + (1 + d_8)}$$

We wish to minimize the SSD error metric:

$$\begin{aligned} E(\mathbf{d}) &= \sum_i \left[\tilde{I}_1(\mathbf{q}_i'') - I_0(\mathbf{q}_i)\right]^2 \\ &\approx \sum_i \left[\tilde{I}_1(\mathbf{q}_i) + \nabla \tilde{I}_1(\mathbf{q}_i)\,\frac{\partial \mathbf{q}_i''}{\partial \mathbf{d}}\mathbf{d} - I_0(\mathbf{q}_i)\right]^2 \\ &= \sum_i \left[\mathbf{g}_i^T \mathbf{J}_i^T \mathbf{d} + e_i\right]^2 \end{aligned} \tag{4.2}$$

where $e_i = \tilde{I}_1(\mathbf{q}_i) - I_0(\mathbf{q}_i)$ is the intensity error, $\mathbf{g}_i^T = \nabla \tilde{I}_1(\mathbf{q}_i)$ is the image gradient of $\tilde{I}_1$ at $\mathbf{q}_i$, $\mathbf{d} = \begin{bmatrix} d_0 & \dots & d_8 \end{bmatrix}^T$ is the incremental motion parameter vector, and $\mathbf{J}_i = \mathbf{J}_d(\mathbf{q}_i)$ where

$$\mathbf{J}_d(\mathbf{q}_i) = \frac{\partial \mathbf{q}_i''}{\partial \mathbf{d}} = \begin{bmatrix} u & v & 1 & 0 & 0 & 0 & -u^2 & -uv & -u \\ 0 & 0 & 0 & u & v & 1 & -uv & -v^2 & -v \end{bmatrix}^T$$

is the Jacobian of the resampled point coordinate $\mathbf{q}_i$ with respect to $\mathbf{d}$.

This least-square problem in Eq. (4.2) can be solved using normal equations with symmetric positive definite solvers like Cholesky decomposition [67]. In practice, we set $d_8 = 0$, and only solve an 8×8 system:

$$\mathbf{A}\mathbf{d} = -\mathbf{b}$$

where

$$\mathbf{A} = \sum_i \mathbf{J}_i \mathbf{g}_i \mathbf{g}_i^T \mathbf{J}_i^T \tag{4.3}$$

is the Hessian, and

$$\mathbf{b} = \sum_i e_i \mathbf{J}_i \mathbf{g}_i^T \tag{4.4}$$

is the accumulated gradient or residues.

As a matter of fact, most examples in this report are based on this 2D alignment transformation. Such mosaics provide a complete representation of the scene segment in scenarios where there is little scene activity nor camera motion, or when the entire scene can be approximated by a single parametric surface (typically a plane). In practice, this algorithm is also a good approximation even when there are small violations to the above conditions. For instances, if a camera translates slowly and/or the relative distances between surface elements (Δz) are small compared with their ranges (z), the effects of parallax (as a result of camera motion) can be neglected. Similarly, if independent scene activity is confined to a small number of pixels, this effect can be neglected during mosaic construction. However, since the motion model contains

more free parameters than necessary, both of these 2D transformation recovery algorithms suffer from slow convergence and sometimes get stuck with local minima, if the initial transformation estimates are not close enough to their true values. In addition, scene changes and parallax are not represented in these 2D alignment frameworks, and have to be considered as residues. For these reasons, Shum [74] proposed the following 3-parameter rotational model.

Three-parameter Rotational Alignment

For a camera centered at the origin, the relationship between a 3D point $\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ and its image coordinates $\mathbf{q} = \begin{bmatrix} u & v & 1 \end{bmatrix}^T$ can be described by

$$\mathbf{q} \sim \mathbf{TVRp}$$

where

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & c_x \\ 0 & 1 & c_y \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{V} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } \mathbf{R} = \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix}$$

are the image plane translation, focal length scaling, and 3D rotation matrices respectively. For simplicity of notation, we assume that the origin is at the image center so that $c_x = c_y = 0$ and thus $\mathbf{T}$ can be ignored. The 3D direction corresponding to a screen pixel $\mathbf{q}$ is given by $\mathbf{p} \sim \mathbf{R}^{-1}\mathbf{V}^{-1}\mathbf{q}$.

For a camera rotating around its optical center, the mapping between two images I_k and I_l is therefore given by

$$\mathbf{M} \sim \mathbf{V}_k\mathbf{R}_k\mathbf{R}_l^{-1}\mathbf{V}_l^{-1} = \mathbf{V}_k\mathbf{R}_{kl}\mathbf{V}_l^{-1}$$

where each image is represented by $\mathbf{V}_k\mathbf{R}_k$, i.e. a focal length and a 3D rotation.

Assume for now that the focal length is known and is the same for all images, i.e. $\mathbf{V}_k = \mathbf{V}_l$. To recover the rotation, we perform an incremental update to $\mathbf{R}_k$ based on the angular velocity $\Omega = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$,

$$\mathbf{R}_{kl} = \hat{\mathbf{R}}(\Omega)\mathbf{R}_{kl} \text{ or } \mathbf{M} \leftarrow \mathbf{V}\hat{\mathbf{R}}(\Omega)\mathbf{R}_{kl}\mathbf{V}^{-1} \tag{4.5}$$

where the incremental rotation matrix $\hat{\mathbf{R}}(\Omega)$ is given by

$$\hat{\mathbf{R}}(\hat{\mathbf{n}}, \theta) = \mathbf{I} + \sin\theta \mathbf{X}(\hat{\mathbf{n}}) + (1 - \cos\theta)\mathbf{X}(\hat{\mathbf{n}})^2$$

$\theta = \|\Omega\|$, $\hat{\mathbf{n}} = \Omega/\theta$, and

$$\mathbf{X}(\Omega) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

is the cross product operator. Keeping only terms linear in Ω, we get

$$\mathbf{M}' \approx \mathbf{V}[\mathbf{I} + \mathbf{X}(\Omega)]\mathbf{R}_k\mathbf{R}_l^{-1}\mathbf{V}^{-1} = (\mathbf{I} + \mathbf{D}_\Omega)\mathbf{M}$$

where

$$\mathbf{D}_\Omega = \mathbf{V}\mathbf{X}(\Omega)\mathbf{V}^{-1} = \begin{bmatrix} 0 & -\omega_z & f\omega_y \\ \omega_z & 0 & -f\omega_x \\ -\omega_y/f & \omega_x/f & 0 \end{bmatrix}$$

is the deformation matrix which plays the same role as $\mathbf{D}$ in Eq. (4.1).

Computing the Jacobian of the entries in $\mathbf{D}_\Omega$ with respect to Ω and applying chain rule, we obtain the new Jacobian,

$$\mathbf{J}_\Omega = \frac{\partial \mathbf{q}''}{\partial \Omega} = \frac{\partial \mathbf{q}''}{\partial \mathbf{d}} \frac{\partial \mathbf{d}}{\partial \Omega} = \begin{bmatrix} -uv/f & f + u^2/f & -v \\ -f - v^2/f & uv/f & u \end{bmatrix}^T$$

This Jacobian is then plugged into the previous minimization pipeline to estimate the incremental rotation vector $\begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$, after which $\mathbf{R}_k$ can be updated using Eq. (4.5). Readers are referred to [41][74][70] for details about more complex 3D models and layered representations.

4.2.3 Image Integration

Once registration is finished, the desired image regions in I_0 and I_1 are aligned (or in the dynamic sense, the current mosaic and new frame are aligned), and can be integrated using any of following criteria:

1. **A temporal average or median filter** to produce panoramic images of the dominant background scene, while moving objects in the foreground either completely disappear or significantly fade out and leave image ghosts. Temporal averages usually result in blurrier mosaic images than those obtained by temporal medians.

2. **A weighted temporal average or median filter** to reduce discontinuity in intensity and alignment inaccuracies near image boundaries, as a result of the low order 2D parametric transformations (especially when the field of view is wide). We apply a simple *featuring algorithm* to weigh the pixels in each warped image inversely proportional to their Euclidean distances $d(\mathbf{q})$ to the nearest stitching edge (or more precisely, their distance to the nearest invisible or transparent pixel) [77]. We then blend all of the warped images using

$$C(\mathbf{q}) = \frac{\sum_k w(d(\mathbf{q}))\,\tilde{I}_k(\mathbf{q})}{\sum_k w(d(\mathbf{q}))}$$

where w is a monotonic function. In our current implementation, we use $w(x) = x$, and a seam is further divided into a number of segments (each of which intensity difference is resampled) in order to have even better stitching results.

3. **A most recent filter** to reflect the most update changes in the scene. This is especially useful in dynamic mosaic construction. Of course, the update can be more gradual by incorporating a decay parameter to give more weight to recent information, and forget those more distant in time.

Finally we can clip the ends (and optionally the top and bottom) of the integrated image, and write out a single panorama.

4.2.4 Significant Residual Estimation

Residues are information in the scene not represented in the mosaic since it was built. These differences between frames and mosaic occur for several reasons: object or illumination changes, camera motion parallax, interpolation errors during warping, and feature mis-alignments as a result of quantization of discrete pixel coordinates or intensity values. For instance, as most mathematical calculations are carried out in sub-pixel domain, rounding-errors are inevitably introduced when mapping back

to pixel coordinates. In addition, there may be uncertainty in manual processes like feature points selection.

On the other hand, the locations and magnitudes of residues differ between static and dynamic mosaics, even for the same set of component frames. For example, as moving objects (with respect to scene background) in static mosaics tend to blur out or even disappear, these changes will not appear in the resulting mosaic, and hence the residues between component frames will be significant. On the other hand, dynamic mosaics are constantly being updated with the most recent information, and therefore the changes in image regions as a result of independently moving objects will be smaller. However, owing to their dynamic nature, additional parts of the scene that are revealed to the camera for the first time would induce significant residues. This does not occur in the static case, as the support of a static mosaic is the union of all available frames.

As we have mentioned above, a complete mosaic representation also includes the residual differences between a panorama to individual frames, in addition to the alignment transformations described above. To reconstruct any given frame in its own coordinate system, a mosaic image is warped using the geometrical transformations and residues associated with that frame. This reconstruction is straightforward for static mosaics, as residues are directly estimated between the mosaic and every component frames. In dynamic mosaics, however, the residues are incremental with respect to the previous mosaic frame. In this case the reconstruction proceeds sequentially from frame to frame. The efficiency of this representation can be maximized by assigning heavier weights to semantically significant residues, by considering not only the residual intensity but also the magnitude of local mis-alignments. More details about this significance analysis are described in [38][39].

4.3 Advanced Alignment Algorithms

The normal equations given in previous sections, together with appropriately chosen Jacobian matrix and Hessian matrix, can be used directly to solve for motion estimates. Unfortunately, these gradient descent methods suffer from several drawbacks: they are susceptible to local minima and outliers, and are also unnecessarily inefficient. In addition, mis-registration errors accumulate for long image sequences, and result in visible gaps (or overlaps) between the two ends of a panoramic mosaic. In

this section, we present some improvements proposed by Shum [74] to make them more robust and efficient.

4.3.1 Patch-based Alignment

The computational effort required to take a single gradient descent step in parameter space can be divided into three major parts: (i) the warping (resampling) of $I_1(\mathbf{x}')$ into $\tilde{I}_1(\mathbf{x})$, (ii) the computation of the local intensity errors e_i and gradients $\mathbf{g}_i$, and (iii) the accumulation of the entries in $\mathbf{A}$ and $\mathbf{b}$ (Eqs. (4.3) and (4.4)). This last step can be quite expensive, since it involves the computations of the monomials in $\mathbf{J}_i$ and the formation of the products in $\mathbf{A}$ and $\mathbf{b}$. If we divide the image up into little patches P_j, and make the approximation that $\mathbf{J}(\mathbf{x}_i) = \mathbf{J}_j$ is constant within each patch (say by evaluating it at the patch center), we can write the normal equation as

$$\mathbf{A} \approx \sum_j \mathbf{J}_j \mathbf{A}_j \mathbf{J}_j^T \text{ with } \mathbf{A}_j = \sum_{i \in P_j} \mathbf{g}_i \mathbf{g}_i^T$$

and

$$\mathbf{b} \approx \sum_j \mathbf{J}_j \mathbf{b}_j \text{ with } \mathbf{b}_j = \sum_{i \in P_j} e_i \mathbf{g}_i$$

where $\mathbf{A}_j$ and $\mathbf{b}_j$ describe a local error surface. This new formulation therefore augments step (ii) above with the accumulation of $\mathbf{A}_j$ and $\mathbf{b}_j$ (only 10 additional multiply/add operations, which could potentially be done using fixed-point arithmetic), and performs the computations required to evaluate $\mathbf{J}_j$ and accumulate $\mathbf{A}$ and $\mathbf{b}$ only once per patch.

Once the displacements have been estimated for each patch, they must somehow be integrated into the global parameter estimation algorithm. The easiest way to do this is to compute a new set of patch Hessians $\mathbf{A}_j$ and patch residues $\mathbf{b}_j$ to encode the search results:

$$E(\mathbf{s}_j) = \mathbf{s}_j^T \mathbf{A}_j \mathbf{s}_j + 2\mathbf{s}_j^T \mathbf{b}_j + c = (\mathbf{s}_j - \mathbf{s}_j^*)^T \mathbf{A}_j (\mathbf{s}_j - \mathbf{s}_j^*) + c'$$

where

$$\mathbf{s}_j^* = -\mathbf{A}_j^{-1}\mathbf{b}_j$$

is the optimal flow estimate. More details about this formulation and its solution method can be found in [74].

4.3.2 Global Alignment (Block Adjustment)

In this section, we present a global alignment method that reduces accumulated error by simultaneously minimizing the mis-registration between all overlapping pairs of images. For a patch j in image I_k, let $l \in N_{jk}$ be the set of overlapping images in which patch j is totally contained. Let $\mathbf{q}_{jk}$ be the center of this patch. To compute the patch alignment, we use image I_k as I_0 and image I_l as I_1 and invoke the local search algorithm, which returns an estimated displacement $\mathbf{s}_{jl} = \mathbf{s}_j^*$. The corresponding point in the warped image $\tilde{I}_1$ is thus $\tilde{\mathbf{q}}_{jl} = \mathbf{q}_{jk} + \mathbf{s}_{jl}$. In image I_l, this point's coordinate is $\mathbf{q}_{jl} \sim \mathbf{M}_l\mathbf{M}_k^{-1}\tilde{\mathbf{q}}_{jl}$ or $\mathbf{q}_{jl} \sim \mathbf{V}_l\mathbf{R}_l\mathbf{R}_k^{-1}\mathbf{V}_k^{-1}\tilde{\mathbf{q}}_{jl}$ if the rotational panoramic representation is used.

Given these point correspondences, one way to formulate the global alignment is to minimize the difference between screen coordinates of all overlapping pairs of images

$$E(\{\mathbf{M}_k\}) = \sum_{j,k,l \in N_{jk}} \left|\mathbf{q}_{jk} - P\left(\mathbf{M}_k\mathbf{M}_l^{-1}\mathbf{q}_{jl}\right)\right|^2$$

where $P\left(\mathbf{M}_k\mathbf{M}_l^{-1}\mathbf{x}_{jl}\right)$ is the projected screen coordinate of $\mathbf{q}_{jl}$ under the inter-frame transformation $\mathbf{M}_k\mathbf{M}_l^{-1}$ ($\mathbf{M}_k$ could be a general homography, or could be based on the rotational panoramic representation). This has the advantage of being able to incorporate local certainties in the point matches (by making the above norm be a matrix norm based on the local Hessian $\mathbf{A}_{jk}$). The disadvantage, however, is that the gradients with respect to the motion parameters are complicated.

On the other hand, let the ray direction in the final composite image mosaic be a unit vector $\mathbf{p}_j$, and its corresponding ray direction in the kth frame as $\mathbf{p}_{jk} \sim \mathbf{R}_k^{-1}\mathbf{V}_k^{-1}\mathbf{q}_{jk}$. We can formulate block adjustment to minimize over pose ($\{\mathbf{R}_k, f_k\}$) for all frames I_k. More specifically, we estimate the pose by minimizing the difference in ray directions between all pairs (I_k and I_l) of overlapping images,

$$E(\{\mathbf{R}_k, f_k\}) = \sum_{j,k,l \in N_{jk}} |\mathbf{p}_{jk} - \mathbf{p}_{jl}|^2 = \sum_{j,k,l \in N_{jk}} \left|\mathbf{R}_l^{-1}\hat{\mathbf{q}}_{jk} - \mathbf{R}_l^{-1}\hat{\mathbf{q}}_{jl}\right|^2 \quad (4.6)$$

4.3.3 Local Alignment (Deghosting)

After the global alignment has been run, there may still be localized mis-registration present in the image mosaic, due to deviations from the idealized parallax-free camera model. Such deviations might include camera translation (especially for hand-held camera), radial distortion, the mis-location of the optical center (which can be significant for scanned photographs), and moving objects.

To compensate for these effects, we would like to quantify the amount of mis-registration and to then locally warp each image so that the overall mosaic does not contain visible ghosting (double images) or blurred details. If our mosaic contains just a few images, we could choose one image as the base, and then compute the optical flow between it and all other images, which could then be deformed to match the base. One possibility would be to explicitly estimate the camera motion and residual parallax [41][69][76], but this would not compensate for other distortions. Another approach might be to warp each image so that it best matches the current mosaic. For small amounts of mis-registration, where most of the visual effects are simple blurring (loss of detail), this should work fairly well. However, for large mis-registration when ghosting is present, the local motion estimation would likely fail.

An alternative approach, which is the one we have adopted, is to compute the flow between all pairs of images, and to then infer the desired local warps from these computations. While in principle, any motion estimation or optical flow technique could be used, we use the patch-based alignment algorithm, since it provides us with the required information and allows us to reason about geometric consistency.

Recall that the block adjustment algorithm (Eq. (4.6)) provides an estimate $\mathbf{p}_j$ of the true direction in space corresponding to the jth patch center in the kth image, $\mathbf{q}_{jk}$. The projection of this direction onto the kth image is

$$\bar{\mathbf{q}}_{jk} \sim \mathbf{V}_k\mathbf{R}_k \frac{1}{n_{jk}+1} \sum_{l \in N_{jk} \cup k} \mathbf{R}_l^{-1}\mathbf{V}_l^{-1}\mathbf{q}_{jl} = \frac{1}{n_{jk}+1}\left(\mathbf{q}_{jk} + \sum_{l \in N_{jk}} \mathbf{q}_{jl}\right)$$

or

$$\bar{\mathbf{q}}_{jk} \sim \mathbf{M}_k \frac{1}{n_{jk}+1} \sum_{l \in N_{jk} \cup k} \mathbf{M}_l^{-1} \mathbf{q}_{jl} = \frac{1}{n_{jk}+1} \left(\mathbf{q}_{jk} + \sum_{l \in N_{jk}} \mathbf{q}_{jl} \right)$$

if an 8-parameter perspective is used. This can be converted into a motion estimate

$$\bar{\mathbf{s}}_{jk} = \bar{\mathbf{q}}_{jk} - \mathbf{q}_{jk} = \frac{1}{n_{jk}+1} \sum_{l \in N_{jk}} (\tilde{\mathbf{q}}_{jl} + \mathbf{q}_{jk}) = \frac{1}{n_{jk}+1} \sum_{l \in N_{jk}} \mathbf{s}_{jl}$$

where $n_{jk} = |N_{jk}|$ is the number of overlapping images where patch j is completely visible.

The local motion required to bring patch center j in image m into global registration is simply the average of the pair-wise motion estimates with all overlapping images, down-weighted by the fraction $n_{jk}/(n_{jk}+1)$. This factor prevents local motion estimates from overshooting in their corrections (consider, for example, just two images, where each image warps itself to match its neighbor). Thus, we can compute the location motion estimate for each image by simply examining its mis-registration with its neighbors, without having to worry about what warps these other neighbors might be undergoing themselves.

4.4 Mosaic Application

4.4.1 Visualization Tool

The most obvious application of mosaic representations is as an enhance visualization tool that can provide a wide and stabilized field of view, and the necessary context for a viewer to better appreciate the events that take place in the scene. In addition, the 2D alignment methodologies mentioned in previous sections may be combined with the various mosaic representations and integration techniques to fit in with other application. For example, given contiguous scene subsequences (segmented from an input video clip), a *key frame mosaic* can be constructed for each scene subsequence as a snapshot view of the subsequence. This provides an even better representation of the most salient features in the scene than any single frame in the sequence. While such a mosaic is useful for capturing the background, in some cases it may be desirable to get a synopsis of the event that takes place within a video sequence. This can be achieved

through a mosaic that captures foreground events. A *synopsis mosaic* is constructed using the outlier maps obtained during the background alignment process. By using the inverse of outlier maps as weights during the integration process, foreground objects can be retained within the mosaic. It provides a snapshot view of the entire synopsis in the sequence, and is very useful for rapid browsing. On the other hand, a new video sequence (called *mosaic video*) consisting of a sequence of dynamic mosaic images may be generated. This type of visualization simulates the output of a virtual camera with any desired features, e.g. expanded field of view, or along any specific trajectories by applying the appropriate coordinate transformations to each of the mosaic video frames. The simplest example of this is stabilized mosaic video, in which case the camera motion is completely removed. Such a display is useful in various application like remote navigation and remote surveillance. The benefit of mosaic visualizations for various other application has also been recognized. Readers are referred to [74] for more application examples.

4.4.2 Video Manipulation

Video is a very rich source of information. Its two basic advantages over still images are the ability to obtain a continuously varying set of views of a scene, and the ability to capture the temporal evolution of phenomena. Since successive images within a video sequence usually overlap by a large amount, the use of panoramic images provides significant reductions in the total amount of data needed to represent the scene, and is thus becoming a popular and efficient way to describe or compress a collection of frames [55][75].

An entire video sequence can be represented by a mosaic image of the background scene, with the appropriate transformations that relate each frame. However, changes in the scene (e.g. moving objects) with respect to the background are not captured by the mosaic image, and additional representations are needed to take them into account [55][41][75]. Indeed, these residues can either be represented independently for each frame, or can be represented more efficiently as another mosaic layer [1]. The mosaic image, along with the frame alignment transformations, and with the residues together constitute a complete and efficient representation, from which the video sequence can be fully reconstructed. These issues have been addressed to a limited extent with respect to video compression in [1], although that work does not consider how to assign a significance measure to the residues or how to handle non-rigid layers.

A number of application has recently emerged that involve processing the entire information within video sequences, and require efficient representations of large video streams, and efficient methods of accessing and analyzing the information contained in video data. Typical application include video compression, scene change detection, digital libraries, low-bit-rate video transmission, and interactive video manipulation. In video editing environments, it is sometimes necessary to take a video segment and alter its content in post-production stages. For example, pulling an existing actor or object out of the sequence (while filling in the occluded regions convincingly), or inserting a non-existing object into the video sequence. These processes are currently very tedious, as they are done frame-by-frame manually. They can be significantly sped up, as well as done more accurately, using motion analysis and mosaic constructions. During sequence reconstruction from the mosaic, the changes made to the static mosaic image can be automatically applied to each of the individual frames of the sequence, since the coordinate relationships between the frames and the mosaic are known. On the other hand, panoramic views can also be used to index video data. For instance, it may suffice to retrieve the key frame mosaic alone (or alternatively the synopsis mosaic) when browsing or identifying video clips. Once a scene (mosaic) of interest has been detected, the part of the video tape which corresponds to it can be retrieved on demand.

4.5 Experimental Results

Three algorithms were implemented for building real-scene panoramas:

1. Eight-parameter planar projective transformation,
2. Three-parameter rotational alignment with patch-based compensation, local compensation and global compensation, and
3. Our modified version of the 8-parameter planar projective transformation with heavier use of spatial pyramid, correlations and local compensation.

In all of the experiments, we have used a fixed dummy focal length, patch size of 16 pixels, search range of four pixels, alignment accuracy of 0.04 pixel, and source image frames of size 384×300 pixels with initial mis-registration of about 30 pixels. Table 4.1 shows the computation time required by the algorithms in Visual C++

	(a)	(b)
8-parameter	38	117
3-parameter	53	513
Our algorithm	21	63

Table 4.1: Computation time in seconds for aligning (a) two image frames of size 384×300 pixels and initial mis-registration of about 30 pixels, and (b) 16 frames of the same size by the three stitching algorithms.

Figure 4.6: Sample panorama #1 by our stitching algorithm.

implementations on a Pentium 200MHz PC. The output panoramas of size 2700×300 pixels from these algorithms look very much alike, some samples are shown in Figs. 4.6 to 4.8, readers are referred to Appendix for more experimental results.

Figure 4.7: Sample panorama #2 by our stitching algorithm.

Figure 4.8: Sample panorama #3 by our stitching algorithm.

Chapter 5

Panoramic Walkthrough

In recent years, image-based rendering is becoming a popular way in the construction of virtual environments in computer graphics application. Instead of building a complete 3D model of the environment, a collection of images is used to synthesize the scene while supporting virtual camera motions. There are commercial software such as QuickTime VR^{TM} and PhotoVista^{TM} that provide tools for users to create and navigate inside image-based virtual environments. As suggested by Chen [11], the major procedures in building a virtual environment involves three basic elements (Fig. 5.1).

In the first step, a number of hot-spots of special interests are selected in the real environment. At each of these spots, a series of images is taken at every possible viewing direction with respect to an observer standing at that spot. Then by using the algorithms discussed in previous chapter, these individual images are stitched together to form panoramas, and pairs of corresponding features are located among these hot-spot panoramas to link them up. Finally, in a panoramic viewer, a user can click on these links (either as visual prompts like icons or messages showing that it is possible to jump from the current hot-spot to somewhere else) to replace the current panorama with that of its adjacent hot-spot. Navigation is therefore made possible by traveling from one node to another.

Instead of describing the whole process in details, this chapter would concentrate on the last element, and present our approach in creating a smooth transition between multiple views in a panoramic environment. The next section is the problem statement and notations. Section 5.2 gives a brief overview of the previous work related to panoramic walkthrough. Section 5.3 describes our walkthrough approach,

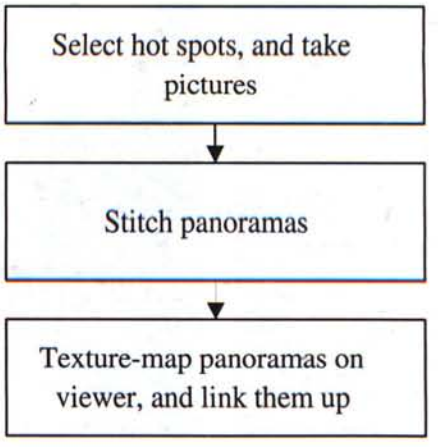

Figure 5.1: Procedure for building a virtual environment.

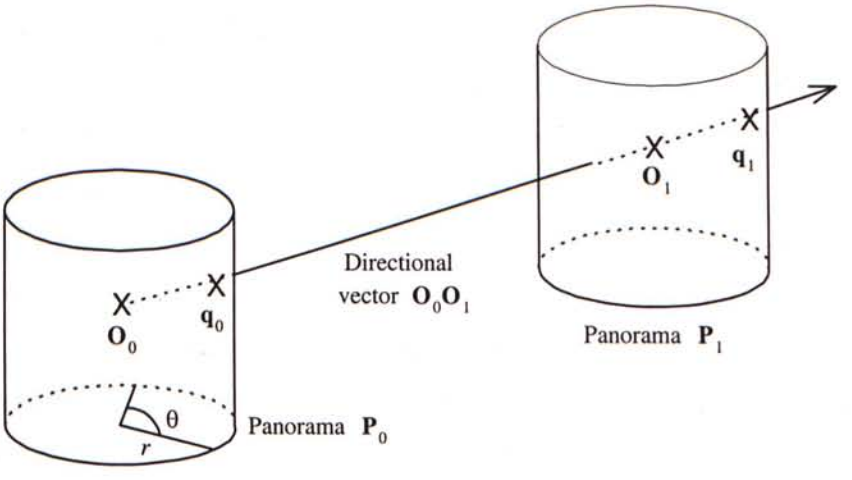

Figure 5.2: Notations for $\mathbf{P}_i$, $\mathbf{q}_i$, r and θ.

and experimental results on real images are presented in the last section.

5.1 Problem Statement and Notations

Given two neighboring panoramic nodes $\mathbf{P}_0$ and $\mathbf{P}_1$ with centers $\mathbf{O}_0$ and $\mathbf{O}_1$, r is the radius of the panorama, θ is the horizontal field of view angle, and $\mathbf{O}_0\mathbf{O}_1$ is the directional vector from $\mathbf{P}_0$ to $\mathbf{P}_1$ that intersects the two panoramas at image points $\mathbf{q}_0$ and $\mathbf{q}_1$ respectively (Fig. 5.2).

Every time a user looks through a panoramic viewer, only a portion of the whole panoramic image is visible on the image plane. Suppose we have two such image

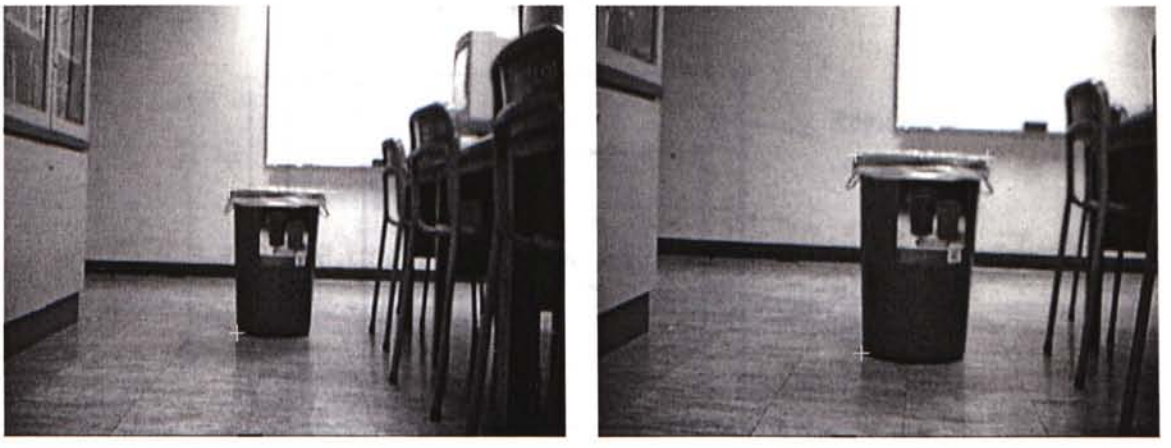

Figure 5.3: Sample I_0 (left) and I_1 (right).

segments I_0 and I_1 (both of size $m_0 \times n_0$ pixels, where m_0 is the number of rows and n_0 is the number of columns) from two adjacent panoramas $\mathbf{P}_0$ and $\mathbf{P}_1$. Without loss of generality, we consider that I_0 covers a large field of view, and I_1 is a close-up photo at nearly the same viewing angle as that in I_0 (Fig. 5.3).

If occlusion can be ignored, the image content in I_1 is assumed to be completely enclosed in a rectangular region inside I_0 of size $m_1 \times n_1$. Based on some feature correspondence $\mathbf{q}_{0j} = \mathbf{q}_{1j}$, the approximate position of I_1 inside I_0 can be found, which upper-left and lower-right corners are denoted by $\mathbf{w}_1$ and $\mathbf{w}_2$ respectively (so that a scaled-down version of I_1 would properly fit into the region enclosed by $\mathbf{w}_1$ and $\mathbf{w}_2$ in I_0, probably with some mis-alignments, Fig. 5.4).

With these definitions, the problem can now be stated formally as:

> Given panoramas $\mathbf{P}_0$ and $\mathbf{P}_1$, their field of view θ, camera focal length f, Euclidean norms among three arbitrary object points $\mathbf{p}_i$ in the scene $|\mathbf{p}_i\mathbf{p}_j|$ ($i \neq j, |\mathbf{p}_i\mathbf{p}_j| \neq 0$), and the three corresponding image points $\mathbf{q}_i$ and $\mathbf{q}_j$ on the mosaic images, estimate the depths for $\mathbf{p}_i$, and based on these approximate z_i, synthesize a sequence of image frames which shows a smooth transition from $\mathbf{P}_0$ to $\mathbf{P}_1$.

5.2 Previous Work

A key component in most VR systems is the ability to navigate in a virtual environment from different viewing positions and orientations. However, being able to move

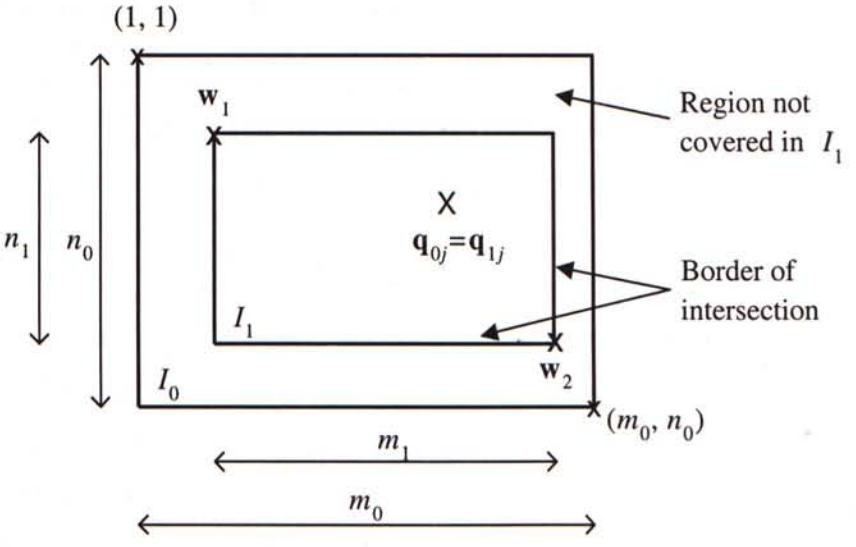

Figure 5.4: Merging I_1 into I_0.

freely in a photographic scene requires the change of both viewpoint and viewing direction, which is exceedingly difficult to solve. Although view interpolation may be a solution for computer rendered scenes, this method requires depth and camera information for automatic image registration, which is not easily obtainable from photographic scenes [75][43]. Previous navigation approaches require the synthesis of the virtual environment and the simulation of camera movements, which are usually accomplished with one of the following methods: 3D modeling/rendering, and branching movies.

5.2.1 3D Modeling and Rendering

A traditional approach to navigate in a virtual environment is to synthesized the scene as a collection of 3D geometrical entities, that are rendered in real-time and with the massive support from expensive graphic hardware. Despite the rapid advance of computer graphics software and hardware in the past, this approach still suffers from several major problems. First, building up geometrical models is a laborious manual process. Second, although special purpose 3D rendering engines are used to provide an interactive walkthrough experience, the real-time requirement often places a limit on the scene complexity and rendering quality.

5.2.2 Branching Movies

Another method that has been used extensively in the video game industry is branching movies. Multiple movie or panoramic segments depicting a virtual environment and its spatial navigation paths are connected together via some branching points, or nodes, that are linked up manually in the authoring stage. In this way, walking in a space is accomplished by "hopping" to different points. Indeed, when a user moves from one node to another, the panorama corresponding to the scene of the next node will replace the current panorama. This node-based approach solves most of the problems mentioned in the 3D approach, and does not require 3D modeling and rendering. An obvious problem with this approach, however, is its limited navigability and interaction. In addition, it requires every displayable view to be created and stored in the authoring stage that may sum up to a large amount of storage space. Furthermore, whenever a user jump from one node to another, a new panoramic scene usually replaces the old one instantaneously with little or no smooth view transition during the movement from one node to another. This could hardly provide users with any sense of scene immersion and is thus definitely unfavorable for a VR navigation system.

An early example of the movie-based approach is the Movie-map [45], in which the streets of the city of Aspen were filmed at 10-foot intervals. At playback time, two computer-driven laser-disc (LD) players were used to retrieve the corresponding views of photographic movies, or pre-rendered animation sequences interactively to simulate the effects of walking on the streets. With Digital Video Interactive technology [68], users are allowed to wander around a scene using digital video playback from optical disks. This decoupling of scene rendering from interactive playback allowed rendering to be performed at the highest quality with the greatest complexity without affecting the playback performance. Four cameras were used to shoot the views at every point in the authoring stage of Movie-map, thereby giving users the ability to pan to the left and right at every point. On the other hand, a Virtual Museum from CD-ROM-based computer rendered images was described in [59]. In this example, a 360^{o} panning movie (45 views were stored for each full-view pan movie) was rendered at selected points to let users look around. Walking from one point to another was simulated with a bi-directional transition movie, which contained a frame for each step in both directions along the path connecting the two points. This resulted in smooth panning motion but at the cost of more storage space and frame creation

time.

The Navigable Movie [4] is another example of the movie-based approach. Unlike the Movie-map or the Virtual Museum, which only have the panning motion in one direction, the Navigable Movie offers 2D rotation. An object was photographed with a camera pointing at an object's center and orbiting in both the longitude and the latitude directions at roughly 10^o increments. This process resulted in hundreds of frames corresponding to all the available viewing directions. The frames were stored in a 2D array which were indexed by two rotational parameters in interactive playback. When displaying the object against a static background, the effect was the same as rotating the object. Panning to look at a scene was accomplished in the same way. The frames in this case represented views of the scene in different view orientations.

After the conception of image-based rendering, creating a realistic scene with low-end PC become feasible. In particular, with the introduction of QuickTime VR^{TM} [11] and the pioneer work by McMillan *et al.* [57], many other navigation methods have been proposed towards better performance and higher realism. Recently, Hirose *et al.* proposed their virtual dome [30] for generating wide-range virtual environment. On the other hand, Darsa *et al.* also developed their image-based navigation system [15] by means of mesh triangulation and morphing. However, in most commercial panoramic packages, moving in space is currently accomplished by jumping to points where panoramic images are attached, in such a way that the whole panoramic view changes all of a sudden whenever a user walks from one node to another. In order to minimize the disturbing discontinuity in view transitions and to provide users with a more immersed feeling, some methods [15][29] have been proposed. Nevertheless, these algorithms require either high-performance machines or z-buffered images. To get rid of these limitations, a simpler and real-time algorithm which makes use of image scaling was proposed by Fu *et al.* [20]. Their implementation would be described in greater details below, and used as the basis for our discussions.

5.2.3 Texture Window Scaling

Theoretically speaking, changing the camera field of view is equivalent to zooming in and out in the image space. Let $\lambda \in [0, 1]$ be a walkthrough parameter (also known as the directional linear magnification factor) proportional to the relative distance between nodes, so that the whole I_0 (or I_1 respectively) will be displayed in the viewer when $\lambda = 0$ (or $\lambda = 1$ respectively), and that a sequence of in-between frames

Figure 5.5: Sample panoramic walkthrough results from direct texture window scaling (left and right: original panoramic frames, center: an in-between synthesized view).

from I_0 to I_1 can be generated along different values of λ. For example, given a starting node P_0 and an ending node P_1, nine intermediate views can be generated by scaling up or down I_0 and I_1 with $\lambda = 0.1, 0.2, ..., 0.9$. In short, a synthesized view $R(\lambda)$ can be described by the following rendering equation:

$$R(\lambda) = (1 - \lambda^n) I_0(\lambda) + \lambda^n f(I_1)$$

where $f(I_1)$ is the reposition function of I_1 inside $I_0(\lambda)$, and $n \in [1, \infty]$ describes the amount of occlusion (n will be large if the total feature matching error is large in order to reduce the amount of incorrect blending). For the actual rendering of these intermediate views, various interpolation or scaling methodologies, e.g. simple linear interpolation, positional weighted blending, view morphing, etc. have been suggested to mix I_0 and I_1 and to smooth out their pixel differences. In this way, by properly adjusting the magnification ratios for I_0 and I_1, the problem of panoramic walkthrough is simplified into a texture window scaling problem (Fig. 5.5).

5.2.4 Problems with Simple Texture Window Scaling

However, there exists several problems with this simple scaling of texture windows. First of all, owing to the inaccurate modeling of nonlinear camera projections, slight deviations in point correspondences among panoramas would be highly magnified as observable patch mis-alignments or image ghosts. This significant discontinuity in contrast or intensity levels among texture windows, and jagged edges near the stitching boundary can be easily seen at the center image in Fig. 5.5. In addition, although sophisticated smoothing filters and weighted patch-based merging techniques

have been applied in the stitching process, zooming out through image reduction may create aliasing artifacts as the sampling rate falls below the Nyquist limit. Direct synthesis under such conditions usually results in severe blurring, especially when the contents and qualities of source images differ by a rather large margin (close-up images usually contain more details than distant snapshots). Furthermore, using image magnification to zoom in does not provide more details. Instead, pixel enlargement could very probably lead to coarse images with ragged feature edges. Empirical experiments suggest that enlarging a typical image (of 300 dot per inch) by more than three times its original size will lead to observable glass-block effects. Another limitation associated with this method is the frequent image resampling that contributes to accelerated image quality degradation. Although this problem can be readily solved by using high resolution source images or other multi-resolution techniques, this is usually not a good choice for the increase in storage, equipment requirements and computational cost [57].

5.3 Our Walkthrough Approach

Our approach in achieving panoramic walkthrough would be described in later sections, after an overview of some modeling techniques used in dealing with cylindrical projection.

5.3.1 Cylindrical Projection onto Image Plane

As mentioned in section 5.1, only a portion of a complete panorama could be seen at a time. The exact coordinates or size of this texture window can be calculated from viewing parameters like camera field of view, size of panoramic cylinders, etc., as shown in Fig. 5.6.

Given $l_{visible} = r\theta$, where r is the radius of the panorama, and θ is the horizontal camera angle of view (in radian). A texture window on the image plane after cylindrical warping can be described by:

$$\phi_i = \frac{\theta}{w_{image}}(i - \frac{1}{2})$$

where ϕ_i is the angle from left hand side of the panorama for the i-th pixel column, and w_{image} is the horizontal resolution or width of the image plane. Then the height

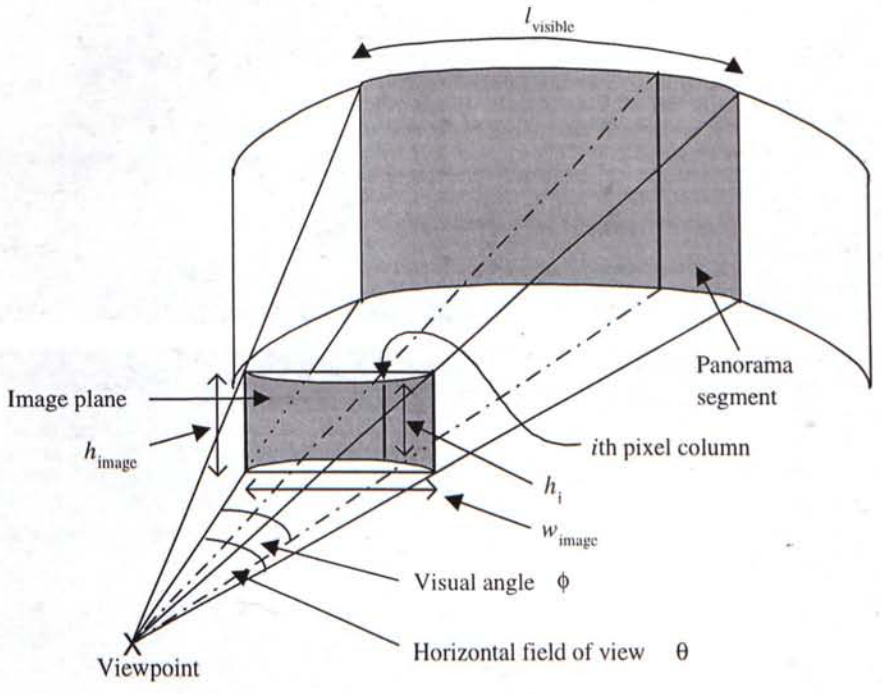

Figure 5.6: Cylindrical projection of panoramic segment onto an image plane.

of the i-th column in the image plane (h_i) can also be estimated:

$$h_i = h_{image}\left(1 - \frac{d_i - d_{min}}{r}\right)$$
$$= h_{image}\left[1 - \cos\left(\phi - \frac{\theta}{2}\right) + \cos\left(\frac{\theta}{2}\right)\right]$$

where h_{image} is the maximum height of the image plane, d_i is the object depth for column i in the image plane, and d_{min} is the minimal object depth given by

$$d_i = r\cos\left(\phi - \frac{\theta}{2}\right)$$
$$d_{min} = r\cos\left(\frac{\theta}{2}\right)$$

Fig. 5.7 shows a panorama segment with curved horizontal edges created with PhotoVista^{TM}, and its projection on the image plane after the full perspective cylindrical projection as discussed above. Note that the curved horizontal edges in the top image are rectified after this process.

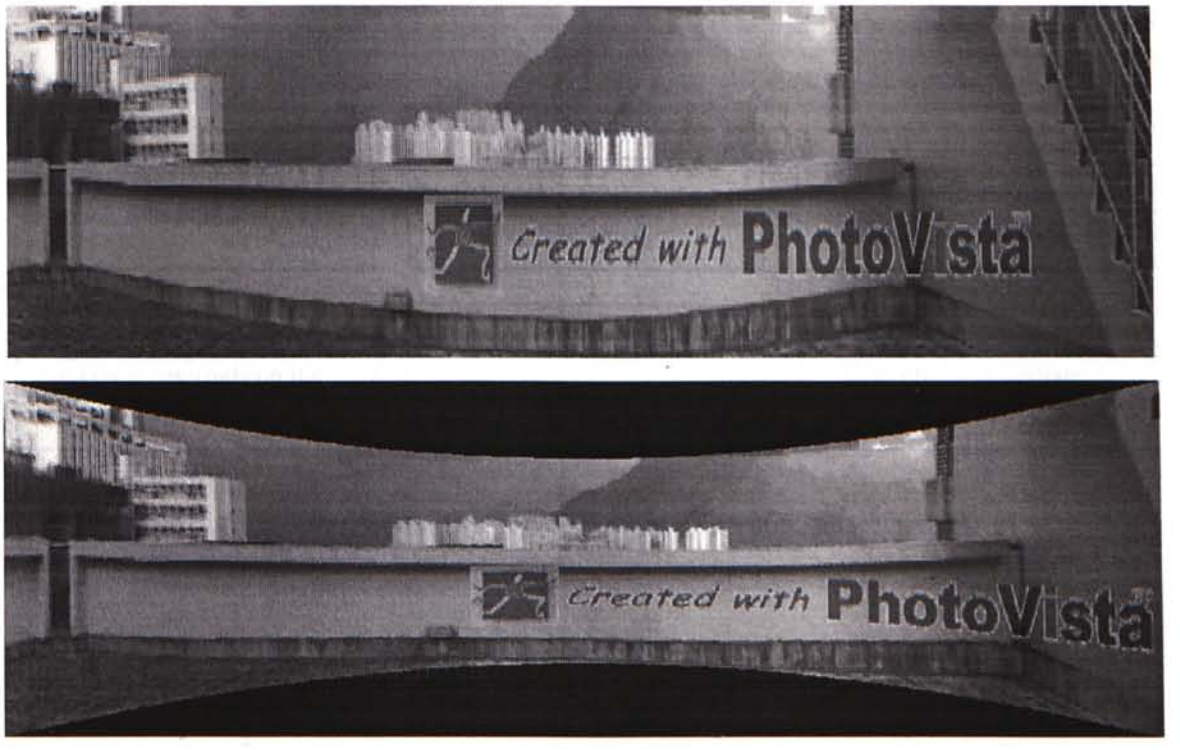

Figure 5.7: (Top) A panorama segment with curved horizontal edges. (Bottom) The same segment texture-mapped onto the inner surface of a cylinder, followed by full perspective projection.

5.3.2 Generating Intermediate Frames

Imagine what a user observes when he/she zooms-in an object with a physical camera. In the first place, the user could see a large part of the scene but with little details. When he/she zooms-in gradually, more details are revealed for each object in the scene. Our virtual zooming algorithm also makes use of a similar idea. The goal is that when a user click on an object in our viewer program, a sequence of intermediate frames is generated on-the-fly which progressively morph from a distant view to a near view to mimic a forward movement (Fig. 5.8). The main criteria are obviously the accuracy of synthesized intermediate frames, and how images segments are aligned. As shown in Fig. 5.9,

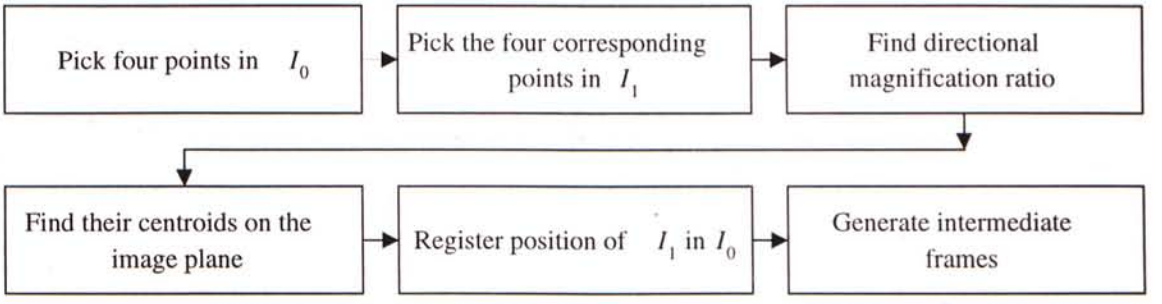

Figure 5.8: Procedure for generating intermediate frames.

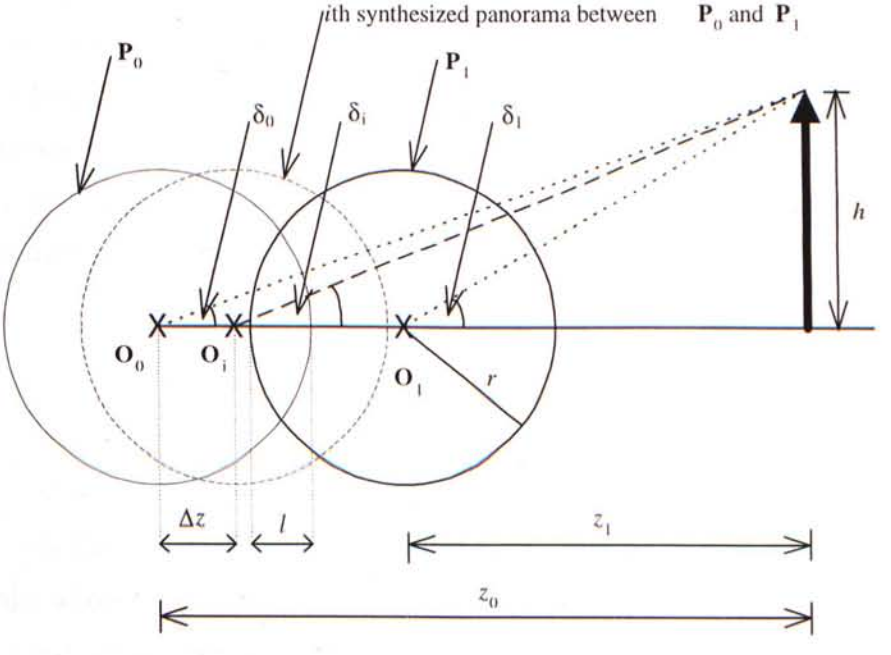

Figure 5.9: Relationship among visual angles between two panoramas.

$$\delta_0 = \tan^{-1}\left(\frac{h}{z_0}\right)$$
$$\delta_1 = \tan^{-1}\left(\frac{h}{z_1}\right)$$
$$\delta_i = \tan^{-1}\left(\frac{h}{z_0 - \Delta z}\right)$$
$$= \tan^{-1}\left(\frac{z_0 \tan \delta_0}{z_0 - \Delta z}\right)$$

where δ_0, δ_1 and δ_i are the visual angles of an object as viewed at nodes $\mathbf{P}_0$, $\mathbf{P}_1$ and $\mathbf{P}_i$ respectively, l is the maximum overlapping of two panoramas, h is the height of the object, z_0 and z_1 are the depth information of the object from $\mathbf{P}_0$ and $\mathbf{P}_1$ respectively, and Δz is the walkthrough distance from $\mathbf{P}_0$ to $\mathbf{P}_i$.

For each possible pair of I_0 and I_1, N pairs of corresponding points $\mathbf{q}_{ki} = (u_{ki}, v_{ki})$ are selected as input, where $k = 0$ or 1 is an index to the two source images, and $i = 1, ..., N$ in symbols $\mathbf{q}$, u or v is an index to the i-th corresponding point. This extraction process may be automatic with feature extraction and correlation algorithms. However, in our current implementation, we use four pairs of manually-selected points because both feature extraction and correlation are time-consuming processes. By involving a little user intervention, the running time of our program can be tremendously shortened. These points are then used to calculate the linear magnification ratio of I_0 with respect to I_1 along both u-axis and v-axis, so that $\mathbf{w}_1$ and $\mathbf{w}_2$ are given by simple mathematics:

$$\mathbf{w}_1 = \frac{1}{2}\begin{bmatrix} m_0 - m_1 & n_0 - n_1 \end{bmatrix} + \boldsymbol{\eta}_{\mathbf{c}_0\mathbf{c}_1}$$
$$\mathbf{w}_2 = \mathbf{w}_1 + \begin{bmatrix} m_1 & n_1 \end{bmatrix}$$

where $\boldsymbol{\eta}_{\mathbf{c}_0\mathbf{c}_1}$ is the offset of object centroids.

After the above calculations, we can identify the region in I_0 which do not appear in I_1. Suppose we are going to generate a frame sequence of M frames (including I_0 as the zeroth frame and I_1 as the $(M-1)$-th frame) to zoom-in gradually from I_0 to I_1, we would divide those uncovered regions in I_0 into $M-1$ rings. For example, to compose the j-th frame I_{2j} (note that we have introduced a second subscript $j = 0, ..., M-1$ as the frame index, in addition to the first subscript $k = 0, 1$ or 2 as

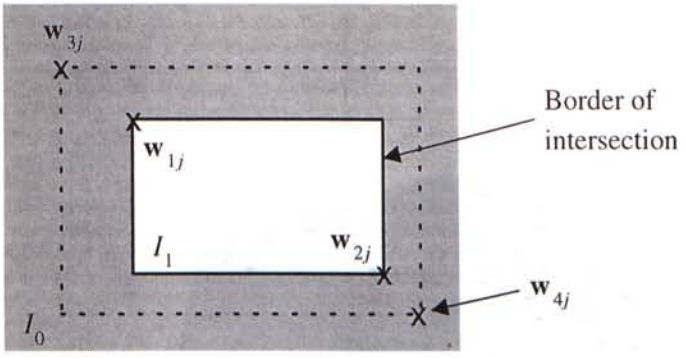

Figure 5.10: Boundary coordinates of the $M-j-1$ rings in I_0.

image index), we first calculate the boundary coordinates of the inner most $M-j-1$ rings in I_0.

In the next step (Fig. 5.10), all we need to do is to magnify the region enclosed by $\mathbf{w}_{3j}$ and $\mathbf{w}_{4j}$ in I_0 to the original image size $m_0 \times n_0$ pixels to give I_{0j}, and scale down I_1 from size $m_0 \times n_0$ to $m_{1j} \times n_{1j}$ to give I_{1j}. By overlapping I_{0j} and I_{1j} within a region $\mathbf{B}$ bounded by $\mathbf{w}_{1j}$ and $\mathbf{w}_{2j}$ (see below), we have our j-th resulting frame I_{2j} in our movie sequence. In our current implementation, all image resizing are done by bilinear interpolation, though it would inevitably lead to quality degradation and loss of details. In addition, the 3D counterparts of $\mathbf{q}_{ki}$ should at best lie on a plane parallel to the focal plane to get more accurate estimates. In an ideal case when the approximations are accurate, the centroids of I_0 and I_1 should always overlap at a point. However, owing to input error and rounding error in practice, the calculated centroid positions are only very close to each other.

Mathematically $\mathbf{w}_{1j}$ to $\mathbf{w}_{4j}$ are given by:

$$
\begin{aligned}
\mathbf{w}_{1j} &= \mathbf{c}_{0j} - \mathbf{c}_{1j} \\
\mathbf{w}_{2j} &= \mathbf{w}_1 + \begin{bmatrix} m_{1j} & n_{1j} \end{bmatrix} \\
\mathbf{w}_{3j} &= j\left(\boldsymbol{\delta}_{\mathbf{c}_{I_0}\mathbf{c}_{I_1}} - \boldsymbol{\delta}_{\mathbf{c}_{I_0}\mathbf{c}_0}\right) \\
\mathbf{w}_{4j} &= \begin{bmatrix} m_0 & n_0 \end{bmatrix} - j\left(\boldsymbol{\delta}_{\mathbf{c}_{I_0}\mathbf{c}_{I_1}} + \boldsymbol{\delta}_{\mathbf{c}_{I_0}\mathbf{c}_0}\right)
\end{aligned}
$$

where $\mathbf{c}_{kj}$ is the object centroids from I_k with respect to the resulting image I_{2j}, $\begin{bmatrix} m_{1j} & n_{1j} \end{bmatrix}$ is the new image size for I_{1j}, $\boldsymbol{\delta}_{\mathbf{c}_{I_0}\mathbf{c}_{I_1}}$ is the incremental offset of image plane centroids, and $\boldsymbol{\delta}_{\mathbf{c}_{I_0}\mathbf{c}_0}$ is the incremental offset between image plane centroid and

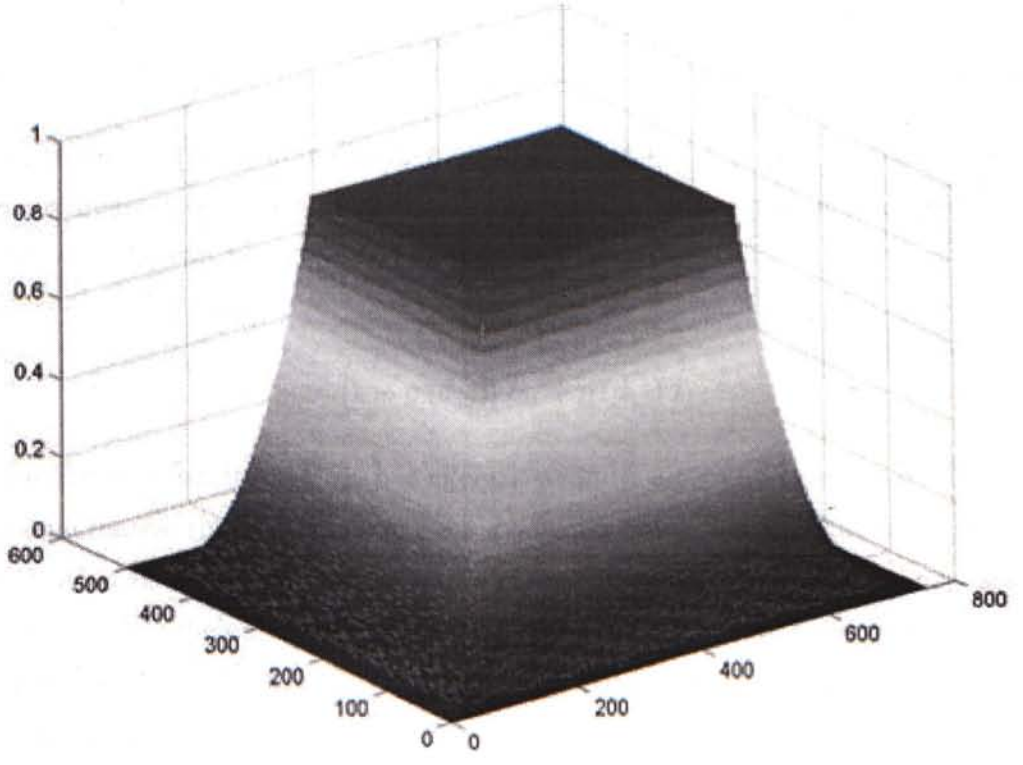

Figure 5.11: Weight function $g(d_{1j})$ for stitching images.

object centroid (cf. Appendix for derivation).

In addition, a weight function $g(d_{1j}) \in [0,1]$ is defined (Fig. 5.11) with respect to I_{1j}. This function has two important characteristics: First of all, it has a constant value of 1 for $\begin{bmatrix} u & v \end{bmatrix}^T \in \mathbf{B}$ such that the central area of I_{2j} consists of mainly I_{1j} (as can be seen at the flat mountain tip in Fig. 5.11). Secondly, a cubic function in Eq. (5.1) is used to decrease its weight exponentially towards the image boundaries (as can be seen at the peripheral cubic slopes). It should be note that our particular choice of fall-off function is not unique, Eq. (5.2) or similar functions can also be used.

$$\begin{aligned} g(d_{1j}) &= 1, \begin{bmatrix} u & v \end{bmatrix}^T \in \mathbf{B} \\ &= \frac{(d_{2j} - 1 - d_{1j})^3}{d_{1j}^3 + (d_{2j} - 1 - d_{1j})^3}, \begin{bmatrix} u & v \end{bmatrix}^T \notin \mathbf{B} \end{aligned} \tag{5.1}$$

$$g(d_{1j}) = \cos\left(\frac{d_{1j}\pi}{2(d_{2j}-1)}\right) \tag{5.2}$$

where d_{1j} is the perpendicular distance of an image point $\begin{bmatrix} u & v \end{bmatrix}^T$ under consideration to the nearest edge of region $\mathbf{B}$, and d_{2j} is the maximum among all d_{1j}.

In this way, I_{2j} can be given by the stitching formula:

$$I_{2j} = [1 - g(d_{1j})]\, I_{0j} + g(d_{1j})\, I_{1j}$$

so that for the first few frames, I_{0j} has exclusive significance and less is taken from I_{1j}. Then as j increases, the contribution from I_{1j} grows quickly while that for I_{0j} decays. The motivation of using this rather odd weight function is that, to compensate for the problem of insufficient resolution in digital zooming, I_{0j} must contribute heavily for the first few frames while I_{1j} plays a major role in subsequent frames. Indeed, $I_{20} = I_0$ and $I_{2(M-1)} = I_1$. The final effect would be like morphing from I_0 to I_1.

Up to this stage, a series of intermediate frames can be generated. However, owing to lens imperfection and accumulative errors in coordinate calculations, noticeable ghosting or stitching borders may be found in synthesized frames, even they are gradually fading away. An example is shown in Fig. 5.12 (left), which is the fifth frame of a 10-frame sequence. Note the blurring around frame center as a result of combining I_{04} and I_{14}. To tackle this problem, we have to take into account the merging of I_{0j} and I_{1j} more carefully, and use certain image stitching strategies. A much better result is obtained in Fig. 5.12 (right), which clearly demonstrates the improvement and effectiveness of our strategies.

First of all, a correlation filter is implemented to locate genuine feature points in the neighborhood of user-selected points, and register these probably more accurate and reliable feature points instead. Since only a few small windows are of concern, the computational efforts in refining manually-selected feature points are minimal. In addition, I_0 and I_1 are first pre-processed to smooth out any observable difference in intensity levels or image quality. However, its computational complexity may sometimes out-balanced the improvement in image quality, and there is a trade-off between output image quality and algorithm execution speed.

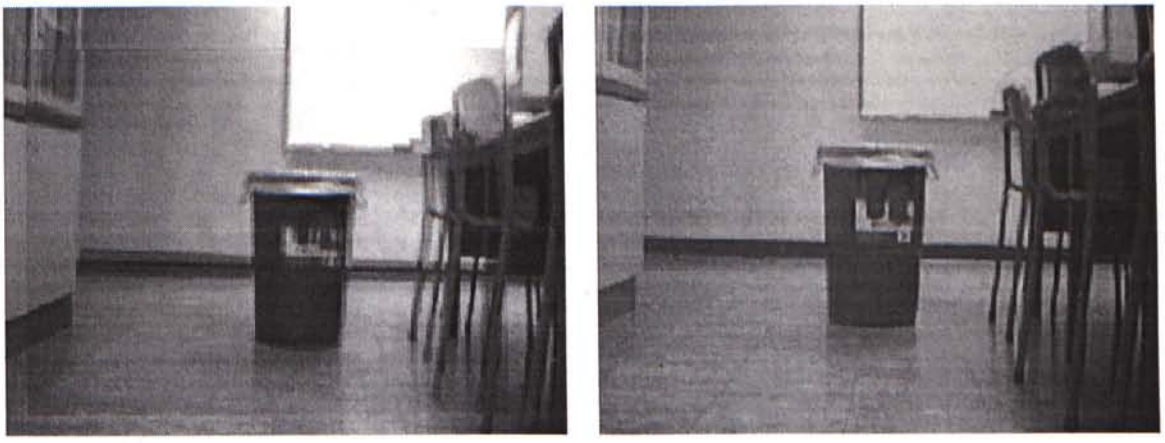

Figure 5.12: Synthesized frames: (left) without stitching optimization, (right) with optimization.

5.3.3 Occlusion Handling

Theoretically speaking, forward navigation from $\mathbf{P}_0$ to $\mathbf{P}_i$ induces an occlusion problem on the background scene, as can be seen by the fact that $\eta_{00} \geq \eta_{01}$ in Fig. 5.13 (the second subscript denotes time instants). This problem is due to the fact that according to the formulation of the full perspective projection, the projected size of an object in the image plane Ψ_i depends on its dimensions in the 3D object space and varies inversely-proportional to its depth (z_i), so that a nearby object will always has a higher rate of change in image size than that of a distant object whenever there is a transition of viewpoint:

$$\left|\frac{\partial \Psi_i}{\partial z_i}\right| \propto \frac{1}{z_i}$$

and

$$\left|\frac{\partial \Psi_i}{\partial z_i}\right| > \left|\frac{\partial \Psi_j}{\partial z_j}\right|$$

for any $z_i < z_j$, where $|.|$ denotes absolute value in this occasion. These unequal rates of change in image sizes create an occlusion problem that some part of the background would be occluded by nearby objects in front of the camera, especially when the background scene is very far behind those foreground objects. But for practical purposes, if the depth disparity between foreground and background is small

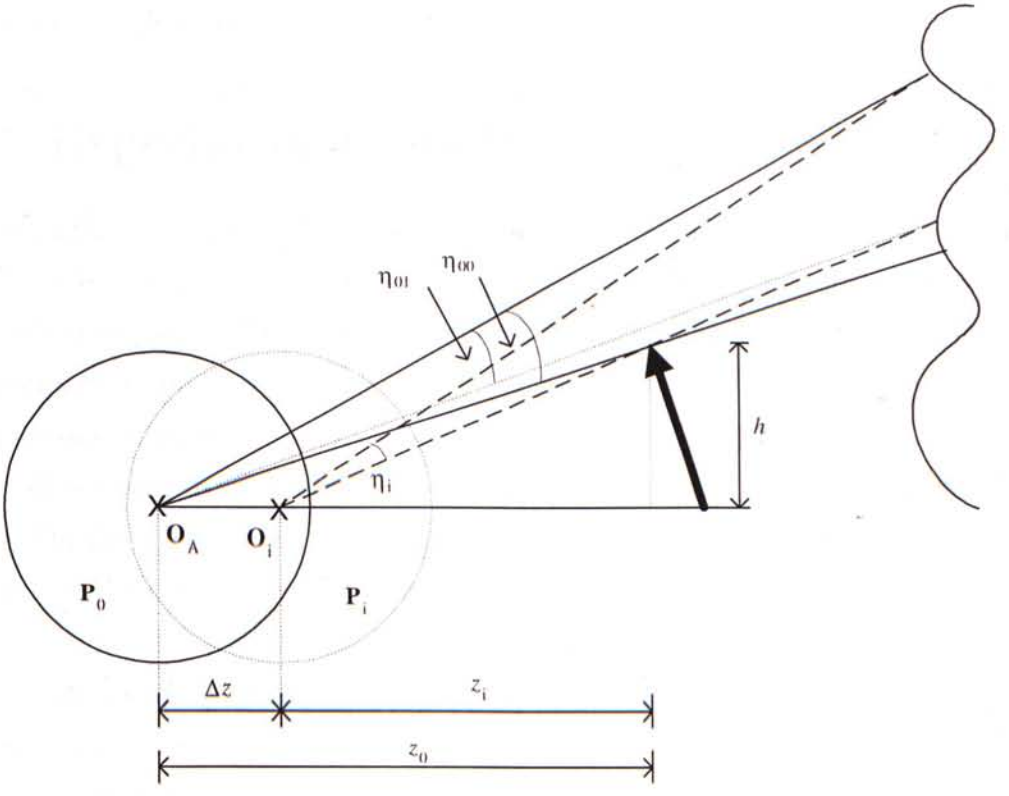

Figure 5.13: Background occlusion by nearby objects in panoramic walkthrough.

relative to that between camera and background, and that the translation in camera viewpoint Δz is small, we may take the following assumption:

$$\eta_{01} = \eta_i$$

Without loss of generality,

$$\eta_i = \frac{\theta}{2} - \delta_i$$

so that given any three features visible in both panoramas with their dimensions in the 3D object space , we could further enhance our walkthrough algorithm to handle partial occlusions:

1. Get a circular patch of panoramic segment of visual angle $[0, \delta_0]$ from $\mathbf{P}_0$.

2. Scale up this segment so that it covers a visual angle $[0, \delta_i]$ from $\mathbf{P}_i$.

3. Get another circular patch of panoramic segment of visual angle $[\frac{\theta}{2} - \eta_i, \frac{\theta}{2}]$ to cover a visual angle of $[\delta_i, \frac{\theta}{2}]$ in $\mathbf{P}_i$.

5.4 Experimental Results

Eleven panoramic images of size 2700×300 pixels were used to test our algorithm. Figs. 5.14 to 5.16 show part of the movie sequences. For each figure, frames (a) and (f) are the two panoramic segments (I_0 and I_1), while frames (b) to (e) are synthesized intermediate frames between I_0 and I_1.

With our Matlab 5.1 implementation executed on a Pentium II 300 PC, it takes about 20 minutes to synthesize a 10-frame movie sequence at a resolution 500×300 pixels. On the other hand, our Visual C++ implementation takes about 15 seconds to generate the same movie sequence at an even higher resolution 756×504 pixels on the same machine. Such processing time can be further reduced if source images of lower resolutions, e.g. 300×200 pixels, are used. Although there are noticeable delays in generating a sequence on-the-fly, the resulting image quality is much superior to the case of digital zooming (pixel enlargement), as can be seen from the sharp feature outlines in the figures above. This clearly demonstrates the value of our algorithm, and justifies itself for the cost.

5.5 Discussions

We have implemented an algorithm to bridge two mosaic nodes smoothly and realistically. Though there is still room for improvements, our methods can synthesize intermediate movie frames to simulate panoramic walkthrough with minimal sacrifice in image quality and execution speed. We are currently investigating the following possible enhancements:

1. Other computer vision tools may be applied to enhance both the speed and output quality in panoramic walkthrough. For example, as there are viewpoint difference among starting and ending nodes, epipolar geometry [89] may be used to estimate the fundamental matrix $\mathbf{F}$ and transform the 2D feature matching search space down to 1D. Together with model fitting methods like RANSAC [19], information about $\mathbf{F}$ helps to drop out significant portions of false matches which effectively increase the image quality in synthesized views.

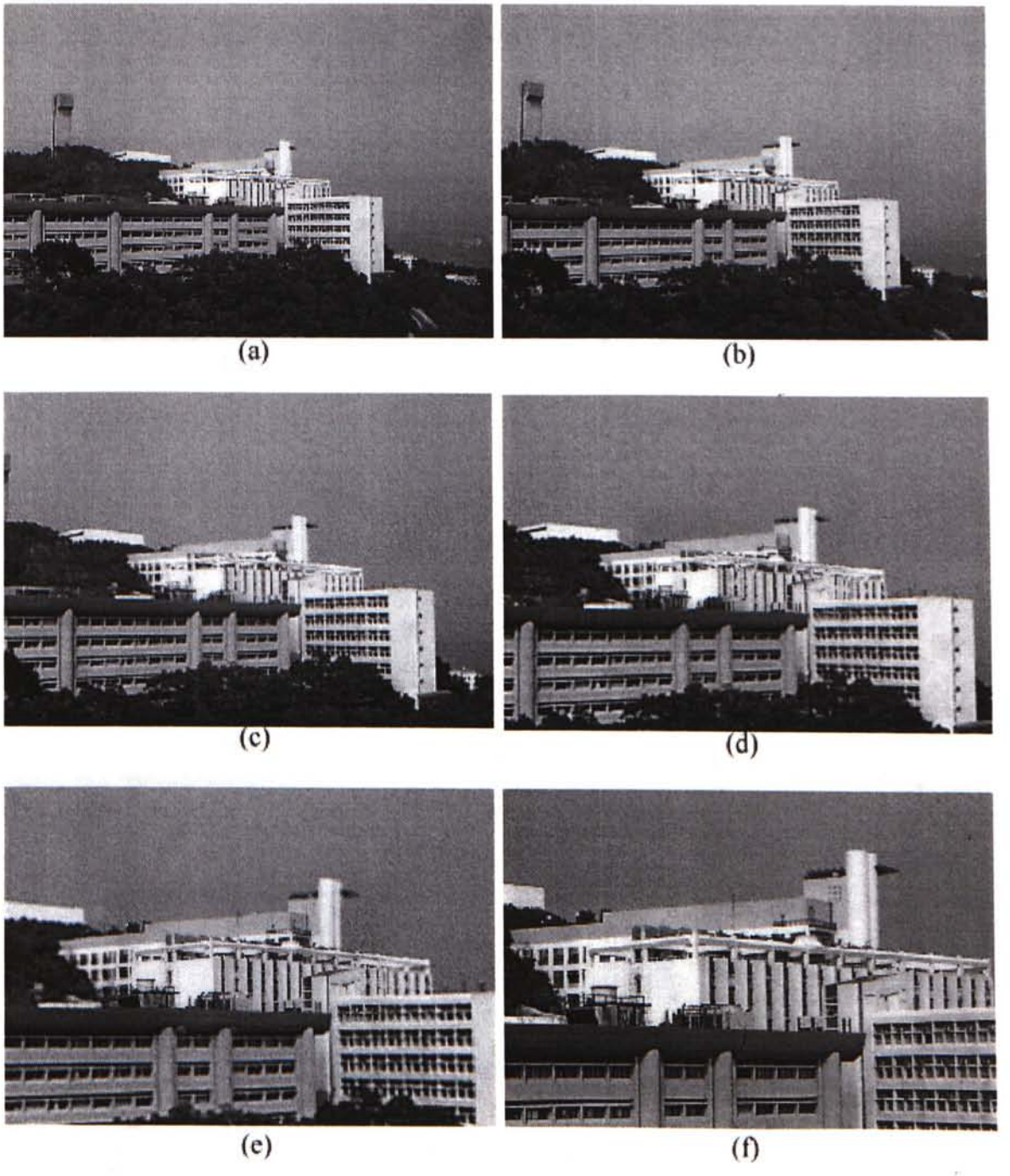

Figure 5.14: Sample movie sequence #1. Frames (a) and (f) are source images, and frames (b) to (e) are synthesized intermediate frames.

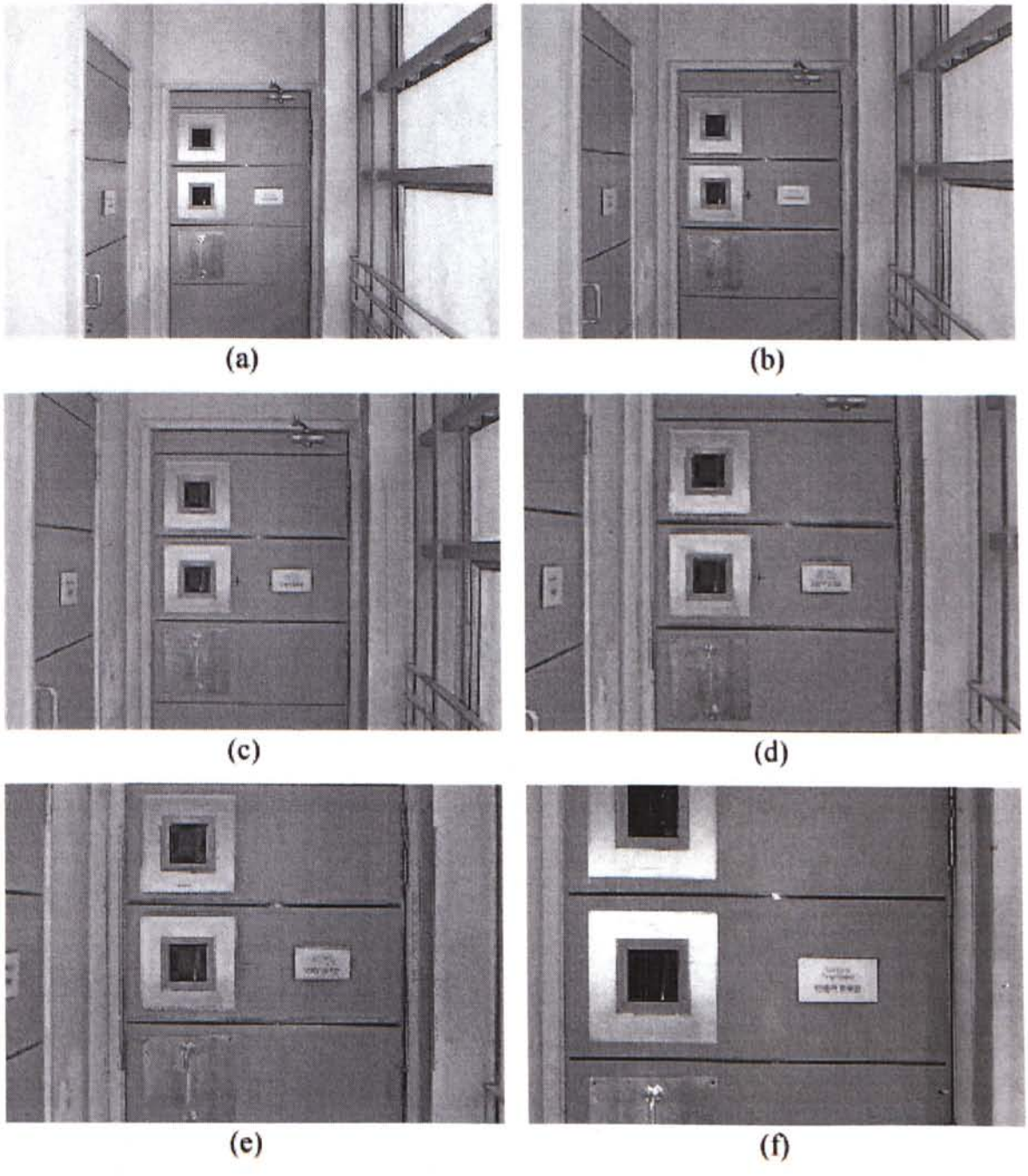

Figure 5.15: Sample movie sequence #2. Frames (a) and (f) are source images, and frames (b) to (e) are synthesized intermediate frames.

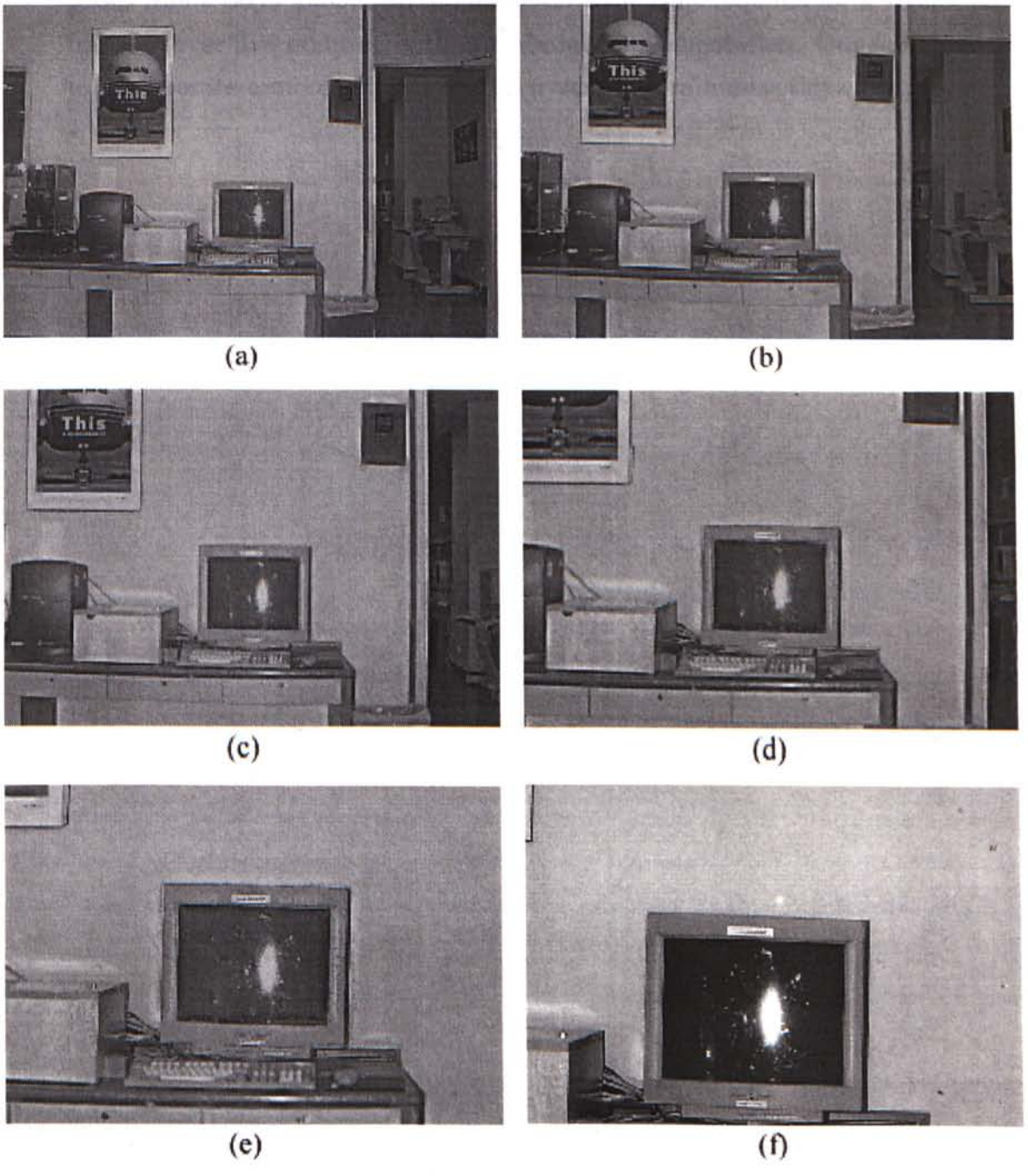

Figure 5.16: Sample movie sequence #3. Frames (a) and (f) are source images, and frames (b) to (e) are synthesized intermediate frames.

2. The performance of our panoramic viewer is greatly affected by the resolution and size of the source images. For example, synthesizing a single 500×300 pixels frame takes about two minutes, while the time required for a 1000×500 frame is over five minutes, with our Matlab implementation. Our next goal is to incorporate a more efficient spatial pyramid to minimize this effect.

Chapter 6

Conclusion

An iterative algorithm based on the Gauss-Newton method for the Perspective-3-Point (P3P) problem is derived. A majority of existing methods make use of linearized camera models (e.g. weak-perspective projection model) to ameliorate problems that arise with closed-form perspective solutions (such as ambiguity and ill-conditioned error propagation). Unfortunately, these models are approximations which validity is sometimes questionable in practice. In our system, by given the 3D dimensions of a rigid body in the object space, its motion parameters, $\mathbf{R}$ and $\mathbf{t}$, are calculated from subsequent image points under the full perspective projection model. Both real images and synthetic data have been used to verify our algorithm, showing that it is efficient (a thousand frames in 0.11 sec.) and may be employed for real-time applications. Another major contribution of our work is the enhancements to the classic P3P formulation to reduce multiple solutions. Applying our proposed constraints to the P3P calculations greatly simplifies the efforts in locating unique control points for view synthesis. Model fitting methods like RANSAC may be used to further improve the system reliability.

On the other hand, a robust model-based walkthrough algorithm is proposed and implemented. Traditional approaches to navigate in a virtual environment are usually accomplished by viewpoint-hopping techniques (jumping from one node to another instantaneously) that provide users with little sense of scene immersion. Also they require massive storage space and support from expensive graphic hardware. By applying computer vision techniques like P3P, our method can estimate the depths of feature points in the 3D space from their subsequent image point correspondences. Based on this sparse point correspondences and the 3D object dimensions, appropriate

segments are extracted out from source mosaic images to synthesize intermediate views. Real panoramic images have been used to verify our method with satisfactory results. More research work will be focused on relaxing the requirements on *apriori* point correspondences or feature dimensions, and the further application of computer vision methodologies in creating realistic walkthrough video sequences.

Appendix A

Formulation of Fischler and Bolles' Method for P3P Problems

With all the assumptions and simplifications made in Chapter 3, and by using Cosines rule for each pair of line-of-sight, we can get a set of quadratic equations involving the unknowns. The resulting system in l_{12}, l_{13} and l_{23} is shown below:

$$
\begin{aligned}
l_{12}^2 &= |\mathbf{Op}_2|^2 + |\mathbf{Op}_3|^2 - 2\,|\mathbf{Op}_2|\,|\mathbf{Op}_3|\,\rho_{23} \\
l_{13}^2 &= |\mathbf{Op}_1|^2 + |\mathbf{Op}_3|^2 - 2\,|\mathbf{Op}_1|\,|\mathbf{Op}_3|\,\rho_{13} \\
l_{23}^2 &= |\mathbf{Op}_1|^2 + |\mathbf{Op}_2|^2 - 2\,|\mathbf{Op}_1|\,|\mathbf{Op}_2|\,\rho_{12}
\end{aligned}
$$

where

$$
\begin{aligned}
\rho_{23} &= \cos\left(\angle \mathbf{p}_2\mathbf{O}\mathbf{p}_3\right) = \left(\widehat{\mathbf{Op}_2} \cdot \widehat{\mathbf{Op}_3}\right) / \left(|\mathbf{Op}_2|\,|\mathbf{Op}_3|\right) \\
\rho_{13} &= \cos\left(\angle \mathbf{p}_1\mathbf{O}\mathbf{p}_3\right) = \left(\widehat{\mathbf{Op}_1} \cdot \widehat{\mathbf{Op}_3}\right) / \left(|\mathbf{Op}_1|\,|\mathbf{Op}_3|\right) \\
\rho_{12} &= \cos\left(\angle \mathbf{p}_1\mathbf{O}\mathbf{p}_2\right) = \left(\widehat{\mathbf{Op}_1} \cdot \widehat{\mathbf{Op}_2}\right) / \left(|\mathbf{Op}_1|\,|\mathbf{Op}_2|\right)
\end{aligned}
$$

and $\widehat{\mathbf{Op}_1}$, $\widehat{\mathbf{Op}_2}$, $\widehat{\mathbf{Op}_3}$ are unit vectors along directions $\mathbf{Op}_1$, $\mathbf{Op}_2$ and $\mathbf{Op}_3$ respectively. Expressing $|\mathbf{Op}_2|$ and $|\mathbf{Op}_3|$ in terms of $|\mathbf{Op}_1|$

$$
|\mathbf{Op}_2| = \omega_1\,|\mathbf{Op}_1| \tag{A.1a}
$$

$$
|\mathbf{Op}_3| = \omega_2\,|\mathbf{Op}_1| \tag{A.1b}
$$

we have

$$l_{12}^2 = \omega_1^2 |\mathbf{Op}_1|^2 + \omega_2^2 |\mathbf{Op}_1|^2 - 2\omega_1\omega_2 |\mathbf{Op}_1|^2 \rho_{23} \tag{A.2a}$$

$$l_{13}^2 = |\mathbf{Op}_1|^2 + \omega_2^2 |\mathbf{Op}_1|^2 - 2\omega_2 |\mathbf{Op}_1|^2 \rho_{13} \tag{A.2b}$$

$$l_{23}^2 = |\mathbf{Op}_1|^2 + \omega_1^2 |\mathbf{Op}_1|^2 - 2\omega_1 |\mathbf{Op}_1|^2 \rho_{12} \tag{A.2c}$$

From Eqs. (A.2a) and (A.2b),

$$l_{12}^2 \left[1 + \omega_2^2 - 2\omega_2\rho_{13}\right] = l_{13}^2 \left[\omega_1^2 + \omega_2^2 - 2\omega_1\omega_2\rho_{23}\right] \tag{A.3}$$

From Eqs. (A.2a) and (A.2c),

$$l_{12}^2 \left[1 + \omega_1^2 - 2\omega_1\rho_{12}\right] = l_{23}^2 \left[\omega_1^2 + \omega_2^2 - 2\omega_1\omega_2\rho_{23}\right] \tag{A.4}$$

Let $\left(\frac{l_{12}}{l_{13}}\right)^2 = \omega_3$ and $\left(\frac{l_{12}}{l_{23}}\right)^2 = \omega_4$, from Eqs. (A.3) and (A.4),

$$\begin{aligned} \omega_2^2 (1 - \omega_3) + 2\omega_2 \left[\omega_3\rho_{13} - \omega_1\rho_{23}\right] + \omega_1^2 - \omega_3 = 0 \\ \omega_2^2 - 2\omega_1\omega_2\rho_{23} + \omega_1^2 (1 - \omega_4) + 2\omega_1\omega_4\rho_{12} - \omega_4 = 0 \end{aligned} \tag{A.5}$$

Eq. (A.5) has the form:

$$m\omega_2^2 + p\omega_2 + q = 0 \tag{A.6a}$$

$$m'\omega_2^2 + p'\omega_2 + q' = 0 \tag{A.6b}$$

Multiplying Eqs. (A.6a) and (A.6b) by m' and m respectively and subtracting,

$$(pm' - p'm)\,\omega_2 + (m'q - mq') = 0 \tag{A.7}$$

Multiplying Eqs. (A.6a) and (A.6b) by q' and q respectively, subtracting and dividing by ω_2,

$$\begin{aligned} (m'q - mq')\,\omega_2^2 + (p'q - pq')\,\omega_2 = 0 \\ (m'q - mq')\,\omega_2 + (p'q - pq') = 0 \end{aligned} \tag{A.8}$$

Multiplying Eqs. (A.7) by $(m'q - mq')$ and (A.8) by $(pm' - p'm)$, and subtract to obtain

$$(m'q - mq')^2 - (pm' - p'm)(p'q - pq') = 0 \tag{A.9}$$

Expanding Eq. (A.9) and grouping terms we obtain a quartic polynomial in ω_1:

$$G_4\omega_1^4 + G_3\omega_1^3 + G_2\omega_1^2 + G_1\omega_1 + G_0 = 0 \tag{A.10}$$

where

$$\begin{aligned}
G_4 &= (\omega_3\omega_4 - \omega_3 - \omega_4)^2 - 4\omega_3\omega_4\rho_{23}^2 \\
G_3 &= 4\omega_4(1 - \omega_3)(\omega_3\omega_4 - \omega_3 - \omega_4)\rho_{12} \\
&\quad + 4\omega_3\rho_{23}\left[(\omega_3\omega_4 - \omega_3 + \omega_4)\rho_{13} + 2\omega_4\rho_{12}\rho_{23}\right] \\
G_2 &= \left[2\omega_4(1 - \omega_3)\rho_{12}\right]^2 + 2(\omega_3\omega_4 + \omega_3 - \omega_4)(\omega_3\omega_4 - \omega_3 - \omega_4) \\
&\quad + 4\omega_3\left[(\omega_3 - \omega_4)\rho_{23}^2 + (1 - \omega_4)\omega_3\rho_{13}^2 - 2\omega_4(1 + \omega_3)\rho_{12}\rho_{13}\rho_{23}\right] \\
G_1 &= 4\omega_4\rho_{12}(1 - \omega_3)(\omega_3\omega_4 + \omega_3 - \omega_4) \\
&\quad + 4\omega_3\left[(\omega_3\omega_4 - \omega_3 + \omega_4)\rho_{13}\rho_{23} + 2\omega_3\omega_4\rho_{12}\rho_{13}^2\right] \\
G_0 &= (\omega_3\omega_4 + \omega_3 - \omega_4)^2 - 4\omega_3^2\omega_4\rho_{13}^2
\end{aligned}$$

For each positive real root of Eq. (A.10), we determine a single positive real value for each of the sides $|\mathbf{Op}_1|$ and $|\mathbf{Op}_2|$. From Eq. (A.2c) we have

$$|\mathbf{Op}_1| = l_{23}\left(\omega_1^2 - 2\omega_1\rho_{12} + 1\right)^{-1/2}$$

and from Eq. (A.1a), we can find $|\mathbf{Op}_2|$ as

$$|\mathbf{Op}_2| = \omega_1 |\mathbf{Op}_1|$$

If $m'q \neq mq'$, then from Eq. (A.8) we have

$$\omega_2 = \frac{p'q - pq'}{mq' - m'q} \tag{A.11}$$

If $m'q = mq'$, then Eq. (A.11) is undefined and we obtain two values of ω_2 from Eq. (A.2b):

$$\omega_2 = \rho_{13} \pm \left[\rho_{13}^2 + \left(l_{13}^2 - |\mathbf{Op}_1|^2\right) / |\mathbf{Op}_1|^2\right]^{1/2}$$

For each real positive value of ω_2, we obtain a value of $|\mathbf{Op}_1|$ from Eq. (A.1b):

$$|\mathbf{Op}_3| = \omega_2 |\mathbf{Op}_1|$$

With this formulation of $|\mathbf{Op}_1|$, $|\mathbf{Op}_2|$ and $|\mathbf{Op}_3|$, z_1, z_2 and z_3 are given by:

$$z_1 = \left[|\mathbf{Op}_1|^2 f^2 / \left(u_1^2 + v_1^2 + f^2\right)\right]^{1/2}$$
$$z_2 = \left[|\mathbf{Op}_2|^2 f^2 / \left(u_2^2 + v_2^2 + f^2\right)\right]^{1/2}$$
$$z_3 = \left[|\mathbf{Op}_3|^2 f^2 / \left(u_3^2 + v_3^2 + f^2\right)\right]^{1/2}$$

Appendix B

Derivation of z_1 and z_3 in terms of z_2

From Eq. (3.15), Eq. (3.18) and Eq. (3.19),

$$l_{12}^2 f^2 = \lambda_1 z_1^2 - 2\lambda_4 z_1 z_2 + \lambda_2 z_2^2 \tag{B.1a}$$

$$l_{13}^2 f^2 = \lambda_1 z_1^2 - 2\lambda_5 z_1 z_3 + \lambda_3 z_3^2 \tag{B.1b}$$

$$l_{23}^2 f^2 = \lambda_2 z_2^2 - 2\lambda_6 z_2 z_3 + \lambda_3 z_3^2 \tag{B.1c}$$

Without loss of generality, rearranging (B.1a) gives

$$\lambda_1 z_1^2 - 2\lambda_4 z_1 z_2 + \lambda_2 z_2^2 - l_{12}^2 f^2 = 0$$

Dividing both sides by λ_1, and completing the squares for z_1 and z_2 gives

$$\left(z_1 - \frac{\lambda_4}{\lambda_1} z_2\right)^2 + \left[\frac{\lambda_2}{\lambda_1} - \left(\frac{\lambda_4}{\lambda_1}\right)^2\right] z_2^2 - \frac{l_{12}^2 f^2}{\lambda_1} = 0$$

or

$$\left(z_1 - \frac{\lambda_4}{\lambda_1} z_2\right)^2 = \frac{l_{12}^2 f^2}{\lambda_1} - \left[\frac{\lambda_2}{\lambda_1} - \left(\frac{\lambda_4}{\lambda_1}\right)^2\right] z_2^2$$

so that

$$\begin{aligned} z_1 &= \frac{1}{\lambda_1}\left[\pm\sqrt{l_{12}^2 f^2 \lambda_1 - \left(\lambda_1\lambda_2 - \lambda_4^2\right) z_2^2} + \lambda_4 z_2\right] \\ &= \frac{1}{\lambda_1}\left[\pm\sqrt{\Delta_1} + \lambda_4 z_2\right] \end{aligned}$$

by Eq. (3.21a). Similarly, z_3 can be expressed in terms of z_2 from Eq. (B.1c) and Eq. (3.21b) as

$$\begin{aligned} z_3 &= \frac{1}{\lambda_3}\left[\pm\sqrt{l_{23}^2 f^2 \lambda_3 - \left(\lambda_2\lambda_3 - \lambda_6^2\right) z_2^2} + \lambda_6 z_2\right] \\ &= \frac{1}{\lambda_3}\left[\pm\sqrt{\Delta_3} + \lambda_6 z_2\right] \end{aligned}$$

Appendix C

Derivation of e_1 and e_2

Rearranging Eq. (3.22a) gives the following error function:

$$e_1(z_1, z_3) = \left(\lambda_1 z_1^2 - 2\lambda_5 z_1 z_3 + \lambda_3 z_3^2 - l_{13}^2 f^2\right)^2$$

On the other hand, the object centroid (x_c, y_c, z_c) as calculated by the three given object points is

$$(x_c, y_c, z_c) = \left(\frac{x_1 + x_2 + x_3}{3}, \frac{y_1 + y_2 + y_3}{3}, \frac{z_1 + z_2 + z_3}{3}\right)$$

which after full perspective projection gives

$$\begin{aligned}(u_{c,1}, v_{c,1}) &= \left(f\frac{x_1 + x_2 + x_3}{z_1 + z_2 + z_3}, f\frac{y_1 + y_2 + y_3}{z_1 + z_2 + z_3}\right) \\ &= \left(\frac{u_1 z_1 + u_2 z_2 + u_3 z_3}{z_1 + z_2 + z_3}, \frac{v_1 z_1 + v_2 z_2 + v_3 z_3}{z_1 + z_2 + z_3}\right) \\ &= \left(\frac{\sum u_i z_i}{\beta_3}, \frac{\sum v_i z_i}{\beta_3}\right)\end{aligned}$$

However, the same image plane should have also been calculated via the three given image points as

$$(u_{c,2}, v_{c,2}) = \left(\frac{\beta_1}{3}, \frac{\beta_2}{3}\right)$$

Thus e_2 is given by

$$e_2 = \max\left(\left(\frac{\sum u_i z_i}{\beta_3} - \frac{\beta_1}{3}\right)^2, \left(\frac{\sum v_i z_i}{\beta_3} - \frac{\beta_2}{3}\right)^2\right)$$

from Eqs. (3.23a), (3.23b) and (3.23c).

Appendix D

Derivation of the Update Rule for Gauss-Newton Method

Rearranging Eq. (3.22a), we get another form for $e_1(z_1, z_3)$ in terms of z_2

$$e_1(z_2) = \lambda_1 z_1^2 - 2\lambda_5 z_1 z_3 + \lambda_3 z_3^2 - l_{13}^2 f^2$$

$$\begin{aligned}\frac{\partial e_1}{\partial z_2} &= 2\lambda_1 z_1 \frac{\partial z_1}{\partial z_2} - 2\lambda_5 z_3 \frac{\partial z_1}{\partial z_2} - 2\lambda_5 z_1 \frac{\partial z_3}{\partial z_2} + 2\lambda_3 z_3 \frac{\partial z_3}{\partial z_2} \\ &= 2\left(\lambda_1 z_1 - \lambda_5 z_3\right)\frac{\partial z_1}{\partial z_2} - 2\left(\lambda_5 z_1 - \lambda_3 z_3\right)\frac{\partial z_3}{\partial z_2}\end{aligned}$$

$$\begin{aligned}z_1 &= \frac{1}{\lambda_1}\left[\pm\sqrt{l_{12}^2 f^2 \lambda_1 - \left(\lambda_1\lambda_2 - \lambda_4^2\right) z_2^2} + \lambda_4 z_2\right] \\ \frac{\partial z_1}{\partial z_2} &= \frac{1}{\lambda_1}\left[\pm\left[l_{12}^2 f^2 \lambda_1 - \left(\lambda_1\lambda_2 - \lambda_4^2\right) z_2^2\right]^{-\frac{1}{2}}\left[-\left(\lambda_1\lambda_2 - \lambda_4^2\right) z_2\right] + \lambda_4\right] \\ &= \frac{1}{\lambda_1}\left[\mp\frac{1}{\sqrt{\Delta_1}}\left(\lambda_1\lambda_2 - \lambda_4^2\right) z_2 + \lambda_4\right]\end{aligned}$$

$$\begin{aligned}z_3 &= \frac{1}{\lambda_3}\left[\pm\sqrt{l_{23}^2 f^2 \lambda_3 - \left(\lambda_2\lambda_3 - \lambda_6^2\right) z_2^2} + \lambda_6 z_2\right] \\ \frac{\partial z_3}{\partial z_2} &= \frac{1}{\lambda_3}\left[\mp\frac{1}{\sqrt{\Delta_3}}\left(\lambda_2\lambda_3 - \lambda_6^2\right) z_2 + \lambda_6\right]\end{aligned}$$

Then by the Gauss-Newton method, the depth estimate z_2 can be updated according to the following formula,

$$z_{2,n+1} = z_{2,n} + \alpha\frac{\partial e_1}{\partial z_{2,n}}$$

where $z_{2,i}$ is the estimation after i iterations, and α is a stabilization factor less than 1.

Appendix E

Proof of $\left(\lambda_1\lambda_2 - \lambda_4^2\right) > 0$

$$
\begin{aligned}
&\left(\lambda_1\lambda_2 - \lambda_4^2\right) \\
&= \left(u_1'^2 + v_1'^2 + f^2\right)\left(u_2'^2 + v_2'^2 + f^2\right) - \left(u_1'u_2' + v_1'v_2' + f^2\right)^2 \\
&= \left(u_1'^2u_2'^2 + u_1'^2v_2'^2 + u_1'^2f^2 + u_2'^2v_1'^2 + v_1'^2v_2'^2 + v_1'^2f^2 + u_2'^2f^2 + v_2'^2f^2 + f^4\right) \\
&- \left(u_1'^2u_2'^2 + v_1'^2v_2'^2 + f^4 + 2u_1'u_2'v_1'v_2' + 2u_1'u_2'f^2 + 2v_1'v_2'f^2\right) \\
&= \left(u_1'^2v_2'^2 - 2u_1'v_2'u_2'v_1' + u_2'^2v_1'^2\right) + \left(u_1'^2f^2 - 2u_1'u_2'f^2 + u_2'^2f^2\right) + \left(v_1'^2f^2 - 2v_1'v_2'f^2 + v_2'^2f^2\right) \\
&= \left(u_1'v_2' - u_2'v_1'\right)^2 + f^2\left(u_1' - u_2'\right)^2 + f^2\left(v_1' - v_2'\right)^2 \\
&> 0
\end{aligned}
$$

The last step in the above proof follows from the fact that the three image points are distinct and that the second and third terms cannot be zero simultaneously. Similarly, $\left(\lambda_2\lambda_3 - \lambda_6^2\right) > 0$.

Appendix F

Derivation of ϕ and h_i

$$\begin{aligned}\phi &= \frac{\theta}{w_{image}}(i-1) + \frac{\theta}{2w_{image}} \\ &= \frac{\theta}{w_{image}}(i-\frac{1}{2})\end{aligned}$$

Given d_i, d_{min} and d_{max} as

$$\begin{aligned}d_i &= r\cos\left(\phi - \frac{\theta}{2}\right) \\ d_{min} &= r\cos\left(\frac{\theta}{2}\right) \\ d_{max} &= r\end{aligned}$$

$$\begin{aligned}h_i &= h_{image} - h_{image} \times \frac{d_i - d_{min}}{d_{max} - d_{min}} \times \left(1 - \frac{d_{min}}{d_{max}}\right) \\ &= h_{image}\left(1 - \frac{d_i - d_{min}}{r}\right) \\ &= h_{image}\left[1 - \cos\left(\phi - \frac{\theta}{2}\right) + \cos\left(\frac{\theta}{2}\right)\right]\end{aligned}$$

where d_{max} is the maximal object depth.

Appendix G

Derivation of $\mathbf{w}_{1j}$ to $\mathbf{w}_{4j}$

Let $m_0 \times n_0$ be the image size of I_k, where m_0 is the number of columns and n_0 is the number of rows, and the first subscript $k = 0$ or 1 is an index to the two source images. The image content in I_1 is assumed to be completely enclosed in a rectangular region inside I_0 of size $m_1 \times n_1$, which upper-left and lower-right corners are denoted by $\mathbf{w}_1$ and $\mathbf{w}_2$ respectively (so that a scaled-down version of I_1 would properly fit into the region enclosed by $\mathbf{w}_1$ and $\mathbf{w}_2$ in I_0, probably with some misalignments).

For each of the possible pair of I_0 and I_1, N pairs of corresponding points $\mathbf{q}_{ki} = (u_{ki}, v_{ki})$ from I_k are selected as input, where $k = 0$ or 1 is an index to the two source images, and $i = 1, ..., N$ in $\mathbf{q}$, u or v is an index to the i-th pair of corresponding points. These points are then used to calculate the linear magnification ratio of I_0 with respect to I_1 along both u-axis and v-axis:

$$u \text{ ratio} = \frac{\max(u_{1\alpha_1} - u_{1\alpha_2})}{\max(u_{0\alpha_3} - u_{0\alpha_4})}$$
$$v \text{ ratio} = \frac{\max(v_{1\alpha_1} - v_{1\alpha_2})}{\max(v_{0\alpha_3} - v_{0\alpha_4})}, \quad \alpha_1 \neq \alpha_2, \alpha_3 \neq \alpha_4$$

$\begin{bmatrix} m_1 & n_1 \end{bmatrix}$ is then given by:

$$\begin{bmatrix} m_1 & n_1 \end{bmatrix} = \begin{bmatrix} u \text{ ratio} \times m_0 & v \text{ ratio} \times n_0 \end{bmatrix}$$

We further define the object centroids $\mathbf{c}_k$ and the image plane centroids $\mathbf{c}_{I_k}$ as

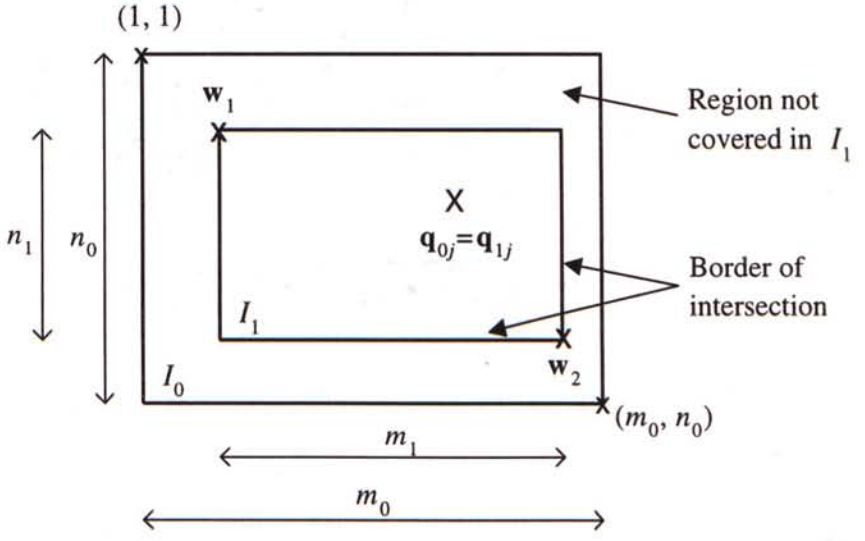

Figure G.1: Notations and conventions.

$$\mathbf{c}_k = \frac{1}{N}\sum_{j=1}^{N}\mathbf{q}_{kj}$$

$$\mathbf{c}_{I_k} = \frac{1}{2}\begin{bmatrix} m_k & n_k \end{bmatrix}$$

giving the offset of object centroids $\boldsymbol{\eta}_{\mathbf{c}_0\mathbf{c}_1}$,

$$\boldsymbol{\eta}_{\mathbf{c}_0\mathbf{c}_1} = \mathbf{c}_0 - \mathbf{c}_1$$

so that $\mathbf{w}_1$ and $\mathbf{w}_2$ are given by simple mathematics:

$$\mathbf{w}_1 = \frac{1}{2}\begin{bmatrix} m_0 - m_1 & n_0 - n_1 \end{bmatrix} + \boldsymbol{\eta}_{\mathbf{c}_0\mathbf{c}_1}$$

$$\mathbf{w}_2 = \mathbf{w}_1 + \begin{bmatrix} m_1 & n_1 \end{bmatrix}$$

After the above calculations, we can identify the region in I_0 which do not appear in I_1. Suppose we are going to generate a frame sequence of M frames (including I_0 as the zeroth frame and as I_1 the $(M-1)$th frame) to zoom-in gradually from I_0 to I_1, we would divide those uncovered regions in I_0 into $M-1$ rings. For example, to compose the jth frame I_{2j} (note that we have introduced a second subscript $j = 0, ..., M-1$ as the frame index, in addition to the first subscript $k = 0, 1, 2$ as image index), we

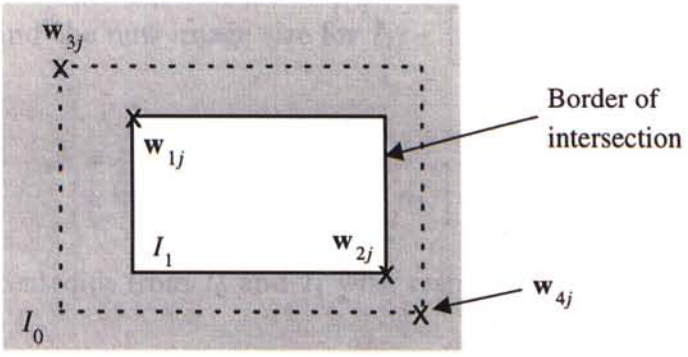

Figure G.2: Boundary coordinates of the $M-j-1$ rings in I_0.

first calculate the boundary coordinates of the inner most $M-j-1$ rings in I_0 (Fig. 5.10).

In the next step, all we need to do is to magnify the region enclosed by $\mathbf{w}_{3j}$ and $\mathbf{w}_{4j}$ in I_0 to the original image size $m_0 \times n_0$ to give I_{0j}, and scale down I_1 from size $m_0 \times n_0$ to $m_{1j} \times n_{1j}$ to give I_{1j}. By overlapping I_{0j} and I_{1j} (with additional feathering and other stitching tools to smooth out image difference within region bounded by $\mathbf{w}_{1j}$ and $\mathbf{w}_{2j}$), we have our jth resulting frame I_{2j} in our movie sequence. In our current implementation, all image resizing are done by bilinear interpolation. However, it should be note that, to get more accurate estimates, the 3D counterparts of $\mathbf{q}_{ki}$ should at best lie on a plane parallel to the focal plane. In an ideal case when the approximations are accurate, the centroids of I_0 and I_1 should always overlap at a point. However, owing to input error and rounding error in practice, the calculated centroid positions are only very close to each other.

Mathematically $\mathbf{w}_{3j}$ and $\mathbf{w}_{4j}$ are given by:

$$\mathbf{w}_{3j} = j \times (\boldsymbol{\delta}_{\mathbf{c}_{I_0}\mathbf{c}_{I_1}} - \boldsymbol{\delta}_{\mathbf{c}_{I_0}\mathbf{c}_0})$$
$$\mathbf{w}_{4j} = \begin{bmatrix} m_0 & n_0 \end{bmatrix} - j \times (\boldsymbol{\delta}_{\mathbf{c}_{I_0}\mathbf{c}_{I_1}} + \boldsymbol{\delta}_{\mathbf{c}_{I_0}\mathbf{c}_0})$$

where $\boldsymbol{\delta}_{\mathbf{c}_{I_0}\mathbf{c}_{I_1}}$ is the incremental offset of image plane centroids, and $\boldsymbol{\delta}_{\mathbf{c}_{I_0}\mathbf{c}_0}$ is the incremental offset of image plane centroid with object centroid:

$$\boldsymbol{\delta}_{\mathbf{c}_{I_0}\mathbf{c}_{I_1}} = \frac{\mathbf{c}_{I_0} - \mathbf{c}_{I_1}}{M-1}$$
$$\boldsymbol{\delta}_{\mathbf{c}_{I_0}\mathbf{c}_0} = \frac{\mathbf{c}_{I_0} - \mathbf{c}_0}{M-1}$$

We further find the new image size for $I_{1j} - \left[\begin{array}{cc} m_{1j} & n_{1j} \end{array}\right]$:

$$\left[\begin{array}{cc} m_{1j} & n_{1j} \end{array}\right] = \left[\begin{array}{cc} m_1 & n_1 \end{array}\right] \cdot \frac{\left[\begin{array}{cc} m_0 & n_0 \end{array}\right]}{\mathbf{w}_{4j} - \mathbf{w}_{3j}}$$

and the object centroids from I_0 and I_1 with respect to the resulting image I_{2j} are

$$\mathbf{c}_{0j} = (\mathbf{c}_0 - \mathbf{w}_{3j}) \cdot \frac{\left[\begin{array}{cc} m_0 & n_0 \end{array}\right]}{\mathbf{w}_{4j} - \mathbf{w}_{3j}}$$
$$\mathbf{c}_{1j} = \mathbf{c}_1 \cdot \left[\begin{array}{cc} \frac{m_{1j}}{m_0} & \frac{n_{1j}}{n_0} \end{array}\right]$$

and eventually

$$\mathbf{w}_{1j} = \mathbf{c}_{0j} - \mathbf{c}_{1j}$$
$$\mathbf{w}_{2j} = \mathbf{w}_1 + \left[\begin{array}{cc} m_{1j} & n_{1j} \end{array}\right]$$

Appendix H

More Experimental Results on Panoramic Stitching Algorithms

Figure H.1: Sample panorama #1 by 8-parameter planar projective transformation algorithm.

Figure H.2: Sample panorama #2 by 8-parameter planar projective transformation algorithm.

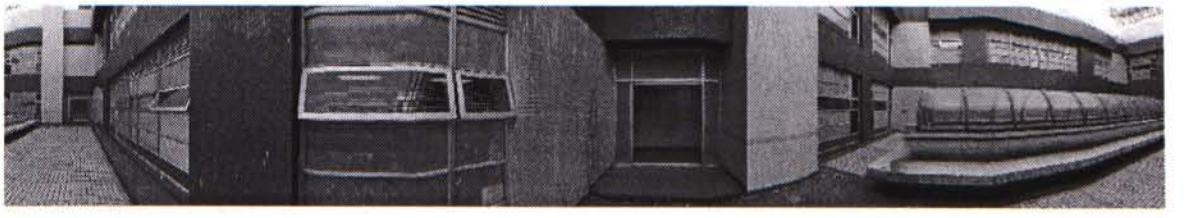

Figure H.3: Sample panorama #3 by 8-parameter planar projective transformation algorithm.

Figure H.4: Sample panorama #1 by 3D rotational alignment algorithm.

Figure H.5: Sample panorama #2 by 3D rotational alignment algorithm.

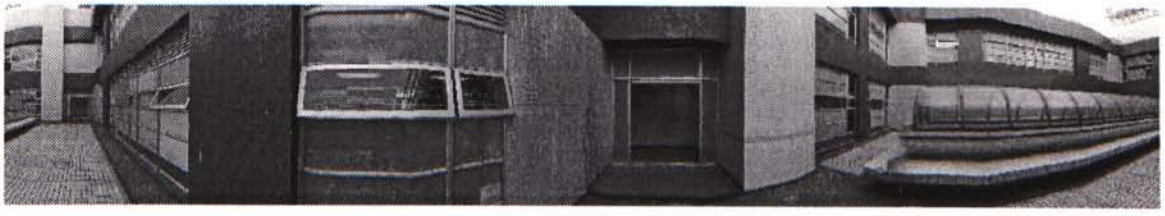

Figure H.6: Sample panorama #3 by 3D rotational alignment algorithm.

Figure H.7: Sample panorama #1 by our stitching algorithm.

Figure H.8: Sample panorama #2 by our stitching algorithm.

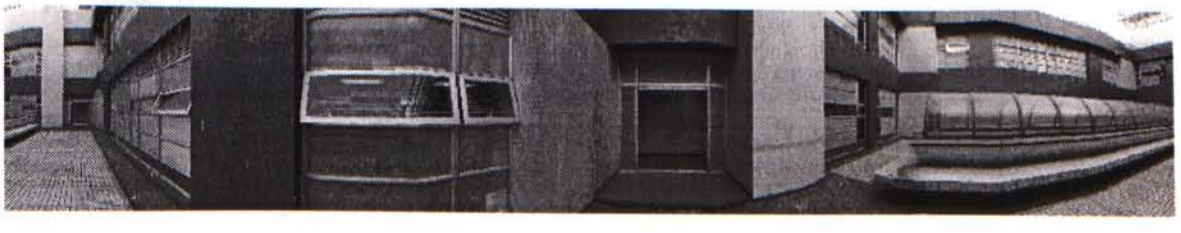

Figure H.9: Sample panorama #3 by our stitching algorithm.

Bibliography

[1] Adelson E.H., *Layered Representations for Image Coding*, Technical Report 181, MIT Media Lab. Vision and Modeling Group, Dec. 1991.

[2] Alter T.D., *3D Pose from Three Points Using Weak-perspective*, IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), vol. 16, no. 8, pp. 802-808, Aug. 1994.

[3] Anandan P., *A Computational Framework and an Algorithm for the Measurement of Visual Motion*, Intl. Journal of Computer Vision, vol. 2, no. 3, pp. 283-310, Jan. 1989.

[4] Apple Computer Inc., QuickTime, Version 1.5, 1992.

[5] Arun K.S., Huang T.S. and Blostein S.D., *Least Square Fitting of Two 3D Point Sets*, IEEE Trans. PAMI, vol. 9, no. 5, pp. 698-700, Sept. 1987.

[6] Bergen J.R., Anandan P., Hanna K.J. *et al.*, *Hierarchical Model-based Motion Estimation*, Proc. of European Conference on Computer Vision (ECCV), pp. 237-252, May 1992.

[7] Black M. and Anandan P., *A Framework for the Robust Estimation of Optical Flow*, Proc. of Intl. Conf. on Computer Vision (ICCV), pp. 231-236, 1993.

[8] Brooks R.A., *Model-based Computer Vision*, UMI Research Press, 1990.

[9] Araujo H., Carceroni R.L. and Brown C.M., *A Fully Projective Formulation for Lowe's Tracking Algorithm*, Technical Report 641, University of Rochester Computer Science Dept., Nov. 1996.

[10] Chen S.E. and Williams L., *View Interpolation for Image Synthesis*, Proc. SIGGRAPH-93, pp. 279-288, Aug. 1993.

[11] Chen S.E., *QuickTime VR - An Image-based Approach to Virtual Environment Navigation*, Proc. SIGGRAPH-95, pp. 29-38, Aug. 1995.

[12] Chiang M.C. and Boult T.E., *Efficient Image Warping and Super-resolution*, Proc. of IEEE Workshop on Applications of Computer Vision, pp. 56-61, Dec. 1996.

[13] Christy S. and Horaud R., *Euclidean Reconstruction: from Paraspective to Perspective*, Proc. ECCV-96, vol. 2, pp. 129-140, 1996.

[14] Craig J.J., *Introduction to Robotics: Mechanics and Control*, Addison-Wesley, 1989.

[15] Darsa L, Costa B. and Varshney A., *Navigating Static Environment Using Image-space Simplification and Morphing*, Proc. of Symposium on 3D interactive Graphics, pp. 25-34, Apr. 1997.

[16] DeMenthon D.F. and Davis L.S., *Exact and Approximate Solutions of the Perspective-Three-Point Problem*, IEEE Trans. PAMI, vol. 14, no. 11, pp. 1100-1105, Nov. 1992.

[17] DeMenthon D.F. and Davis L.S., *Model-based Object Pose in 25 Lines of Code*, Intl. Journal of Computer Vision, vol. 15, pp. 123-141, 1995.

[18] Dhome M., Richetin M., Lapreste J-T., and Rives G., *3D Pose from 3 Points Using Weak-perspective*, IEEE Trans. PAMI, vol. 11, no. 12, pp. 1265-1278, Dec. 1989.

[19] Fischler M.A. and Bolles R.C., *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*, Communications of the ACM, vol. 24, no. 6, pp. 381-395, June 1981.

[20] Fu C.W. and Heng P.A., *A Perceptually Acceptable Walk-through Algorithm in Multi-node Panoramic Environment*, Internal Report, CSE CUHK, May 1998.

[21] Fung Y.F. and Wong K.H., *A Three-point Model-based Algorithm for Pose Estimation*, Proc. of Image, Speech, Signal Processing and Robotics, vol. 1, pp. 123-128, Sept. 1998.

[22] Gee A. and Cipolla R., *Determining the Gaze of Faces in Images*, Image and Vision Computing, vol. 12, no. 10, pp. 639-647, Dec. 1994.

[23] Gee A. and Cipolla R., *Fast Visual Tracking by Temporal Consensus*, Image and Vision Computing, vol. 14, pp. 105-114, 1996.

[24] Gennery D.B., *Visual Tracking of Known Three-dimensional Objects*, Intl. Journal of Computer Vision, vol. 7, no. 3, pp. 243-270, 1992.

[25] Hanna K.J. and Okamoto N.E., *Combining Stereo and Motion Analysis for Direct Estimation of Scene Structure*, Proc. ICCV-93, pp. 357-365, 1993.

[26] Haralick R.M., Lee C.N., Ottenberg K and Nolle M., *Review and Analysis of Solutions of the Three Point Perspective Pose Estimation Problem*, Intl. Journal of Computer Vision, vol. 13, no. 3, pp. 331-356, 1994.

[27] Horaud R., Conio B., Leboulleux O. and Lacolle B., *An Analytic Solution for the Perspective-4-Point Problem*, Computer Vision, Graphics, and Image Processing, vol. 47, pp. 33-44, 1989.

[28] Horaud R., Christy S., Dornaika F. and Lamiroy B., *Object Pose: Links between Para-perspective and perspective*, Proc. ICCV, pp. 426-433, 1995.

[29] Hirose M., *Image-based Virtual World Generation*, IEEE Multimedia, vol. 4, no. 1, pp. 27-33, 1997.

[30] Hirose M., Watanabe S. *et al.*, *Generation of Wide-range Virtual Spaces Using Photographic Images*, Proc. VRAIS-98, pp. 234-241, Mar. 1998.

[31] Horn B.K.P., *Robot Vision*, MIT/McGraw-Hill, New York, 1986.

[32] Horn B.K.P., *Closed-form Solution of Absolute Orientation Using Unit Quaternions*, J. Opt. Soc. Am. A, vol. 4, no. 4, pp629-642, Apr. 1987.

[33] Horn B.K.P. and Weldon E.J. Jr., *Direct Methods for Recovering Motion*, Intl. Journal of Computer Vision, no. 2, pp. 51-76, 1988.

[34] Huang T.S. and Netravali A.N., *Motion and Structure from Feature Correspondences: A Review*, Proc. of IEEE, vol. 82, no. 2, pp.252-268, Feb. 1994.

[35] Huttenlocher D.P. and Ullman S., *Object Recognition by Alignment*, Proc. ICCV-87, pp. 102-111, 1987.

[36] Huttenlocher D.P. and Ullman S., *Recognizing Solid Objects by Alignment*, Proc. of Image Understanding Workshop, vol. 2, pp. 1114-1122, Apr. 1988.

[37] Irani M., Rousso B. and Peleg S., *Detecting and Tracking Multiple Moving Objects Using Temporal Integration*, Proc. ECCV-92, pp. 282-287, May 1992.

[38] Irani M., Rousso B. and Peleg S., *Computing Occluding and Transparent Motions*, Intl. Journal of Computer Vision, no. 12, pp. 5-16, Feb. 1994.

[39] Irani M., Hsu S. and Anandan P., *Video Compression Using Mosaic Representations*, Signal Processing: Image Communication, no. 7, pp. 529-552, 1995.

[40] Kang S.B. and Weiss R., *Characterization of Errors in Compositing Panoramic Images*, Technical Report 96/2, Digital Equipment Corp., Cambridge Research Lab., June 1996.

[41] Kumar R., Anandan P. and Hanna K., *Shape Recovery from Multiple Views: A Parallax-based Approach*, Proc. of Image Understanding Workshop, pp. 947-955, Nov. 94.

[42] Kumar R., Anandan P., Irani M. *et al.*, *Representation of Scenes from Collections of Images*, Proc. of IEEE Workshop on Representations of Visual Scenes, pp. 10-17, June 1995.

[43] Laveau S. and Faugeras O., *3-D Scene Representation as a Collection of Images and Fundamental Matrices*, INRIA Technical Report No. 2205, Feb. 1994.

[44] Linnainmaa S., Harwood D. and Davis L.S., *Pose Determination of a Three-dimensional Object Using Triangle Pairs*, IEEE Trans. PAMI, vol. 10, no. 5, pp. 634-647, Sep. 1988.

[45] Lippman A., *Movie Maps: An Application of the Optical VideoDisc to Computer Graphics*, Proc. SIGGRAPH-80, pp. 32-43.

[46] Liu M.L. and Wong K.H., *Computer Vision-based Real-time Pose Estimation Using Four Corresponding Points*, Proc. of Workshop on 3D Computer Vision, pp.44-48, May 1997.

[47] Liu M.L. and Wong K.H., *A Direct Six-point Model-based Pose Estimation Algorithm*, Internal Report, CSE CUHK, 1998.

[48] Long Q., Deriche R., Faugeras O. and Papadopoulo T., *On Determining the Fundamental Matrix: Analysis of Different Methods and Experimental Results*, INRIA technical report RR-1894, 1993.

[49] Longuet-Higgins H. and Prazdny K., *The Interpretation of a Moving Retina Image*, Proc. of Roy. Soc. London B, pp. 385-397, 1980.

[50] Lowe D.G., *Solving for the Parameters of Object Models from Image Descriptions*, Proc. of ARPA Image Understanding Workshop, pp. 121-127, Apr. 1980.

[51] Lowe D.G., *Three-dimensional Object Recognition from Single Two-dimensional Images*, Artificial Intelligence, vol. 31, no. 3, pp. 355-395, Mar. 1987.

[52] Lowe D.G., *Fitting Parameterized Three-dimensional Models to Images*, IEEE Trans. PAMI, vol. 13, no. 5, pp. 441-450, May 1991.

[53] Lowe D.G., *Robust Model-based Motion Tracking through the Integration of Search and Estimation*, Intl. Journal of Computer Vision, vol. 8, pp. 113-122, 1992.

[54] Lucas B.D. and Kanade T., *An Iterative Image Registration Technique with an Application in Stereo Vision*, Proc. 7th IJCAI-81, pp. 674-679, 1981.

[55] Mann S. and Picard R.W., *Virtual Bellows: Constructing High-quality Images from Video*, Proc. of IEEE Intl. Conf. on Image Processing, vol. 1, pp. 363-367, Nov. 1994.

[56] Marr D., *Vision*, W.H. Freeman Co., San Francisco, CA, 1982.

[57] McMillan L., Bishop G., *Plenoptic Modelling: An Image-based Rendering System*, Proc. SIGGRAPH-95, pp. 39-46, 1995.

[58] McReynolds D.P and Lowe D.G., *Rigidity Checking of 3D Point Correspondences Under Perspective Projection*, IEEE Trans. PAMI, vol. 18, no. 12, pp. 1174-1185, Dec. 1996.

[59] Miller G. *et al.*, *The Virtual Museum: Interactive 3D Navigation of a Multimedia Database*, Journal of Visualization and Computer Animation, no. 3, 183-197, 1992.

[60] Moravec H.P., *Robot Rover Visual Navigation*, pp. 13-14, UMI Research Press, 1981.

[61] Navab N. and Faugeras O., *Monocular Pose Determination from Lines: Critical Sets and Maximum Number of Solutions*, Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 254-260, June 1993.

[62] Oberkampf D., DeMethon D.F. and Davis L.S., *Iterative Pose Estimation Using Coplanar Feature Points*, CVGIP, Proc. of Image Understanding Workshop, vol. 63, no. 3, pp. 495-511, May 1996.

[63] Oliensis J., *A Critique of Structure from Motion Algorithms*, NEC Technical Report, Oct. 1997.

[64] Or S.H.,Wong K.H. and King I., *A Model-based Real Time Pose Tracking System*, Internal Report, CSE CUHK, Oct. 1996.

[65] Poelman C. and Kanade T., *A Paraperspective Factorization Method for Shape and Motion Recovery*, Technical Report CMU-CS-92-208, 1995.

[66] Polhemus. *Polhemus 3Space User's Manual*, Polhemus Inc., 1993.

[67] Press W.H., Flannery B.P., Teukolsky S.A., and Vetterling W.T., *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, 2nd Ed., 1992.

[68] Ripley D.G., *DVI - A Digital Multimedia Technology*, Communications of the ACM, 32(7), pp. 811-822, 1989.

[69] Sawhney H.S., *Simplifying Motion and Structure Analysis Using Planar Parallax and Image Warping*, Proc. of Intl. Conf. on Pattern Recognition (ICPR), vol. A, pp. 403-408, Oct. 1994.

[70] Sawhney H.S. and Ayer S., *Compact Representation of Videos through Dominant Multiple Motion Estimation*, IEEE Trans. PAMI, vol. 18, no. 8, pp. 814-830, Aug. 1996.

[71] Seitz S., *Image-based Transformation of Viewpoint and Scene Appearance*, Ph.D. Thesis, 1998.

[72] Shakunaga T. and Kaneko H., *Perspective Angle Transform: Principle of Shape from Angles*, Intl. Journal of Computer Vision, no. 3, pp. 239-254, 1989.

[73] Shi J. and Tomasi C., *Good Features to Track*, Proc. CVPR-94, pp. 593-600, 1994.

[74] Shum H.Y. and Szeliski R., *Panoramic Image Mosaics*, Technical Report MSR-TR-97-23, Microsoft Research, 1997.

[75] Szeliski R., *Image Mosaicing for Tele-reality Applications*, DEC Cambridge Research Lab Technical Report, CRL94/2, May 1994.

[76] Szeliski R. and Kang S.B., *Direct Methods for Visual Scene Reconstruction*, Proc. of IEEE Workshop on Representations of Visual Scenes, pp. 26-33, June 1995.

[77] Szeliski R., *Video Mosaics for Virtual Environments*, IEEE Computer Graphics and Applications, pp. 22-30, Mar 1996.

[78] Szeliski R. and Shum H.Y., *Creating Full View Panoramic Image Mosaics and Texture-mapped Models*, Proc. SIGGRAPH-97, pp. 251-258, Aug. 1997.

[79] Tekalp A., *Digital Video Processing*, Prentice Hall, NJ, 1995.

[80] Tomasi C. and Kanade T., *Shape and Motion from Image Streams under Orthography: A Factorization Method*, Intl. Journal of Computer Vision, vol. 9, pp. 137-154, 1992.

[81] Tsai R.Y., *A Versatile Camera Calibration Technique for High-accuracy 3D Machine Vision Metrology Using Off-the-shelf TV Cameras and Lenses*, IEEE Journal of Robotics and Automation, vol. RA-3, no. 4, Aug. 1987.

[82] Umeyama S., *Least-square Estimation of Transformation Parameters Between Two Point Pattern*, IEEE Trans. PAMI, vol. 13, no. 4, pp. 376-380, Apr. 1991.

[83] Vernon D., *Machine Vision: Automated Visual Inspection and Robot Vision*, Prentice Hall, 1991.

[84] Weng J., Ahuja N. and Huang T.S., *Error Analysis of Motion Parameter Determination from Image Sequences*, Proc. ICCV-88, pp. 703-707, June 1988.

[85] Wiles C. and Brady M., *Ground Plane Motion Camera Models*, Proc. ECCV-96, vol. 2, pp. 238-247, 1996.

[86] Wolfe W.J., Mathis D., Sklair C.W. and Magee M., *The Perspective View of Three Points*, IEEE Trans. on PAMI, vol. 13, no. 1, pp. 66-73, Jan. 1991.

[87] Wood D.N., Finkelstein A., Hughes J.F., Thayer C.E. and Salesin D.H., *Multiperspective Panoramas for Cel Animation*, Proc. SIGGRAPH-97, pp. 243-250, 1997.

[88] Wu Y., Iyengar S.S., Jain R. and Bose S., *A New Generalized Computational Framework for Finding Object Orientation using Perspective Trihedral Angle Constraint*, IEEE Trans. PAMI, vol. 16, no. 10, pp. 961-975, Oct. 1994.

[89] Xu G. and Zhang Z., *Epipolar Geometry in Stereo, Motion and Object Recognition: A Unified Approach*, Kluwer Academic Publishers, 1996.