Iterative Computation of Camera Paths in an Image Based Rendering Application

Ming Yan

A thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

in partial fulfillment of the requirements for the degree of

Master of Applied Science

in Electrical Engineering

May, 2007

Ottawa-Carleton Institute of Electrical and Computer Engineering

School of Information Technology and Engineering

University of Ottawa

Abstract

This thesis presents a novel algorithm to iteratively compute camera paths of long image sequences. Scale Invariant Features are first extracted from the ordered set of images. These images are then matched pair-wise sequentially and correspondences are computed. An initial geometric path is found after by applying a bundle adjustment algorithm on these correspondences. Distances between cameras are computed from this initial estimation. The iteration process starts by grouping nearby cameras and then bundle adjusting the groups, and ends by merging the groups. This process is repeated until the reprojection errors fall into the preset tolerance. The key point in this algorithm is to take advantage of loopbacks in the image sequences. We have obtained excellent results for two camera paths, namely the spiral path and the snake like path. Our algorithm achieves both precise and stable results.

Acknowledgments

Firstly, I would like to express my sincere gratitude to Dr. Robert Laganière for being a great supervisor and mentor during my graduate studies. This thesis is completed under his invaluable guidance, support and patience.

Secondly, I would like to thank my co-supervisor, Dr. Gerhard Roth, who offered help and discussion whenever I was stuck with a research problem. His previous experiences helped me a lot to avoid many unnecessary work. Several major problems were solved after following his suggestions. This work would not have been possible without his help and suggestions.

I would also like to thank the many VIVA lab members who gave me various assistant.

Finally, I am grateful to my parents and my family for their precious support.

Contents

List of Figures

Chapter 1

Introduction

In this chapter, we first describe the problem that will be solved in this thesis. After a brief introduction to bundle adjustment, we analyze both its merits and demerits. Several indirect bundle adjustment approaches are also introduced. The objectives of the thesis is to compute the camera path from a set of images and it is described next. Finally, the organization of this thesis is presented.

1.1 Problem Description

Camera pose estimation has been explored for the past few decades and it still remains an active topic. The object is to find both the camera positions and the camera orientations from a sequence of images taken by a camera. Although this problem seems to be straightforward, there is no direct solution that can solve it in a simple way. The applications of camera pose estimation are in the field of Image Based Rendering (IBR) [1], Robotics [2], Photogrammetry [3] and Virtual Navigation(VI) [4]. Several successful pose estimation methods have been proposed for specific short image sequences, such as in [5, 6, 3, 7]. We see from these works that bundle adjustment produces good results since it provides a true maximum likelihood estimation even when some of the input data is missing [8].

Bundle adjustment is an iterative method that provides an optimized solution to minimize the overall error between the measured 2-D feature points and the projected 2-D feature points. It does this by simultaneously adjusting the camera parameters and the 3-D scene structure as a bundle. It is a general method that can be used to solve many reconstruction and optimization problems. Unfortunately, bundle adjustment has intrinsic drawbacks [9, 8]: i) it requires a good initialization; ii) it is an extremely time consuming process; and iii) it does not always converge. These problems become severe when dealing with long image sequences that contain hundreds of images.

The nonlinear nature of the bundle adjustment process makes the convergence sensitive to the initial estimate. For this reason, bundle adjustment is always used as the last step in a optimization problem to increase the overall precision. Poor initialization will result in an incorrect solution or even divergence. The bundle adjustment process takes a large amount of input data and tries to minimize the overall error by altering all the input data at the same time and then re-computes a solution. This could be very complex and time consuming. In general, bundle adjustment is rarely used directly to solve large scale camera path problems.

Two methods that try to deal these issues are the hierarchical merging of sub-sequences and the incremental bundle adjustment [8]. In the hierarchical scheme, a long sequence is divided into smaller, possibly overlapping, sequences that are then merged together. i.e. the long sequence is subdivided over and over again until the bundle adjustment converges at each sub-sequence, and then the sub-sequences are combined to generate the original sequence. A global bundle adjustment is then applied on the sequence to find the complete reconstruction. However, it is not efficient to run the global bundle adjustment on long image sequences because of the very large number of frames involved. Moreover, the hierarchical methods suffer from the accumulated error built during the merging process. It results an initial reconstruction that has drifted away from the real location and the final bundle adjustment may not converge due to the poor initial reconstruction.

The incremental bundle adjustment starts by processing only a small portion of the original input data since the sub-sampled data are more likely to converge and an initial estimate can be found more easily. When either new frames or new feature points added, the bundle adjustment is executed again to refine the initial reconstruction. Although better results are theoretically expected using the incremental bundle adjustment, it is obvious that the computation costs are significant and this is not appropriate when processing large number of frames.

In this thesis, we propose to fuse these two possible strategies into an integrated framework that will make scalable 3-D reconstruction possible, e.g. where hundreds of viewpoints have to be simultaneously processed.

1.2 Thesis Objectives

The goal of this thesis is to relate the different camera views to each other in order to obtain the camera pose information, and to extract some depth cues concerning the observed scene. This information will be used for both data compression and view synthesis. We will investigate different bundle adjustment formulations and adapt them to the particular configuration of each defined problem. Feature matching is also a key element in this work. Stable features must be detected and reliably matched. Different invariant descriptors must therefore be identified and validated. In particular, invariants based on SIFT (Scale Invariant Feature Transform) features are explored since they appear to constitute a promising tool to achieve both rotational and scale invariance.

In this thesis, we present an iterative algorithm that computes the camera path of long image sequences. It consists in applying successive bundle adjustment phases on different segments of the image sequence. The local camera path models thus obtained are merged together into a common reference frame. The procedure is then repeated on a new grouping of the cameras, until the reconstruction error is below a given error tolerance.

Figure 1.1 depicts the flow chart of our system. A long image sequence is first separated into overlapping segments through segmentation and feature points are detected in each frame. These points are then matched across the images of each segment [7]. We obtain the reconstruction of the segments using bundle adjustment on the matches. Registration of the segments is possible because the segments are overlapping. We then reconstruct the camera path of the entire image sequence after registering all the segments. This is the initial reconstruction and it is unlikely to pass the error tolerance test. Next we partition the initial reconstruction into overlapping groups based on the relative positions among the reconstructed cameras. Matches are extracted from the images in a group and are sent to the bundle adjuster. Then the reconstructed groups will be registered and a better reconstruction can be found. The refined reconstruction will also go through the error tolerance test. We output the reconstructed path if the error limit is reached, otherwise we process the path again.

The main objective of the proposed approach is to ensure the scalability of the reconstruction and the good convergence of the bundle adjustment process by imposing a limit on the number of views for which the structure and motion parameters have to be simultaneously optimized. Indeed, we never run a global bundle adjustment on the full set of images. Error accumulation is also prevented by exploiting the presence of loopbacks in the camera path.

1.3 Thesis Organization

This thesis is organized as follows:

Chapter 1 introduces the problem that this work will be targeting. The objectives of the thesis and the organization of the thesis are also described in this chapter.

We explain in detail the camera models and camera representations in computer vision in Chapter 2. A simplified camera model is described first. A three dimensional (3-D) scene point is projected to a two dimensional (2-D) point on a plane. The relationship between

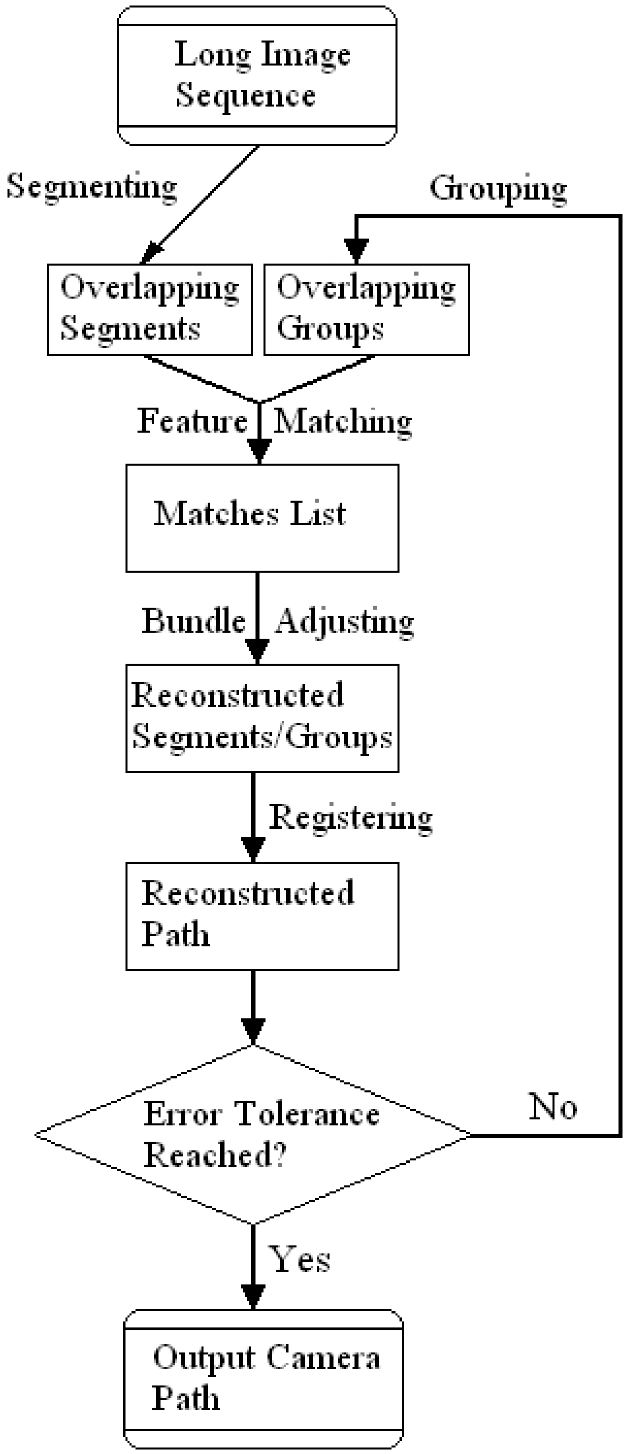

Figure 1.1: Flow Chart.

the 3-D point and the 2-D point can be described by a projection matrix. Then, we clarify several specific matrices used to represent camera poses, and also define the rotations and translations with respect to reference coordinate systems. Correct camera poses can not be computed without precisely defined rotations and translations with a specified coordinate system. In the last section, we describe a way to view 3-D camera poses from normalized camera matrices.

Bundle adjustment is the core throughout this work. Chapter 3 explains how the bundle adjustment works in details, including the algorithm analysis and a possible implementation. We also review some bundle adjustment applications in Chapter 3.

Chapter 4 describes how to reconstruct the camera paths from long image sequences in great detail. The original long image sequence is separated into overlapping segments based on the sequential information. An initial camera path is found after bundle adjusting and registering the segments. We then use a robust grouping algorithm to partition the initial cameras into overlapping groups with desired group size and overlapping scale by assigning the closest neighbors to the same group. We bundle adjust and register the groups to create a better path reconstruction. The grouping-bundle adjusting-registering iteration will go on until an acceptable result is found.

In Chapter 5 we demonstrate how to reconstruct the long camera paths step by step. We focus on the spiral path since it contains simultaneous rotation and translation. Images were taken with an ordinary digital camera and they are processed according to the algorithms described in Chapter 4.

In Chapter 6, we conclude the thesis and describe the contributions. We also suggest some work to be done in the future.

Chapter 2

Cameras in Computer Vision

This chapter introduces the camera models that will be used throughout this thesis. We first introduce a simplified camera model and derive how images are formed. Then, we demonstrate how cameras are to be represented. Several commonly used camera matrices are explained and their relationships are discussed. We also explore the rotations and the translations associated with the cameras. Finally some reference coordinate system that are used in the camera models are defined.

2.1 Camera Model

Cameras in computer vision are modeled as a projection from a 3-D scene to 2-D images. The reflected rays from the real world objects pass through the camera lenses, and are then recorded on a film or on a CCD sensors as images. An image is composed of thousands of pixels. Each pixel reflects the strength of the incoming rays that was reflected by a object. Figure 2.1 is a simplified camera model. In this model, we only consider one image pixel and the corresponding ray along with the 3-D point that generates the ray since other image pixels are generated in the same fashion. Two assumptions are made in this model:

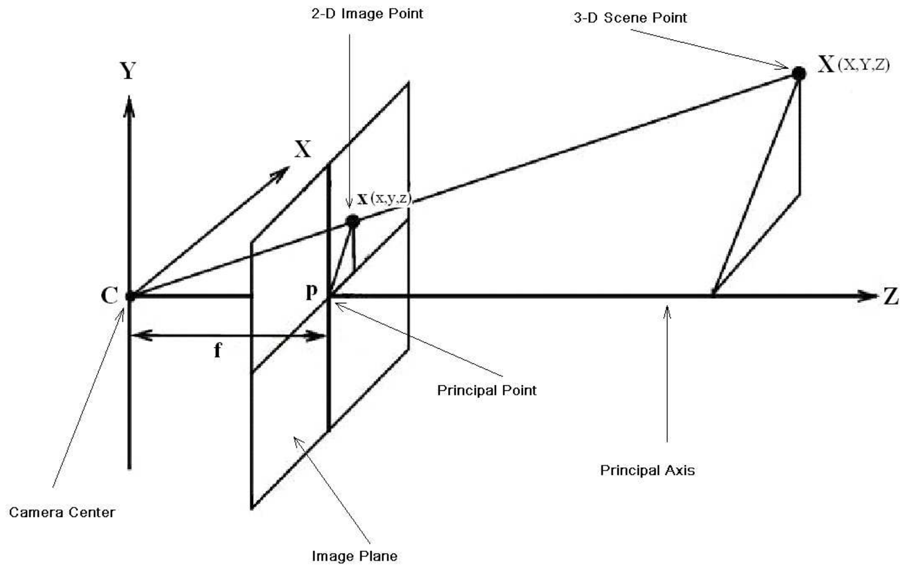

Figure 2.1: Pinhole camera model.

1. Assume that the origin of the coordinates in the image plane is at the principal point.

2. Assume that the world coordinate frame is the same to the camera coordinate frame.

These assumptions will be dropped later on to accommodate general cases.

Consider a scene point $\mathbf{X} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$ in a 3-D space. This particular point will be mapped to a image point $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$ on a 2-D camera plane through projection. If f is the camera focal length, we have the following equations from similar triangles:

$$\frac{f}{Z} = \frac{x}{X} = \frac{y}{Y} \tag{2.1}$$

Rewriting and separating Equation (2.1), we obtain the direct perspective projection

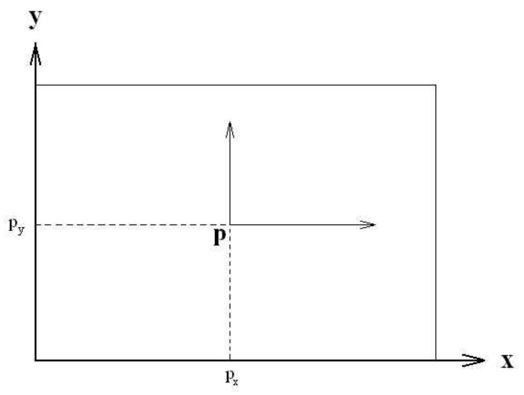

Figure 2.2: Principal point offset.

equations:

$$\begin{cases} x = \frac{fX}{Z} \\ y = \frac{fY}{Z} \end{cases} \tag{2.2}$$

The principal point is not at the origin of the coordinates in the image plane. In fact, there is always an offset between the image plane coordinate origin and the principal point. If the coordinate of the principal point in the image plane is $\mathbf{p} = \begin{bmatrix} p_x \\ p_y \end{bmatrix}$, as shown in Figure 2.2, then Equation (2.2) needs to be amended to include the principal point offset:

$$\begin{cases} x = \frac{fX}{Z} + p_x \\ y = \frac{fY}{Z} + p_y \end{cases} \tag{2.3}$$

The first assumption has been dropped in Equation (2.3) since we have considered the the principal point offset.

The mapping we are seeking is a 3-D to 2-D projection:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \longrightarrow \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.4}$$

Replacing the 2-D point in Equation (2.4) by Equation (2.3), we have:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \longrightarrow \begin{bmatrix} \frac{fX}{Z} + p_x \\ \frac{fY}{Z} + p_y \end{bmatrix} \tag{2.5}$$

Using the homogeneous form of both the 2-D point $\hat{\mathbf{x}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ and the 3-D point $\hat{\mathbf{X}} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$, the right side of Equation (2.5) can be rewritten as:

$$\begin{bmatrix} \frac{fX}{Z} + p_x \\ \frac{fY}{Z} + p_y \\ 1 \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} fX + p_x Z \\ fY + p_y Z \\ Z \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.6}$$

Define $\mathbf{K}$ as the calibration matrix:

$$\mathbf{K} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

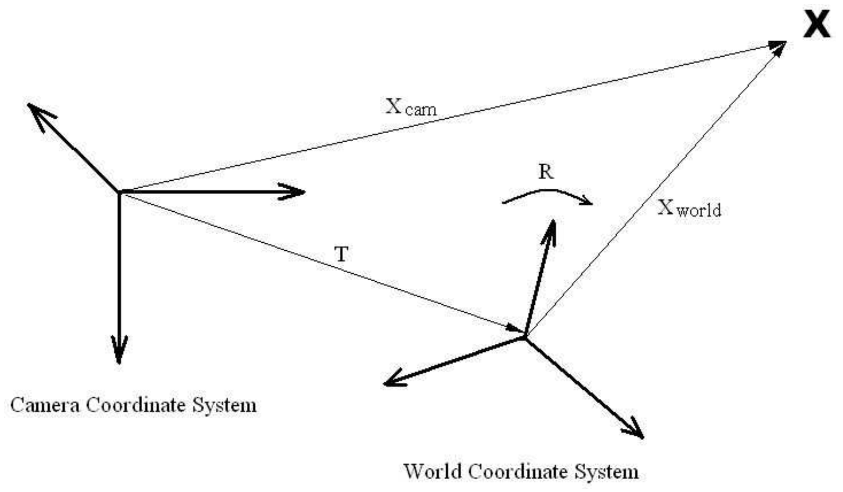

Figure 2.3: Rotation and Translation between world coordinate system and camera coordinate system.

After we combining Equations (2.4),(2.5),(2.6) and (2.7), and rewriting it in a concise form, the 3-D to 2-D mapping can be expressed as:

$$s\ \hat{\mathbf{x}} = \mathbf{K}[\mathbf{I}|\mathbf{0}]\hat{\mathbf{X}} \tag{2.8}$$

where s is a scale factor that counteracts Z.

Since the 3-D scene points are always referred in the world coordinate system instead of in the camera coordinate system, we are now ready to drop the second assumption. In other words, we need to consider the difference between the world coordinate system and the camera coordinate system, as shown in Figure 2.3. A 3-D rotation and a 3-D translation will be applied to one of the coordinate systems so that it is completely overlaps the other coordinate system.

The difference between a world coordinate system and a camera coordinate system is symmetrical. The rotation and translation applied on the first coordinate system for it to be superposed on the second one is the same to the inversed rotation and inversed translation applied on the second coordinate system for it to be superposed on the first one. A reference coordinate system has to be selected first. Although the world coordinate system will be the reference eventually, it is more convenient and straightforward to find the rotation and translation if we choose the camera coordinate system as the reference. Thus the rotation $\mathbf{R}$ is defined as *the rotation of the world coordinate system in the camera coordinate system*, and the translation $\mathbf{T}$ is defined as *the coordinates of the world coordinate system origin in the camera coordinate system*. We then can convert the 3-D point coordinates from the world coordinate system to the camera coordinate system:

$$\mathbf{X_{cam}} = \mathbf{R}\mathbf{X_{world}} + \mathbf{T} \tag{2.9}$$

Using the homogeneous forms of $\mathbf{X_{cam}}$ and $\mathbf{X_{world}}$, and substituting Equation (2.9) into Equation (2.8), ($\hat{\mathbf{X}}$ in Equation (2.8) is actually $\hat{\mathbf{X}}_{\mathbf{cam}}$), we have:

$$s\ \hat{\mathbf{x}} = \mathbf{K}[\mathbf{I}|\mathbf{0}](\mathbf{R}\hat{\mathbf{X}}_{world} + \mathbf{T}) = \mathbf{K}[\mathbf{R}|\mathbf{T}]\hat{\mathbf{X}}_{world} \tag{2.10}$$

Define the projection matrix:

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{T}] \tag{2.11}$$

The camera model can finally be described as:

$$s\ \hat{\mathbf{x}} = \mathbf{P}\hat{\mathbf{X}} \tag{2.12}$$

2.2 Camera Representations

In order to position a camera in a 3-D space, we need to know both the location and the orientation of the camera. In this section, we demonstrate how to represent camera poses as well as the techniques used to obtain camera locations and camera orientations. A world reference coordinate frame is first defined to include all the cameras. A scaling factor is also chosen such that all the cameras are in a relatively close area and the distances between them are moderate. The camera pose is defined in the form of matrices. Each camera pose is associated with a matrix from which the camera location and orientation can be extracted.

2.2.1 Camera Parameters and Camera Matrices

The complete camera information comprises the *internal camera parameters* and the *external camera parameters*. The internal camera parameters are those that are attached to a specific camera and they do not change. These parameters can be found from the camera specifications given by the manufacturer, or by any camera calibration process. In computer vision, the internal camera parameters are represented by a calibration matrix **K**. The matrix **K** gives the camera information such as the camera focal length, the principal point coordinates, etc. The external camera parameters are those that define the camera location (or it is called the camera center) and the camera orientation. For each image that was taken by a camera, there is a specific camera location and a specific camera orientation associated with that image. We will use the term “image” and “camera” interchangeably in this thesis. A camera location defines where the camera is, and it is represented by a 3-vector **T** called translation vector. A camera orientation defines what direction the camera faces, and it is represented by a 3 by 3 matrix **R** called a rotation matrix.

In the computer vision field, the terms ”projection matrix”, ”camera matrix” and ”normalized camera matrix” are used. We will now explore these concepts and derive the relationships among them.

Camera Internal Parameters.

We know that the internal camera parameters can be represented by a calibration matrix **K**. **K** is a mapping from 3-D points in a camera coordinate system to 2-D points in an image coordinate system. For a CCD camera that uses the pixel coordinates, it is in the form of

$$\mathbf{K}^1 = \begin{bmatrix} m_x\, f & 0 & m_x\, p_x \\ 0 & m_y\, f & m_y\, p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.13}$$

where f is the camera focal length, $\begin{bmatrix} p_x \\ p_y \end{bmatrix}$ are the coordinates of the principal point, m_x and m_y are the number of pixels per unit distance in image coordinates in the x and y directions [8].

The calibration matrix $\mathbf{K}$ converts the 3-D Euclidean coordinates $\mathbf{X}_{cam}$ to 2-D Pixel coordinates $\mathbf{x}_{image}$.

$$s\ \hat{\mathbf{x}}_{image} = \mathbf{K}\mathbf{X}_{cam} \tag{2.14}$$

Camera External Parameters.

We know that the external camera parameters include the camera location and camera orientation. A camera location is a 3-vector in a reference coordinate system. In computer vision, it is represented by the translation $\mathbf{T}$. A camera orientation is the combined rotations of the X, Y and Z axis in a reference coordinate system. In computer vision, it is represented by the rotation matrix $\mathbf{R}$. A normalized camera matrix $\mathbf{Q}$ is a mapping from 3-D points in a world coordinate system to 3-D points in a camera coordinate system. It is defined [10] as:

$$\mathbf{Q} = [\mathbf{R}|\mathbf{T}] = \left[\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{array}\right] \tag{2.15}$$

[1]Here the pixel coordinates are used to define the matrix $\mathbf{K}$, while Euclidean coordinates are used to define the matrix $\mathbf{K}$ in Equation (2.7).

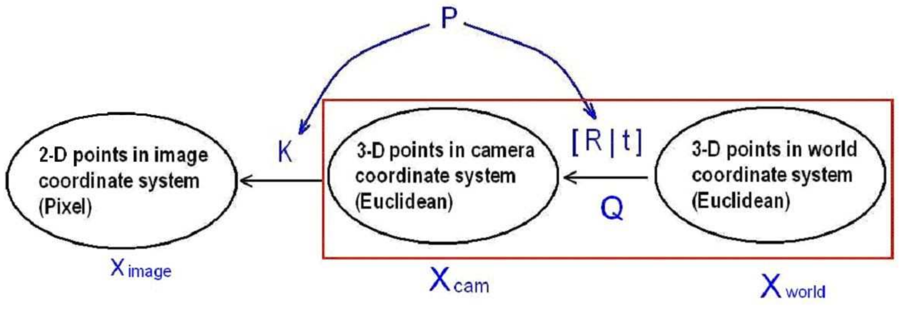
P
K
[R | t]
Q
2-D points in image coordinate system (Pixel)
3-D points in camera coordinate system (Euclidean)
3-D points in world coordinate system (Euclidean)
X image
X cam
X world

Figure 2.4: The relationship among P, K and Q.

The normalized camera matrix $\mathbf{Q}$ converts the 3-D points coordinates in the world reference frame $\mathbf{X}_{world}$ to the 3-D points in the camera reference frame $\mathbf{X}_{cam}$:

$$\mathbf{X}_{cam} = \mathbf{Q}\hat{\mathbf{X}}_{world} \tag{2.16}$$

where $\hat{\mathbf{X}}_{world}$ is the homogeneous form of $\mathbf{X}_{world}$.

Projection Matrix.

The projection matrix, which is denoted by $\mathbf{P}$, is also called the camera matrix. It is a mapping from 3-D object points to 2-D image points. The 3-D object points are in a Euclidean coordinate system, while the 2-D image points are in a *Pixel* coordinate system. The 3 by 4 projection matrix is defined [8] as:

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{T}] \tag{2.17}$$

The projection matrix maps $\mathbf{X}_{world}$ to $\mathbf{x}_{image}$:

$$s\ \hat{\mathbf{x}}_{image} = \mathbf{P}\hat{\mathbf{X}}_{world} \tag{2.18}$$

where s is a scaling factor and $\hat{\mathbf{x}}_{image}$ is the homogeneous 2-D image point of the corresponding homogeneous 3-D object point $\hat{\mathbf{X}}_{world}$.

Comparing Equation (2.15) and (2.17) we find that the projection matrix is the product of the calibration matrix and the normalized camera matrix. Figure 2.4 gives a clear view of this relationship. The 3-D points in a world reference frame are first converted to the 3-D points in a camera reference frame by applying a normalized camera matrix, and the 3-D points in a camera reference frame are then converted to the 2-D pixel image points in an image reference frame by applying a calibration matrix. Since the calibration matrix is the internal camera parameters, we do not need it when computing the camera pose. We will only focus on the normalized camera matrix $\mathbf{Q}$ because it contains all the camera pose information.

2.2.2 Rotation and Translation

Considering rotations and translations without clearly describing the context can lead to ambiguity. One has to know whether the rotations and translations refer to points or coordinate systems.

We will explain how the rotation is computed [11]. For simplicity, we only discuss the process in 2-D but 3-D rotations can be derived similarly.

Rotation of Points.

Let us suppose a point $\mathbf{p_1} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$ is rotated counter clockwise with a rotation angle θ to $\mathbf{p_2} = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$ in a fixed coordinate system. As can be seen from Figure 2.5, ρ is the distance from the coordinate origin to the point $\mathbf{p_1}$, while α is the angle from the positive x direction of the coordinate system to the vector $\overrightarrow{O\mathbf{p_1}}$. The coordinates of $\mathbf{p_1}$ and $\mathbf{p_2}$ are defined by the equations below:

$$\begin{cases} x_1 = \rho\cos\alpha \\ y_1 = \rho\sin\alpha \end{cases} \tag{2.19}$$

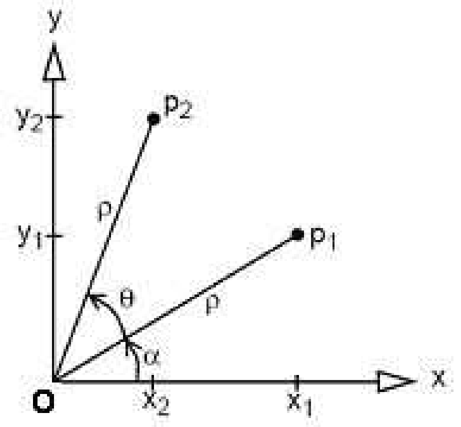

Figure 2.5: Rotation of a point.

$$\begin{cases} x_2 = \rho\cos(\alpha+\theta) = \rho\cos\alpha\cos\theta - \rho\sin\alpha\sin\theta \\ y_2 = \rho\sin(\alpha+\theta) = \rho\cos\alpha\sin\theta + \rho\sin\alpha\cos\theta \end{cases} \tag{2.20}$$

Substituting Equation (2.19) into Equation (2.20), we obtain:

$$\begin{cases} x_2 = x_1\cos\theta - y_1\sin\theta \\ y_2 = x_1\sin\theta + y_1\cos\theta \end{cases} \tag{2.21}$$

Let us rewrite Equation (2.21) in a matrix form, the coordinates of $\mathbf{p_2}$ can be computed as:

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \tag{2.22}$$

Defining the rotation matrix as: $\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$, we have:

$$\mathbf{p_1} = \mathbf{R}\mathbf{p_2} \tag{2.23}$$

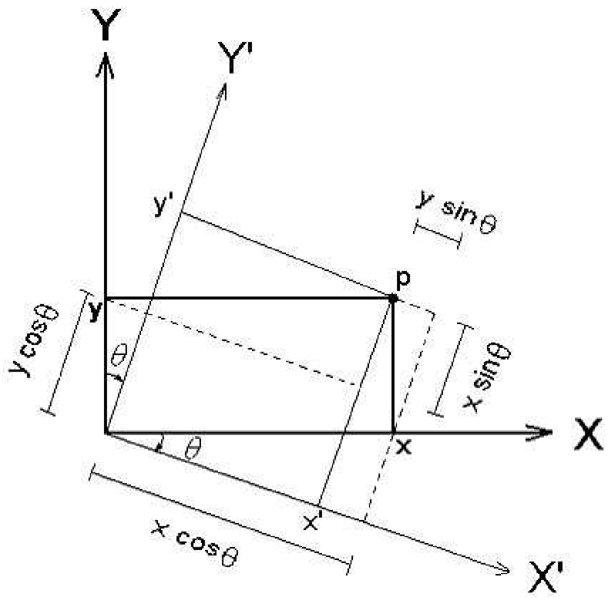

Figure 2.6: Rotation of a coordinate system.

Rotation of Coordinate Systems.

Let us suppose the coordinates of a fixed point **p** in the original coordinate frame is $\begin{bmatrix} x \\ y \end{bmatrix}$, the coordinates of the same fixed point p in the new coordinate frame that has been clockwise rotated with a rotation angle θ is $\begin{bmatrix} x' \\ y' \end{bmatrix}$. According to Figure 2.6, the new coordinates of **p** can be computed as:

$$\begin{cases} x' = x\cos\theta - y\sin\theta \\ y' = x\sin\theta + y\cos\theta \end{cases} \tag{2.24}$$

Let us rewrite Equation (2.24) in a matrix form, the coordinates of **p** in the clockwise rotated coordinate frame can be computed as:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.25}$$

Comparing Equation (2.22) with (2.25) we conclude that:

$$\boxed{\text{Rotation of a point in a fixed coordinate system is equivalent to the inverse rotation of a coordinate frame for a fixed point.}} \tag{2.26}$$

Unless specified explicitly, we do not know whether the rotation in the equation is applied on a point or is applied on a coordinate system. In the next subsection we present some definitions for the rotations and translations that should be helpful in removing the possible ambiguities.

2.2.3 Reference Coordinate Systems

In the literatures [8, 12, 9], rotation and translation are often used without explicitly indicating the object and reference. Consider the equation below:

$$\mathbf{X_{cam}} = [\mathbf{R}|\mathbf{T}]\mathbf{X} \tag{2.27}$$

No information is given in this equation as to whether $\mathbf{R}$ and $\mathbf{T}$ is rotating and translating a point or a coordinate system. Furthermore, we do not know what the reference coordinate system is for the rotation and translation.

We propose the following definitions for the different rotations and translations.

Reference Coordinate System for Moving Points:

$\mathbf{R_{pt \to w}}$: Rotation of points in the world coordinate frame.

$\mathbf{T_{pt \to w}}$: Translation of points in the world coordinate frame.

(2.28)

The reference coordinate system is always fixed for moving points. It is convenient to define a global reference when dealing with points. In this thesis, point movements are all defined w.r.t.(with respect to) the world coordinate frame.

Reference Coordinate System for Moving Coordinate Systems:

$\mathbf{R_{c\rightarrow w}}$: Rotation of the camera coordinate frame w.r.t. the world coordinate frame.

$\mathbf{T_{c\rightarrow w}}$: Translation of the camera coordinate frame w.r.t. the world coordinate frame.

$\mathbf{R_{w\rightarrow c}}$: Rotation of the world coordinate frame w.r.t. the camera coordinate frame.

$\mathbf{T_{w\rightarrow c}}$: Translation of the world coordinate frame w.r.t. the camera coordinate frame.

(2.29)

When moving a coordinate system, we always need to specify the reference. The above definitions clearly state which coordinate system is moving and which coordinate system is the reference. Performing a left multiplication of the point coordinates by a matrix means moving the point to a new position. Then, as has been presented in Equation (2.23), the implicit expression

$$\mathbf{X_{cam}} = [\ \mathbf{R} \mid \mathbf{T}\ ]\hat{\mathbf{X}} \tag{2.30}$$

should be rewritten explicitly as

$$\mathbf{X_{cam}} = [\ \mathbf{R_{pt\rightarrow w}} \mid \mathbf{T_{pt\rightarrow w}}\ ]\ \hat{\mathbf{X}}_{\mathbf{world}} \tag{2.31}$$

We have concluded that moving a point in a fixed frame is the same to moving the reference in an opposite direction. For the above example, moving the point in the world coordinate system is equivalent to the same point seen in the inversely moved camera coordinate system. In other words, the above equation is equivalent to:

$$\mathbf{X_{cam}} = [\ \mathbf{R^{-1}_{c\rightarrow w}} \mid \mathbf{T^{-1}_{c\rightarrow w}}\ ]\ \hat{\mathbf{X}}_{\mathbf{world}} \tag{2.32}$$

where $\mathbf{R^{-1}}$ is the inversed rotation and $\mathbf{T^{-1}}$ is the inversed translation.

This equation computes the coordinates of a fixed point in the camera reference frame that were inversely moved w.r.t. the world reference frame.

Since the movement between the world frame and the camera frame are relative, one more equivalent equation can be derived:

$$\mathbf{X_{cam}} = [\ \mathbf{R_{w\to c}} \mid \mathbf{T_{w\to c}}\ ]\ \mathbf{\hat{X}_{world}} \tag{2.33}$$

Comparing the Equation (2.16) with Equation (2.33), we conclude that:

The normalized camera matrix is actually the movement of the world coordinate system w.r.t the camera coordinate system:

$$\mathbf{Q} = [\ \mathbf{R_{w\to c}} \mid \mathbf{T_{w\to c}}] \tag{2.34}$$

2.3 Direct Viewing of Camera Pose

Camera poses are defined by camera matrices. It is desirable that the matrices be interpreted in 3-D space so that we can obtain a good representation of the relative cameras positions of a real scene. We first extract the camera center and the camera orientation from a normalized camera matrix, and then convert the camera orientation into an *axis angle* representation. At this stage, a camera pose can be defined by its position expressed as a 3-vector camera center and its orientation expressed as a 4-vector axis angle. The 3-vector camera center along with the 4-vector axis angle can be interpreted by any 3-D viewer and a direct view of the camera pose is possible.

2.3.1 Camera Center and Orientation

Rotations and translations have been defined for several different cases, and the camera matrices have been clearly explained. We are now ready to bring forth the way we obtain camera center $\mathbf{C}$ and camera orientation $\mathbf{O}$ from a normalized camera matrix $\mathbf{Q}$.

The camera center is defined as the coordinates of the camera frame origin in the world coordinate system, and the camera orientation is defined as the camera frame rotation in the world coordinate system. Considering the definitions we proposed in Box (2.29), we have the following identical equations:

$$\mathbf{C} \equiv \mathbf{T_{c\rightarrow w}} \tag{2.35}$$

$$\mathbf{O} \equiv \mathbf{R_{c\rightarrow w}} \tag{2.36}$$

From Equation (2.34) we understand that the normalized camera matrix contains the world rotation and world translation in a camera reference frame. To display the relative camera poses in 3-D mode, we need the camera rotation and camera translation in a world reference frame, which is the inverse of what we have in the normalized matrix. In other words, the problem becomes to convert $\mathbf{R_{w\rightarrow c}}$ to $\mathbf{R_{c\rightarrow w}}$ and to convert $\mathbf{T_{w\rightarrow c}}$ to $\mathbf{T_{c\rightarrow w}}$. This can be done by computing the inversion of $\mathbf{R_{w\rightarrow c}}$ and $\mathbf{T_{w\rightarrow c}}$.

The rotation matrix is orthogonal; its inversion is just the transpose of itself.

$$\mathbf{R_{c\rightarrow w}} = \mathbf{R^{-1}_{w\rightarrow c}} = \mathbf{R^{T}_{w\rightarrow c}} \Longrightarrow \mathbf{O} = \mathbf{R^{T}_{w\rightarrow c}} \tag{2.37}$$

It is a little bit complex to compute the translation. If there is no rotation between the camera frame and the world frame, the camera coordinate origin in the world reference frame is just the negated world frame translation $(-\mathbf{T_{w\rightarrow c}})$. Unfortunately, there always exists a rotation between the camera reference frame and the world reference frame. We must take this relative rotation into consideration. Note that the negative world frame translation is still in the camera reference frame, the camera frame rotation $\mathbf{R_{c \to w}}$ must be applied to $(-\mathbf{T_{w \to c}})$ in order to transform it to the world reference frame:

$$\mathbf{T_{c \to w}} = \mathbf{T_{w \to c}^{-1}} = \mathbf{R_{c \to w}}(-\mathbf{T_{w \to c}}) \Longrightarrow \mathbf{C} = \mathbf{O}(-\mathbf{T_{w \to c}}) \tag{2.38}$$

The above derivations demonstrate how to compute the camera center and camera orientation separately. We also have an alternative approach that can compute the camera center and camera orientation simultaneously.

Using the homogeneous forms for both $\mathbf{X_{cam}}$ and $\mathbf{X_{world}}$, and also using the augmented $\mathbf{Q}$ so that $\mathbf{Q}$ becomes a square matrix and invertible, we can rewrite Equation (2.16) as the following by left multiplication of both sides with an inverted $\hat{\mathbf{Q}}$:

$$\hat{\mathbf{Q}}^{\mathbf{-1}} \ \hat{\mathbf{X}}_{cam} = \hat{\mathbf{X}}_{world} \tag{2.39}$$

The left side of Equation (2.39) shows that the 3-D point in the camera reference frame is left multiplied by the inversed normalized camera matrix. This means the 3-D point in the camera reference frame is moved to a new position, and the coordinates of this new position in the camera reference frame is exactly the same to its original coordinates(before it was moved) in the world reference frame. Also, it is equivalent to inversely move the world coordinate frame w.r.t. the camera coordinate frame. So we have:

$$\mathbf{Q^{-1}} = \left[\begin{array}{c|c} \mathbf{R_{w \to c}^{-1}} & \mathbf{T_{w \to c}^{-1}} \end{array} \right] \tag{2.40}$$

This is equivalent to:

$$\mathbf{Q^{-1}} = \left[\begin{array}{c|c} \mathbf{R_{c \to w}} & \mathbf{T_{c \to w}} \end{array} \right] \Longrightarrow \mathbf{Q^{-1}} = \left[\begin{array}{c|c} \mathbf{O} & \mathbf{C} \end{array} \right] \tag{2.41}$$

It can be seen clearly from Equation (2.41) that the reversed normalized camera matrix contains both camera center and camera orientation in the world reference frame. In other words, we can find the camera center and camera orientation in the world reference frame simultaneously by computing the inversed normalized camera matrix.

2.3.2 Axis Angle Representation of Camera Pose

Now we have the rotation and translation of the camera coordinate frame w.r.t. the world coordinate frame, we want to view the 3-D camera pose directly. A camera translation w.r.t. a world coordinate frame $\mathbf{T_{c \rightarrow w}}$ is in the form of a 3-vector, it can be used without any modifications. A camera rotation w.r.t. a world coordinate frame $\mathbf{R_{c \rightarrow w}}$ is in the form of a matrix, it needs to be converted to the 4-vector axis angle before being sent to a 3-D viewer.

The axis angle is a 4-vector that can be used to represent rotations which consists of a unit vector and an angle of revolution about that vector. It is defined as [13]:

$$\mathbf{A} = \begin{bmatrix} x \\ y \\ z \\ \theta \end{bmatrix} \tag{2.42}$$

where the first 3 elements define the rotation axis, and the 4^{th} element defines the rotation angle.

Given a rotation matrix:

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.43}$$

The corresponding axis angle is:

$$\begin{aligned}
x &= \tfrac{r_{32}-r_{23}}{\sqrt{(r_{32}-r_{23})^2+(r_{13}-r_{31})^2+(r_{21}-r_{12})^2}} \\
y &= \tfrac{r_{13}-r_{31}}{\sqrt{(r_{32}-r_{23})^2+(r_{13}-r_{31})^2+(r_{21}-r_{12})^2}} \\
z &= \tfrac{r_{21}-r_{12}}{\sqrt{(r_{32}-r_{23})^2+(r_{13}-r_{31})^2+(r_{21}-r_{12})^2}} \\
\theta &= \cos^{-1}(\tfrac{r_{11}+r_{22}+r_{33}-1}{2})
\end{aligned} \tag{2.44}$$

For each camera in a sequence, the translation $\begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}$ defines the camera center, and the axis angle $\begin{bmatrix} x \\ y \\ z \\ \theta \end{bmatrix}$ defines the camera orientation.

After generating the file that contains the cameras defined by the translations and axis angles, we can view the corresponding 3-D camera poses directly.

Chapter 3

Bundle Adjustment

Bundle adjustment is widely used because of its remarkable advantages [8] such as: i) it can provide a true maximum likelihood estimate even when some input data are missing; ii) it allows assignment of individual covariance to each measurement and can be extended to include estimates of priors and constraints on camera parameters or 3-D scene point positions. However, bundle adjustment is not an ideal algorithm since it has inherent weaknesses that prevent it from being used directly. Thus bundle adjustment should be used in a way that can maximize its benefits while at the same time suppressing its drawbacks. This chapter explores the bundle adjustment algorithm and also introduces some applicable bundle adjustment applications.

3.1 Bundle Adjustment Algorithm

Bundle adjustment is the process by which globally visually consistent solutions are found for the structure and motion of a scene viewed by multiple cameras. As can be seen from the camera model described in Section 2.1, the 3-D scene point $\mathbf{X}$ is to be reprojected to the 2-D point $\mathbf{x}'$ in the image plane through the camera matrix. Ideally, this reprojected point $\mathbf{x}'$ is identical to the measured 2-D point $\mathbf{x}$ in the image. However, due to noise and other errors, $\mathbf{x}'$ fails to be exactly the same as $\mathbf{x}$. If an initial estimate of the structure and motion is available, bundle adjustment is able to find a solution with minimal errors for all 2-D, 3-D points and projection matrices. The bundle adjustment procedure has been described by many authors [8, 14, 15]. The problem is usually formulated as follows:

Given $\mathbf{x}_{ij}$, the i^{th} 2-D point of the j^{th} image, find the maximum likelihood camera projection matrix $\mathbf{P}'_j$ and the maximum likelihood 3-D point $\mathbf{X}'_i$ simultaneously such that the reprojected image point $\mathbf{x}'_{ij}$ is as close as possible to the given image point $\mathbf{x}_{ij}$. In general, the reprojected image point $\mathbf{x}'_{ij}$ is not identical to the measured image point $\mathbf{x}_{ij}$ because of the noise. Bundle adjustment tries to minimize the overall error between the given 2-D points and the reprojected points by adjusting all the camera projection matrices and the 3-D points. Several equivalent minimization equations of the bundle adjustment are shown below:

$$min \sum_{i,j} d(\mathbf{x}'_{ij}\ ,\ \mathbf{x}_{ij})^2 = min \sum_{i,j} d(\mathbf{P}'_j \cdot \mathbf{X}'_i\ ,\ \mathbf{x}_{ij})^2 = min \sum_{i,j} d(\mathbf{K}\mathbf{Q}'_j \cdot \mathbf{X}'_i\ ,\ \mathbf{x}_{ij})^2\ . \quad (3.1)$$

where $d(\mathbf{a}\ ,\ \mathbf{b})$ is the geometric image distance between the homogeneous points $\mathbf{a}$ and $\mathbf{b}$.

Bundle adjustment is an iterative process. First the reprojected point $\mathbf{x}'_{ij}$ is computed from the given initial estimate of the projection matrices $\mathbf{P_j}'$ and the 3-D points $\mathbf{X_i}'$. After a small adjustment to both $\mathbf{P_j}'$ and $\mathbf{X_i}'$, we again compute the reprojected point.

$$\mathbf{x}''_{ij} = (\mathbf{P}' + \mathbf{\Delta P})_j \times (\mathbf{X}' + \mathbf{\Delta X})_i \qquad (i = 1 \cdots M\ ,\ j = 1 \cdots N) \quad (3.2)$$

An iterative process based on the steps above will continue until the final error tolerance is reached. In practice, bundle adjustment does not always return a correct answer. It is a non-linear minimization process and it relies heavily on the initial estimate of the camera position and the 3-D scene points [9]. A bundle adjustment method either diverges or return a wrong estimation if the initial values were not close enough to the real values.

Another issue of bundle adjustment is that it is a very time consuming process. It is believed [9] that the bundle adjustment is computationally expensive due to the large number of input frames and features. It therefore requires the solution of a very large minimization problem because it involve a great number of parameters [8]. We are now ready to further explore the bundle adjustment algorithm and see how the input data are manipulated in the bundle adjustment process, especially how the projection matrices increments $\mathbf{\Delta P}$ and the 3-D points increments $\mathbf{\Delta X}$ are found.

3.1.1 Bundle Adjustment Implementation

There are several bundle adjustment implementation strategies that can be found in the literatures [14, 15], such as the second order Newton style method and the first order Gauss-Newton method. The second order method takes the advantages of the quadratic Taylor expansion series:

$$f(\mathbf{x}+\mathbf{\Delta}) \approx f(\mathbf{x}) + \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\mathbf{\Delta} + \frac{1}{2}\frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}^2}\mathbf{\Delta}^T\mathbf{\Delta} \tag{3.3}$$

By setting $\frac{\partial f(\mathbf{x}+\mathbf{\Delta})}{\partial \mathbf{x}} \approx \frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}^2}\mathbf{\Delta} + \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$ to zero, we have the Newton increment:

$$\mathbf{\Delta} = -(\frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}^2})^{-1}\,\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \tag{3.4}$$

The iterations based on the Newton increment makes this a second order Newton style method. The benefit of the second order method is fast convergence and fewer iterations to complete. However, the problem is that the second derivatives of the projection model $f(\mathbf{x}+\mathbf{\Delta})$ are difficult and complex to implement. Furthermore, this process may also converge to a saddle point instead of a minimum. An alternative is the first order Gauss-Newton method. If the predicted error is small (which is usually true because bundle adjustment is always used as the last step to optimize some already obtained results), we can drop the second derivatives and use only the first order Taylor expansion series. The tradeoff is that more iterations are required but each iteration is much cheaper. Faugeras [14] implements the first order method as described below:

Assuming there are N images and M 3-D points, the first order Taylor expansion is:

$$f(\mathbf{x}+\mathbf{\Delta}) = f(\mathbf{x}) + \mathbf{J}\mathbf{\Delta} + O(\| \mathbf{\Delta} \|) \tag{3.5}$$

where $\mathbf{J} = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$ is the Jacobian matrix.

The target here is to find the step $\mathbf{\Delta}$ which is the amount and the direction of the increment. It seems that building a square matrix $\mathbf{J}^T\mathbf{J}$ is a good idea to solve the problem. This $N \times N$ matrix is to be generated through the following steps.

a) Define the symmetric matrices $\mathbf{U}_j$ for each of the N images:

$$\mathbf{U}_j = \sum_{i=1}^{M} (\frac{\partial \mathbf{x}'_{ij}}{\partial \mathbf{P}_j})^T \mathbf{C}_{ij} (\frac{\partial \mathbf{x}'_{ij}}{\partial \mathbf{P}_j}) \qquad j = 1, 2, \dots N \tag{3.6}$$

$$\text{And compute:} \quad \mathbf{U} = diag(\mathbf{U}_1, \mathbf{U}_2, \dots \mathbf{U}_N) \tag{3.7}$$

b) Define the symmetric matrices $\mathbf{V}_i$ for each of the M 3-D points:

$$\mathbf{V}_i = \sum_{j=1}^{N} (\frac{\partial \mathbf{x}'_{ij}}{\partial \mathbf{X}_i})^T \mathbf{C}_{ij} (\frac{\partial \mathbf{x}'_{ij}}{\partial \mathbf{X}_i}) \qquad i = 1, 2, \dots M \tag{3.8}$$

$$\text{And compute:} \quad \mathbf{V} = diag(\mathbf{V}_1, \mathbf{V}_2, \dots \mathbf{V}_M) \tag{3.9}$$

c) Define the matrices $\mathbf{W}_{ij}$ for the N images and M 3-D points:

$$\mathbf{W}_{ij} = (\frac{\partial \mathbf{x}'_{ij}}{\partial \mathbf{P}_j})^T \mathbf{C}_{ij} (\frac{\partial \mathbf{x}'_{ij}}{\partial \mathbf{X}_i}) \qquad i = 1, 2, \dots M, \quad j = 1, 2, \dots N \tag{3.10}$$

$$\text{And compute:} \quad \mathbf{W}[i, j] = \mathbf{W}_{ij} \tag{3.11}$$

Using $\mathbf{C}_{ij} = (\mathbf{x}_{ij} - \mathbf{x}'_{ij})(\mathbf{x}_{ij} - \mathbf{x}'_{ij})^T$ in the above definitions, we can generate the $N \times N$ matrix $\mathbf{J}^T\mathbf{J}$ as:

$$\mathbf{J}^T\mathbf{J} = \begin{bmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^T & \mathbf{V} \end{bmatrix} \tag{3.12}$$

Let us define two more vectors:

$$\eta_j = \sum_{i=1}^{M} (\frac{\partial \mathbf{x}'_{ij}}{\partial \mathbf{P}_j})^T \ (\mathbf{x}_{ij} - \mathbf{x}'_{ij}) \ d^2(\mathbf{x}_{ij}, \mathbf{x}'_{ij}) \qquad j = 1, 2, \ldots N \tag{3.13}$$

$$\nu_i = \sum_{j=1}^{N} (\frac{\partial \mathbf{x}'_{ij}}{\partial \mathbf{X}_i})^T \ (\mathbf{x}_{ij} - \mathbf{x}'_{ij}) \ d^2(\mathbf{x}_{ij}, \mathbf{x}'_{ij}) \qquad i = 1, 2, \ldots M \tag{3.14}$$

where $d^2(\mathbf{a}\ ,\ \mathbf{b})$ is the squared image distance between $\mathbf{a}$ and $\mathbf{b}$.

Then we solve the following matrix equation to find the projection matrices increments $\mathbf{\Delta P}$ and the 3-D points increments $\mathbf{\Delta X}$:

$$\begin{bmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^T & \mathbf{V} \end{bmatrix} \begin{bmatrix} \mathbf{\Delta P} \\ \mathbf{\Delta X} \end{bmatrix} = \begin{bmatrix} \eta(\mathbf{P}) \\ \nu(\mathbf{X}) \end{bmatrix} \tag{3.15}$$

After obtaining $\mathbf{\Delta P}$ and $\mathbf{\Delta X}$, we plug them in Equation (3.2) to do the iterations.

3.2 Bundle Adjustment Applications

Bundle adjustment was first applied in photogrammetry, and later on was introduced to computer vision. In this section we introduce how bundle adjustment is utilized in the computer vision community. We start from the classical incremental bundle adjustment and hierarchical bundle adjustment, then we discuss the constrained bundle adjustment that applied to specific motions. We also talk about several degenerate bundle adjustment situations that expedites the process. As a final point, a novel ray-point bundle adjustment will be introduced.

3.2.1 Hierarchical Bundle Adjustment

The hierarchical bundle adjustment is somewhat like a recursive algorithm where the original problem is recursively separated into small pieces until each piece can be easily solved. The solution will then be propagated back to solve the entire problem.

Royer et al. [2] presented such a hierarchical bundle adjustment. The original long image sequence is recursively subdivided into two parts with two overlapping frames until there are only three frames in each final segment. The initial estimate of the first triplet was obtained by computing an essential matrix. Making use of the overlappings, they deduce the first two frames of the second triplet from the previous triplet. The pose estimation algorithm was used to compute the third camera position. Local estimations are done by running the bundle adjustment over all the triplet frames. These triplet frames are then merged and a global bundle adjustment is performed to find the reconstruction.

Another hierarchical method was presented by Shum et al. [9] from which the bundle adjustment was exploited efficiently with virtual key frames. Instead of recursively subdividing the original image sequence, they only divided the sequence into small segments once and no further subdivision is performed. Each segment is then processed in the same way. First a local model is initialized with a two frame structure from motion algorithm [16]. The first frame and the last frame within a segment are used to solve the structure from motion problem, and all the in-between frames are interpolated for the initialization. A few more local models are built between the first frame and some intermediate frames within a segment for better interpolations. The scale ambiguities must be eliminated before the multiple local models are combined. The scale is a coefficient applied to the 3-D points of one local model so that the 3-D points as a whole are as close as possible to the corresponding 3-D points of another local model. The scale is computed by the following formula:

$$min \sum_{i=1}^{M} (\parallel \mathbf{x}_i^m - s\ \mathbf{x}_i^n \parallel) \tag{3.16}$$

where M is the number of 3-D points, $\mathbf{x}_i^m$ and $\mathbf{x}_i^n$ are the corresponding 3-D points in model m and model n.

The bundle adjustment is applied on the segments and partial 3-D reconstructions can be found. These partial reconstructions are then merged into a complete estimate. The initial complete estimate obtained this way is usually not accurate enough. A global bundle adjustment is required to optimize the results. For long image sequences this is error prone and is inefficient. As has been described earlier in this chapter (Section 3.1), bundle adjustment is a complex process and is quite time cosuming. The complexity of the bundle adjustment for one iteration is $O(mn^3)$, where m is the number of correspondences and n is the number of frames. Obviously, reducing the number of frames will greatly simplify the bundle adjustment process. Indeed, the global bundle adjustment is only applied on the two virtual key frames selected from each reconstructed segment. The k^{th} virtual frame contains the 2-D points $\tilde{\mathbf{x}}_{ik}$ that were projected from the i^{th} 3-D reconstructed points $\mathbf{X}_i$:

$$\tilde{\mathbf{x}}_{ik} = \frac{1}{\mathbf{p}_{k3}^T\ \hat{\mathbf{X}}_i} \begin{bmatrix} \mathbf{p}_{k1}^T\ \hat{\mathbf{X}}_i \\ \mathbf{p}_{k2}^T\ \hat{\mathbf{X}}_i \end{bmatrix} \equiv \mathbf{f}(\mathbf{Q}_k, \mathbf{X}_i) \tag{3.17}$$

where $\mathbf{p}_{kr}$ is the r^{th} row of the k^{th} projection matrix $\mathbf{P}_k$, and $\mathbf{Q}_k$ is the k^{th} camera motion.

Its covariance matrix is given by:

$$\mathbf{\Lambda}_{\tilde{\mathbf{x}}_{ik}} = \frac{\partial \mathbf{f}}{\partial \mathbf{Q}} \mathbf{\Lambda}_{\mathbf{Q}_k} \frac{\partial \mathbf{f}^T}{\partial \mathbf{Q}} + \frac{\partial \mathbf{f}}{\partial \mathbf{X}} \mathbf{\Lambda}_{\mathbf{X}_i} \frac{\partial \mathbf{f}^T}{\partial \mathbf{X}} \tag{3.18}$$

Then bundle adjusting the virtual frames is equivalent to minimizing the following equation:

$$\sum_k \sum_i (\tilde{\mathbf{x}}_{ik} - \tilde{\tilde{\mathbf{x}}}_{ik})^T \mathbf{\Lambda}_{\tilde{\mathbf{x}}_{ik}}^{-1} (\tilde{\mathbf{x}}_{ik} - \tilde{\tilde{\mathbf{x}}}_{ik}) \tag{3.19}$$

where $\tilde{\tilde{\mathbf{x}}}_{ik}$ is the projection of the i^{th} estimated 3-D point $\mathbf{X}_i$ in the k^{th} image.

Bundle adjusting the virtual key frames from all the segments improved the precision of the final reconstruction. This method significantly speeds up the bundle adjustment process for long image sequences. However, the question is that if the reconstructed local segments are not close enough to the actual values, or the errors accumulated during the merging process are significant, will the the bundle adjustment still returns optimal results on virtual frames? Compared to this method, the iterative approach that we propose in the next chapter does not require precise local reconstructions and is not sensitive to accumulated errors.

3.2.2 Incremental Bundle Adjustment

Mouragnon et al. [17] presented an example of the incremental approach for reconstruction and localization. The idea is to execute the bundle adjustment whenever a new key frame and 3-D points are detected and added to the system. First the Harris s [18] are detected and matched through normalized cross correlation. Then camera poses are obtained using the five-point relative pose algorithm [19], and 3-D points are obtained using the standard triangulation. This is the sequence initialization process and optimization is achieved through bundle adjusting the key frames. Key frames are identified such that they are as far apart as possible and at the same time have enough common correspondences. If these criterions are met, then a new key frame I_i is added to the system. New points are identified as those observed only in the last three key frames (I_{i-2}, I_{i-1} and I_i). Finally when a new key frame I_i is identified, triangulation is used to reconstruct the new 3-D points. These points are then added to the system and a fresh bundle adjustment is performed to minimize the overall errors.

Another incremental bundle adjustment approach was proposed by Zhang and Shan [20] that applies a sliding window on image triplets to accommodate new frames. The first two camera motions are obtained using two-view structure from motion techniques. The third camera motion is determined by applying the three-view partial bundle adjustment [20] to the triplet. They used feature points from both two views and three views within a triplet. The window slides to the next image triplet when a new frame is added, thus reconstructing the scene incrementally. This approach is especially suitable for sparse image sequences where the difference between consecutive images is quite large. The reason is that feature points from two views are accurately estimated since the baseline between them is large. Moreover, the three view partial bundle adjustment only determines the motion of the third camera in a triplet sliding window, which ensures a consistent camera motion across the sequence.

Besides the benefit from the incremental bundle adjustment approaches discussed above, we found that they require that the involved frames be relatively far from each other(Sparse image sequences). This may not be appropriate for a video sequence since in this case the images are very close to each other (Dense image sequences with large overlapping areas between images). Also, the final result may suffer from accumulated errors arising from the multiple window sliding processes.

3.2.3 Constrained Bundle Adjustment

There are constraints on the bundle adjustment process that can be used to improve the results. One constraint is dealing with drift by finding a closed loop in the camera path. The other constraint is the use of special motions, such as an object on a turntable, to simplify the bundle adjustment process.

Drift Correction A major factor that causes bundle adjustment to diverge is drift, which is a common phenomena in extended long images due to noise and accumulated errors. Cornelis et al. proposed a method to detect and remove drift for structure from motion algorithms[21]. They detect drift based on the two major constraints, namely the proximity constraint and the similarity constraint.

The proximity constraint requires that for a closed loop sequence whose first image is identical to the last image, the distance between reprojected feature points from the same 3-D point in the first and last image must be within a certain tolerance.

$$\| \mathbf{x}_{ij} - \mathbf{P}_k \mathbf{X}_i \| \leq \text{proximity distance} \tag{3.20}$$

where $\mathbf{x}_{ij}$ is the 2-D feature point in the j^{th} image that was reprojected from the i^{th} 3-D point $\mathbf{X_i}$, and $\mathbf{P}_k$ is the projection matrix of the k^{th} image.

This constraint leads to a drastic decrease in the number of correspondences but increases the confidence of the correspondences being correct, but only if the drift was not excessive. Moreover, this constraint is quite helpful when we are processing 3-D scenes with small areas that have similar textures, such as a chessboard pattern.

The second constraint considers the textured neighborhood similarity that is a synthesis of the the color information extracted from a neighboring small area around the 2-D features. The similarity check is accomplished by comparing the differences of the textured neighborhood histograms of the corresponding 3-D points. Higher differences reflect lower similarity and potential drift.

After the drift has been detected, the adaptive bundle adjustment is exploited to remove the drift. The adaptive bundle adjustment is a weighted general bundle adjustment which minimizes the following cost function:

$$\epsilon = \sum_{i,j} \parallel \omega_{ij}(\mathbf{P}'_j \; \mathbf{X}'_i - \mathbf{x}_{ij}) \parallel^2 \tag{3.21}$$

$$\text{where } \omega_{ij} = \begin{bmatrix} \omega_{xx} & \omega_{xy} \\ \omega_{xy} & \omega_{yy} \end{bmatrix}_{ij}$$

Supposing a non-drifted 3-D point was reprojected to m feature points, and a drifted 3-D point was reprojected to n feature points. The target is to find the camera pose with similar average reprojection errors for both the non-drifted and drifted features. Thus, the non-drifted features weights are set to 1, while the drifted feature weights are set to $\sqrt{m/n}$. Since m and n are different in each frame, so are the weights. The imposed weight will force the non-drifted and drifted feature tracks to be equal for the adaptive bundle adjustment.

Turntable Image Sequence Another kind of constrained bundle adjustment may be used if the camera path is known a-priori. The turntable image sequence [22, 5] is a common example of a constrained bundle adjustment. Images were taken by a stationary camera facing the object which rotates around a fixed axis without any translational motion. Referring to the camera model described in Section 2.1, we rewrite the general image projection equation representing the i^{th} 3-D point $\mathbf{X_i}$ reprojected by the j^{th} projection matrix $\mathbf{P_j}$ to the 2-D feature point x_{ij} below:

$$\mathbf{x}_{ij} = \mathbf{P}_j\mathbf{X}_i = \mathbf{K}[\mathbf{R}^j_{w\to c}|\mathbf{T}^j_{w\to c}]\mathbf{X}_i \tag{3.22}$$

Since the object is fixed to a rotation axis, the translation vector is the same for all the images in the sequence. So the translation constrain is:

$$\mathbf{T}^1_{w\to c} = \mathbf{T}^2_{w\to c} = \cdots = \mathbf{T}^M_{w\to c} = \mathbf{T}_{w\to c} \tag{3.23}$$

The rotation matrix is a combination of the three separate rotations around x, y and z axis. Assuming the rotation around these three axis are ψ, φ and θ, respectively, we have the separate rotation matrices shown below:

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix} \tag{3.24}$$

$$\mathbf{R}_y = \begin{bmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\varphi & 0 & \cos\varphi \end{bmatrix} \tag{3.25}$$

$$\mathbf{R}_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.26}$$

The overall rotation is then the product of these three rotations:

$$\mathbf{R}_{w\to c} = \mathbf{R}_z\mathbf{R}_y\mathbf{R}_x \tag{3.27}$$

In a turntable sequence, the only rotation happens to the object itself, and the rotation is around a fixed axis. Keeping this constraint in mind, we carefully select a coordinate system to make sure that the y axis is aligned with the object rotation axis. The result is that there is no rotation around the x or z axis, and both the rotation angle for these two axis are zero:

$$\begin{aligned} \psi &= 0 \\ \theta &= 0 \end{aligned} \tag{3.28}$$

This means that the rotation matrices around these two axis become identity matrices. Then the overall rotation solely depends on $\mathbf{R}_y$:

$$\mathbf{R}_{w\to c} = \mathbf{R}_z\mathbf{R}_y\mathbf{R}_x = \mathbf{I}\mathbf{R}_y\mathbf{I} = \mathbf{R}_y \tag{3.29}$$

If the translation and rotation constraints are imposed, the unknowns in the bundle adjustment minimization equation are reduced drastically, resulting in a more stable and efficient bundle adjustment. However, the constrained bundle adjustment only benefits those image sequences that were taken with a camera that was moving in specific paths.

3.2.4 Degenerate Bundle Adjustment

Several degenerated bundle adjustment approaches can also be found in the literatures, such as the rotation free bundle adjustment [23, 24] and the intrinsic free bundle adjustment [25]. The rotation free bundle adjustment does not solve for camera orientations. In fact, the camera rotation parameters were eliminated through algebraic manipulation based on invariant theory [26]. The idea is to consider the rotational parameters as Lie group[1] parameters acting on the rest parameters of the problem. After removing the camera orientation parameters, Zhang et al. created a new structure from motion equation without any rotation matrix and then formulated a rotation matrix free cost function for

[1]Lie group was named after Sophus Lie (A Norwegian mathematician). It has compatible operations to the smooth structure.

the bundle adjustment. A rotation matrix free bundle adjustment is more robust to errors arose from the initial estimation and this is especially useful for translational motions [24].

Malis and Bartoli [25] proposed an intrinsic free bundle adjustment. They do not consider the unknown camera internal parameters in the optimization process. Instead, they proposed a bundle adjustment approach that utilizes the camera external parameters and the 3-D structure to parameterize the reconstruction problem with no internal camera parameters being considered, even they are unknowns. The intrinsic free bundle adjustment converges faster than traditional bundle adjustment. If a image sequence was taken by a same camera, the internal parameters are fixed. The intrinsic free bundle adjustment uses fewer unknowns which leads to a prompt reconstruction. Also, the solution is more accurate than the results obtained by traditional bundle adjustment.

Based on the above analysis, we find that before the benefits of the degenerated bundle adjustment can be obtained, we must convert and remove some unknown parameters that are difficult to be solved by traditional bundle adjustment. This may be suitable for some type of image sequences, but converting and removing parameters may introduce new problems.

3.2.5 Ray-Point Bundle Adjustment

Finally let us consider a novel idea to implement bundle adjustment, that is the so called *ray-point bundle adjustment* [27]. Instead of finding a optimized solution to minimize the overall errors between measured feature points and reprojected feature points, Ramalingam et al. minimize the distance between 3-D points and projection rays over camera motion and 3-D structure.

Assuming the camera is calibrated, the k^{th} projection ray of a camera is defined by a base point $\mathbf{A}_k$ and a unit normal direction $\mathbf{B}_k$. Also, the i^{th} camera pose is defined by $\mathbf{R}^i_{c\to w}$ and $\mathbf{T}^i_{c\to w}$, and the j^{th} 3-D point corresponding to the k^{th} projection ray in the i^{th} image is $\mathbf{C}_j = (X_j, Y_j, Z_j)^T$. The 3-D points on the considered projection ray can be represented by the following equation with different scalar value λ:

$$\mathbf{A}_k + \mathbf{T}^i_{c\to w} + \lambda \mathbf{R}^i_{c\to w}\mathbf{B}_k \tag{3.30}$$

The bundle adjustment minimization equation seeks the optimal solution to minimize the summation of all distances between the projection rays and 3-D points:

$$min\Sigma_{ijk} \parallel \mathbf{A}_k + \mathbf{T}^i_{c\to w} + \lambda_{ijk} \mathbf{R}^i_{c\to w}\mathbf{B}_k - \mathbf{C}_j \parallel^2 \tag{3.31}$$

This is again a non-linear equation and it can be solved by the first order Gauss-Newton method introduced in Section 3.1.1. This novel bundle adjustment works with projection rays, and it is a generic approach that can be applied to any camera as long as the camera is calibrated. The limitation is that the ray-point bundle adjustment requires sufficient data to work. Good results can be obtained only when the images are densely matched and there are sufficiently large number of rays. Comparing with the traditional bundle adjustment, we find the ray-point approach has a slower convergence rate. More iterations are needed for the ray-point bundle adjustment to produce good estimates.

3.2.6 Discussion

We have reviewed many indirect bundle adjustment methods to solve the reconstruction problems. However, few of the methods deal with loopback efficiently. This valuable information is not explicitly used. The hierarchical method speeds up the bundle adjustment process, but it still needs a good initial estimate, otherwise it can not find a converged result. The incremental method is not sensitive to the initial estimate, but it requires that the camera locations corresponding to the images be far away from each other. It is very weak to handle dense image sequences. Moreover, both the hierarchical method and incremental method may fail if there are signification accumulated errors introduced by the merging or sliding process. Constrained bundle adjustment only works for specific image sequences, while the degenerated bundle adjustment is not stable in general cases, either.

The novel ray-point method seems to be better than the above methods but it is slow to converge to an optimal solution. We believe it is possible to find the camera positions from a sequence that contains hundreds of images without a prior camera calibration, and that the images are close to each other. Furthermore, our proposed method does not require a good initial estimate, and the accumulated errors are reduced during the iterations. It also has a fast convergence rate, with the only limitation that loopback or intersection must exist in the path, as will be demonstrated in the following two chapters.

Chapter 4

Path Reconstruction of Long Image Sequences

We will now introduce our approach for the 3-D camera pose estimation of long image sequences. The goal was to devise a method that is scalable. This chapter explains in detail how we iteratively process a large number of images while at the same time taking advantage of the loop backs among the images. Our approach is capable of dealing with different lengths of sequences.

4.1 The Proposed Approach

Our full path reconstruction approach is composed of two major steps: i) camera grouping and ii) camera registration.

In the grouping step, the images of the sequence are divided into short overlapping groups. After grouping is accomplished the images in a group correspond to pictures of the scene taken from nearby locations. Consequently, the disparity between adjacent images of a group is relatively small, which means correspondences can be easily established. However, at the same time, a sufficiently large baseline must exist within the group in order to ensure that the reconstruction process remains sufficiently accurate.

It is also necessary to have a significant amount of overlap between each group. The connected groups must therefore share a certain number of common images. This redundancy makes it possible to connect the different groups together during the registration step where the individual reconstruction results are merged into a common reference frame. Since the accuracy of the resulting representation depends on the level of overlap, the groups are built to ensure that at least half of the images in a group are shared with at least one other group.

The complete path of the sequence is reconstructed by iteratively processing each group, merging the camera together through registration and then re-grouping the camera set based on the new estimated positional information. Section 4.2 will describe the grouping process, while Section 4.3 will explain how each group is connected to each other using the registration process. Figure 4.1 shows that different processes interact and how they are explained in several separate sections.

4.2 Grouping

In the initial grouping process, also called segmentation, the goal is to group together spatially neighboring cameras; however for the first iteration the pose of the cameras is unknown. Consequently, the segments are initially built based on the ordering of the image sequence. The assumption is that the images have been taken in sequence while moving the camera across the scene. As it will be shown, this is sufficient to obtain an acceptable initial estimate of the scene and to detect the potential loops in the camera path.

As explained in the next section, feature points extracted from the images of a segment are matched together. The resulting match set is sent to the Photomodeler bundle adjustment library [10] to find camera positions as well as the 3-D reconstruction. The reconstruction of all the segments of the sequence is obtained in the same way. Registra-

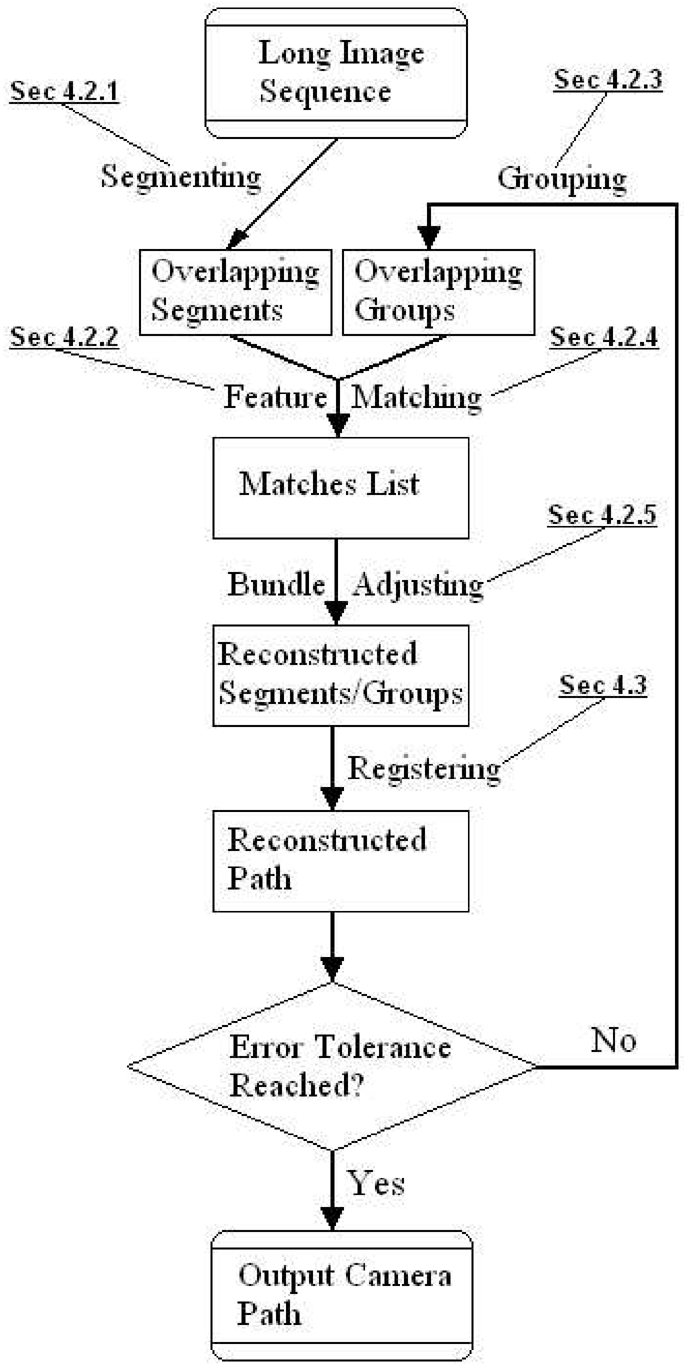

Figure 4.1: Sections associated with the system Flow Chart.

tion (see Section 4.3) is then required to merge these segments to obtain an initial 3-D model.

The grouping process will then have to be repeated on the reconstructed cameras in order to form new groups. This new grouping aims at taking into consideration the possible loops in the camera sequence that connects together non-consecutive image sub-sequences because of their spatial proximity. This grouping is realized by using the available 3-D camera pose estimates obtained from the previous iteration.

4.2.1 Camera Segmentation

This is the initial grouping process and also is the first step of the process. Segmentation is required to reduce the complexity of the bundle adjustment. From the discussion in Chapter 3 we know that bundle adjustment is very weak to handle large number of input. By reducing the amount of input data, we have a higher confident that the bundle adjustment returns good resutls. The purpose of segmentation is to divide a large input into small pieces that bundle adjustment perfers. We may need to apply bundle adjustment a few more times, but each time the bundle adjustment returns a more reliable result at a simpler computational cost. During the image acquisition process, images were taken one by one sequentially and also numbered sequentially. All the images in the long sequence will have their own IDs after the original long sequence is generated. Suppose there are N images in the sequence, and we we want segments of L images with at least $t\%$ common images between adjacent segments. This is done from the first image sequentially. The first L images are assigned to segment 1. Overlapping images will be at the rear part of a segment, so the last $t\%$ images in a segment are the common images of both the current segment and the next segment. Figure 4.2 shows that the original N images in a sequence are grouped into segments that contains L images each; the first $t\%$ images in a segment are overlapped with the previous segment and the last $t\%$ images are overlapped with the next segment.

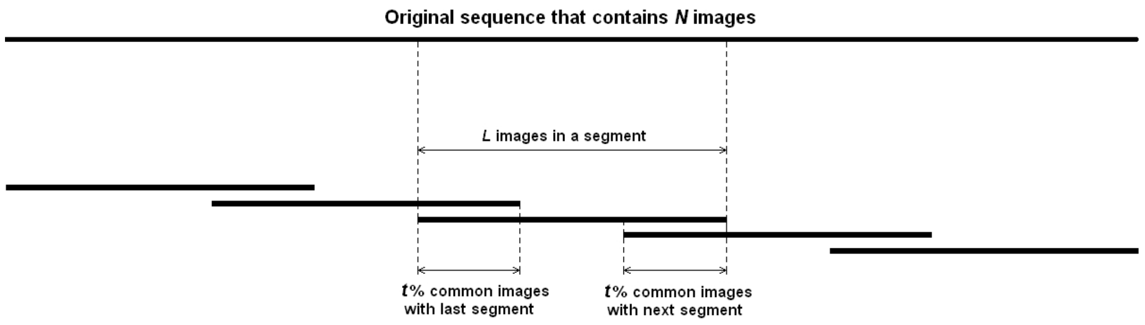
Original sequence that contains N images
L images in a segment
t% common images with last segment
t% common images with next segment

Figure 4.2: Segmentation illustration. The original N-image sequence is divided into L-image segments with $t\%$ overlappings.

4.2.2 Ordered Multi-view Correspondence

To obtain the initial match set that will be used by the bundle adjuster to reconstruct the scene, the image sequence is processed following its natural order. First the Scale Invariant Feature Transform (SIFT) features [28] are extracted. Then a RANdom SAmple Consensus RANSAC [29] strategy based on both fundamental matrix and tensor estimations is used to find reliable correspondences between images [8]. The resulting triplets of matches are then chained together across the sequence segment to get multi-view correspondences. These steps can be performed with the help of the Projective Vision Toolkit(PVT) [7].

We perform the same steps on all the segments using PVT. For any L-image segment, features will be extracted from each image first. Features are pixels with significant vertical and horizontal changes in intensity. We use the SIFT approach to obtain features from the images [28]. In order to stabilize the result, a fixed number of features are returned using an automatically selected threshold. Next, we match the features between adjacent images in a segment. Matching is done by computing the correlations of each feature in one image with nearby features in another image. Potential matches are those features with correlations greater than a specified threshold and also within a certain disparity. Dealing with false matches is an important issue in any correspondence process. The original

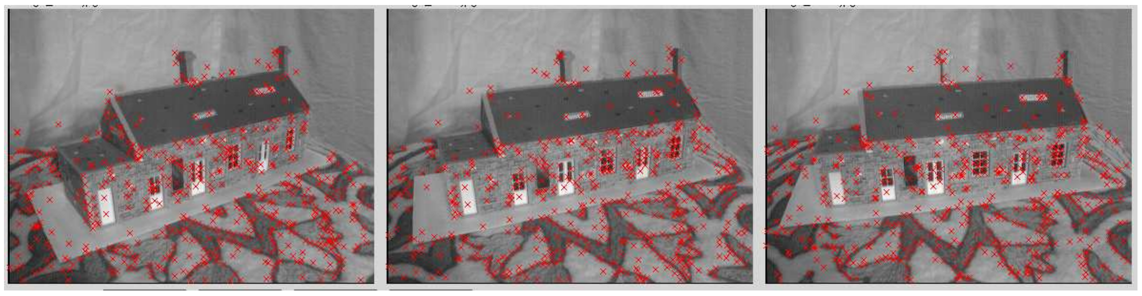

Figure 4.3: Detected correspondences from the first three images.

matches must pass a symmetry test and a consistency test in order to reduce some false matches. Suppose feature B in the second image has the strongest correlation with feature A in the first image. The symmetry test requires that feature A in the first image also has the largest correlation with feature B in the second image. In other words, feature A and feature B must have the largest correlation in either direction. The consistency test is based upon the observation that if two features are close together in the first image, then their correspondences should also be close to each other in the second image as well. The consistency test examines the disparity gradient [30] of the correspondences and keeps only those correspondences with a disparity gradient less than a certain amount. After passing the symmetry test and the consistency test, most false matches will be filtered out from the original matches list.

The model house sequence[1] is used as an sample segment to illustrate our *ordered multi-view correspondence* approach. There are ten images in this sequence and they are numbered sequentially to make the sequence an ordered segment. In Figure 4.3 we list the first three images of the segment along with the detected correspondences denoted by the cross marks. Observing the figure we can tell that the cross marks are indeed the features which match and pass the symmetry and consistency test.

The next step is to compute the fundamental matrices from the matches. Random

[1]Images are from Visual Geometry Group of Oxford University.

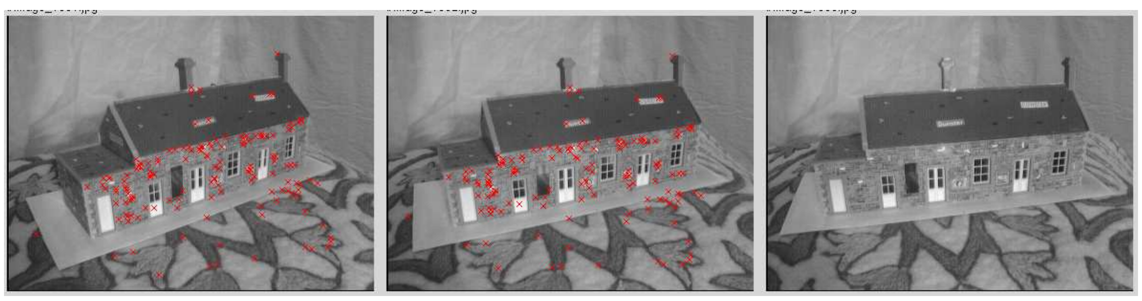

Figure 4.4: Support set correspondences of the first fundamental matrix.

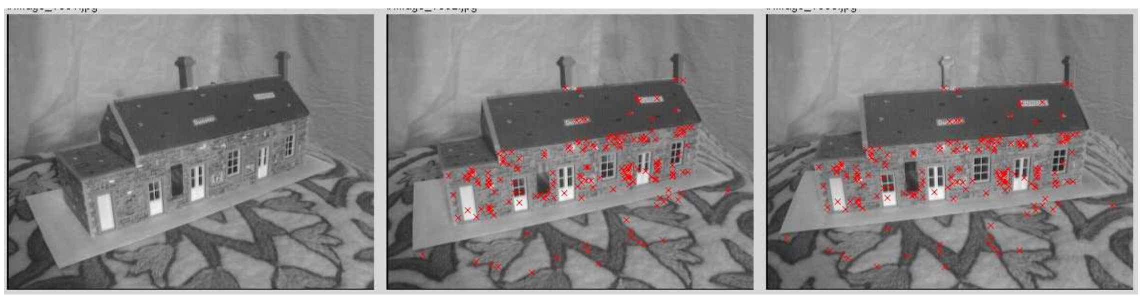

Figure 4.5: Support set correspondences of the second fundamental matrix.

sampling is used here to make the process robust. In general, at least 7 correspondences are required to define a fundamental matrix. We randomly select 7 correspondences from the matches list to compute the fundamental matrix. All the matches that satisfy the fundamental matrix form the support set. Finally, the random sampling keeps the fundamental matrix with the largest support set. One fundamental matrix and its associated largest support set will be generated for any adjacent two images. We show the support set correspondences of the first fundamental matrix in Figure 4.4 and the support set correspondences of the second fundamental matrix in Figure 4.5. At this stage, there are still invalid matches in the support set. To increase the number of valid matches, trilinear tensors will be computed from three consecutive images.

The trilinear tensor is more stable than the fundamental matrix since it relates correspondences among three images instead of two images [8]. More false matches will be

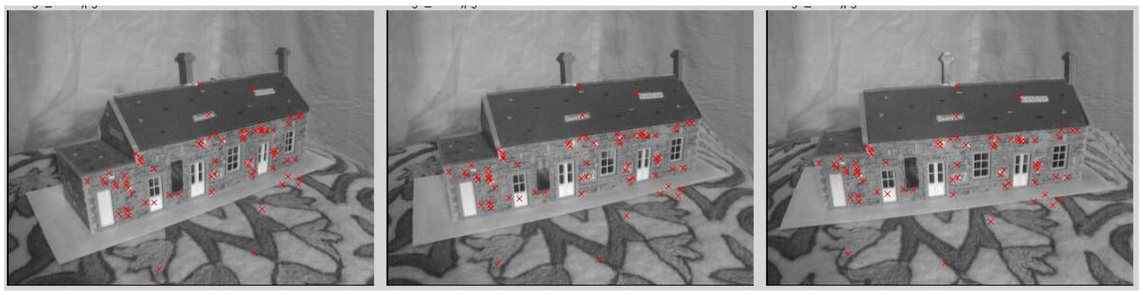

Figure 4.6: Correspondences in the support set for the trilinear tensor of the first 3 images.

filtered out after the trilinear tensor computation process. Trilinear tensors are to be computed from the support sets of two consecutive fundamental matrices. Random sampling is used again to ensure robustness. Consider any two consecutive fundamental matrices along with their corresponding support sets. A triple is formed by combining these two support sets if the matches in the sets overlap. Random sampling is performed on the matches in the triple to find the tensor. The same steps are repeated on all the triples in a segment. The final results are the tensors and the support triples, which are the 3-image matches lists. The matches in the trilinear tensor support triples are inherently more stable and more accurate than the matches in the fundamental matrix support sets. The reason is that trilinear tensors involve three images while fundamental matrices involve only two images. False matches that happened to be included in the 2-image matches lists are often removed from the 3-image matches lists when computing the trilinear tensors. The tensor computation process takes the two features that were matched in the first two images and reprojects them into the third image. The triliner tensor rejects the feature if the reprojected feature fails to be close enough to the corresponding measured feature in the third image. Figure 4.6 shows the support set features of the first trilinear tensor among the first 3 images. There are fewer remaining features since many of the original features are false matches that have been filtered out because they failed to pass those tests. Consequently, the remaining features are more accurate and more stable.

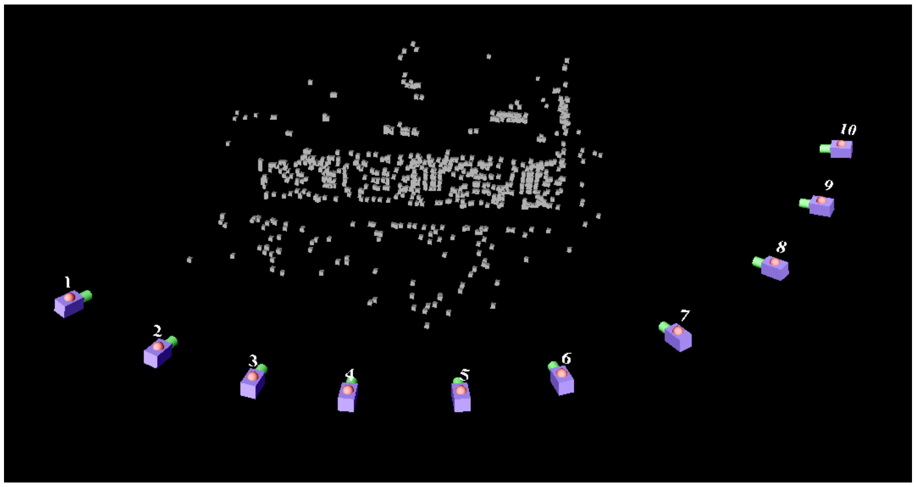

Figure 4.7: The reconstruction of the sample 10-image segment. The box like objectes represent the camera locations where the corresponding images where taken, the small white dots are the reconstructed 3-D matches corresponding to the 2-D image matches in Figure 4.6.

When the 3-image matches lists are ready, all the matched features will be tracked and be given a unique ID sequentially. Also, the image from which a feature comes will also be recorded and the corresponding image number is assigned to the feature. Finally, a text file that contains all the features with their 2-D coordinates in each image along with their ID and image number will be generated. This finalizes the ordered multi-view correspondence process. The text file will then undergo the bundle adjustment (Chapter 3) to reconstruct segments. Figure 4.7 is the 3-D reconstruction of the sample 10-image segment. When all the segments are reconstructed, they will be registered (Section 4.3) and an initial path can be found.

4.2.3 Camera Grouping

The objective here is to create a new partition of the cameras from their estimated spatial locations to form equally-sized groups of neighboring cameras so that the full group set will be connected and a level of overlap will exist between the groups.

Kanungo et al. presented the K-means clustering algorithms [31, 32]. K-means is a learning algorithms to solve the clustering problem. The main idea is to classify the given set of n data points into K clusters. The first step is to define K centroids, one for each cluster. Then it is required to assign each point to a cluster based on the nearest distance between a centroid and a point beging considered. When all the points have been assigned to clusters, the first level clustering is completed. Since the original K centroids were randomly selected, the first level cluster is not a optimal solution. Thus K new centroid have to be re-calculated. The same processes will be applied to assign the points to clusters based on the K new centroids. These continue until the re-calculated centroids remain the same to the former centroids. For the n points to be separated into K clusters, the K-means cluster algorithm aims at minimizing the following equation:

$$\sum_{j=1}^{K}\sum_{i=1}^{n}(\parallel x_i^j - c_j \parallel)^2 \tag{4.1}$$

where x_i is the i^{th} points belongs to the j^{th} cluster, and c_j is the j^{th} centroid.

Although the K-means clustering algorithm is simple to implement, it has obvious weaknesses. First of all, the algorithm is extremely sensitive to the initial randomly selected centroids and it does not necessarily find the most optimized solution. Secondly, the K-means algorithm undergoes the iterative process to find better centroids and this could be tedious. Thirdly, this algorithm forms disjoint clusters and does not handle the overlappings between clusters.

The minimum of the maximum grouping algorithm we proposed here does not have the K-means algorithm drawbacks. Our grouping algorithm not only patition the set of points into the most compact groups, but also form the most appropriate overlappings that are the closest to neighboring groups. The minimum of the maximum grouping algorithm also starts by randomly selecting a starting point, but it does not rely on this point to find an optimal solution. Instead of manually select the K centroids in the K-means algorithm, our grouping algorithm forms the groups automatically without any intervention. Also, unlike the K-means algorithm that requires iterations, our algorithm finishes its job in one step. Furthermore, since overlapping is a major factor in consideration, the grouping algorithm deals with the overlappings nicely and easily and it always ensures the minimal diameter between group of points.

Let us see how the minimum of the maximum grouping algorithm works. We have N cameras $C_1, \cdots, C_N$, and we want to create a partition made of groups $\{G_i\}$, each containing L cameras. Each camera must belong to at least one group and, to ensure good overlap, at least $t\%$ of the cameras in a group must belong to at least one other group. To create these groups, we proceed by iteratively adding cameras to the groups. To ensure that these groups form compact clusters in which all cameras are as close as possible to all other cameras in the group, we proceed as follows. The distances between an unassigned camera and all cameras in a group are computed, and the maximum distance is retained. This procedure is repeated for all the unassigned cameras and the one that has the minimum maximum-distance is selected. This camera is indeed the one that is the closest to its farthest group member thus is the one that should produce the most compact cluster. Figure 4.8 suggests one possible scenario where the current group contains three points (11, 15 and 16) and point 5 is the potential one that will be added to the current group.

1. Create two disjoint sets $\mathcal{A}$ and $\mathcal{U}$, where $\mathcal{A}$ is the assigned camera set and $\mathcal{U}$ is the ungrouped camera set. Initially, set $\mathcal{A}$ to empty while $\mathcal{U}$ contains all cameras to be processed.

2. Start with $n = 1$, randomly selected an image from $\mathcal{U}$ as the starting point; the corresponding camera C is assigned to G_n, added to $\mathcal{A}$ and removed from $\mathcal{U}$.

3. For each C_i in $\mathcal{U}$ find $d_{max}(C_i, G_n)$ by computing the distance between C_i and all cameras in G_n, where $d_{max}(C_i, G_n) = \max_{C_j \in G_n} d(C_i, C_j)$.

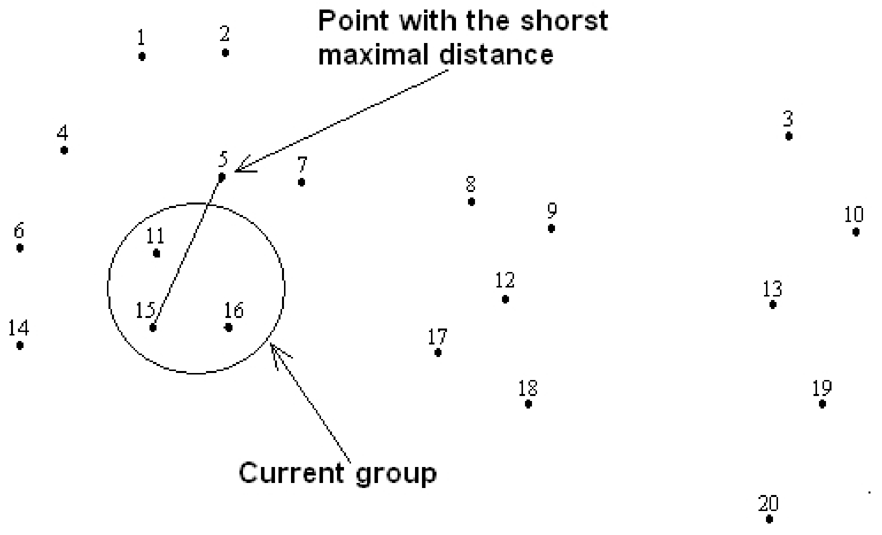

Figure 4.8: Distances comparison. The point with the shortest maximal distance is added to the group.

4. Get C_{min} that is the camera with the smallest $d_{max}(C_i, G_n)$. C_{min} is assigned to G_n, added to $\mathcal{A}$ and removed from $\mathcal{U}$.

5. Repeat step 3 and step 4 until the group size is reached or $\mathcal{U} = \emptyset$; then $n = n + 1$.

6. For each C_i in $\mathcal{U}$ find $d_M(C_i, \mathcal{A})$ with $M = tL$ (e.g. with $t = 50\%$, $M = L/2$). $d_M(C_i, \mathcal{A})$ is defined as the distance between camera C_i and its M^{th} nearest neighbor in $\mathcal{A}$.

7. Get C_{min} which is the camera with the smallest $d_M(C_i, \mathcal{A})$. C_{min} is assigned to G_n, added to $\mathcal{A}$ and removed from $\mathcal{U}$.

8. Get the M closest cameras to C_{min} in $\mathcal{A}$. All these cameras are assigned to G_n. (They constitute the overlapping cameras in the group G_n). Go to step 3.

Figure 4.9 illustrates how the groups are formed. Camera orientations do not affect the grouping approach, we use small dots to represent only the camera locations. Assuming

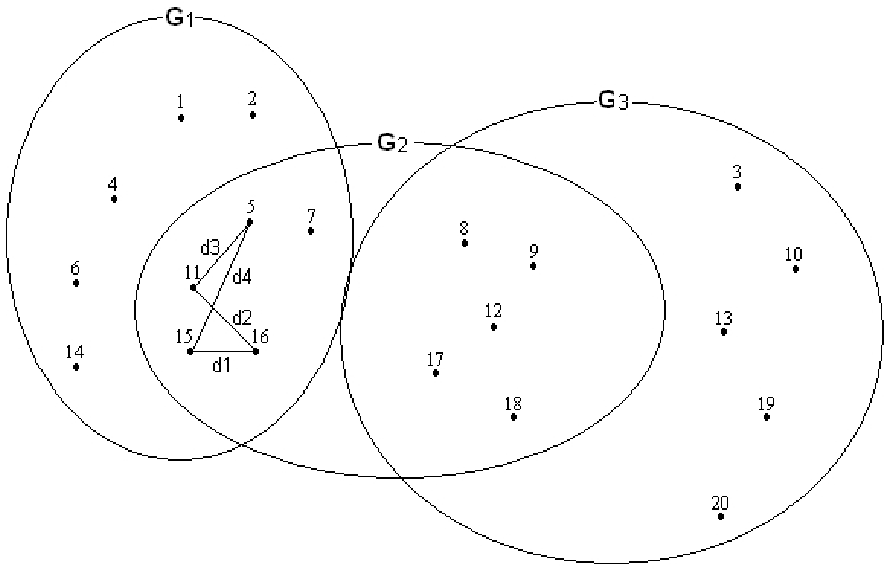

Figure 4.9: Grouping illustration. G_1, G_2 and G_3 are the three groups, G_2 is overlapped with G_1 and G_3.

there are $N = 20$ cameras to be grouped, and we want each group to contain $L = 10$ cameras with $M = 5$ overlaps. Starting from a randomly selected camera, say, camera 11, we assign it to the first group G_1. Next, the distance between an unassigned camera and camera 11 is computed. This is applied to all the unassigned cameras and the camera with the shortest distance is assigned to G_1. In this example, camera 15 is the closest camera to camera 11 and it is added to the first group. Now the group contains more than one camera and we must consider the cameras as a whole in a group. Our target is to augment a group with a camera that is the closest one to the whole group instead of the closest one to any specific camera in that group. This is accomplished by our 2-step *minimum of maximum* approach. The first step is to find the distances between an unassigned camera and all the cameras in the group and then associate the maximum distance to the unassigned camera. After we apply this step to all the unassigned cameras, each unassigned cameras will have a associated maximum distance. Let us consider camera 16 and camera 5 in this example. We compare the distances between camera 16 and all the cameras in G_1 and find that camera 11 is the one with the maximum distance ($d_2 > d_1$). For camera 5, the camera with the maximum distance is camera 15 ($d_4 > d_3$). The second step is find the minimum distance by comparing all the unassigned cameras with their associated maximum distances. The corresponding camera is the closest one to the group. We can see from Figure 4.9 that d_2 is the shortest distance , so camera 16 is added to G_1. The first group is formed this way until the group size is reached. Overlap has to be considered when we are forming the second and the following groups ($G_2, G_3, \ldots$). First the distances between an unassigned camera and all the cameras in G_1 are computed and sorted. The distance between the nearest 5^{th} neighbor and the unassigned camera is associated with that camera since the overlap size is 5. Again, the process will be applied to all the unassigned cameras. Next, we find the minimum distance by comparing distances associated to the unassigned cameras. The corresponding unassigned camera along with its 5 nearest neighboring cameras are assigned to G_2. Finally the minimum of maximum approach is used again to find the remaining cameras to be assigned to the second group. All the other groups are form in this fashion until there is no camera left. Figure 4.9 shows that the 20 cameras have been partitioned into 3 groups. Each group contains 10 cameras with 5 overlaps.

Let us analyze the complexity of the grouping algorithm here. The overall complexity is the sum of the complexities of the separate steps, as described by the following equation:

$$\begin{aligned} O(\text{grouping}) &= O(\text{ step 1}) + O(\text{step 2}) + \mathbf{O(step\ 3)} + O(\text{step 4}) + \mathbf{O(step\ 5)} \\ &\quad + \mathbf{O(step\ 6)} + O(\text{step 7}) + O(\text{step 8}) \end{aligned} \tag{4.2}$$

Steps with similar complexities are to be considered together. We understand that step 1, step 2 and step 8 contain only basic operations and consume constant time. Thus they all have the same complexity $O(1)$. For step 4 and step 7, they identify a camera through comparing. The complexities of these two steps are liner to the number of cameras

N, and their complexities are both $O(N)$. The complexities of the rest three steps are of higher orders. We analyze them one by one.

Step 3 searches for the maximum distance between cameras in the unsigned set and cameras in a group. The unsigned set could contain up to N cameras, and the group size is L. We have:

$$\mathbf{O}(\textbf{step 3}) = O(LN) \tag{4.3}$$

Step 5 repeats step 3 and step 4 until the group size L is reached, this is described by:

$$\begin{aligned} \mathbf{O}(\textbf{step 5}) &= L(O(\text{step 3}) + O(\text{step 4}) = L(O(LN) + O(N)) \\ &= O(L^2N) + O(LN) = O(L^2N) \end{aligned} \tag{4.4}$$

Step 6 searches for the M^{th} nearest neighbor in the assigned set. First the distances between an unassigned camera and the cameras in the assigned set are computed, then we sort these distances by identifying the M^{th} nearest neighbor. In the worst case, the assigned set contain all the N cameras. The complexity of this stage is $O(N + N^2)$. Since we will process all the unassigned cameras and there are up to N unassigned cameras, the complexity for Step 6 should be $O((N + N^2)N)$:

$$\mathbf{O}(\textbf{step 6}) = O((N + N^2)N) = O(N^2 + N^3) = O(N^3) \tag{4.5}$$

We know that the *Big-O* notation has the property that:

$$O(f(n)) + O(g(n)) = O(max\{|f(n)|, |g(n)|\}) \tag{4.6}$$

Thus the overall complexity is obtained below:

$$\begin{aligned} O(\text{grouping}) &= O(1) + O(1) + \mathbf{O(LN)} + O(N) + \mathbf{O(L^2N)} + \mathbf{O(N^3)} + O(N) + O(1) \\ &= \mathbf{O(N^3)} \end{aligned} \tag{4.7}$$

This is so because $N \geq L$ and $O(N^3)$ is the dominant component ($O(1) \leq O(N) \leq O(LN) \leq O(L^2N) \leq O(N^3)$).

Unlike the segmentation process that mainly relies on the continuity of the cameras, the grouping process does not require any sequential information at all. Instead, it takes advantage of the initial reconstructed path. It groups nearby cameras based on the spatial location information that can be obtained from the initial path. Furthermore, it is not necessary to have loop backs for our general grouping algorithm to be working. That is, it can be used to partition 3-D feature point cloud, too. The only information required is the coordinates of the 3-D points and there is no assumption on the form of the 3-D feature point cloud.

4.2.4 Unordered Multi-view Correspondence

Once the groups formed, valid correspondences within each group must be found. Since a group is generally made of distinct image sub-sequences, some correspondences have already been established from the previous step. The sub-sequences are then connected together using a multi-view correspondence strategy [3]. The Valbonne Church[2] image package is used as a sample unordered group since the images were taken with the camera being moved randomly.

Again, SIFT features are found and matched. More stable and accurate features are required in the iterations steps and SIFT features meet this requirement. Matchings were done on consecutive images when we were searching for ordered multi-view correspondences. For the non-ordered case here, matches will be computed between all possible image pairs. For a group containing L images each of these L images will be matched with the other $L - 1$ images. Obtaining matches between any image pairs are the same to the steps used in the segmentation process (Section 4.3). Matches must pass the symmetry test and consistency test as well. One match file will be created for one image

[2]Images are from Visual Geometry Group of Oxford University.

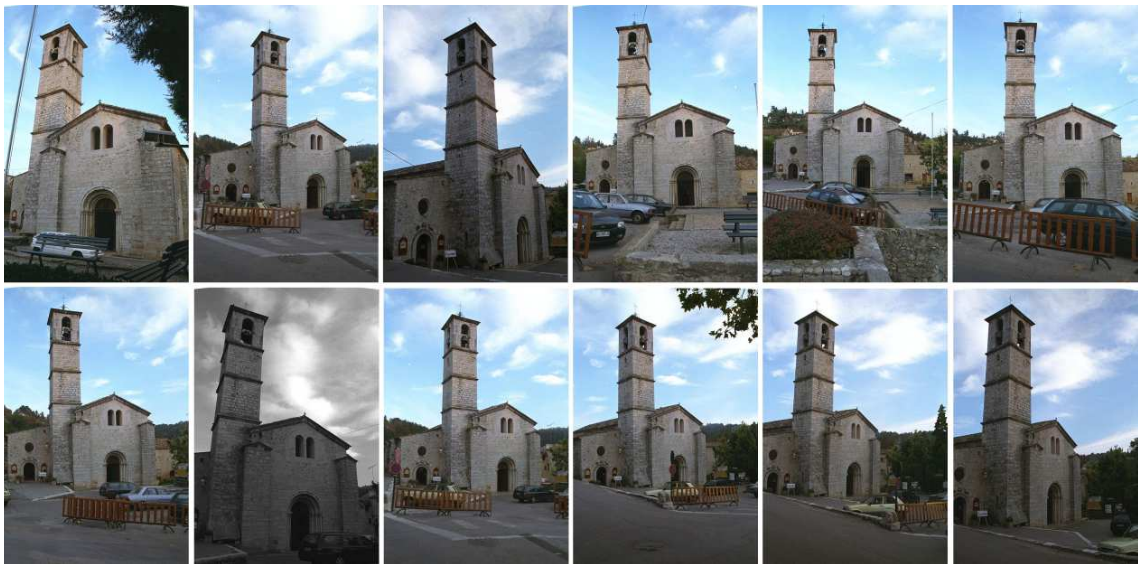

Figure 4.10: The unordered church image sequence.

pair and there are a total of $L(L-1)$ image pairs and there are this many match files. Fundamental matrices are computed from the matches between all possible image pairs. A total of $L(L-1)$ fundamental matrices will be found. Those matches that can generate valid fundamental matrices are kept and they are the support pair sets. Trilinear tensors are computed from the support pair sets and we expect a total of $L(L-1)(L-2)$ tensors. Again, those matches that can generate valid trilinear tensors are kept and they are the support triple sets. Finally we chain the triple sets and a list of the correspondences from all the images in the group are obtained.

These correspondences are again sent to the bundle adjustment package to find normalized camera matrices. Unlike the segments that are formed by sequential ordered images, the computations for finding the correspondences in the groups are more complex. It is not guaranteed that the bundle adjustment will converge on all the groups. Further processing is necessary for those groups that do not converge. We believe that the divergence is mainly caused by outliers. The difficulty is that there is no simple way to identify and remove outliers.

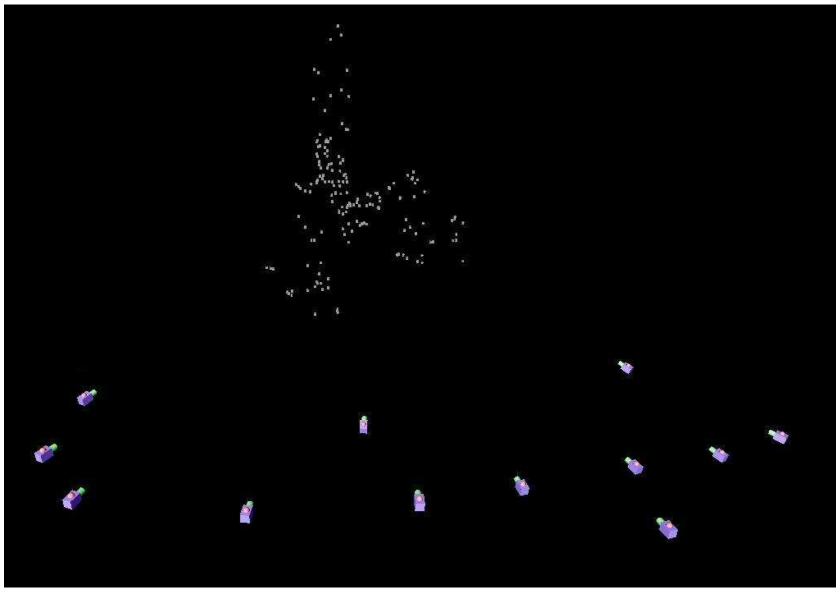

Figure 4.11: Reconstructed group. The church images are unordered, there is no IDs assigned to the reconstructed camera positions.

4.2.5 Reliable Bundle Adjustment

A bundle adjustment process is now applied to the correspondences in each group in order to compute the 3-D reconstructions of feature points and camera positions, as shown in Figure 4.11 4.11. Bundle adjustment is a complex multi-variable optimization process that is not always guaranteed to converge. It is highly affected by the presence of outliers and since the automatic correspondence process tend to produce large number of matches, the presence of such outliers is difficult to avoid. A good initial estimate of the 3-D camera positions is an important factor in obtaining reliable solutions. In addition, we also introduced additional strategies that contribute to the improvement of the reliability of the bundle adjustment process.

The automatic correspondence method produces more matches than the bundle adjustment requires. The matches set is therefore subsampled by a factor of 2 and then sent to the bundle adjuster for testing. If no convergence to a solution with a sufficiently large support is found, then the match set is further subsanpled by another factor of 2. This process is repeated until one of the subsampled set converges with good support. As the original match set is expected to contain very few false matches, two pass of subsampling are generally sufficient to obtain good convergence. It also has been observed that the bundle adjuster has a better stability when it starts with a smaller number of matches, with more matches being added as the solution improves in accuracy.

Feature points that are matched over a large number of images are also preferable for the estimation of the global 3-D structure of a group. We therefore scan the matches list and keep only those matches that exist in more than a certain number of images. By default, matches on 3 or more images are kept as tensors were obtained on 3 images. If we increase the number of images that a match must exist, more matches will be filtered out and the left ones are more reliable.

4.3 Registration

Registration is the process by which two adjacent 3-D reconstructions of points and camera positions are merged into a single reference frame. This is possible because the adjacent groups exhibit a high degree of overlap. The registration process consists in finding the similarity transform that will bring two corresponding 3-D points and 3-D camera positions to the same location, as depicted by Figure 4.12. Although registration on the overlapping 3-D feature points is possible, we found that it was more reliable to register the groups based on camera positions only.

4.3.1 Identifying the 3-D Rigid Body Transformation between Point Sets

There are many general 3-D rigid body transformation algorithms that meet our registration requirement. Eggert et al. [33] compared four popular algorithms, namely the

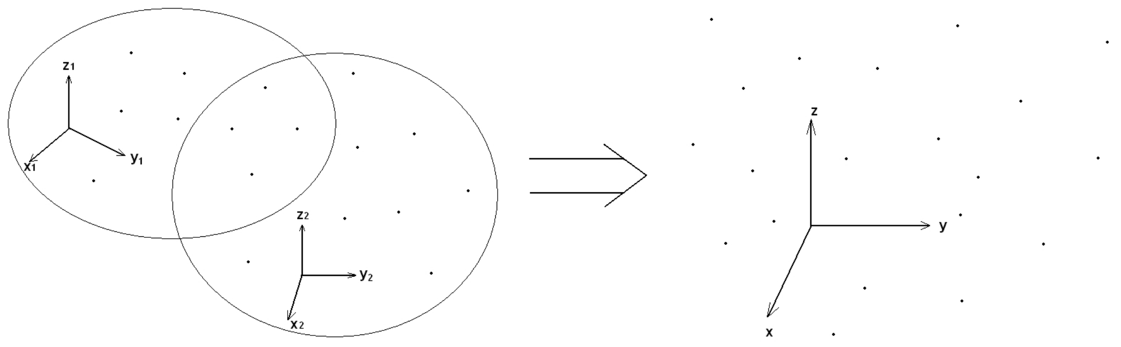

Figure 4.12: Registration of two overlapping groups.

singular value decomposition solution by Arun et al. [34], the orthonormal solution by Horn et al. [35], the unit quaternion solution by Horn [36] and the dual quaternions solution by Walker et al. [37]. These four solutions were compared with regards to accuracy, robustness, stability and efficiency. Efficiency here is not a problem since we are dealing with a small number of cameras instead of a dense 3-D point cloud. We are more concerned about the accuracy and stability of the registration. The singular value decomposition solution is more suitable than the other solutions for our small data sets(a few known camera positions without any outlier) and it is briefly summarized below.

The two 3-D point sets $\{p_i\}$ and $\{q_i\}$ are related by $q_i = \mathbf{R_{pt\to w}}\ p_i + \mathbf{T_{pt\to w}} + \mathbf{N}_i$, where $\mathbf{R_{pt\to w}}$ is the rotation matrix applied on 3-D points in the world coordinate system, $\mathbf{T_{pt\to w}}$ is the translation vector applied on 3-D points in the world coordinate system, $\mathbf{N}_i$ is a noise vector, and $i = 1, 2, \cdots, N$. The problem is to find the optimized rotation $\hat{\mathbf{R}}_{\mathbf{pt\to w}}$ and the optimized translation $\hat{\mathbf{T}}_{\mathbf{pt\to w}}$ to minimized the least squares error:

$$\sum{}^2 = \sum_{i=1}^{N} \|q_i - (\hat{\mathbf{R}}_{\mathbf{pt\to w}}\ p_i + \hat{\mathbf{T}}_{\mathbf{pt\to w}})\|^2 \tag{4.8}$$

The least squares solution to Equation (4.8) assures the same centroid for both $\{q_i\}$ and $\{q'_i = \hat{\mathbf{R}}_{\mathbf{pt\to w}}\ p_i + \hat{\mathbf{T}}_{\mathbf{pt\to w}}\}$. Define the distances between each 3-D point and the centroid of the two sets as $\bar{p}_i = p_i - \dot{p}$ and $\bar{q}_i = q_i - \dot{q}$, where $\dot{p}$ and $\dot{q}$ are the respective centroid, Equation (4.8) can be reduced to:

$$\sum{}^2 = \sum_{i=1}^{N} \|\bar{q}_i - \hat{\mathbf{R}}_{\mathbf{pt}\rightarrow\mathbf{w}}\ \bar{p}_i\|^2 = \sum_{i=1}^{N}(\bar{q}_i^T\bar{q}_i + \bar{p}_i^T\bar{p}_i - 2\bar{q}_i^T\hat{\mathbf{R}}_{\mathbf{pt}\rightarrow\mathbf{w}}\ \bar{p}_i) \tag{4.9}$$

Minimizing this equation is the same as maximizing its last term $\bar{q}_i^T\hat{\mathbf{R}}_{\mathbf{pt}\rightarrow\mathbf{w}}\ \bar{p}_i$, which is equivalent to maximizing the sum of the main diagonal elements $Trace(\hat{\mathbf{R}}_{\mathbf{pt}\rightarrow\mathbf{w}}\ \bar{p}_i\ \bar{q}_i^T)$. Define the correlation matrix $\mathbf{H} = \sum_{i=1}^{N} \bar{p}_i\ \bar{q}_i^T$, and find its singular value decomposition $\mathbf{H} = \mathbf{U\Lambda V^T}$, the optimal rotation matrix $\hat{\mathbf{R}}_{\mathbf{pt}\rightarrow\mathbf{w}}$ that can maximize the trace is:

$$\hat{\mathbf{R}}_{\mathbf{pt}\rightarrow\mathbf{w}} = \mathbf{VU^T} \tag{4.10}$$

Accordingly, the optimal translation can be found as:

$$\hat{\mathbf{T}}_{\mathbf{pt}\rightarrow\mathbf{w}} = \dot{q} - \hat{\mathbf{R}}_{\mathbf{pt}\rightarrow\mathbf{w}}\ \dot{p} \tag{4.11}$$

This is so because $\dot{q}' = \hat{\mathbf{R}}_{\mathbf{pt}\rightarrow\mathbf{w}}\ p_i + \hat{\mathbf{T}}_{\mathbf{pt}\rightarrow\mathbf{w}}$ and $\dot{q} = \dot{q}'$.

4.3.2 Registration of Camera Groups with 3-D Points

Let us first consider the case where we have to register both the cameras and the 3-D feature points from two groups. The optimal rotation and translation are computed using the overlapping camera positions of each group as the sets $\{p_i\}$ and $\{q_i\}$ of the procedure described in Section 4.3.1. The thus obtained rotation and translation is applied on the non-overlapping cameras of the second group in order for them to be fused to the first group. The overlapping cameras of the second group are discarded because they are duplicated in the first group. We use the same rotation and translation computed from the overlapping cameras and apply it to the 3D point sets.

It is not uncommon that a group overlaps with more than one groups. In this case the registration is done incrementally. Suppose group 3 overlaps with both group 1 and group 2 and these two group are themselves overlapped (If not, they can be registered to group

3 separately). We first register group 1 and group 2 and consider it as one bigger group. Group 3 is then registered to this bigger group using the procedures described below.

First of all, the relative position of a camera in the n^{th} group G_n are extracted from the normalized camera matrix $\mathbf{Q_i^n} = [\ \mathbf{R^n_{i,w\to c}} \mid \mathbf{T^n_{i,w\to c}}\ ]$ obtained as a result of the bundle adjustment step. Referring to Equation (2.37) and Equation (2.38), the i^{th} camera center as computed in the reference frame of G_n can be obtained through:

$$\mathbf{C_i^n} = (\mathbf{R^n_{i,w\to c}})^T\ (-\mathbf{T^n_{i,w\to c}}) \tag{4.12}$$

where $(\mathbf{R^n_{i,w\to c}})^T$ is the transpose of the matrix $\mathbf{R^n_{i,w\to c}}$.

Secondly, since each bundle adjustment procedure was applied independently on each group, the scales in the reconstructed camera sets are different. A consistent scaling factor must therefore be identified. This is done using the cameras that belong to more than one group. Let's consider $\mathbf{C_i}$ and $\mathbf{C_j}$ that both belong to G_n and G_m. The ratio of the distance between these two cameras, as computed in each reference is then equal to the scale factor that exists between the two groups, that is:

$$S^{mn} = \frac{d(\mathbf{C_i^m}, \mathbf{C_j^m})}{d(\mathbf{C_i^n}, \mathbf{C_j^n})} \tag{4.13}$$

where $d(\mathbf{a}, \mathbf{b})$ computes the distance between $\mathbf{a}$ and $\mathbf{b}$.

In practice, we use the mean of the scale factors computed from all pairs that are common to groups G_m and G_n.

Next, we scan the two groups and identify the overlapping cameras. Then the group will be separated into an overlapping portion and a non-overlapping portion. Based on the overlapping portion of two groups, a maximum likelihood rotation and translation is to be computed [34] in order to minimize:

$$\sum{}^2 = \sum_{i=1}^{M} \|\mathbf{C_i^m} - [\mathbf{R^{mn}_{pt\to w}}\ (S^{mn}\ \mathbf{C_i^n}) + \mathbf{T^{mn}_{pt\to w}}]\|^2 \tag{4.14}$$

where M is the number of overlapping cameras, $\mathbf{R^{mn}_{pt\rightarrow w}}$ is a 3 by 3 rotation matrix representing the orientation difference between two 3-D sets, $\mathbf{T^{mn}_{pt\rightarrow w}}$ is a 3-vector representing the translation between two 3-D sets.

After we obtained $\mathbf{R^{mn}_{pt\rightarrow w}}$ and $\mathbf{T^{mn}_{pt\rightarrow w}}$, the registration is done by applying $\mathbf{R^{mn}_{pt\rightarrow w}}$ and $\mathbf{T^{mn}_{pt\rightarrow w}}$ to the non-overlapping cameras of group G_n and fusing this registered portion of G_n to G_m. The complete registration is then obtained by iteratively connecting the non overlapping portion of each group to the registered set of cameras in this fashion. A complete estimate of the camera positions is thus obtained. Initially, this estimate will be approximate, but sufficient to form new groups, taking into account the potential loop backs in the sequence as detected by the grouping procedure. The positional estimates are then refined through a few iterations of the grouping, bundle adjusting and registering procedure.

4.3.3 Coplanar Cameras Registration

The registration algorithm we used works fine for 3-D cloud in general situations where coplanar 3-D points rarely occurs. In our case, the registration is based on a small number of overlapping cameras between the two groups, and it is normal that all the overlapping cameras are in a same plane. The coplanar cameras could be flipped after registration. Our registration algorithm must therefore accommodate coplanar cameras and give correct result.

If the cameras being registered are in a same plane, then one of the singular values of the correlation matrix $\mathbf{H}$ is zero. Observing the singular value decomposition $\mathbf{H} = \mathbf{U \Lambda V^T}$, we found that $\mathbf{H}$ is the same whether one of the columns(corresponding to the zero singular value) of $\mathbf{U}$ and $\mathbf{V}$ is positive or negative. In other words, there are two optimal rotation matrices that can minimize the least square error (Equation (4.8)). We determine the correct rotation by checking the determinant of the rotation matrix. If the determinant of $\mathbf{\hat{R}_{pt\rightarrow w}}$ is 1, then this rotation is the correct one. If the determinant of $\mathbf{\hat{R}_{pt\rightarrow w}}$ is -1, then the last column of $\mathbf{V}$ is negated and a new rotation matrix is computed. This new rotation will give the correct result.

Chapter 5

Experimental Results

We have tested the proposed algorithm on two specific camera paths: a spiral path and a snake like path, as shown in figure 5.1. We choose these two particular paths because they represent two basic scenarios: the spiral path contains both translational changes and rotational changes, while the snake like path contains only translational changes. We do not consider paths with only rotational changes since this situation is rare. Other complex paths can be built with different combinations of these two scenarios. The spiral path and the snake like path contain a number of loopbacks, which are critical for the iteration process to converge. In fact, our algorithm works for paths that contain intersections, too. The spiral and snake like paths are two examples but our method would work on any path where there is loopback or intersection in the sequence. The loopback and intersection are there to pull the drifting path due to accumulated errors back to the actual location. When the loopback and intersect camera locations are being moved to the correct position, other camera locations move toward their correct positions accordingly. As a result, the overall error in the complete path is reduced. Given a general image sequence, we can always break the path into the portions, one portion of the path contains only translational moving camera, and the other portion of the path contains translational and rotational moving camera. The translational and rotational moving camera path can be processed in

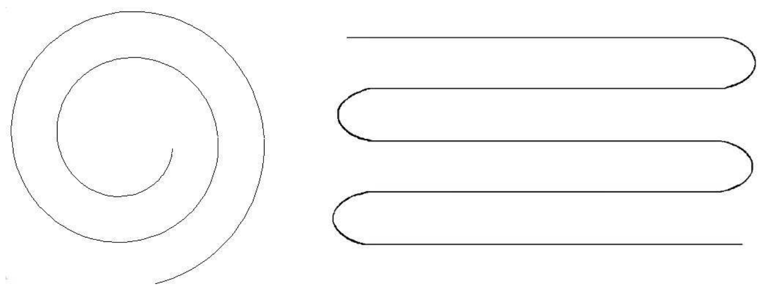

Figure 5.1: Spiral path and snake like path.

the same fashion as we did on the spiral path, while the translational only moving camera path can be processed in the same fashion as we did on the snake like path. The only limitation is that we require loopback or intersection in the path. Our path can handle complex general camera paths as long as there is loopback or intersection in the sequence.

5.1 Image Acquisition

Images were taken with the camera moving on the spiral and snake like path to generate the two image sequences. The spiral sequence was built by moving the camera on several drifting concentric circle while gradually moving the camera away from the objects. Also, the camera is always facing to the center of the circle where the objects are located. Figure 5.2 displays the first 20 images of the spiral sequence. Each camera pose has a different location and orientation, which means both translation and rotation change when the camera is moved. The objects used to build the spiral sequence were a few large boxes with rich textures. These large boxes take at least 80% area of an image and their rich textures ensure that enough matches can be extracted. Over two hundred images were taken to generate the spiral path made of about two complete turns. Abundant loopbacks can be found from the turns.

The snake like sequence was built by moving the camera from left to right and then back to left, and also moving the camera away from the object when the path changes

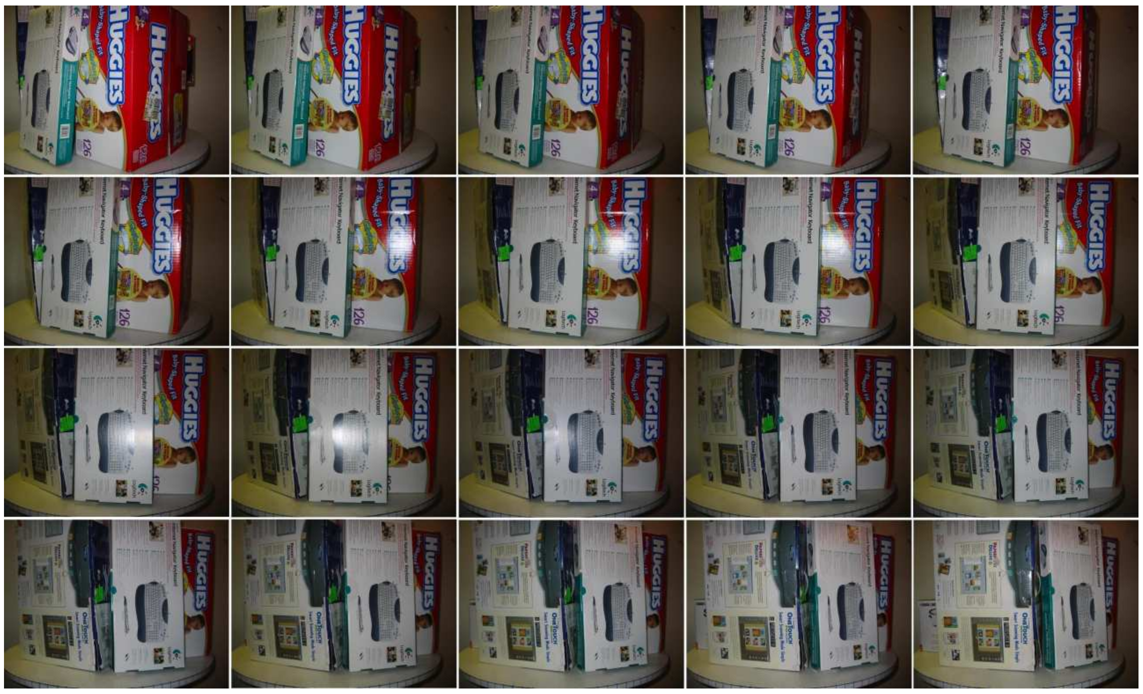

Figure 5.2: The first 20 images from the spiral path.

direction. We did not rotate the camera when we were taking the images of the snake like sequence to make sure all the cameras have the same orientation. The camera was mounted on a tripod facing a row of bookcases. These bookcases take the full image area and the many books on the bookcases give rich textures. We will describe the spiral path in more detail since the snake like path is processed similarly.

5.2 Initial Reconstruction

In this section we demonstrate how to obtain the initial reconstruction of the spiral path. The path we acquired contains 213 images. Each image is given a numbered ID sequentially. The image sequence will go through the segmentation—partial reconstruction—registration circle as described in Chapter 4.

5.2.1 Segments Reconstruction

The initial segmentation is straightforward based on the sequential image IDs. We just need to decide the length of the segment and the size of the overlap. There is always a trade off between the size of the groups and the precision of the result. Larger segments are preferred because fewer registrations are needed for the same long image sequence. The difficulty is that the longer the sequence, the less likely it is that the bundle adjustment will converge. We tried the bundle adjuster on different length of segments to find an appropriate segment length. We know that bundle adjustment does not behave very well where there are more than 30 images. This can be seen from Figure 5.3 which clearly shows that segments with less than 30 images are stable. Accordingly, 20 images in a segment seems to be a good choice. Next, we need to determine the number of overlapping images in the segments for registration. Although three images are enough to compute the rotation and translation, involving more images stabilizes the registration process. The number of overlapping images has therefore been set to be half of the total number of images in

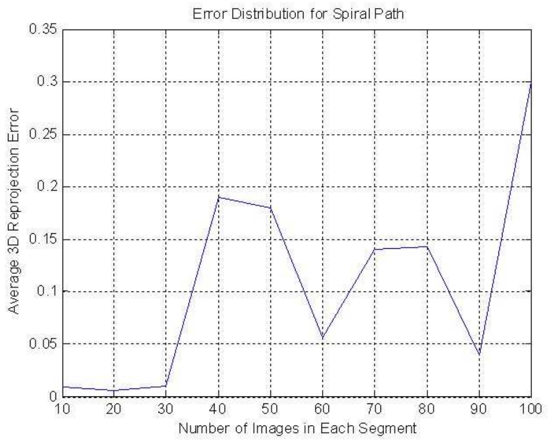

Figure 5.3: Reprojection error of different length of segments.

a segment to ensure both stability and efficiency. In our case, there are 10 overlapping images in each segment.

The projective vision toolkit is then applied on the segments. features between images are detected and matched and then fundamental matrices are computed. Images are processed according to the orders. Fundamental matrices will be found between images 1-2, 2-3, 3-4 and so on. Next, we use the fundamental matrix support sets to compute trilinear tensors. Tensors will be found among images 1-2-3, 2-3-4, 3-4-5 and so on. Figure 5.4 shows the matching features in the first a few images. After that, we chain the tensor support sets and create a list of the matches with their pixel coordinates and their own ID along with the image number to which that particular match belongs.

The final match list is sent to the bundle adjuster so that the normalized camera matrices can be found. We do not need precise calibration here. A roughly estimated calibration matrix is enough to obtain good result. Assuming the principal point is at the center of an image, the corresponding principal point offset coordinates in pixels for a 640×480 image is $(320, 240)$. A typical camera focal length is about $180mm$, and a typical pixel size is about $0.26mm$. We then use $180 \div 0.26 \approx 692$ as the focal length. The final estimated calibration matrix for the spiral image sequence is:

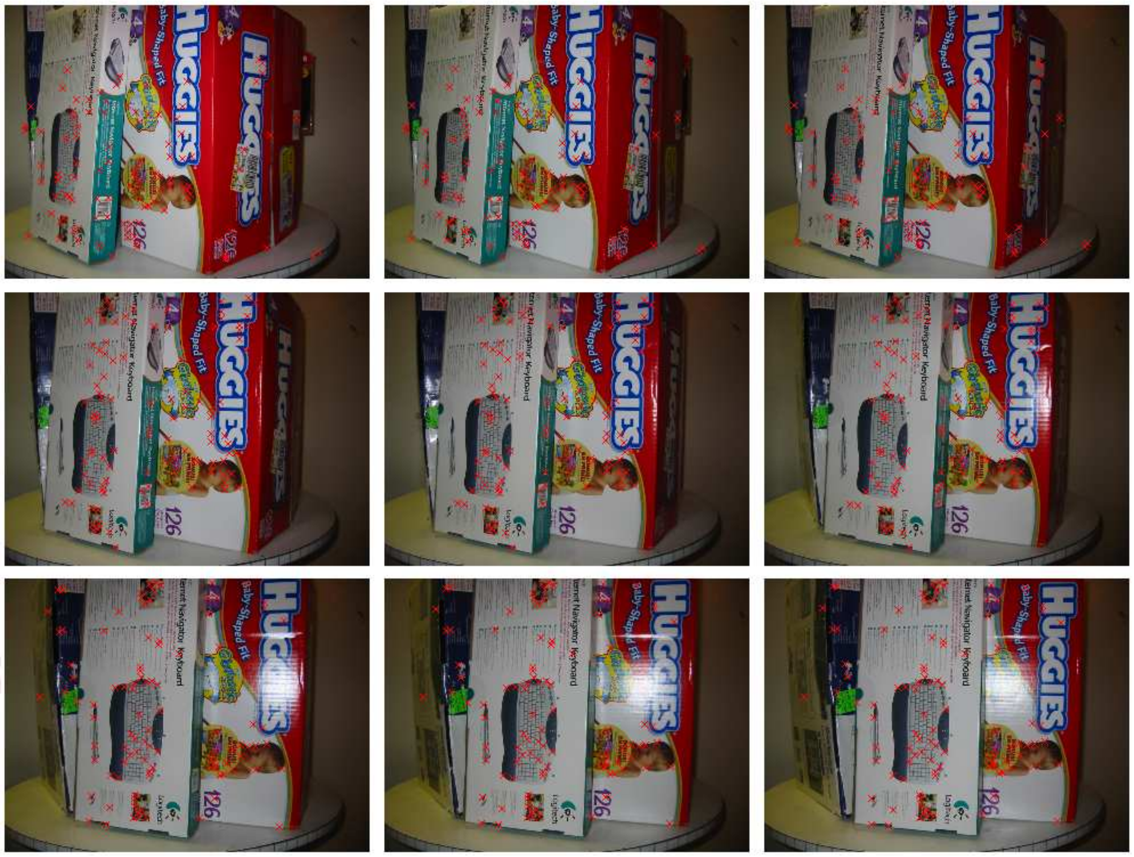

Figure 5.4: The matching features in the first a few images.

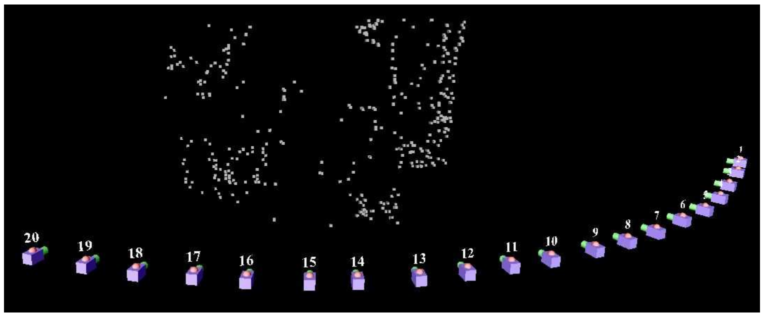

Figure 5.5: The reconstruction of the first segment.

$$\mathbf{K} = \begin{bmatrix} 692 & 0 & 320 \\ 0 & 692 & 240 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

After feeding the match list and the calibration matrix to the bundle adjuster and running the bundle adjustment, we obtain a normalized camera matrix for each of the image in a segment. We know that the inverse normalized camera matrix contains the camera center and camera orientation with respect to the world reference frame. The normalized camera matrix will first be inverted, and then the camera translation and camera rotation will be extracted from it. The camera rotation is converted to axis angle and displayed in VRML along with the camera translation. Then we can reconstruct all the segments in the same way. The first two reconstructed segments of the spiral sequence are displayed in figure 5.5 and figure 5.6. The box-like objects in the graphs are the cameras and the small dots are 3-D feature points.

5.2.2 Segment Registration

After having reconstructed all the segments, we will register these segments so that a complete reconstruction can be found. The first segment contains camera 1 to camera 20, and the second segment contains camera 11 to camera 30. Registration is done on the

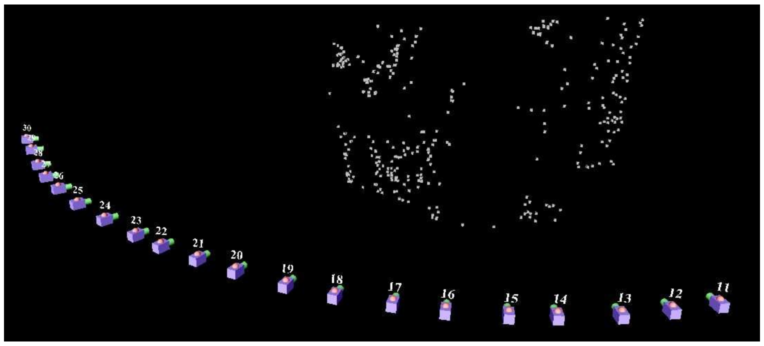

Figure 5.6: The reconstruction of the second segment.

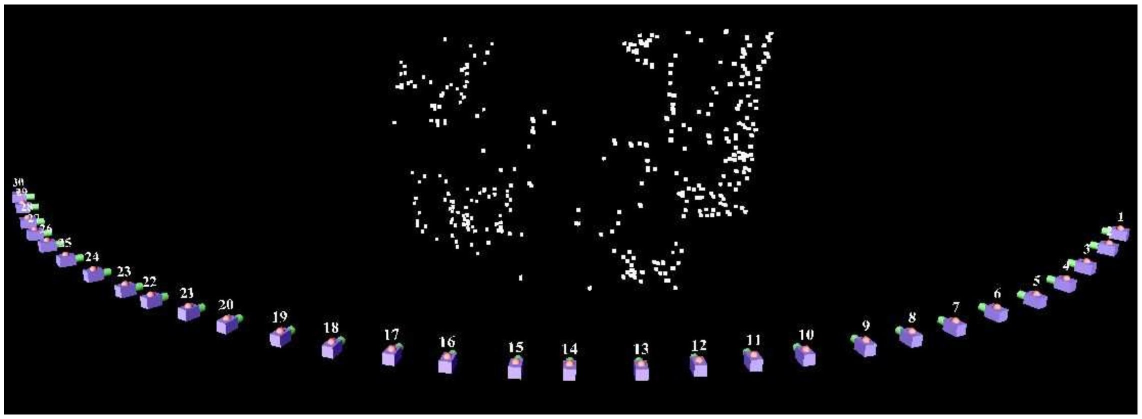

Figure 5.7: Registration of the first two segments.

common cameras (11 $\sim$ 20). We compute the camera centers of the 10 common cameras from both the first and second segment, and then try to find the best rotation and translation that can bring the common cameras of the second segment as close as possible to the corresponding common cameras of the first segment. The computed rotation and translation will then be applied to the non-overlapping cameras of the second segment. These registered non-overlapping cameras will be combined with the first segment. Figure 5.7 shows the result of fusing the first two segments.

The reconstructed segments are registered in an incremental way. Segment 2 is registered to segment 1 to form segment(1+2), and segment 3 is registered to segment(1+2)

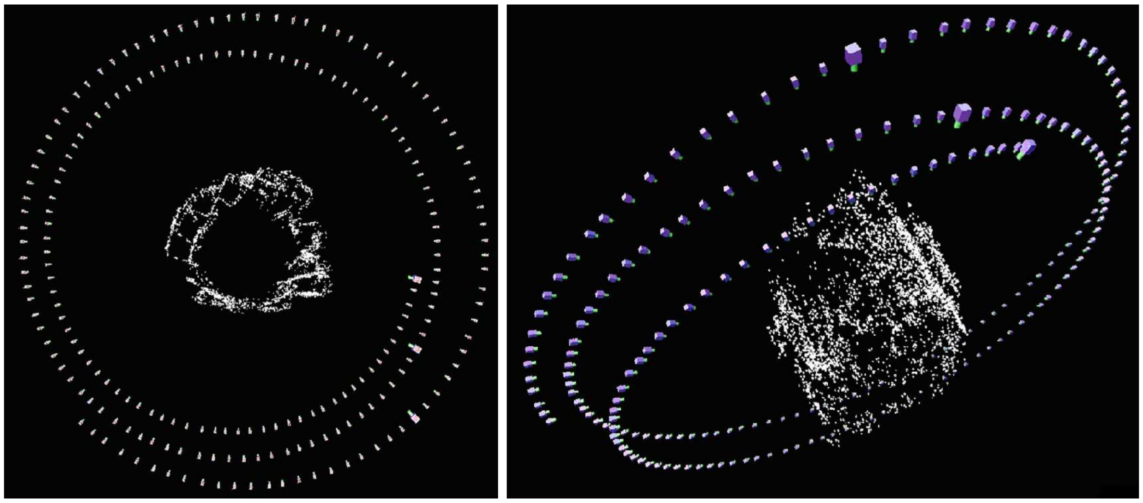

Figure 5.8: Top view and side view of the initial reconstruction. Camera 1, 96 and 191 have been enlarged; these ones should be aligned according to views shown in figure 5.9.

to from segment(1+2+3). These are done until we reach the last segment. The complete initial 3-D reconstruction of the spiral path is shown in figure 5.8.

5.3 Refined Reconstruction

In this section, we will describe how to refine the initial 3-D reconstruction. In the spiral path, a complete circle contains 95 images. Starting from camera 1, the first detected loopback is camera 96, and the second detected loopback is camera 191. We enlarged these three particular cameras in figure 5.8 for better viewing and comparing purposes. The corresponding three images are in figure 5.9.

Observing the initially reconstructed spiral path, we see three major problems:

1. Drifting errors. Camera 96 (The enlarged one on the middle circle) is supposed to be aligned to camera 1 (The enlarged one on the inner circle). The top view graph shows camera 96 is drifting away and camera 191 (The enlarged one on the outer circle) is drifting even farther.

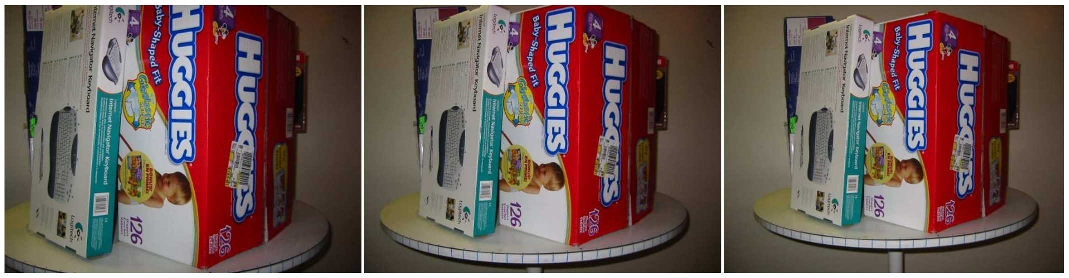

Figure 5.9: Loop back images: from left to right are image 1, 96 and 191.

2. Off path errors. The distance between the inner and outer circles is not constant while it was constant when the images were taken.

3. Off plane errors. The side view shows that the camera path is not in the same plane while the actual path is in the same plane.

However, this initial estimate is sufficient to have these cameras included in the same group at the next iteration. More iterations are performed until the reprojection error falls below the error tolerance.

5.3.1 Groups Reconstruction

Grouping is done using the algorithm described in Section 4.2.3. Starting from a randomly selected camera, the closest neighbors along with the starting camera will be collected to form group 1. Again, an appropriate group size and overlapping size have to be decided upon best reliability and efficiency. Unlike the ordered images that were processed one by one sequentially in a segment, it is much more complex to process the unordered images in a group. Hundreds of fundamental matrices and thousands of trilinear tensors will have to be computed for each group. After comparing several different combinations, we found that a group of 15 images with 8 overlapping elements meets our requirements. We apply the **Multi-view Correspondence** algorithm on the groups. The final correspondences list and the roughly estimated calibration matrix (Equation (5.1)) are sent to the bundle

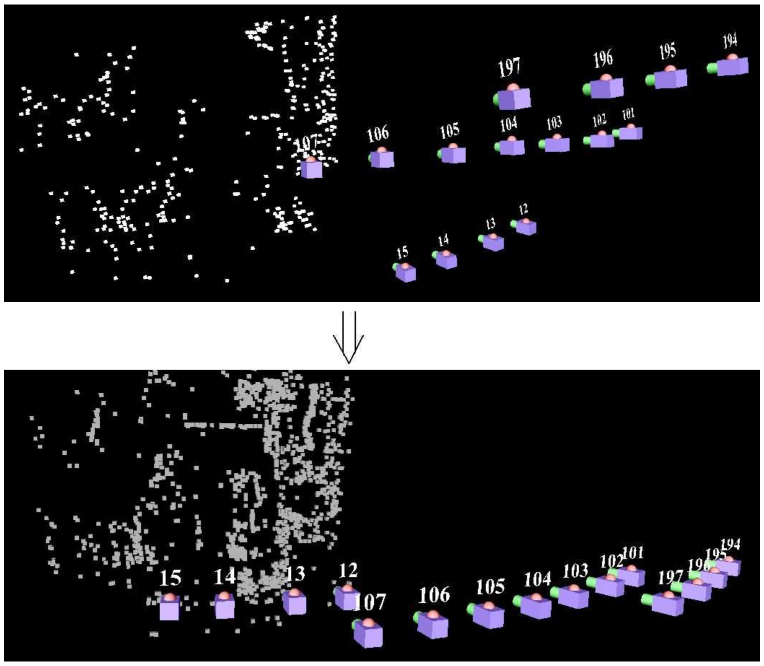

Figure 5.10: The lower figure is the reconstructed group, the upper figure is the corresponding group camera positions in the initial estimation.

adjuster package where the camera positions are computed. One of the reconstructed groups is displayed at the lower portion of Figure 5.10, where the upper potation of Figure 5.10 is the corresponding cameras in the initial 3-D reconstruction.

We form the groups with the closest neighbors based on the initial 3-D reconstruction. This can easily be identified from the upper graph in Figure 5.10, where each camera is the closest one to the rest of the group. But the lower graph in Figure 5.10 shows that shift exists between the partial circles of the path. This has been anticipated since we know there are drifting errors in the initial 3-D reconstruction and our iteration process is doing its job to reduce the drifting errors during the group reconstruction.

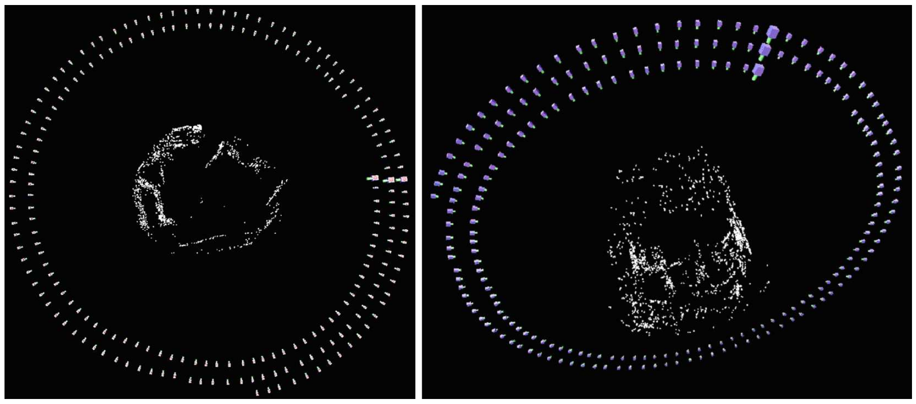

Figure 5.11: Top view and side view of the final reconstructed camera paths.

5.3.2 Groups Registration

Unlike the segments that were registered incrementally, group registration can be done in either direction. That is, for the two groups to be registered, we can keep either group unchanged but rotate and translate the other group in order for it to be connected to the unchanged group. The group being moved is called $\mathcal{G}_{\mathcal{M}}$ and the unchanged base group is called $\mathcal{G}_{\mathcal{B}}$. The optimal rotation and optimal translation are applied to $\mathcal{G}_{\mathcal{M}}$ and the non-overlapping cameras of $\mathcal{G}_{\mathcal{M}}$ will be moved in order to be connected to $\mathcal{G}_{\mathcal{B}}$, while all the cameras in $\mathcal{G}_{\mathcal{B}}$ remain unchanged. The registration result is slightly different depending on which group was chosen as the base group. We chose the group with a lower reconstruction error as $\mathcal{G}_{\mathcal{B}}$.

Next, we scan the cameras on both the two groups and identify the overlapping cameras, and then partition the groups into a non-overlapping portion ($\dot{\mathcal{G}}_{\mathcal{M}}, \dot{\mathcal{G}}_{\mathcal{B}}$) and an overlapping portion ($\ddot{\mathcal{G}}_{\mathcal{M}}, \ddot{\mathcal{G}}_{\mathcal{B}}$). The groups registration is done in two steps. The first step is to compute the rotation $\mathbf{R_{MB}}$ and translation $\mathbf{T_{MB}}$ as well as the aspect ratio S_{MB} based on the overlapping portions $\ddot{\mathcal{G}}_{\mathcal{M}}$ and $\ddot{\mathcal{G}}_{\mathcal{B}}$. The second step is to apply S_{MB}, $\mathbf{R_{MB}}$ and $\mathbf{T_{MB}}$ on the non-overlapping portion of the moving group $\dot{\mathcal{G}}_{\mathcal{M}}$ and add the moved non-overlapping cameras to the base group $\mathcal{G}_{\mathcal{B}}$. We process all the groups in the same

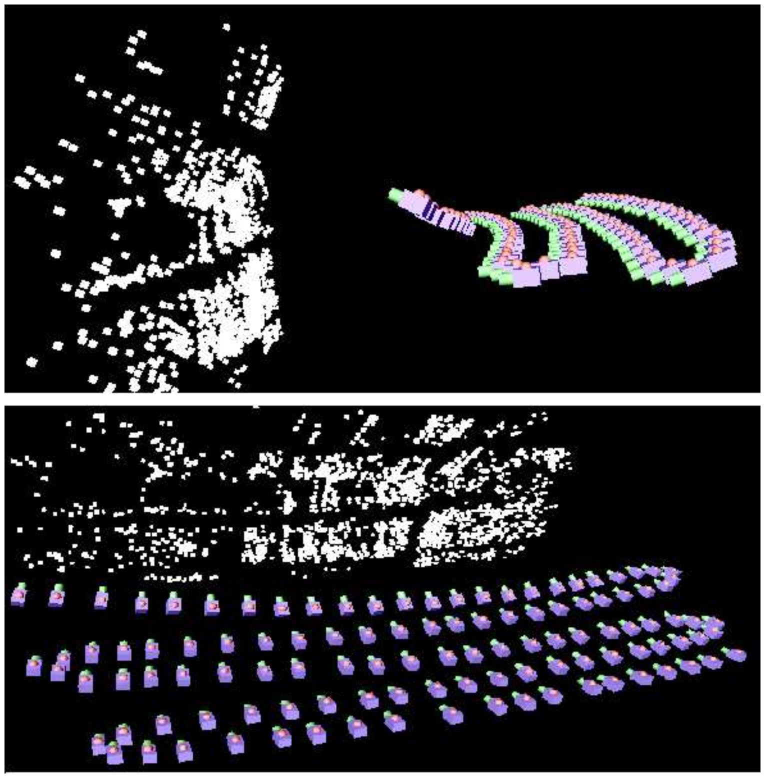

Figure 5.12: Side view and top view of the snake like path.

way. Figure 5.11 shows the final 3-D reconstruction of the spiral camera path, from which we can see that the drifting errors, the off path errors and the off plane errors have all been greatly reduced.

Our algorithm can be applied to the snake like path and other long image sequence similarly as long as loopbacks exist in the path. We process the snake like path using the same procedures that were applied on the spiral path. First the original long snake like path is separated into short overlapping segments. After applying the bundle adjustment and registration on the segments, we obtain the initial reconstruction of the snake like path. Figure 5.12 is the side view and the top view of the reconstructed path. From these two figures we can easily see the off path error and off plane error.

Again, the initial reconstruction will be used to form groups. Then bundle adjust-

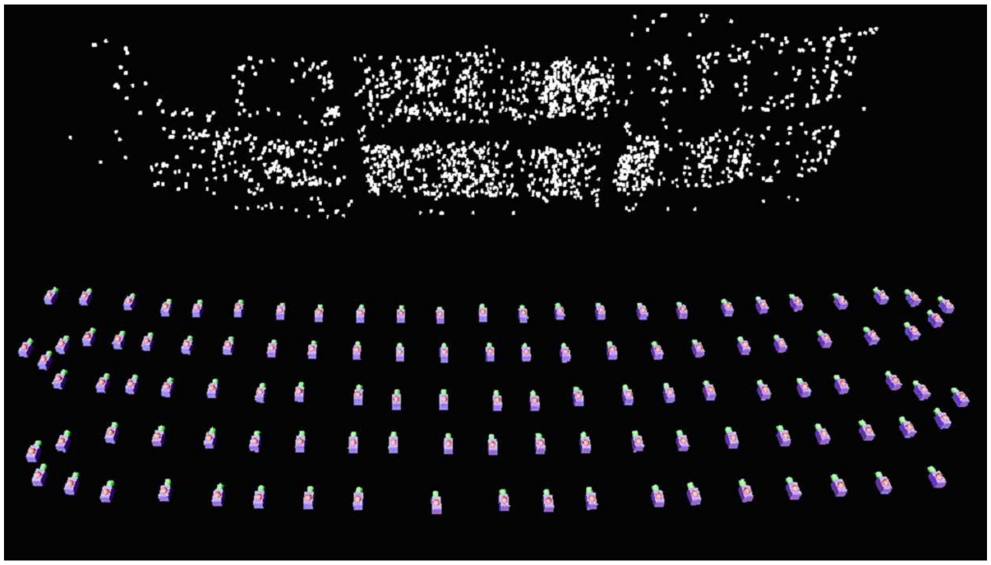

Figure 5.13: The reconstruction of the snake like path.

ment and registration will be applied on the groups and a refined reconstruction can be found. We display the reconstructed camera positions along with the 3-D feature points in Figure 5.13. When taking the images of the snake like path, we deliberately move the camera with unequal paces to demonstrate that our algorithm is also capable of dealing with unevenly separated paths.

Finally, we compare the reconstruction errors of the spiral path and the snake path. Typically there are 30 to 50 correspondences in an image. We randomly select 30 correspondences from each image to compute the reprojection errors. The obtained camera matrices and 3-D points are reprojected through:

$$\mathbf{x}'_{ij} = \mathbf{K}\mathbf{Q}'_j \cdot \mathbf{X}'_i \tag{5.2}$$

where $\mathbf{K}'$ is the camera internal calibration matrix, $\mathbf{Q}'_j$ is the j^{th} obtained camera matrix, $\mathbf{X}'_i$ is the i^{th} obtained 3-D point, and $\mathbf{x}'_{ij}$ is the projected 2-D feature point.

We then compute the square distances between the measured correspondences and the reprojected correspondences and display the results in Figure 5.14. The thin line shows

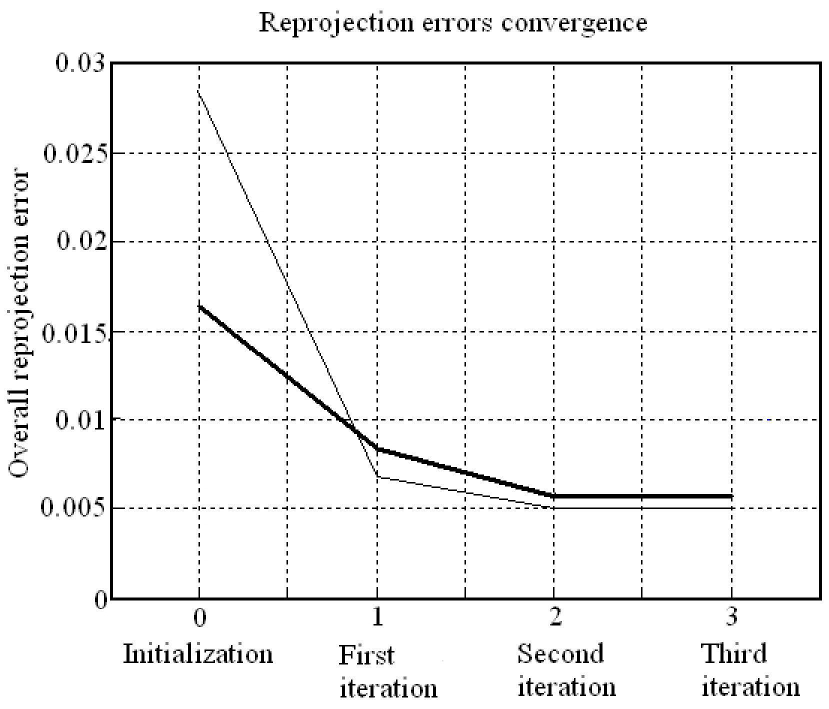

Figure 5.14: The convergency comparison of the spiral path and the snake like path. The thin line represents the spiral path error trend, the thick line represents the snake path error trend.

how the overall reprojection error of the spiral path are reduced, and the thick line shows how the overall reprojection error of the snake path are reduced. The errors in the initial estimations are very high. It is 0.28 for the spiral path and 0.16 for the snake path. The errors are all greatly reduced after the first iteration. The reason is that the loopbacks are being used during the grouping process so that the paths are moved closer to their actual locations. The following iterations slightly improve the results. Figure 5.14 clearly demonstrates the error reduction trend.

Chapter 6

Conclusion

6.1 Conclusions

We have presented an iterative algorithm to compute long camera paths. This system deals with long image sequences by using loopbacks. Loopbacks are camera locations where the corresponding images were taken and these corresponding images share field of view. We limit the bundle adjustment to only local reconstructions and this ensures precise reconstruction of ordered segments and unordered groups. Errors introduced by the registration are reduced by the iteration process and a precise complete reconstruction can be expected. Furthermore, our system does not require that the distance and angle between images to be constant. In fact, it tolerates a large difference of distances and angles between images. This is critical for videos taken with hand held cameras or vehicle mounted cameras instead of cameras being controlled by smoothly moving motors. Thus our algorithm is also appropriate for non-constant moving cameras as long as loopbacks are contained in the path.

6.2 Contributions

This work contains the following contributions:

1. Developed an algorithm to reconstruct camera paths of a corresponding long image sequence by an iterative bundle adjustment iteratively process. Due to its limitations, the bundle adjustment fails to converge when given a large number of images. We therefore divide the sequence into small groups such that good results can be obtained from local bundle adjustment. Images in a group are close to each other and at the same time the group has a large enough baseline. These ensure both reliable matches preferred by the bundle adjustment and precise 3-D reconstruction. More iterations can further refine the result.

2. Proposed to used the combination of subsample-and-test approach and long span matching approach to make the bundle adjustment more reliable. Outliers in the matching features are generally inevitable and hard to identify. They are one of the major factors that cause the bundle adjustment unstable. The subsample-and-test approach removes outliers as well as some good matching features at the same time. Having fewer matching features is not a problem because far more features are available than what are required in order to obtain a good result. Long span matching is another approach that aims at removing outliers. Keeping the matching features that exist in more than a certain number of images increases the confidence of good features. If bundle adjustment fails to converge on a given matching features list, we apply either or both the subsample-and-test approach and long span matching approach on the features list so that the bundle adjustment returns good results.

3. Our system is appropriate for camera paths that contain invariant spacing between images. It is not sensitive to the smoothness of the path. Unlike concentric mosaics [38] where the cameras are fixed to a horizontal beam supported by the tripod

and the movement is continuous and uniform, our paths are not on a strictly defined route and are not precisely separated. Our system is robust enough to handle cameras with unregulated movement easily.

4. Developed a robust grouping algorithm that can partition cameras into overlapping groups. Since we use the camera centers (3-D vectors) in the grouping, the algorithm can be used to partition 3-D points cloud as well. The K-means clustering algorithm is simple to implement but it only separate points into disjoint clusters and this dose not meet our overlapping requirement. The K centroids have to be manually selected and the K-means algorithm is sensitive to these randomly selected centroids. On the contrary, our grouping algorithm accomplish all the job automatically and it is stable with any randomly selected starting point. Our minimum of the maximum idea ensures that: i)the closest neighbors are assigned to a group and ii)the overlapping cameras are the closest to both the two neighboring groups and the diameter between groups of points is minimal.

5. Proposed the definitions of rotations and translations with specific reference systems. Rotation *between* two objects are too ambiguous. Rotation of object A with respect to object B is different than the rotation of object B with respect to object A. These two rotations are indeed the reverse of each other and one rotation matrix is the transpose of the other one. Even worse for the translations, negating one translation will not give a correct reversed translation if the reference of the two object are rotated. We defined rotations and translations explicitly with specific references and do not use the word ”between” when talking about rotations and translations.

6. One more contribution was in the automatic detection of loop back in the two sequences. Indeed it has been found that bundle adjustment is more stable when estimated on sequences that form a complete loop. Work has been done to match

features between potential loop back images in a sequence, and run the bundle adjustment again to find the camera path. This procedure is repeated until the bundle adjustment converges. Once the camera positions is obtained, a graph of all camera positions could be created from which 2-D navigation inside the environment would be possible.

6.3 Future Work

This work studies the computation of camera paths that were constrained to the size of a small room. One possible extension of this work would be 3-D camera path estimation in large-scale environments.

To achieve this goal, a large number of images have to be captured on the site of interest. Visual information will be acquired using commercial digital cameras, camcorders and specialized panoramic cameras, mounted on tripods or on small dollies. In addition, we plan to attach geo-references to part of the captured visual data. This will be done through the use of GPS (Global Positioning Systems) and digital compasses that will provide approximate information concerning absolute camera positions. The acquisition procedure must permit the complete coverage of the scene such that virtual walkthroughs will become possible. This is accomplished by moving the camera in the environment and capturing images from different points of view.

One objective of the research will be to obtain more accurate 3-D pose information from a large number of views and their approximate positions. Obtaining such 3-D information in a general context is very difficult, especially when a large number of views are used. However, the complexity of the multi-view analysis must be reduced by constraining the type of camera motion allowed in the shooting of a sequence. More specifically, the case of planar motions, obtained when a camera is moving on a flat surface, will be studied. This kind of translational motion is indeed fundamental in the production of realistic virtual walkthroughs. Different approaches to this problem have been proposed in the past, but most often the experimentation is limited to a small number of views. We would like to derive a constrained bundle adjustment solution that would be applicable to the proposed image-based rending context.

Bibliography

[1] H. Y. Shum, S. B. Kang, and S. C. Chan, "Survey of image-based representations and compression techniques," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 1020–1037, Nov. 2003.

[2] E. Royer, M. Lhuillier, M. Dhome, and T. Chateau, "Towards an alternative gps sensor in dense urban environment from visual memory," in *Proceedings of the fifteenth British Machine Vision Conference*, (London, United Kingdom), 2004.

[3] G. Roth, "Automatic correspondences for photogrammetric model building," in *International Society for Photogrammetry and Remote Sensing*, (Istanbul, Turkey), pp. 713–720, July 2004.

[4] J. Dehmeshki, X. Lin, M. Siddique, X. Ye, F. Dehmeshki, and M. Roddie, "An innovative path planning and camera direction calculation method for virtual navigation," in *Proceedings of Biomedical Engineering*, (Innsbruck, Austria), Feb. 2004.

[5] A. W. Fitzgibbon, G. Cross, and A. Zisserman, "Automatic 3d model construction for turn-table sequences," in *Proceedings of European Conference on Computer Vision Workshop on 3D Structure from Multiple Images of Large-Scale Environments*, (Freiburg, Germany), June 1998.

[6] G. Jiang, Y. Wei, L. Quan, H. Tsui, and H. Y. Shum, "Outward-looking circular motion analysis of large image sequences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 271–277, Feb. 2005.

[7] G. Roth and A. Whitehead, "Using projective vision to find camera positions in an image sequence," in *Proceedings of International Conference on Vision Interface*, (Montreal, Quebec), pp. 87–94, May 2000.

[8] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision, 2nd Edition.* Cambridge University Press, 2003.

[9] H. Y. Shum, Q. Ke, and Z. Zhang, "Efficient bundle adjustment with virtual key frames: A hierarchical approach to multi-frame structure from motion," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1999.

[10] EOS, "Photomodeler pro by eos systems inc.." http://www.photomodeler.com.

[11] P. Payeur, "Coordinate systems and transformations." Machine Vision Course Notes, the University of Ottawa, 2005.

[12] B. Triggs, "Camera pose and calibration from 4 or 5 known 3d points," in *Prococeedings of International Conference on Computer Vision*, (Kerkyra, Greece), pp. 278–284, 1999.

[13] M. Baker, "Maths - conversion matrix to axis angle." http://www.euclideanspace.com/maths/geometry/rotations/conversions/matrixToAngle.

[14] O. Faugeras and Q. Luong, *The Geometry of Multiple Images.* Cambridge, Massachusetts: The MIT Press, 2001.

[15] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, “Bundle adjustment - a modern synthesis,” in *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, pp. 298–372, Sept. 1999.

[16] Z. Zhang, “Determining the epipolar geometry and its uncertainty: A review,” *International Journal of Computer Vision*, vol. 27, pp. 161–195, Nov. 1998.

[17] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P.sayd, “Real time locolization and 3d reconstruction,” in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (New York), June 2006.

[18] C. Harris and M. Stephens, “A combined corner and edge detection,” in *Proceedings of The Fourth Alvey Vision Conference*, pp. 147–151, 1988.

[19] D. Nister, “An efficient solution to the five-point relative pose problem,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 756–770, June 2004.

[20] Z. Zhang and Y. Shan, “Incremental motion estimation through modified bundle adjustment,” in *Prococeedings of International Conference on Image Processing*, (Barcelona, Catalonia, Spain), pp. 343–346, Sept. 2003.

[21] K. Cornelis, F. Verbiest, and L. V. Gool, “Drift detection and removal for sequential structure from motion algorithms,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 1249–1259, Oct. 2004.

[22] K. Wong and M. Chang, “3d model reconstruction by constrained bundle adjustment,” in *Prococeedings of the 17th International Conference on Pattern Recognition*, pp. 902–905, Aug. 2004.

[23] J. Zhang, M. Boutin, and D. Aliaga, "Robust bundle adjustment for structure from motion," in *Prococeedings of International Conference on Image Processing*, pp. 2185–2188, Oct. 2006.

[24] P. Hammarstedt and A. Heyden, "Euclidean reconstruction from translational motion using multiple cameras," in *Proceedings of Fifth International Conference on 3-D Digital Imaging and Modeling*, pp. 352–359, June 2005.

[25] E. Malis and A. Bartoli, "Euclidean reconstruction independent on camera intrinsic parameters," in *Proceedings of International Conference on Intelligent Robots and Systems*, pp. 2313–2318, Sept. 2004.

[26] P. Bazin and M. Boutin, "Structure from motion: A new look from the point of view of invariant theory," *SIAM Journal on Applied Mathematics*, vol. 64, no. 4, pp. 1156–1174, 2004.

[27] S. Ramalingamand, S. Lodhaand, and P. Sturm, "A generic structure-from-motion framework," *Computer Vision and Image Understanding*, vol. 103, pp. 218–228, Sept. 2006.

[28] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

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

[30] R. Klette, K. Schluns, and A. Koschan, *Computer vision : three-dimensional data from images.* New York: Springer, 1998.

[31] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "A local search approximation algorithm for k-means clustering," in *Proceedings of the eigh-*

teenth annual symposium on Computational geometry, (Barcelona, Spain), pp. 10–18, 2003.

[32] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An efficient k-means clustering algorithm: analysis andimplementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 881–892, July 2002.

[33] D. W. Eggert, A. Lorusso, and R. B. Fisher, "Estimating 3-d rigid body transformations: a comparison of four major algorithms," *Machine Vision and Applications*, vol. 9, pp. 272–290, Mar. 1997.

[34] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, pp. 698–700, Sept. 1987.

[35] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," *Journal of the Optical Society of America A: Optics, Image Science, and Vision*, vol. 5, pp. 1127–1135, July 1988.

[36] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.

[37] M. W. Walker, L. Shao, and R. A. Volz, "Estimating 3-d location parameters using dual number quaternions," *CVGIP: Image Understanding*, vol. 54, pp. 358–367, Nov. 1991.

[38] H. Y. Shum and L. W. He, "Rendering with concentric mosaics," in *Proceedings of Computer Graphics (SIGGRAPH)*, (Los Angeles, CA), pp. 299–306, Aug. 1999.