Panorama Interpolation for Novel View Composition

Yeung Kwok Ho

A dissertation submitted to
the University of Hong Kong
in partial fulfillment of the requirements for
the degree of Master of Philosophy

August 2000

Abstract of thesis entitled
Panorama Interpolation for Novel View Composition

submitted by Yeung Kwok Ho

for the degree of Master of Philosophy
at the University of Hong Kong in August 2000

The major task of computer graphics is to generate computer images. There are various rendering methods, distinguished from one another in aspects such as the modeling method, the storage requirement, the quality of output and the rendering speed. Some rendering methods also impose restrictions on the possible view positions. Each method has its own advantages and drawbacks.

We have developed a framework for an image based rendering system which allows the viewer to move freely within the environment with modest storage requirement. In addition, the method can acquire the input from existing images to avoid difficult modeling jobs. Our method is designed in a way that it can be fully supported by graphics hardware for real time display. Hence we have combined the advantages of different rendering methods.

We have chosen to generate the novel views by interpolating the reference panoramas. The reason for using panorama is that it has unlimited field of view, therefore the viewer is able to rotate freely within the novel panorama. The method is based on establishing the correspondence information by matching homogeneous regions between the reference panoramas. The matching algorithm is performed automatically. We have provided a new matching score for finding the best match among the image regions. The matching process is transformed to the problem of finding the maximum weighted matching in a bipartite graph.

Contents

List of Figures

Chapter 1

Introduction

The major task of computer graphics is to generate computer images. Researches in computer graphics aim at producing better and better images. This leads to various rendering methods and they are different from how the images are generated. These methods can be distinguished in terms of several characteristics.

One characteristic is the *modeling method*, which determines how the input is acquired and the type of input. There are two major approaches. Geometric based renderings require 3D models as input and image based renderings accept images as input. A related characteristic is the *storage requirement*, that is, the space required to store the input. Of course small storage requirement is the preference, but this may affect the *quality* of the final images. The quality of the images is determined by how they resemble the real photographs. Often, we need to make a tradeoff between these two characteristics. Another important feature of the rendering methods is how *efficient* the images can be generated.

Different applications have different requirements on the rendering methods, thus some methods are more suitable than others for a particular application. While quality is the most important factor in generating static images, an additional characteristic we need to think about is how we obtain the model when dealing with animation. For virtual reality applications, real time rendering is the first priority. However, the modeling method is also a concern if we are attempting to build a complex virtual environment. The quality of output is also important to the high-end systems. The major issus in the fast growing Internet graphics applications is how to keep the file size small, so that the download time can be minimized and the application can survive with the narrow bandwidth.

When designing a rendering system, we have to determine how to integrate those good features so that it can be used with most applications. Difficult modeling jobs should be avoided and the input to the system have to be easy to acquire. We will prefer less storage requirement, but still can benefit from the high quality of output. It would be advantageous if the images can be generated in an interactive rate.

1.1 Motivation

Image based rendering has some advantages over geometric based rendering. The result is more photo-realistic, and most importantly, modeling of 3D objects can be avoided. However, it will restrict the available motion of the virtual camera. On the other hand, geometric based rendering can be supported by existing graphics hardware, which can ensure real time display in most cases.

Panorama can give unlimited field of view which allows the viewer to rotate freely within a fixed view position. This is important as the viewer can experience the feeling of immersion in the virtual environment by looking around. However, it would be better if we can allow the viewer to translate. Panorama can be easily created from planar images taken from the same view position with ordinary cameras. Therefore it can be obtained without the help of any special hardware.

Correspondence refers to the relation between image points of different images. This is useful as this can be used to study the relation between the image points and the scene points and is very important for transforming the reference images to the novel view. Although the correspondence information is difficult to obtain, it is the key to reduce the storage requirement in image based rendering.

Based on the above discussions, we see that we can combine the advantages of both image based rendering and geometric based rendering by building an image based system based on correspondence matching with panormamas as reference.

1.2 Contributions

We have developed a framework for image based rendering. From two given panoramas of an environment, we are able to generate a set of new views by interpolating the panoramas. The contributions of this thesis is listed as follows.

- We have provided a method to align the reference panoramas so that the correspondence matching can be performed easily.

- The correspondence problem is solved by matching homogeneous regions extracted from the reference panoramas.
- A new matching score for the region matching algorithm is introduced, which can take into account the colors, shapes and positions of the corresponding regions.
- An efficient warping algorithm is presented to render the novel views from matched regions. This includes a compatible triangulation algorithm which can divide the matched regions into matched triangles. By working with triangles, the warping algorithm is able to improve its performance with hardware acceleration.

1.3 Related Work

This section will give a brief review of different methods for generating computer images. This will serve as a background for us to design an image based rendering system in the following chapters. Different methods have their own advantages and limitations. Depending on the requirements of the applications, some methods are preferred to the others. They can be classified into three main catergories as follows.

1.3.1 Geometric Based Rendering

Geometric based rendering is the traditional method to generate computer images. With the 3D models as input, it simulates how light rays are reflected on the surface of the 3D objects. The light rays will then go through the camera center and project onto an image plane. The color of the ray is recorded as a pixel in the final image. This process is supported by standard APIs like OpenGL and implemented in some dedicated graphics hardwares for real time rendering. There are various rendering techniques to increase the realism of the images [WAT92], such as texture mapping, ray tracing and radiosity.

Geometric based rendering requires 3D models as input. These models are usually created by human operators using CAD software. Modeling is always the most time consuming task in the whole process. It requires the users to specify the shape and the surface property of each 3D object, together with its position and orientation within the scene. It is common to take weeks or even months to create moderately complex environment.

1.3.2 Image Based Modeling

In order to get around the difficult modeling task, people start to search for alternatives. *Image based modeling* obtains 3D models from images. This has been studied as the major research topic in Computer Vision, in which the reconstructed 3D models are used for object recognition. Point correspondence between images are established by correlation, and the projective model is then reconstructed by finding the fundamental matrix [LUO96]. Five known scene points can be used as a projective basis to convert the projective model to an Euclidean model. Alternatively, with further information about the scene, such as knowledge of parallel lines and angles, we can reconstruct the Euclidean model from the projective model [FAU95].

Determining the correspondence is a difficult problem, and the correlation based correspondence is reliable only if the view positions are different by a small displacement. With user specified correspondence, however, 3D models can be recovered from widely spaced images. Instead of asking for pixel to pixel correspondence from the user, they are required to specify the correspondence by matching larger primitives, such as boxes and prisms [DEB96]. This can reduce the number of mouse clicks and the parameters to estimate in the reconstruction process. Another advantage of image based modeling is that the reconstructd 3D models come with textures from real images. This can make the final rendering more realistic.

Besides surface models, voxel coloring [SEI97] reconstruct a volume model by discretizing the scene space into voxels and color them iteratively according to the color in the reference images. The scene is reconstructed without computing the correspondence. The volume model is then converted to surface model for rendering.

1.3.3 Image Based Rendering

Another class of methods to render new views from existing images is *image based rendering*, in which 3D models are not explicitly recovered. It can be further classified into two approaches, *ray based approach* and *correspondence based approach*. Again, as the input is from images, this can avoid the modeling jobs. However, it may impose some limitations on the rendered view positions, as there may not be enough information from the reference images. On the other hand, because the images are generated from exisiting images, the result can be more photo-realistic.

Ray Based Approach

Each pixel in an image can be represented as a ray from the view position pointing to a particular direction. This ray is associated with a color value which is the reflectance of the scene point hit by that ray. In geometric based rendering, with the known 3D models of the scene, the image is generated by intersecting these rays with the models. Each time the viewing parameters are changed, the rays are computed to render the new image. Instead of computing the rays again and again, however, the ray based approach precomputes and stores and rays. New images are generated by retrieving and intergrating the stored rays.

The rays can be stored in a *panorama*, which captures rays originated from a view position. QuickTime VR [CHE95] is a commercial system which contains cylindrical maps at discrete view positions called nodes. The user is allowed to rotate at those nodes, and new image is rendered by reprojecting the panorama. It allows the virtual camera to zoom and rotate at the nodes. The file size is small, thus make it useful for Internet applications such as virtual site visiting. However, movement from one node to another is not handled well in this method.

We can consider the collection of rays in a scene as a *plenoptic function.* This function returns the color of the ray given its view position and direction,

$$color = plenoptic(C_x, C_y, C_z, \theta, \phi)$$

where θ and ϕ are the azimuth and elevation angles respectively which determine the view direction, and (C_x, C_y, C_z) is the view position. By generalizing the idea of panorama, lumigraph [GOR96] and light field rendering [LEV96] sample the plenoptic function in a range of view positions and reduce the dimension of the function to four with a better indexing method. Novel images from different view positions can be generated by combining different samples of the plenoptic function. The major drawback of these methods is that it requires a huge amount of storage space to store the function. In addition, the input is a set of closely sampled images, which may be difficult to acquire in some situations.

Concentric mosaic [SHU99] tries to further reduce the plenoptic function to a 3D function by restricting the view position within a planar circle. However, the novel views are distorted in the vertical direction. It is due to the lack of depth values of the pixels. Although it has significantly reduce the storage space, it did not properly address the depth problem. This can be thought as an approximate ray based method.

Correspondence Based Approach

With the idea of plenoptic function, the problem of image based rendering can be defined as [MCM95b],

> *Given a set of discrete samples (complete or incomplete) from the plenoptic function, the goal of image based rendering is to generate a continuous representation of that function.*

Hence, this allows us to acquire and store less reference images than those ray based methods. The problem is now changed to how to recover the plenoptic function.

Traditional image morphing [BEI92] interpolates feature points between two images in the hope of generating the in-between views. However, these in-between views are not shape-preserving, because it did not take the camera transformation into account. View Interpolation [CHE93] can give physically valid views if the two given views are both parallel to the line joining the centres of projection. View Morphing [SEI96] generalizes this idea by first transforming the given images to this special configuration. Plenoptic Modeling [MCM95b] develops a complete image based rendering system which works in cylindrical maps. It is further modified [FU98] to support triangle based rendering, therefore to improve the performance by hardware acceleration. Another image based system is developed for virtual reality application [CHA99]. It also takes in reference images from cylindrical panoramas, which are triangulated into patches for efficient rendering after the correspondence is derived.

As we are transferring the reference images to the novel image, we need to determine where the image points will go to. The above mentioned methods are distinguished by how they compute this mapping. A common property of these methods is that they all require the *correspondence information* as one of the inputs. This information is crucial in determining the final position of the image point, and hence they are classified as correspondence based.

Recall that in image based modeling, if the correspondence is given, we can reconstruct a projective model. Instead of explicitly recover the Euclidean model, the projective model can be rendered as a new view if a projective basis can be selected [CHE97] or the camera information is known [LAV94].

A related problem in correspondence based rendering is the fold problem which is due to the change of visibility between the reference images and the novel image. Folds can be solved by comparing the disparities of the points [SEI96], or by drawing the new image in a back-to-front order [MCM95c, FU99].

The main problem to solve in this correspondence based approach is how to compute the correspondence information of the reference images. Although this method can greatly reduce the storage requirement, the quality of the resultant images will degrade accordingly if the correspondence is not reliable. In addition, there are always limitations to the view positions of the novel images.

1.4 Thesis Organization

Chapter 2 will define the problem we are trying to solve, and describe how we can reduce it to a simpler problem — panorama interpolation. This problem is further divided into three sub-problems. Chapter 3 talks about how to align the reference panoramas, so that they can be used for correspondence matching in Chapter 4. Chapter 5 will describe how we can generate novel views by warping the reference panoramas. Chapter 6 will compare our work with other methods, and present some experimental results. Chapter 7 concludes the thesis with applications and future works.

Chapter 2

Problem Statement

This chapter will first state the goal of this thesis and explain how to reduce it to a simpler problem. Then we will define the scope of the thesis. This is followed by a brief introduction of the correspondence information, which is vital to this problem. Lastly, we will describe how to solve the problem by breaking it down into several sub-problems.

2.1 Goal of this Thesis

As we discussed in the previous chapter, the main advantage of image based rendering over geometry based rendering is that it can produce photo-realistic images without difficult modeling jobs. However, it may impose certain limitation on the viewer's motion, and possibly requires enormous storage space. *The goal of this work is to develop a framework for image based rendering which allows the user to move freely within the scene with modest storage requirement.* Hence, we want to combine the advantages of both image based and geometry based computer graphics.

2.2 Reduction to a Simpler Problem

Instead of allowing the viewer to move freely within the scene, we will first ask a simpler question: Is it possible to reconstruct the views along a line segment, if we are given the panoramas taken at the two end points of the line segment? We define this problem as *panorama interpolation.* Given two panoramas which are taken from view positions C_1 and C_2 respectively, panorama interpolation computes the panorama with view position located along the line segment $\overline{C_1C_2}$. Figure 2.1 illustrates the meaning of panorama interpolation. A panorama is represented by a sphere, in which a point on the surface represents a ray from the view position to that point. Every point on the sphere is associated with a color value, which is the color of the corresponding ray. The green spheres located at C_1 and C_2 are the given panoramas. By panorama interpolation, we are going to recover the panorama along $\overline{C_1C_2}$, which is represented by a blue sphere.

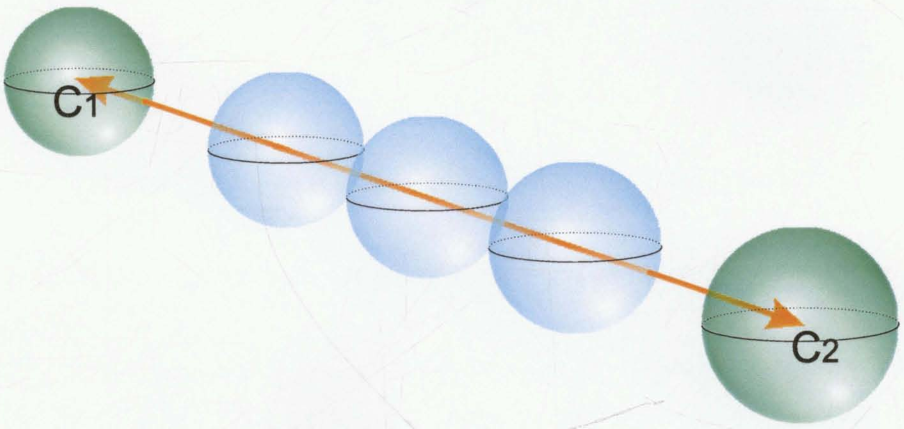

Figure 2.1: Panorama Interpolation

If panorama interpolation is possible, then by generalizing this idea, we can achieve the goal of allowing the viewer to move freely within the scene, provided that we are given enough reference panoramas.

In Figure 2.2, again the green spheres are the given panoramas. If the panoramas at view positions C_1, C_2 and C_3 are given and the three view positions are not collinear, then we can compute any panorama with view position located within $\triangle C_1C_2C_3$. Assume we want to reconstruct the panorama at D located within $\triangle C_1C_2C_3$, we first reconstruct the panorama at B, which is the intersection of $\overline{C_1C_2}$ and $\overline{C_3D}$, by interpolating the panoramas at C_1 and C_2. Then the interpolation can be applied to panoramas at B and C_3 to reconstruct the panorama at D.

Similarly, if we are given four panoramas at C_1, C_2, C_3 and C_4, which are not coplanar and no three of them are collinear, we can perform panorama interpolation three times and obtain any panorama with view position located within the tetrahedron $C_1C_2C_3C_4$. Hence, with N reference panoramas, we can partition the space into tetrahedrons and by using these tetrahedrons as building blocks,

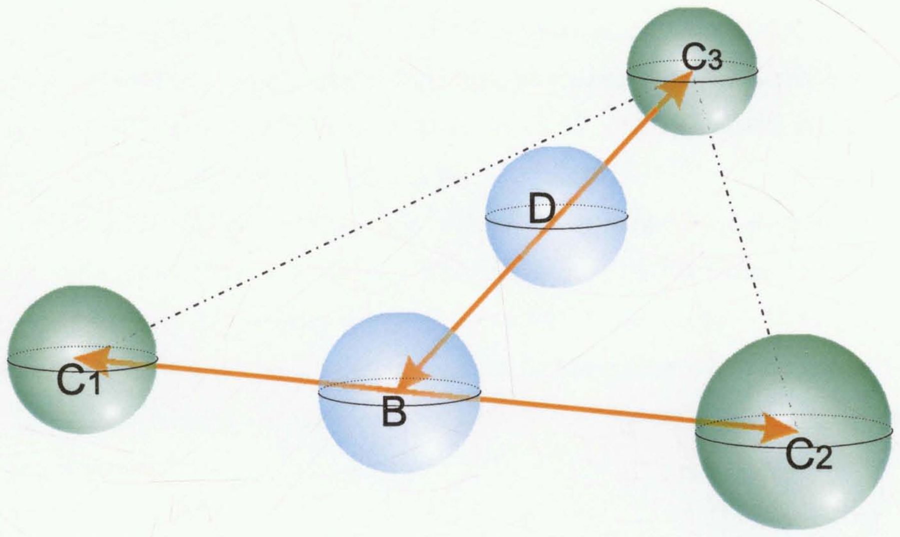

Figure 2.2: Generalization of Panorama Interpolation

we can reconstruct any panorama located within the convex hull of these N view positions.

Therefore, we have reduced a difficult problem — allowing the viewer to move and look freely within the scene, to a simpler problem — panorama interpolation.

2.3 Problem Statement

We will define the scope of this thesis in this section. Given two panoramas taken from two view positions C_1 and C_2, we want to reconstruct the panorama located along the line segment $\overline{C_1C_2}$. The given panoramas at C_1 and C_2 are called the *reference panoramas*, and the reconstructed one is called the *novel panorama*. In order to reconstruct the novel panorama, we also require the correspondence information between the reference panoramas. We will explain how to derive this information in the upcoming chapters.

There are two assumptions about the scene in which the reference panoramas are taken from. The objects of the scene are assumed to be made up of Lambertian surfaces, which means that light will reflect from the surface equally in all direction independent of the view direction [FOL92]. This assumption is important to our correspondence matching algorithm, as one of the component of our matching score is the color component. If the reflection on a surface is not Lambertian, the projections on different images will have different colors. This happens when the surface is a metallic surface, which causes specular reflection.

The final rendering will be incorrect if the correspondence matching is inaccurate. There are other image based rendering techniques which consider the effect of different lighting conidtions to the scene [SAT97, YU98], however in this work, we assume there is only diffuse reflection.

The other assumption is that the scene is required to be static. There is no moving object within the scene. Therefore the projections of the same 3D element can be used to recover the information of the scene. There are studies which attempt to render new views from reference images which are taken at different times [MAN98]. Moving objects in the scene are needed to be identified for correct warping.

2.4 Correspondence Information

It is not enough for us to perform panorama information, if only the reference panoramas are given. We also require some extra information of the scene, such as the depth information, the correspondence information and the camera information. However, these information are not independent to each other. That is, one information can be derived from the others. This section will talk about the correspondence information and explain why we choose to base on this information.

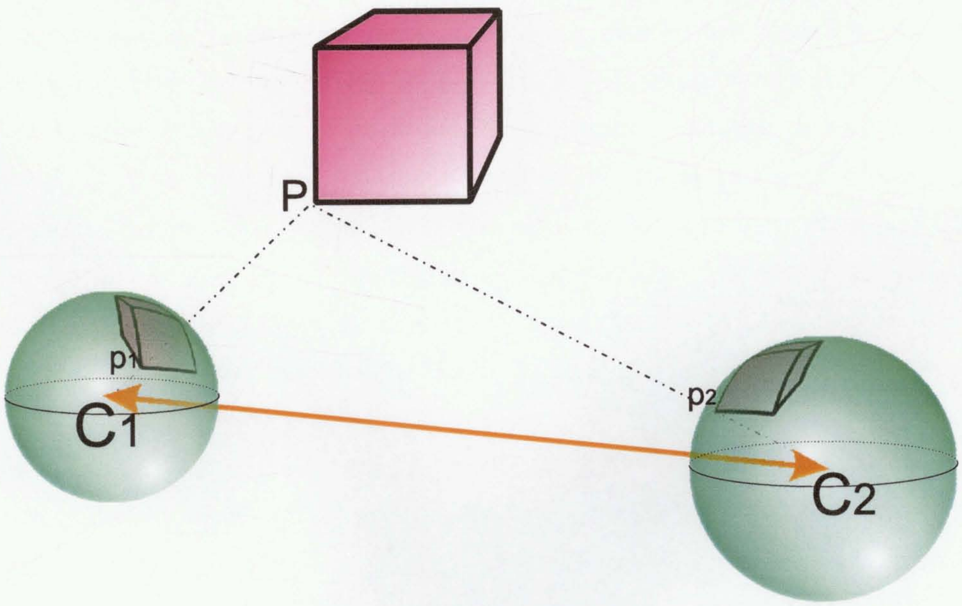

Figure 2.3: Correspondence Information

In Figure 2.3, a scene point P is projected onto two reference panoramas I_1 at C_1 and I_2 at C_2. This produces two image points p_1 and p_2. This pair of image points (p_1, p_2) is called a corresponding point pair, because they are projections of the same physical point P. We define the *correspondence information* as a mapping M_{12} from I_1 to I_2 such that $M_{12}(p_1) = p_2$ if (p_1, p_2) is a corresponding point pair.

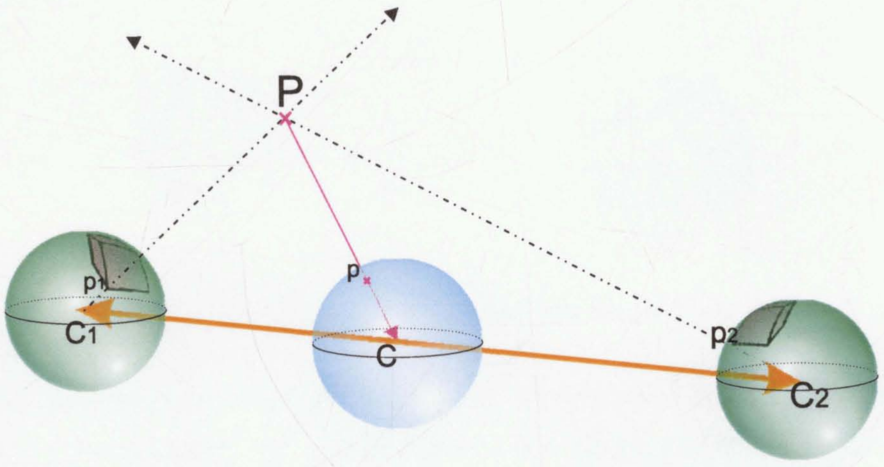

Figure 2.4: Computing the Novel Panorama from Correspondence

It is easy to understand that if the correspondence information is given and we know the positions of the reference panoramas, then we can compute the novel panorama, as illustrated in Figure 2.4. p_1 and p_2 are image points on the panoramas I_1 and I_2 respectively, and from the correspondence information, we know that they are a pair of corresponding points. Assume p_1 and p_2 are the image points of an unknown point P in the scene. As p_1 is a projection of P, P should lie on the ray $\overrightarrow{C_1p_1}$. Similarly, P is also on the ray $\overrightarrow{C_2p_2}$. Hence, P can be located as the intersection of the two rays $\overrightarrow{C_1p_1}$ and $\overrightarrow{C_2p_2}$. In order to reconstruct the novel panorama at view position C, P is projected to give the image point p. We can repeat this process for all corresponding point pairs and reconstruct the whole panorama at C.

Besides using the correspondence information, we can reconstruct the novel panorama from a single reference panorama if the depth information is available. In Figure 2.5, the panorama I_1 at C_1 is the reference panorama, in which every pixel in I_1 is associated with a color value and also a depth value. The depth value of p_1 is defined as the distance from C_1 to the scene point P of p_1. Therefore with this depth information, we know the 3D position of P relative to the view position and it can be projected to give the image point p of the novel panorama I at C. Again we can repeat this process for all points in I_1 to reconstruct I.

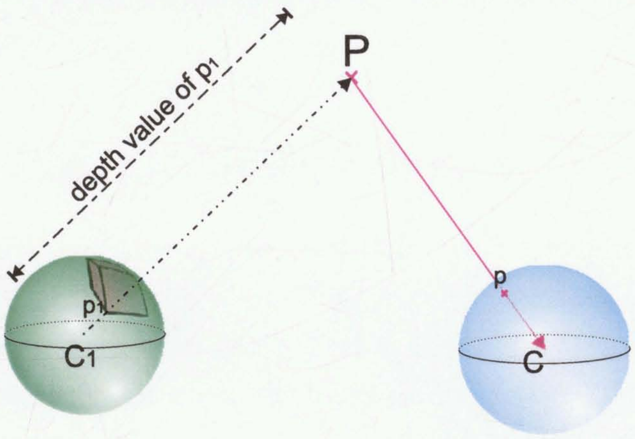

Figure 2.5: Computing the Novel Panorama from Depth

We can see that the novel panorama can be recovered from either the correspondence information or the depth information, if we know where the camera is. Actually, the relationships among the correspondence information, the depth information and the camera information is that if we know any two of them, the third one can be derived. According to Figure 2.4, if we know the camera information and the correspondence information, then by intersecting the rays of the corresponding point pair (p_1, p_2), we can recover the 3D position of the scene point P. Thus, the depth values of p_1 and p_2 can be computed directly. In Figure 2.5, if the depth information is known instead of the correspondence information, then the corresponding point of p_1 in I can be found by projecting the scene point P onto I. Therefore the correspondence information can be derived from the depth information.

Different image based rendering techniques will require different information. Some assume the correspondence information is known [MCM95b], and some assume the depth information [SHA98, CHA98]. The practicability and usefulness of a method is determined by the choice of information to be known. It is because that some information may be more difficult to acquire. This is the major reason we choose to use the correspondence information instead of using the depth information. The depth information can be acquired by using expensive hardware such as range finders. On the other hand, the correspondence information is derived from matching the corresponding point pairs of the reference panoramas, therefore ordinary camera is capable of capturing the scene. Both of them are information of the scene but only the correspondence information can be computed directly from the reference panoramas alone. Chapter 4 will describe in details an automatic approach to compute the correspondence information of two reference panoramas.

2.5 Splitting into Sub-problems

The whole problem of panorama interpolation can be divided into three sub-problems. We will discuss each sub-problem and the solution in the subsequent chapters one by one. Figure 2.6 give a brief overview of this framework.

The next chapter will talk about the input to the system, that is, the panorama. Panorama can be created from several planar images taken from the same view position with different view directions. There are well studied methods to stitch these images into a single panorama. After we have two reference panoramas, we need to establish the relationship between them. That means we have to find out the relative orientation between the reference panoramas. This procedure is related to finding the camera information and is very important to the next sub-problem, computing the correspondence. This procedure is called panorama alignment and will be the main subject of the chapter.

After we have two aligned reference panoramas, Chapter 4 will describe an automatic approach to obtain the correspondence information between them. Instead of matching the point pairs, our method will match the regions between the reference panoramas. Therefore we need to segment the panoramas into regions before performing the matching. A region is a set of connected points in an image which are homogenous in some sense.

Finally in Chapter 5, we will talk about how to generate the novel views from the reference panoramas, with the correspondence information derived from Chapter 4. Because we want our implementation to be supported by hardware accelerators, the matched regions are first divided into matched triangles. We will then present an equation to warp the matched triangles to the novel panorama for generating the new images.

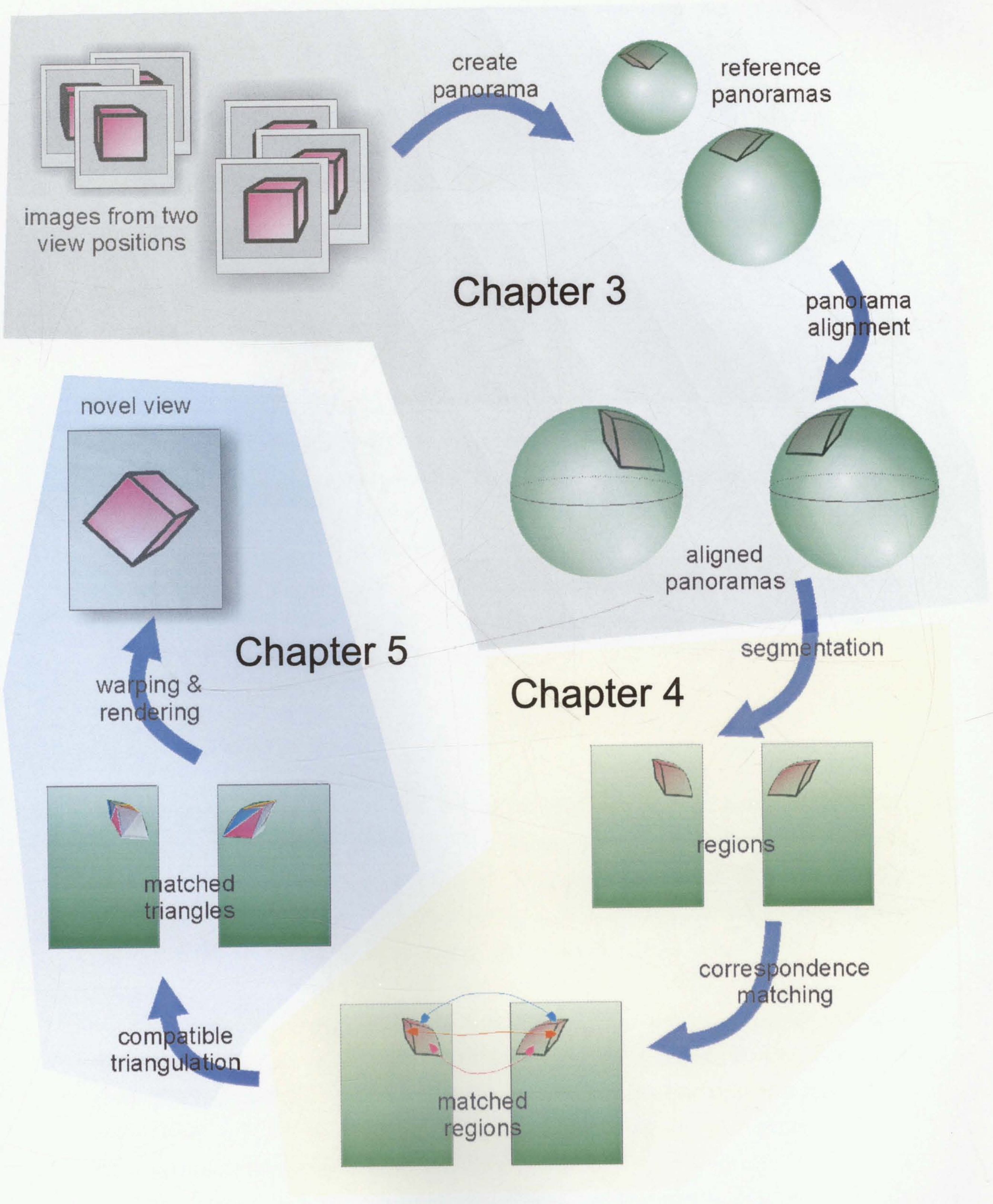

Figure 2.6: Overview of Framework

Chapter 3

Panorama Alignment

This chapter explains how to align two reference panoramas which tries to compute the relative orientation between them. This serves as a preparation step for the next sub-problem, matching correspondence, as described in the next chapter. We will first take a look at the planar image and then the panorama, and study how to relate the three-dimensional points with the image points. The main subject of this chapter is to establish the relationship between two reference panoramas. We will also explain why panorama is chosen as the reference image, instead of using planar image.

3.1 Planar Image

An *image* can be considered as a collection of image points. Each image point, which is indexed by its position in the image space, is associated with certain information about the point. For a color image, the associated information is the color value of that point. For other images, it may include other information, such as a depth value for a range image, or an intensity value for a gray scale image.

The image is usually stored in digital form, and the image space contains discrete samples of image point. A digital image is a two dimensional array of these discrete image points, usually referred as pixels. Each pixel is assigned a color value in RGB form, where R, G and B represent the red, green and blue color components respectively.

Planar image is what we usually see from camera and television image. It is formed by a perspective projection of the scene to an image plane. We will spend a few paragraphs to study the relationship between the scene points and the image points.

3.1.1 Pin-Hole Camera Model

Consider the camera located at view position C in Figure 3.1. Points in the scene are projected onto the image plane Π through the view position C. In the figure, the 3D point P is projected to an image point p which is located by finding the intersection of the ray $\overrightarrow{CP}$ and the plane Π. The image is formed in the image plane Π by projecting every 3D point through the view position C. This kind of camera is known as a *pin-hole camera*.

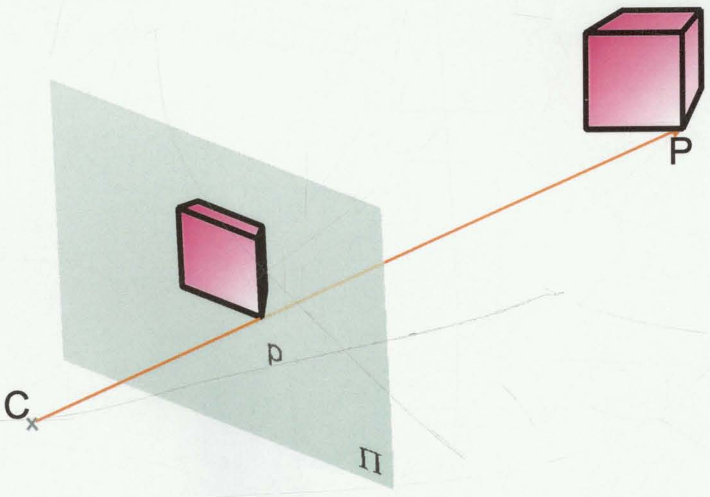

Figure 3.1: Pin-Hole Camera Model

3.1.2 Projection Matrix

In the pin-hole camera model, the positions of a 3D point P and its image point p are related by a perspective projection with the center of projection at the view position C. The camera configuration is defined by the view position C and the image plane Π. To study their relationship, we have to define two coordinate systems. All 3D points are specified in the *world coordinate system*, which has an origin and three mutually perpendicular axes X, Y and Z. A 3D point P is represented by its projective coordinates $P = [x, y, z, w]$ where $[x/w, y/w, z/w]$ represents the point in the world coordinate system. Another coordinate system is the *image coordinate system* which is defined in the image plane Π. The image coordinate system is defined by an origin and two perpendicular axes U and V

in the image space. Each image point p is represented by projective coordinates $p = [u, v, t]$ where $[u/t, v/t]$ gives the image coordinates.

Let us look at the camera configuration shown in Figure 3.2. The view position is located at the origin of the world coordinate system and the image plane has the equation $z = -1$. The line going through the view position and perpendicular to the image plane is known as the optical axis and its intersection with the image plane is the optical center. In this case, the origin of the image coordinate system is at this optical center and the U and V axes have the same direction as the X and Y axes in the world coordinate system respectively. This camera setup is known as the *standard coordinate system* and this will give a very simple relationship between a scene point P and its projection p in the image plane. They are related by a 3×4 projection matrix M,

$$p = MP \tag{3.1}$$

where

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{3.2}$$

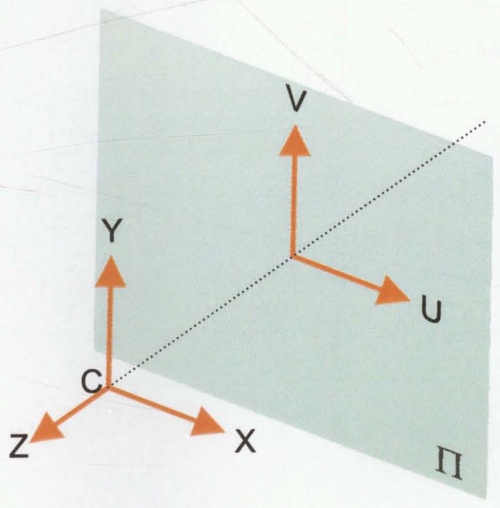

Figure 3.2: Standard Coordinate System

3.1.3 Intrinsic and Extrinsic Parameters

If the camera configuration is different from that of the standard coordinate system, the projection matrix M will not have the simple form as in equation 3.2. However, we can still use a projection matrix to relate P with p.

The image coordinate system can be changed by moving the origin to a position other than the optical center and applying different scales to the axes U and V. This is equivalent to applying an affine transformation to the image points in the original image coordinates system. This transformation can be represented by a 3×3 matrix H and hence equation 3.1 becomes,

$$p' = Hp = HMP \tag{3.3}$$

where p' is the new image point. Therefore the new projection matrix is HM.

On the other hand, we can move the view position C and also rotate the view direction such the view direction is no longer the same as the negative Z axis. This is equivalent to thinking the coordinate system as remaining intact, but moving every scene points with an inverse transformation. This inverse transformation can be represented by a 4×4 matrix K, and equation 3.3 becomes,

$$p = HMKP \tag{3.4}$$

And the new projection matrix is HMK.

We can see from equation 3.4 that the projection of a scene point to an image point can be defined by a projection matrix. The projection matrix can be decomposed into three matrices H, M and K. In Figure 3.1, a scene point P is projected to an image point p. Consider a ray $\overrightarrow{CP}$ from view position C to point P. The matrix H specify how we map the ray $\overrightarrow{CP}$ to the image point p. The mapping is different with different H. The parameters embedded in H are called intrinsic parameters. They include the position of the optical center and the scales of the two axes in the image space. On the other hand, the matrix K relates the position of P with the ray $\overrightarrow{CP}$. The parameters are called extrinsic parameters and they are governed by the view position and the view direction in the 3D space. Calibrating a camera means that we want to estimate these parameters of the projection matrix.

3.2 Panorama

Panorama is another type of image. In contrast to the planar image, panorama has a larger 'field of view'. This section will give the definition of panorama, and describe how it can be obtained from planar images.

3.2.1 Plenoptic Function

As defined in Section 1.3.3, we can describe all the visual information within an environment with the plenoptic function. It returns the color of a ray when we are looking from a view position C at a particular direction. We can write the function as $plenoptic(C_x, C_y, C_z, \theta, \phi)$ where $[C_x, C_y, C_z]$ is the view position and the view direction is specified by the angles θ and ϕ. Each sample of the plenoptic function can be considered as a ray originated from the view position pointing to the view direction. The value of this sample is the color of the scene point which is first hit by this ray. Figure 3.3 shows a sample of the plenoptic function at $(C_x, C_y, C_z, \theta, \phi)$.

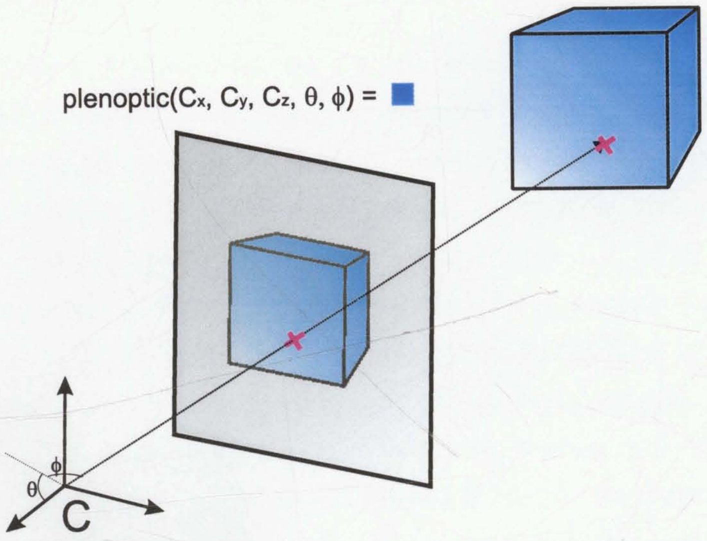

Figure 3.3: Plenoptic Function

An image taken from a view position can be considered as a set of samples of the plenoptic function with some appropriate parameters. For example, in Figure 3.4, by choosing a suitable range for the two viewing angles θ and ϕ, we can sample the plenoptic function at the view position C and generate the image as it is taken from C.

A panorama taken from view positon $C = [C_x, C_y, C_z]$ is defined as the set of samples of the plenoptic function $f(C_x, C_y, C_z, \theta, \phi)$ with θ and ϕ covering all possible view directions. We say that a panorama is a *complete sample* of the plenoptic function, while a planar image is a *partial sample* because a planar image does not include rays to all view directions from a single view position.

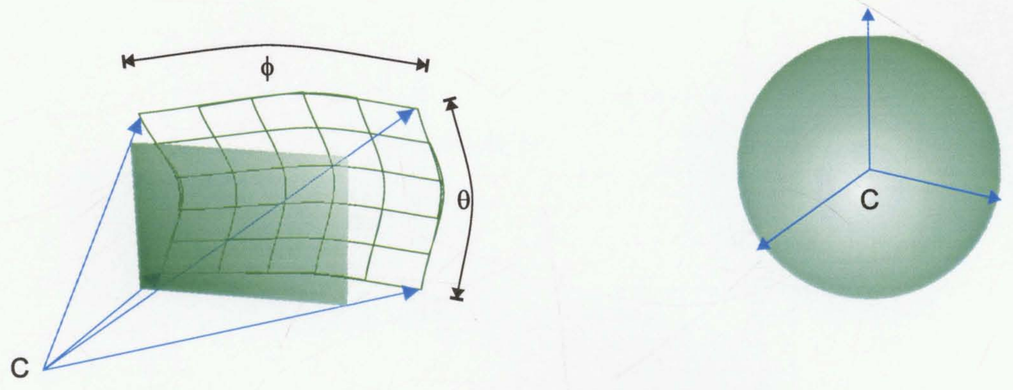

Figure 3.4: Planar Image and Panorama

3.2.2 Spherical Coordinate System

When we are taking a panorama, we are still using the pin-hole camera model. Instead of using an image plane, however, we need to use a different geometry to model the image space in order to capture more rays than the planar image.

There are generally two types of panorama, *cylindrical* and *spherical.* Consider a cylinder surrounding a view position in Figure 3.5. The surface of the cylinder can be used to store the color values of the projections of the scene points. Strictly speaking, it is not a complete sample of the plenoptic function as the rays going towards the top and the bottom are missing in this cylindrical representation. If we unfold the cylinder, it can be represented by a rectangular block of pixels. We can build a coordinate system in this rectangular image with an origin and two perpendicular axes, just as any two dimensional Euclidean space. It is easy to store the panorama in computer format, and the sampling density is more or less the same.

Spherical panorama, on the other hand, is a complete sample of the plenoptic function. It is also known as environment map, because it can capture all visual information of the environment from a single view position. With the unit sphere centered at the view position in Figure 3.5, each point on the surface of the sphere corresponds to a ray originated from the view position. Figure 3.6 illustrates the *spherical coordinate system*, which defines the correspondence between a ray and two angles θ and ϕ. In Figure 3.6, the view position is located at the origin of the coordinate system with three mutually perpendicular axes X, Y and Z. The points $[1, 0, 0]$ and $[-1, 0, 0]$ are called the poles of the sphere. A point p on the sphere can be represented by its coordinates $[x, y, z]$ with $x^2 + y^2 + z^2 = 1$, or can be parameterized by two angles θ and ϕ. Consider the plane Π_p which contains

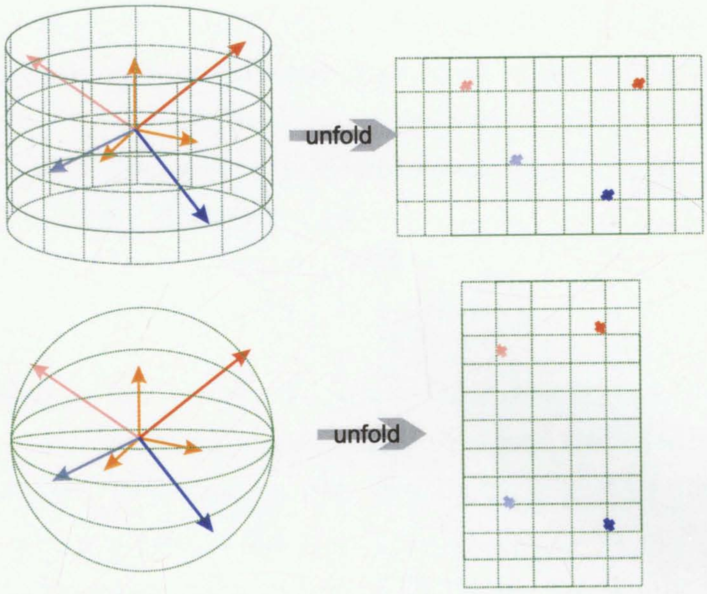

Figure 3.5: Cylindrical and Spherical Panorama

p and the poles. θ is defined as the angle between the plane Π_p and the plane $y = 0$. We will take the direction towards the Y axis as positive, thus θ ranges from $-\pi$ to π. z_p is a point in Π_p which is the point $[0, 0, 1]$ after a rotation about the X axis by θ. ϕ is defined as the angle between the rays $\overrightarrow{Cz_p}$ and $\overrightarrow{Cp}$. We take the direction towards the X axis as positive, so ϕ ranges from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$.

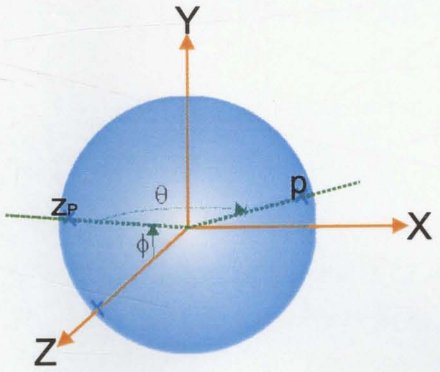

Figure 3.6: Spherical Coordinate System

If the surface of the sphere is unfolded, we can obtain a rectangular block of image points indexed by the spherical coordinates $[\theta, \phi]$. A drawback of this representation of the spherical panorama is that the sampling density is not even. Note that the regions around the poles are seriously over-sampled.

Recall that in equation 3.3, the matrix H defines the mapping of the rays originated from the view position to the image points in the image space of the planar image. Here we want to study how these rays are related to the image points on the sphere. As mentioned before, each ray can be represented by a point on the sphere with coordinates $[x, y, z]$ such that $x^2 + y^2 + z^2 = 1$. Therefore the relationship is,

$$\begin{cases} \tan\theta &= \frac{y}{z} \\ \tan\phi &= \frac{x}{\sqrt{y^2 + z^2}} \end{cases} \tag{3.5}$$

Note that the surface of the unit sphere is only an abstract representation. Actually, any geometry which can completely enclose the view position is good enough to represent the panorama. An alternative is to store the rays onto the six faces of a cube centered at the view position, as shown in Figure 3.7. Each ray will map to an image point of one of the faces. Again, the regions around the corners and the edges are over-sampled.

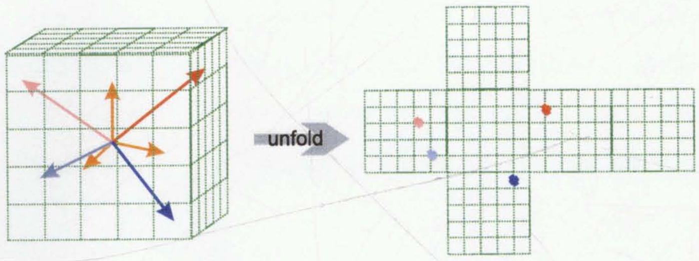

Figure 3.7: Cube Representation

3.2.3 Creating Panorama

A panorama collects rays originated from a view position. Therefore, to generate a panorama, we require a set of images taken from the same view position. In practice, these images are taken with ordinary camera mounting on a tripod, thus ensuring the view position is fixed while changing view direction. The result is a set of planar images such that each image has some overlapping parts with the neighboring images, and the whole set of images can cover all rays from the view position.

The overlapping part is important for matching the image with its neighbors, hence to recover the relationships among the images. This matching process is known as *image registration* [BRO92]. Note that because all images are taken from the same view position, each image is different from the other by a 3D rotation and a 2D affine transformation if the camera is zooming. Therefore each image is associated with a projective transformation. Image registration can recover the relative transformation between a pair of overlapping images. This is done by choosing the transformation with maximum correlation of the overlapping part. By using these transformations, the images can be transformed properly and stitched together to form a panorama.

By panning the camera horizontally about a tripod, we have a sequence of overlapping images. Image registration can be performed by estimating only the horizontal panning angle [MCM95a] and a cylindrical panorama is obtained. However, you cannot view the top nor the bottom of the panorama. In order to construct a spherical panorama, we have to estimate eight parameters in the general transformation matrix for each image. With some assumptions on the camera, we can reduce the number of parameters to estimate to three [SZE97] and create a full panorama from a set of images with free rotation.

Besides, a panorama can be created if a few pair of corresponding points are identified between each pair of images, hence the relative transformation can be found. Also, we can use a fisheye lens which simplifies the registration process by reducing the number of images due to its large field of view [XIO97] .

3.3 Epipolar Geometry

This chapter is about aligning two reference panoramas. Aligning two panoramas will cause the corresponding image points appear on the same line, which will make it easier to match the corresponding points. This constraint on the corresponding points is known as the *epipolar constraint*. In this section, we will explain this constraint in the case of planar image. Next section will talk about how this constraint can be applied to panoramas.

3.3.1 Epipolar Constraint

Figure 3.8 shows two cameras located at view positions C_1 and C_2. A scene point P is projected to images I_1 and I_2 to form image points p_1 and p_2 respectively. An image point in the first image and an image point in the second image are corresponding points if they are projections of the same scene point. In this case, p_1 and p_2 are corresponding points.

For a given image point p_1 in I_1, computing the correspondence means that

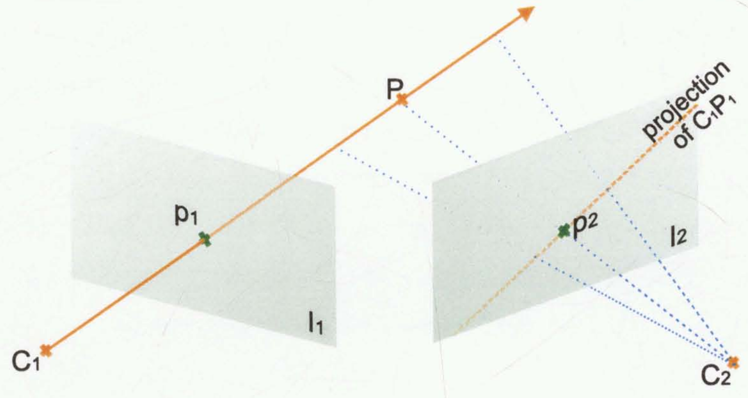

Figure 3.8: Epipolar Constraint

we want to find the corresponding point of p_1 in I_2. Instead of searching the whole image for the possible correspondence, we observe the following hint. If the position of P is unknown a priori, given only the image point p_1, we can predict that P can be any point on the infinite ray $\overrightarrow{C_1p_1}$. By projecting this ray to I_2, a line is obtained, which restricts the possible position of the corresponding point p_2. This constraint on the possible position of the corresponding point is known as the epipolar constraint.

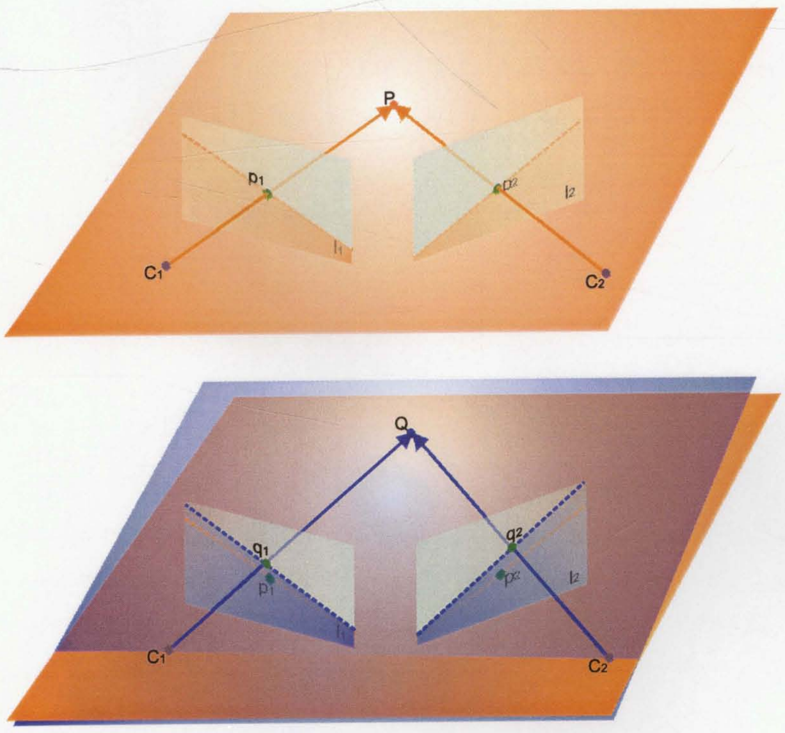

Figure 3.9: Epipolar Geometry

Figure 3.9 illustrates the epipolar geometry between two cameras. In the upper figure, the orange plane containing the scene point P and two view positions C_1 and C_2 is the epipolar plane Π_P defined by P. The intersection of Π_P and the second image plane I_2 is the epipolar line l_{p_2} of image point p_1 in the second image. This epipolar line is identical to the projection of the ray $\overrightarrow{C_1p_1}$ onto I_2, which limits the possible position of the corresponding point p_2. Similarly, the epipolar line l_{p_1} of point p_2 is the intersection of Π_P and the first image plane I_1. These two epipolar lines are called *conjugate epipolar lines.*

The projection of C_2 onto I_1 is called the *epipole* e_1 of I_2 in I_1. Similarly, epipole e_2 is the projection of C_1 onto I_2. Consider another scene point Q in the lower figure. The epipolar plane Π_Q, in blue color, defines two epipolar lines l_{q_1} and l_{q_2} in images I_1 and I_2 respectively. Note that l_{p_1} and l_{q_1} intersect at e_1. This is because every epipolar plane must contain view positions C_1 and C_2 and this forms a pencil of epipolar planes, with $\overline{C_1C_2}$ as their common intersection. When the planes are projected onto I_1, they form a pencil of epipolar lines going through the common intersection e_1. Therefore the epipole is the common intersection of all epipolar lines within the image.

One useful property of the epipolar geometry is that it ensures the view positions, corresponding image points and the relavent scene point to be coplanar, thus can be analyzed in 2D. From now on, if appropriate, the figures will be drawn in 2D corresponding to the appropriate epipolar plane.

3.3.2 Camera Rectification

When we are searching for the corresponding point of an image point in the first image, epipolar constraint limits the possible position within the conjugate epipolar line, hence it reduces the search space from two dimensional to one dimensional. It will be more convenient for the matching process if we can transform the images such that conjugate epipolar lines can be found in the same horizontal scanline. Two images satisfying this condition are said to be *rectified.* Figure 3.10 shows that two images are rectified if they have a common image plane which is parallel to the baseline joining the view positions. If they do not have such configuration, we can re-project the images to a new image plane which satisfies this requirement [SEI96]. If the images are rectified, all epipolar lines are parallel to each other, and the epipoles are located at infinity.

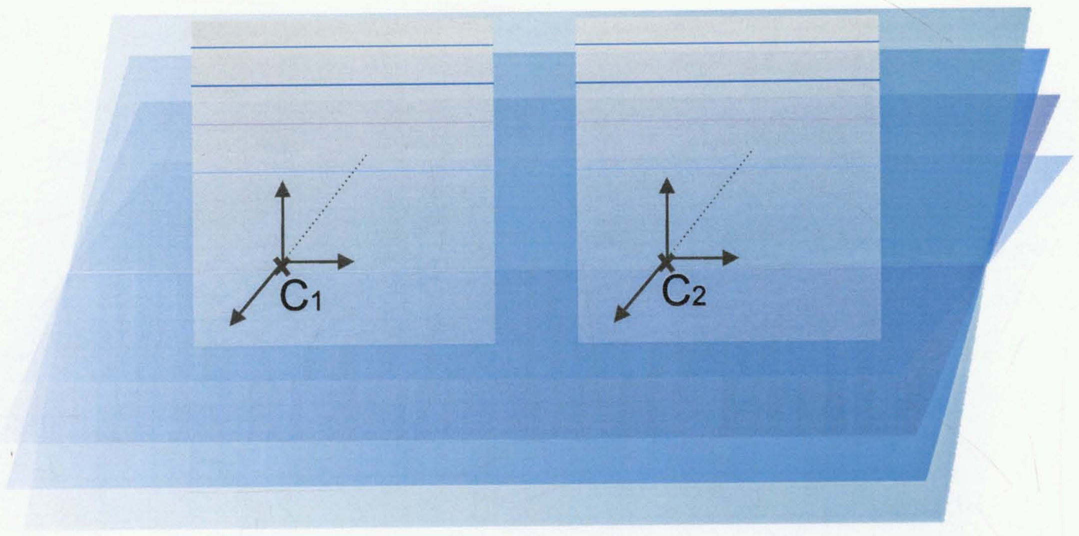

Figure 3.10: Rectified Cameras

3.4 Panorama Alignment

This section describes how to align two panoramas. The reason to align the panoramas is the same as that of rectifying the planar images. That is, we want to force the conjugate epipolar lines to be appeared on the same scanline, so that the matching process can be performed easily.

3.4.1 Epipolar Constraint in Panorama

With the spherical coordinate system, an epipolar line is the intersection of the unit sphere and the epipolar plane. Because the epipolar plane passes through the center of the sphere, the epipolar line is a great circle. In general, this great circle is not a horizontal line in the unfolded θ-ϕ space because the world coordinate systems for the two cameras have different orientations. This can be done by rotating the coordinate systems, such that the X axes are parallel to the line connecting the view positions C_1 and C_2 and the Z axes are pointing to the same direction. With this configuration, the epipole will be appeared at the pole of the sphere and conjugate epipolar lines will be parallel and have the same value of θ.

In order to compute these rotations, we have to study the relationship between the two coordinate systems. This is achieved by using the essential matrix.

3.4.2 Essential Matrix

In the spherical coordinate system, each image point $[\theta, \phi]$ can be mapped to a point $[x, y, z]$ on the surface of an unit sphere. In Figure 3.11, let $p_1 = [x_1, y_1, z_1]$ be the position of an image point in the first coordinate system and $p_2 = [x_2, y_2, z_2]$ be the position of the corresponding image point in the second coordinate system. The different between the two coordinate systems can be specified as a rotation R followed by a translation t where $t = [t_x, t_y, t_z]$ is the displacement $\overrightarrow{C_1C_2}$ in the first coordinate system. The corresponding point p_2 is then represented by Rp_2 in the first coordinate system.

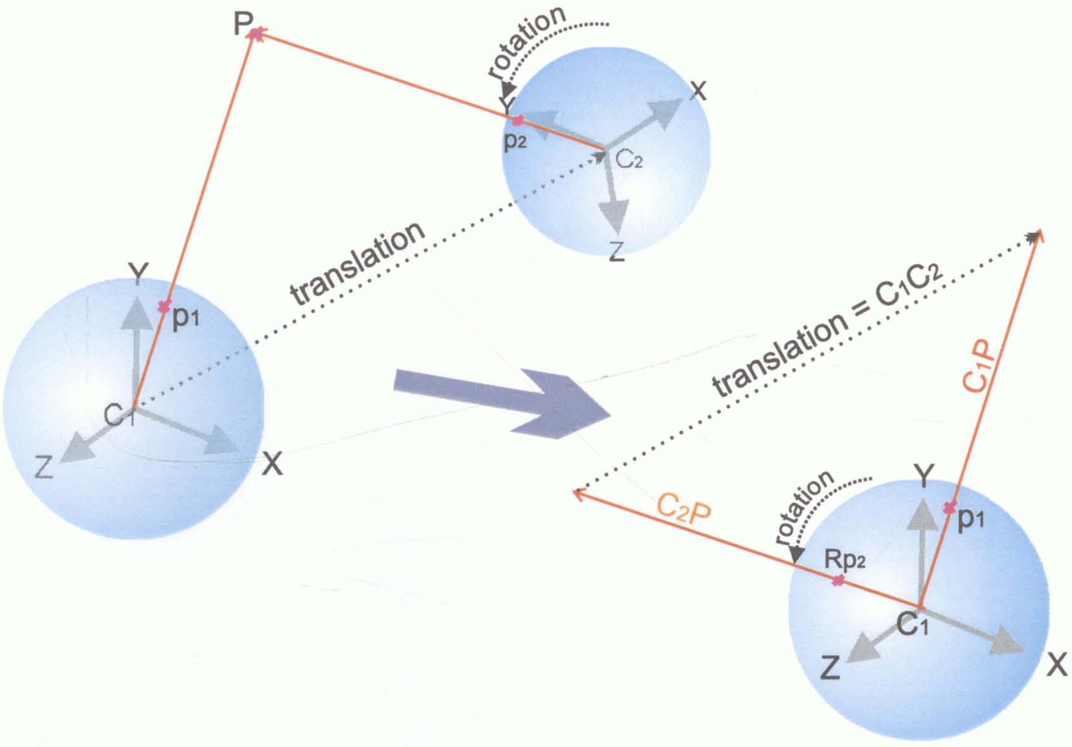

Figure 3.11: Two Coordinate Systems Related by a Rotation and a Translation

Observe that if p_1 and p_2 are corresponding points, the three vectors $\overrightarrow{C_1P}$, $\overrightarrow{C_1C_2}$, and $\overrightarrow{C_2P}$ must be coplanar. When these vectors are represented in the first coordinate system, it is equivalent to saying that p_1, t and Rp_2 are coplanar, that is,

$$p_1 \cdot (t \wedge Rp_2) = 0 \tag{3.6}$$

Let T be the antisymmetric matrix of t which has a property that $Tx = t \wedge x$

for all x. T is given by,

$$T = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}$$

Then equation 3.6 can be written as,

$$p_1^T T R p_2 = 0 \tag{3.7}$$

The matrix $E = TR$ is defined as the *essential matrix*, which relates the corresponding image points in the two coordinate systems.

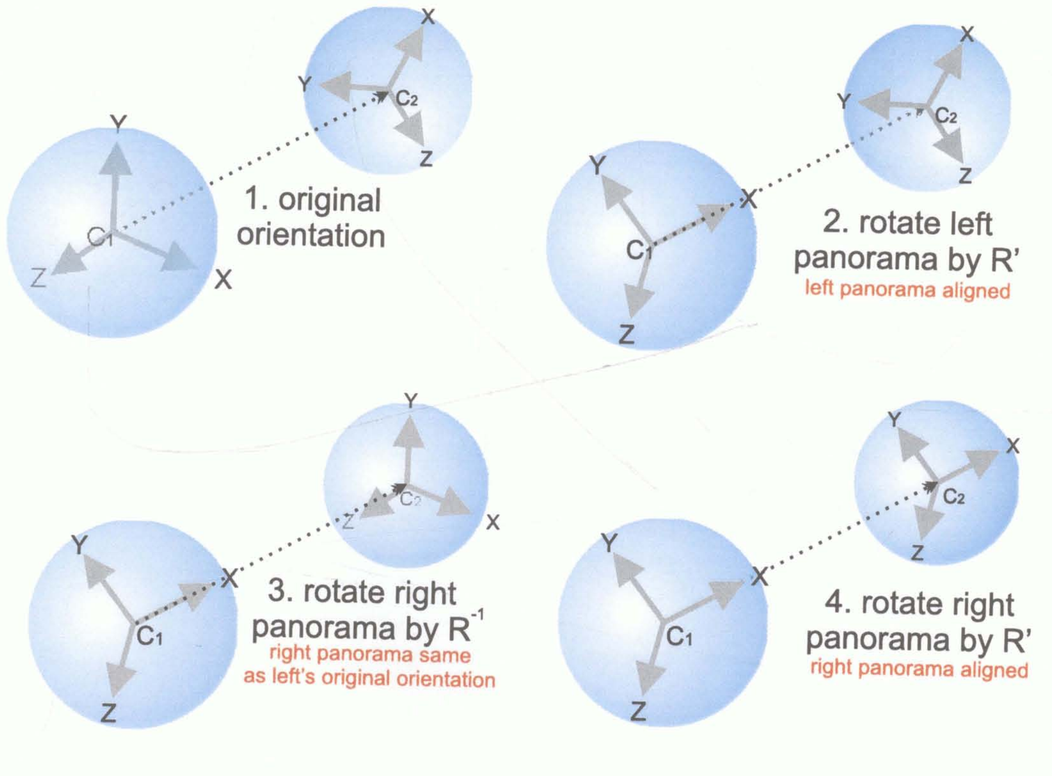
1. original orientation
2. rotate left panorama by R'
left panorama aligned
3. rotate right panorama by R^{-1}
right panorama same as left's original orientation
4. rotate right panorama by R'
right panorama aligned
X
Y
Z
C1
C2

Figure 3.12: Panorama Alignment by Rotating the Coordinate System

Because of some special properties of the essential matrix E, it can be decomposed to an antisymmetric matrix T and rotation matrix R by a singular value decomposition. After recovering T and R from E, we can align the panoramas by the following method. First we want to rotate the first coordinate system such that $\overline{C_1C_2}$ is on the X axis. This is equivalent to finding a rotation R' which rotates $t = [t_x, t_y, t_z]$ to the pole $[1, 0, 0]$. There is a family of such rotations and we just need to pick one of them as R'. Then the first panorama is aligned. The second coordinate system is first rotated by the inverse of R and then by

R'. That is, transforming all image points in the second panorama by a rotation matrix $R'R^{-1}$. Figure 3.12 illustrates these transformations.

The essential matrix can be computed from knowing a few pairs of corresponding point of the two panoramas. There are non-linear methods which require at least five point pairs to recover the essential matrix. For linear methods, note that the essential matrix is defined up to a scalar, and can be determined by eight parameters. Eight point pairs forms a system of equations of eight unknowns with equation 3.7. Hence the essential matrix can be estimated by linear programming. The material presented in this section can be found in [FAU93].

3.5 Why Panorama

In this section, we will explain why we choose to use panoramas instead of using planar images as our input. One obvious reason of using panorama is that it has a *larger field of view* than that of planar image. Therefore it can capture more rays from the view position. In correspondence based rendering, an image point can be warped to the novel view if the original scene point can be found in both reference images, so that a correspondence can be established between them. If the reference images are planar images, their overlapping part will be limited by its field of view. Using panoramas can ensure that the overlapping part is maximized. With panoramas as the reference images, we can reconstruct a novel panorama along the baseline, hence the virtual camera is allowed to rotate freely at the new view position.

Another reason to use panorama is also related to its large field of view. We will talk about how to find the correspondence information in the next chapter. This will involve a matching process which match regions in one panorama with regions in another panorama. If planar images are used instead, a region in one image may not appear in the other, because it may be clipped away by the image boundary. It will make the matching process more complicated. This is avoided in the case when matching panoramas.

For other correspondence based rendering methods, beside the correspondence information, camera information is also required to computing the warping. Camera calibration is not required with panorama, as the intrinsic parameters are recovered implicitly during the creation of the panorama. Image points in a panorama are indexed with their spherical coordinates, thus the mapping between the image points and the rays are known directly. We still have to know the extrinsic parameters of the panorama which are obtained from aligning two panoramas.

Chapter 4

Matching Correspondence

After the reference panoramas are created and aligned, we need to determine the correspondence between them. In this chapter, we will define the correspondence information and describe a method to obtain it from the reference panoramas.

4.1 Correspondence

The correspondence information is important for computing the final warping. As the reference panoramas are taken from different view positions, the reference panoramas are different because scene points are projected to different positions in the two images. The correspondence information relates the image points which are projections from the same 3D point.

4.1.1 Correspondence as a Mapping

Figure 4.1 defines the meaning of correspondence between a pair of image points. Given two panoramas I_1 and I_2 with view positions at C_1 and C_2 respectively, the *correspondence information* is a mapping M_{12} from I_1 to I_2 such that $M_{12}(p_1) = p_2$ means p_1 and p_2 are projections of a 3D scene point. The image point pair p_1 and p_2 is called a corresponding point pair.

Note that this correspondence mapping may not be one-to-one in most cases. The reason is that some scene points which are visible from one panorama may not be visible from the other one. We say that there is an occlusion in the correspondence. Figure 4.2 illustrates the situation of occlusion.

In the figure, five facets are projected onto the view positions C_1 and C_2 to give the two reference panoramas I_1 and I_2 respectively. These facets are labeled

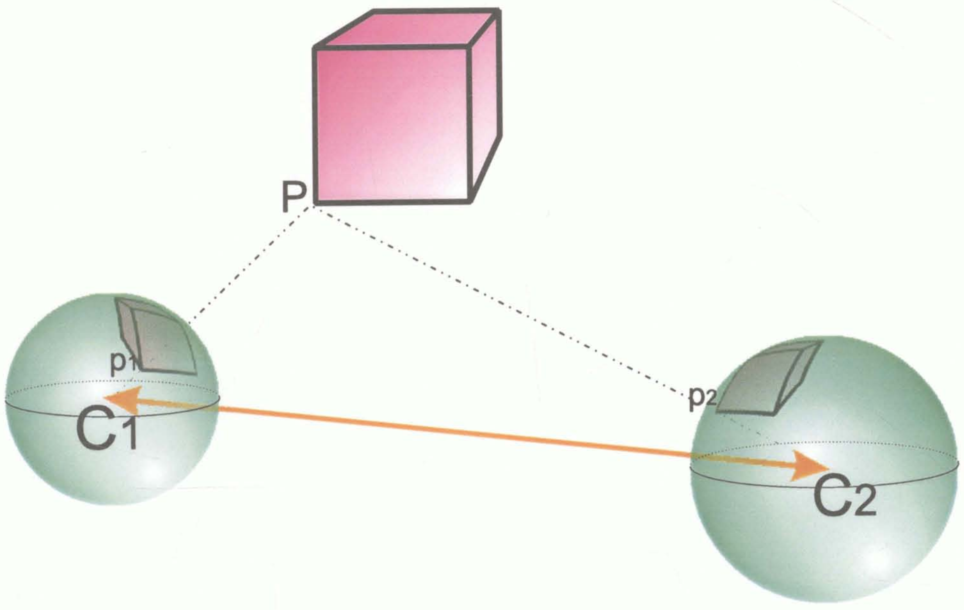

Figure 4.1: Correspondence Mapping

by capital letters F, G, H, J and K. The projections of these facets are labeled by small letters. For example, f_1 is the projection of facet F in panorama I_1 and K_2 is the projection of facet K in panorama I_2. We can see that facet H can be viewed from both C_1 and C_2, therefore the we can map the image points in h_1 to the points in h_2. This mapping is a one-to-one mapping.

There is something special about facet G. It can be viewed from C_2 but not from C_1. Therefore we cannot find any image point in I_1 which maps to the points in g_2 by M_{12}. It is because facet G is occluded by other objects from C_1. On the other hand, facet J is visible from C_1 but not from C_2. Therefore points in j_1 will be mapped to some undefined points. We can think that points in j_1 still map to somewhere in I_2, however, in this case, they are covered by those points mapped from the region k_1, because points from both regions j_1 and k_1 are mapped to the same place.

4.1.2 Full Correspondence and Partial Correspondence

Correspondence information can be classified into two groups. *Full correspondence* means that other than those occluded parts, the correspondence mapping is defined for every image points in both panoramas. The other type of correspondence information is called *partial correspondence.* With partial correspondence, only some special points are mapped from I_1 to I_2. These special points are

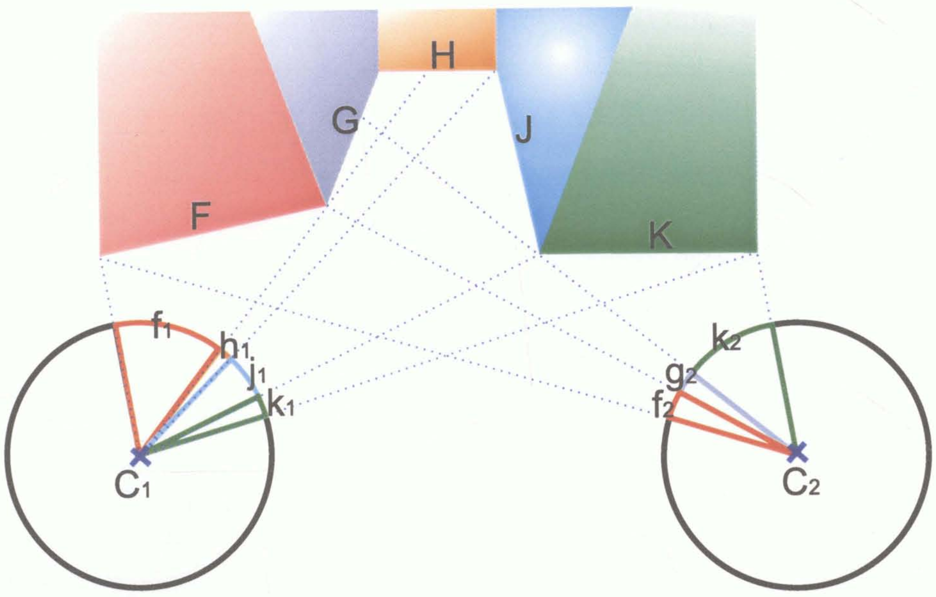

Figure 4.2: Occlusion in the Correspondence

usually feature points or corner points with high gradient in the image.

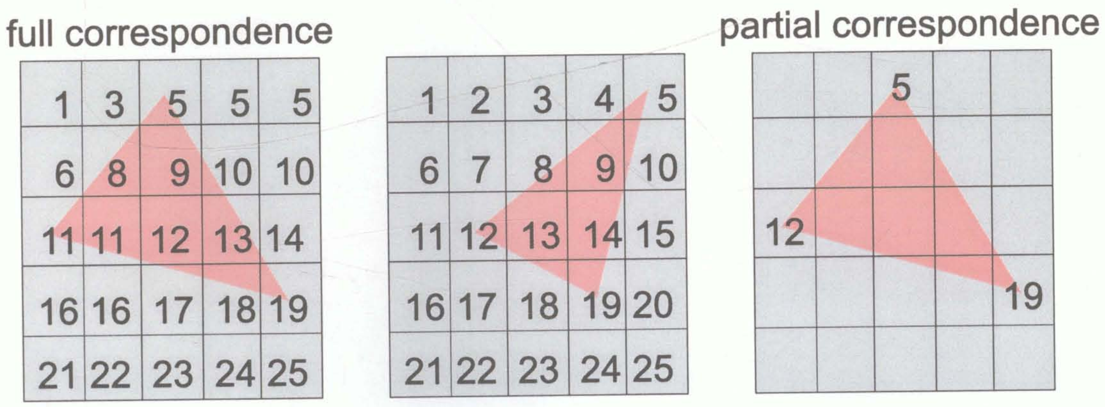

Figure 4.3: Full Correspondence and Partial Correspondence

Figure 4.3 depicts the difference between full correspondence and partial correspondence. The middle image is one of the reference image. The pixels are indexed from 1 to 25. A full correspondence is obtained in the left image, so that each pixel in the image can be mapped to a pixel in the middle image. The images are rectified, thus the corresponding points are appeared in the same horizontal line, and we can obtain the correspondence scanline by scanline. A partial correspondence is obtained in the right image. Note that only those pixels at the edges or the vertices are mapped.

In general, full correspondence is more difficult to compute, and is usually only given by synthetic scenes. This is because in synthetic scenes, the depth information can be known and the correspondence information can be derived from the depth information. In order to reduce the problem size of matching, the usual practice is that feature points are first extracted from the reference images, and only those feature points are matched. In this case, a partial correspondence is obtained. If a full correspondence is given, then we can generate the most accurate result when we have to warp the reference panoramas to the novel panorama. On the other hand, if only partial correspondence is known, we have to guess for the non-feature points, in order to determine how to warp them to create the final images.

4.2 Matching with Epipolar Constraint

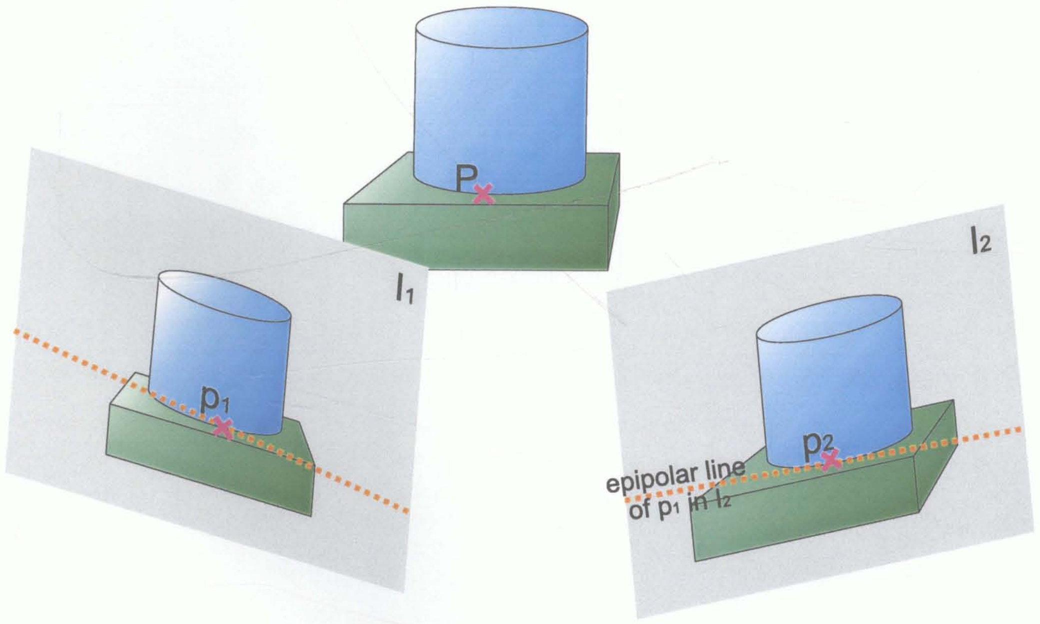

Figure 4.4: Corresponding Points have Similar Properties

This section will present the conventional method to compute the correspondence information between two panoramas I_1 and I_2. In order to solve the correspondence problem, we need to find out, for each image point p_1 in I_1, the corresponding point p_2 in I_2. Because both p_1 and p_2 are projections of the same scene point P, they should be similar to each other. For example, if P is a surface point of a red object, then both projection points should have a color similar to red. Depending on the material property of the surface, there may be slight color difference due to different view directions of the two projections. In addition, if P is an edge point between a blue object and a green object, then after projection, p_1 and p_2 should also be on an edge between a blue region and a green region, as illustrated in Figure 4.4. Hence, the color, the gradient and the neighborhood of the image point can all act as a measure of similarity, or matching score, between two image points from the two panoramas. For each image point in I_1, we can search for the image point in I_2 which possesses the highest matching score. In consequence, we can establish the correspondence between them.

Recall from Chapter 3, because of the epipolar constraint between I_1 and I_2, the corresponding point pair can appear only on conjugate epipolar lines. That means, if we want to find the corresponding point of p_1, we only need to search along the epipolar line of p_1 in I_2, because p_2 is restricted to lie on this epipolar line by the epipolar constraint.

This can reduce the matching problem from a two-dimensional search to a one-dimensional search as follows. After the epipolar geometry is established between the two reference images, the matching process proceeds by matching one pair of conjugate epipolar lines in each step. For each image point along an epipolar line, we consider a small matching window around that image point and find the closest matching window along the conjugate epipolar line. Then a corresponding point pair is found and we continue for the next image point.

As the above method proceeds by working on one scanline a time, it can be implemented efficiently in terms of both space and time, and possibly implemented in parallel. However, one drawback of this scanline approach is that it cannot take into account the *inter-scanline coherence*. Normally, scene points do not exist alone. They are points on the surface of an object, and these surfaces usually are large enough to cover more than one image point, and also more than one scanline, in the reference images. Therefore the correspondence mapping should not vary too much from one scanline to the other. As in Figure 4.5, when we are determining the correspondence mapping on scanline y, we should also consider the correspondence information on scanline $y - 1$ and scanline $y + 1$. Therefore when we are maximizing the matching score of the corresponding point pair, we also need to minimize the difference between the correspondence mapping across the scanlines. This will make the matching problem more difficult.

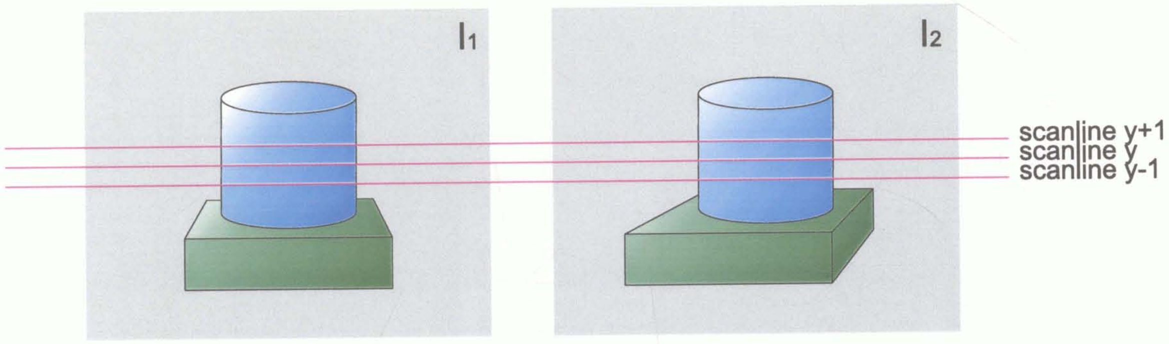

Figure 4.5: Inter-scanline Coherence

4.3 Regions and Segmentation

In order to incorporate the inter-scanline coherence into the matching process, we choose to use *image region* as the entity to match, instead of using image points. Image region is defined as a set of connected image points which are *homogenous* in some sense. For example, all points within a region may have similar color. In most cases, image points of a region are projections of scene points belonging to the same surface of an object. Therefore matching regions can ensure that related image points can be matched correctly, even if they are across different scanline and hence implicitly taking into account the inter-scanline coherence.

4.3.1 Full Correspondence from Matched Regions

We will explain that full correspondence can still be obtained, although only regions are matched. Figure 4.6 shows two matched regions r_1 and r_2. To establish the full correspondence from r_1 to r_2, we will look at one scanline each time. Consider scanline y. y intersects r_1 to give a horizontal line segment which starts from point s_1 and ends at e_1. Similarly, the line segment in r_2 starts from s_2 and ends at e_2. As both s_1 and s_2 are at the left edges of the regions, and they are on the same scanline, they are projections of the same scene point and hence a pair of corresponding points. Similarly, e_1 and e_2 form another corresponding point pair.

The remaining is to determine the correspondence of the internal points of the line segments $\overline{s_1e_1}$ and $\overline{s_2e_2}$. As all points within the region are similar, it is meaningless to talk about how to determine the exact correspondence between them. Therefore we will approximate the correspondence by some mappings from line segment $\overline{s_1e_1}$ and $\overline{s_2e_2}$. Let a point in line segment $\overline{s_1e_1}$ be $p_1(t) =$

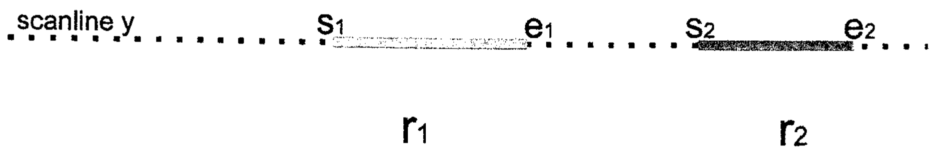

Figure 4.6: Full Correspondence from Matched Regions

$(1-t)s_1 + te_1$ and a point in line segment $\overline{s_2e_2}$ be $p_2(u) = (1-u)s_2 + ue_2$, for some $t \in (0,1)$ and $u \in (0,1)$. If we use linear mapping, then a point $p_1(t)$ will map to $p_2(u)$ where $u = t$. On the other hand, if the depth values of the end points can be determined, we can use projective mapping. Under such projective mapping, a point $p_1(t)$ will map to $p_2(u)$, where u is the value of t after a projective transformation. This projective transformation is determined by the depth values of the end points. If projective mapping is used, we are approximating the scene structure from scene point S to E by a straight line.

As the number of regions in an image is smaller than the number of image points, it is more efficient to match the regions instead of matching the image points. In addition, with the help of hardware texture mapping, warping the matched regions can be implemented more efficiently, which will be discussed in the next chapter. Therefore we choose to match regions as it is more intuitive and more efficient than matching image points.

4.3.2 Image Segmentation

Before we can proceed to match the regions, we need to first identify the regions in the image. This process is known as *image segmentation*. We use a simple method to segment the image, which is described as follows. Edges in the image are first detected by applying a gradient operator on every image points. The edges, which are located at those points with high gradient, are considered as boundaries of the regions. Then in each step of our method, a single image point is selected randomly. The single image point does not belong to any region defined so far. A region is then grown from this single point by taking in the neighboring points until the boundary is reached. The process stops when the union of all regions equals to the whole image. Otherwise, another new region is created. The process is illustrated in Figure 4.7.

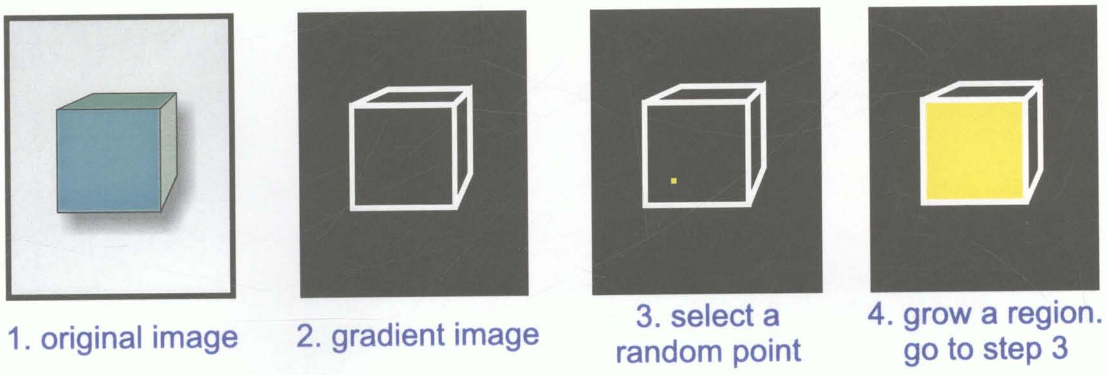

Figure 4.7: Segmentation by Growing a Region

Segmentation is a well developed research topic in image processing and they are other techniques like splitting and merging, relaxation [FAU93]. These more advanced techniques are more robust in handling noisy data and give better result. For the ease of implementation and understanding, we just adopt the above simple growing method.

4.4 Matching Method

After the reference panoramas are segmented into regions, the next thing to do is to match the regions between them. Assume I_1 is segmented into a set of m regions $L = l_1, l_2, \ldots, l_m$ and I_2 is segmented into a set of n regions $R = r_1, r_2, \ldots, r_n$. To match the regions means that for each region l_i in L, we want to find a corresponding region r_j in R and if l_i has been matched to r_j, it cannot be match to another region in R. For each possible match (l_i, r_j), we define a *matching score* $score(l_i, r_j)$ which indicate the possibility that these two regions are corresponding regions. Higher match score suggests a better match. The definition of this matching score will be given in the next section.

4.4.1 Maximum Weighted Matching in Bipartite Graph

A *weighted bipartite graph* $G =< V, E >$ is a graph with a node set V and an edge set E which satisfies the following conditions:

- V can be partitioned into two subset V_L and V_R;
- E only contains edges linking a node from V_L and a node from V_R; and
- every edge e in E is assigned a weight $w(e)$.

A *matching* M is a subset of the edge set E such that no two edges in M share a node in V. The weight of a matching M, denoted by $w(M)$, is the sum of weights of all edges in M. A matching M is a maximum weighted matching of G if for any matching M' of G, we have $w(M) \geq w(M')$. A weighted bipartite graph with its maximum weighted matching is shown in Figure 4.8. In the figure, the circles labeled F, G, H, J and K are the nodes, which can be partitioned into two sets V_L and V_R. The line joining the nodes are the edges and the associated numbers are the weights. The maximum weighted matching is shown in red color, which is consisted of two edges (H, G) and (K, J).

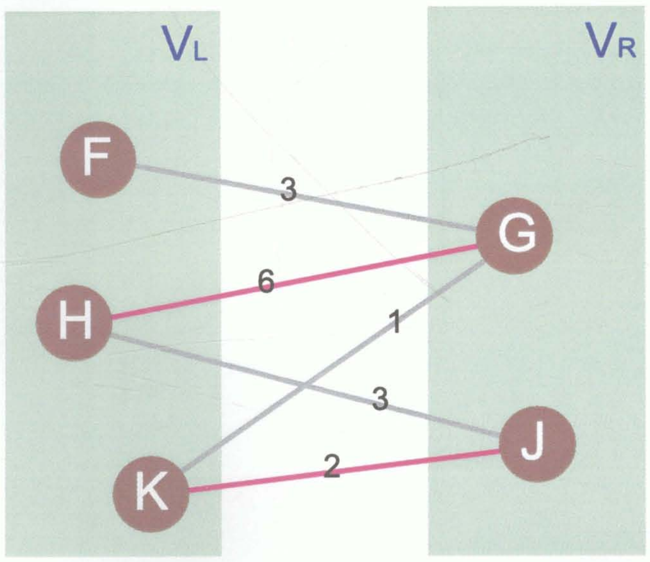

Figure 4.8: A Weighted Bipartite Graph and its Maximum Weighted Matching

We can see the similarity of matching two sets of regions and the maximum weighted matching in bipartite graph problem. Define a bipartite graph $G =< V, E >$. V is partitioned into two subsets V_L and V_R. Each region in the first panorama is represented by a node in V_L and each region in the second panorama is represented by a node in V_R. We represent a pair of regions $l_i \in V_L$ and region $r_j \in V_R$ as an edge e_{ij} in E such that $w(e_{ij})$ equals to the matching score of the pair $score(l_i, r_j)$. In this case, G is a complete bipartite graph. By finding the maximum weighted matching M of G, we can determine the correspondence among the regions as follows. In the matching M, if there is an edge connecting two nodes l_i and r_j, we decide that the regions represented by l_i and r_j are a pair of corresponding regions.

Note that a matching of G represents a correspondence between the two panoramas. As the matching score stands for the similarity between two regions, we prefer a higher matching score than a lower one. Therefore by selecting a maximum weighted matching, we can ensure that we have chosen the correspondence which has the highest sum of matching scores. The method to obtain a maximum weighted matching of a bipartite graph can be found in [GAL86]. The time complexity of this process is $O(|E||V|\log|V|)$. Maximum weighted matching has been used to solve the correspondence problem in the similar way to match image points and lines [CHE94].

4.5 Matching Score

For the matching to be success, we need to define the matching score for every pair of regions from the two reference panoramas. If the regions are projections of the same 3D surface, then the match score should give a high value. Otherwise, it should give a low value to indicate a false match. We have devised a formula to define the matching score which is composed of three components, the color, the position and the shape.

4.5.1 The Color Component

It is easy to represent the similarity of the color component. The color of a region is defined as the average color over all the image points within it. The color is usually represented in the RGB space.

As we want to normalize the value of this component to 1, the score of the color component between region l and region r is defined as,

$$score_{color} = 1 - \sqrt{\frac{(R(l)-R(r))^2 + (G(l)-G(r))^2 + (B(l)-B(r))^2}{3}} \tag{4.1}$$

where R, G and B are the red, green and blue component respectively, with value ranging from 0 to 1.

4.5.2 The Position and Shape Components

To represent the position component, consider Figure 4.9 which shows some regions in the two reference panoramas. The two panoramas are aligned so that conjugate epipolar lines are held in the same scanline. As we want to find a match for region l_1, we have to determine the matching scores for (l_1, r_1), (l_1, r_2), (l_1, r_3) and (l_1, r_4). If all the five regions have the same color, which one do you want to match with l_1?

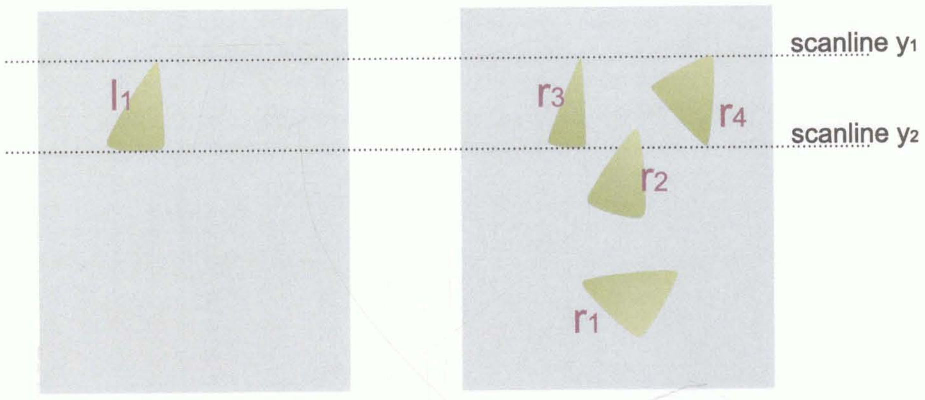

Figure 4.9: Which Pair is the Best Match?

Let's start with region r_1. Obviously we will give a very low score to the matching (l_1, r_1). It is because the location of l_1 is too far from that of r_1. As we know that corresponding points must be on the same scanline, because the two reference panoramas are aligned. However, these two regions do not even share a single scanline, hence they are most unlikely be matched as a pair. Although the set of scanlines occupied by l_1 partly overlap with that of r_2, we will not assign a very high score to their matching score for a similar reason.

Both sets of scanlines spanned by r_3 and r_4 overlap exactly with that of l_1, do you prefer r_3 over r_4? The reason we give a higher matching score to r_3 is that r_3 has approximately the same shape as l_1. The lower part is wider than the upper part in both l_1 and r_3. Therefore the matching score for r_3 is greater than the matching score for r_4, which is in turn greater than that for r_2 and r_1.

We observe that both the position and the shape of the region play an important role in determining the matching score. More precisely, if the vertical range of region l_1 starts from y_1 and ends at y_2, we should match l_1 with a region which also encloses a similar vertical range.

Span and Vertical Profile

Define the *span* of a region across a scanline as the horizontal line segment which is the intersection of the region and the scanline. For example, in Figure 4.10, the span of region r across scanline y is the red line segment which starts from point u and ends at point v. Then the length of this span is the horizontal difference of these two end points. For each scanline in the image, there is a corresponding length of span of the region across this scanline. Define the *vertical profile* V as a set of these lengths indexed by the scanline. For example, the vertical profile at scanline y $V(y)$ has the value of $v - u$. The whole profile is also shown at the right hand side of the figure.

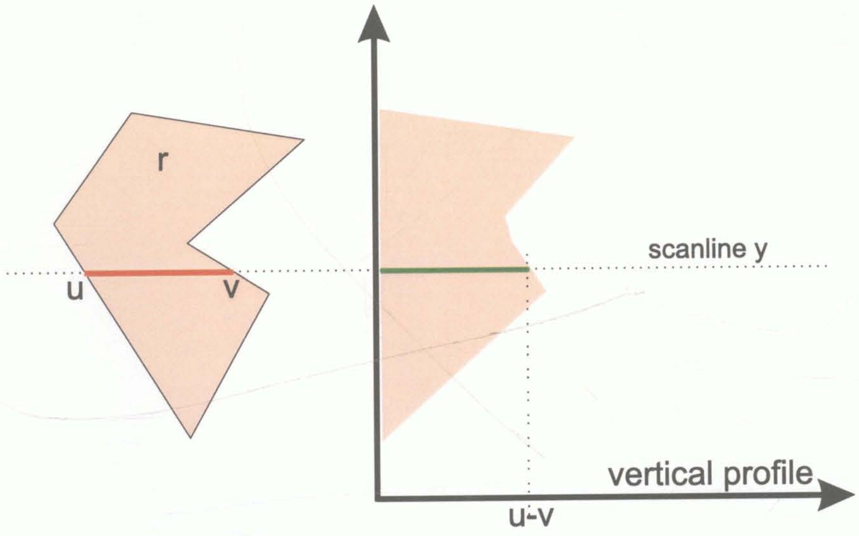

Figure 4.10: Span and Vertical Profile

Figure 4.11 compares the vertical profiles of the regions we have considered in Figure 4.10. Observe that if the vertical profiles are more alike, we will assign a higher matching score to that pair. The position and shape information of the region has been encapsulated in the vertical profile of that region. It is obvious that the vertical profile is also capable of differentiating the case between regions r_3 and r_4. Hence, by comparing the vertical profile alone, we can determine which pair of regions should receive the highest matching score.

Assume l_1 and r_3 are projections of the same 3D triangle T, so we should have a perfect match between these two regions. However, we see that the vertical profile of l_1 is 'fatter' than that of r_3. This is because the two images are taken from different view positions, and the viewing directions make two different angles

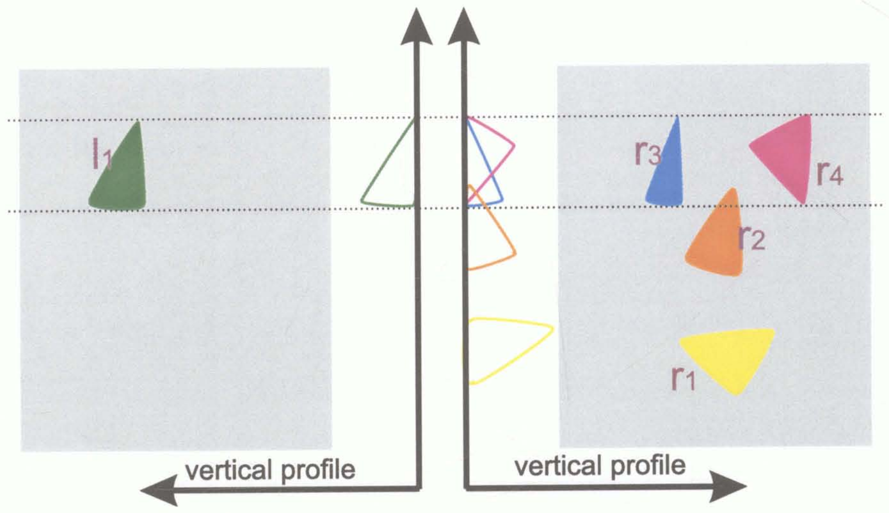

Figure 4.11: Comparing Vertical Profiles

with the normal of T. Region r_3 is taken with a more oblique direction, therefore the projection r_3 is smaller in size.

Normalized Vertical Profile

It is better if we can modify the vertical profile so that this effect can be canceled out. To make the vertical profiles of the two matching regions exactly the same, the vertical profile is divided by the total area of the region. The reason behind this is, we see that if the total area of the region is larger, the region will have a 'fatter' vertical profile. We call this the *normalized vertical profile*, which is the vertical profile divided by the total area of the region. Figure 4.12 shows the vertical profiles and the normalized vertical profiles of regions l_1 and r_3.

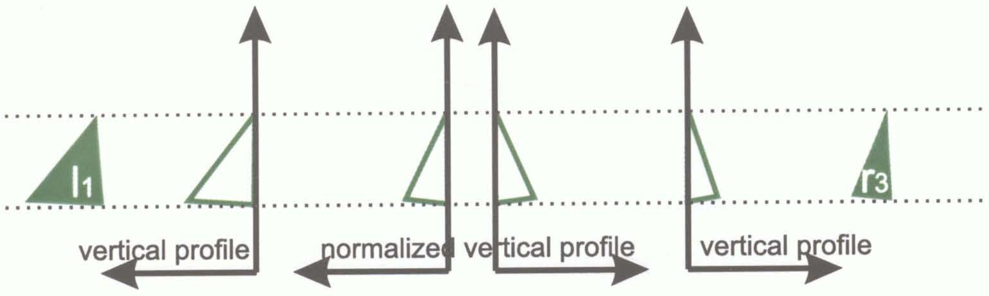

Figure 4.12: Vertical Profile and Normalized Vertical Profile

We will show that the normalized vertical profiles of two corresponding regions are the same, if they are projections of the same 3D triangles. Assume the

l_2. Note that because the upper triangle and the lower triangle share the same baseline, we have,

$$\begin{aligned} & \frac{A_1}{B_1} \\ = & \frac{\text{height of upper triangle}}{\text{height of lower triangle}} \\ = & \frac{A_2}{B_2} \end{aligned}$$

As scanline y lies within the range of the upper triangles, the normalized vertical profiles at scanline y are the same for the two upper triangles. That is,

$$\frac{l_1}{A_1} = \frac{l_2}{A_2}$$

Then,

$$\begin{aligned} \frac{A_1 + B_1}{l_1} &= \frac{A_1}{l_1} + \frac{B_1}{l_1} \\ &= \frac{A_2}{l_2} + \frac{B_1}{A_1}\frac{A_1}{l_1} \\ &= \frac{A_2}{l_2} + \frac{B_2}{A_2}\frac{A_1}{l_1} \\ &= \frac{A_2}{l_2} + \frac{B_2}{A_2}\frac{A_2}{l_2} \\ &= \frac{A_2 + B_2}{l_2} \end{aligned}$$

Hence,

$$\begin{aligned} & \text{normalized vertical profile of the left triangle at scanline } y \\ = & \frac{l_1}{A_1 + B_1} \\ = & \frac{l_2}{A_2 + B_2} \\ = & \text{normalized vertical profile of the right triangle at scanline } y \end{aligned}$$

The case is treated in the same way if scanline y lies within the lower triangle. This completes the proof for matching regions which are triangles. If the shape of the region is irregular, we can construct a similar proof by generalizing the idea of splitting the region into triangles, and the same result will follow.

Hence we will use the normalized vertical profile for the position and shape component of our matching score. This component is given as,

$$score_{pos+shape}(l, r) = 1 - \sqrt{\frac{\sum_{y=1}^{Y} (NVP_l(y) - NVP_r(y))^2}{Y}} \tag{4.2}$$

where Y is the total number of horizontal scanline, NVP_l and NVP_r are the normalized vertical profile of region l and r respectively. Note that the maximum value of NVP for a single scanline is 1, because the area of the region is defined as the number of pixels it covers.

4.5.3 Combining the Components

We have defined the color component in equation 4.1 and the position and shape component in equation 4.2 of the matching score for matching two regions. Both components have been normalized so that they have a range from 0 to 1. If we think that these component are equally important, the matching score will be given as,

$$score(l,r) = score_{color}(l,r) + score_{pos+shape}(l,r)$$

Otherwise, we can give a weight to indicate their significance,

$$score(l,r) = weight \times score_{color}(l,r) + score_{pos+shape}(l,r)$$

Chapter 5

Warping

Recall that our goal is to perform interpolation between two reference panoramas. After we obtain the correspondence information, we need to reconstruct the in-between panorama, according to the current location of the viewer. This chapter will describe how we can create the novel view by transforming the image points in the reference panoramas. We will also talk about an efficient rendering method for fast interactive display.

5.1 Warping Image Points

With two reference panoramas taken from view position C_1 and C_2, panorama interpolation reconstruct the panorama with view position located along the line segment $\overline{C_1C_2}$. The line segment $\overline{C_1C_2}$ is referred as the baseline. We will parameterize the position C_t in terms of C_1 and C_2, such that,

$$C_t = (1 - t)C_1 + tC_2$$

where $t \in [0, 1]$. Different panorama will be generated when C_t moves from C_1 to C_2.

In Figure 5.1, a pair of corresponding points p_1 and p_2 is given on the reference panoramas I_1 and I_2 respectively. From this corresponding point pair, we back project them through the green rays into the 3D space and the intersection is located as P. Therefore the scene point can be recovered from the corresponding points. If the position of the viewer is at C_t, we can reproject the recovered scene point through the blue ray onto the novel panorama I_t. The projection p_t is the new image point.

tangents of its two corresponding point positions,

$$\text{relative depth} = \frac{1}{|\tan\phi_1 - \tan\phi_2|}$$

If two corresponding point pairs happen to warp to the same position in the novel panorama, we can resolve this by comparing the relative depths of these two pairs. The one with smaller relative depth will occlude the other, hence be displayed in the final image. This can be efficiently implemented with z-buffering, by setting the z-value of each point as the relative depth.

5.2.2 The Hole Problem

In the fold problem, more than one pair of corresponding points are warped to the same position in the novel panorama. On the contrary, in the hole problem, no point pair will warp to some positions in the novel panorama, leaving a hole there which cannot be filled by any color. This happens when a part of the scene, which is invisible from both reference panoramas, suddenly becomes visible from some view positions during panorama interpolation. Figure 5.8 illustrates this situation. The scene surface F is not visible from both reference view positions, and therefore we cannot find the projection of F between the projections of surfaces D and H in the reference panoramas. When the view position moves to C_t, there is a gap between the projections of surfaces D and H. This gap should be occupied by the projection of surface F, but since the information of F is simply missing in both reference panoramas, it is clear that we cannot assume anything about this gap in the novel panorama.

In practice, however, we will not have such problem. Because we do not have any information about F from the reference panoramas, F simply does not exist in our perception. The scene information is derived only from the reference panoramas.

Since the actual structure of the scene is unknown, the warping of the gap will depend on how we interpret the available information. In the previous example, because D is visible from both view positions, the geometry of D can be estimated by locating the matched end points in the image space. This is the same for H, so there is no problem in warping D and H to d_t and h_t in the novel panorama.

The projection of G appears in the hole in the first panorama, and that of E appears in the second panorama. Although they are unmatched regions, we

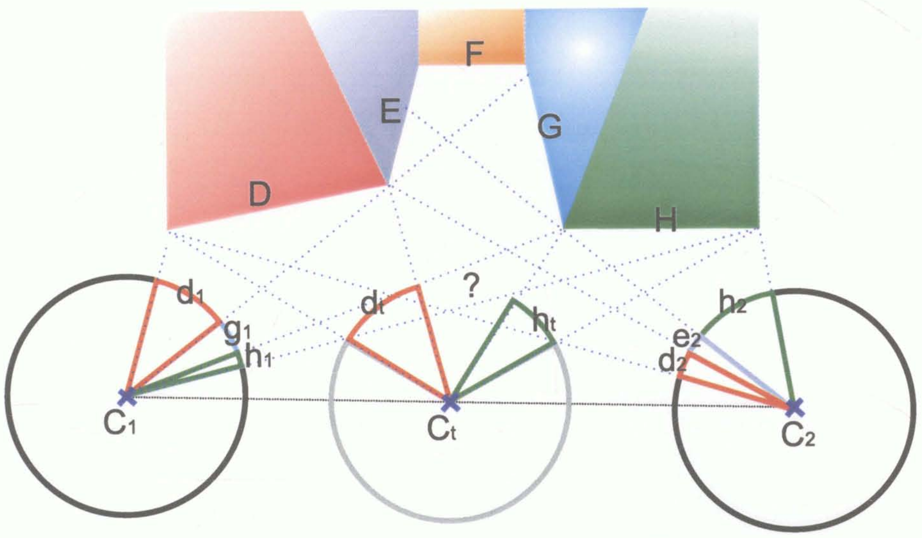

Figure 5.8: The Hole Problem

should still fill the hole by warping these projections into the hole. Therefore the question now changes to how we handle these *unmatched regions*.

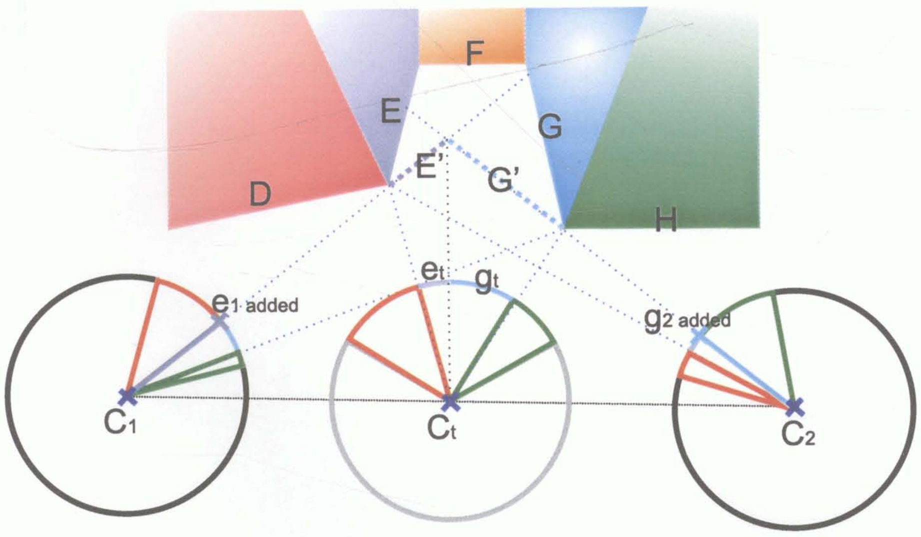

Figure 5.9: Filling the Hole with Degenerated Projections

There are several different cases to consider. We have two reference panoramas in Figure 5.9. They both have projections of D and H. The first panorama has the projection of G in the region between d_1 and h_1 and the second panorama has the projection of E in the region between d_2 and h_2. In the novel panorama, we want to fill the hole by the projections of E and G. This can be done by inserting a projection e_1 in the first panorama, which is a single image point between projections d_1 and g_1. e_1 can be considered as a *degenerated projection*. Then by matching e_1 with e_2, the hole in the novel panorama can be partly filled by e_t. We also add a degenerated projection g_2 of G between the projections e_2 and h_2 in the second panorama. The warped region g_t fills the remaining part of the hole.

Let us look at the geometry of the scene represented by this setup. A degenerated projection corresponds to a planar region which is parallel to the ray defined by that projection. Therefore, instead of recovering the real shape of E, this region is approximated by the planar region E' in Figure 5.9. Similarly, G is approximated by a new facet G'. When the view position is traveling between C_1 and C_2, what we observe will be a concave V-shaped structure formed by E' and G'.

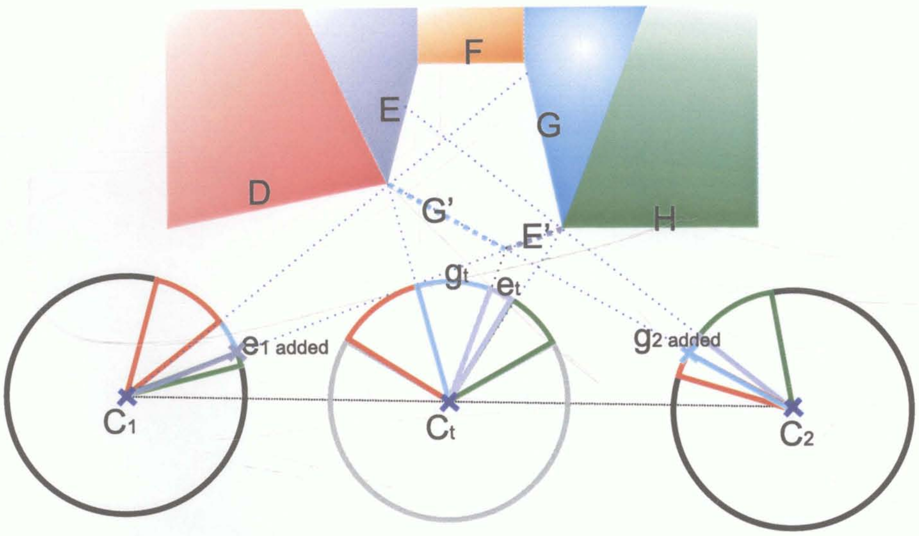

Figure 5.10: A Convex V-Shaped Structure

Including the degenerated projections, the projections in the reference panoramas are in the order of D, E', G' and H, from left to right. We have the freedom to change the order to D, G', E' and H, by inserting e_1 between g_1 and h_1 in the first panorama and inserting g_2 between d_2 and e_2 in the second panorama. This will correspond to a convex V-shaped structure as in Figure 5.10.

Sometimes there is a hole in the first panorama, and the hole does not appear in the second panorama, as shown in Figure 5.11. The regions in the first panorama are d_1, g_1 and h_1, from left to right, which are projections of D, G and H respectively. In the second panorama, g_2 is missing and d_2 is next to h_2. In this case, we will add a degenerated projection g_2 between d_2 and h_2 which

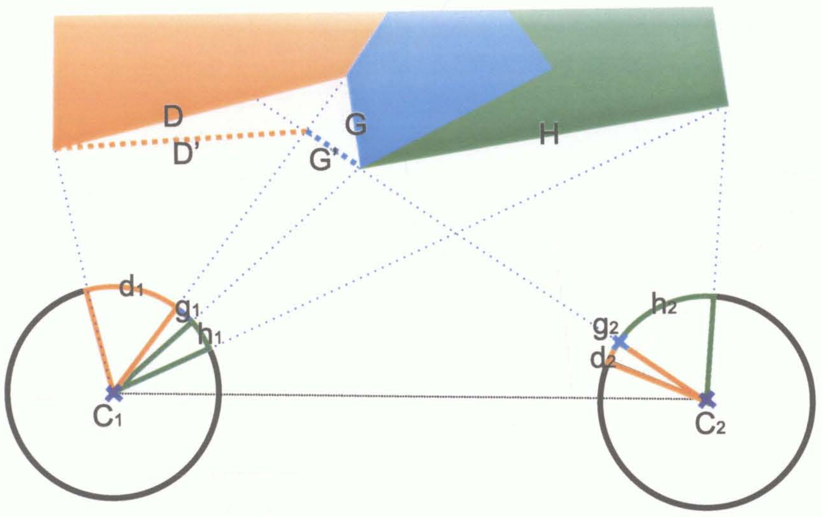

Figure 5.11: Hole Appears in Only One Panorama

will match with g_1 in the first panorama. Then the structure of the hole will be approximated by G' connecting D and H. This is equivalent to filling the hole with the color of the singleton region g_1. Note that in this case, the real structure of D will be approximated by D'.

In summary, if a region in the first panorama cannot be matched with any region in the second panorama, we will insert a degenerated projection into an appropriate position in the second panorama, and match the unmatched region with this newly added projection. The position of the degenerated projection is determined by the neighboring regions of the unmatched region in the other panorama. For example in Figure 5.11, because the unmatched region is surrounded by d_1 and h_1, the degenerated projection will be added between d_2 and h_2 in the second panorama. By matching the degenerated projection with g_2, the hole can be filled.

5.2.3 False Match

Refer to Figure 5.9. If E and G have similar color, the matching algorithm may mistakenly match their projections g_1 and e_2, just because their matching score is high enough and it cannot find a better match for them. Then the original concave structure will be approximated by a single facet E' in Figure 5.12. It is not a bad approximation, though, as the colors of the two regions are similar.

A related problem is shown in Figure 5.13. Consider a curved surface of an object. The projections s_1 and s_2 of this surface in the two reference panoramas

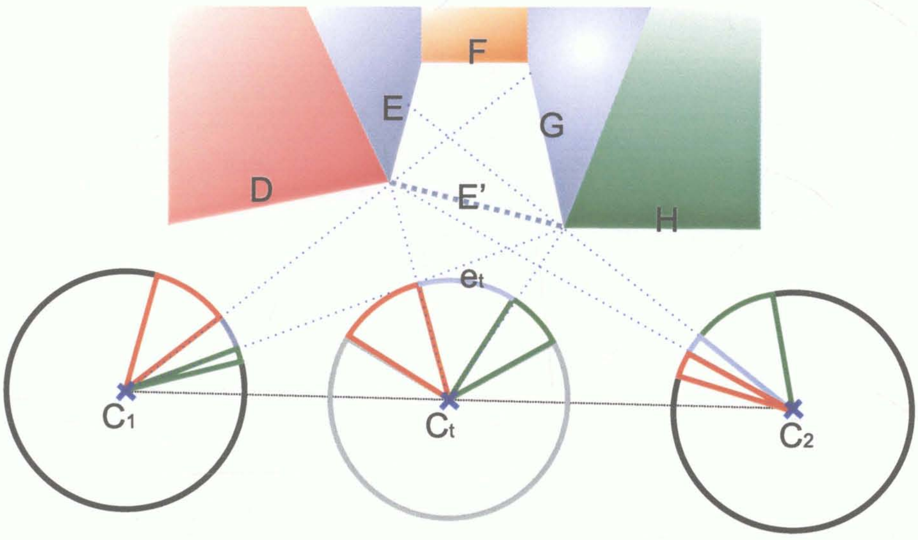

Figure 5.12: A False Match

are matched. As in the previous analysis, these regions are warped according to the location of the boundaries. However, observe that the boundaries do not match. From left to right, denote the starting position of s_1 as t_1 and the ending position as u_1. Similarly, let t_2 and u_2 bound s_2 in the second panorama. We see that t_1 is the projection of an edge point T, but t_2 is the projection of another point T' on the curved surface. They are not projected from the same physical point. This point will be warped as if it is located at T'' in 3D. There is a similar case for u_1 and u_2. Therefore, the curved surface will be approximated by a planar region $T''U''$.

These two examples illustrate that although we rely on the matching of the boundaries of the corresponding regions to compute the warp, even if the matching is not accurate, what we observe in the final image will not be too different from the original structure.

5.3 Warping Regions

Instead of repeating the warping for each pair of corresponding points, we will treat a region as the unit to warp. As we will see in this section, the interior of the region can be warped by interpolating the image points at the boundary. This will greatly improve the efficiency because the warping equation is only required to evaluate at some boundary points.

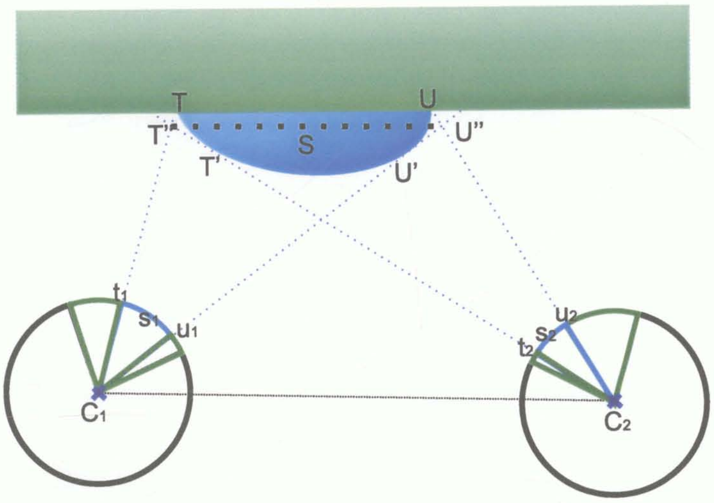

Figure 5.13: A Curved Surface Viewed from Different View Positions

5.3.1 Compatible Triangulation

We will present a method to warp a pair of matched triangles. For a region which is not a triangle, it will be first triangulated into several triangles. There are several points that we need to take extra care when performing triangulation on matched regions.

Assume the two matched regions r and s are triangulated into two sets of triangles $T = \{t_1, t_2, ..., t_m\}$ and $U = \{u_1, u_2, ..., u_n\}$. Because at the end, we will warp the triangles in T and U, the triangles in these two sets are required to be matched. In other words, we want to perform a *compatible triangulation*, such that T and U has the same number of triangles and each triangle t_i in T can be matched with a triangle u_i in U and vice versa. Two triangles can be matched if they have corresponding vertices on the same epipolar line.

We will use a simple method to achieve this triangulation for two convex matched regions. We first trace the boundary of a region and try to fit a straight line with the boundary. Whenever we reach a point where the boundary can no long fit with the straight line, the point is recorded and a new straight line is used. This process is repeated until the whole boundary is traced. The boundary of the other region is also traced. Figure 5.14 shows two matched regions. The recorded points for region r are represented by red dots and those for region s are represented by blue dots.

Let the set of the recorded points for region r and s be $V_r = \{v_{r1}, v_{r2}, \ldots\}$ and points of the first reference triangle $\triangle H_1K_1G_1$ to the warped triangle $\triangle H_tK_tG_t$, we can consider triangle $\triangle H_1K_1G_1$ as a photograph of a 3D triangle $\triangle HKG$, which is being projected to our final image. We have performed two perspective projections. One is from triangle $\triangle H_1K_1G_1$ to the 3D triangle $\triangle HKG$ and the other is from triangle $\triangle HKG$ to the warped triangle $\triangle H_tK_tG_t$ in the final image. This transformation can be implemented with hardware texture mapping by considering the projective texture coordinates of the vertices of the triangle [SEG92]. The warped triangle is correctly rendered by taking the depths as the fourth texture coordinates of the vertices. The final image is obtained by averaging the warped triangles of the two reference triangles.

Note that the result is the same as how we obtain a full correspondence from matching the regions in Section 4.3. By considering the regions as planar and matching the interior points, we can perform a warping on every point pair. The method presented in this section is more efficient, as we only have to warp the vertices of the triangle. The interior points are handled by hardware texture mapping.

5.4 Novel View Composition

This section summarizes what we have proposed so far for generating novel views from warping reference panoramas. We will discuss several issues to improve the performance in the warping stage to make real time rendering possible.

5.4.1 Preprocessing Steps

Recall that in Chapter 3, we talk about how to create panorama from images taken from the same view position. Two panoramas of the same scene are then aligned by finding the essential matrix between them. The essential matrix can be computed from manually specifying a few pairs of corresponding points.

Homogenous regions are extracted from the panoramas. Because the panoramas have been aligned, we can establish the epipolar geometry between them. The extracted regions are then matched. The matching problem is transformed to finding a maximum weighted matching in bipartite graph. The matching score between two regions takes into account the color, the relative position and the shape of the regions. These are described in details in Chapter 4.

For each unmatched region, a degenerated projection is added to the appropriate position. In order to correctly handle the hole problem. The location of the degenerated projection can be found by considering the locations of the neighboring regions.

In order to make use of the hardware graphics accelerator, we need to triangulate those irregular shaped regions to triangles. This is done by compatible triangulation, as introduced in this chapter, which ensures that the resulting triangles can also be matched.

This finishes the preprocessing steps. The reference panoramas are broken up into matched triangle pairs. We can just store the vertices of the triangles as corresponding point pairs and also the connectivity of the triangles in the reference panoramas.

5.4.2 Warping Steps

The novel view is generated according to the current view position. It is allowed to move along the line segment between the view positions of the reference panoramas. Every time the view position is changed, we have to warp the corresponding vertex pairs and compute the position for the vertices of triangles in the novel panorama. We do not explicitly construct the panorama located at the new view position. Instead, we will just store the position and the depth of the vertices. The depth of the vertices can be computed from the relative depth and the ϕ angle of the warped point. After the viewing parameters, such as viewing direction and zoom ratio, are known, the novel image can be rendered using ordinary 3D graphics approaches by drawing the warped triangles on the novel image.

We may need to frequently evaluate the tangent of an angle when we are warping the corresponding point pairs by equation 5.1. We can avoid this expensive evaluation by constructing in advance a table which precomputes the tangents of angles ranging from $-\frac{1}{2}\pi$ to $\frac{1}{2}\pi$, in 0.005 interval. Thus we can save the time by just looking up the table.

The depth of a vertex plays an important role during the rendering of the triangles. As explained before, the triangles are drawn as 3D triangles by assigning depth values to their vertices. The Z-buffer algorithm for visible surface determination can be used to eliminate occluded triangles. This is supported in most hardware accelerators, so the fold problem can be quickly handled. Another use of the depth is to provide the projective coordinates of texture mapping, so the interior of a reference triangle can be mapped to that of the warped triangle. Again texture mapping is implemented in most hardware accelerators.

When warping the interior points of a triangle, we need to blend the colors of two reference triangles. This can be implemented with *multitexturing*, which is a new feature supported by OpenGL 1.2 [WOO99]. Multitexturing can apply two textures unit to the same polygon. By setting the alpha value of the second texture unit to 0.5, we can blend the result together to form the final triangle. Alternatively, we can use the technique of *view dependent texture* [DEB96], which can combine textures from several photographs according to the viewer's position to produce realistic rendering.

Chapter 6

Comparisons and Experiments

After we have a thorough understanding of the whole system, in this chapter, we will study its features and compare it with other rendering methods. The chapter is then concluded with some experimental results of the system.

6.1 Comparisons

In this framework, correspondence is first established between the reference panoramas. With the help of this correspondence information, we are able to obtain a warping for generating novel views from the reference panoramas. Therefore, this framework can be classified into the group of image based rendering using correspondence.

It has the general advantages of the correspondence based approach over those geometric based and ray based approaches. Against those geometric based methods, this approach avoids the tedious and labor intensive *modeling task*. The input can be obtained directly by taking images of the scene. The advantage of this method over those ray based methods is that *storage requirement* is significantly reduced, as we do not need to save every rays in the scene. In addition, we do not need to acquire a large sets of images of the scene, which makes this approach more practical. However, the *correctness* of the novel views generated from our approach is highly dependent upon the result of the correspondence matching. If the matching is accurate, we are able to obtain correct views from the reference panoramas. This problem does not exist in geometric based rendering, as 3D models are given as input. Ray based methods can also ignore this problem, as novel views are generated by resampling existing rays only. There is nothing needed to guess during the whole process.

We will now look at the features of this framework and compare this work with other correspondence based rendering methods.

6.1.1 Panorama as Reference

We have explained in chapter 3 why we choose to use panoramas as our reference images, instead of using planar images [CHE93, SEI96]. The main reason is that panorama has the *largest field of view*, which enables better correspondence matching. In addition, less camera parameters are need to recover for warping.

6.1.2 Region Matching Algorithm

We have provided an automatic algorithm for correspondence matching. Instead of matching image pixels, we found that matching regions is *more intuitive* as what make up an object in the image are regions which can be thought as the projections of the surface of the object. In addition, computing the correspondence information within a region is meaningless as the region is homogeneous. It is difficult, if not impossible, to determine the correspondence between the interiors of two regions. Matching the regions has made the problem easier to solve. Moreover, the matched regions can be trianglulated into triangles, so they can be *rendered by graphics hardware.*

The most important advantage of matching regions is that *interscanline coherence* is implicitly incorporated. As a region is a connected set of image points, a pair of matched regions ensure that the matching is consistent across neighboring epipolar lines. In constrast, special care must be taken if we were matching the image points.

An early paper [OHT85] has presented a stereo matching algorithm to match regions by dynamic programming. It generalizes a 2D interval matching algorithm to a 3D region matching algorithm. It has the same advantage as our method, in which interscanline coherence is considered. However, it requires that the scene to be monotonic, that is, the order of the image points in the images has to be the same. Although it has restricted the usefulness of the algorithm, it should be good if we can include this feature into our system by adding a component to the matching score to account for this factor. This is because this assumption is valid in most times.

Another image based virtual reality system [CHA99] has also used cylindrical panoramas as the reference images. While developed independently, their approach is very similar to the framework proposed in this thesis. The only difference is the way to match the correspondence. Their matching method attempts to match corresponding line segments with correlation techniques and the epipolar constraint, and the correspondence information derived can be used for assisting a human operator to further refine the matching. Although our method is trying to solve the correspondence problem automatically, the result is only satisfactory. It would be nice if our method can allow the user to interactively improve the correspondence with minimal efforts.

6.1.3 New Matching Score

Beside matching the color property of the region, our matching score is capable to consider the *shape* and the *position* of the region. While the position can be constrained by the epipolar geometry, it is very difficult to consider the shape of the region, as it may be quite different from the shape of the original 3D surface. However, our matching score is designed in a way so that it is invariant when the 3D surface is viewed from different angles in the reference panoramas.

6.1.4 Free to Rotate

In our method, when the viewer changes its view position, a novel panorama is obtained from the correspondence information. This computed panorama allows the virtual camera to point at any view direction. This will give the viewer a *sense of immersion* as he or she can freely rotate and look around the scene. This is different from those methods [CHE93, SEI96] which use planar images, in which the possible viewing directions are restricted.

6.1.5 Efficient Rendering

After the correspondence information is obtained, the reference panoramas are triangulated into matched triangle pairs. These are done in the preprocessing steps. We only need to store these matched triangle pairs in the storage. When we come to the rendering stage, every time the viewer changes its position by moving forward or backward, our method allows us to *only warp the vertices of the triangles.* We do not need to compute the warping in the interior of the triangles. These triangles are then rendered to give the novel view. The warped vertices are stored and they are not recomputed until the view position is changed again. Therefore, if the viewer choose to just rotate its view direction, which is true in most times, we can just use the warped triangles for display. In this way, we can make the rendering more efficient by saving the number of evaluations of the warping equation.

6.1.6 Hardware Support

Our framework is designed so that the result of matched correspondence can be converted to a set of triangles with projective texture coordinates, therefore the rendering can be *fully supported by existing graphics hardware.* For example, the vertices of the triangles are associated with depth values, thus they can be rendered with z-buffering to eliminate occluded triangles. The interior of the triangle is filled with texture mapping, which is compatible with the hardware implementation. This is similar to the approach taken in [FU98], which convert a pixel based warping method [MCM95b] to a triangle based method. We have successfully transferred the processing requirement from the CPUs to the graphics boards, which is so popular that they have been equipped with most PCs. With the graphics board generating tens of millions of triangles per second [CRE00], it is a waste to still warping the novel view in a pixel by pixel way.

6.2 Experimental Results

We have implemented several computer programs to verify the feasibility of our proposed work. This section will show some results of these experiments.

We will start with a pair of panoramas taken from different view positions. These panoramas are captured from a virtual reality application as a cube representation, as shown in Figure 6.1. They are taken within a virtual graphics laboratory. These panoramas are aligned by capturing from the program with special viewing parameters.

We have developed a program to match the correspondence by allowing an user to identify matched triangles between the reference panoramas. During the matching process, the user picks a vertex of a triangle in one image and the corresponding epipolar line is shown in the other image. In Figure 6.2, the corner of a door is selected in the left image, and the corresponding epipolar line

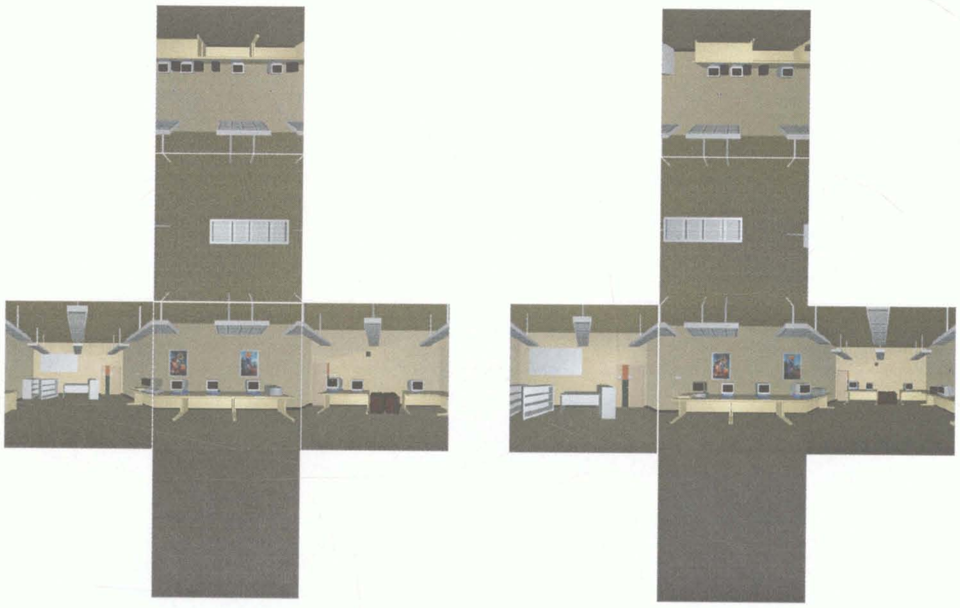
Figure 6.1: Two Aligned Panoramas in Cube Representation

is displayed in red in the other image. The corresponding point is then snapped to the epipolar line in the other image to facilitate the matching. The matched vertices are then stored.

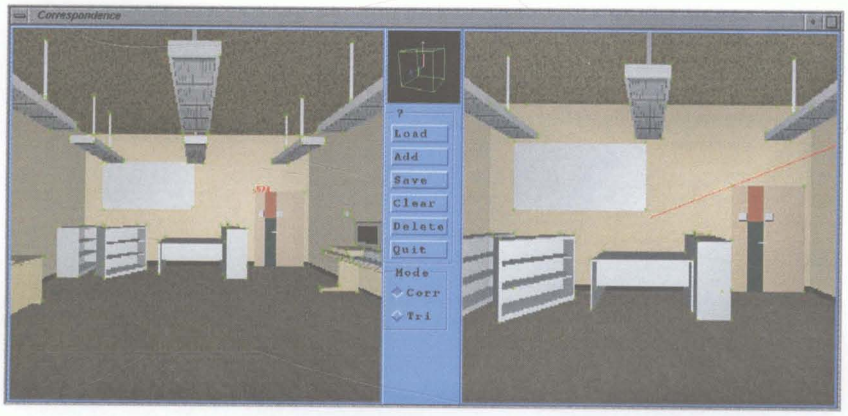

Figure 6.2: The Matching Program

In addition, we have developed another program to test the performance of the matching algorithm presented in Chapter 4. The test starts with a set of random polygons in the 3D space. These polygons are then projected onto two image planes to form two sets of projected regions. The matching score among these regions are then computed and the regions are matched with our matching algorithm. The test is performed with 100 pairs of polygons for 100 times.

The test program shows that we can find the exact matching with our matching algorithm. In order to test the stability of our algorithm, we intentionally introduce some random noise to the vertices of the regions before computing the matching. Figure 6.3 shows the result of the matching with noise introduced. We can see that the matching accuracy is above 98% when the magnitude of the noise is within five pixels, which means that less than two pairs of regions are mismatched out of the 100 pairs. The matching accuracy is around 90% when the noise is increased to ten pixels. The introduction of noise to the vertices is useful to simulate the real situation, in which the regions are extracted from images and therefore may be a bit different from the real projections.

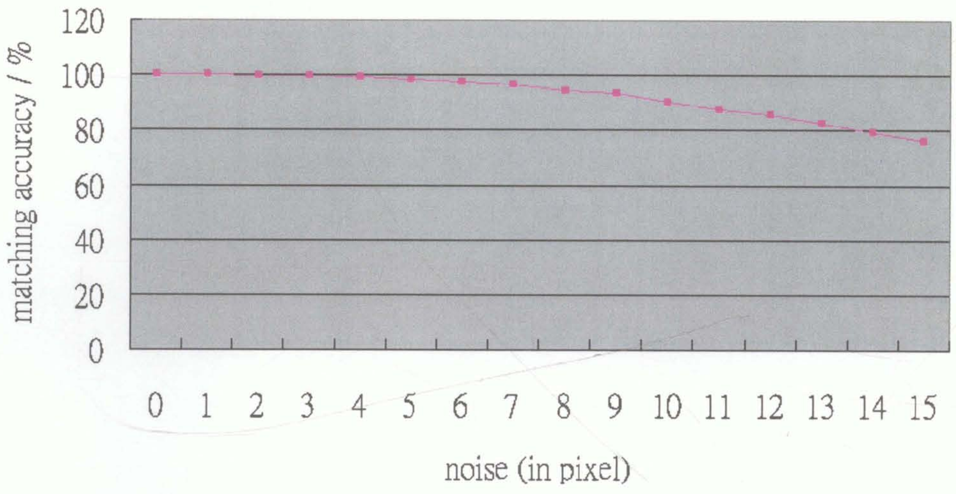

Figure 6.3: Matching Accuracy

In the rendering program, the user is allowed to move along the baseline by draging the mouse over the screen. Every time the user changes the view position, the vertices of the matched triangles are warped to computed the warped triangles with relative depth values. These warped triangles are then projected to form the new view. The triangles are rendered with texture from the reference panoramas. The texture coordinates are computed from the projective depth to ensure correct perspective projection. The user is also allowed to rotate by draging the other mouse button. Figure 6.4 shows the rendering program, in which the value of the interpolating parameter t is shown at the lower left corner.

The fold problem is solved by comparing the projective depth associated with each vertex of a warped triangle. The z-buffer is enabled during rendering. This is supported by the graphics hardware, so that the images are displayed in real time. This can also give a sense of motion parallax when the user is moving to

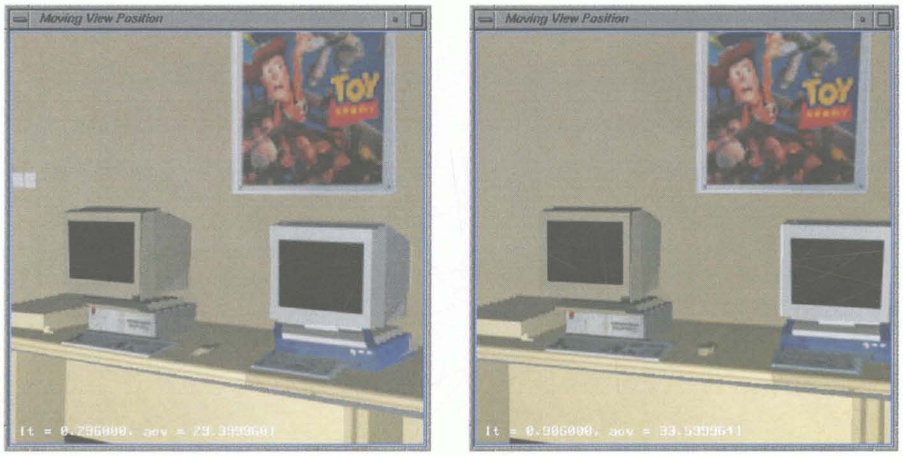

Figure 6.4: The Rendering Program

and fro along the baseline. Figure 6.5 shows a series of images generated when the view position is moving. We can see that the part of the door which was occluded by the monitor becomes visible when the view position changes. This shows that we can give a better perception of the depth of the environment by allowing the user to translate.

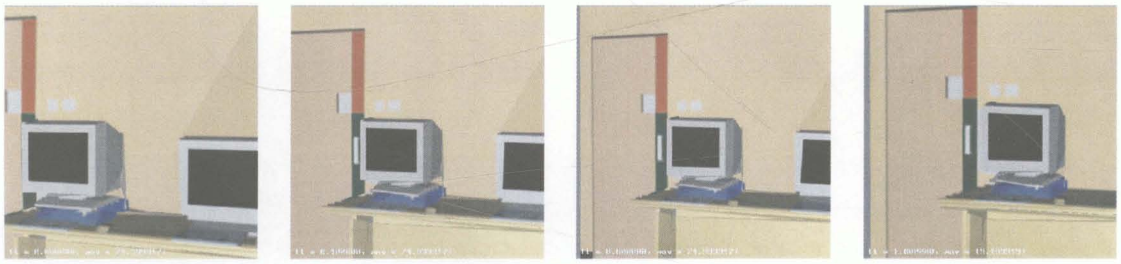

Figure 6.5: Motion Parallax

Chapter 7

Conclusions

This chapter will give a summary of this work and conclude the thesis with applications and possible future works.

7.1 Summary

The goal of this work is to develop an image based rendering system which allows the viewer to move freely within the environment with modest storage requirement. To reduce the complexity of our problem, we have reduced it to the problem of panorama interpolation, which is simpler to solve. We have further divided it into three sub-problems. Panorama alignment finds out the relative orientation between the reference panoramas. The aligned panoramas are then matched for establishing the correspondence between the reference panoramas. The correspondence is obtained by a region matching algorithms with a new matching score. Finally we perform a warping based on the correspondence information to generate the novel view at the new view position. The rendering is supported by graphics hardware.

In order to achieve the goal, we have to make some important decisions when designing the framework. We decide to take in images as input instead of using 3D models, thus the *modeling job* is avoided. We rely on the correspondence information to compute the warping, because this can greatly reduce the *storage requirement*, as we do not need to store all the visual information of the scene. Although the correspondence information and the depth information are equivalent, we prefer to use the correspondence information because this can be computed from the reference panoramas, while the depth information requires special hardware during image acquisition. We have chosen panorama as the reference image because we know that the output will also be a panorama. Therefore the field of view is not limited, and with panorama interpolation, the viewer will have the freedom to *move freely* within the environment. Finally, when computing the correspondence information, we have decided to match the regions instead of matching the pixels. This has the effect that the correspondence information can be represented by matched triangles, thus the novel view can be generated with graphics accelerators to ensure *real time rendering.*

In summary, we have developed a framework for an image based rendering system. The usefulness and practicability of this system will depend largely on how we can combine the advantages of various rendering methods. We have achieved our goal by creating a balance among different characteristics of rendering methods.

7.2 Applications

With panorama interpolation, we are able to reconstruct the novel panorama between two reference panoramas. As discussed in Chapter 2, from a sets of reference panoramas, we can obtain the novel panorama within the convex hull of the reference view positions by applying panorama interpolation a few times. This is useful for building a *virtual reality application* in which the viewer is allowed to freely navigate within a large environment.

On the other hand, panorama interpolation can be considered as a *compression technique* for ray based rendering. In ray based rendering, all the visual information has to be acquired and stored beforehand. With the help of panorama interpolation, we can store less information, but can still allow the view position to change.

Another possible application is to create the *panorama movie.* Panoramas are stored along some points on a path. The viewer is allowed to move along the path and rotate freely. View is generated with panorama interpolation depending on the view position and the view direction. The path can have branches, so that the viewers can choose to follow different paths. This application can be launched as an Internet application, with its interactivity and fast download time.

7.3 Future Work

This section suggests the possible ways to further improve our work.

Currently, our method only considers two reference panoramas each time during panorama interpolation. If more panoramas can be included in the interpolation, more information about the scene is available, hence the quality of the novel view will be higher. There are rendering techniques which involve multiple reference images [EVG96, AVI97]. Our method can also be improved in this aspect.

We have an assumption on the scene in which the scene is composed of Lambertian objects. This makes our matching algorithm simpler, but this assumption may be violated in some situations. Besides recovering the shape of the scene, there are studies about estimating the reflectance model of the scene from images [SAT97, YU98]. We can improve our method if this technique can be incorporated into our system, so that the correspondence matching can be more accurate and the warping will be valid with non-Lambertian scenes.

We have only implemented part of the whole system, which is mainly for testing purpose only. The ultimate goal is to build a complete system, from acquiring images to producing novel views. In addition, we can improve our implementation in various aspects like image registration and image segmentation, by incoporating some useful tools from compute vision and those well developed image processing techniques.

Bibliography

[AVI97] Shai Avidan, Theodoros Evgeniou, Amnon Shashua and Tomaso Poggio. Image-based View Synthesis. Research Report A.I. Memo No. 1603, Artificial Intelligence Laboratory, Massachusetts Instituteof Technology, January 1997.

[BEI92] Thaddeus Beier and Shawn Neely. Feature-Based Image Metamorphosis. *SIGGRAPH '92 Conference Proceedings*, page 35-42, 1992.

[BRO92] Lisa Gottesfeld Brown. A Survey of Image Registration Techniques. *ACM Computing Surveys*, Vol. 24, No. 4, December 1992.

[CHA98] Chun-Fa Chang, Gary Bishop and Anselmo Lastra. LDI Tree: A Hierarchical Representation for Image-based Rendering. *Technical Report 98-030*, Department of Computer Science, University of North Carolina at Chapel Hill, 1998.

[CHA99] Yan-Fai Chan, Man-Hong Fok, Chi-Wing Fu, Pheng-Ann Heng and Tien-Tsin Wong. A Panoramic-based Walkthrough System using Real Photos. *Pacific Graphics '99*, page 231-237, 1999.

[CHE93] Shenchang Eric Chen and Lance Williams. View Interpolation for Image Synthesis. *SIGGRAPH '93 Conference Proceedings*, page 279-288, 1993.

[CHE94] Yong-Qing Cheng, Victor Wu, Robert T. Collins, Allen R. Hanson, Edward M. Riseman. Maximum-weight bipartite matching technique and its application in image feature matching. Department of Computer Science, Lederle Graduate Research Center, University of Massachusetts.

[CHE95] Shenchang Eric Chen. QuickTime VR — An Image-Based Approach to Virtual Environment Navigation. *SIGGRAPH '95 Conference Proceedings*, page 29-38, 1995.

[CHE97] Qian Chen and Gerard Medioni. Image Synthesis From A Sparse Set of Views. *Proceedings of Visualization '97*, page 269-275, 1997.

[CRE00] 3D Blaster Annihilator2. http://www.3dblaster.com/products/annihilator2/. Creative Technology, Ltd.

[DEB96] Paul E. Debevec, Camillo J. Taylor, Jitendra Malik. Modeling and Rendering Architecture from Photographs: A hybrid geometry- and image-based approach. *SIGGRAPH '96 Conference Proceedings*, page 11-20, 1996.

[EVG96] Theodoros Evgeniou. Image based Rendering Using Algebraic Techniques. *Research Report A.I. Memo No. 1592*, Artificial Intelligence Laboratory, Massachusetts Instituteof Technology, November 1996.

[FAU93] Olivier Faugeras. *Three-Dimensional Computer Vision A Geometric Viewpoint.* The MIT Press, 1993

[FAU95] Olivier Faugeras, Stephane Laveau, Luc Robert. 3-D Reconstruction of Urban Scenes from Sequence of Images. *Research Report N2572*, INRIA, June 1995.

[FOL92] James Foley, Andries van Dam, Steven Feiner and John Hughes. *Computer Graphics, Principles and Practice*, Second Edition, Addison Wesley, 1992.

[FU98] Chi-Wing Fu, Tien-Tsin Wong and Pheng-Ann Heng. Triangle-based View Interpolation Without Depth-Buffering. *Journal of Graphics Tools*, Vol. 3, No. 4, 1998, page 13-31.

[FU99] Chi-Wing Fu, Tien-Tsin Wong and Pheng-Ann Heng. Computing Visibility for Triangulated Panoramas. *Proceedings of the 10th Eurographics Workshop on Rendering*, June 1999, page 169-182.

[GAL86] Zvi Galil. Efficient Algorithms for Finding Maximum Matching in Graphs. *Computing Surveys*, Vol. 18, No. 1, March 1986.

[GOR96] Steve J. Gortlet, Radek Grzeszczuk, Richard Szeliski and Michael F. Cohen. The Lumigraph. *SIGGRAPH '96 Conference Proceedings*, page 43-54, 1996.

[LAV94] Stephane Laveau and Olivier Faugeras. 3-D Scene Representation as a Collection of Images and Fundamental Matrices. *Research Report N2205*, INRIA, Feburary 1994.

[LEV96] Marc Levoy and Pat Hanrahan. Light Field Rendering. *SIGGRAPH '96 Conference Proceedings*, page 31-42, 1996.

[LUO96] Quan-Tuan Luong and Oliver D. Faugeras. The Fundamental Matrix: Theory, Algorithms, and Stability Analysis. *International Journal of Computer Vision, Vol 17, 1996*, page 43-75.

[MAN98] Russell A. Manning and Charles R. Dyer. Dynamic View Morphing. *Technical Report 1387*, Department of Computer Science, University of Wisconsin Madison, September 1998.

[MCM95a] Leonard McMillan. Acquiring Immersive Virtual Environments with an Uncalibrated Camera. *Technical Report 95-006*, Department of Computer Science, University of North Carolina at Chapel Hill, 1995.

[MCM95b] Leonard McMillan and Gary Bishop. Plenoptic Modeling: An Image-Based Rendering System. *SIGGRAPH '95 Conference Proceedings*, page 39-46, 1995.

[MCM95c] Leonard McMillan. Computing Visibility Without Depth. *Technical Report 95-047*, Department of Computer Science, University of North Carolina at Chapel Hill, 1995.

[OHT85] Yuichi Ohta and Takeo Kanade. Stereo by Intra- and Inter Scanline Search Using Dynamic Programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 7, No. 2, March 1985.

[SAT97] Yoichi Sato, Mark D. Wheeler and Katsushi Ikeuchi. Object Shape and Reflectance Modeling from Observation. *SIGGRAPH '97 Conference Proceedings*, page 379-387, 1997.

[SEG92] Mark Segal, Carl Korobkin, Rolf van Widenfelt, Jim Foran, Paul Haeberli. Fast Shadows and Lighting Effects Using Texture Mapping. *SIGGRAPH '92 Conference Proceedings*, page 249-252, 1992.

[SEI96] Steven M. Seitz and Charles R. Dyer. View Morphing. *SIGGRAPH '96 Conference Proceedings*, page 21-30, 1996.

[SEI97] Steven M. Seitz and Charles R. Dyer. Photorealistic Scene Reconstruction by Voxel Coloring. *Proc. Computer Vision and Pattern Recognition Conf.*, page 1067-1073, 1997.

[SHA98] Jonathan Shade, Steven Gortler, Li-wei He and Richard Szeliski. Layered Depth Images. *SIGGRAPH '98 Conference Proceedings*, page 231-242, 1998.

[SHU99] Heung-Yeung Shum and Li-Wei He. Rendering with Concentric Mosaics. *SIGGRAPH '99 Conference Proceedings*, page 299-306, 1999.

[SZE97] Richard Szeliski and Heung-Yeung Shum. Creating Full View Panoramic Image Mosaics and Environment Maps. *SIGGRAPH '97 Conference Proceedings*, page 251-258, 1997.

[WAT92] Alan Watt and Mark Watt. *Advanced Animation and Rendering Techniques.* Addison Wesley, 1992.

[WOO99] Mason Woo, Jackie Neider, Tom Davis and Dave Shreiner. *OpenGL Programming Guide Third Edition.* Addison Wesley, 1999.

[XIO97] Yalin Xiong and Ken Turkowski. Creating Image-Based VR Using a Self-Calibrating Fisheye Lens. *Computer Vision and Pattern Recognition '97 Proceedings*, page 237-243, 1997.

[YU98] Yizhou Yu and Jitendra Malik. Recovering Photometric Properties of Architectural Scenes from Photographs. *SIGGRAPH '98 Conference Proceedings*, page 207-217, 1998.