Computer Science Department of The University of Auckland
CITR at Tamaki Campus (http://www.citr.auckland.ac.nz)

CITR-TR-42 March 1999

Three-dimensional View Synthesis from Multiple Images

Shou-kang Wei

Abstract

We develop a view synthesis system, which is capable to collect/reconstruct 3-D information of scenes from multiple calibrated/uncalibrated images; to store and access the 3-D information in an efficient way; and to synthesize, in real time, arbitrary views of the scene with a pre-specified viewing area.

1.CITR, Tamaki Campus, University of Auckland, Private Bag 92019

THREE-DIMENSIONAL VIEW SYNTHESIS FROM MULTIPLE IMAGES

By
Shou-Kang,Wei

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
AT
THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
NOVEMBER 1998

Abstract

We develop a view synthesis system, which is capable to collect/recover 3-D information of scenes from multiple calibrated/uncalibrated images; to store and access the 3-D information in an efficient way; and to synthesize, in real-time, arbitrary views of the scene within a pre-specified viewing area. We propose a new scene representation for our view synthesis system. Consequently, it is designed to be expressive enough for storing the reconstructed data of complicate real scenes, yet simple enough for the fast rendering.

To found the basis of image transformations for various applications in our system, all possible image mappings between two logical image forms (planar and cylindrical) are established by deriving the corresponding mapping equations. However, the mapping equation alone is insufficient to render a view because multiple scene points may map to a single position of the desired image in an arbrary order. Our solution for the visibility determination is inspired and derived by the geometric observations and the existing approach [1]. Besides, our approach achieves the constant time complexity, O(1).

Fully automatic depth-recovery algorithms have not yet reached the level that allows us to obtain dense and accurate depths independently from the scene complexity. Instead of trapping ourselves into such short-term unsolvable problems, we construct a development system, where the human intervention is adopted, that is capable to effciently recover a dense depth map with acceptible quality.

Chapter 1

Introduction

It is intriguing to consider one day in the future when the technology may exist that we can freely travel all over the world in our own pace and preferences without physically being there. Our high-expectations are already being set by the depiction of such virtual navigation system in the various works of science fiction. The technological realities, however, present many problems to be solved before these fantasies could be realized.

3-D scene generation and navigation were investigated for many years in both computer graphics and computer vision communities. Computer graphics considers mainly the problem of synthesizing images from specified geometric models. Great effort has been made to develop computer aided design (CAD) systems that allow us to create realistic images by modeling complex scene geometry and material attributes as well as by simulating the light propagation through virtual environments. In spite of this effort, it is still difficult, if possible, to replicate much of the complex geometry and subtle lighting effects found in the real world.

Computer vision considers the problem in the opposite direction. The geometric models are synthesized from multiple images. Given the 2-D projection of a point in the world, its position in 3-D space could be anywhere along a ray propagating in a particular direction from the camera's optical center. However, when the projections of a sufficient number of 3-D points are observed in multiple images taken from different positions, it is theoretically possible to deduce the 3-D locations of the points as well as the relative positions of the original cameras, up to an unknown scale factor (cf. Appendix C). The efforts of computer graphics and computer vision are generally considered as complementary because the results of one field can frequently serve as an input to the other. Computer graphics often looks to the field of computer vision for the generation of complex geometric models, whereas computer vision relies on computer graphics for viewing results or testing algorithms with synthetic images.

1.1 Motivations and Goals

One of the primary subjective measures of image quality is the extent to which the rendered image is indistinguishable from a photograph. Needless to say, using photographs as the underlying scene representation inherits intrinsic structure and texture of real scene naturally in a way that is currently beyond capabilities of geometric modeling. If one wants to render a 3-D scene by computing the shading of composed scene objects, given surface properties of the objects and positions of light sources, then the 3-D model of the objects has to be very accurate due to the fact that the computed shading is very sensitive to 3-D noise. On the other hand, when rendering a 3-D scene with surface textures directly extracted from the real images, the quality of the rendered image is more tolerable to inaccurate 3-D data.

Recently a new approach to rendering has emerged: image-based rendering [2, 3, 1, 4, 5, 6, 7, 8]. The image-based rendering systems generate different views of a scene using digitized photographs or rendered images (for increasing the frame rate), and their corresponding depth maps. The key observation in the image-based rendering is that when the depth of every point in an image is known, the image can be re-rendered from any nearby point of view by projecting the pixels of the image to their proper 3-D locations and back-projecting them into a desired image (cf. Chapter 4). A principal attraction of the image-based rendering is that it offers a method of rendering arbitrarily complex scenes with a constant amount of computation required per pixel. Thus it is suitable for a real-time implementation on workstations and personal computers to produce a virtual environment.

Our goal is to develop an image-based system for the view synthesis. The system should be capable to collect/recover 3-D information of scenes from multiple calibrated/uncalibrated images; to store/access the 3-D information in an economical/fast manner; and to render in real-time the arbitrary view(s) of a virtual scene, within a pre-specified viewing area.

1.2 Related Work

A number of techniques have been proposed for flying through scenes by redisplaying previously rendered or digitized views. Some techniques have also been proposed for interpolating between views by warping input images, using depth information or correspondences between multiple images. Below we give an overview of previous works that are partially or conceptually related to our system.

Apple's QuickTimeVR system [2] was one of the first systems to suggest that the traditional modeling/rendering process can be skipped. The multi-nodes version of this system creates environment maps at key locations in a scene. The user is able to discretely navigate from location to location (jumping in between), and to continuously changes the viewing direction while at each location. Many overlapping images of the scene are taken and then stitched together. The simplest stitching occurs when the camera motion includes only rotation. In this case the transformation between the views is parametric and does not include any 3-D shape information. R. Szeliski and S. Kang [9] create high-resolution mosaics from low-resolution video streams, and S. Peleg and J. Herman [10] relax the fixed camera constraint by introducing the projection manifold. The forerunner to these techniques is the use of environment maps to capture the incoming light in a texture map. An environment map records the incident light arriving from all directions at a point. The original use of environment maps was to efficiently approximate reflections of the environment on a surface. However, environment maps may also be used to quickly display any outward looking view of the environment from a fixed location but at a variable orientation. A drawback of these approaches is that one cannot correctly simulate a translational camera motion from a set of model images.

S. Laveau and O. Faugeras [3] were the first to make use of the epipolar constraint for view synthesis, allowing views to to extrapolated as well as interpolated, between the reference images. Epipolar constraints, however, are subject to singularities that arise under certain camera motions, for example, when the virtual camera optical axis is collinear with the line joining centers of the base cameras. L. McMillan and G. Bishop [1] use a full depth map together with the epipolar constraint to provide a direct connection between the virtual camera motion and the rendering engine. The epipolar constraint is somewhat indirect and hence requires the specification of matching points.

Image interpolation is designed to create in-between images among two or more reference images. This includes image morphing [4], direct interpolation from image-flows [11, 12], image interpolation using 3-D models instead of image-flow [13], and "physically correct" image interpolation [5, 6]. All but the last two do not guarantee to produce physically correct images, and all cannot extrapolate from the set of input images. Instead of the flow-field interpolation among the reference images, it is possible to interpolate directly over the plenoptic function [14]. E. H. Adelson and J. R. Bergen assigned the name, plenoptic function, to the set of rays visible from any point in space, at any time, and over any range of wavelengths, which means all of the radiant energy received at the viewpoint of an observer rather than at an object or an illumination source. M. Levoy and P. Hanrahan [7] and Gortler et al. [8] interpolate between a dense set of rays derived from several thousands of reference images to reconstruct a reduced plenoptic function under an occlusion-free world assumption. Hence, they considerably increase the number of input images to avoid computing image-flow between the reference images.

1.3 Overview of Our View Synthesis System

Our system comprises three phases: image acquisition, scene reconstruction and rendering. Each phase consists of several processes. The processes may also contain sub-processes or a sub-system (e.g. depth

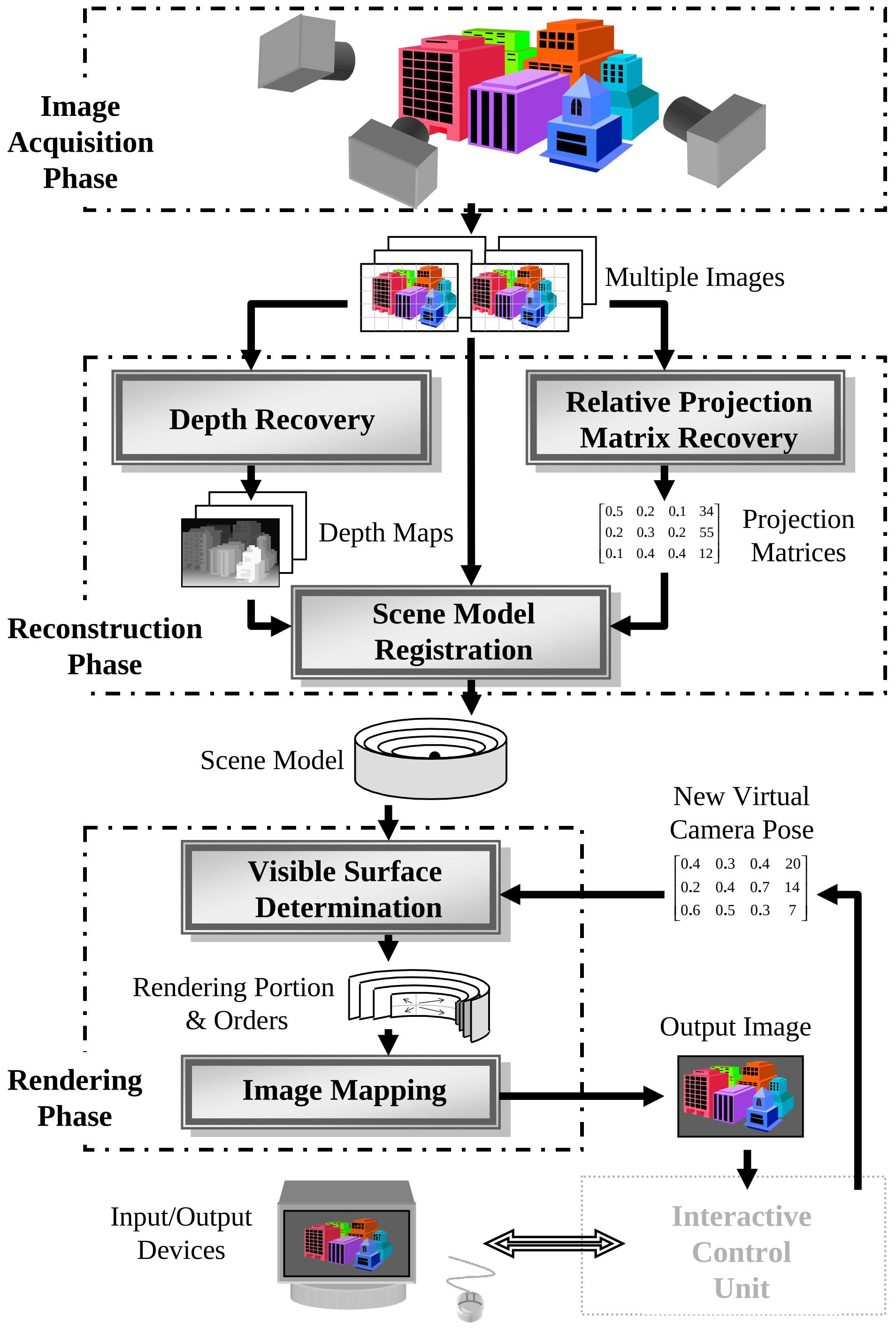


Figure 1.1: The architecture of our view synthesis system.

recovery development system cf. Chapter 6). The system is organized as pipeline structure (an output of one process becomes input of the other). So it is extendible and expandable to meet various requirements of the specific applications, in particulars, for crime or underwater scene visualizations. Consequently, as technology becomes more and more advanced, one or more processes/sub-processes can be replaced by new technologies without rebuilding the whole system. Figure 1.1 shows the main structure of the system.

In the image acquisition phase, the camera model can be a setup of acquiring stereo panoramic images [15], or acquiring standard binocular stereo images [16], or both. After digitization, input images are sent to the reconstruction phase. For each binocular stereo pair, a depth map associated with either of the pair (we choose the right image) can be obtained through our depth recovery development system (cf. Chapter 6). Since the input images are assumed to be uncalibrated, the geometric relations among those images are retrieved through the relative camera projection matrices recovery process (cf. Appendix A, B, C) . The scene model registration process collects previous recovered information as well as the input images into our scene model, using image mapping equation(s) (a planar-to-cylindrical or a cylindrical-to-cylindrical image mapping equation, cf. Chapter 4). Once all input image data are registered into our scene representation, the reconstruction phase is complete.

Navigation to the 3-D scene is done interactively with the user input(s), for example the desktop application through a mouse device. Users are allowed to move and turn around in the 3-D space within a pre-specified region, similar to walk-through in the real world. Two modes MOVE and TURN are specified and switched by a particular key on the keyboard. The indication of user movement is transmitted to our rendering engine, and a desired view corresponding to the new virtual viewing pose is generated. The interactive control unit coordinates the events triggered by input/output devices and the rendering engine. Since our system makes no special request to the control unit and interactive scheme(s) used in the recent PC or workstations are sufficient for our application, we will not give further discussion on it in the remainder of this thesis.

During the rendering phase, two major processes are performed. The first is the visible surface determination. It determines which portion of data stored in the scene representation is interesting to the new view, and computes a specific traverse order such that the visible surfaces are determined correctly in the new view (cf. Chapter 5). The second is the image forward-mapping which maps each scene data of the interesting portion to the desired view (cf. Chapter 4). The illusion of the walk-through in a real 3-D space would be maintained by continually mapping the images to correspond to the user's changing perspective.

Two image reconstruction techniques, splatting and polygonal texture mapping, are provided for on-line and off-line rendering. The splatting process estimates an arrived (mapped) point size on the resulting image for each scene data mapping, so the potential gaps which may appear in the resulting image are logically "filled-up" or at least shrunk noticeably. Because it is fast (a look-up table), it can be used for a real-time implementation. However, the higher the rendering speed, the lower the quality of resulting images. Thus, for the off-line applications the polygonal texture mapping is used to ensure an adequate image quality. It regards each reference pixel as a mesh polygon defined by four corners of the pixel. Instead of mapping a single pixel to the desired image, it maps four corners of reference pixel onto the desired image and forms a polygon filled by an average color of the four. Unlike W. R. Mark's approach [17], our approach is based on the inter-relations between foreground and background objects established in the reconstruction phase (cf. Chapter 6), the expensive connectedness computations for the inter-polygon relations are therefore avoided.

1.4 Organization of Thesis

The thesis provides a complete description of a 3-D scene reconstruction and display system. Each chapter addresses one or more issues of the processes in the system. Chapter 2 provides a few preliminaries that will be used in subsequent Chapters. In Chapter 3, we first investigate some existing 3-D scene representations. Particularly, the scene representations used in other image-based approaches are analyzed. Our scene representation, a key to the realization and success of the system, and the performance analysis of it are elaborated in the second and third part of Chapter 3.

Chapter 4 and 5 cover up the topics of processes in the rendering phase, which discuss a few major progresses we make to the performance of the rendering. In Chapter 4, we derive image mapping equations for all the image transformations required in our view synthesis system. This establishment permits such possibility that multiple images can be registered to as well as novel views can be synthesized from our scene representation. Furthermore, the optimization of the image mapping, a quality loss-less speedup, is also explored and explained at the end of Chapter 4.

Chapter 5 solves the visibility problem for the real-time rendering implementation. We interpret visibility problems using epipolar geometry and then explore L. McMillan's solution for the visibility determination [1]. Our solution points out those cases where his approach fails and fixes the problems. Besides, the formulation and analysis of the culling process, which filters out the needless scene data in our scene representation before other rendering processes carried out, are given in the last part of Chapter 5.

Chapter 6 describes a complete depth recovery development system that is capable to generate dense disparity maps. The motivations of building such system and a blueprint of an underlying system are elaborated. The experimental results for each intermediate development step in the system are shown. Chapter 7 concludes this thesis by pointing out the advantages, limitations and future directions of the current system.

Appendix A introduces an invariant of the epipolar geometry, the fundamental matrix. The definition and properties of such invariant are stated formally. Appendix B describes a modern approach based on the algebraic structure and numerical computation to recover the fundamental matrix from a given stereo pair. Finally, the use of the fundamental matrix to recover the relative camera projection matrices is explained in Appendix C.

Chapter 2

Preliminaries

2.1 Notations

We follow the convention that a matrix and a vector (a point) are in bold-faced font, and a scalar or a constant number is in italic font. An arbitrary point in 3-D space is denoted as $\mathbf{P}$, and its projection on an image is denoted as $\mathbf{p}$. The camera optical center, the origin of the camera coordinate system, is denoted as $\mathbf{C}$. The image is denoted as E. The image plane that describes the planar image model is referred to as I, and the image cylinder as J.

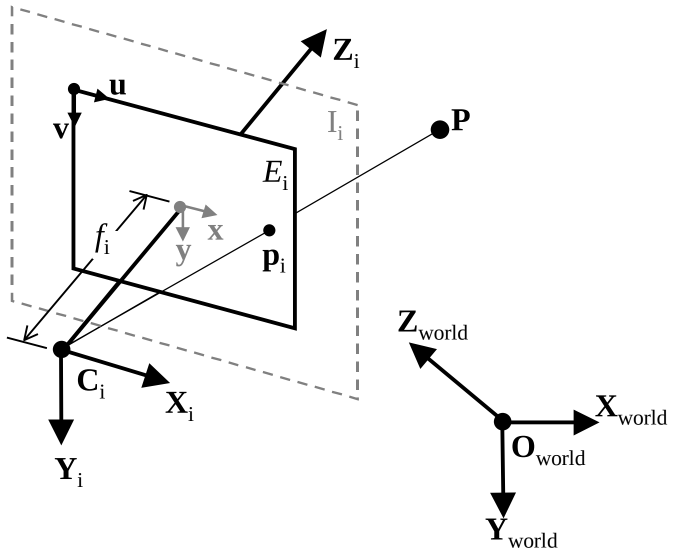


Figure 2.1: Perspective camera geometry in world coordinate system.

Every symbol may come with a subscript, which gives connection between different symbols while multiple images are considered at a time. For example, camera optical center $\mathbf{C}_1$ should associate with the image E_1, and $\mathbf{C}_2$ with E_2, etc. We depict these notations in Fig. 2.1.

The *camera projection matrix* specifies a transformation from 3-D to 2-D, and it is used to map a space-point to an image-point. It describes internal camera parameters, e.g. focal length, image unit length; and external pose with respect to the world coordinate system, e.g. orientation, translation. A perspective projection matrix, denoted as Π, is a 3×4 matrix in the following form, $\Pi = \left[\mathbf{H} \middle| -\mathbf{HC}\right]$[1] or simply

[1]The notation $[\cdot|\cdot]$ means the concatenation of matrices from both sides of the vertical bar.

$\Pi = \left[\mathbf{H}\middle|\mathbf{h}\right]$. The camera optical center $\mathbf{C}$ is a 3-vector with respect to the world coordinate system. $\mathbf{H}$ is a 3×3 matrix composed as $\mathbf{H} = \mathbf{AR}$. The matrix $\mathbf{A}$ is the camera intrinsic matrix containing all the intrinsic parameters of the camera, and the matrix $\mathbf{R}$ describes the camera's orientation with respect to the world coordinate system.

2.2 Epipolar Geometry

Epipolar geometry is a fundamental geometric concept applicable in many computer vision problems, such as stereo analysis, motion detection, and object recognition [18, 19, 20]. In the next chapters, it is also found useful to interpret some problems in a manifest and comprehensive manner (e.g. visibility analysis, relate camera projection matrices recovery).

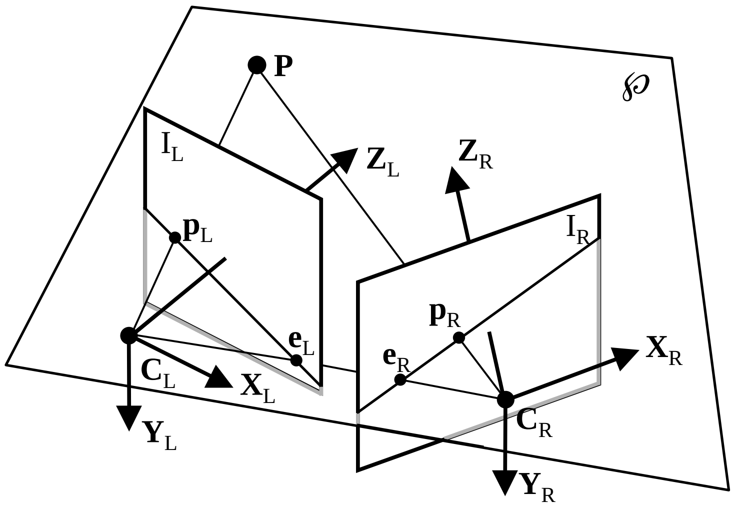


Figure 2.2: Epipolar geometry of a stereo pair.

For any pair of images, their camera optical centers $\mathbf{C}_L$, $\mathbf{C}_R$ plus a space point $\mathbf{P}$ define a plane in 3-D space called *epipolar plane*, denoted by $\wp$, as illustrated in Fig. 2.2. The projections of $\mathbf{C}_L$ and $\mathbf{C}_R$ on the image planes I_R and I_L are called *epipoles*, denoted by $\mathbf{e}_R$ and $\mathbf{e}_L$, respectively. The lines joining $\mathbf{p}_i$ and $\mathbf{e}_i$ are called *epipolar lines*. An implicit property given by Fig. 2.2 is that all 3-D points lying on the same epipolar plane are projected onto the same epipolar line of each image plane. Thus, the corresponding points between two reference images are constrained to sit on the epipolar lines. Moreover, as shown in Fig. 2.3, we can extend this to infinite number of epipolar planes, $\mathrm{U} = \{\wp_i : i = 1, 2, \ldots, \infty\}$, intersecting the line joining two camera optical centers $\mathbf{C}_L$ and $\mathbf{C}_R$.

2.3 Standard Binocular Stereo Geometry

If two image planes are coplanar and the x-axes of their associated camera coordinate systems are collinear, then it is called *standard binocular stereo geometry*, as shown in Fig. 2.4, where the epipoles lie at infinity and the effective focal lengths of both camera are the same. The distance between two camera optical centers $\mathbf{C}_L$ and $\mathbf{C}_R$ is called *base distance*, denoted by b.

The major advantage using epipolar constraints concerning with the stereo correspondence analysis problem is effectively eliminating the search space from 2-D to 1-D. In addition, the binocular stereo geometry further simplifies the 1-D searching path from an arbitrary image epipolar line to a regular image row, being in this case the epipolar line. The correspondence search is therefore performed on the same row of two stereo images in a scan-line fashion.

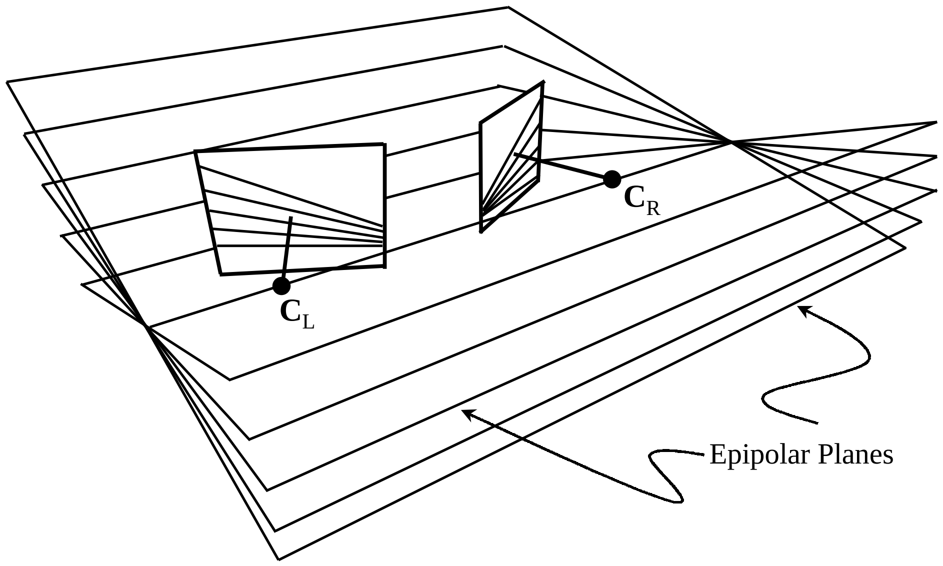


Figure 2.3: The abstraction of infinite number of epipolar planes commonly intersect the line joining two camera optical centers $\mathbf{C}_L$ and $\mathbf{C}_R$.

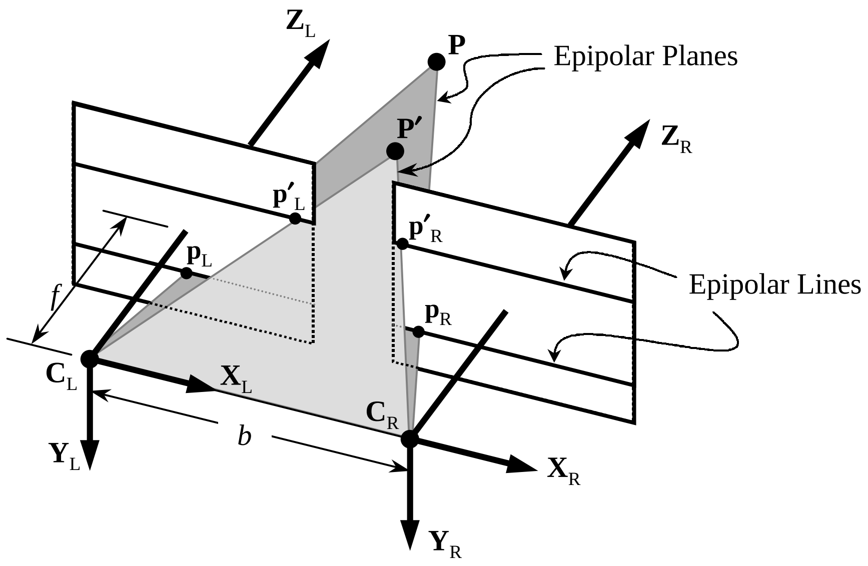


Figure 2.4: The geometric interpretation of spatial relations between the depth and disparity in the standard binocular stereo geometry.

2.4 Disparity to Depth

Disparity can be interpreted as the difference between two corresponding points with respect to the x- and y-axis in image coordinate systems. Let two corresponding points in the left and the right image be (x_{p_L}, y_{p_L}) and (x_{p_R}, y_{p_R}) in the standard binocular stereo case. Then we have $y_{p_L} = y_{p_R}$ and x_{p_L} is always greater or equal to x_{p_R}, so disparity, denoted as d_P, is simply x-disparity

$$d_P = x_{p_L} - x_{p_R}. \tag{2.4.1}$$

A *depth* associated with an image pixel describes the shortest distance between its corresponding 3-D point and a plane parallel to the image plane through the camera optical center. It is the Z-component of a 3-D point with respect to the camera coordinate system,

$$Z = \frac{bf}{d_P},$$

where b is the base distance (distance between $\mathbf{C}_L$ and $\mathbf{C}_R$) and f is the effective focal length of both cameras.

What is the relationship between pixel's depth (we want) and disparity (we have)? Considering Fig. 2.4, it is easy to understand that $Z_P \geq Z_{P'}$ implies $d_P \leq d_{P'}$. Formally, we may assume that the right camera coordinate system is coincident with the world coordinate system, so we have

$$x_{p_R} = \frac{Xf}{Z},\ x_{p_L} = \frac{(X+b)f}{Z}, \quad \text{and} \quad y_{p_R} = y_{p_L} = \frac{Yf}{Z}.$$

Thus, the world coordinates of a point $\mathbf{P}$ is as follows,

$$\mathbf{P} = (X_P, Y_P, Z_P)^T = \left(\frac{bx_{p_R}}{d_P}, \frac{by_{p_R}}{d_P}, \frac{bf}{d_P}\right)^T. \tag{2.4.2}$$

It implies that the depth of a pixel is inversely proportional to its disparity since both b and f are constant values. If the depth values of an image E_i are stored in a depth map $\mathcal{D}_i$, then the function value $\mathcal{D}_i(u_{pi}, v_{pi})$ gives the depth of an image pixel $\mathbf{p}_i = (u_{pi}, v_{pi})$.

2.5 Depth Conversion for Cylindrical Images

The depth of a cylindrical image pixel means the shortest distance between the relevant 3-D point and the y-axis of the cylindrical camera coordinate system, which is different from the definition for a planar image. When warping a planar image into a cylindrical image, the depth map of the planar image must be transformed into a cylindrical depth map.

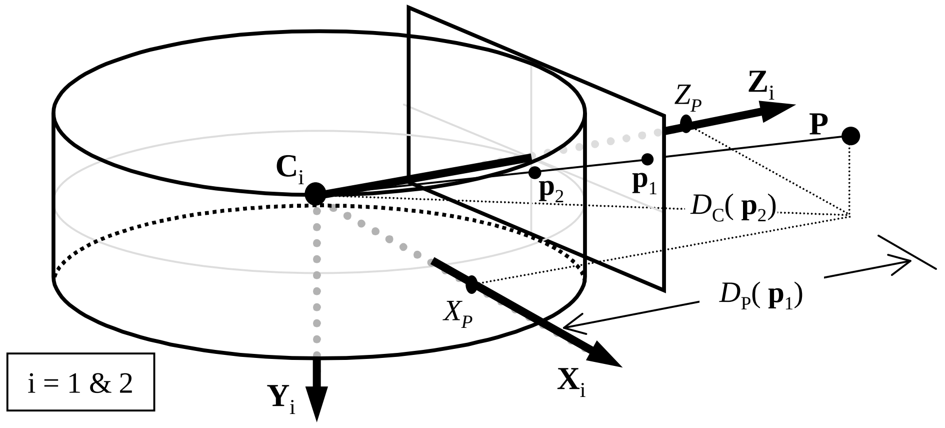


Figure 2.5: The geometric interpretation of the depth conversion from a planar to a cylindrical image.

The relationship between the depth values of a 3-D point $\mathbf{P}$ with respect to the planar and cylindrical camera coordinate systems is illustrated in Fig. 2.5. Let $\mathcal{D}_P(\mathbf{p}_1)$ denote the depth value of the image point $\mathbf{p}_1 = (x_{p_1}, y_{p_1})$ in the planar image, and $\mathcal{D}_C(\mathbf{p}_2)$ denote the depth value of the image point $\mathbf{p}_2$ in the cylindrical image. We already have

$$\mathcal{D}_P(\mathbf{p}_1) = Z_P = \frac{bf}{d_P},$$

where d_P is the disparity value for the point $\mathbf{p}_1$ in a planar image, b is the base distance and f is the effective camera focal length. The depth value for the cylindrical image can be computed as follows,

$$\mathcal{D}_C(\mathbf{p}_2) = \sqrt{X_P^2 + Z_P^2} = \sqrt{\left(\frac{bx_{p_1}}{d_P}\right)^2 + \left(\frac{bf}{d_P}\right)^2} = \frac{b}{d_P}\sqrt{x_{p_1}^2 + f^2} = \frac{Z_P}{f}\sqrt{x_{p_1}^2 + f^2}.$$

Thus,

$$\mathcal{D}_C(\mathbf{p}_2) = \mathcal{D}_P(\mathbf{p}_1)\sqrt{\left(\frac{x_{p_1}}{f}\right)^2 + 1}.$$

This equation shows the conversion from the depth of a planar image pixel to the depth of a cylindrical image pixel.

Chapter 3

Scene Representation

The key element of the view synthesis system is a proper scene representation. It should be capable to describe the scene structure expressively and store the reconstructed data effectively. Additionally, considering the role it plays in the real-time implementation, an efficient access to the stored scene data is a critical issue to its performance in the application. Furthermore, as more and more range data become available, merging the new data with the existing data should be easy to perform.

There exist many scene representations because different applications suggest different design issues. It is important to understand and analyze those approaches in the first place. In this chapter, we investigate a few existing representations with selected examples based on image-oriented representations. Then, we propose a new scene representation for our view synthesis system. The ideas, design issues, and performance analysis are presented in details. Consequently, we focus on the functionality and contributions of the proposed scene representation to our view synthesis system, as well as reason why it can be characterized as expressive enough for storing the reconstructed data of complicate real scenes, yet simple enough for the fast rendering.

3.1 Existing Representations

Point-based representation is a basic 3-D scene representation, which can be expressed in various coordinate systems, such as Cartesian, polar or stereographic [16] coordinate systems depending on application. Other geometric representations, *mesh-based* and *patch-based*, are also commonly used in both computer vision and computer graphics communities. A mesh, or a planar surface unit, is normally defined by three or four points in a pre-specified order. A patch, or a curved surface unit, is a more sophisticated primitive for scene description.

Besides, there is a volumetric primitive, called *voxel*, that can be interpreted geometrically as a solid polyhedron in 3-D space. Conventionally, it is used for visualizing internal structure of a volumetric object from a set of image slices generated either by image acquisitions or simulation-generated. Recently, the voxel representations for view synthesis, such as [21, 22], have been receiving growing interests.

Considering input data as a set of images, the scene representation is intuitively close to the point-based representation or the voxel-based representation, where each point/voxel is associated with an image pixel. Contrarily, the mesh- and the patch-based representations introduce at least one undesired problem, namely, that all foreground-background objects of the scene in the images are adhered together. Then it is very difficult to separate them automatically if high-level knowledge, e.g. human cognition, is not involved into the process.

One remarkable example using voxel-based representation is the *inverse polar octree* [22]. It utilizes the characteristics of spatial relation that the resolutions are inversely proportional to the distance between a viewpoint and an observed object in the projective space. The unbounded 3-D data is then transformed into bounded 3-D space and into uniform quantization. Figure 3.1 shows the illustration of the inverse polar octree and Fig. 3.2 depicts the inverse polar transformation.

Recently, image-based techniques have received increasingly attentions from computer graphics and

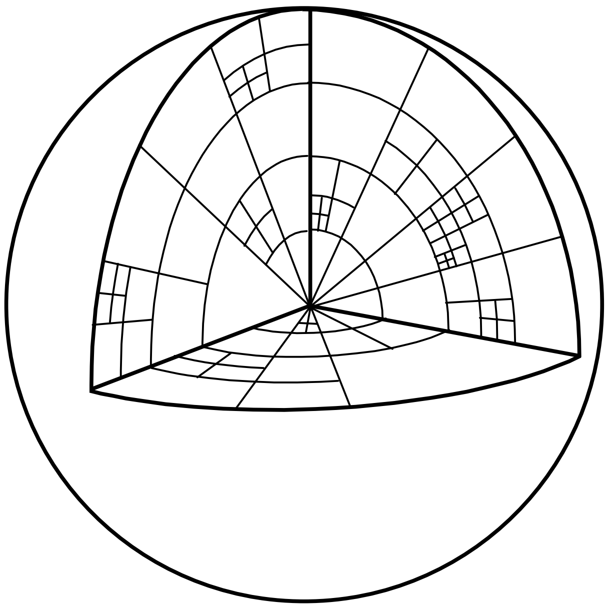


Figure 3.1: Abstraction of inverse polar octree representation.

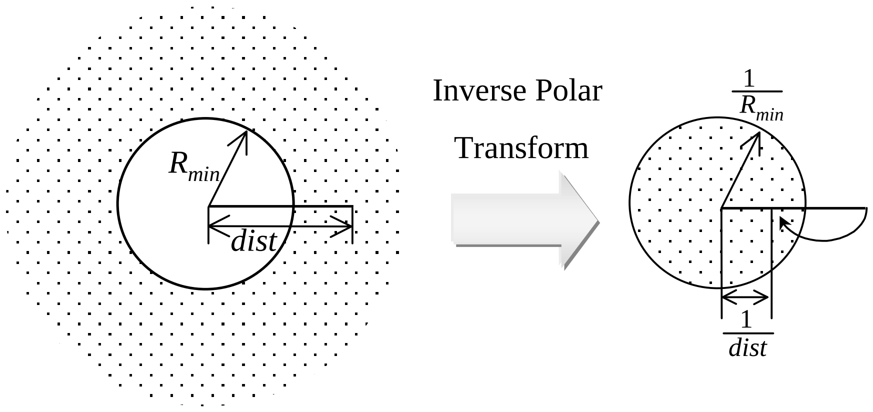


Figure 3.2: Inverse polar transformation (*dist* denotes the distance between a viewpoint and an observed point and R_{min} represents a minimum distance away from a viewpoint with respect to the closest observed point in the scene).

computer vision communities. The techniques concern input data uniquely from one or more images. The primary intention is to reuse those finite data, from either synthetic or real images, to compute desired images without actually worrying about underlying 3-D scene structures. One apparent scene representation is the image itself. S. Laveau and O. Faugeras [3] first demonstrated the potentiality of 3-D scene representation as a collection of images. Other recent reconstruction techniques [6, 23] also use images as their representations exclusively.

However, transforming images into a logical geometric form may produce desired result(s) instantly. A simple example is Apple's QTVR. S. E. Chen [2] used the image warping, stitching and blending techniques to transform multiple mosaics into a single seamless cylindrical representation, called panoramic image. Another commercial product, PhotoVista, is implemented in a similar way with two additional geometric forms, spherical and cubic representations. Scene reconstruction processes based on this simple representation are therefore simplified, because neither 3-D data nor depth information is needed for rendering the scene. The results are promising, because the image transformations can be performed uniformly and independently from any extent of scene complexity. Other related works based on this idea can be found in [1, 24, 25].

M. Levoy and P. Hanrahan [7] have described another representation in which the underlying model-primitives are rays rather than images. They combined and interpreted huge arrays of rendered/acquired images into 2-D slices of a 4-D function. All of the rays that pass through a slab of empty space, which is enclosed between the two slices, can be described using only four parameters (a 2-D coordinate for each slice). Such ray-based representation sets the scene reconstruction processes free from recovery of 3-D data (or depth), as well as free from the fragileness of image registrations (which is part of S. E. Chen's approach). On the other hand, it misuses the human resource and the data storage severely during the images acquisition and the digitalization processes (thousands or more images required). It will continue to be impractical in the near future. The similar representation can also be found in [8] approach.

3.2 Depth-Layered Cylinder Representation

As reflected from the given name, depth-layered cylinder representation (DLC in short), our scene representation is intended to provide a full-view with additional capability to store scene data layered by depth information.

Let us recall from S. E. Chen's cylindrical model. He considered all the 3-D scene points observed from a single viewpoint. However, if multiple panoramic images are available, his solution to the *walk-through* (transition of movement between viewpoints) is "jumping" in-between. Even a view with a slight displacement from the original viewpoint is not allowed in his approach. Furthermore, if additional images are available, such as widely available planar images, his model allows new data nowhere to contribute to.

We simply extend his model (cylindrical representation) to a unified representation (depth-layered cylinder representation) that additional range data can easily be integrated into. Let us consider a cylinder in 3-D space, where the center of the cylinder can be defined as a projection center emitting cylindrically to the 3-D world. Each projection point/grid on the cylinder defines one class of 3-D space points, lying on/in a projection ray/frustum from the cylinder center to infinity. Each classifier, a projection point/grid on the cylinder, is associated with a linked list. Reconstructed scene data classified into the same class are stored in the associated linked list and sorted by depth. Each element of the list contains a depth, colors (ARGB, 4 bytes) and pointers to the previous and the next element. The concept of our scene representation can be visualized in Fig. 3.3.

Resolutions of the DLC, theoretically, can be of arbitrary fineness. Practically, however, with this model we should only consider integrating 3-D scene information available from its spatial neighborhoods to certain extent, instead of doing globally. The reason is to take advantages of the efficiency, in terms of both data access and storage, provided by the localization scheme. So, the DLC can be thought as compressed version of S. E. Chen's model, where the redundant scene information from a few nearby panoramas is largely eliminated. Besides, this representation relaxes S. E. Chen's fixed-viewpoint problem, allowing arbitrary view(s) of a virtual scene within a pre-specified viewing area.

One intuitive way of setting up our scene representation is choosing an existing panorama to coincide with the DLC, normally a centroid of multi-panoramas topology. However, if no panorama is given, our model can still be set to one of existing images and allows image registrations from others. One intrinsic feature of this model is its extensibility, the more data integrated into it the better desired views can be obtained, i.e. less gaps to appear in the resulting images.

3.3 Performance Analysis

The role of our scene representation in the view synthesis system is a bridge between the reconstruction and the rendering phases, as from a system's point of view. However, it is not just a way to store the scene data. There are further considerations involved in its design. From a point of view of the reconstruction phase, the scene representation should be expressive enough to be able to store the reconstructed scene data. On

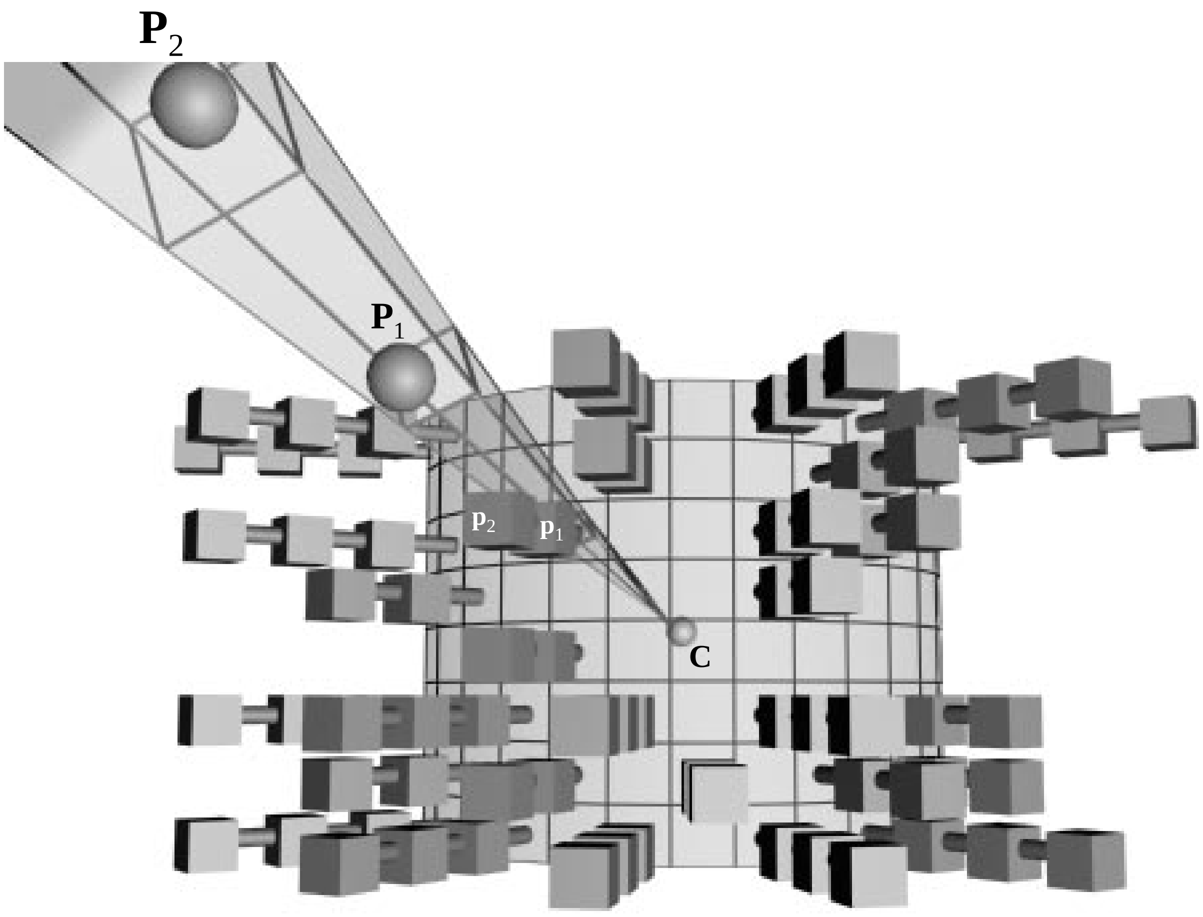


Figure 3.3: The visual data structure of the depth-layered cylinder scene representation. 3-D scene points $\mathbf{P}_1$ and $\mathbf{P}_2$ are commonly projected along a ray/frustum from the cylinder center **C** to infinity through an intersection point **p** (i.e., the classifier). Boxes linked encompassing the cylinder are elements of sorted-lists, containing actual reconstructed scene data.

the other hand, from a point of view of the rendering phase, the scene representation should be as simple to access as possible. So the desired data for generating a new view can be extracted out in a regular and fairly fast manner, such as scan-line traverse.

Our scene representation satisfies such demands from both sides, the reconstruction and the rendering processes. Let us start from the reconstruction side. The space outside the cylinder, as shown in Fig. 3.3, is unbounded, with fully horizontal perspective. And the number of elements in the linked lists is not limited. So our scene representation is capable to handle complicate scenes in any extents, i.e. real-scene images. On the other hand, what the rendering engine sees is an inside view of a cylinder. For each view rendering, it only needs to determine which portion of cylinder is required to scan through (for the image mapping) and in what particular order (for solving visibility problem). So a regular, fast rendering process is feasible based on such scene representation. In reality, rendering a view in our system we traverse each element of a small portion of the DLC row by row (or column by column) in pre-computed order and deliver the scene data to the desired image using the image mapping equations (cf. Chapter 4.4).

Our scene representation is designed for real-time applications, in the sense of “software-only” rendering acceleration. The time complexity is linear, O(N), where N denotes total number of elements stored in the DLC. However, the horizontal field of view of a virtual camera is limited and normally less than 45°. The actual portion of DLC required to be traversed is fairly small, determined by the culling process (cf.

Chapter 5.4). We show, in the worst case, 75% of the cylinder can be culled away before the traverse takes place for each desired view synthesis, whereas 93.6% in the best case. Furthermore, based on simplicity of our scene representation, we can propagate the current computed result of image mapping to the next mapping computation incrementally until the end of image row (or column) is reached. Hence, each mapping computation originally requiring 17 multiplication, 1 division, and 13 additions operations is dramatically reduced down to 4 multiplication, 1 division, and 8 additions, i.e. approximately four times speedups than the original one. We show this superior algebraic property and its applications later in Chapter 4.6.

A 3-D scene data reconstructed can be registered into the DLC if and only if the differences between its depth value and the ones of the stored data are above a threshold. The threshold is set up proportionally to the distance away from the projection center. In other words, an element of linked lists in our scene representation is created and inserted to the list only when a new scene data passes the thresholding scheme. The huge redundant 3-D data contributed from multiple input images, especially those of close-view, are hence discarded.

The efficiency of storage in our scene representation is apparently sufficient for practical use. We do not directly store complete raw-data (multiple images) into our scene representation, as in O. Faugeras's approach [3], nor transform them into a large database-like representation such as the rays-based one in the light fields approach [7]. Instead, we only store non-repeated scene data to our scene model under certain quantization, which is known adequate for new view(s) generation within a pre-specified viewing area.

Chapter 4

Image Mappings with Optimizations

We consider an image as a set of image-points. The image mapping is a transformation from a 2-D point on one image to a 2-D point on another image. Both 2-D image points are referred to as the projections of the same 3-D space point. An image mapping equation describes the correspondence between the projections of a 3-D point on any pair of images. Given an image point, its corresponding point on any other images, viewing in different poses, can be derived through the mapping equation if its corresponding 3-D space point is also visible from other viewpoints.

The concept of image mapping can be formulated as an image point (pixel) travels (transformations) from one image through several different geometrical spaces (coordinate systems) and finally arrives to at another image. Figure 4.1 shows such travels (transformations) for all combinations of mapping between planar and cylindrical images.

There are four geometrical spaces, namely, the world coordinate system, the camera coordinate system, the logical image model coordinate system[1], and the actual image coordinate system. Each next geometrical space is the subspace of the previous one. There are four possible combinations of the image mapping, shown in Fig. 4.1, with vertical downward arrows indicating the transformation sequence. There is a certain conversion between any two adjacent coordinate systems. Condensing all the intermediate conversion equations into a compact form, we can then map a pixel in an input image directly to a pixel in an output image. To build up such equation, we need depth information associated with each pixel on a reference image (cf. Chapter 6), a projection matrix associated with the reference image and a projection matrix associated with the desired image (cf. Appendix C for an uncalibrated case).

In this chapter, we show, in detail, the derivations of the four image mapping equations. The derivations are necessary because of the two following reasons. First, the intermediate results provide useful information to the successive process(es). For instance, a new depth information of a scene point, with respect to the camera coordinate system of the virtual camera, can be obtained in the middle of the derivation steps and used to assist the determination of the splatting size in the desired image (cf. Chapter 1.3).

Second, the intermediate step(s) inspire further computational optimization(s). We illustrate such optimizations for both the planar and the depth-layered cylinder models in the last section of this chapter. The purpose of it is to save the mapping computations by reusing the previous computed result in the current mapping calculation. It is found tremendously useful for the time-critical applications. We also perform the computational complexity analysis in which the significance of speedup is disclosed.

4.1 Notation

We call the reference image *source image* and the desired image *destination image*. Our convention is to use subscript 1 for the indication of a source image. The related notations are an image E_1, an image plane I_1, an image cylinder J_1, a camera optical center $\mathbf{C}_1$, and an image projection $\mathbf{p}_1$ of the 3-D point $\mathbf{P}$, in the image coordinate system (u_{p_1}, v_{p_1}) and in the image plane coordinate system (x_{p_1}, y_{p_1}). Furthermore, we have the image dimension $\mathcal{H}_1 \times \mathcal{W}_1$, the camera effective focal length f_1, the camera planar perspective projection

[1]It is an image coordinate system of a specific logical model, e.g. planar or cylindrical model.

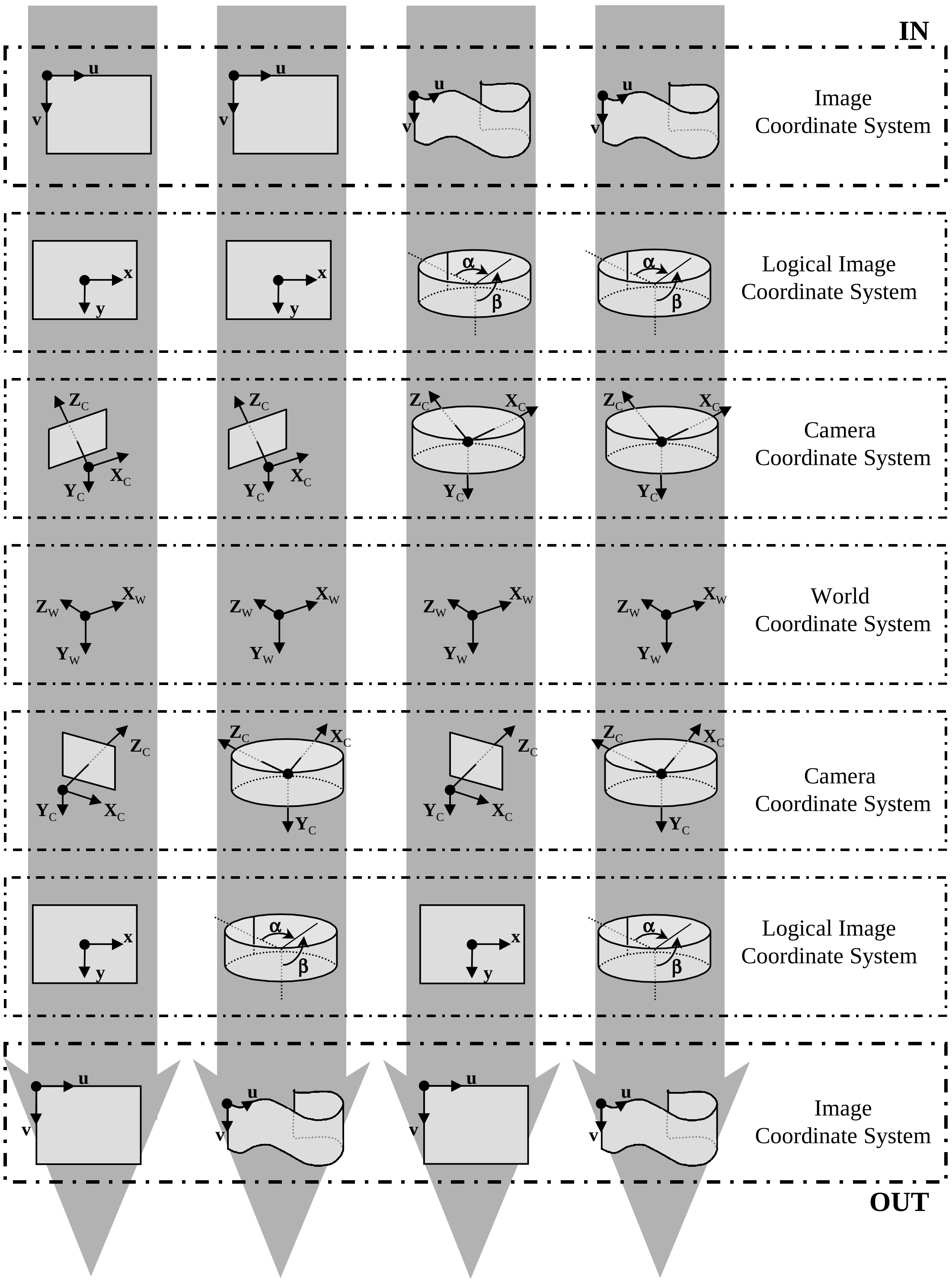


Figure 4.1: Image mappings through several intermediate geometrical-space transformations.

matrix $\Pi_1 = \left[\mathbf{H}_1 \middle| \mathbf{h}_1\right] = [\mathbf{A}_1\mathbf{R}_1|\mathbf{h}_1]$ (cf. Chapter 2.1), and finally the associated depth map $\mathcal{D}_1$ (cf. Chapter 2.4). Subscript 2, likewise, is used for the destination image. In particular, we use $\mathcal{Z}_2(u_{p_2}, v_{p_2}, k)$ to denote the depth value of the kth layer of the class (pixel), (u_{p_2}, v_{p_2}), in the DLC.

4.2 Planar to Planar Image Mapping

Planar image is currently the most available image format among all. The mapping equation between two planar images should be established first. Suppose we are required to map a point (u_{p_1}, v_{p_1}) in the source planar image to a point (u_{p_2}, v_{p_2}) in the destination planar image, which are the projections of the same 3-D point $\mathbf{P}$.

First of all, the actual image coordinates of a point $\mathbf{p}_1$ are converted to its logical image plane coordinates as follows,

$$\begin{pmatrix} x_{p_1} \\ y_{p_1} \\ 1 \end{pmatrix} = \begin{pmatrix} u_{p_1} - \dfrac{\mathcal{W}_1}{2} + 0.5 \\ v_{p_1} - \dfrac{\mathcal{H}_1}{2} + 0.5 \\ 1 \end{pmatrix}, \tag{4.2.1}$$

where $\mathcal{W}_1$ and $\mathcal{H}_1$ are the source image's dimensions, width and height, in pixel units and the camera z-axis is assumed to exactly pass through the image center.

Second, the 3-D point $\mathbf{P}$, with respect to the camera coordinate system of the source image, can be computed using both the camera intrinsic matrix $\mathbf{A}_1$ and the depth map $\mathcal{D}_1$ like this,

$$\begin{pmatrix} s_1 x_{p_1} \\ s_1 y_{p_1} \\ s_1 \\ 1 \end{pmatrix} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{pmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{pmatrix}_{\mathbf{C}_1},$$

where s_1 is some nonzero scalar, $\mathbf{0}$ is a 3-D zero vector, and $(\cdot)_{\mathbf{C}_1}$ denotes a vector with respect to the camera coordinate system of $\mathbf{C}_1$. So inversely, $(X_p, Y_p, Z_p)^T_{\mathbf{C}_1}$ can be determined by applying the inverse of $\mathbf{A}_1$ to the vector $(sx_{p_1}, sy_{p_1}, s)^T$. The intrinsic matrix $\mathbf{A}_i$ for image E_i has a general form,

$$\mathbf{A}_i = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

where a_{ij} are some real numbers determined by camera intrinsic parameters. Thus, the scalar s_1 is equal to the value Z_p, that is, the depth of $\mathbf{p}_1$. The depth value of the image point $\mathbf{p}_1$ can be looked up from the depth map $\mathcal{D}_1(u_{p_1}, v_{p_1})$. Hence, the following holds,

$$\begin{pmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{pmatrix}_{\mathbf{C}_1} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}^{-1} \begin{pmatrix} \mathcal{D}_1(u_{p_1}, v_{p_1}) x_{p_1} \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) y_{p_1} \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \\ 1 \end{pmatrix}.$$

The conversion from the camera coordinate system to the world coordinate system is straightforward,

$$\begin{pmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{pmatrix}_{world} = \begin{bmatrix} \mathbf{R}_1 & -\mathbf{R}_1 \mathbf{C}_1 \\ \mathbf{0}^T & 1 \end{bmatrix}^{-1} = \begin{pmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{pmatrix}_{\mathbf{C}_1},$$

where $\mathbf{R}_1$ is the 3×3 rotation matrix with respect to the world coordinate system. Combining the last two transformations together we have

$$\begin{aligned}
\begin{pmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{pmatrix}_{world} &= \begin{bmatrix} \mathbf{R}_1 & -\mathbf{R}_1\mathbf{C}_1 \\ \mathbf{0}^T & 1 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{A}_1 & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}^{-1} \begin{pmatrix} \mathcal{D}_1(u_{p_1}, v_{p_1})x_{p_1} \\ \mathcal{D}_1(u_{p_1}, v_{p_1})y_{p_1} \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \\ 1 \end{pmatrix}, \\
&= \left(\begin{bmatrix} \mathbf{A}_1 & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_1 & -\mathbf{R}_1\mathbf{C}_1 \\ \mathbf{0}^T & 1 \end{bmatrix} \right)^{-1} \begin{pmatrix} \mathcal{D}_1(u_{p_1}, v_{p_1})x_{p_1} \\ \mathcal{D}_1(u_{p_1}, v_{p_1})y_{p_1} \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \\ 1 \end{pmatrix}, \\
&= \begin{bmatrix} \Pi_1 \\ \mathbf{0}^T \ 1 \end{bmatrix}^{-1} \begin{pmatrix} \mathcal{D}_1(u_{p_1}, v_{p_1})x_{p_1} \\ \mathcal{D}_1(u_{p_1}, v_{p_1})y_{p_1} \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \\ 1 \end{pmatrix}. \qquad (4.2.2)
\end{aligned}$$

One may notice that it is just an inverse of 4×4 camera projection matrix, which implies the back-projection[2] can be applied to an image as long as its associated depth map and relative projection matrices are recovered (cf. Chapter 6 and Appendix C).

Once the coordinates of a 3-D point $\mathbf{P}$ in world coordinate system are found, the next is simply projecting it onto the destination image plane by the camera perspective projection matrix, Π_2. Add this projection to Eq. 4.2.2, we get

$$\begin{pmatrix} s_2 x_{p_2} \\ s_2 y_{p_2} \\ s_2 \\ 1 \end{pmatrix} = \begin{bmatrix} \Pi_2 \\ \mathbf{0}^T \ 1 \end{bmatrix} \begin{bmatrix} \Pi_1 \\ \mathbf{0}^T \ 1 \end{bmatrix}^{-1} \begin{pmatrix} \mathcal{D}_1(u_{p_1}, v_{p_1})x_{p_1} \\ \mathcal{D}_1(u_{p_1}, v_{p_1})y_{p_1} \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \\ 1 \end{pmatrix},$$

where s_2 is an arbitrary nonzero scalar. Substitute Eq. 4.2.1 into above equation we have

$$\begin{pmatrix} s_2 x_{p_2} \\ s_2 y_{p_2} \\ s_2 \\ 1 \end{pmatrix} = \begin{bmatrix} \Pi_2 \\ \mathbf{0}^T \ 1 \end{bmatrix} \begin{bmatrix} \Pi_1 \\ \mathbf{0}^T \ 1 \end{bmatrix}^{-1} \begin{pmatrix} \mathcal{D}_1(u_{p_1}, v_{p_1}) \left(u_{p_1} + \frac{1 - \mathcal{W}_1}{2} \right) \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \left(v_{p_1} + \frac{1 - \mathcal{H}_1}{2} \right) \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \\ 1 \end{pmatrix}, \qquad (4.2.3)$$

[2]It is referred to as projecting an image pixel back to its 3-D space point.

or, alternatively

$$\begin{pmatrix} x_{p_2} \\ y_{p_2} \\ 1 \\ \frac{1}{s_2} \end{pmatrix} \simeq \begin{bmatrix} \Pi_2 \\ \mathbf{0}^T \ 1 \end{bmatrix} \begin{bmatrix} \Pi_1 \\ \mathbf{0}^T \ 1 \end{bmatrix}^{-1} \begin{pmatrix} u_{p_1} + \dfrac{1-\mathcal{W}_1}{2} \\ v_{p_1} + \dfrac{1-\mathcal{H}_1}{2} \\ 1 \\ \dfrac{1}{\mathcal{D}_1(u_{p_1}, v_{p_1})} \end{pmatrix}, \qquad (4.2.4)$$

where $\simeq$ means "equal" up to a scale factor.

Equations 4.2.3 and 4.2.4 can be used to map the image pixels from one planar image to the other whenever their relative projection matrices are available (cf. Appendix C). Additionally, if both the source and the destination images are calibrated, that is, all the intrinsic and extrinsic parameters associated with both the images are available, then Eq. 4.2.3 can be rewritten as

$$\begin{pmatrix} x_{p_2} \\ y_{p_2} \\ 1 \\ \frac{1}{s_2} \end{pmatrix} \simeq \begin{bmatrix} \mathbf{H}_2 & -\mathbf{H}_2\mathbf{C}_2 \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{H}_1 & -\mathbf{H}_1\mathbf{C}_1 \\ \mathbf{0}^T & 1 \end{bmatrix}^{-1} \begin{pmatrix} \mathcal{D}_1(u_{p_1}, v_{p_1}) \left(u_{p_1} + \dfrac{1-\mathcal{W}_1}{2} \right) \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \left(v_{p_1} + \dfrac{1-\mathcal{H}_1}{2} \right) \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \\ 1 \end{pmatrix}, \qquad (4.2.5)$$

where

$$\mathbf{H}_2 = \mathbf{A}_2\mathbf{R}_2 \quad \text{and} \quad \mathbf{H}_1 = \mathbf{A}_1\mathbf{R}_1.$$

According to the theorem below (known from linear algebra [26]), we can calculate the inverse of our 4×4 matrix

$$\begin{bmatrix} \mathbf{H}_1 & -\mathbf{H}_1\mathbf{C}_1 \\ \mathbf{0}^T & 1 \end{bmatrix}.$$

Theorem 4.2.1. *If a nonsingular matrix $\boldsymbol{B}$ can be partitioned into the form*

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{B}_1 & \boldsymbol{B}_2 \\ \boldsymbol{B}_3 & \boldsymbol{B}_4 \end{bmatrix},$$

such that $\boldsymbol{B}_1$ is nonsingular, then

$$\boldsymbol{B}^{-1} = \begin{bmatrix} \boldsymbol{B}' & -\boldsymbol{B}_1^{-1}\boldsymbol{B}_2\boldsymbol{B}'' \\ -\boldsymbol{B}''\boldsymbol{B}_3\boldsymbol{B}_1^{-1} & \boldsymbol{B}'' \end{bmatrix},$$

where

$$\boldsymbol{B}'' = \left(\boldsymbol{B}_1 - \boldsymbol{B}_2\boldsymbol{B}_4^{-1}\boldsymbol{B}_3\right)^{-1}, \quad \textit{and} \quad \boldsymbol{B}' = \boldsymbol{B}_4^{-1} + \boldsymbol{B}_4^{-1}\boldsymbol{B}_3\boldsymbol{B}''\boldsymbol{B}_2\boldsymbol{B}_4^{-1}.$$

Since the matrix $\mathbf{H}_1$ is always invertible, Eq. 4.2.5 becomes

$$\begin{pmatrix} x_{p_2} \\ y_{p_2} \\ 1 \\ \frac{1}{s_2} \end{pmatrix} \simeq \begin{bmatrix} \mathbf{H}_2 & -\mathbf{H}_2\mathbf{C}_2 \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{H}_1^{-1} & \mathbf{C}_1 \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{pmatrix} \mathcal{D}_1(u_{p_1}, v_{p_1}) \left(u_{p_1} + \frac{1-\mathcal{W}_1}{2} \right) \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \left(v_{p_1} + \frac{1-\mathcal{H}_1}{2} \right) \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \\ 1 \end{pmatrix},$$

$$= \begin{bmatrix} \mathbf{H}_2\mathbf{H}_1^{-1} & \mathbf{H}_2(\mathbf{C}_1 - \mathbf{C}_2) \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{pmatrix} \mathcal{D}_1(u_{p_1}, v_{p_1}) \left(u_{p_1} + \frac{1-\mathcal{W}_1}{2} \right) \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \left(v_{p_1} + \frac{1-\mathcal{H}_1}{2} \right) \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \\ 1 \end{pmatrix}.$$

We may further expand it as follows,

$$\begin{pmatrix} x_{p_2} \\ y_{p_2} \\ 1 \end{pmatrix} \simeq \mathcal{D}_1(u_{p_1}, v_{p_1}) \mathbf{H}_2\mathbf{H}_1^{-1} \begin{pmatrix} u_{p_1} + \frac{1-\mathcal{W}_1}{2} \\ v_{p_1} + \frac{1-\mathcal{H}_1}{2} \\ 1 \end{pmatrix} + \mathbf{H}_2(\mathbf{C}_1 - \mathbf{C}_2). \tag{4.2.6}$$

Finally the actual image coordinates can be derived by

$$\begin{pmatrix} u_{p_2} \\ v_{p_2} \\ 1 \end{pmatrix} = \begin{pmatrix} \left\lfloor x_{p_2} + \frac{\mathcal{W}_2}{2} \right\rfloor \\ \left\lfloor y_{p_2} + \frac{\mathcal{H}_2}{2} \right\rfloor \\ 1 \end{pmatrix}, \tag{4.2.7}$$

where $\mathcal{W}_2$ and $\mathcal{H}_2$ are the destination image's width and height in pixel units.

To illustrate an application of the planar-to-planar image mapping equation, different views of a single-planar-image scene are generated. We use the right image of the stereo pair, shown in Fig. 6.5, as our source image to generate a set of destination images from various viewing directions toward the scene center (image center). The results are displayed in Fig. 4.2.

4.3 Planar to Cylindrical Image Mapping

Cylindrical image model is the natural representation of a full view panoramic image. The camera optical center locates at the center of the cylinder and the radius of the cylinder is equal to the effective camera focal length. The mapping from a planar image to a cylindrical image is depicted in Fig. 4.3. We use subscript 2 for the destination cylindrical image, and subscript 1 for the source planar image.

The back-projection from an image point $\mathbf{p}_1$ to its 3-D space point is shown in the previous planar-to-planar image mapping derivation. To project a 3-D point $\mathbf{P}$ onto a cylindrical image there are several intermediate steps similar to those in the planar image mapping. First we need to transform the coordinates

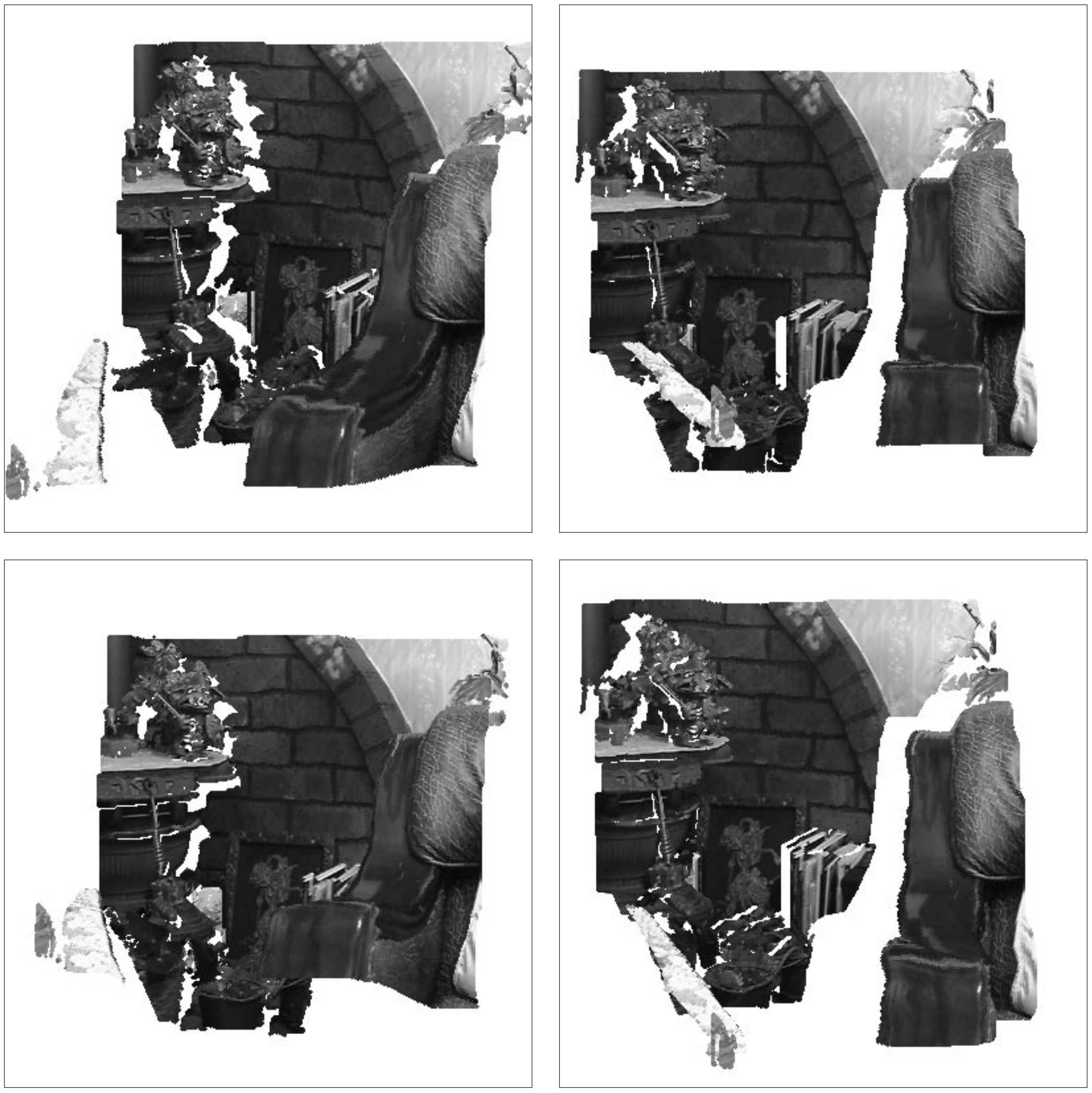


Figure 4.2: Novel views of a scene from the single planar image in Fig. 6.5, i.e. various viewing directions toward the scene center. The white gaps indicate the scene regions that are occluded in the source image.

of a 3-D point **P** from the world coordinate system to the camera coordinate system which is associated with the cylindrical image. Then we project the point onto the cylinder model. This projection can not be written in the matrix from, because the mapping is non-linear. A point on the cylinder is represented by a 2-D ordered pair $(\alpha_{p_2}, \beta_{p_2})$, called image cylinder coordinate system. The definition of these two parameters are shown in Fig. 4.4. Finally, the mapping is accomplished by the transformation from the image cylinder coordinate system to the actual cylindrical image coordinate system, which corresponds to the step of the transformation from the image plane coordinate system to the actual planar image coordinate system.

The combination of the transformations from (u_{p_1}, v_{p_1}) to the 3-D point in world coordinate system and

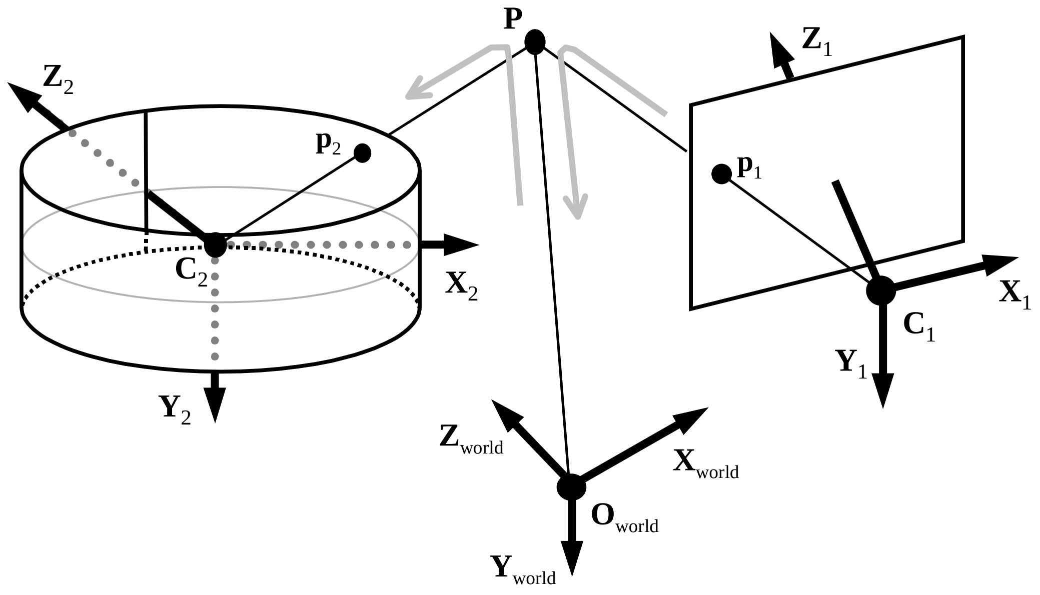


Figure 4.3: Geometry of the planar-to-cylindrical image mapping.

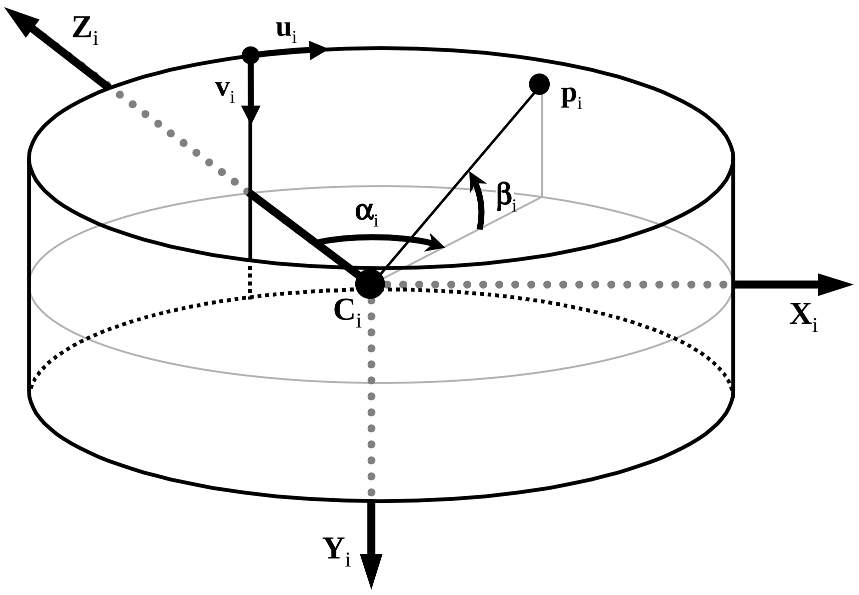


Figure 4.4: Geometry of the image cylinder coordinate system.

from the world coordinate system to the destination (cylinder) camera coordinate system is as follows,

$$\begin{pmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{pmatrix}_{\mathbf{C}_2} \simeq \begin{bmatrix} \mathbf{R}_2 & -\mathbf{R}_2\mathbf{C}_2 \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \Pi_1 \\ \mathbf{0}^T \; 1 \end{bmatrix}^{-1} \begin{pmatrix} u_{p_1} + \dfrac{1-\mathcal{W}_1}{2} \\ v_{p_1} + \dfrac{1-\mathcal{H}_1}{2} \\ 1 \\ \dfrac{1}{\mathcal{D}_1(u_{p_1}, v_{p_1})} \end{pmatrix}, \tag{4.3.1}$$

where $\simeq$ means "equal" up to a scale factor, and $\mathbf{R}_2$ is a 3×3 rotation matrix with respect to the world coordinate system. Equation 4.3.1 can be applied when a projection matrix Π_1, which is associated with the uncalibrated source image, is found relative to the world coordinate system. If both the camera intrinsic and extrinsic parameters of the source image are known, then Eq. 4.3.1 can be simplified by using Theorem 4.2.1 as follows,

$$\begin{pmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{pmatrix}_{\mathbf{C}_2} = \begin{bmatrix} \mathbf{R}_2 & -\mathbf{R}_2\mathbf{C}_2 \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{H}_1 & -\mathbf{H}_1\mathbf{C}_1 \\ \mathbf{0}^T & 1 \end{bmatrix}^{-1} \begin{pmatrix} \mathcal{D}_1(u_{p_1}, v_{p_1}) \left(u_{p_1} + \frac{1-\mathcal{W}_1}{2} \right) \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \left(v_{p_1} + \frac{1-\mathcal{H}_1}{2} \right) \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \\ 1 \end{pmatrix},$$

$$= \begin{bmatrix} \mathbf{R}_2 & -\mathbf{R}_2\mathbf{C}_2 \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{H}_1^{-1} & \mathbf{C}_1 \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{pmatrix} \mathcal{D}_1(u_{p_1}, v_{p_1}) \left(u_{p_1} + \frac{1-\mathcal{W}_1}{2} \right) \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \left(v_{p_1} + \frac{1-\mathcal{H}_1}{2} \right) \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \\ 1 \end{pmatrix},$$

$$= \begin{bmatrix} \mathbf{R}_2\mathbf{H}_1^{-1} & \mathbf{R}_2(\mathbf{C}_1 - \mathbf{C}_2) \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{pmatrix} \mathcal{D}_1(u_{p_1}, v_{p_1}) \left(u_{p_1} + \frac{1-\mathcal{W}_1}{2} \right) \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \left(v_{p_1} + \frac{1-\mathcal{H}_1}{2} \right) \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \\ 1 \end{pmatrix},$$

where

$$\mathbf{H}_1 = \mathbf{A}_1\mathbf{R}_1.$$

Furthermore, it can be written as

$$\begin{pmatrix} X_p \\ Y_p \\ Z_p \end{pmatrix}_{\mathbf{C}_2} = \mathcal{D}_1(u_{p_1}, v_{p_1})\mathbf{R}_2\mathbf{H}_1^{-1} \begin{pmatrix} u_{p_1} + \frac{1-\mathcal{W}_1}{2} \\ v_{p_1} + \frac{1-\mathcal{H}_1}{2} \\ 1 \end{pmatrix} + \mathbf{R}_2(\mathbf{C}_1 - \mathbf{C}_2). \tag{4.3.2}$$

So far we have 3-D information of a point $\mathbf{P}$ in respect to the destination camera coordinate system. The projection of $\mathbf{P}$ onto the image cylinder is nonlinear, so the remaining warping steps cannot be incooperate with the Eq. 4.3.2 above. Given $\mathbf{P} = (X_p, Y_p, Z_p)^T_{\mathbf{C}_2}$, we have its projection in the image cylinder coordinate system as

$$\begin{pmatrix} \alpha_{p_2} \\ \beta_{p_2} \\ 1 \end{pmatrix} = \begin{pmatrix} \tan^{-1}\left(\frac{X_p}{Z_p}\right) \\ \tan^{-1}\left(\frac{Y_p}{\sqrt{X_p^2 + Z_p^2}}\right) \\ 1 \end{pmatrix}.$$

Followed by the transformation between the image cylinder coordinate system and the actual image coordinate system,

$$\begin{pmatrix} u_{p_2} \\ v_{p_2} \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{\alpha_{p_2}}{360} \times 2\pi f_2 \\ f_2 \tan \beta_{p_2} + \frac{\mathcal{H}_2}{2} \\ 1 \end{pmatrix},$$

where f_2 is the effective camera focal length of the destination image and $\mathcal{H}_2$ is the height of the destination image cylinder in pixel unit. The complete projection formula for the cylindrical image becomes like this,

$$\begin{pmatrix} u_{p_2} \\ v_{p_2} \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{\pi f_2 \tan^{1}\left(\frac{X_p}{Z_p}\right)}{180} \\ \frac{f_2 Y_p}{\sqrt{X_p^2 + Z_p^2}} + \frac{\mathcal{H}_2}{2} \\ 1 \end{pmatrix}. \tag{4.3.3}$$

If the focal length f_2 is not available, then the projection formula can also be expressed as follows,

$$\begin{pmatrix} u_{p_2} \\ v_{p_2} \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{\mathcal{W}_2 \tan^{-1}\left(\frac{X_p}{Z_p}\right)}{360} \\ \frac{\mathcal{W}_2 Y_p}{2\pi\sqrt{X_p^2 + Z_p^2}} + \frac{\mathcal{H}_2}{2} \\ 1 \end{pmatrix},$$

where $\mathcal{W}_2$ is the length of the destination image in pixel unit, or equivalent to the diameter of the image cylinder.

The major application of the planar-to-cylindrical image mapping equation in our system is registering multiple planar images into the DLC. In Chapter 3 we have described the registration scheme that a source-image pixel can be registered into the DLC if and only if the differences between its depth value and the ones of the stored data are above a threshold[3]. The procedures of the registration process are follows. First, we use Eq. 4.3.3 to determine which class (pixel), of the DLC, the source-image pixel is potentially registered to (cf. Chapter 3.3). Then, the depth value of the source pixel, Z_p, obtained in Eq. 4.3.2, is used to compare with others where there may be one or more existing scene data already in the determined class of the DLC. The scene data is qualified to be registered into the DLC if the depth comparisons satisfy the registration scheme; otherwise it is just ignored. Upon the acceptance, an element of the linked list of that class, which contains the color information and the depth value, is created and inserted.

[3]The threshold is proportional to the distance from the projection center, see Chapter 3.3.

4.4 Cylindrical to Planar Image Mapping

The depth of an image pixel may be defined differently in cylindrical images, and it is not equal to the Z-component of the 3-D point with respect to its camera coordinate system as does in planar images. For a cylindrical image point $\mathbf{p}_1$, the depth value $\mathcal{D}_1(u_{p_1}, v_{p_1})$ means the distance between the origin $\mathbf{C}_1$ and the projection of the 3-D space point $\mathbf{P}$ onto xz-plane of the cylindrical camera coordinate system. Thus we can determine easily the 3-D point $\mathbf{P}$ in the camera coordinate system in terms of α_{p_1}, β_{p_1}, and the depth $\mathcal{D}_1(u_{p_1}, v_{p_1})$ as follows,

$$\begin{pmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{pmatrix}_{\mathbf{C}_1} = \begin{pmatrix} \mathcal{D}_1(u_{p_1}, v_{p_1}) \sin \alpha_{p_1} \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \tan \beta_{p_1} \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \cos \alpha_{p_1} \\ 1 \end{pmatrix}.$$

The transformation from (u_{p_1}, v_{p_1}) to $(\alpha_{p_1}, \beta_{p_1})$ is established as

$$\begin{pmatrix} \alpha_{p_1} \\ \beta_{p_1} \\ 1 \end{pmatrix} = \begin{pmatrix} \dfrac{u_{p_1} + 0.5}{2\pi f_1} \times 360 \\ \tan^{-1} \left(\dfrac{v_{p_1} - \dfrac{\mathcal{H}_1}{2} + 0.5}{f_1} \right) \\ 1 \end{pmatrix}.$$

Combining these two together, we have a cylindrical image back-projection formula like this,

$$\begin{pmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{pmatrix}_{\mathbf{C}_1} = \begin{pmatrix} \mathcal{D}_1(u_{p_1}, v_{p_1}) \sin \left(\dfrac{180 u_{p_1} + 90}{\pi f_1} \right) \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \left(\dfrac{2 v_{p_1} - \mathcal{H}_1 + 1}{2 f_1} \right) \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \cos \left(\dfrac{180 u_{p_1} + 90}{\pi f_1} \right) \\ 1 \end{pmatrix}.$$

Once we have 3-D information of the point from the source cylindrical image, the rest of transformations to the destination planar image are straightforward. First, transform the point $\mathbf{P}$ into the world coordinate system, then project it onto the destination planar image using the planar camera projection matrix. It gives

$$\begin{pmatrix} x_{p_2} \\ y_{p_2} \\ 1 \\ 1 \end{pmatrix} \simeq \begin{bmatrix} \mathbf{H}_2 & -\mathbf{H}_2 \mathbf{C}_2 \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_1 & -\mathbf{R}_1 \mathbf{C}_1 \\ \mathbf{0}^T & 1 \end{bmatrix}^{-1} \begin{pmatrix} \mathcal{D}_1(u_{p_1}, v_{p_1}) \sin \left(\dfrac{180 u_{p_1} + 90}{\pi f_1} \right) \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \left(\dfrac{2 v_{p_1} - \mathcal{H}_1 + 1}{2 f_1} \right) \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \cos \left(\dfrac{180 u_{p_1} + 90}{\pi f_1} \right) \\ 1 \end{pmatrix},$$

where

$$\mathbf{H}_2 = \mathbf{A}_2 \mathbf{R}_2.$$

This equation can be simplified as follows,

$$\begin{pmatrix} x_{p_2} \\ y_{p_2} \\ 1 \end{pmatrix} \simeq \mathcal{D}_1(u_{p_1}, v_{p_1})\mathbf{H}_2\mathbf{R}_1^{-1} \begin{pmatrix} \sin\left(\dfrac{180u_{p_1}+90}{\pi f_1}\right) \\ \left(\dfrac{2v_{p_1}-\mathcal{H}_1+1}{2f_1}\right) \\ \cos\left(\dfrac{180u_{p_1}+90}{\pi f_1}\right) \\ 1 \end{pmatrix} + \mathbf{H}_2(\mathbf{C}_1 - \mathbf{C}_2).$$

And finally,

$$\begin{pmatrix} u_{p_2} \\ v_{p_2} \\ 1 \end{pmatrix} = \begin{pmatrix} \left\lfloor x_{p_2} + \dfrac{\mathcal{W}_2}{2} \right\rfloor \\ \left\lfloor y_{p_2} + \dfrac{\mathcal{H}_2}{2} \right\rfloor \\ 1 \end{pmatrix},$$

where $\mathcal{W}_2$ and $\mathcal{H}_2$ are the destination image's width and height in pixel units.

A very important role of this mapping equation played in our view synthesis system is being part of the rendering processes. The rendering process comprises three major steps. First, it determines which portion of the DLC is necessary to the desired view, referred to as culling process (cf. Chapter 5.4). Second, it computes a traverse order such that the visible surfaces are determined correctly in the desired view (cf. Chapter 5.1, 5.2, 5.3). Third, it transfers scene data to the designation image by the equation(s) established in this section. The mapping process scans through a small portion of the DLC in the specific order and forwards the pixels to the destination image (planar image) with a chosen image reconstruction method. We use the splatting method for on-line rendering, and polygonal-texture mapping method for off-line rendering (cf. Chapter 1.3). The mapping process is further accelerated up using the incremental computation for both on-line and off-line applications. We return to this subject later on in this chapter.

4.5 Cylindrical to Cylindrical Image Mapping

We have shown the back-projection equation of a cylindrical image point to the 3-D space point and the projection equation of the 3-D point onto a cylindrical image in the previous two sections. The mapping equation between a pair of cylindrical image points is a combination of these two equations. As we have mentioned, the cylindrical image mapping is nonlinear so we separate the mapping equation into two parts.

First, the mapping from the source cylindrical image point $\mathbf{p}_1$ to the 3-D point $\mathbf{P}$ in the destination camera coordinate system is

$$\begin{pmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{pmatrix}_{\mathbf{C}_2} \simeq \begin{bmatrix} \mathbf{R}_2 & -\mathbf{R}_2\mathbf{C}_2 \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_1 & -\mathbf{R}_1\mathbf{C}_1 \\ \mathbf{0}^T & 1 \end{bmatrix}^{-1} \begin{pmatrix} \mathcal{D}_1(u_{p_1}, v_{p_1}) \sin\left(\dfrac{180u_{p_1}+90}{\pi f_1}\right) \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \left(\dfrac{2v_{p_1}-\mathcal{H}_1+1}{2f_1}\right) \\ \mathcal{D}_1(u_{p_1}, v_{p_1}) \cos\left(\dfrac{180u_{p_1}+90}{\pi f_1}\right) \\ 1 \end{pmatrix}.$$

It can be expressed in a simpler form as follows,

$$\begin{pmatrix} X_p \\ Y_p \\ Z_p \end{pmatrix}_{\mathbf{C}_2} = \mathcal{D}_1(u_{p_1}, v_{p_1})\mathbf{R}_2\mathbf{R}_1^{-1} \begin{pmatrix} \sin\left(\dfrac{180u_{p_1}+90}{\pi f_1}\right) \\ \left(\dfrac{2v_{p_1}-\mathcal{H}_1+1}{2f_1}\right) \\ \cos\left(\dfrac{180u_{p_1}+90}{\pi f_1}\right) \\ 1 \end{pmatrix} + \mathbf{R}_2(\mathbf{C}_1 - \mathbf{C}_2), \tag{4.5.1}$$

or if the focal length f_1 is unknown, then

$$\begin{pmatrix} X_p \\ Y_p \\ Z_p \end{pmatrix}_{\mathbf{C}_2} = \mathcal{D}_1(u_{p_1}, v_{p_1})\mathbf{R}_2\mathbf{R}_1^{-1} \begin{pmatrix} \sin\left(\dfrac{360u_{p_1}+180}{\mathcal{W}_1}\right) \\ \left(\dfrac{\pi(2v_{p_1}-\mathcal{H}_1+1)}{\mathcal{W}_1}\right) \\ \cos\left(\dfrac{360u_{p_1}+180}{\mathcal{W}_1}\right) \\ 1 \end{pmatrix} + \mathbf{R}_2(\mathbf{C}_1 - \mathbf{C}_2).$$

Secondly, the mapping of the 3-D point onto the destination cylindrical image is characterized by

$$\begin{pmatrix} u_{p_2} \\ v_{p_2} \\ 1 \end{pmatrix} = \begin{pmatrix} \dfrac{\pi f_2 \tan^{-1}\left(\dfrac{X_p}{Z_p}\right)}{180} \\ \dfrac{f_2 Y_p}{\sqrt{X_p^2+Z_p^2}} + \dfrac{\mathcal{H}_2}{2} \\ 1 \end{pmatrix}, \tag{4.5.2}$$

or, again, if the focal length f_2 is unknown, then by

$$\begin{pmatrix} u_{p_2} \\ v_{p_2} \\ 1 \end{pmatrix} = \begin{pmatrix} \dfrac{\mathcal{W}_2 \tan^{-1}\left(\dfrac{X_p}{Z_p}\right)}{360} \\ \dfrac{\mathcal{W}_2 Y_p}{2\pi\sqrt{X_p^2+Z_p^2}} + \dfrac{\mathcal{H}_2}{2} \\ 1 \end{pmatrix}.$$

Equations 4.5.1 and 4.5.2 altogether specify the mapping between two cylindrical images.

In our system the application of the cylindrical-to-cylindrical image mapping equation is to register each cylindrical image to the DLC, for a multiple-panoramas case. The procedures of the registration are the same as the ones introduced in the planar-to-cylindrical case, except that the mapping equations used are different.

4.6 Incremental Mapping Computation

In the previous sections, we have shown that every image-point mapping is independent from the mapping of others on the source image. In other words, mapping an image point to another image does not require, or depend on, any information from its neighboring image points. What is really required for a single image-point mapping, as mentioned before, are the depth information associated with the image point and the camera information associated with the source and the destination images. However, if we have an uniform traverse order through the source image, such as a scan-line fashion (pixel by pixel), the coherence of mapping computations along the scan-line may further be explored.

In this section, we start from the simplest one, the planar image case, to show the basic idea, followed by a practical application, the depth-layered cylinder case, used in our system. Furthermore, the speed-up due to this computational optimization is analyzed.

4.6.1 Planar Image

In Eq. 4.2.4, we have established the mapping from a source-image pixel (u_{p_1}, v_{p_1}) to a destination image-plane point (x_{p_2}, y_{p_2}) as follows,

$$\begin{pmatrix} x_{p_2} \\ y_{p_2} \\ 1 \\ 1 \end{pmatrix} \simeq \begin{bmatrix} \Pi_2 \\ \mathbf{0}^T \; 1 \end{bmatrix} \begin{bmatrix} \Pi_1 \\ \mathbf{0}^T \; 1 \end{bmatrix}^{-1} \begin{pmatrix} u_{p_1} + \dfrac{1-\mathcal{W}_1}{2} \\ v_{p_1} + \dfrac{1-\mathcal{H}_1}{2} \\ 1 \\ \dfrac{1}{\mathcal{D}_1(u_{p_1}, v_{p_1})} \end{pmatrix}.$$

For each new view rendering the camera projection matrices Π_1 and Π_2 are kept unchanged. Thus the mapping equation can be rewritten as

$$\begin{pmatrix} x_{p_2} \\ y_{p_2} \\ 1 \\ 1 \end{pmatrix} \simeq \mathbf{T}_{12} \begin{pmatrix} u_{p_1} + \dfrac{1-\mathcal{W}_1}{2} \\ v_{p_1} + \dfrac{1-\mathcal{H}_1}{2} \\ 1 \\ \dfrac{1}{\mathcal{D}_1(u_{p_1}, v_{p_1})} \end{pmatrix}, \tag{4.6.1}$$

where

$$\mathbf{T}_{12} = \begin{bmatrix} \Pi_2 \\ \mathbf{0}^T \; 1 \end{bmatrix} \begin{bmatrix} \Pi_1 \\ \mathbf{0}^T \; 1 \end{bmatrix}^{-1}$$

is the transfer matrix for a source-image point to a destination-image point.

By linearity of this matrix operation, we observe that the computation of the mapping for the next pixel along the image row, i.e. $((u_{p_1} + 1), v_{p_1})$, can in fact reuse the partial computation of the mapping for the current pixel (u_{p_1}, v_{p_1}). The idea is to decompose the mapping equation of Eq. 4.6.1 into several terms, and to reuse some of them for the next computation. The following equation demonstrates the decomposition of

Eq. 4.6.1 into two terms.

$$\mathbf{T}_{12}\begin{pmatrix} u_{p_1}+\dfrac{1-\mathcal{W}_1}{2} \\ v_{p_1}+\dfrac{1-\mathcal{H}_1}{2} \\ 1 \\ \dfrac{1}{\mathcal{D}_1(u_{p_1},v_{p_1})}\end{pmatrix} = \mathbf{T}_{12}\begin{pmatrix} u_{p_1}+\dfrac{1-\mathcal{W}_1}{2} \\ v_{p_1}+\dfrac{1-\mathcal{H}_1}{2} \\ 1 \\ 0\end{pmatrix} + \frac{1}{\mathcal{D}_1(u_{p_1},v_{p_1})}\mathbf{T}_{12}\begin{pmatrix}0\\0\\0\\1\end{pmatrix},$$

$$= \mathbf{V}_{base} + \frac{1}{\mathcal{D}_1(u_{p_1},v_{p_1})}\mathbf{V}_{depth}.$$

The first term is called *base vector*, denoted as $\mathbf{V}_{base}$. The second term involves a scalar, which is the inverse depth-value of (u_{p_1}, v_{p_1}), and a *depth vector*, denoted as $\mathbf{V}_{depth}$. The following equation shows how both terms are reused in the mapping computation of the next pixel, $((u_{p_1}+1), v_{p_1})$.

$$\mathbf{T}_{12}\begin{pmatrix} (u_{p_1}+1)+\dfrac{1-\mathcal{W}_1}{2} \\ v_{p_1}+\dfrac{1-\mathcal{H}_1}{2} \\ 1 \\ \dfrac{1}{\mathcal{D}_1((u_{p_1}+1),v_{p_1})}\end{pmatrix} = \mathbf{V}_{base} + \mathbf{T}_{12}\begin{pmatrix}1\\0\\0\\0\end{pmatrix} + \frac{1}{\mathcal{D}_1((u_{p_1}+1),v_{p_1})}\mathbf{V}_{depth},$$

$$= \mathbf{V}_{base} + \mathbf{V}_{incr} + \frac{1}{\mathcal{D}_1((u_{p_1}+1),v_{p_1})}\mathbf{V}_{depth}. \qquad (4.6.2)$$

The above two equations suggest that we only need to calculate $\mathbf{V}_{incr}$ and $\mathbf{V}_{depth}$ once per view rendering, and increment the base vector $\mathbf{V}_{base}$ by $\mathbf{V}_{incr}$ for each pixel mapping along the same image row. Moreover, the inverse of the depth-value, which is pre-computed and stored in the lookup table, needs to be fetched for each pixel mapping. Figure 4.5 outlines the actual steps of the mapping processes.

The procedure `IncrementalMapping`[4] is called when a new view is to be rendered. The names of variables follow the conventions used in Eq. 4.6.1 and 4.6.2, where the underscore of variables is equivalent to the subscript of symbols, otherwise they will be explained. The 2-D array `invD_1` denotes the inverse-depth lookup table. All `rowStart`, `rowEnd`, `colStart` and `colEnd` are determined before the procedure is called (cf. Chapter 5 for the traverse order determination). Clearly, `V_incr` and `V_depth` are computed once per view rendering, both are instantiated before the outer for-loop. The vector `V_base` is initialized for each row-iteration. The code in the inner for-loop does the real mapping and sends the result to the output image buffer via the function `SendToImgBuffer`. The incremental computation for each pixel is coded by `V_base = V_base.add(V_incr)`.

4.6.2 Depth-layered Cylinder

To generate a new view from our scene representation, which is the depth-layered cylinder, the cylindrical-to-planar image equation should be used. The equation can be rewritten in a form similar to Eq. 4.6.1 as

[4]This is not an actual source code for the implementation. For instance, the term $(1 - ImgW_1)/2$ is actually a constant. It should be pre-computed, once the source image is loaded, not in the place of the for-loop. However, it corresponds to the equations we gave to simplify the explanations.

```
public void IncrementalMapping(int[][] invD_1,
                               Matrix3d T_12){

   Vec3d V_incr = T_12.times(new Vec3d(1,0,0,0));
   Vec3d V_depth = T_12.times(new Vec3d(0,0,0,1));

   for(int row=rowStart; row<rowEnd; row++){
      Vec3d V_base = T_12.times(new Vec3d( colStart+(1-ImgW_1)/2,
                                           row+(1-ImgH_1)/2,
                                           1,
                                           0));
      for(int col=colStart; col<colEnd; col++){
         Vec3d p_2 = V_base.add(V_depth.times(invD_1[row][col]));
         SendToImgBuffer(p_2.TransformToUV());
         // incremented for the next iteration (!)
         V_base = V_base.add(V_incr);
      }
   }
}
```

Figure 4.5: The procedure of the incremental mapping computation.

follows,

$$\begin{pmatrix} x_{p_2} \\ y_{p_2} \\ 1 \\ 1 \end{pmatrix} \simeq \mathbf{T}_{12} \begin{pmatrix} \sin\left(\dfrac{180u_{p_1}+90}{\pi f_1}\right) \\ \left(\dfrac{2v_{p_1}+1-\mathcal{H}_1}{2f_1}\right) \\ \cos\left(\dfrac{180u_{p_1}+90}{\pi f_1}\right) \\ \dfrac{1}{\mathcal{Z}_1(u_{p_1}, v_{p_1}, k)} \end{pmatrix},$$

$$= \mathbf{T}_{12} \begin{pmatrix} f_1 \sin\left(\dfrac{180u_{p_1}+90}{\pi f_1}\right) \\ v_{p_1} + \left(\dfrac{1-\mathcal{H}_1}{2}\right) \\ f_1 \cos\left(\dfrac{180u_{p_1}+90}{\pi f_1}\right) \\ \dfrac{1}{f_1 \mathcal{Z}_1(u_{p_1}, v_{p_1}, k)} \end{pmatrix}, \tag{4.6.3}$$

where

$$\mathbf{T}_{12} = \begin{bmatrix} \mathbf{H}_2 & -\mathbf{H}_2\mathbf{C}_2 \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_1 & -\mathbf{R}_1\mathbf{C}_1 \\ \mathbf{0}^T & 1 \end{bmatrix}^{-1}$$

is the transfer matrix for a DLC's image pixel (u_{p_1}, v_{p_1}) to a destination-image point (x_{p_2}, y_{p_2}), and $\mathcal{Z}_1(u_{p_1}, v_{p_1}, k)$ is the depth value of the kth layer at (u_{p_1}, v_{p_1}).

Here we intend to derive an equation similar to Eq. 4.6.2 with three terms, a base vector, an increment vector, plus a depth vector multiplying the inverse depth value, so the before-mentioned idea of the incremental mapping computation can be re-applied to the DLC case. We first decompose Eq. 4.6.3 as follows,

$$\begin{aligned}
&\mathbf{T}_{12}\begin{pmatrix} f_1 \sin\left(\dfrac{180u_{p_1}+90}{\pi f_1}\right) \\ v_{p_1} + \left(\dfrac{1-\mathcal{H}_1}{2}\right) \\ f_1 \cos\left(\dfrac{180u_{p_1}+90}{\pi f_1}\right) \\ \dfrac{1}{f_1 \mathcal{Z}_1(u_{p_1}, v_{p_1}, k)} \end{pmatrix} \\
= \; &\mathbf{T}_{12}\begin{pmatrix} f_1 \sin\left(\dfrac{180u_{p_1}+90}{\pi f_1}\right) \\ v_{p_1} + \left(\dfrac{1-\mathcal{H}_1}{2}\right) \\ f_1 \cos\left(\dfrac{180u_{p_1}+90}{\pi f_1}\right) \\ 0 \end{pmatrix} + \frac{1}{\mathcal{Z}_1(u_{p_1}, v_{p_1}, k)}\mathbf{T}_{12}\begin{pmatrix} 0 \\ 0 \\ 0 \\ f_1 \end{pmatrix}, \\
= \; &\mathbf{V}_{base} + \frac{1}{\mathcal{Z}_1(u_{p_1}, v_{p_1}, k)}\mathbf{V}_{depth}.
\end{aligned}$$

Due to the involvement of sin and cos functions, the incremental computation may not be applied to image rows as in the planar example. Fortunately, the relation between two adjacent pixels along the column is still linear in this cylindrical case. Therefore, we have

$$\begin{aligned}
\mathbf{T}_{12}\begin{pmatrix} f_1 \sin\left(\dfrac{180u_{p_1}+90}{\pi f_1}\right) \\ (v_{p_1}+1) + \left(\frac{1-\mathcal{H}_1}{2}\right) \\ f_1 \cos\left(\dfrac{180u_{p_1}+90}{\pi f_1}\right) \\ \dfrac{1}{f_1 \mathcal{Z}_1(u_{p_1}, (v_{p_1}+1), k)} \end{pmatrix} &= \mathbf{V}_{base} + \mathbf{T}_{12}\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + \frac{1}{\mathcal{Z}_1(u_{p_1}, (v_{p_1}+1), k)}\mathbf{V}_{depth}, \\
&= \mathbf{V}_{base} + \mathbf{V}_{incr} + \frac{1}{\mathcal{Z}_1(u_{p_1}, (v_{p_1}+1), k)}\mathbf{V}_{depth}.
\end{aligned} \tag{4.6.4}$$

Equation 4.6.4 provides an incremental mapping computation in column-wise traverse order. The vectors $\mathbf{V}_{incr}$ and $\mathbf{V}_{depth}$ are constant for each new view rendering, and are pre-calculated whenever the matrix $\mathbf{T}_{12}$ becomes available during the rendering phase. The procedure of the incremental mapping computation is similar to the one in the planar image case, except the column-wise traverse is used[5]. Moreover, for each class of the DLC (image pixel (u_{p_1}, v_{p_1})), we traverse the linked list (sorted by depth) in back-to-front order for the correct visibility determination in the destination image (cf. Chapter 5.3). The incremental mapping computation for each class of the DLC is simply achieved by iterating the linked list, fetching the depth value from $\mathcal{Z}_1(u_{p_1}, v_{p_1}, k)$, plugging it into Eq. 4.6.4, and leaving the base vectors, the increment vectors and the depth vectors unchanged.

[5]Considering the memory cache-hit optimization, from an implementation point of view, the DLC is rotated about 90° so the miss-hit rate is dramatically reduced.

4.6.3 Performance Analysis

To evaluate the merits of using the incremental mapping computation, we compare the arithmetic operations involved for Eq. 4.6.3 and for Eq. 4.6.4. What we evaluate in Eq. 4.6.3 is as follows,

$$\mathbf{T}_{12}\begin{pmatrix} f_1 \sin\left(\dfrac{180u_{p_1}+90}{\pi f_1}\right) \\ v_{p_1}+\left(\dfrac{1-\mathcal{H}_1}{2}\right) \\ f_1 \cos\left(\dfrac{180u_{p_1}+90}{\pi f_1}\right) \\ \dfrac{1}{f_1\mathcal{Z}_1(u_{p_1},v_{p_1},k)} \end{pmatrix}, \tag{4.6.5}$$

where $\mathbf{T}_{12}$ is a 4×4 matrix.

The computations pre-computable in (4.6.5) with moderate lookup-table size will not be considered in the evaluation. For example, the value of the term $(1-\mathcal{H}_1)/2$ is known once the source image is loaded; and the sin() and the cos() terms are pre-computed as follows,

$$SinTable\,[u_{p_1}] = f_1 \sin\left(\frac{180u_{p_1}+90}{\pi f_1}\right),$$

and

$$CosTable\,[u_{p_1}] = f_1 \cos\left(\frac{180u_{p_1}+90}{\pi f_1}\right),$$

where the sizes of both tables $SinTable$ and $CosTable$ are equal to $\mathcal{W}_1$(i.e. affordable size).

The number of arithmetic operations involved in (4.6.5) is therefore analyzed as follows. There are 1 addition for adding v_{p_1} with the constant number $(1-\mathcal{H}_1)/2$; and 1 multiplication plus 1 division for the inverse value of $f_1\mathcal{Z}_1(u_{p_1},v_{p_1},k)$; and 16 multiplication plus 12 additions for the multiplication of the 4×4 matrix $\mathbf{T}_{12}$ and the 4-vector. So totally there are 17 multiplication, 1 division and 13 additions in (4.6.5).

On the other hand, for the incremental mapping computation, we have the following,

$$\mathbf{V}_{base}+\mathbf{V}_{incr}+\frac{1}{\mathcal{Z}_1(u_{p_1},v_{p_1},k)}\,\mathbf{V}_{depth}, \tag{4.6.6}$$

where $\mathbf{V}_{base}$, $\mathbf{V}_{incr}$ and $\mathbf{V}_{depth}$ are all 4-vectors, and are always available for the current mapping except for the mapping of the initial pixel in the scan-line. The arithmetic operations involved in (4.6.6) are 1 division for the inverse value of $\mathcal{Z}_1(u_{p_1},v_{p_1},k)$; and 4 multiplication for multiplying the value $1/\mathcal{Z}_1(u_{p_1},v_{p_1},k)$ and the 4-vector $\mathbf{V}_{depth}$; and 8 additions for adding these three 4-vectors. So there are totally 4 multiplication, 1 division and 8 additions in (4.6.6).

We can then conclude that at least three fourth of multiplication in the mapping computations are saved because of this computational optimization.

Chapter 5

Visibility Determination

During the rendering processes, many expensive yet redundant computations can be eliminated prior if surface points are known to be invisible to the viewpoint. The preprocess, which does visibility checking before costly computations are involved, is called *visible surface determination*. Many object/image precision algorithms [27], removing the invisible surface effectively, have been developed. However, as the number of scene objects becomes extremely large, the performance of those algorithms is degraded significantly. They are classified as scene-complexity-dependent algorithms.

Unlike traditional computer graphics approaches, the image-based rendering techniques attempt to bypass the visible surface determination process, classified as scene-complexity-independent algorithms. S. E. Chen [2] posed his model in a proper way that all visible scene points are uniquely defined with respect to a fixed viewpoint as they are acquired. So no visible surface determination is required for the rendering in his model. Similarly, M. Levoy and P. Hanrahan [7] resamped/interpolated a finite, yet huge, set of rays to obtain the desired view without the visible surface determination. M. Seitz [6] generated in-between views using an image morphing technique (bi-directional interpolation) under the monotonicity assumption. It also bypasses the visible surface determination. Their approaches are all based on the fact that pixels in an image inherently represent the visible 3-D scene points from the viewpoint. Thus the new viewpoint(s) either lies exactly on the previous viewpoint(s), where the images were acquired, or somewhere along the line joining two original viewpoints.

S. Laveau and O. Faugeras [3], allowing views to be extrapolated as well as interpolated, discussed the combination of information from several views and provided a 2-D ray-tracing-like solution to the visibility problem, which does not appeal to an underlying geometric description. L. McMillan [28] proposed a visible surface determination algorithm that arbitrary views extrapolated from a single image are generated without depth comparisons. His approach is feasible, but it is limited to a single image.

We solve the visibility problem for our rendering model (forward image mapping model). To explain the solution, in this chapter we first analyze the problem and interpret it using epipolar geometry. Consequently, the observations presented in the visibility interpretations are indispensable information for understanding of our approach. Next, we study the McMillan's algorithm [1]. The key idea, advantages, and unfortunately some faults, of his approach are illustrated visually. Then, the detailed description of our solution is coming next. For removing unnecessary parts of the DLC for each view rendering before the image mapping takes place, the culling scheme has been analyzed and formulated extensively. The performance analysis for both processes is conducted at the end of this chapter.

5.1 Interpretation of Visibility Using Epipolar Geometry

Recall from Chapter 2.2, a bundle of epipolar planes commonly intersect a line passing through two camera optical centers associated with the source and the destination images. The visibility of reference image points with respect to the optical centers of desired images can be characterized based on the following two observations. First, any 3-D space point, except those points lying on the line joining both camera optical centers, can be uniquely classified into one of these epipolar planes of the bundle where it lies on. Second, every 3-D point on the same epipolar plane should project to the same epipolar line in any image.

We see the independence among epipolar planes in terms of visibility measurement. In other words, no 3-D points on an epipolar plane can occlude or be occluded by any 3-D points on the other epipolar planes. Without loss of validity, the visibility measurement only needs to be performed on those 3-D points lying on the same epipolar plane; likewise, the pixels on the same epipolar line in image space. Additionally, the nature of computational parallelism is clear to see in the visibility determination process among epipolar planes. Our attention is paid on the determination of the traverse order along the epipolar line in such a way that all pixels, mapping to the same location in the destination image, are guaranteed to arrive in back-to-front order.

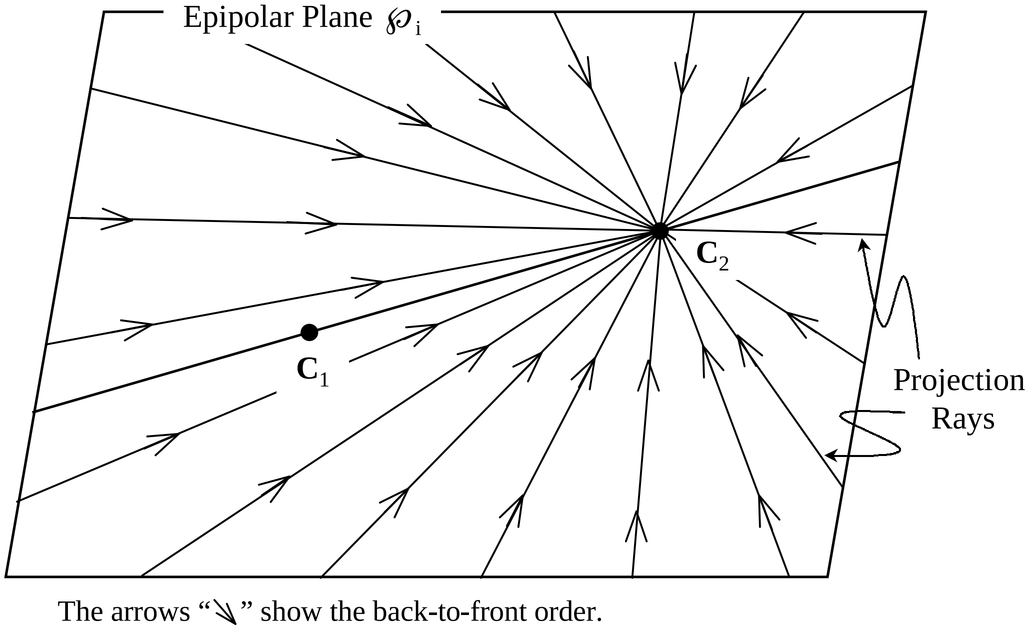


Figure 5.1: Back-to-front visibility order with respect to the destination camera optical center $\mathbf{C}_2$.

Considering an epipolar plane $\wp_i$, as shown in Fig. 5.1, there is an infinite number of projection rays emitting from the destination camera optical center $\mathbf{C}_2$ outwards[1]. Every 3-D point lying on the same projection ray projects onto the same image pixel of the destination image. To ensure a valid visibility along each projection ray, the 3-D points should arrive onto the image plane in back-to-front order with respect to $\mathbf{C}_2$, which is shown by those arrows heading to $\mathbf{C}_2$ inwards. Moreover, the visibility measurements are allowed to be computed independently from one projection ray to the others.

We denote the source image plane as I_1 and its associated camera optical center as $\mathbf{C}_1$, similarly, I_2 and $\mathbf{C}_2$ for the destination image. We call an eipipole on the source image plane *positive epipole*, denoted as $+\mathbf{e}_1$, if the point $\mathbf{C}_2$ with respect to the source image's camera coordinate system has positive Z-value (i.e. $\mathbf{C}_2 = (X_{\mathbf{C}_2}, Y_{\mathbf{C}_2}, Z_{\mathbf{C}_2})_{\mathbf{C}_1}$ and $Z_{\mathbf{C}_2} \geq 0$). Otherwise, we call it *negative epipole*, denoted as $-\mathbf{e}_1$.

Two geometric forms of source image are considered, one is planar and the other is cylindrical. We start from the planar image case. All the epipolar lines appear straight and intersect at the epipole, which is either positive or negative. Figures 5.2(a) and 5.3(a) depict the 3-D situations of a positive and a negative epipole on the image plane I_1 respectively. Let us consider two projection rays r_1 and r_2 emitted from $\mathbf{C}_2$ along the epipolar plane $\wp_i$. The back-to-front visibility orders, $\mathbf{P}_1$ before $\mathbf{P}_2$ and $\mathbf{P}_3$ before $\mathbf{P}_4$, imply a correct traverse order in the image plane I_1. In the positive epipole case, the image mapping should traverse inwards from the image boundary(ies) to the positive epipole as shown in Fig. 5.2(b). Inversely, in the

[1]The arrows on the projection rays, showing the back-to-front order of the visibility-constraint with respect to the center $\mathbf{C}_2$, should not be confused with the projection direction.

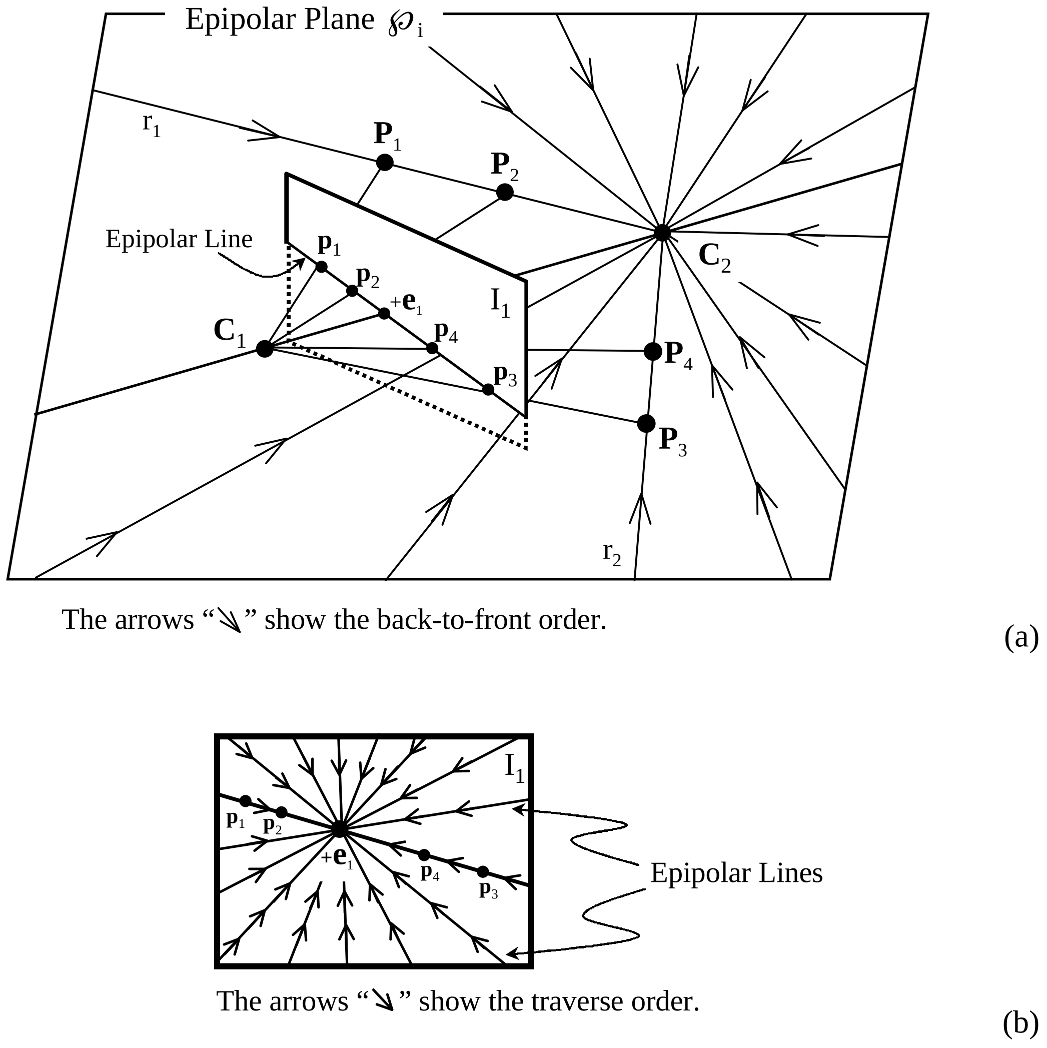
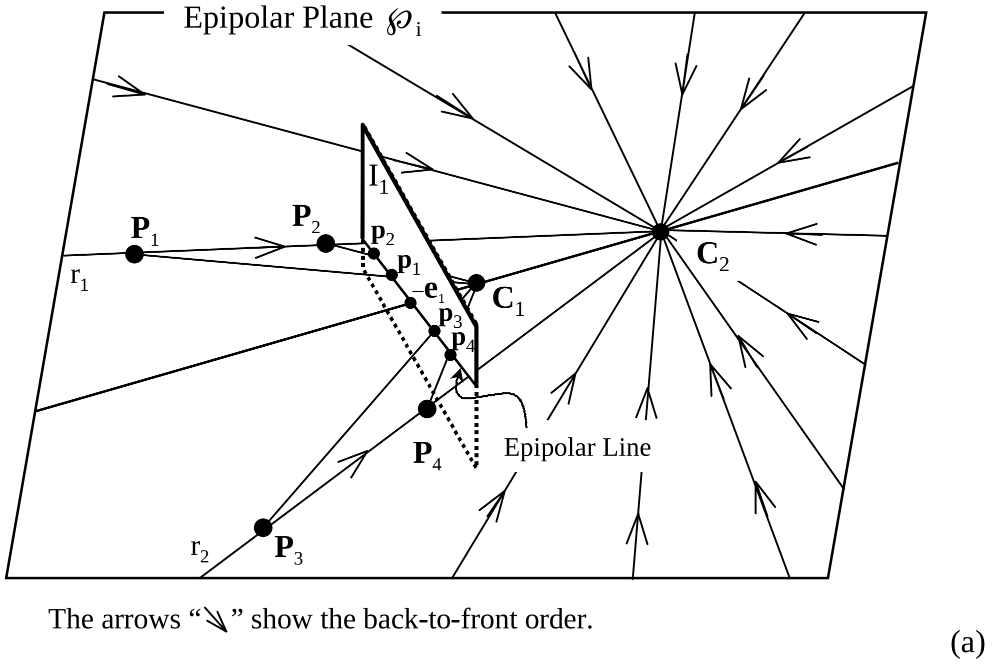


Figure 5.2: Geometric interpretation of visibility for a planar image with a positive epipole. (a) A 3-D case where the correct visibility orders with respect to the destination camera optical center $\mathbf{C}_2$ are: $\mathbf{P}_1$ before $\mathbf{P}_2$, $\mathbf{P}_3$ before $\mathbf{P}_4$ along the projection rays r_1 and r_2. (b) The actual traverse order for the image plane I_1.

negative epipole case, shown in Fig. 5.3(b), the traverse starts outwards from the negative epipole to the boundary(ies) of the source image.

Similar observations in the cylindrical case, the traverse order from the negative epipole to the positive epipole is easily recognized from Fig. 5.4(a). However, one may notice that the epipolar lines on the cylindrical image are no longer straight, compared with the planar case. Figure 5.4(b) illustrates the curved epipolar lines, with the correct traverse orders indicated by arrows, on the unfold cylindrical image. This may cause a difficulty for the implementation of traversing along the curved epipolar lines on the source image.

A key observation of visibility interpretation in epipolar geometry is that the traverse order of the image mapping must follow the epipolar lines in such an order, from the negative epipole to the positive epipole, that the visible surfaces are correctly determined in the destination image. Besides, L. McMillan has proved that this observation holds for all image geometric forms in [1].



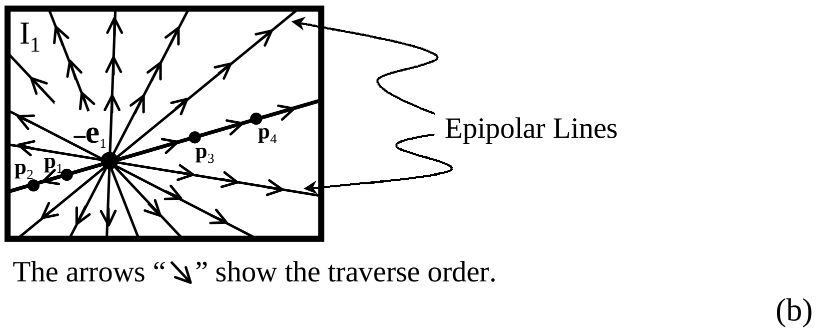


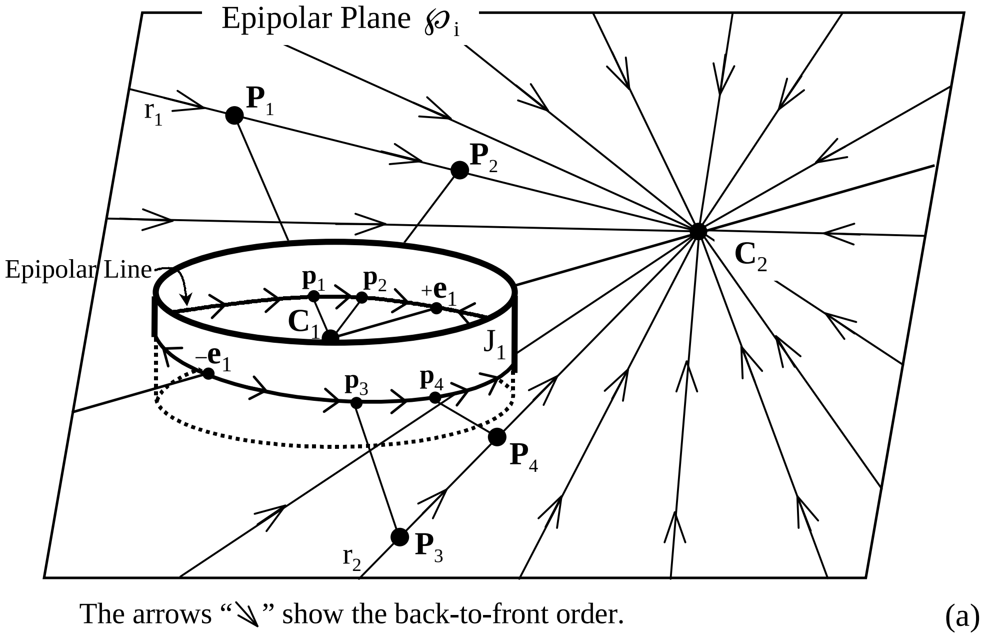
Figure 5.3: Geometric interpretation of visibility for a planar image with a negative epipole. (a) A 3-D case where the correct visibility orders with respect to the destination camera optical center $\mathbf{C}_2$ are: $\mathbf{P}_1$ before $\mathbf{P}_2$, $\mathbf{P}_3$ before $\mathbf{P}_4$ along the projection rays r_1 and r_2. (b) The actual traverse order for the image plane I_1.

5.2 McMillan's Visibility Algorithm

In this section, we study L. McMillan's visibility algorithm for the two different image forms. First, the planar image case is investigated. The advantages and limitations of his approach are also discussed. Second, we illustrate his algorithm for the cylindrical image case. Consequently, few situations his approach fails are elaborated.

5.2.1 Planar Image

L. McMillan summaries all the possible occurrences of the visibility determination for a planar image into two groups, one for positive epipole case and the other for negative epipole case. Each group contains nine possible cases. Each case is defined by a partition of the image plane into *regions*. The partitioning is done by extending lines along the actual image boundaries to infinity, which results into nine regions as shown in Fig. 5.5(a). The actual image E_1, outlined by the thicker rectangle, occupies region 5 or region 14 in the



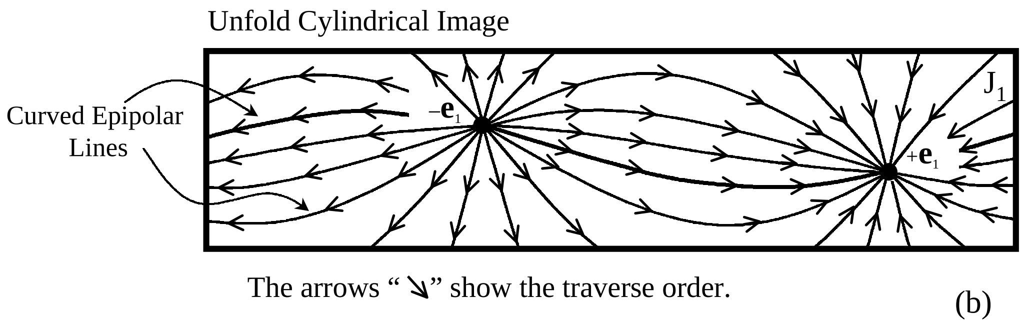


Figure 5.4: Geometric interpretation of visibility for a cylindrical image with a positive and a negative epipoles. (a) A 3-D case where the correct visibility orders with respect to the destination camera optical center $\mathbf{C}_2$ are: $\mathbf{P}_1$ before $\mathbf{P}_2$, $\mathbf{P}_3$ before $\mathbf{P}_4$ along the projection rays r_1 and r_2. (b) The actual traverse order for the image cylinder J_1.

negative epipole or the positive epipole groups, respectively, and the other regions are also clearly defined.

The traverse order of the image mapping is determined by the fact to which region the positive/negative epipole belongs. As the region is determined, the correct traverse order can be looked up by its corresponding region illustrated in Fig. 5.5(b). Since all the epipolar lines in Fig. 5.5(b) are straight lines and independent from one and the other, a scan-line fashion traverse, as shown in Fig. 5.5(c), can be used to ease the implementation without loss of validity. It is otherwise very difficult to traverse systematically along the real epipolar lines depicted in Fig. 5.5(b). Furthermore, in Fig. 5.5(c), the case 1 and 12, 3 and 10, 7 and 18, 9 and 16, are exactly the same, hence only 14 cases are actually required to be implemented.

The advantages of his approach are as follows. It is independent from the scene complexity and from the depth information. It blends the visibility determination processes and the image mapping processes into one with no additional comparison computations, unlike one does it in the z-buffer algorithm. It suggests a simple and feasible method, which is the scan-line fashion traverse, and allows a real-time implementation.

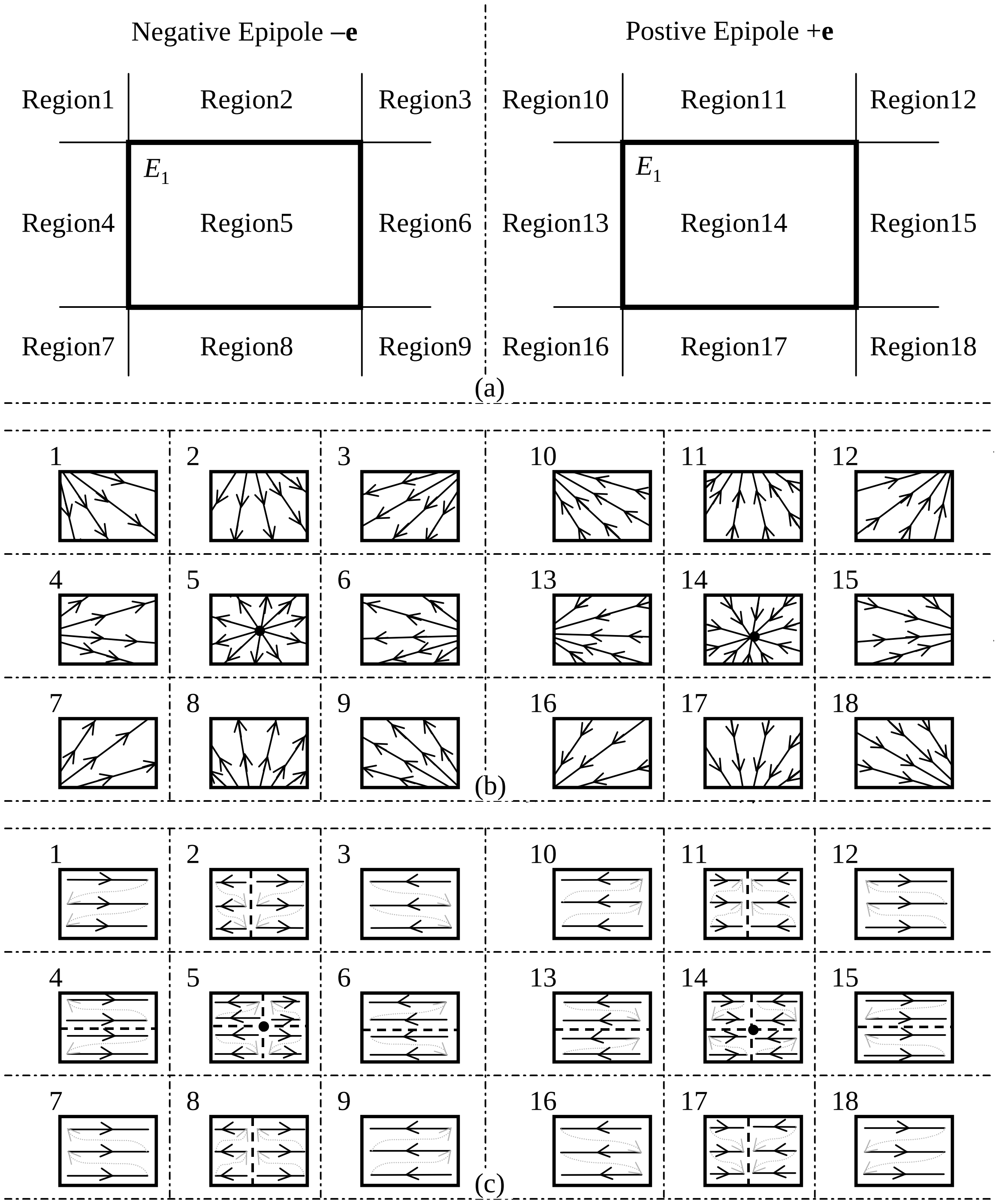


Figure 5.5: Illustrations of McMillan's visibility algorithm for a planar image. (a) All possible regions that $+\mathbf{e}_1/-\mathbf{e}_1$ may lie on. For each region where the epipole may belong to, there is a corresponding traverse order map, illustrated in (b). (c) The actual scan-line paths of the corresponding region for the underlying implementation.

Unfortunately, if more than one input (source) image is used, this approach is not applicable because we do not know which input image the traverse should start from before the others.

5.2.2 Cylindrical Image with Faults

McMillan's visibility algorithm for a cylindrical source image is illustrated in [1]. He successfully pointed out that the visibility determination can be done through the enumeration of curved epipolar lines. However, as described before, to traverse along an arbitrary path of a straight epipolar line is already a difficult task (due to the discrete nature of an image). To traverse along an arbitrary path of a curved epipolar line is even harder. He illustrated his approach, cf. Fig. 5.6(a), by first partitioning an unfold cylindrical image into four regions (labeled by the numbers in the corners) and enumerating each in the order indicated by the black, thick arrows.

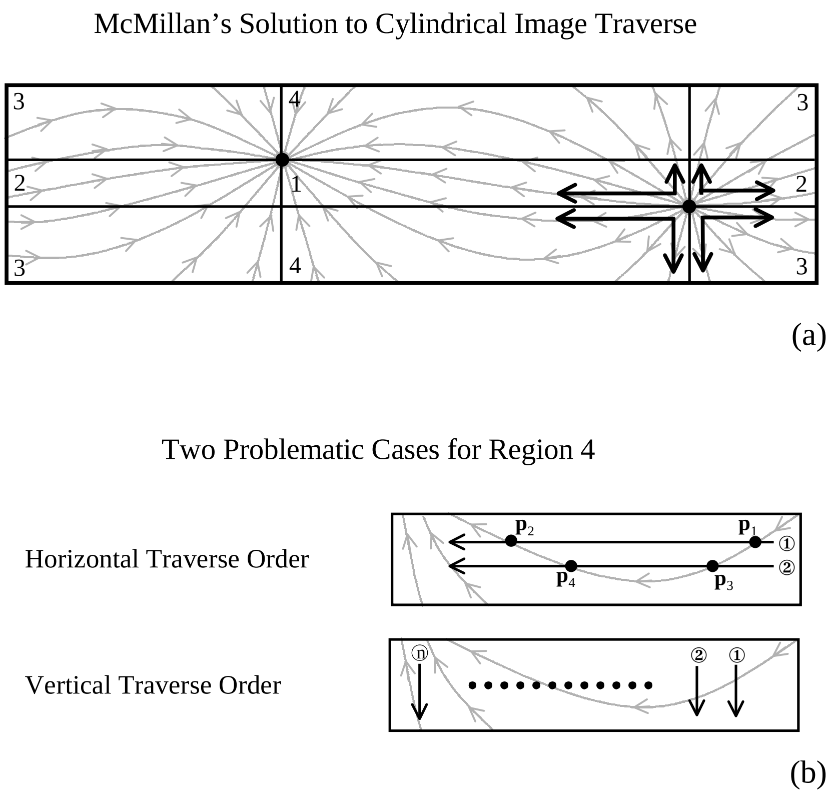

McMillan's Solution to Cylindrical Image Traverse
3
4
3
2
1
2
3
4
3
(a)
Two Problematic Cases for Region 4
Horizontal Traverse Order
p_2
p_1
①
②
p_4
p_3
Vertical Traverse Order
ⓝ
②
①
(b)

Figure 5.6: McMillan's solution for a cylindrical image, and its problems.

Unfortunately, there is a problem with his approach. Let us consider solely the region 4 in Fig. 5.6(a) to illustrate the problem. His solution (black arrows) indicates two possible traverse orders depicted in Fig. 5.6(b). One is traversing horizontally (row-scan) from top to bottom, the other is vertically (column-scan) from right to left. In the horizontal case, the enumeration reaches point $\mathbf{p}_1$ first, $\mathbf{p}_2$ second, $\mathbf{p}_3$ third and $\mathbf{p}_4$ last, but this does not follow the correct traverse order which is $\mathbf{p}_1$ first, $\mathbf{p}_3$ second, $\mathbf{p}_4$ third and $\mathbf{p}_2$ last. Furthermore, in the vertical case, when the enumeration reaches the nth column, the traverse direction (downwards) is exactly opposite to the correct order (upwards). So both of them fail to follow the correct order along the curved epipolar lines in the cylindrical image.

5.3 Our Solution

In our system, the visible surface determination process contains two sub-processes, culling and traverse order determination. The culling is performed at first so only relevant portion of scene data stored in the DLC is traversed and forwarded to the destination image. We discuss the culling in a great detail in the next section. In this section, we focus on the second sub-process — traverse order determination.

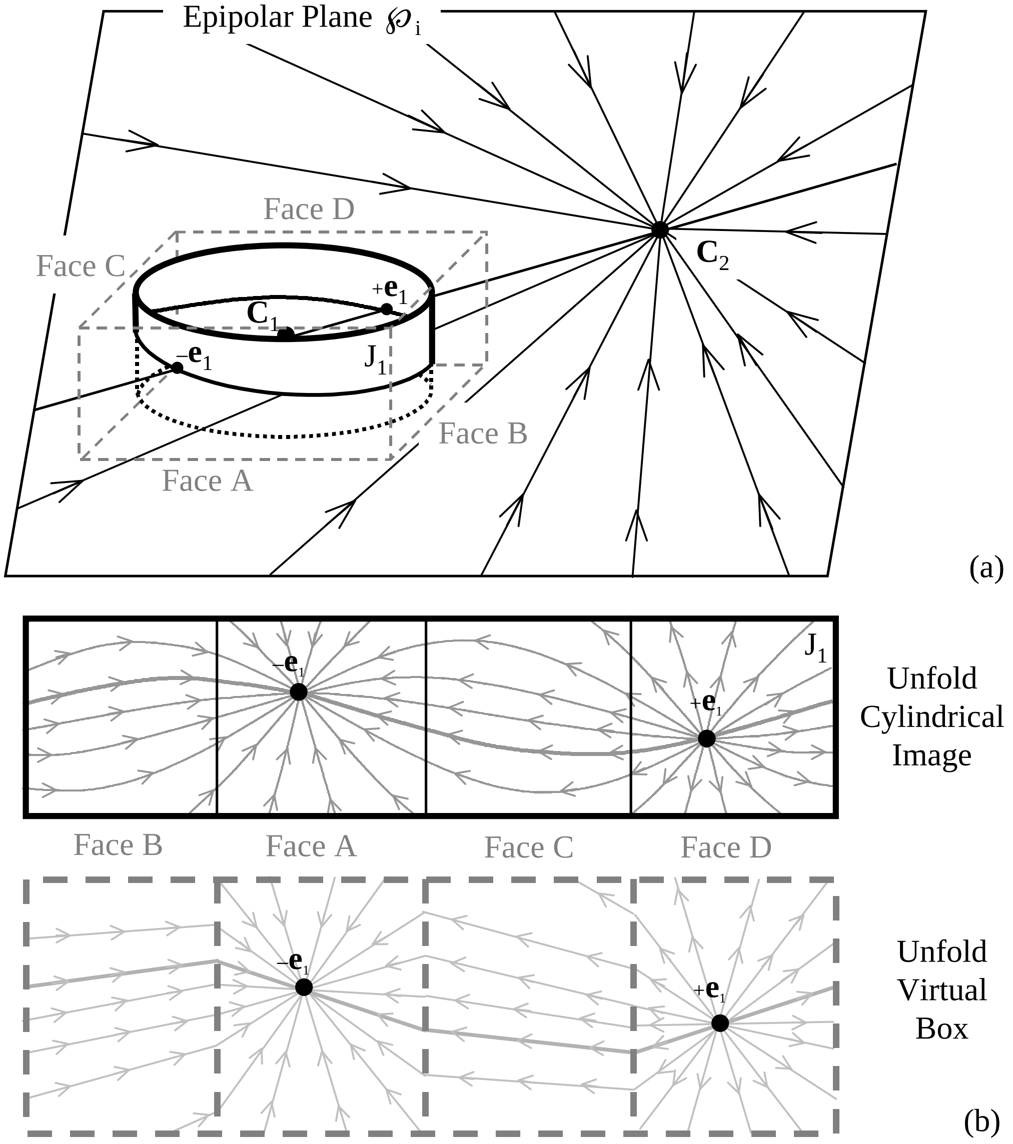


Figure 5.7: Illustration of an ordering map (virtual box). (a) It shows a virtual box, with no top and bottom faces, surrounding a cylinder (i.e. the DLC). (b) The straightness of reprojected epipolar lines on the unfold virtual box below in comparison with the curved epipolar lines on the unfold cylindrical image above.

In the last section, we have discussed the key idea, advantages and problems of McMillan's algorithm. To utilize his key idea and take advantages of his approach, two problems need to be solved in advance. The first is the multiple source-images ordering problem; the second is the difficulty of traversing along the curved path in the cylindrical image. Based on our scene representation, we explain our solution to the second problem first and then come to the first problem later on.

Our approach to the curved traverse problem is to reuse the provably correct property of McMillan's visibility algorithm for planar images. Let us imagine a virtual box (without top and bottom faces) surrounding a cylinder, such as depicted in Fig. 5.7(a). The planarity of each box face guarantees the straightness of the intersection lines. Based on this, the curved epipolar lines would appear as straight as they are projected onto the planar surfaces, the faces of the box. Figure 5.7(b) shows the straightness of the projected epipolar lines on an unfold virtual box in comparison to the original curved epipolar lines on the unfold cylindrical image above.

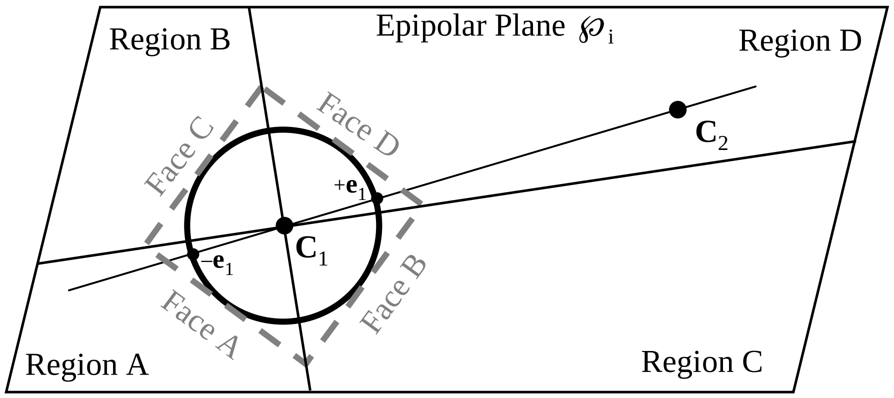


Figure 5.8: Four disjoined partitions of an epipolar plane for the traverse-order determination of four virtual box faces.

Instead of calculating the complicated curve function of traverse paths on the DLC for every new view rendering, we simply use the box as an ordering map that is only computed once when the DLC is constructed. The connection between the ordering map (the box) and the DLC is built by projecting the cylinder's surface onto each face of the box. Without loss of generality, we can then virtually traverse along epipolar lines on the face(s) of the box using McMillan's visibility algorithm for a planar image, yet actually traverse along the curved epipolar lines in the DLC.

The reason why every face of the box can be treated separately is because it partitions each epipolar plane into four disjoined regions as shown in Fig. 5.8. As long as those 3-D scene points of the furthest region, with respect to the destination camera optical center, are projected to the destination image first and those of the closest region at last, then our approach guarantees the back-to-front order in the destination image. The rule of thumb is that the traverse should always start from the face that contains the projection of the negative epipole, and the face containing the positive epipole at last. For instance, in Fig. 5.8, region A, containing $-\mathbf{e}_1$, is the furthest region with respect to $\mathbf{C}_2$, and region D, containing $+\mathbf{e}_1$, is the closest one. Hence we start from the face A, then B or C, and finally the face D. In practice, at most two faces may be required for each new view rendering because the others may be culled away (see next section).

Our solution to the second problem directly inherits all advantages of McMillan's approach. Furthermore, we have also shown in Chapter 4 that the scan-line traverse, proposed by our solution for the visibility determination, ensures the feasibility of fast rendering through the incremental mapping computation. It is therefore simple and practical and supports a real-time implementation. Consequently, the establishment of such a method for the second problem implicitly contributes to the solution of the first problem — the multiple source-images ordering problem.

Since our scene representation, the DLC, is already capable to store scene information from multiple images as well as we have build up the access to ensure a back-to-front order for the cylindrical model in the previous section, the solution of the multiple source-images ordering problem is simplified. For each class of the DLC (i.e. each pixel of a cylindrical image) along the traverse path, the associated linked list is traversed from the last element to the first. Because the linked list is pre-sorted by depth, where the closest scene data with respect to the center of the DLC is stored at first place in the linked list, this method guarantees the back-to-front order.

5.4 Culling

Traditional computer graphics approaches, such as back-face culling, remove the invisible parts of objects before starting expensive rendering computations. The same concept can also be applied to our rendering system. Our culling process is performed before anything else in the rendering phase of the view synthesis system. The culling process does not eliminate all the occluded scene points, instead, we remove a portion of the DLC which is out of the frustum of the virtual camera projection. So only the remainder portion (subset) of scene data in the DLC is actually rendered for each new view generation.

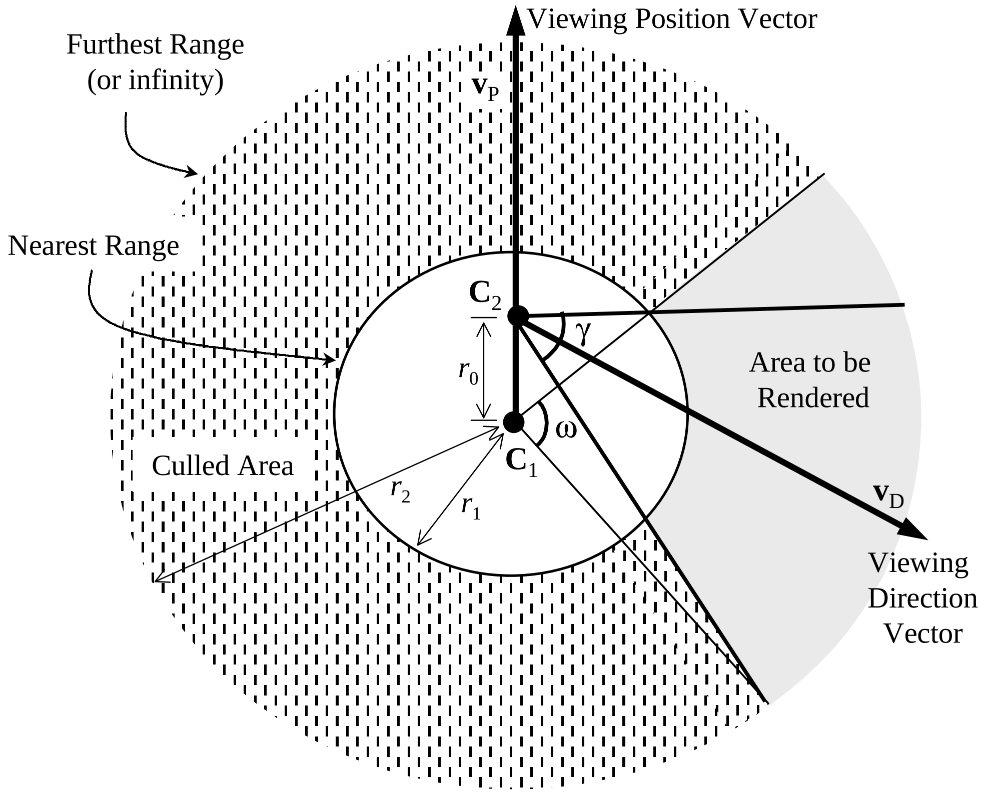


Figure 5.9: The setup for culling computation, where $\mathbf{C}_1$ is the center of the cylinder (the DLC) and $\mathbf{C}_2$ is the optical center of a virtual camera.

Figure 5.9 depicts the culling computation setup. The unbounded outer circle stands for the furthermost range of scene data in the DLC, i.e. the maximum depth value. Similarly, the inner circle is for the nearest range. A point $\mathbf{C}_1$ is fixed at the center of two cycles (the center of the DLC) and a point $\mathbf{C}_2$ indicates the optical center of a virtual camera that can be moved arround from frame to frame within the inner circle during the rendering. The inner circle encloses the valid navigation area, where we assume there is no object inside this region so no collision detection is required. The vector $\mathbf{v}_D$ indicates the virtual camera's viewing direction and vector $\mathbf{v}_P$, which starts from the point $\mathbf{C}_1$ passing through the point $\mathbf{C}_2$ to infinity, indicates the virtual camera's position with respect to the center of the DLC. The distance between $\mathbf{C}_1$ and $\mathbf{C}_2$ is r_0, the radius of the inner circle is r_1, and the radius of the outer circle is r_2. Note that the value of r_2 may tend to infinity theoretically, but practically we can assign a fairly large number for an underlying implementation. The values r_1 and r_2 are known and kept constant once the scene reconstruction is built up. Both vectors $\mathbf{v}_P$ and $\mathbf{v}_D$ are dynamically determined by the values r_0 and the field of view of the virtual camera γ. The angle ω indicates the angle of the portion to be rendered. Given the optical center of a virtual camera $\mathbf{C}_2$, a viewing direction $\mathbf{v}_P$ and a field of view of the virtual camera γ, the culling process is to calculate two boundaries between the lighter gray area (to be rendered) and the vertical-bar filled area (to be culled away).

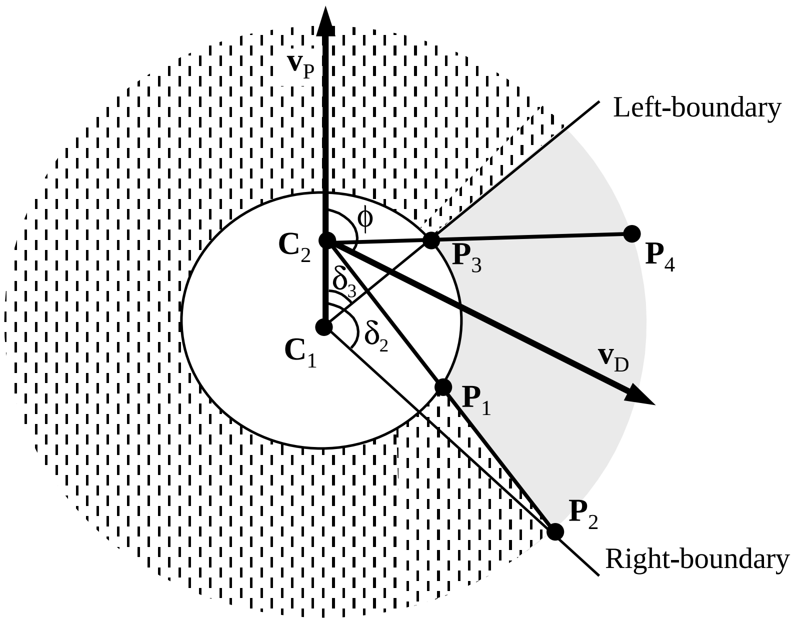


Figure 5.10: Culling geometry.

We call the boundaries *left-boundary* and *right-boundary* as shown in Fig. 5.10. They are specified as viewing from the center $\mathbf{C}_1$ to the light gray area. Geometrically, the left- or right-boundary is defined by the angle to the vector $\mathbf{v}_P$. For instance, the left-boundary is specified by angle $\angle\delta_3$, and the right-boundary by angle $\angle\delta_2$ in Fig. 5.10. The angle $\angle\delta_n$ is clockwise starting from vector $\mathbf{v}_P$, defined formally as

$$\angle\delta_n = \angle\mathbf{C}_2\mathbf{C}_1\mathbf{P}_n, \quad \text{where} \quad 0^\circ \leq \delta_n < 360^\circ.$$

Points $\mathbf{P}_i$ indicate the intersections of the virtual camera's frustum to the inner and the outer cycles. Our target is to calculate the two angles associated with the left-boundary and the right-boundary.

The virtual camera can freely move around within the inner circle and take a shot towards any direction. The angle $\angle\delta_n$ for the left- and the right-boundary are therefore varied arbitrarily. We summaries all the occurrences into four cases, as shown in Fig. 5.11. Each case is defined by a value of ϕ which is the angle between vectors $\mathbf{v}_P$ and $\mathbf{v}_D$. The specifications of these four cases are as follows.

Case 1: When $0 \leq \phi \leq \frac{\gamma}{2}$ or $\left(360 - \frac{\gamma}{2}\right) \leq \phi < 360$, then
the left-boundary is specified by $\angle\delta_4$ and the right-boundary is specified by $\angle\delta_2$.

Case 2: When $\frac{\gamma}{2} < \phi < \left(180 - \frac{\gamma}{2}\right)$, then
the left-boundary is specified by $\angle\delta_3$ and the right-boundary is specified by $\angle\delta_2$.

Case 3: When $\left(180 - \frac{\gamma}{2}\right) \leq \phi \leq \left(180 + \frac{\gamma}{2}\right)$, then
the left-boundary is specified by $\angle\delta_3$ and the right-boundary is specified by $\angle\delta_1$.

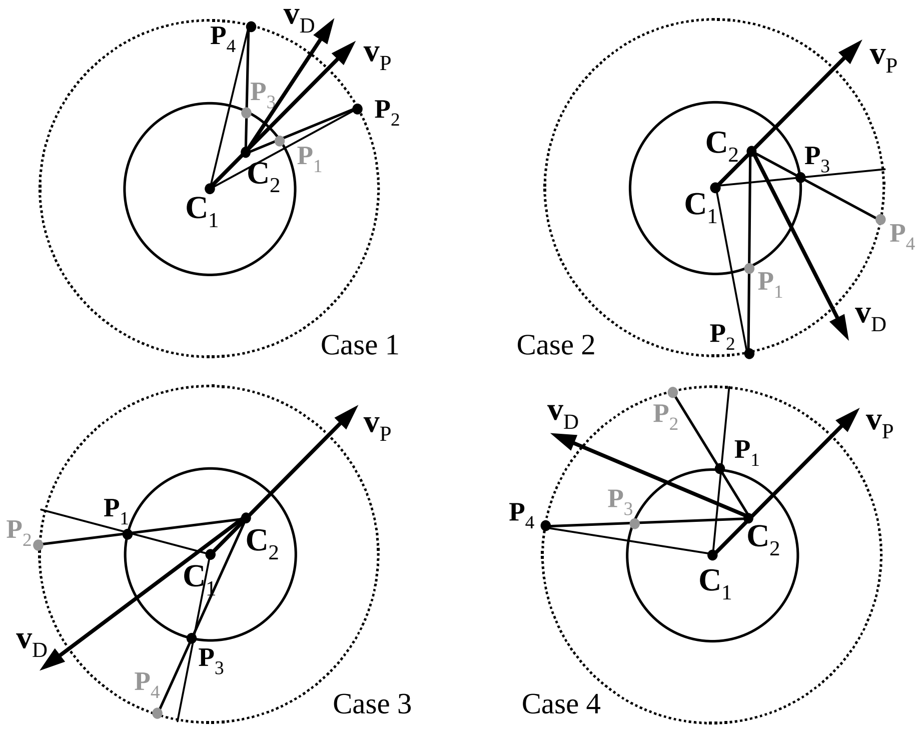


Figure 5.11: Four possible culling cases.

Case 4: When $\left(180 + \frac{\gamma}{2}\right) < \phi < \left(360 - \frac{\gamma}{2}\right)$, then the left-boundary is specified by $\angle\delta_4$ and the right-boundary is specified by $\angle\delta_1$.

Given the vectors $\mathbf{v}_P$ and $\mathbf{v}_D$, we can then derive a general equation to calculate the angle δ_n for $n \in [1, 2, 3, 4]$,

$$\delta_n = (\varphi - 90^o) + \cos^{-1}\left(\frac{r_0 \sin\varphi}{r_i}\right),$$

where i$\in \{1, 2\}$. To calculate the left-boundary, we have

$$\varphi = \phi - \frac{\gamma}{2};$$

for the right-boundary,

$$\varphi = \phi + \frac{\gamma}{2},$$

where φ is an angle between $\mathbf{v}_P$ and a ray from $\mathbf{C}_2$ to $\mathbf{P}_n$. Figure 5.12 depicts this geometry.

5.5 Performance Analysis

The time complexity is constant, O(1), for the process of the traverse order determination. Regardless of the amount of the scene data stored in the DLC, only a projection calculation for the epipole and a classification calculation (switch-cases computation) for the case recognition of the epipole are required to determine the traverse order for each new view rendering. The information used for those calculations is configurations of the virtual camera and the DLC model (e.g. the projection matrix of a virtual camera and the DLC's

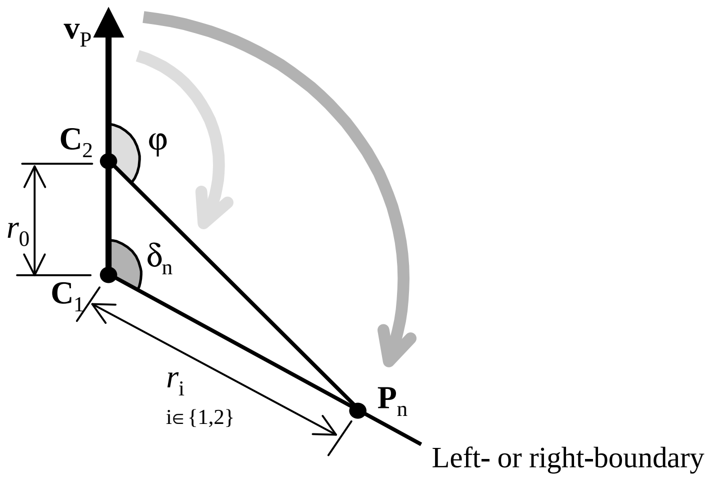


Figure 5.12: Geometry for calculation of culling boundaries.

height, radius, coordinate system, etc). No further information or procedure is involved in this process. So the process is independent from the scene complexity (e.g. depth complexity) as well as the quantity of the scene data stored in the DLC.

For the culling process, what we are interested in is by how much percentage of upper bound and lower bound the DLC can be culled away. To evaluate this, an angle ω is computed (by subtracting the left-boundary angle from the right-boundary angle) which gives an idea of how much the DLC is required to be rendered. Complementarily,

$$\left(1 - \frac{\omega}{360}\right) \times 100\% \tag{5.5.1}$$

shows how much percentage of the DLC is culled away. The definition of this angle ω, for all the cases, is specified below,

$$\omega = \begin{cases} \gamma + cos^{-1}\left(\frac{r_0}{r_2} sin\left(\phi + \frac{\gamma}{2}\right)\right) - cos^{-1}\left(\frac{r_0}{r_2} sin\left(\phi - \frac{\gamma}{2}\right)\right), \\ \qquad \text{if } 0 \le \phi \le \frac{\gamma}{2} \text{ or } \left(360 - \frac{\gamma}{2}\right) \le \phi < 360; \\ \gamma + cos^{-1}\left(\frac{r_0}{r_2} sin\left(\phi + \frac{\gamma}{2}\right)\right) - cos^{-1}\left(\frac{r_0}{r_1} sin\left(\phi - \frac{\gamma}{2}\right)\right), \\ \qquad \text{if } \frac{\gamma}{2} < \phi < \left(180 - \frac{\gamma}{2}\right); \\ \gamma + cos^{-1}\left(\frac{r_0}{r_1} sin\left(\phi + \frac{\gamma}{2}\right)\right) - cos^{-1}\left(\frac{r_0}{r_1} sin\left(\phi - \frac{\gamma}{2}\right)\right), \\ \qquad \text{if } \left(180 - \frac{\gamma}{2}\right) \le \phi \le \left(180 + \frac{\gamma}{2}\right); \\ \gamma + cos^{-1}\left(\frac{r_0}{r_1} sin\left(\phi + \frac{\gamma}{2}\right)\right) - cos^{-1}\left(\frac{r_0}{r_2} sin\left(\phi - \frac{\gamma}{2}\right)\right), \\ \qquad \text{if } \left(180 + \frac{\gamma}{2}\right) < \phi < \left(360 - \frac{\gamma}{2}\right), \end{cases} \tag{5.5.2}$$

where we assume that

$$0 \le r_0 \le r_1,\ r_1 < r_2,\ 0^\circ \le \phi < 360^\circ,\ \text{and } 20^\circ \le \gamma \le 70^\circ.$$

Note that the γ set between 20 to 70 degrees is reasonable for a wide range of applications.

Let us consider a particular γ. The achievement of a maximum culling is equivalent to the task to minimize the function

$$\mathcal{G}_{pos}(\phi) = cos^{-1}\left(\frac{r_0}{r_i} sin\left(\phi + \frac{\gamma}{2}\right)\right),$$

and to maximize the function

$$\mathcal{G}_{neg}(\phi) = cos^{-1}\left(\frac{r_0}{r_i} sin\left(\phi - \frac{\gamma}{2}\right)\right),$$

where i$\in \{1, 2\}$, with respect to the same value of ϕ. Similarly, we have a minimum culling, when function $\mathcal{G}_{pos}(\phi)$ is maximized and function $\mathcal{G}_{neg}(\phi)$ is minimized, with respect to the same value of ϕ. We conclude that when $\phi = 0°$ we have maximum culling, and when $\phi = 180°$ we have minimum culling. Figure 5.13 depicts two possible cases, one for the maximum culling and the other for the minimum culling.

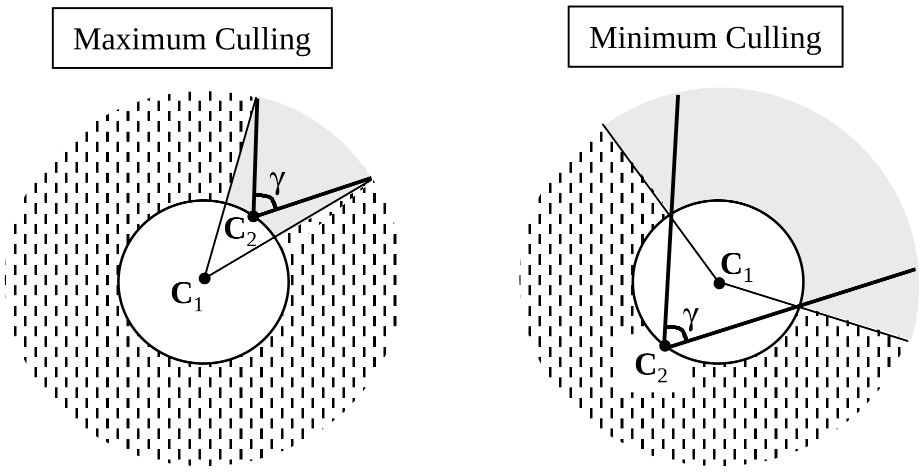


Figure 5.13: The possible cases of maximum and minimum cullings.

As in a usual indoor scene, for instance, the relation of the nearest and the furthest scene range may have $r_2 = 2r_1$. If $\gamma = 45°$ (average horizontal field of view) is used for a virtual camera, the maximum culling goes up to 93.6% of the DLC and 75% for the minimum culling, computed by Eqs. 5.5.2 and 5.5.1. Similarly, for the outdoor scene with $\gamma = 45°$ and the furthest scene range to the infinity, $r_2 \to \infty$, the maximum culling goes up to 87.5% of the DLC and 75% for the minimum culling.

Chapter 6

Depth Recovery Development System

An image stores 3-D data into 2-D form. One third of geometric information is lost from this transformation during the image acquisition process. There are many computer vision techniques, described in [16, 29, 30], available to recover the lost part of 3-D information (depth) exclusively from given images. They commonly require that at least two cameras capture a static scene in a certain time interval when lighting conditions in the scene are assumed tolerably constant. Unfortunately, they may not guarantee a highly accurate depth measurement as the result. Nevertheless, this stereo-based approach is more accessible and affordable to gain the 3-D information of a scene compared to a use of specialized equipment for range measurement.

Depth-from-stereo using correspondence analysis has been well studied for its long history yet remains difficult and unstable. It is an ill-posed mathematical problem. Many algorithms in [16, 31, 29, 30, 32, 33] have been developed to challenge different aspects in the stereo correspondence problem. They may perform surprisingly well in one particular image contents but poor in the others due to content dependency. Furthermore, their attentions focus merely on featured data in the images, such as edge and corner, and yield either a sparse depth map from matched featured-data or a dense depth map where the depths of unmatched pixels are coarsely interpolated. The later enforces foreground objects incorrectly adhering with background while the former delivers insufficient depth data to its applications where the pixel-wise depth may be required. Consequently, the accuracy of recovered depth from those discrete nature of input data (digital image) is inherently inexact, even through the correspondences of stereo images are perfectly established.

In order to contend with those problems for our application we propose an alternative approach — a systematical approach. A semi-automatic system, generating a dense disparity map from a pair of standard binocular stereo images (left/right image in short), is constructed. Human intervention is adopted so high-level problems and ambiguities raised from performing correspondence search can be easily solved. In the remainder of this chapter, we start from describing our motivation to this development system, followed by a blueprint of this system. Those intermediate experimental results through each process in the system are shown in the last part of this chapter.

6.1 Motivations

Automatically constructing correspondences between pairs of stereo images is unfortunately fairly fragile and may not always find the correct correspondences. Some well-known problems, such as homogenous region, occlusion and ordering problems, raised in the depth recovery processes, and others found in [16, 29, 30] can degrade the desired results significantly. Fully automatic depth-recovery algorithms have not yet reached to the level that dense and accurate depths can be obtained independently from scene and illumination complexity. Still, there is a long way to go till reaching that level. Instead of trapping ourselves into the short-term unsolvable problems, our intention is to seek an access of obtaining dense and reasonable quality depths instantly.

In spite of many stereo image matching problems, to what extents those existing algorithms can serve to achieve the task is still an open question. Generally speaking, they can at least provide a coarse yet useful disparity information to a human animator who then can do the refinements. Nevertheless, they cannot deal with all the complicated cases, such as homogenous regions, occlusions and many others acquired from our complex world. On the other hand, people can easily tell, in a high-level sense, what the background and foreground objects are from the given images. Furthermore, people are good at high-level object discriminations but poor at specifying complicated tiny-scale details manually, especially in pixel-scale level. It is therefore too tedious for human to build up the correspondence, pixel by pixel, among the multiple stereo images.

It is not difficult to see the complementary relation between a human animator and the computer algorithms. Our approach, which combines both of them, is therefore feasible because it splits the task of depth recovery into sub-tasks that are easily accomplished by a person (but not a computer algorithm), and sub-tasks that are easily performed by a computer algorithm (but not a person). Moreover, as we have mentioned in Chapter 1, we do not intend to reconstruct a highly accurate 3-D model of the scene but a visually convinced approximations. A dense depth map generated by our development system in such a hybrid way is satisfactory to the application in the view synthesis system.

6.2 Blueprint

In this section a blueprint of a dense depth recovery development system is elaborated. One major difference between our approach and traditional computer vision approaches is that we make use of human cognition to recognize and specify the foreground and background objects appearing in the images before performing dense correspondence analysis using computer algorithms.

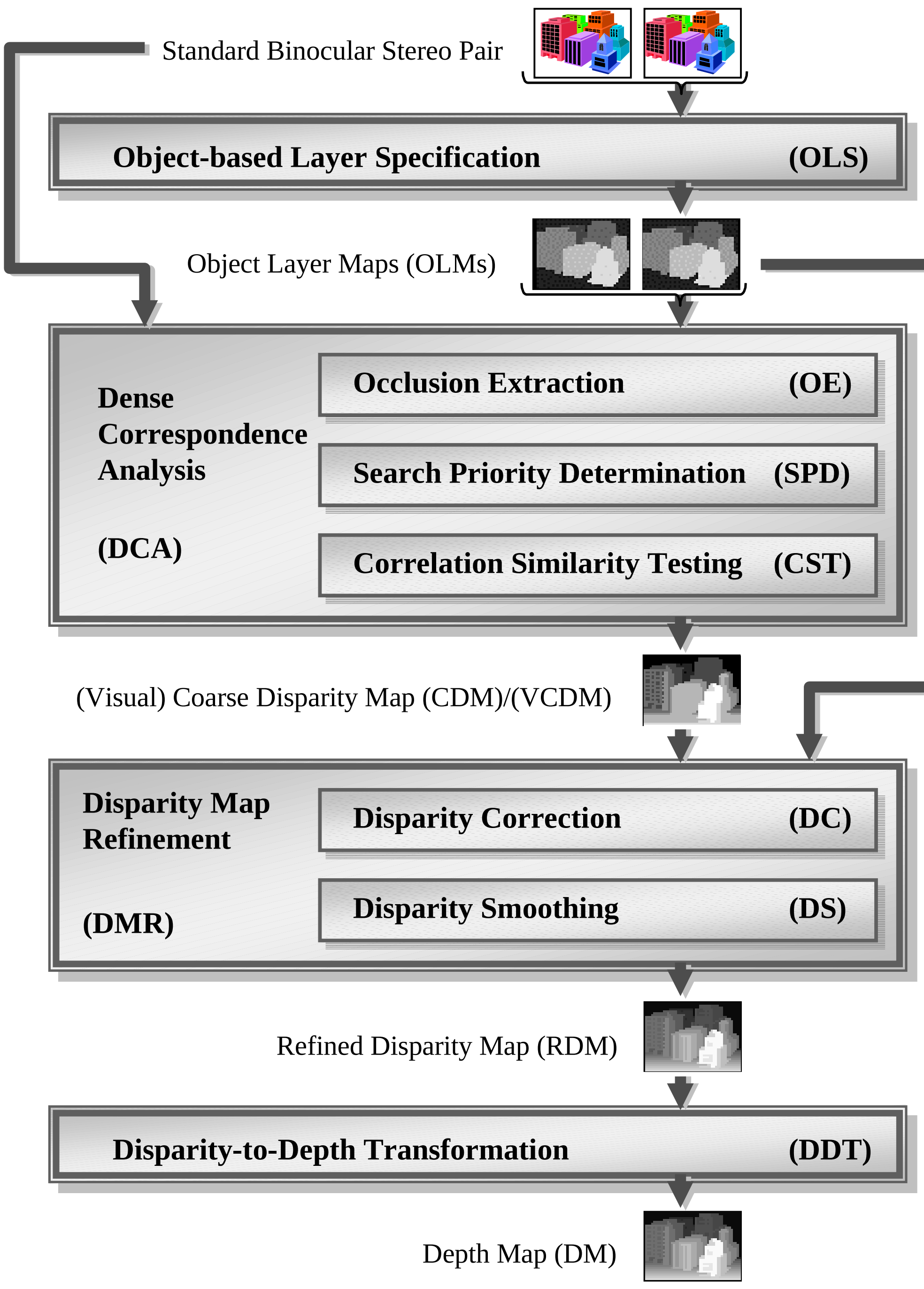
The whole development system consists of four major processes and it is organized into a pipeline structure. Each process receives one or more inputs, generated by previous processes and/or provided by an image acquisition, and produces a single output. These processes are object-based layer specification (OLS), dense correspondence analysis (DCA), disparity map refinement (DMR) and disparity-to-depth transformation (DDT). A diagram of the system is given in Fig. 6.1.

The process OLS is mainly implemented by a human animator assisted by a computer aided imaging system, such as an interactive tool *Intelligent Scissors* presented in [34]. This process aims to help solve the inter-occlusion problem[1]. The key idea of this process is to discriminate between foreground and background objects appearing in input images. Each distinctive object is specified by outlining its silhouette and annotating (filling up) the embraced region with a distinctive gray value determined by relative depth relation to the neighboring objects — the closer to the viewpoint the brighter. Correspondence between corresponding objects in the input images is build up by assigning the same gray value. The choice of object is flexible depending on its application. For instance, in augmented reality applications it is often required to align a virtual object to one or more foreground object(s) of the real-scene image but leaves the background far away behind [35]. In this case, only those applied objects, few foreground objects, need to be specified individually. Furthermore, if cyclic-occlusions are present, a splitting method is used accordingly. Given two standard binocular stereo images, the process OLS outputs object layer maps (OLMs). The OLMs containing a set of uniform intensity regions can be highly compressed using a GIF compressor with no data loss. In our experiment, $\frac{1}{50}$ compression rate was achieved. So the extra overhead it brings to the development system is small.

Once the OLMs are generated, the process DCA is performed next. Two inputs are required by the process DCA, one is a pair of standard binocular stereo images and the other is the OLMs (cf. Fig. 6.1). The process DCA can further be separated into three sub-processes. They are occlusion extraction (OE), search priority determination (SPD) and correlation similarity testing (CST). The processes OE and SPD need to be performed first before the process CST. The process OE extracts out the inter-occlusion area so the process CST skips the risk of searching through the occlusion area. The process SPD ensures that those featured data with higher intensity variation will have higher priority for the establishment of the correspondence through the process CST. The process CST finalizes the dense correspondence analysis and outputs a coarse disparity map (CDM).

Each entry of the CDM is initialized with a valid state for the correspondence search. The process OE is responsible to update this state so long as the occlusion pixels are identified. A criterion for the identification

[1]It means occlusions occurred among objects, not within an object itself.



Standard Binocular Stereo Pair
Object-based Layer Specification (OLS)
Object Layer Maps (OLMs)
Dense Correspondence Analysis (DCA)
Occlusion Extraction (OE)
Search Priority Determination (SPD)
Correlation Similarity Testing (CST)
(Visual) Coarse Disparity Map (CDM)/(VCDM)
Disparity Map Refinement (DMR)
Disparity Correction (DC)
Disparity Smoothing (DS)
Refined Disparity Map (RDM)
Disparity-to-Depth Transformation (DDT)
Depth Map (DM)

Figure 6.1: The pipeline structure of a dense depth recovery development system.

of an occlusion region is as follows. Let $\mathbf{p}_{1_L}$ and $\mathbf{p}_{2_L}$ be any two neighboring pixels on the same image row across objects' boundaries in the left image, similarly $\mathbf{p}_{3_R}$ and $\mathbf{p}_{4_R}$ in the right image. Figure 6.2 depicts this situations. An inter-occlusion region appears in the left or right image around an objects' boundary area if $OLM_{right}(\mathbf{p}_{4_R}) > OLM_{right}(\mathbf{p}_{3_R})$ or $OLM_{left}(\mathbf{p}_{1_L}) > OLM_{left}(\mathbf{p}_{2_L})$. Wherever this criterion is met, a normalized correlation similarity testing is held to find the corresponding points along predefined searching intervals over corresponding object regions (Fig. 6.2 indicates the searching paths) using a shape-adaptive image window. The essential idea of this specialized window is to exclude the irrelevant/misleading information that a normal rectangular or circular window would cover up in comparing the similarities between the two windows. Figure. 6.3 illustrates this idea in comparison with a standard rectangular image window.

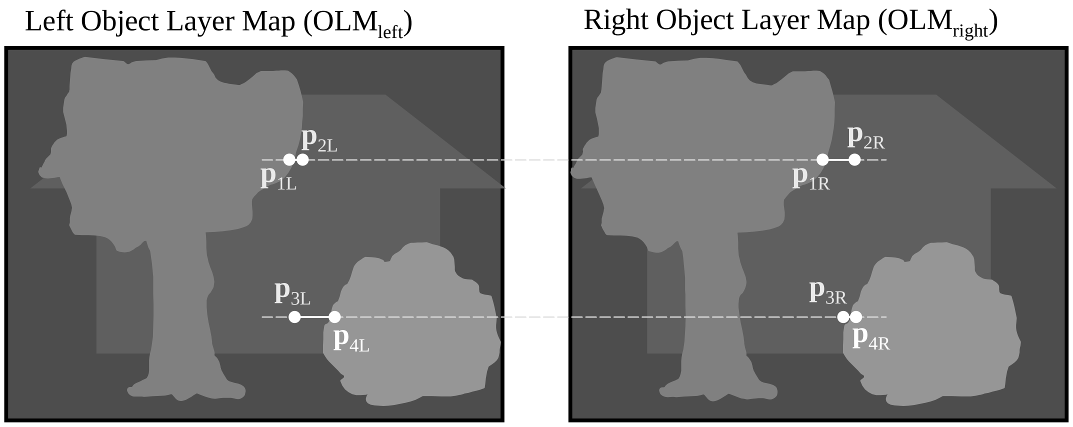

Left Object Layer Map (OLM_{left})
Right Object Layer Map (OLM_{right})
p_{2L}
p_{1L}
p_{3L}
p_{4L}
p_{2R}
p_{1R}
p_{3R}
p_{4R}

Figure 6.2: Illustration of occlusion identification.

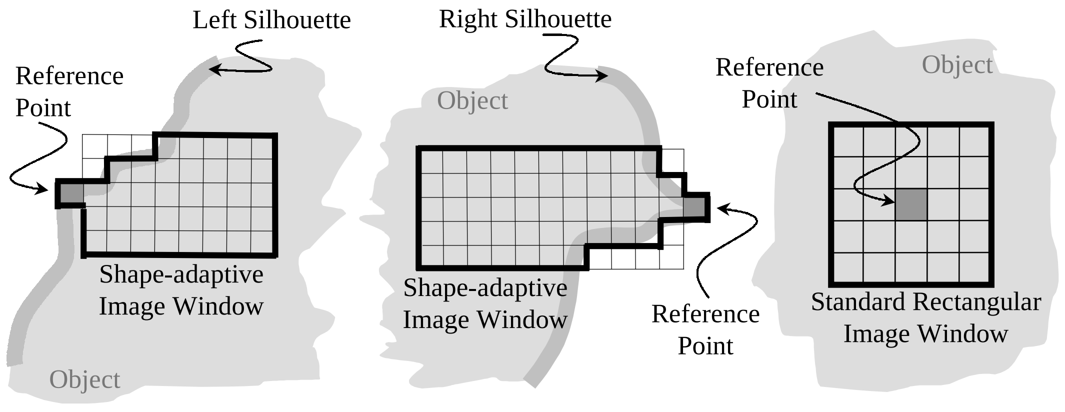


Figure 6.3: Shape-adaptive image windows VS. a standard rectangular image window.

Notice that the thin grid window defines the template of a shape-adaptive image window associated with the current reference point (one of silhouette points), and the thick border defines the real image window from where the correlation coefficients are actually computed. The shape of the window is dynamically determined depending on the shape of the object silhouette. Nevertheless, once it is defined, all its corresponding windows along the searching interval in the other image are fixed with that defined shape. As the corresponding boundary points are found, as in Fig. 6.2, for instance $\mathbf{p}_{1_R}$ and $\mathbf{p}_{2_R}$ from $\mathbf{p}_{2_L}$ and $\mathbf{p}_{2_L}$, the process OE simply updates states of entries between $\mathbf{p}_{1_R}$ and $\mathbf{p}_{2_R}$ in CDM, where the occlusion region is identified. The process CST then consults this state before entering a search procedure for each entry.

The process SPD presorts the searching order for each image row, so a pixel in a row with higher intensity variation has a higher priority. The method used in the presorting is Sobel operator, where each channel is computed separately and the final output is the average of the three. The process CST uses this order to traverse each image row and follows the rule of first-found-first-occupy. This rule restricts less-featured pixels' corresponding search into a certain interval that is constructed by preoccupation of those more featured pixels under the monotonicity ordering constraint. The searching method used in the process CST is the normalized cross-validation correlation algorithm [36]. The output of the process DCS is a coarse yet dense disparity map (CDM).

To be able to refine the CDM by a human animator, a friendly user interface should be provided. Without loss of generality, we can scale the CDM up by a factor $255/Max(CDM)$ and output a gray value image, i.e. visual coarse disparity map (VCDM). The resulting VCDM gives not only a vivid relative-depth visualization (the brighter the closer to the viewpoint) but also a visual representation so a graphical editing for the refinement becomes possible. Further strengthening the connection between the VCDM and actual scene in the image, a layer-concept capable imaging system can be used so a human animator is allowed to perform the disparity map refinement (DMR) visually in an easy yet effective manner. The disparity correction (DC) corrects minor problematic areas of the VCDM, caused by those unsolved stereo problems. The disparities recovered, as said before, are very coarse. The continuity of disparities is sometimes lower than G^0[2] even for a simple surface patch. The process of disparity smoothing (DS) is therefore adopted to smoothen the VCDM using Gaussian smoothing operator. The adherence between foreground and background objects, due to the unconsciously smoothening, is avoided by restricting the applied area within a single object-layer at time. The process DMP outputs a refined disparity map (RDM), which is then transformed to an actual depth map through the process DDT using Eq. 2.4.2.

6.3 Experimental Results

In this section, we present results for each (sub)-process in the depth recovery development system. The photographs were taken with a still digital camera sat on a leveled tripod with a 20mm lens (equivalent to 40mm lens for normal 35mm camera). Captured images of 1536×1146 pixel resolution were processed to correct for lens distortion, then filtered down to 800×600 pixels for use in the development system. The examples shown below are cropped into 300×300 so they can be fitted into the paper. The results are all un-retouched except required, i.e. the disparity map refinement.

With two standard binocular stereo images, shown in Fig. 6.5, a pair of OLMs is produced through the process OLS, depicted in Fig. 6.4. The input images are of 24 bit-depth colors. Fifteen object layers are specified in each OLM. The left OLM originally is 296KB in BMP format, after compressed we have 6KB in GIF without any data loss (checked by the program).

Two stereo images and the generated OLMs are then passed to the next process DCA. In the process DCA, the process OE updates the CDM's entry if the entry is identified as an occlusion pixel. Figure 6.6 visualizes the output of the process OE, where red color regions are identified occlusions and green color components are the corresponding inter-objects' boundaries. The normalized correlation algorithm is implemented to search the corresponding boundary points, where an 11×11 shape-adaptive image window is adopted.

The process SPD generates a search-order output, which is visualized in Fig. 6.7. The brighter gray values indicate the higher priority to perform the corresponding search (through the process CST), the darker the less priority.

The output of the process DCA, a VCDM, is given to the right in Fig. 6.8 in comparison to the left where the process CST is directly applied without any pre-processes taken. Noticeable differences between the two are around the boundary areas of the sofa's arm and the corner of the tea table, where the inter-occlusion areas occur. This is also a strong evidence telling why we employ those preprocessing steps, that is too hard to rectify those erroneous artifacts once they have been produced.

The VCDM generated using our approach gives an easy and effortless way to the further refinements. Two sub-processes in the process DMR, the disparity correction and the disparity smoothing, fine-tune the VCDM. Their results are shown in Fig. 6.9. One remark is the smoothing operation (the Gaussian smoothing operator) is performed object-layer based so inter-objects' boundaries are crisp.

The refined disparity map (RDM) can further be visualized in shaded (Lambertian) and texture-mapped

[2]The G^0 continuity denotes zeroth order geometric continuity, which means only end-points match.

representations, shown in Fig. 6.10.

Additionally, a portion of a panorama image (by 7 mosaics) and the associated disparity map are depicted in Fig. 6.11. The disparity map is generated through three steps. First, a planar disparity map of each mosaic is yielded using our development system. Second, each planar disparity map is converted into a partial cylindrical disparity map (cf. Chapter 2.5). Third, all the partial cylindrical disparity maps are stitched into a single cylindrical disparity map, where the blending function of the stitching is disable. We also render nine views, depicted in Fig. 6.12, from the scene provided by the partial panorama image in Fig. 6.11 using the rendering engine constructed in our view synthesis system.

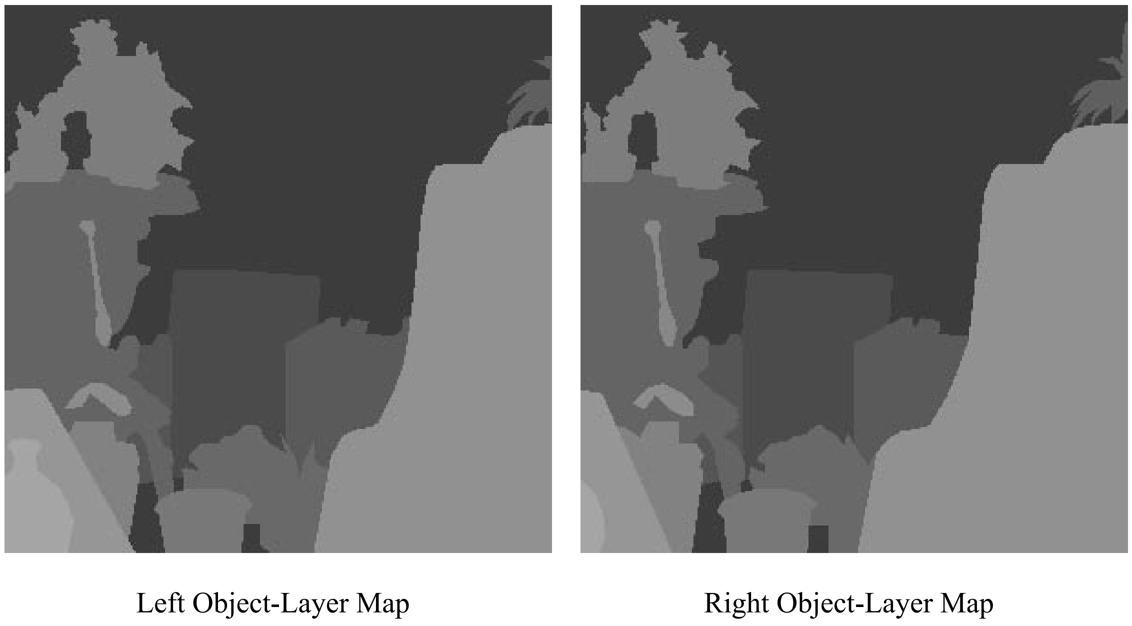


Figure 6.4: A pair of object-layer maps derived from stereo images in Fig. 6.5.

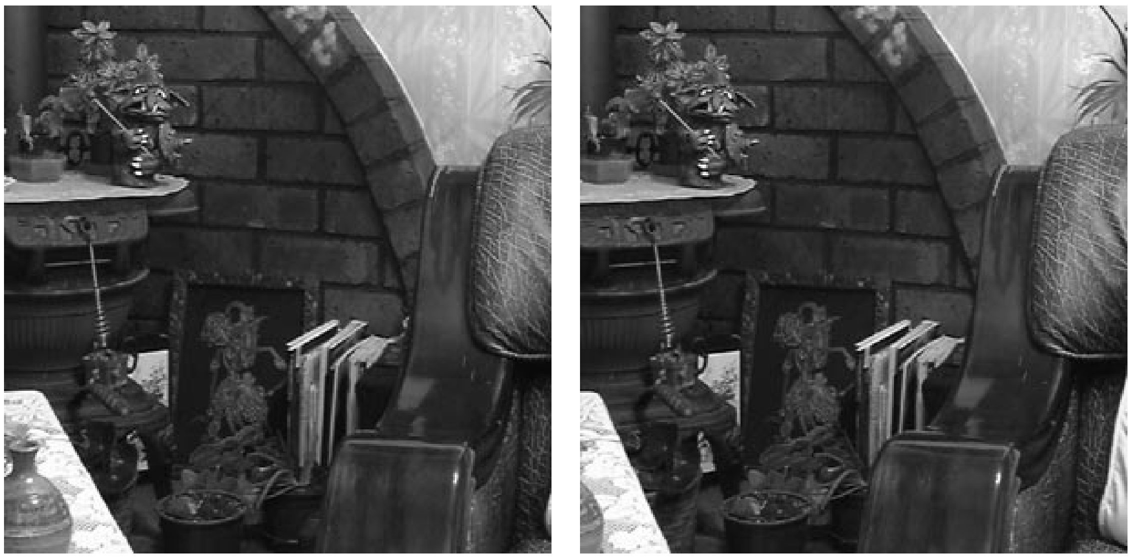


Figure 6.5: Two standard binocular stereo views of an indoor scene.

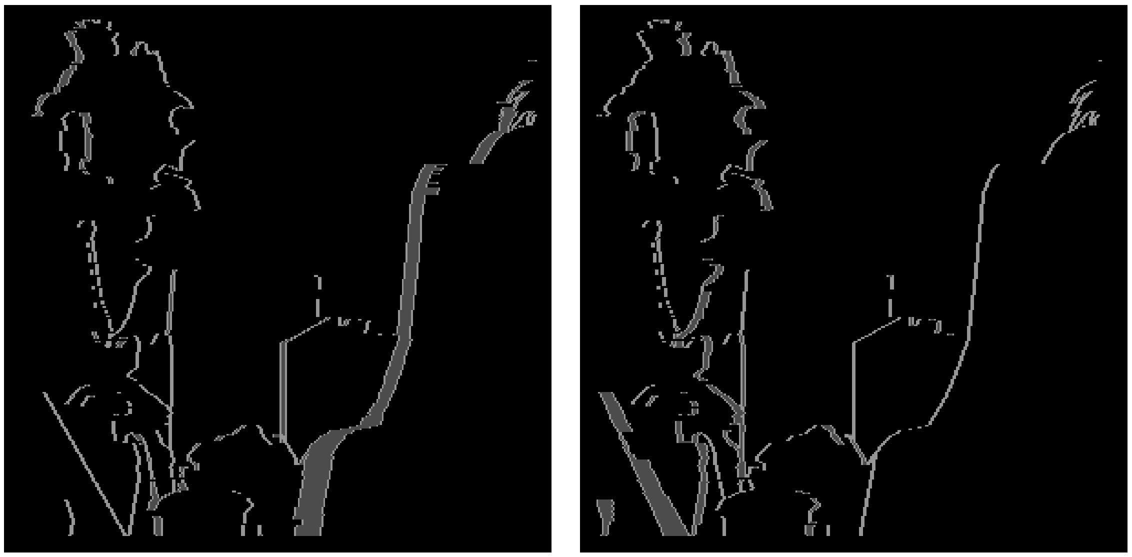


Figure 6.6: Results of occlusion extraction. Red color regions are identified occlusions and green color components are the corresponding inter-objects' boundaries.

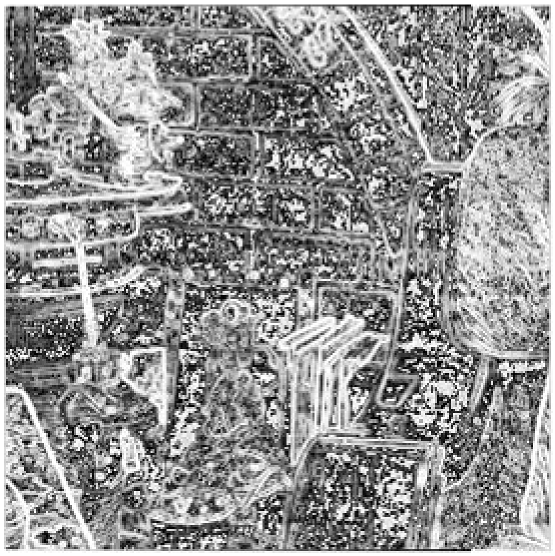


Figure 6.7: Visualization of the search-priority determination result, where the brighter gray values indicate the higher priority.

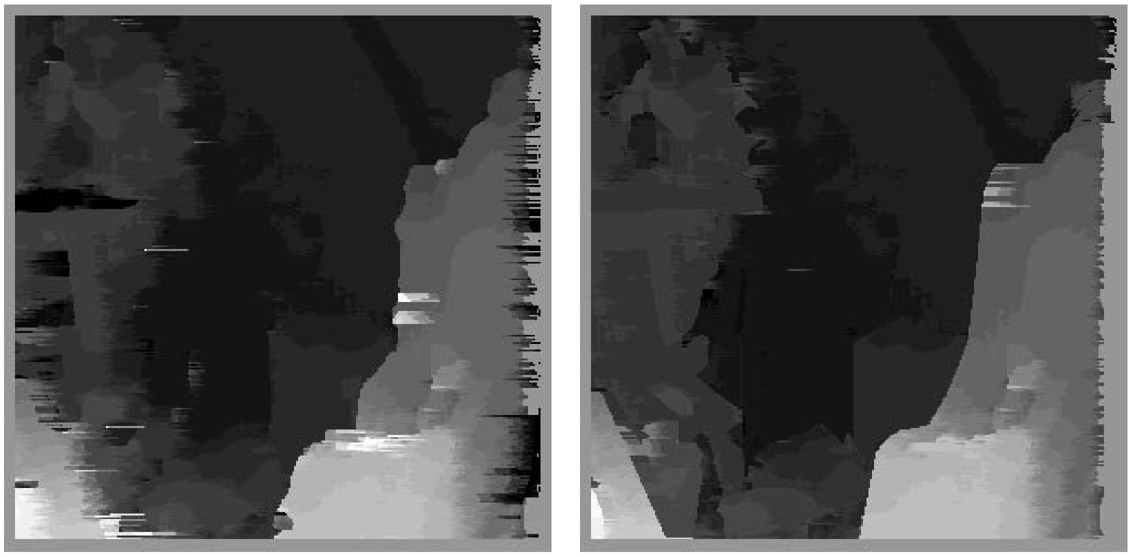


Figure 6.8: Comparison between two disparity maps, one with the preprocessing in our development system (right) and the other without (left).

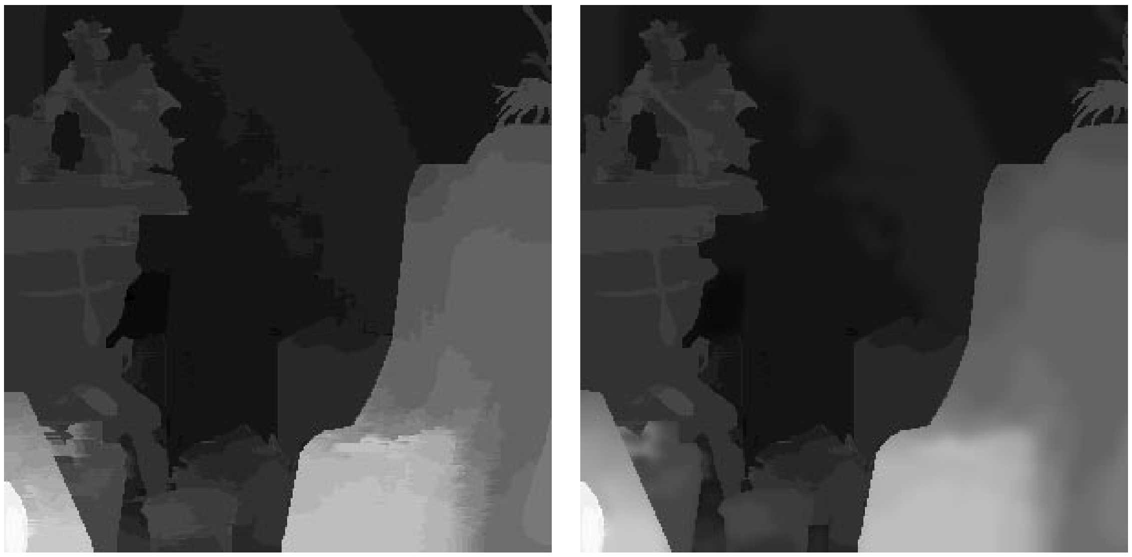


Figure 6.9: The corrected (left) and the smoothened (right) disparity maps.

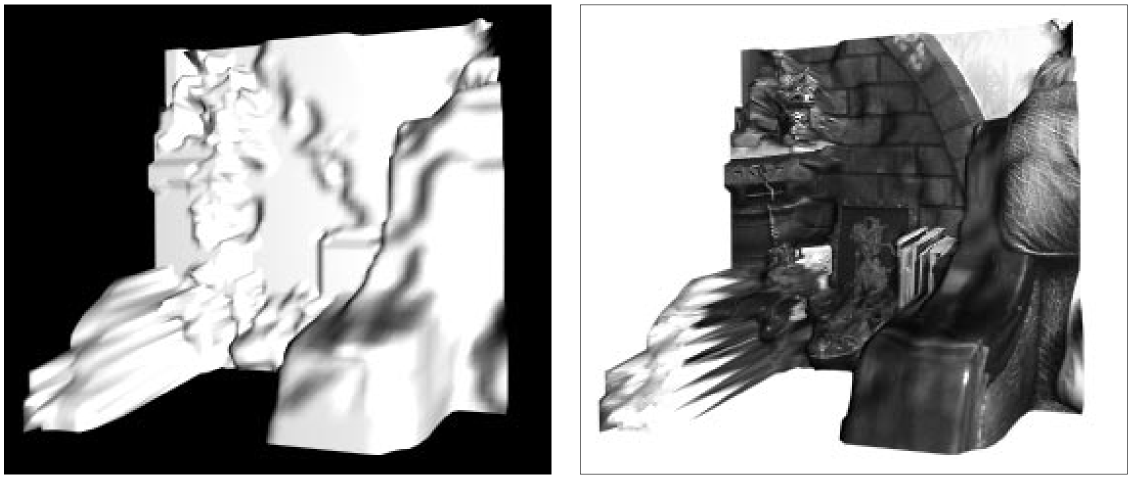


Figure 6.10: Shaded (left) and texture-mapped (right) representation of the refined disparity map.

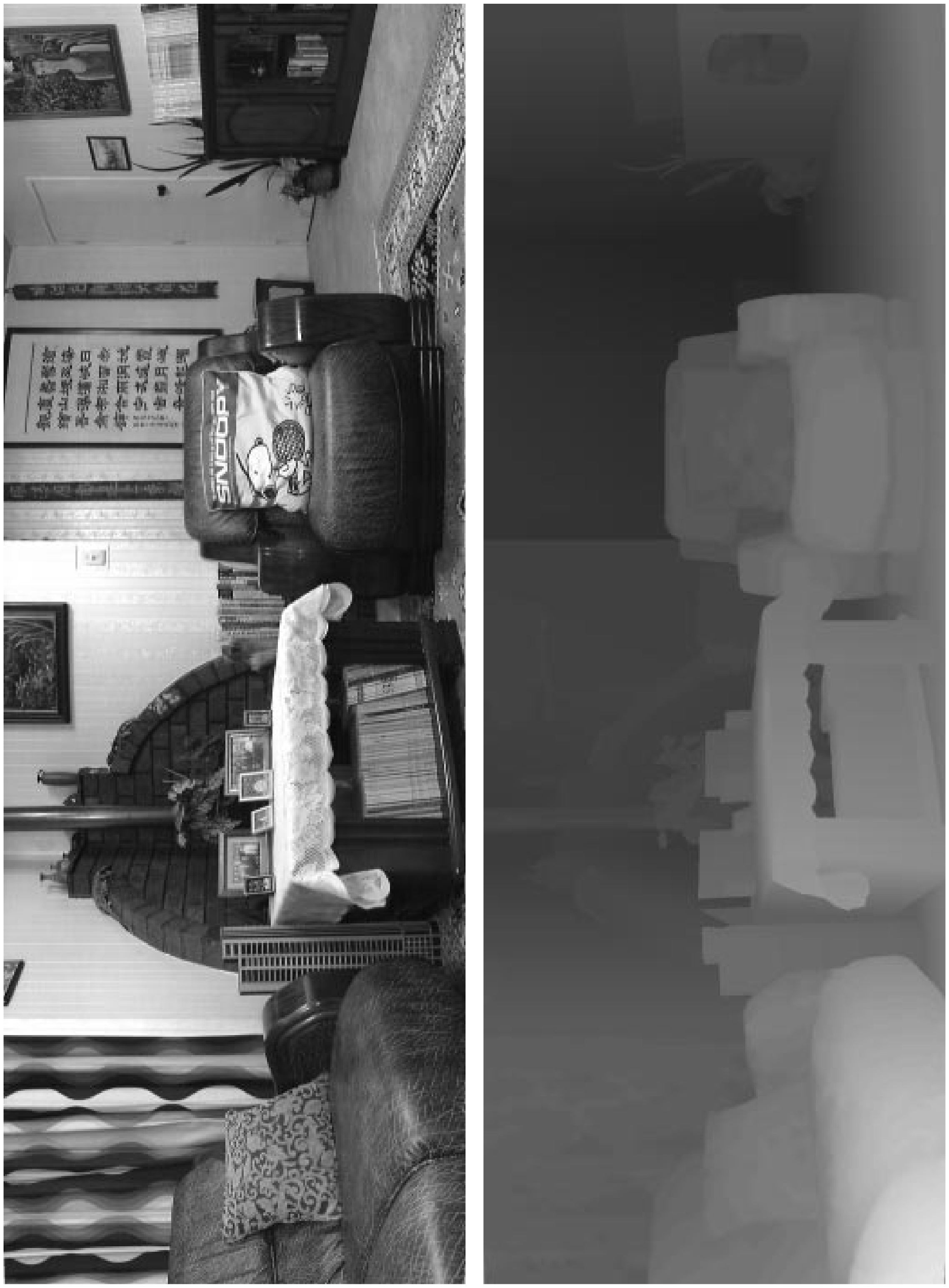


Figure 6.11: Partial panorama image with the associated cylindrical disparity map.

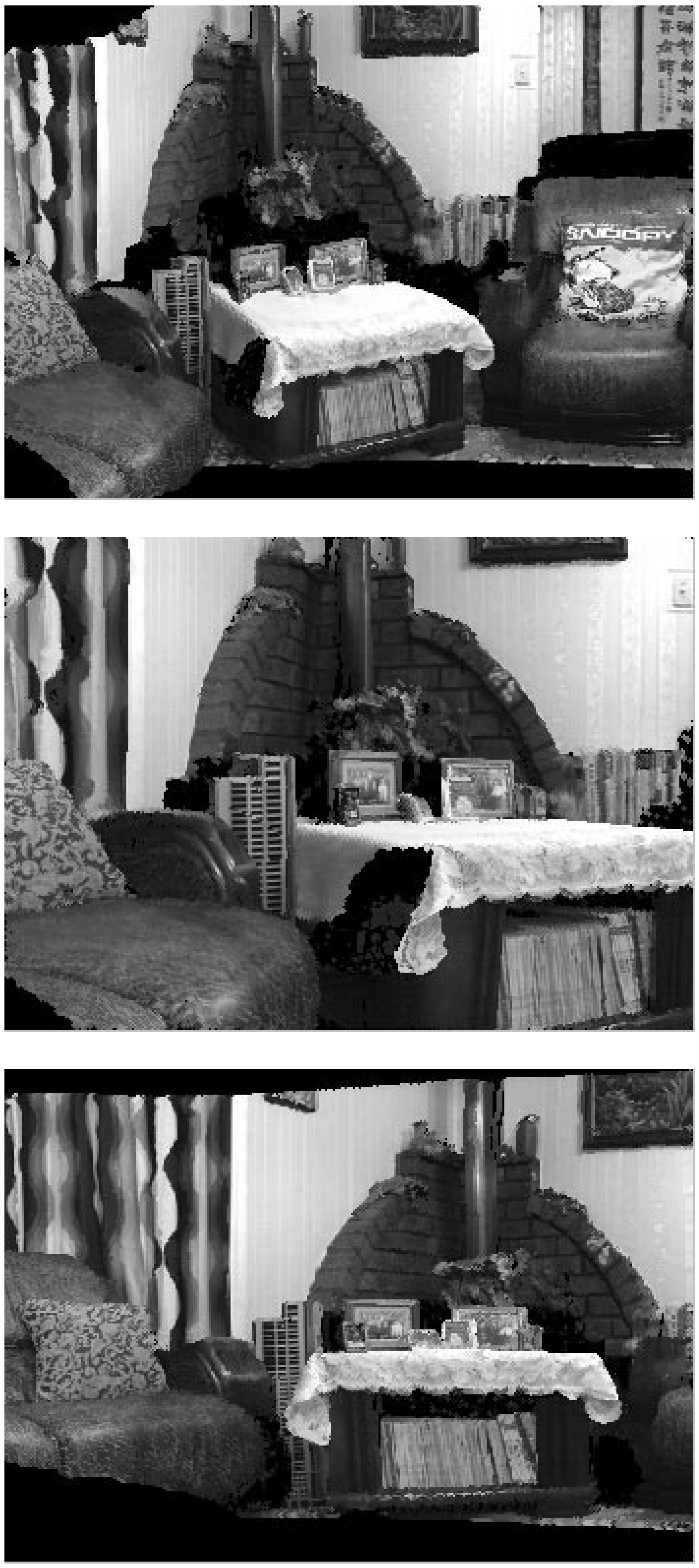


Figure 6.12: Nine views rendered from the scene provided by the partial panorama image in Fig. 6.11 using the rendering engine of our view synthesis system.

SNOOPY

SNOOPY

Chapter 7

Conclusions

The thesis presents the view synthesis system for 3-D scene visualization using a set of uncalibrated real images. It is focused on 3-D scene understanding, geometric problem analysis and practical solutions (with performance analysis) to the problems encountered in building such a system.

The system has been designed as a pipeline structure. Each process/subsystem, with clearly defined interfaces in between, is replaceable if new technology exists. A simple but expressive 3-D scene model, the depth-layered cylinder representation, has been designed and utilized as a bridge between the reconstruction and the rendering phases in the system. It allows a full-view, in terms of 360° horizontal perspective representation of a scene, and expresses the scene inherently. Consequently, the regular shape (cylinder) enables the fast rendering and the dynamic layer creation allows the integration of multiple images. So a real-time display system for 3-D navigation under moderate computing power (e.g. PC or workstation) become realizable.

Although we claim that the more the source images, the better the quality of generated image, the compactness of our scene representation is ensured by the proportional thresholding scheme. The limitation of our scene representation is its locality. It will be undesirable and less economical, in terms of data access and storage, if it attempts to describe a scene globally. Finding an optimal topology of multiple DLCs and exploring the coherence of compression for the multiple DLCs should be further pursued in the future.

The apparent quality of a synthesized view is closely related to the accuracy of the depth information available. To obtain a dense and accurate depth information from multiple real images is known a very difficult problem. Although many approaches have shown their potentials for automatic solutions, in practice they are still difficult to generate the adequate result. A depth recovery development system in the reconstruction phase has been constructed to retrieve depths based on the correspondence analysis of stereo images with human interventions. It separates tasks into a pipeline structure, and assigns the tasks to a human animator or a computer algorithm in a proper way that a satisfactory result can be attained reliably. The future works should further eliminate the extent of human intervention as well as provide a high-level editing tools to ease the intervention.

For each new view rendering, the culling process is first taken place. We have shown at least 75% of cylinder (the DLC), under the horizontal field of view of a virtual camera 45°, can be culled away before a traverse of the image mapping. The computations of the culling process under our scene representation are simpler and faster than the culling computations held in traditional computer graphics (e.g. back-face culling). It plays a significant role to improve the rendering performance under our scene representation.

After the culling, the partial cylinder portion is simply traversed in such a specific order that the visible surfaces with respect to the desired viewpoint are determined correctly. Our virtual box solution for the visibility determination directly inherits all advantages of L. McMillan's visibility algorithm yet a minor payoff of extra linkages to the actual scene representation. No extra depth comparison or sorting is required. The time complexity of it is constant, O(1), it is therefore suitable for the real-time implementation.

To found the basis of image transformations for various applications in our system, we have revealed all possible image mappings between two logical image forms (planar and cylindrical) by the establishment of their mapping equations. Based on regularity of geometric shape of our scene representation (cylinder) and uniformity of the traverse order (scan-line fashion), the computations of the image mapping optimization only require one forth of the original computations to achieve the same mapping result. Furthermore, this optimization is suitable for both on-line and off-line rendering processes.

The photorealism in our system is achieved with no additional operations, since the photometric properties of a scene are determined entirely by pre-acquired values of the input images. Although images are particularity appealing because their visual source helps to form our expectation, the images do not represent points that cannot be seen, as well as visible points are not represented equally.

It is known difficult, in general, to give a definition of measurement for the adequacy of number of images for the 3-D scene reconstruction, as well as an optimal scheme to determine where camera should give a capture. Consequently, the inherited weakness of image's nature, e.g. limited image resolutions and discontinuity of digital signals, may inherently constrain the computational accuracy to a certain degree. Moreover, several aspects of the image-based approaches, while theoretically correct, suffer from sensitivities to errant inputs and singularities under certain geometric configurations.

Appendix A

Fundamental Matrix

Given any pair of reference images, if their associated cameras' perspective projection matrices are known, we may derive a robust relationship between any pair of corresponding points as stated in Theorem A.0.1 below, it is usually referred to as *epipolar constraint.*

Theorem A.0.1. *For any two images, refer to them as left and right images, there exists a* 3×3 *matrix* $\boldsymbol{F}$ *such that the equation,*

$$\boldsymbol{p}_L^T \boldsymbol{F} \boldsymbol{p}_R = 0,$$

holds for any pair of corresponding points $\mathbf{p}_L$ *and* $\boldsymbol{p}_R$ *in left and right images respectively.*

The matrix $\mathbf{F}$ which describes a relationship between a pair of corresponding points is called *fundamental matrix* [31]. Let Π_L and Π_R be the cameras' perspective projection matrices for left and right images respectively. Let Π_L (Π_R) be decomposed as the concatenation of a 3×3 sub-matrix $\mathbf{H}_L$ ($\mathbf{H}_R$) and a 3-vector $\mathbf{h}_L(\mathbf{h}_R)$, i.e. $\Pi_L = \left[\mathbf{H}_L \middle| \mathbf{h}_L\right]$ and $\Pi_R = \left[\mathbf{H}_R \middle| \mathbf{h}_R\right]$. Then the fundamental matrix $\mathbf{F}$ has the form

$$\mathbf{F} = \left[\mathbf{h}_L - \mathbf{H}_L \mathbf{H}_R^{-1} \mathbf{h}_R\right]_\times \mathbf{H}_L^{-1} \mathbf{H}_R^{-1}. \tag{A.0.1}$$

The derivation of fundamental matrix has been described extensively in [18]. In general, the rank of $\mathbf{F}$ is equal to two; thus it defines a one-to-one mapping from a set of image points to a set of image lines. This sort of mapping is called *correlation.* The fundamental matrix, if known, is very useful to assist the corresponding point searching between two reference images. For example, given an image point say $\mathbf{p}_L$ in the left image, the epipolar constraint guaranties that its corresponding point $\mathbf{p}_R$ in the right image must lie on its epipolar line. Hence the 2-D searching space over the image plane are reduced down to 1-D searching, i.e. along the mapped epipolar line.

Appendix B

Fundamental Matrix Recovery

The epipolar geometry can be discovered if we have their associated projection matrices, which combine all the camera intrinsic and extrinsic parameters. In other words, we must know the geometric relationship between every pair of images and their corresponding camera configurations. However, usually the complete geometric information between any pair of reference images and their camera parameters are unknown. Thus we cannot use Eq. A.0.1 directly to compute the fundamental matrix. Instead, a partial geometric relationship may be established by involving some human interventions[1] in which a few matching points are identified. We may therefore apply the parameter estimating algorithms to find an optimal solution for a matrix $\mathbf{F}$ numerically.

Assuming $\mathbf{p}_L = (x_{p_L}, y_{p_L})$ and $\mathbf{p}_R = (x_{p_R}, y_{p_R})$ are one pair of matched points in two reference images respectively. From Theorem A.0.1, we know that

$$\mathbf{p}_L^T \mathbf{F} \mathbf{p}_R = 0,$$

and it can be expanded as follows,

$$(x_{p_L}, y_{p_L}, 1) \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{pmatrix} x_{p_R} \\ y_{p_R} \\ 1 \end{pmatrix} = 0,$$

where f_{ij} are the elements of $\mathbf{F}$. This equation can be rewritten as a linear and homogenous function G with four variables and nine unknown coefficients, i.e.

$$\begin{aligned} & G(x_{p_L}, y_{p_L}, x_{p_R}, y_{p_R}) \\ = \; & f_{11}x_{p_L}x_{p_R} + f_{12}x_{p_L}y_{p_R} + f_{13}x_{p_L} + f_{21}y_{p_L}x_{p_R} + f_{22}y_{p_L}y_{p_R} + f_{23}y_{p_L} + f_{31}x_{p_R} + f_{32}y_{p_R} + f_{33} \\ = \; & 0. \end{aligned} \tag{B.0.1}$$

By given a set of m pairs of corresponding points $\{(x_{p_{iL}}, y_{p_{iL}}, x_{p_{iR}}, y_{p_{iR}})\}$ where $i = 1, \ldots, m$, the task becomes finding those nine coefficients that best fit to Eq. B.0.1. In shorter form, let $\mathbf{w} = (f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33})^T$, and $\mathbf{d}_i = (x_{p_{iL}}x_{p_{iL}}, x_{p_{iL}}y_{p_{iR}}, x_{p_{iL}}, y_{p_{iL}}x_{p_{iR}}, y_{p_{iL}}y_{p_{iR}}, y_{p_{iL}}, x_{p_{iR}}, y_{p_{iR}}, 1)^T$, so that $G(x_{p_{iL}}, y_{p_{iL}}, x_{p_{iR}}, y_{p_{iR}}) = \mathbf{d}_i^T\mathbf{w}$.

The fundamental matrix $\mathbf{F}$ is defined up to a scalar factor, so we may set one of nine elements of $\mathbf{F}$ to be 1. It is then only left eight parameters in its place need to be estimated. Furthermore, the rank of $\mathbf{F}$ is at most 2, the number of free parameter of $\mathbf{F}$ can then be further reduced down to seven [18]. Ideally, to determine the fundamental matrix $\mathbf{F}$, we need at least seven pair of matched points between two reference

[1]Some automatic approaches using point detector, e.g. Canny detector or Plessey-Harris detector, have attempted to build up the correspondences among them. However, we find that they are unstable so we specify the corresponding points manually.

images. In practice, due to the factors of false matching and discontinuity of discrete data arisen from the image digitization, an optimization scheme is required to minimize the potential errors it may occur. One of approaches is to strengthen and constrain the estimation by giving more matching points, i.e. more than seven. There are many of numerical approximation methods available to estimate **F** with over-specified data, we demonstrate a simple approach — linear least-squares.

For more than seven pair of matching points (i.e. $m > 7$), the problem of finding an optimal solution is equivalent to minimize the following function:

$$U(\mathbf{w}) = \sum_{i=1}^{n} G^2(x_{p_{iL}}, y_{p_{iL}}, x_{p_{iR}}, y_{p_{iR}}).$$

Clearly, there exists a trivial solution $f_{ij} = 0$ for all $i, j = 1..3$, which is not what we want. We should impose some constraints to the coefficients of $G(x_{p_L}, y_{p_L}, x_{p_R}, y_{p_R})$ in order to avoid it. One way is to set one of the coefficients to 1. Without loss of generality, we assume that f_{33} is not equal to zero, and hence we can set f_{33} = -1. Let $\mathbf{w}' = (f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32})^T$, and $\mathbf{d}'_i = (x_{p_{iL}}x_{p_{iL}}, x_{p_{iL}}y_{p_{iR}}, x_{p_{iL}}, y_{p_{iL}}x_{p_{iR}}, y_{p_{iL}}y_{p_{iR}}, y_{p_{iL}}, x_{p_{iR}}, y_{p_{iR}})^T$. Moreover, $\mathbf{D} = [\mathbf{d}'_1, \mathbf{d}'_2, \ldots, \mathbf{d}'_k]^T$, and $\mathbf{v} = (v, v, \ldots, v)^T$, where $\mathbf{v}$ is a k-vector and its elements v is the last element of $\mathbf{d}_i$. By given m point, the system equation can be rewritten as

$$G(x_{p_{iL}}, y_{p_{iL}}, x_{p_{iR}}, y_{p_{iR}}) = \mathbf{d}'^T_i \mathbf{w}' - v = 0.$$

Thus the function to be minimized becomes

$$U(\mathbf{w}) = (\mathbf{D}\mathbf{w}' - \mathbf{v})^T(\mathbf{D}\mathbf{w}' - \mathbf{v}).$$

The solution can be obtained by setting its first derivative to be zero and yield

$$\frac{\partial U(\mathbf{w})}{\partial \mathbf{w}'} = 2\mathbf{D}^T(\mathbf{D}\mathbf{w}' - \mathbf{v}) = 2\mathbf{D}^T\mathbf{D}\mathbf{w}' - \mathbf{D}^T\mathbf{v} = 0.$$

Hence, function $U(\mathbf{w})$ is minimum when

$$\mathbf{w}' = \left(\mathbf{D}^T\mathbf{D}\right)^{-1}\mathbf{D}^T\mathbf{v},$$

$$\mathbf{w} = \begin{pmatrix} \mathbf{w}' \\ -1 \end{pmatrix},$$

if there exists an unique global minimum. This method will fail if the element we set to 1 is actually zero or much smaller than the other elements. As we do not know priori which element is not zero, we can then set each element of **F** to -1 for nine iterations and choose the one with minimum value of $U(\mathbf{w})$.

Appendix C

Relative Camera Projection Matrices Recovery

Camera projection matrix specifies a transformation from 3-D to 2-D, and it is used to map a space-point to an image-point. A camera projection matrix consists of camera intrinsic and extrinsic parameters. It is well known that if we have the camera parameters and the depths of any two stereo images, we are able to reconstruct the 3-D scene from those 2-D information. Unfortunately, the camera parameters are usually not available unless additional measurements are taken, such as camera calibration.

Our system assumes all the input images are uncalibrated, which means no camera parameters are known priori. It is known impossible to retrieve actual camera projection matrices only from those uncalibrated stereo images. However, the relative camera projection matrices of stereo images are possible to be recovered and are sufficient for our application of the scene registration (cf. Chapter 4).

Given two uncalibrated reference images, the task is to recover the relative perspective camera projection matrices associated with those two images. Z. Zhang [18] has pointed out a nice property that, if the epipolar geometry of two uncalibrated images is known, i.e. the fundamental matrix, then the relative camera projection matrices can be determined [37]. However, it is only up to a linear transformation depending on the projection model of the camera. For example, with the perspective projection the recovered cameras projection matrices are defined up to a *projective transformation* in 3-D space while with the orthographic, weak projective and paraperspective projections, their recovered camera projection matrices are defined up to an *affine transformation.* A projective transformation is represented by a non-singular 4×4 matrix acting on homogenous vectors. When we said "only up to a projective transformation", it means if Π_1 and Π_2 are two perspective camera projection matrices and their epipolar geometry, the fundamental matrix $\mathbf{F}$, is established, then for any arbitrary projective transformation $\mathbf{T}$ in 3-D space, $\Pi_1' = \Pi_1\mathbf{T}$ and $\Pi_2' = \Pi_2\mathbf{T}$ remain constantly consistent with $\mathbf{F}$.

The property described above has suggested that there are infinite numbers of projective base satisfying the epipolar constraint, hence there is no way to recover the absolute camera projection matrices associated with those images. Nevertheless, any pair of recovered camera projection matrices satisfying the epipolar constraint has fulfilled the partial requirements of the images mapping between two uncalibrated images (cf. Chapter 4). One way to represent the relative camera projection matrices recovered from the fundamental matrix $\mathbf{F}$ is to use the *canonical representation* described in [38, 39, 37]. It can be expressed as follows,

$$\Pi_1 = \left[\mathbf{H}_1 \middle| \mathbf{h}_1\right] \quad \text{and} \quad \Pi_2 = [\mathbf{I} \mid \mathbf{0}]. \tag{C.0.1}$$

Here Π_1 and Π_2 are defined with respect to the world coordinate system that is assumed to be coincided with the second camera coordinate system. This representation is equivalent to the strongly calibrated case in which the camera intrinsic parameters are known. In this case, $\mathbf{F}$ is called the *essential matrix.* A matrix $\mathbf{H}_1$ describes the orientation of the first camera with respect to the second camera coordinate system, i.e. the world coordinate system. A vector $\mathbf{h}_1$ is equal to -$\mathbf{H}_1\mathbf{C}_1$ (cf. Chapter 2, where $\mathbf{C}_1$ is the position of the first camera optical center with respect to the second camera coordinate system).

From Eq. A.0.1 the fundamental matrix $\mathbf{F}$ can be calculated as follows,

$$\mathbf{F} = \left[\mathbf{h}_1 - \mathbf{H}_1\mathbf{H}_2^{-1}\mathbf{h}_2\right]_\times \mathbf{H}_1\mathbf{H}_2^{-1}.$$

Let us considering Eq. C.0.1 above, we have $\mathbf{H}_2 = \mathbf{I}$ and $\mathbf{h}_2 = \mathbf{0}$. Substitute to Eq. A.0.1, we have

$$\mathbf{F} = [\mathbf{h}_1]_\times \mathbf{H}_1,$$

where $[\mathbf{h}_1]_\times$[1] is a skew symmetric matrix of $\mathbf{h}_1$, and $\mathbf{H}_1$is a 3×3 rotation matrix.

Since the epipole $\mathbf{e}_1$ in the image plane I_1 is the projection of the second camera optical center onto the image plane I_1, so we can write

$$\mathbf{e}_1 \simeq \Pi_1\mathbf{C}_2 = \left[\mathbf{H}_1\middle|\mathbf{h}_1\right]\mathbf{C}_2 = \left[\mathbf{H}_1\middle| -\mathbf{H}_1\mathbf{C}_1\right]\mathbf{C}_2 = -\mathbf{H}_1\mathbf{C}_1,$$

where $\simeq$ means "equal" up to a scale factor and we have $\mathbf{C}_2 = (0,0,0)$ as the origin of the world coordinate system. And we know $\mathbf{h}_1$ is equal to -$\mathbf{H}_1\mathbf{C}_1$, so $\mathbf{e}_1 = \mathbf{h}_1$. The epipole $\mathbf{e}_1$ in the image plane I_1 has the property $\mathbf{e}_1^T\mathbf{F} = 0$, which is equivalent to $\mathbf{F}^T\mathbf{e}_1 = 0$. Since $\mathbf{F}$ is singular, we multiply both sides by $\mathbf{F}$ and then we have $\mathbf{F}\mathbf{F}^T\mathbf{e}_1 = 0$. The matrix $\mathbf{F}\mathbf{F}^T$ is symmetric, hence the epipole $\mathbf{e}_1$ is the eigenvector of matrix $\mathbf{F}\mathbf{F}^T$ associated to the smallest eigenvalue.

To calculate Π_1 and Π_2 from the matrix $\mathbf{F}$, we first factorize $\mathbf{F}$ into a product of this form $[\mathbf{e}_1]_\times\mathbf{H}_1$. The factorization is not unique in general, since once we find a matrix $\mathbf{H}_1$ satisfying it, any matrix in this form, $\mathbf{H}_1 + \mathbf{e}_1\mathbf{v}^T$, will also be a solution for any 3-vector $\mathbf{v}$. It can be easily verified that we always have $[\mathbf{e}_1]_\times\mathbf{e}_1\mathbf{v}^T = 0$. In particular, a matrix $\mathbf{H}_1$ can be obtained by the following steps:

$$\begin{array}{rrcll}
 & \mathbf{F} & = & [\mathbf{e}_1]_\times\mathbf{H}_1, & \\
\Rightarrow & [\mathbf{e}_1]_\times\mathbf{F} & = & [\mathbf{e}_1]_\times^{\,2}\mathbf{H}_1, & \\
\Rightarrow & [\mathbf{e}_1]_\times\mathbf{F} & = & (\mathbf{e}_1\mathbf{e}_1^T - \| \mathbf{e}_1 \|^2\mathbf{I}_3)\mathbf{H}_1, & \text{since } \mathbf{v}\mathbf{v}^T = [\mathbf{v}]_\times^{\,2} + \| \mathbf{v} \|^2\mathbf{I}_3 \text{ for any 3-vector } \mathbf{v}, \\
\Rightarrow & [\mathbf{e}_1]_\times\mathbf{F} & = & \mathbf{e}_1\mathbf{e}_1^T\mathbf{H}_1 - \| \mathbf{e}_1 \|^2\mathbf{H}_1, & \\
\Rightarrow & [\mathbf{e}_1]_\times\mathbf{F} & = & -\| \mathbf{e}_1 \|^2\mathbf{H}_1, & \text{since } [\mathbf{e}_1]_\times\tilde{\mathbf{e}}_1\mathbf{v}^T = 0 \text{ for any 3-vector } \mathbf{v}, \\
\Rightarrow & \mathbf{H}_1 & = & (-1/\| \mathbf{e}_1 \|^2)[\mathbf{e}_1]_\times\mathbf{F}. &
\end{array}$$

Therefore,

$$\Pi_1 = \left[(-1/\| \mathbf{e}_1 \|^2)[\mathbf{e}_1]_\times\mathbf{F} \mid \mathbf{e}_1\right] \quad \text{and} \quad \Pi_2 = [\mathbf{I} \mid \mathbf{0}],$$

the relative perspective camera projection matrices are recovered using the fundamental matrix $\mathbf{F}$ derived from two uncalibrated reference images.

[1]The notion $[\mathbf{v}]_\times$ denotes the *skew symmetric matrix* of vector $\mathbf{v}$, which is

$$[\mathbf{v}]_\times = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}_\times = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix}.$$

Bibliography

[1] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In *Proc. SIGGRAPH'95*, pages 39–46, 1995.

[2] S. E. Chen. QuickTimeVR - an image-based approach to virtual environment navigation. In *Proc. SIGGRAPH'95*, pages 29–38, 1995.

[3] S. Laveau and O. Faugeras. 3-d scene representation as a collection of images and fundamental matrices. Technical Report 2205, INRIA, Lucioles, France, February 1994.

[4] T. Beier and S. Neely. Feature-based image metamorphosis. In *Proc. SIGGRAPH'92*, pages 35–42, New York, USA, July 1992.

[5] T. Werner, R. Hersch, and V. Hlavac. Rendering real-world objects using view interpolation. In *Proceedings of the International Conference on Computer Vision*, pages 957–962, 1995.

[6] S. M.Seitz and C. R. Dyer. View morphing. In *Proc. SIGGRAPH'96*, pages 21–30, New Orieans, Louisiana, USA, August 1996.

[7] M. Levoy and P. Hanrahan. Light field rendering. In *Proc. SIGGRAPH'96*, pages 31–42, New Orleans, Louisiana, August 1996.

[8] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *Proc. SIGGRAPH'96*, pages 43–54, New Orleans, Louisiana, August 1996.

[9] R. Szeliski and S. Kang. Direct methods for visual scene reconstruction. In *IEEE Workshop on Representation of Visual Scenes*, 1995.

[10] S. Peleg and J. Herman. Panoramic mosaic by manifold projection. In *CVPR*, pages 338–343, 1997.

[11] D. Beymer, A. Shashua, and T. Poggio. Example based image analysis and synthesis. Technical Report 1431, Artificial Intelligence Laboratory, MIT, USA, 1993.

[12] T. Poggio and R. Brunelli. A novel approach to graphics. Technical Report 1354, Artificial Intelligence Laboratory, MIT, USA, 1992.

[13] S. E. Chen and L. Williams. View interpolation for image synthesis. In *Proc. SIGGRAPH'93*, pages 278–288, Anaheim, California, USA, August 1993.

[14] E. H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. In *Computational Models of Visual Proceeding*, pages 3–20, Cambridge, MA, 1991.

[15] S. K. Wei, Y. F. Huang, and R. Klette. Color anaglyphs for panorama visualizations. Technical Report 19, CITR, Auckland University, New Zealand, Feb. 1998.

[16] R. Klette, K. Schlüns, and A. Koschan. *Computer Vision - Three-Dimensional Data from Images.* Springer, Singapore, 1998.

[17] W. R. Mark, L. McMillan, and G. Bishop. Post-rendering 3d warping. In *Proceedings of 1997 Symposium on Interactive 3D Graphics*, pages 7–16, Providence, RI, April 1997.

[18] G. Xu and Z. Zhang. *Epipolar Geometry in Stereo, Motion and Object Recognition.* Kluwer, Netherlands, 1996.

[19] B. Jähne. *Digital Image Processing: Concepts, Algorithms, and Scientific Applications. Second edition.* Springer, Berlin, 1993.

[20] K. Kanatani. *Geometric Computation for Machine Vision. Oxford Engineering Science Series No.37.* Oxford University Press, New York, 1995.

[21] S. M. Seitz. *Image-Based Transformation of Viewpoint and Scene Appearance.* PhD thesis, University of Wisconsin, 1997.

[22] C.-Y. Lin, S.-W. Shih, and Y.-P. Hung. Toward automatic reconstruction of 3d environment with an active binocular head. Technical Report 004, Academia Sinica, Taipei, Taiwan, 1998.

[23] S. Avidan and A. Shashua. Novel view synthesis in tensor space. In *CVPR97*, pages 1034–1040, June 1997.

[24] R. Szeliski and H.-Y. Shum. Creating full view panoramic image mosaics and environment maps. In *Proc. SIGGRAPH'97*, pages 251–258, Los Angeles, California, USA, August 1997.

[25] S. Peleg and J. Herman. Panoramic mosaic by manifold projection. Technical report, David Sarnoff Research Center, Princeton, NJ, USA, 1996.

[26] B. Noble and J. W. Daniel. *Applied Linear Algebra Third Edition.* Prentice-Hall, London, 1988.

[27] J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics Principles and Practice Second Edition.* Addison-Wesley, Reading, Massachusetts, 1990.

[28] L. McMillan. A list-priority rendering algorithm for redisplaying projected surfaces. Technical Report 95-005, University of North Carolina, Chapel Hill, USA, 1995.

[29] R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision*, volume II. Addison-Wesley, Reading, Massachusetts, 1993.

[30] A. Rosenfeld and A. C. Kak. *Digital Picture Processing*, volume 2. Academic Press, London, England, second edition, 1993.

[31] O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint.* The MIT Press, London, England, 1993.

[32] Z. Zhang, R. Deriche, O. Faugeras, and Q.-T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. Technical Report 2273, INRIA, Lucioles, France, May 1994.

[33] H.-C. Huang and Y.-P. Hung. Disparity morphing and automatic generation of stereo panoramas for photo-realistic virtual reality systems. Technical Report 002, Academia Sinica, Taipei, Taiwan, 1997.

[34] E. N. Mortensen and W. A. Barrett. Intelligent scissors for image composition. In *Proc. SIGGRAPH'95*, pages 191–198, 1995.

[35] R. T. Azuma. A survey of augmented reality. In *Teleoperators and Virtual Environments 6*, pages 355–385, Malibu, CA, August 1997.

[36] Y. F. Huang, S. K. Wei, and R. Klette. Automatic generation of stereo images from multiple monocular object views. Technical Report 28, CITR, Auckland University, New Zealand, Aug. 1998.

[37] Z. Zhang and G. Xu. A general expression of the fundamental matrix for both perspective and affine cameras. In *IJCAI'97*, pages 23–29, Nagoya, Japan, August 1997.

[38] P. Beardsley, A. Zisserman, and D. Murray. Navigation using affine structure from motion. In *Proc. 3rd ECCV*, volume 2, pages 85–96, Stockholm, Sweden, May 1994.

[39] Q.-T. Luong and T.Vieville. Canonic representations for the geometries of multiple projective views. In *Proc. 3rd ECCV*, volume 1, pages 589–599, Stockholm, Sweden, May 1994.