Panorama Interpolation for Image-based Navigation

by

Feng Shi

Thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

In partial fulfillment of the requirements

For the M.A.Sc. degree in

Electrical Engineering

School of Information Technology and Engineering

Faculty of Engineering

University of Ottawa

Abstract

This thesis presents methods for novel image synthesis from cubic panoramas taken with multi-sensor cameras. The pre-captured cubic panoramas are used to interpolate arbitrary views to allow a virtual walkthrough of the remote real environment. In our approach, the "transfer" and "triangulation" methods are adopted to analyse the geometry of cubic panoramas and recover accurate essential matrix between cubes. To generate novel view between two aligned cubes, a warping model is applied to warp cubes to approximate navigation. This technique, called *cube warping*, works by simplifying the model of pixel displacements between cubes. A new raytracing like image-based interpolation method is also proposed for free-viewpoint cube synthesis. Instead of attempting to recover dense reconstruction precisely, our method try to reconstruct colours with *colour invariance constraints*. Due to the fact that photo consistency has more to do with colour than shape, our algorithm can generate a complete novel scene view with maximized photo consistency.

Acknowledgements

I would like to thank my supervisor Dr. Robert Laganière for providing this opportunity to work with him. His supervision and trust in my abilities have proved to be extremely invaluable during the journey of the research and the production of this work. I would also like to thank my co-supervisor Dr. Eric Dubois for his guidance and constant encouragement.

Secondly, I would like to thank Dr. Frédéric Labrosse of *University of Wales, Aberystwyth* for his creative suggestions about *Cube Warping*. Special thanks to Florian Kangni and Colince Donfack for their helps to provide me with the capture and generation of cubic panoramas as well as suggestions about cube geometry.

Finally, I would also like to express gratitude to my wife, Dongmei, whose prompting encouraged me down this academic journey. It is by the selfless supports and sacrifices that she made over the years that I am able to fulfill this accomplishment.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The main theme of this thesis is to perform panorama view synthesis for image-based navigation. The work is part of the NAVIRE project [1] developed at University of Ottawa. The project objective is to develop image-based rendering(IBR) system to allow a person to virtually navigate through a remote real environment using pre-captured panorama images. To achieve smooth and natural navigation, a dense grid of pre-generated panoramic images of the environment and appropriate synthesized virtual views among grid nodes are required.

Ideally, the pre-captured panoramic images are dense enough such that there is no perceivable transition when switching from one cube into another. The early approaches to synthesize and navigate in virtual environments have used movies created by photography or computer rendering. In Lippman's approach[43], the streets were filmed at 10-foot intervals. To simulate navigation of the walk-through, two video disc players were used to retrieve corresponding views. Later, in [52], at every selected point, the virtual museum was simulated with a 360 degree panning movie which generated by computer rendering images. Navigation between two connected view points were simulated with

[1] http://www.site.uottawa.ca/research/viva/projects/ibr/

bi-directional transition movie.

Continuous video can provide seamless visualization of environment. However, there are some problems. On one side, since the movie or video is often arranged between two points with bi-direction, it has limited navigability and interaction. On the other side, taking continuous video pervasively may involve huge amount of data and put enormous burden on storage and network transmission. This is especially the case considering the high resolution(6x1024x1024) display of cube panoramic images. Therefore, it is impractical to use movie-based approach. The navigation should be achieved by both ensuring a dense coverage of the environment and through appropriate viewpoint interpolation [38].

1.1 Relevant Work

1.1.1 Geometry-based rendering

Traditional approaches for the navigation of virtual environments use 3D geometric primitives to render novel views. These methods are called geometry-based rendering. Geometry-based rendering utilizes strategies such as ray-tracing, soft shadows, global illumination, caustics, hierarchies and anti-aliasing to render scenes. These methods often take very long time even for simple scenes rendering. For complex renderings, geometry-based rendering adopts simplified techniques to accelerate processing either by reducing the geometric complexity of the scene [27][31][13][23][47] or by reducing the rendering complexity through texture mapping [8][68][53][24].

Under appropriate models, geometry-based rendering can produce convincing novel view with good solution for occlusion problems. However, it has following limitations:

- Acquisition of realistic surface models is a laborious and difficult task and it is very

difficult, if not impossible, to model very complex scenes.

- It takes days or even more than one week to complete rendering very complex environments, such as those images appearing at Internet Raytracing Competition[1]. Therefore, for real-time rendering, the rendering engine has to make some limitation on scene complexity and rending quality.

McMillan and Bishop said in their renowned paper [49]: *While geometry-based rendering technology has made significant strides toward achieving photorealism, creating accurate models is still nearly as difficult as it was ten years ago. Technological advances in three-dimensional scanning provide some promise in model building. However, they also verify our worst suspicions –the geometry of the real– world is exceedingly complex. Ironically, the primary subjective measure of image quality used by proponents of geometric rendering systems is the degree with which the resulting images are indistinguishable from photographs.*

1.1.2 Image-based rendering

Image-based rendering (IBR), as a powerful alternative to traditional geometry-based techniques for image synthesis, use images as primitives to render novel views. Compared with geometry-based techniques, the IBR can avoid the tedious, laborious 3D modeling stage and produce faster rendering for complex scene due to its independence of the scene complexity. The existing approaches for image synthesis with IBR method can be classified into two categories: (i) reconstruction-based methods; (ii) interpolation-based methods.

Reconstruction-based methods

For reconstruction-based methods, a 3D reconstruction, either explicit or implicit, is performed first. Then, the constructed scene is projected into new viewpoint followed with texture mapping. Although struggled with the difficult problems of estimating dense or quasi-dense correspondence, these methods can handle occlusion issues well and generate novel views with arbitrary viewpoints.

Laveau and Faugeras [40] employ the epipolar constraints to perform a raytracing like search of corresponding pixels in reference images for novel view pixels. In *Layered depth images (LDI)* [60], "multiple overlapping layers" are constructed by using stereo techniques. Both the depths of visible surface points and those of occluded scene points are stored in the input image. To rendering an arbitrary novel view, it is only need to warp a single image, in which each pixel consists of a list of depth and color values. Kang and Szeliski [35] use panoramas to recover 3D scene information. At different locations, the panoramas are produced with sequences of precaptured images. Then, 3D scene data are constructed by using stereo techniques. Finally, the rendering is performed with the models generated by the constructed 3D data.

Interpolation-based approaches

Interpolation-based methods try to generate photorealistic points of novel views directly by interpolating the corresponding pixels of the input images. Even with lifelike novel views and low computation costs, interpolation-based methods [11, 59, 42] often have limitations of producing the novel views with restrained viewpoints.

One typical interpolation-based technique, Seitz and Dyer's *view morphing* [59] approach employs a three-step algorithm to guarantee the intermediate view being geometrically correct. A *prewarp* stage is applied to rectify two reference images first. Then the

intermediate *morphing* stage is performed to interpolate the coordinates and colours of corresponding pixels. At last, a *postwarp* stage is followed to neutralize the prewarping effects. In Chen and williams' *view interpolation* [11], the flow fields are established first, followed with the morphing techniques to interpolate pixels from two input images. The *plenoptic modeling* [49], *light field* [41], *lumigraph* [25], and *concentric mosaics* [62] are all based on plenoptic functions. Given the original seven-dimensional plenoptic functions, all these methods place different constraints on the parameters in order to reduce its dimensions. The more constraints we put, the simpler the plenoptic function will be, and the more limitations we will have on our view spaces.

1.2 Thesis Objective

Cube, introduced by Greene [26], can be stored and rendered very efficiently in modern graphic hardware [10]. Thanks to the Point Grey Ladybug camera, the significant problems relating to cube capture and generation when used with real, non-graphic images are easily solved. However, the greatest difficulty of cubic representation, namely estimating image flow fields across the boundaries between faces and at corners is still a big challenge to solve. In fact, how to solve such problems consists of a necessary step in virtual cube generation.

This thesis, destined to generate virtual cubes for image-based navigation, will apply techniques and applications of computer vision to solve the problems relating to virtual cube generation.

1.3 Thesis Contributions

This thesis has made following original contributions:

- A method for feature matching between cubic panoramas is proposed. The method applies non-panorama feature matching techniques to cubic panoramas, and then adopts a subsequent process for outlier removals. This method has the advantage of estimating accurate feature matches, which leads to a robust computation of essential matrix between cubes.

- An online, low-cost cube warping algorithm is suggested. A warping model is constructed for generating novel views between two aligned cubes. The approach is based on a simplified model of cube pixel displacements when a navigator moves from one cube to another.

- A novel raytracing like image-based interpolation algorithm is proposed. Instead of attempting to adopt traditional dense reconstruction approaches, the method try to reconstruct colours with colour invariance constraints. By designing a guided depth searching strategy, the method can generate a novel scene view with maximized photo consistency.

1.4 Thesis Outline

The thesis begins with an introduction of cubic panoramas in Chapter 2: capture, generation, mathematic model and geometry. The chapter firstly presents a brief explanation of cube image form, the camera used to capture the cube as well as cube generation process. Then cube feature matching techniques with a subsequent process for outlier removals will be followed. The next section of Chapter 2 will discuss the cubic epipolar geometry, mainly on cube essential matrix, rotation matrix and translation vector. After that, the accurate estimation of essential matrix, rotation matrix and translation vector will be illustrated, and a global consideration of the cube as whole and not as a

multi-sensor-camera system will be suggested.

Chapter 3 discusses our method of *cube warping*: a new fast, online approach to generate novel view between two aligned cubes with small translation. We will construct a simplified model of cube pixel displacements for simulating walkthrough. Then, the optical flow techniques will be used to decide the "warping scales". After the warping model applied, the result will be discussed. The algorithm costs will be also covered.

Chapter 4 considers how to generate an arbitrary novel cubic view given a set of input cubes. A new raytracing like image-based interpolation method is proposed. The processes for choosing novel viewpoint and retinal planes will be described. Then the colour reconstruction with *colour invariance constraints* will be emphasized. The occlusion and disocclusion issues will be also covered. Finally, new cubic images will be generated with our algorithm and be compared with the actual cubes.

The conclusions as well as a short comparison of our two algorithms will be given in chapter 5.

A number of appendixes will be given at the end of the thesis. Some properties of cube geometry will be covered in Appendix A, Appendix B and Appendix C, mainly on transformations between cube image point and 3D space point. Appendix D will present the cube intrinsic matrix. The 3D triangulation reconstruction method will be discussed in Appendix E.

Chapter 2

Cubic Panorama: Geometry

2.1 Introduction

Panoramic images can provide an unobstructed or complete view of an area. They can be in the forms of cylinder, sphere or cube. The panoramic images are used in applications such as robotic navigation, virtual environment navigation or immersive viewing. A number of techniques have been developed traditionally to generated panoramic images. Such techniques include capturing with a lens of very large field of view, taking an image onto a long film by using a panoramic camera, using catadioptric cameras (mixture of mirrors and lenses, such as parabolic cameras or mirrored pyramids) and applying image mosaic techniques.

The NAVIRE project [1] developed at University of Ottawa uses cubic panoramas generated from the Point Grey Ladybug spherical digital video camera. The goal of the project is to develop image-based rendering(IBR) system to allow a user to virtually walk through a remote real environment using pre-captured panorama images. The cubic panoramas are captured by a multi-sensor panoramic camera with high resolution. With

[1] http://www.site.uottawa.ca/research/viva/projects/ibr/

six planar images, cubic panoramas are easy to manipulate and are efficient for rapid rendering in modern graphics cards.

Cubic panoramas, or cubes for short forms, are very suitable for 3D reconstruction because of their implicit calibration and cube face relationships. In this chapter, we address the characteristics of the cubes, the geometry of the cubes as well as the rotation and translation of the cubes. Two-view from cubic panoramas is of particular interest because of the special camera configuration and its resulting epipolar geometry.

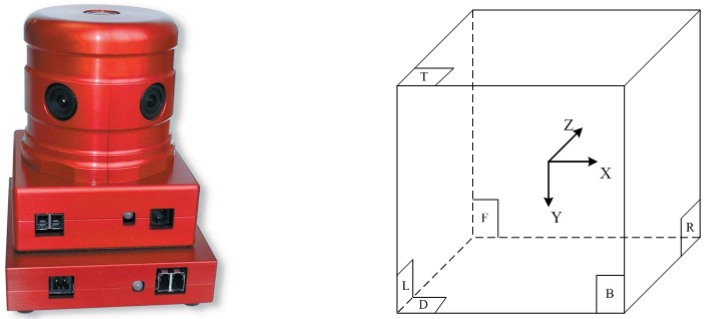
This chapter is structured as follows: The next section discusses the cube capture and generation processes. The feature matching techniques will be presented in section 3. After that, the cubic epipolar geometry are followed. Sections 4 will describe some experimental results and analysis. The chapter is completed with a brief conclusion.

2.2 Cubic panoramas : image generation

2.2.1 Notations

We make heavy use of elementary projective geometry in this chapter. The reader who is unfamiliar with these topics is referred to the recent computer vision books[30, 17]. We write vectors and matrices in bold face and scalars in italic. We also write scene and image quantities in capitals and lowercases, respectively. When possible, the corresponding scene and image quantities are written with the same letters. Cube images, C, cube faces, F, and 3D scenes, S, are all expressed as sets of points. For example, a cube image $2D$ point $(x, y, 1)^T = \mathbf{x} \in C$ or a cube face $2D$ point $(x, y, 1)^T = \mathbf{x} \in F$ is the projection of a scene $3D$ point $(X, Y, Z, 1)^T = \mathbf{X} \in S$.

A cubic panorama is made of six identical faces. Each of them can be seen as a image plane of a standard pinhole camera with 90° field of view. We name the six



(a) Point Grey Ladybug camera.

(b) Cube reference frame.

Figure 2.1: Point Grey Ladybug camera and cube reference frame

faces as: up, left, front, right, back, down, and label each face of the cube as F_i, for $i \in \{\mathtt{U}, \mathtt{L}, \mathtt{F}, \mathtt{R}, \mathtt{B}, \mathtt{D}\}$, with `U` standing for the up face, `L` standing for the left face and so on. As shown in Figure2.1(b), the cube reference frame is chosen as follows: the original point is located at the center of the cube with the x axis pointing to the "right" face, the y axis toward "down" face and the z axis toward the "front" face.

2.2.2 Cube generation

The cubic panoramas used in NAVIRE project are captured with the Point Grey Ladybug camera (see Figure2.1(a)) [2]. The Ladybug camera consists of six 1024x768 color CCDs, with five CCDs positioned in a horizontal ring and one positioned vertically, which capture a view of the environment with 360 degrees around the azimuth as well as a top view.

After captured, the six raw images, with roughly 80 pixels overlap between neighbouring sensors, need to be stitched together to form a panorama image. This can be done by fusing the six images to form a spherical mesh first, and then projecting it into

[2] http://www.ptgrey.com/

Figure 2.2: A cube image laid out in cross pattern

six cube faces. Since there is no sensor positioned downward, there is a black hole in the bottom face of the cube.

An example cube of images taken with the Ladybug and projected to generate cubic panorama is shown in Figure2.2. Here cube faces are laid out in a cross pattern with the faces in the order (from top to bottom and left to right): up, left, front, right, back, down.

2.3 Cubic panoramas : feature matching

Feature matching is a foundation of computer vision. It has been widely used in camera calibration, 3D reconstruction, object recognition, image-based navigation, etc. Even though feature matching has been intensely covered in the literature, its applications to panoramic images are still less-addressed.

In this section, we will introduce some methods of feature matching and their applications in cubic panorama. Since we are only interested in applying such applications in cube, we will discuss them in general. For detail discussions, readers can refer to the related articles.

2.3.1 Feature detection

In order to find feature matches, the feature points should be detected first. There are a lot of feature point detectors in the literature [28, 67, 20]. According to the evaluation work of Schmid [57], Harris feature point [28] detector performs best among some of them due to its reliable and invariant to noise and perspective distortion. In fact, it is the most commonly used feature point detector. The basic idea of Harris detector is to find image locations where the intensity changes in two directions. The most significant intensity changes are given by the eigenvectors of the auto-correlation matrix, which takes into account the first derivatives of the intensity on a local window (Gaussian). Harris detector is widely used in computer vision because of its lower computation cost and strong stability to noise, image deformation, and illumination variance. However, its performances degenerate quickly with scale variation. Also, Harris features are not distinctive enough to match the features.

For a feature to be scale-invariant and distinctive, many region detectors have been developed. Among them, Harris-Laplacian [50], Hessian-Laplacian [51] and Scale In-

variant Feature Transform (SIFT) [45] are some of the most popular and effective ones. Using a scale-adapted Harris function, the Harris-Laplace detector searches keypoints first. Then it selects the feature points for which the Laplacian-of-Gaussian reaches a stable peak at a value considered as the scale. The Hessian-Laplace detector uses Hessian determinant for scale detection. A Laplacian-of-Gaussian function is also used for selecting a maximum over scale. SIFT convolves Difference-of-Gaussian(DoF) function with a image region and searches for scale-space extremes. It detects the blob-like structures, as well as edges.

2.3.2 Feature matching

Feature matching is the process to find corresponding features in two or more different views of same scene. There are a lot of approaches for matching features. They can be classified into two categories: region-based matching and feature-based matching. According to evaluation work of [51], region-based matching perform better than feature-based matching.

A popular region-based matching method is variance normalized correlation (VNC) [72]. The main idea of this corresponding method is: for a correlation window around a feature point in the first image, try to search a window area of the correlated point in second image, and perform a correlation operation. Such correlation approach can give good results as long as the image baseline is not widely separated.

In order to make features invariant to illumination changes and rotation, SIFT applies a descriptor with a 3D histogram of gradient locations and orientations. The contribution to the location and orientation bins is weighted by the gradient magnitude. The descriptor is very robust to small geometric distortions. With the combination of a scale-invariant region detector and the descriptor, SIFT relies on the distinctiveness of features

to identify right correspondences without any ambiguity.

2.3.3 Outlier removal

Although many feature matching processes are robust enough, some false matches may survive. This is especially the case when we try to find as many matches as possible. The reason is quite obvious: the tough matching constraints not only discard the outliers but also block good matches. Therefore, an outlier removal process is needed to eliminate outliers.

Outlier removal: epipolar constraints

The principle to eliminate mismatches through epipolar geometry is very simple: discard points that lie too far from related epipolar lines. Consider a candidate match $(\mathbf{x}, \mathbf{x}')$, if they correspond, $\mathbf{x}'$ lies on the epipolar line $\mathbf{l}' = \mathbf{F}\mathbf{x}$ corresponding to the point $\mathbf{x}$. Here $\mathbf{F}$ is fundamental matrix. In other words $\mathbf{x}'^T\mathbf{F}\mathbf{x} = \mathbf{x}'^T\mathbf{l}' = 0$. This is a necessary condition for points to be matches. A similar condition applies to essential matrix $\mathbf{E}$ with the following equation:

$$\hat{\mathbf{x}}'^T\mathbf{E}\hat{\mathbf{x}} = 0 \tag{2.1}$$

We use following function to compute the distance to the epipolar line and validate correspondences:

$$\frac{1}{2}\left(d\left(\hat{\mathbf{x}}_i', \mathbf{E}\hat{\mathbf{x}}_i\right)^2 + d\left(\hat{\mathbf{x}}_i, \mathbf{E}^T\hat{\mathbf{x}}_i'\right)^2\right) \tag{2.2}$$

Where d is Euclidean distance, and $(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_i')$ is the *ith* candidate match expressed in *normalized coordinates*(see [30], *Page 257*). The initial threshold of tolerated distance from epipolar lines is preset at a high value (for example 6 pixels) first, and then is iteratively decreased during the refinement steps as the epipolar geometry (here we mean

essential matrix) becomes more and more precise.

The effectiveness of this method for outlier removal depends on how accurate the essential matrix E is computed. More details on essential matrix discussions will be given later. If no outlier exists, 8 point algorithm [44] can be used to compute essential matrix. Since points are mismatched, some robust techniques are need to discard the false matches and estimate robustly the essential matrix. Among those robust methods, the *random sampling consensus* (RANSAC) [18] and *least-median-of-squares*(LMedS) [72] are two most popular ones.

The presence of mismatches affects the robustness of estimating epipolar geometry, and the accuracy of epipolar geometry in return determines the effectiveness of outlier removal. As we can see the feature detection, feature matching and outlier removal are no longer considered as separated steps.

Outlier removal: 2D reprojection constraints

Because the intrinsic matrix K of cubic panorama is implicit known, we can also use another method to remove outliers by 2D reprojection constraints.

Considering a candidate match $(\mathbf{x}, \mathbf{x}')$ of the image pair I and I' of the same scene S, if cameras are fully calibrated the 3D point $\mathbf{X}$ can be reconstructed from the points $\mathbf{x}$ and $\mathbf{x}$' by the method *linear triangulation* (see Appendix E). Then, 3D point $\mathbf{X}$ is projected back to the image pair I and I'. If the new back-projected points exhibit large 2D errors (large distance from the original points), the candidate match $(\mathbf{x}, \mathbf{x}')$ is misdetected and should be removed.

2.3.4 Cube feature matching

As noted before, a cubic panorama has six non-overlapping identical faces. Each face may be regarded as a image plane of a standard pinhole camera with 90° field of view, and all the cameras which take the six face images are identical and centered at the same optical center, which is also the cube center.

For cube feature matching, it is natural to detect and match features on a face-by-face basis. However this is often proved to be problematic. Match candidates might not be on the same corresponding face of the cubes. Sometimes features on one face of a cube even end up with being on three different faces of another cube. Also, for virtual navigation applications, the cubes are often taken in any desired positions with any arbitrary rotations. Therefore there may be rotation and/or scale variations. Thus, a feature matching method with scale-invariant should be used.

To sum up, the matching and detection are processed on two cube images laid out in cross pattern (see Figure2.2). We use scale-invariant Lowe's SIFT method to detect and match initial features with stringent threshold. Next, the robust RANSAC method is used to compute essential matrix **E**. Then two correspondence validation steps are followed: "epipolar constraints" step first and "2D reprojection constraints" step next. Again the essential matrix **E** is compute with more accurate matches. After that SIFT method is applied again to find more matches with relaxed threshold, and eventually, more precise matches are found by using recovered essential matrix **E** to remove the outliers.

2.4 Cubic panoramas : epipolar geometry

2.4.1 Notation

We first introduce some notations and conventions used throughout this section. Considering two cubes, cube C and cube C', we choose the reference frame system shown in Figure2.1(b), and attach the world reference frame to cube C. Given a 3D point $\mathbf{X}$, we use $\mathbf{x}_i$ as its projection on the face i of cube C, for $i \in \{\mathtt{U}, \mathtt{L}, \mathtt{F}, \mathtt{R}, \mathtt{B}, \mathtt{D}\}$, with $\mathtt{U}$ standing for the up face, $\mathtt{L}$ standing for the left face and so on. We also denote $\mathbf{x}'_i$ as the projection of point $\mathbf{X}$ on the face i of cube C'. The intrinsic matrices for cube C and cube C' are noted as $\mathbf{M}_{int}$ and $\mathbf{M}'_{int}$, respectively. In the same way, the extrinsic matrices are written as $\mathbf{M}_{ext-i}$ and $\mathbf{M}_{ext-i}{}'$. The projection matrices are expressed with $\mathbf{P}_i$ and $\mathbf{P}'_i$, where i stands for different face.

2.4.2 Intrinsic or calibration matrix

Cubic panoramas are very suitable for 3D reconstruction because of their implicit calibration and cube face relationships. All six cube faces can be regarded as images taken by the same camera with 90° field of view, which rotates about its optical centre 90° at a time with fixed focal length. Therefore, in the case of a cube of side d with the frame shown in Figure2.1(b), the image plane is at a distance $\frac{d}{2}$ from camera center, and the principal point is always at $\left(\frac{d}{2}, \frac{d}{2}\right)$ of image plane. Thus the cube *intrinsic matrix* $\mathbf{M_{int}}$ or *calibration matrix* $\mathbf{K}$ may be written as:

$$\mathbf{K} = \begin{bmatrix} \frac{d}{2} & 0 & \frac{d}{2} \\ 0 & \frac{d}{2} & \frac{d}{2} \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

For two cubes, cube C and cube C', if they have same cube size of side d, their respective intrinsic matrix $\mathbf{M}_{int}$ and $\mathbf{M}'_{int}$ will be equal:

$$\mathbf{M}_{int} = \mathbf{M}'_{int} = \mathbf{K} = \begin{bmatrix} \frac{d}{2} & 0 & \frac{d}{2} \\ 0 & \frac{d}{2} & \frac{d}{2} \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

2.4.3 3D point and its projection to cube faces

As shown in Figure2.1(b), the front face camera of cube C is attached to the world frame. The camera projection matrix for front face of cube C is:

$$\mathbf{P}_F = \mathbf{K}\mathbf{M}_{ext\text{-}F} = \mathbf{K}\,[\mathbf{I}_3|0] \tag{2.5}$$

With $\mathbf{I}_3$ being identity matrix of order 3.

As noted before, all six cube faces can be regarded as images taken by the same camera, which rotates about its optical centre 90° for different faces with fixed focal length. Therefore, the relations between extrinsic matrix of front face and those of other faces can be written as:

$$\mathbf{M}_{ext\text{-}i} = \mathbf{R}_i\mathbf{M}_{ext\text{-}F} = [\mathbf{R}_i|0] \tag{2.6}$$

Where $\mathbf{R}_i$ is given in appendix A.

Then the camera projection matrix of an arbitrary face of cube C becomes:

$$\mathbf{P}_i = \mathbf{K}\mathbf{M}_{ext\text{-}i} = \mathbf{K}\,[\mathbf{R}_i|0] \tag{2.7}$$

Given a 3D point $\mathbf{X}$, its projection on the face i of cube C:

$$\mathbf{x}_i = \mathbf{P}_i\mathbf{X} = \mathbf{K}\mathbf{R}_i\mathbf{X} \tag{2.8}$$

Thus:

$$^3\mathbf{X} = \mathbf{P}_i^{-1}\mathbf{x}_i = (\mathbf{K}\mathbf{R}_i)^{-1}\,\mathbf{x}_i = \mathbf{R}_i\mathbf{K}^{-1}\mathbf{x}_i \tag{2.9}$$

By replacing equation 2.9 into following equation, we have:

$$\mathbf{x}_F = \mathbf{P}_F\mathbf{X} = \mathbf{K}\mathbf{X} = \mathbf{K}\mathbf{R}_i\mathbf{K}^{-1}\mathbf{x}_i \tag{2.10}$$

or

$$\mathbf{x}_i = \mathbf{K}\mathbf{R}_i\mathbf{K}^{-1}\mathbf{x}_F \tag{2.11}$$

In the general case, the conversion of two cube image points between different cube faces, i and j can be written as:

$$\mathbf{x}_j = \mathbf{K}\mathbf{R}_j\mathbf{K}^{-1}\mathbf{x}_F = \mathbf{K}\mathbf{R}_j\mathbf{K}^{-1}\mathbf{K}\mathbf{R}_i\mathbf{K}^{-1}\mathbf{x}_i = \mathbf{K}\mathbf{R}_j\mathbf{R}_i\mathbf{K}^{-1}\mathbf{x}_i \tag{2.12}$$

2.4.4 Fundamental matrix between two cubes

A cubic panorama has six faces. Between any face i of cube C and face j of cube C', there is a fundamental matrix, noted as $\mathbf{F}_{ij}$, $i, j \in (T, L, F, R, B, D)$. Therefore, there are totally $6 \times 6 = 36$ fundamental matrices between two cubes. However, due to the intrinsic relationship between the faces, all these fundamental matrices are related. Thus, after any one of them is estimated, all others can be computed from it.

First, let us denote the fundamental matrix between the two front faces of cube C

[3]For rotation matrix, we have $\mathbf{R}_i^{-1} = \mathbf{R}_i$

and cube C' as $\mathbf{F}_{FF}$. According to epipolar geometry, we have:

$$\mathbf{x}_F'^T \mathbf{F}_{FF} \mathbf{x}_F = 0 \tag{2.13}$$

where $\mathbf{x}_F$, $\mathbf{x}'_F$ are a pair of matches in the front face of cube C and cube C', respectively.

More general, for a pair of correspondences $\mathbf{x}_i$, $\mathbf{x}'_j$, we also have:

$$\mathbf{x}_j'^T \mathbf{F}_{ij} \mathbf{x}_i = 0 \tag{2.14}$$

From 2.10 and 2.13

$$\mathbf{x}_j'^T \left(\mathbf{K}\mathbf{R}_j\mathbf{K}^{-1}\right)^T \mathbf{F}_{FF} \mathbf{K}\mathbf{R}_i\mathbf{K}^{-1}\mathbf{x}_i = 0 \tag{2.15}$$

Comparing 2.14 and 2.15, we can write:

$$\mathbf{F}_{ij} = \left(\mathbf{K}\mathbf{R}_j\mathbf{K}^{-1}\right)^T \mathbf{F}_{FF} \left(\mathbf{K}\mathbf{R}_i\mathbf{K}^{-1}\right) \tag{2.16}$$

In addition, from 2.12 and 2.14, we can get:

$$\mathbf{x}_n'^T \left(\mathbf{K}\mathbf{R}_j\mathbf{R}_n\mathbf{K}^{-1}\right)^T \mathbf{F}_{ij} \mathbf{K}\mathbf{R}_i\mathbf{R}_m\mathbf{K}^{-1}\mathbf{x}_m = 0 \tag{2.17}$$

So, in the general case, we have the following equation for two arbitrary fundamental matrices:

$$\mathbf{F}_{mn} = \left(\mathbf{K}\mathbf{R}_j\mathbf{R}_n\mathbf{K}^{-1}\right)^T \mathbf{F}_{ij} \left(\mathbf{K}\mathbf{R}_i\mathbf{R}_m\mathbf{K}^{-1}\right) \tag{2.18}$$

2.4.5 Essential matrix between two cubes

Essential matrix

The essential matrix is the specialization of the fundamental matrix. For a 3D point $\mathbf{X}$ and a camera matrix $\mathbf{P} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}]$, the image point is $\mathbf{x} = \mathbf{PX}$. By removing intrinsic matrix from $\mathbf{x}$, we have:

$$\hat{\mathbf{x}} = \mathbf{K}^{-1}\mathbf{x} = [\mathbf{R} \mid \mathbf{t}]\,\mathbf{X} \tag{2.19}$$

$\hat{\mathbf{x}}$ is called normalized image coordinate. The defining equation for the essential matrix is:

$$\hat{\mathbf{x}}'^{T}\mathbf{E}\hat{\mathbf{x}} = 0 \tag{2.20}$$

Substituting for $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$, it yields:

$$\mathbf{x}'^{T}\mathbf{K}'^{-T}\mathbf{E}\mathbf{K}^{-1}\mathbf{x} = 0 \tag{2.21}$$

Comparing this with $\mathbf{x}'^{T}\mathbf{F}\mathbf{x} = 0$, we obtain:

$$\mathbf{F} = \mathbf{K}'^{-T}\mathbf{E}\mathbf{K}^{-1} \tag{2.22}$$

or

$$\mathbf{E} = \mathbf{K}'^{T}\mathbf{F}\mathbf{K} \tag{2.23}$$

Essential matrix between two cubes

Same as fundamental matrix, there are totally $6 \times 6 = 36$ essential matrices between two cubes. Between any face i of cube C and face j of cube C', the essential matrices are noted as $\mathbf{E}_{ij}$, $i, j \in (T, L, F, R, B, D)$. As noted before, the intrinsic matrix is same for

any two cubes: $\mathbf{K}' = \mathbf{K}$. Thus, the equation 2.22 and 2.23 can be written as:

$$\mathbf{F}_{ij} = \mathbf{K}^{-T}\mathbf{E}_{ij}\mathbf{K}^{-1} \tag{2.24}$$

and

$$\mathbf{E}_{ij} = \mathbf{K}^{T}\mathbf{F}_{ij}\mathbf{K} \tag{2.25}$$

Therefore, for $\mathbf{E}_{FF}$, the essential matrix between two front faces of cube C and cube C', we have:

$$\mathbf{E}_{FF} = \mathbf{K}^{T}\mathbf{F}_{FF}\mathbf{K} \tag{2.26}$$

Replacing $\mathbf{F}_{FF}$ with $\mathbf{F}_{ij}$ from equation 2.16, and after some manipulations, we get:

$$\mathbf{F}_{ij} = (\mathbf{K}\mathbf{R}_j)^{-T}\,\mathbf{E}_{FF}\,(\mathbf{K}\mathbf{R}_i)^{-1} \tag{2.27}$$

Comparing 2.14 and 2.27, we conclude:

$$\mathbf{x}_j'^{T}\,(\mathbf{K}\mathbf{R}_j)^{-T}\,\mathbf{E}_{FF}\,(\mathbf{K}\mathbf{R}_i)^{-1}\,\mathbf{x}_i = 0 \tag{2.28}$$

Essential matrix $\mathbf{E}_{FF}$ between two front faces is very important. It provides a convenient method to compute *rotation matrix* $\mathbf{R}$ and *translation vector* $\mathbf{t}$ between two cubes. More discussions on this will be covered next. With equation 2.28, we can estimate essential matrix $\mathbf{E}_{FF}$ given a group of matches between two cubes. The correspondence $\left(\mathbf{x}_i, \mathbf{x}'_j\right)$ can be any face image points of cube C and cube C', respectively.

Essential matrix and 3D coordinates of cube points

In last section, we handled every cube as a six-sensor camera and concluded that there are $6 \times 6 = 36$ essential matrices between two cubes. However, if we treat a cube as

a whole and not as a multi-sensor-camera system, there is only one essential matrix involved.

Due to the special representation of cube, the coordinates of a 2D image point on a cube face can be expressed by the corresponding 3D coordinates on the cube face. Each image point of a cube can be mapped to a 3D vector on the cube face. For details and their transformation, please refer to Appendix C.

According to the work of [36], there is only one essential matrix involved if we use 3D vector on the cube face instead of 2D image point. Therefore, the method to computer essential matrix between two cubes can be simplified with following setps: first, find matches between two cubes; second, transform the 2D matching image points into 3D vectors on the cube face; third, estimate essential matrix with the 3D vectors.

2.4.6 Rotation matrix and translation vector between cubes

Each cubic panorama is captured from a point in space. Two different cubes will have different reference frames, related by a *rotation matrix* $\mathbf{R}$ and *translation vector* $\mathbf{t}$.

Considering two cubes, cube C and cube C', we choose the reference frame system shown in Figure2.1(b), and attach the world reference frame to the front face camera of cube C. We use the similar reference frame system for cube C', and cube C and cubeC' will have different world points and orientations, expressed by a *rotation matrix* $\mathbf{R}$ and *translation vector* $\mathbf{t}$. Thus, the camera projection matrices for cube C and cubeC' will be:

$$\mathbf{P}_F = \mathbf{K}\,[\mathbf{I}_3|0] \qquad\qquad \mathbf{P}'_F = \mathbf{K}\,[\mathbf{R}|\mathbf{t}] \tag{2.29}$$

Therefore, we have following equation for the essential matrix $\mathbf{E}_{FF}$:[4]

$$\mathbf{E}_{FF} = [\mathbf{t}]_{\times} \mathbf{R} \tag{2.30}$$

The equation 2.30 is very important in representing the relationship between two cubes, and will be used all through this thesis. To be succinct, we will use $\mathbf{E}$ instead of $\mathbf{E}_{FF}$ in the future part to express cube relationships when there is no confusion involved.

One of the most important properties of the essential matrix $\mathbf{E}$ is: equation 2.30 can be used to extract *rotation matrix* $\mathbf{R}$ and *translation vector* $\mathbf{t}$ (up to scale) as per existing methods.[5] Since all the cubes have the same calibration matrix $\mathbf{K}$ given the same cube size, $\mathbf{R}$ and $\mathbf{t}$ provide the sufficient requirement for 3D reconstruction of the environment or building a cube map for an interactive navigation.

Therefore, with the essential matrix $\mathbf{E}$ and $\mathbf{R}$, $\mathbf{t}$ recovered from it, a cube can be treated as a whole and not as a multi-camera system. This will result in more global approaches and simplify the processing algorithms.

2.4.7 3D reconstruction between two cubes

Since the calibration matrix $\mathbf{K}$ is implicitly known, cubic panoramas are very suitable for 3D reconstruction. After the *rotation matrix* $\mathbf{R}$ and *translation vector* $\mathbf{t}$ (up to scale) are extracted, a 3D point may be reconstructed from a pair of matches by the method of *linear triangulation* (For details, please refer to Appendix E).

Considering two cubes, cube C and cube C', with the reference frame system as Figure2.1(b) and world origin at the center of cube C, their camera projection matrices

[4]For details, please refer to [30], *Section 9.6*
[5]*Section 9.6* of [30],*Chapter 5* of [17]

are:

$$\mathbf{P}_F = \mathbf{K}\,[\mathbf{I}_3|0] \qquad\qquad \mathbf{P}'_F = \mathbf{K}\,[\mathbf{R}|\mathbf{t}]$$

For the correspondence $\left(\mathbf{x}_i, \mathbf{x}'_j\right)$, image point on face i of cube C and face j of cube C', respectively, the equation 2.10 is used to transfer them into front face of cube C and cube C':

$$(x_F, y_F, 1)^T = \mathbf{x}_F = \mathbf{K}\mathbf{R}_i\mathbf{K}^{-1}\mathbf{x}_i \qquad\qquad (x'_F, y'_F, 1)^T = \mathbf{x}'_F = \mathbf{K}\mathbf{R}_j\mathbf{K}^{-1}\mathbf{x}'_j$$

Then, the 3D point X can be reconstructed for the match $\left(\mathbf{x}_i, \mathbf{x}'_j\right)$ by the linear function of $\mathbf{AX} = 0$, with

$$\mathbf{A} = \begin{bmatrix} x_F\mathbf{p}_F^{3T} - \mathbf{p}_F^{1T} \\ y_F\mathbf{p}_F^{3T} - \mathbf{p}_F^{2T} \\ x'_F\mathbf{p}'^{3T}_F - \mathbf{p}'^{1T}_F \\ y'_F\mathbf{p}'^{3T}_F - \mathbf{p}'^{2T}_F \end{bmatrix} \tag{2.31}$$

Where $\mathbf{p}_F^{nT}$ is the row vector of the n^{th} row of the $\mathbf{P}_F$.

This is a linear equation, which is easy to find the solution by using Direct Linear Transformation (DLT) algorithm (see [30]).

2.5 Experiments

We have performed a number of experiments to recover cube epipolar geometry, mainly *essential matrix* $\mathbf{E}$, *rotation matrix* $\mathbf{R}$ and *translation vector* $\mathbf{t}$.

The experiment scheme is as follows:

1. For two cube images laid out in cross pattern, detect and match few but accurate features by using Lowe's SIFT method with stringent threshold.

2. Estimate essential matrix **E** and discard misdetected matches by robust RANSAC method. The equation 2.28 is used to compute **E**. *Epipolar constraints* or *2D reprojection constraints* is used to validate correspondences.

3. Extract *rotation matrix* **R** and *translation vector* **t** from essential matrix **E** by using equation 2.30

4. Find more matches by using Lowe's SIFT method with relax threshold, and use computed **E** to remove mismatches by *epipolar constraints* or *2D reprojection constraints*.

5. Estimate **E**, **R** and **t** more precisely with more accurate matches.

Cube 1 and Cube 2 (shown in Figure 2.3) are two cubes used for feature matches. Figure 2.4 shows the matching results from Lowe's SIFT method. In order to find as many matches as possible, we used relax threshold for feature matching. The matches are drawn with red arrow line, with one end showing matching point of one cube and arrow end pointing to the corresponding matching point of another cube. There are 837 initial putative matches, and some of them are obviously mismatched. These matches are validated with two different methods: outlier removal by *epipolar constraints* and outlier removal by *2D reprojection constraints*. After *epipolar constraints* is applied, there are 219 mismatches are removed (shown in Figure 2.5). Figure 2.6 shows the result of outlier removal by *2D reprojection constraints*. For this method, 222 outliers are deleted and rejected. From the results, we can see both methods can remove the mismatches effectively.

For *epipolar constraints*, it is necessary to explicate the computation of distance between normalized coordinate point and its epipolar line. $\left(\hat{\mathbf{x}} = (\hat{x}, \hat{y}, 1)^T, \hat{\mathbf{x}}' = (\hat{x}', \hat{y}', 1)^T\right)$ is a correspondence with normalized coordinate for cube C and C', respectively. We need

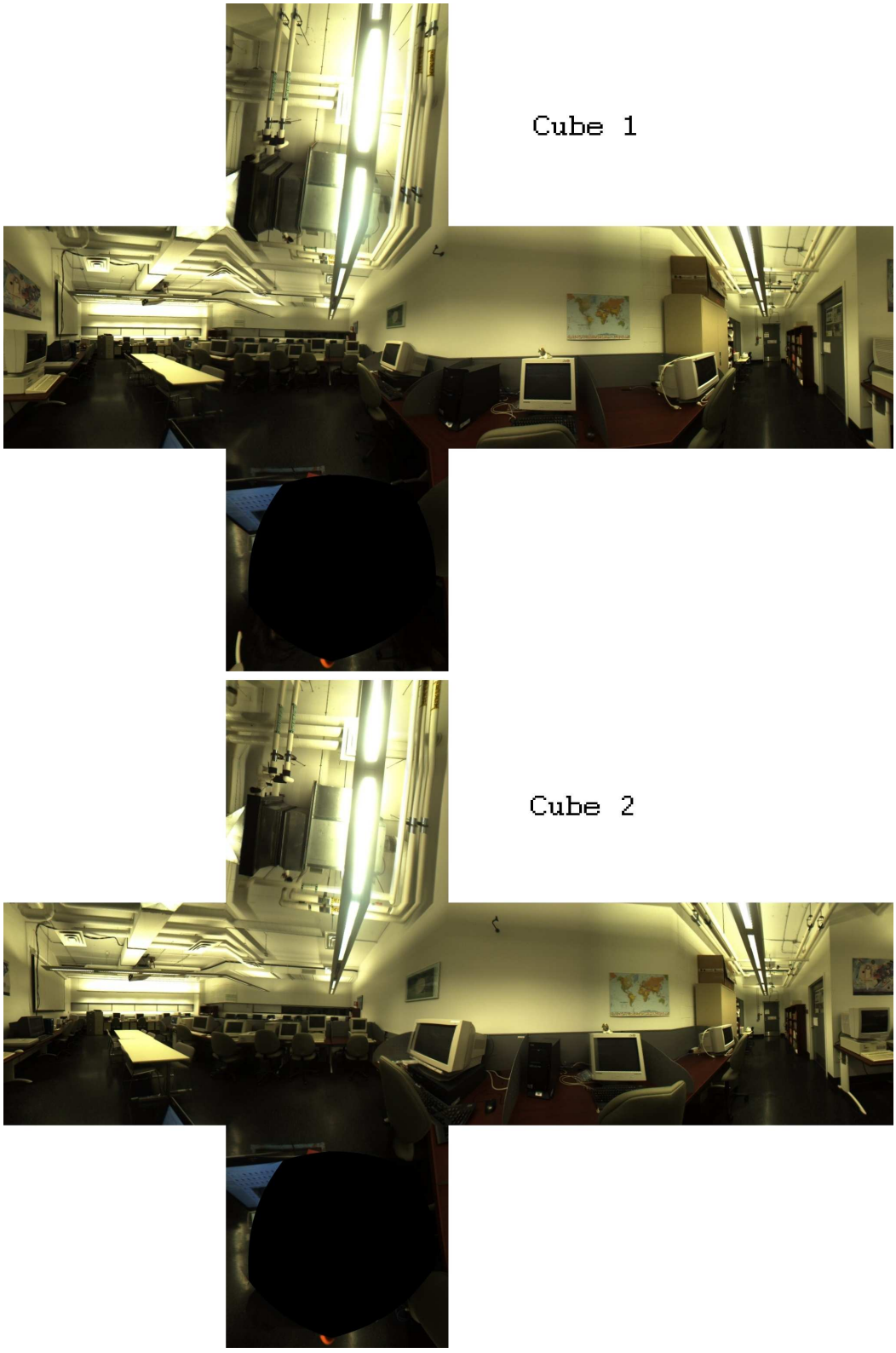


Figure 2.3: Two cubes used to detect matches

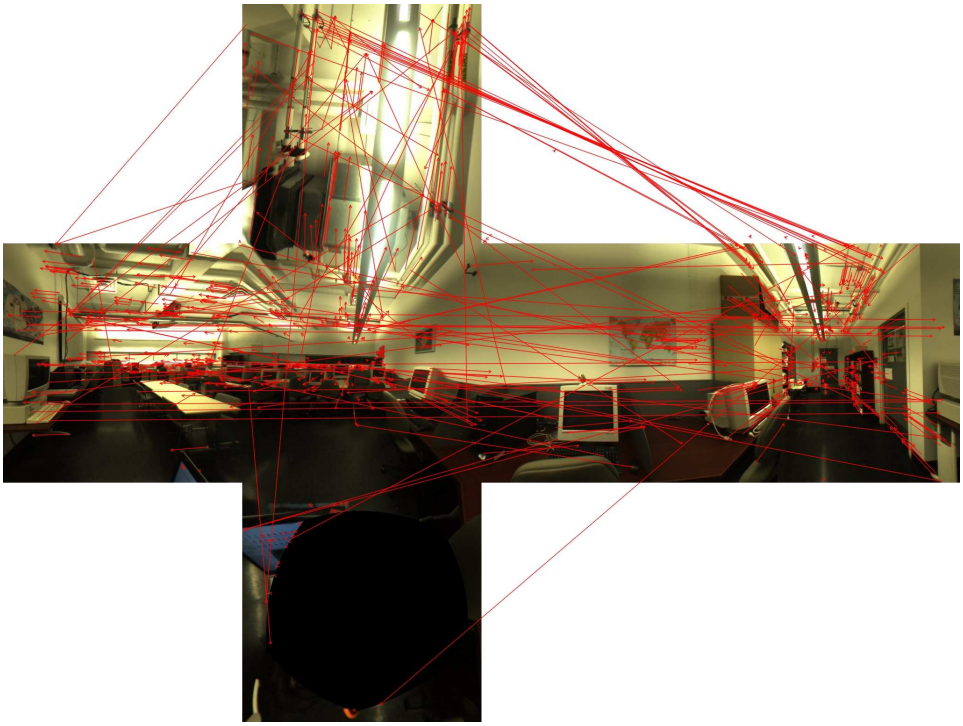


Figure 2.4: Matches obtained by SIFT (837 matches)

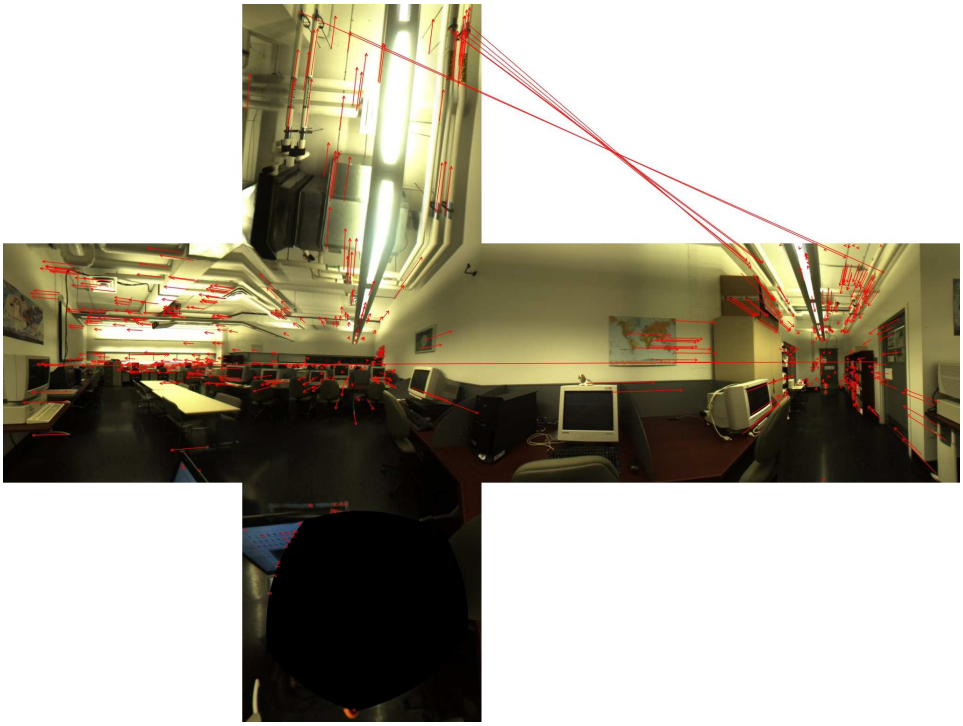


Figure 2.5: Matches after validated with *epipolar constraints* (618 matches)

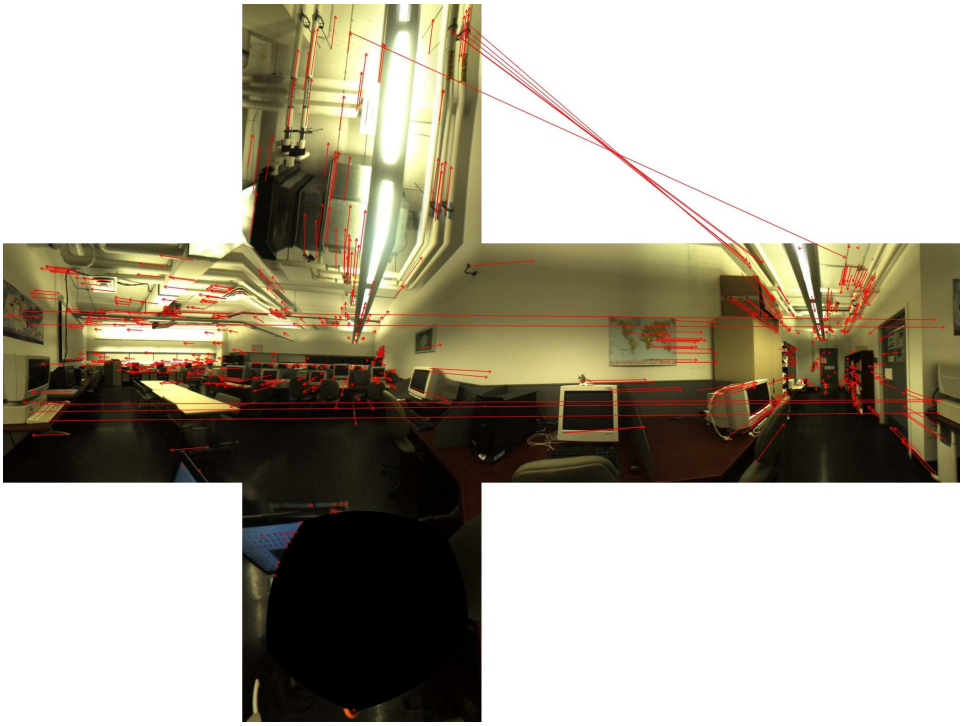


Figure 2.6: Matches after validated with *2D reprojection constraints* (615 matches)

to compute distance of $\hat{\mathbf{x}}'$ to its normalized epipolar line $\mathbf{E}\hat{\mathbf{x}} = (a, b, c)^T$. This distance can be calculated by following equation

$$d\left(\hat{\mathbf{x}}', \mathbf{E}\hat{\mathbf{x}}\right) = \frac{|\hat{\mathbf{x}}' \cdot (\mathbf{E}\hat{\mathbf{x}})|}{\|(\mathbf{E}\hat{\mathbf{x}})\|} = \frac{|a\hat{x}' + b\hat{y}' + c|}{\sqrt{a^2 + b^2}} \tag{2.32}$$

The distance $d\left(\hat{\mathbf{x}}, \mathbf{E}'\hat{\mathbf{x}}'\right)$ can be computed the same way. We average these two distances and compare it with a threshold to decide if the putative match is an outlier. In our experiment, the outlier threshold was set to 2.5. The Figure 2.7 shows the simulation result of distances between matches and their epipolar lines. The distances greater than 5 pixels in *x-axis* are truncated for appropriate display on the figure. The vertical line at 2.5 of this cumulative histogram is epipolar distance applied as a threshold to eliminate outliers. The horizontal line at 618 shows that there are 618 inliers after validation with

epipolar constraints.

For outlier removal with *2D reprojection constraints*, we reconstruct 3D point $\mathbf{X}$ (up to scales) from match $\left(\mathbf{x} = (x, y, 1)^T, \mathbf{x}' = (x', y', 1)^T\right)$ of cube C and cube C', respectively. Then reproject $\mathbf{X}$ back to cube C and cube C', and get the points: $\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}, 1)^T$ and $\tilde{\mathbf{x}}' = (\tilde{x}', \tilde{y}', 1)^T$. The following function is used to compute the *reprojection error*:

$$\epsilon\left(\mathbf{x}, \tilde{\mathbf{x}}\right) = \sqrt{(x - \tilde{x})^2 + (y - \tilde{y})^2} \tag{2.33}$$

Also, the reprojection error $\epsilon\left(\mathbf{x}', \tilde{\mathbf{x}}'\right)$ between $\mathbf{x}'$ and its reprojected point $\tilde{\mathbf{x}}'$ is calculated the same way. Then these two distances are averaged and compared with a threshold to filter outliers.

Figure 2.8 shows the results of reprojection errors for the method of outlier removal with *2D reprojection constraints*. In this cumulative histogram, the vertical line at 0.6 is reprojection error applied as a threshold to eliminate outliers. The horizontal line at 615 shows that there are 615 inliers after validation. The reprojection errors greater than 3 pixels in *x-axis* are truncated for appropriate display on the figure. There are 837 putative matches (shown in Figure 2.4) before validation and the average reprojection error for all these putative matches is 103.09 pixels. After threshold of 0.6 pixels applied, 222 outliers are removed with *2D reprojection constraints*. The average reprojection error for final 615 matches is 0.1876 pixels. This is a quite favourable result considering an ideal pinhole camera model is used for cubic panorama.

We explore two more experiments to test the accuracy of our method in computing *rotation matrix* $\mathbf{R}$ and *translation vector* $\mathbf{t}$ as well as their importance in the relationships among cubic panoramas. We adopt a useful application of projective geometry: *transfer*. That is: for a set of images, given the position of a match in two images, determine the corresponding positions in all other cubes of the set. Our approach consists of the

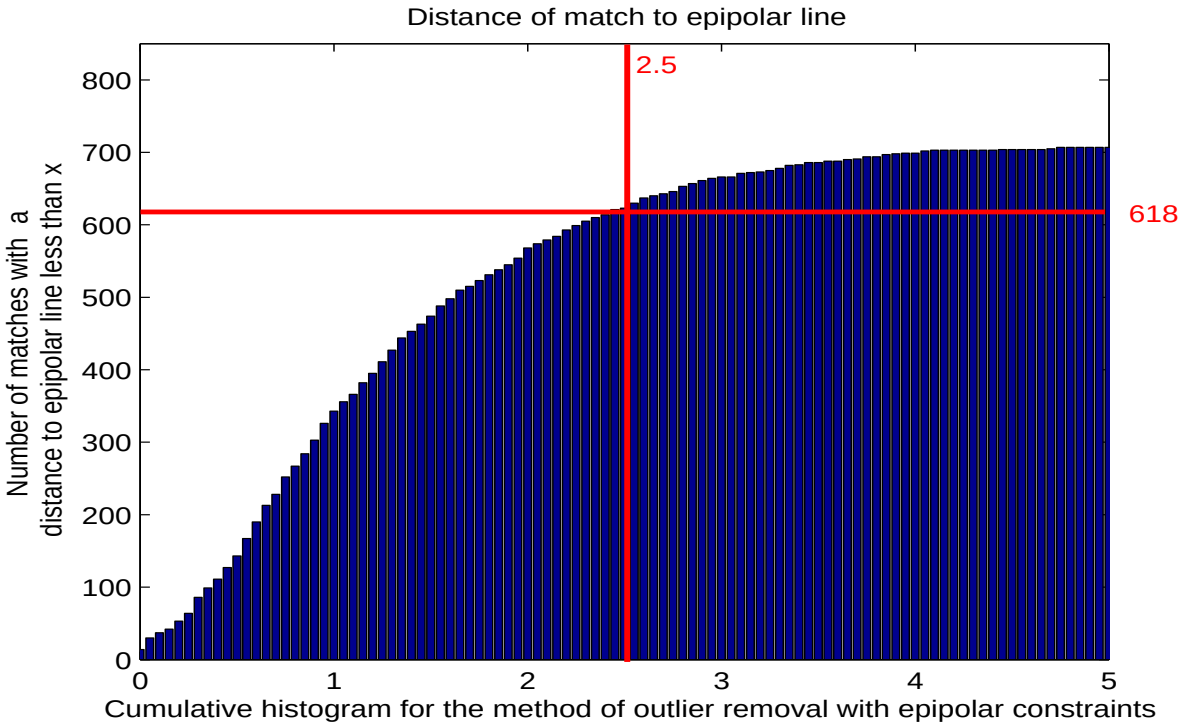


Figure 2.7: Cumulative histogram for the method of outliers removal with *epipolar constraints*. The vertical line at 2.5 is epipolar distance applied as a threshold to eliminate outliers. The horizontal line at 618 shows that there are 618 inliers after validation.

following steps:

1. Use previously stated methods to find matches and compute $\mathbf{R}_j$, $\mathbf{t}_j$ pair by pair for any two cubes of the set.

2. Given correspondences $\mathbf{x}_i \longleftrightarrow \mathbf{x}'_i$ of the first two cubes, triangulate the 3D points $\mathbf{X}_i$ from them using the computed $\mathbf{R}_j$, $\mathbf{t}_j$.

3. Project the 3D points $\mathbf{X}_i$ into all other cubes of the set using the computed $\mathbf{R}_j$, $\mathbf{t}_j$.

4. check if the projected image points are corresponding to the points $\mathbf{x}_i \longleftrightarrow \mathbf{x}'_i$.

For the first experiment we computed *rotation matrix* $\mathbf{R}$ and *translation vector* $\mathbf{t}$ pair by pair among a set of six outdoor cubes. Two corresponding points were chosen manually from two cubes next. Then, a 3D point was constructed from them and projected to

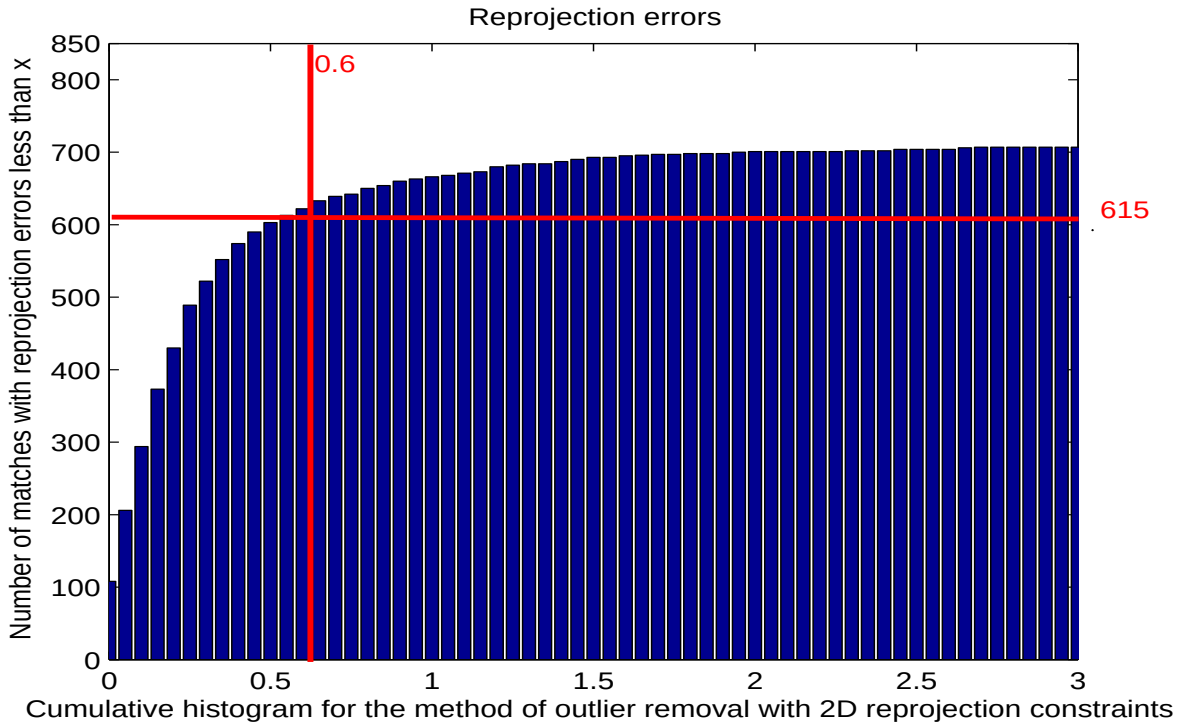


Figure 2.8: Cumulative histogram for the method of outliers removal with *2D reprojection constraints*. The vertical line at 0.6 is reprojection error applied as a threshold to eliminate outliers. The horizontal line at 615 shows that there are 615 inliers after validation.

all other cubes of the set. The experiment results are shown in Figure 2.9. Two points, marked with a cross, were chosen in right face of cube *a* and left face of cube *b*. The "transferred" image points are shown with small crosses in cube *c*, cube *d*, cube *e* and cube *f* of Figure 2.9. These points are pretty accurately "transferred" to the locations they are supposed to be. The reprojection errors expressed as distances between the reconstructed image points and real image point are shown in Table 2.1.

Table 2.1: Reprojection errors

Cube	*Cube c*	*Cube d*	*Cube e*	*Cube f*
Reprojection error	0.012 pixels	0.128 pixels	0.332 pixels	0.002 pixels

In the next experiment, a group of 3D points are constructed from detected matches of the first two cubes, and then projected to all other cubes of the set. We used a set of

six indoor cubes. A part of the detected matches are shown in right face of cube a and cube b of the Figure 2.10. Some "transferred" image points were drawn with red dots in cube c and cube d of Figure 2.11, as well as cube e and cube f of Figure 2.12. Again, the results show that the locations of the "transferred" points are accurate.

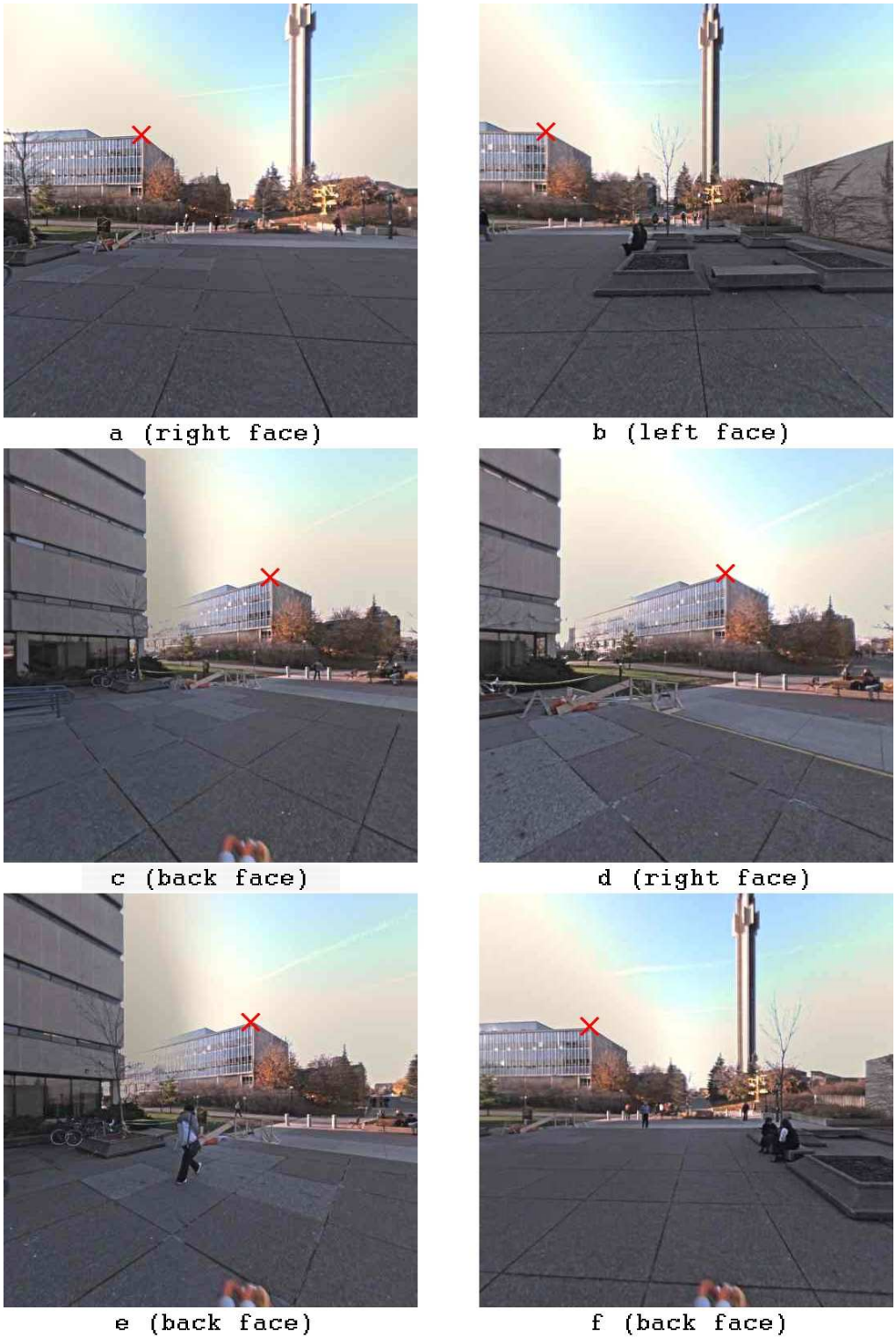
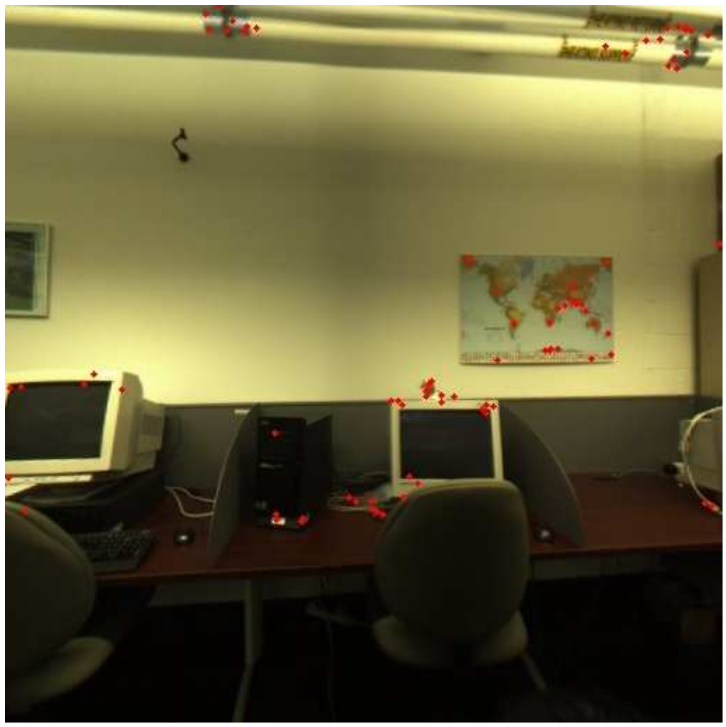


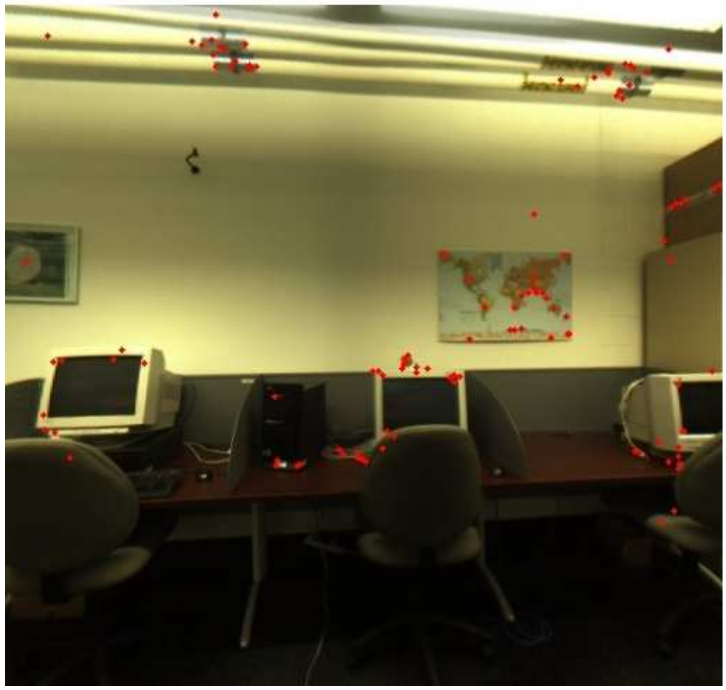
Figure 2.9: Transfer 1: Two corresponding points are chosen from cube **a** and cube **b**. A 3D point is constructed from them, and then projected into all other cubes of the set: cube **c**, cube **d**, cube **e** and cube **f**

Figure 2.10: Transfer 2a: A group of 3D points are constructed from matches shown(drawn with red dots) on cube **a** and cube **b**. They are then projected into all other cubes of the set(shown in Figure 2.11 and 2.12).

Figure 2.11: Transfer 2b: The constructed 3D points are transfered to two cubes of the set: cube **c** and cube **d**



cube e (back face)



cube f (back face)

Figure 2.12: Transfer 2c: The constructed 3D points are transfered to other two cubes of the set: cube **e** and cube **f**

2.6 Discussion and conclusion

All the results show good performance of cube geometry. In particular, very good 3D reconstruction and transfer applications have been presented due to the accurate estimation of *essential matrix* $\mathbf{E}$, *rotation matrix* $\mathbf{R}$ and *translation vector* $\mathbf{t}$.

A number of assumptions are made for our approach. We assume that an ideal pinhole camera model is used for cubic panorama. We also assume that one cubic image is taken with six identical ideal cameras whose optical center are all fixed at cubic center. All these cameras have 90° field of view, and none of their image planes are overlapped.

Based on these assumptions, we explicated cube intrinsic matrix and relationships of cube face. We also deduced epipolar geometry for cubic panoramas. Because of six sensors involved, we concluded that there are total 36 *fundamental matrices* between two cubes. However, all these *fundamental matrices* are related. In fact only one of them is independent and from which the others can be computed.

To simplify the processing of the cube with global approaches, the essential matrix between front faces of two cubes is used. A point in any face of the cube can be changed into front face frame, and therefore processed with a global consideration. This results in the extraction of a more convenient relationship between cubes, namely *rotation matrix* $\mathbf{R}$ and *translation vector* $\mathbf{t}$. With the essential matrix $\mathbf{E}$ and $\mathbf{R}$, $\mathbf{t}$ recovered from it, a cube can be treated as a whole and not as a multi-sensor-camera system.

In spite of ideal pinhole camera model assumption, our experiments show a very good 3D reconstruction result. In addition, the very small *reprojection errors* indicate the effectiveness and efficiency of our approaches. With camera models established successfully and geometry of cubic panoramas constructed accurately, it simplifies the applications in virtual navigations.

Chapter 3

Cube Warping: Single Node Navigation

3.1 Introduction

Visual navigation requires seamless visualization of environment from different viewing positions and orientations. A key component in most virtual environment navigation system is how to produce novel views given a grid of pre-captured reference images.

A lot of image based rendering techniques in the literature are proposed to generate novel views from several referent images. Most algorithms require feature matches. and some even need dense correspondences. The challenges for image based rendering to produce arbitrary views are:

- Dense correspondences are hard to compute automatically, especially when the reference images have large difference in rotation and scale due to viewing orientations and zooming (common in navigation) or large baseline separations.

- Although feature extraction and feature matching can be fast, it often requires

assumptions about the type of features, and the correspondences are often not evenly distributed.

- Even with complex scene rendering in prior, the additional pre-processed data, such as 3-D depth maps or dense/quasi-dense correspondence maps, can put huge burden on storage and network transmission.

- In some scenario, for example walking-through the hallway, the scene is homogeneous and it is practical to use some very low-latency algorithms to approximate virtual navigation.

In addition, there are huge communication costs for transmitting high-resolution cube images to allow on-line users to virtually walk-through. Therefore, it is more practical to provide fast algorithms to enable remote client machines to generate virtual cube images during the transmission of two real cube images.

Because of the computation costs and communication burden of the existing techniques, we want to develop a new method to meet following objectives:

1. Produce photorealistic novel view that is an acceptable approximation for real scene.

2. Guarantee a real-time novel view image synthesis regardless of the scene complexity.

3. Minimize pre-computed information storage and network communication requirements.

4. Allow for novel view rendering on remote client computers.

These goals are very difficult to fulfill. However, for very small translation, we found a warping algorithm which can meet these objectives.

In this chapter, we present an approximate method for panorama navigation called *cube warping*. Our approach is completely automatic and requires only a single cubic panorama as input. By using image warping techniques, our method simulates camera model of walkthroughs. Our algorithm is quite basic. Nevertheless, our resulting novel views look surprisingly realistic and provide good approximation for small cube translation.

3.2 Related work

Image warping is a geometric transformation which maps all pixels in one image plane into pixels in another image plane (see Figure 3.1). The warping addresses the problem of how to transform one image into another smoothly. The warping techniques can be grouped into two classes depending on if interpolation between two or more warped images is involved.

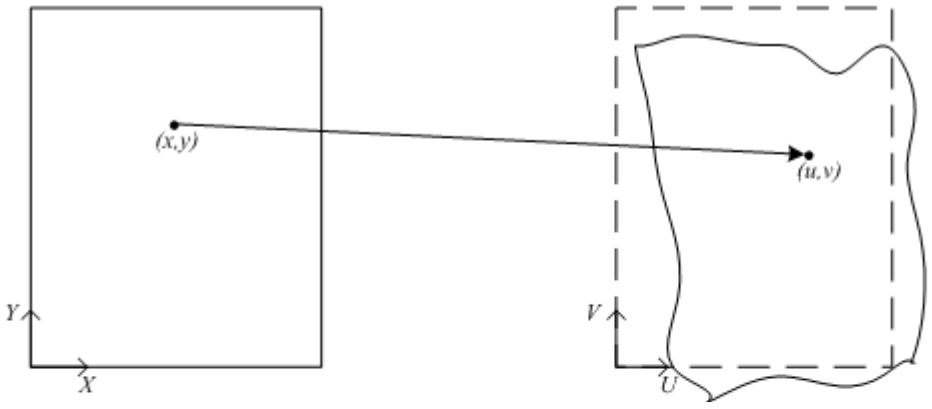


Figure 3.1: Image warping

One group of warping techniques, commonly referred to as "morphing", are often used in educational or entertainment industry. Morphing performs 2D geometric warps on source and destination images, aligns their features into intermediate feature positions,

and interpolates their colors to generate in-between novel images. Field morphing [7] uses multiple line pairs to effectively specify the features on source and destination images. By adopting reverse mapping method, every pair of lines on the images defines a coordinate mapping from the pixels of intermediate image into the pixels of two original images. Because this method uses cross-dissolve to blend colors, it suffers from "ghosts" effect. Mesh warping [70] uses nonuniform meshes to specify the image features. Warps are computed from the correspondence of mesh points with spline interpolation. When complex meshes are involved, it is tedious to specify the features.

Another group of warping techniques generally do not involve image interpolation. They may be used to remove camera optical or perspective distortions [7] or document distortions [4], to register or align two or more images [46, 29], or to stitch mosaic images on smooth surfaces, such as cylindrical [49, 12, 35] or spherical [66, 71], for panorama composing. All these techniques apply geometric transformations to relocate image points from source images to destination images. Transformations can be global or local in nature. Global transformations are often defined by a single equation which is applied to the whole image. Local transformations are applied to a part of image and they are difficult to express concisely.

Our method is belong to the second group. To achieve completely automatic and real-time navigation, we propose to have only a single cube panorama as input, and use warping strategy to approximate walking-through. Several methods are able to perform local navigation from a single image. Hoiem et al.[33] use a single photograph to create a 3D rough model. Instead of attempting to precisely recover computationally intensive model, their algorithm statistically model geometric classes defined by their orientations in the scene. The different areas of input image are labeled and then "cut and folded" to generate novel views. This method produces very good results but works only on some

certain images. *Visual local navigation using warped panoramic images*[9], the main inspiration for our method, presents an algorithm that uses panoramic images to perform local navigation. By modeling image deformations of small camera translation, the panorama image is warped to simulate local homing. Because it omits feature extraction and feature matching, this approach can achieve a fast, on-line local navigation.

3.3 Cube warping

3.3.1 Basic idea

QuickTime VR [12] is probably the earliest effort for the application of the virtual environment navigation. The pre-captured cubic panoramas are distributed as a grid of nodes (2D lattice) in the navigation area, shown in Figure 3.2. The irregular navigation path can be approximated by the path along the grid lines.

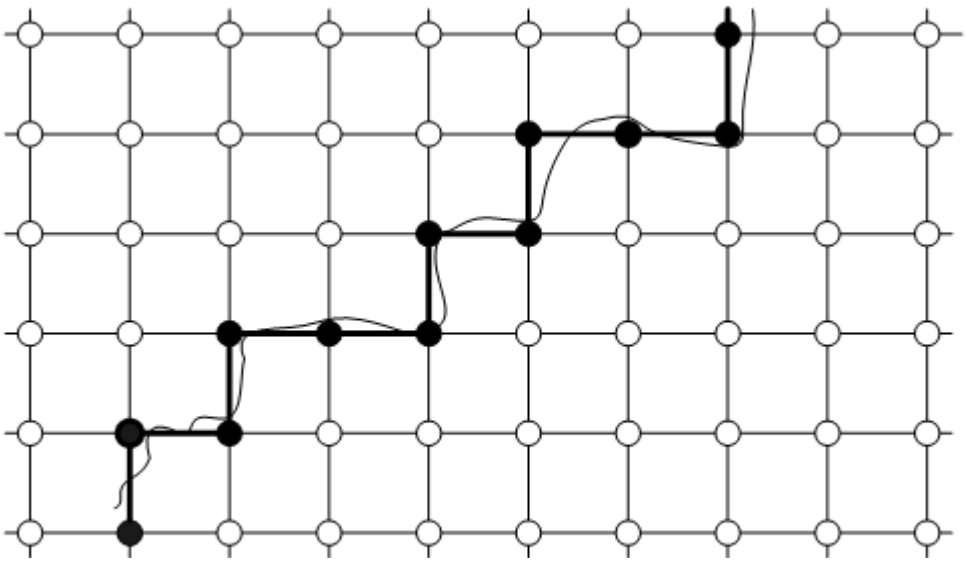


Figure 3.2: Multiple node navigation. An unconstrained navigation path is quantized to the nearest grid nodes

To accomplish smooth navigation, the intermediate views are generated between any two neighbouring nodes of the lattice. Our goal is to synthesize novel view between any two neighbouring cubic panoramas. This means we are only interested in approximating very short displacement: forward, backward and sideways translations.

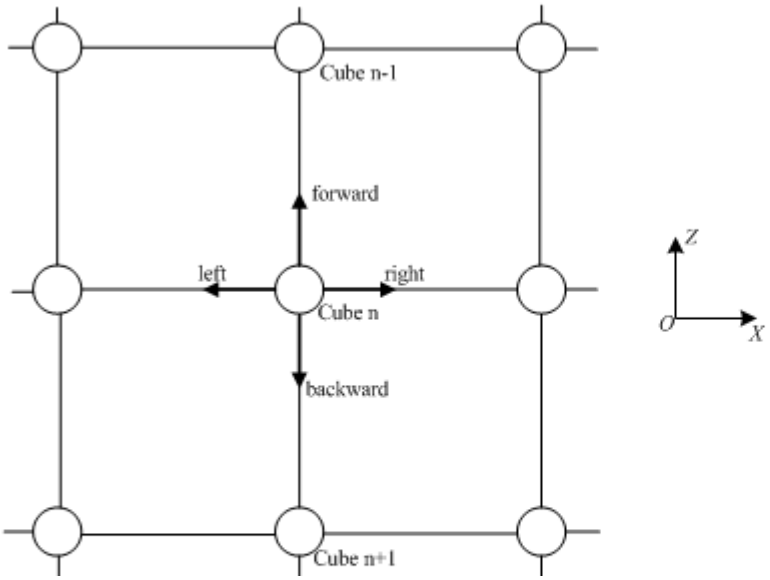


Figure 3.3: Cube navigating

Figure 3.3 shows a cube navigation. We are concerned with four kinds of cube moving: forward, backward, to the left, and to the right. When a cube moves forward, the front face of cube image zooms in, the back face zooms out, and other faces move from front to back. A forward moving model is shown in Figure 3.4. Here cube faces are laid out in a cross pattern with the faces in the order (from top to bottom and left to right): up, left, front, right, back, down.

A similar deformation happens when the cube moves backward or sideways. The backward moving model is shown in Figure 3.5. For sideways' moving (to the left or to the right), we may rotate cube around $y-axis$ for $\pm 90^{\circ}$ first, and then use forward moving model.

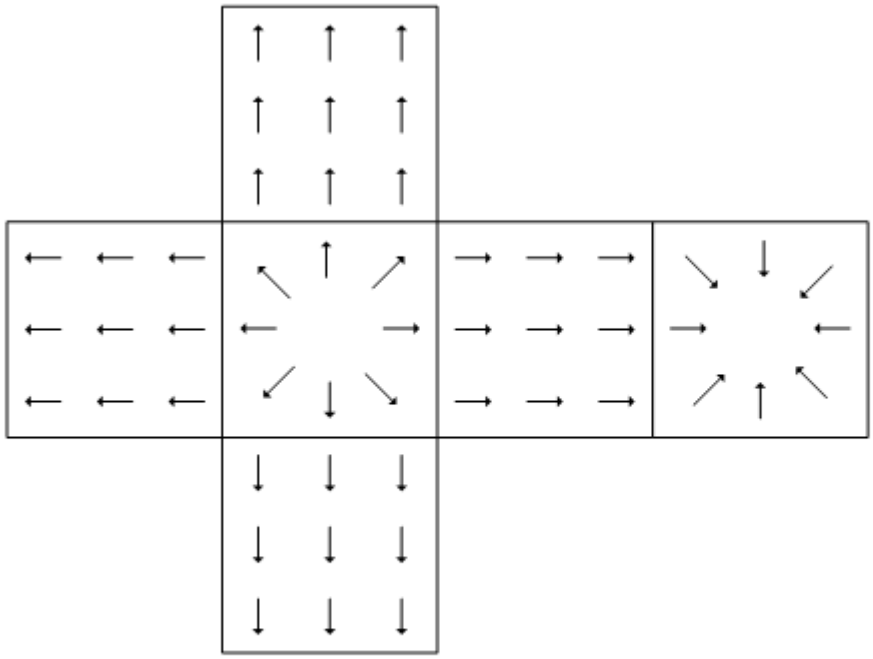


Figure 3.4: Forward moving. A cube is heading forward along $z-axis$ (see cube frame in Figure 2.1(b))

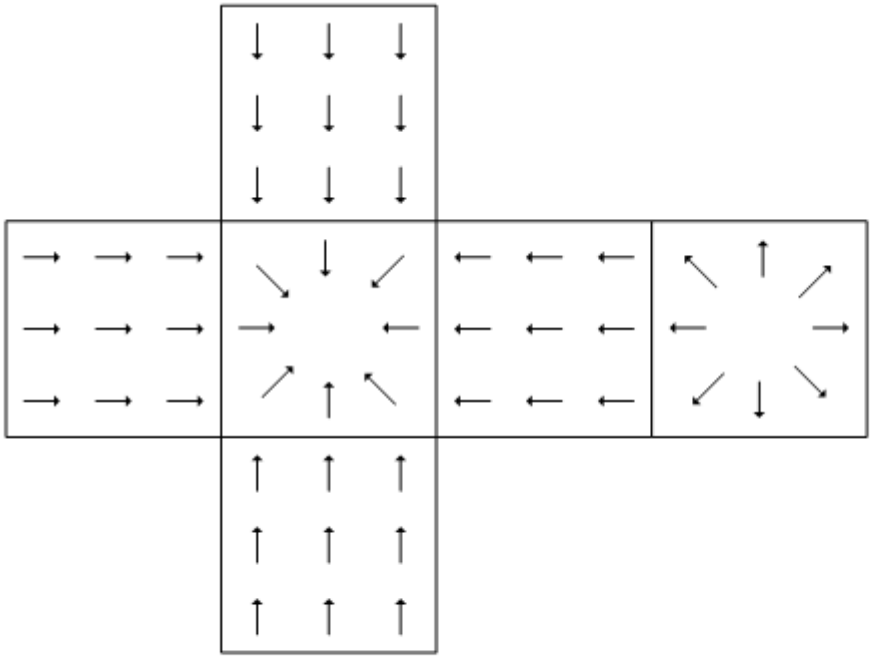
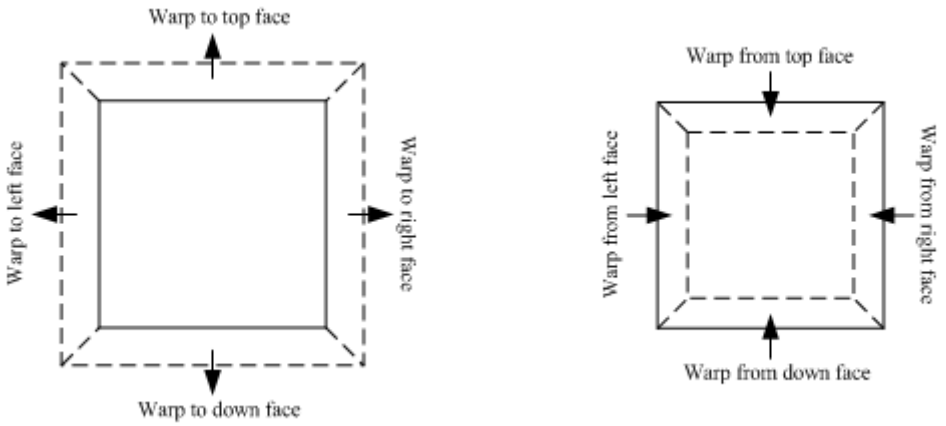


Figure 3.5: Backward moving. A cube is heading backward along minus $z-axis$



(a) Front face warping

(b) Back face warping

Figure 3.6: Cube forward warping

According to cube moving models, we can apply specific local warping to approximate transformations on cube image. The forward moving can be performed by warping six faces of cube as follows:

1. *Front face*: Zooming in (see Figure 3.6(a)) as cube moving forward. The four trapezoidal areas are magnified out of the *front face* boundary. They are warped into four rectangles and moved into *left, top, right and down face* respectively.

2. *Right face*: The face image is moving right. As pixels are moving out, the empty space of the left side is filled with a rectangle, which is warped and moved from *front face*. The right side of the *right face* is moved out of the face boundary. It is warped into a small trapezoid and relocated into right side of *back face*.

3. *Top, right and down face*: The similar transformations are applied as *front face*.

4. *Back face*: Zooming out (see Figure 3.6(b)). The empty space (four trapezoidal areas) through image reduction are filled with pixels moving from *left, top, right and down face* respectively.

3.3.2 Optical flow and warping scale

For two cubic nodes, we are interested in warping a cube into another one. A natural question is, how much scale we need to warp so that the warped cube can approximate the destination cube as closely as possible. We call this problem as *cube homing*. This is the problem similar to robot homing [3, 21, 9]. Our problem, however, is a simplified case in that we already know the heading orientation(we just have small translation involved, and no rotation). To solve cube homing problem, we adopt optical flow to compare warped cube with destination cube and minimize the difference. One of the optical flow algorithms for homing problem can be found in [56].

Pixel displacements: precise and approximate

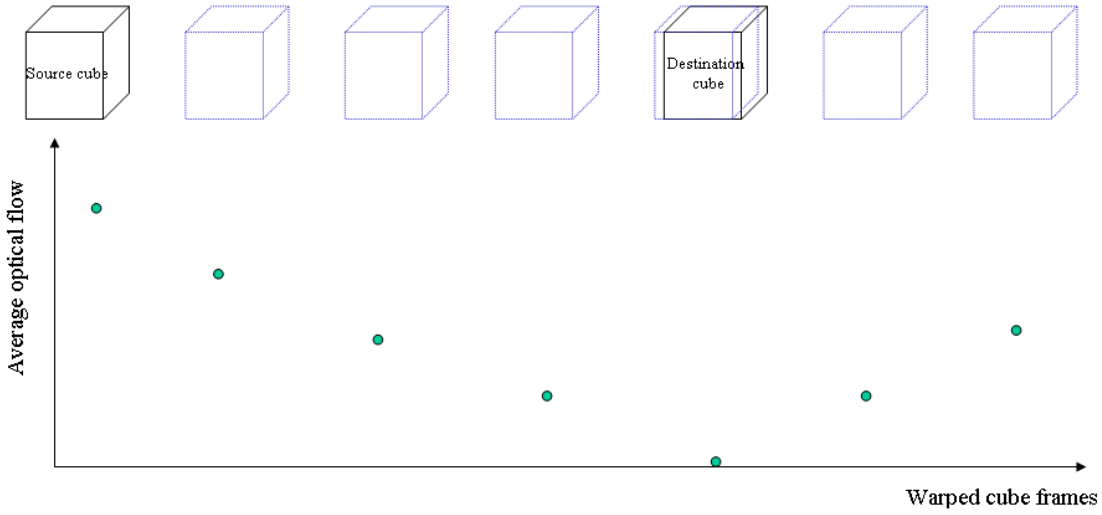


Figure 3.7: Optical flow comparison for cube warping-ideal scenario

As we walk-through from one cube to another cube(with no rotation), we approximate every very small step by warping the first cube by one or two pixels. Thus, we generate a series of warped cubes. All this cubes can be regarded as cubic frames. We compute optical flow of every cube in the frames with the destination cube. Ideally, if a warped cube matched the destination cube, the optical flow between them would be zero, as shown in Figure 3.7. However, since we just use approximate model for cube warping, the real optical flow is similar to Figure 3.8.

In fact, an precise cube warping could be applied if the 3D structure models of the environment were available. The exact optical flow of the objects in the images depends on their distance in the environment relative to the cube. Indeed, the closer the objects are to the camera, the more their pixels will move in the images. Figure 3.9 and Figure 3.10 show pixel displacements between two neighbour cubes. Because the objects (computer and monitor) are much closer to the camera of "right" face , the monitor is

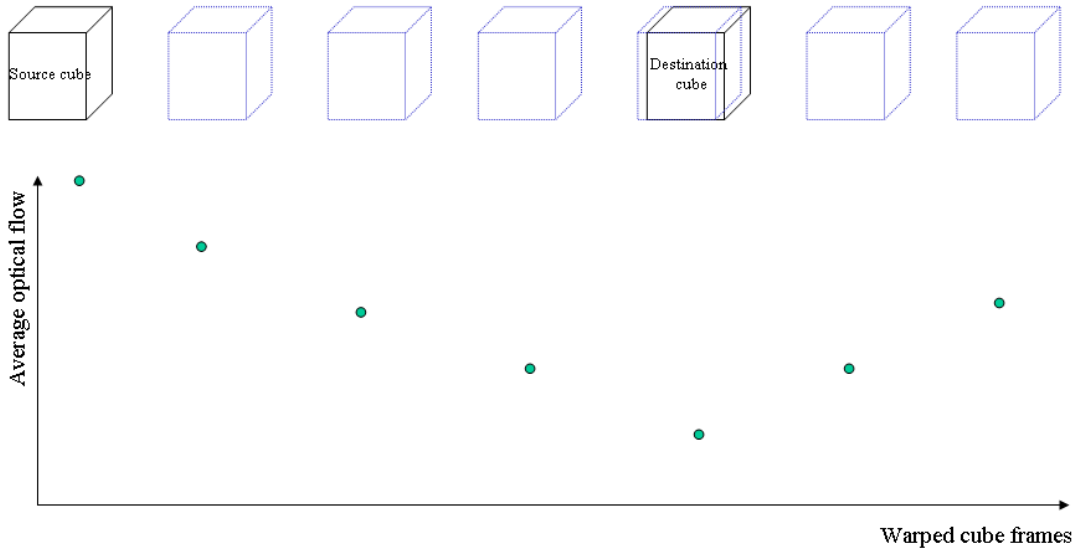


Figure 3.8: Optical flow comparison for cube warping-real scenario

dislocated much larger than other objects in the cube images, as shown in Figure 3.9. In Figure 3.10, the pixel displacements are closer in the whole image due to comparatively same distance of the objects' position relative to cameras.

As we discussed earlier in this chapter, it is both computationally expensive and too much cost on storage and network transmission to perform precise cube warping. Since the cube nodes are dense, the translation between two neighbour cubes is very small. Thus, the difference of the largest pixel displacement and the smallest one will be small. In this case, our simplified warping model can provide good approximation.

Optical flow

We use optical flow method to solve cube homing problem. Optical flow is a measurement of the local image motion based upon local derivatives between the first and the current frame. It quantifies every image pixel displacement between two image frames.

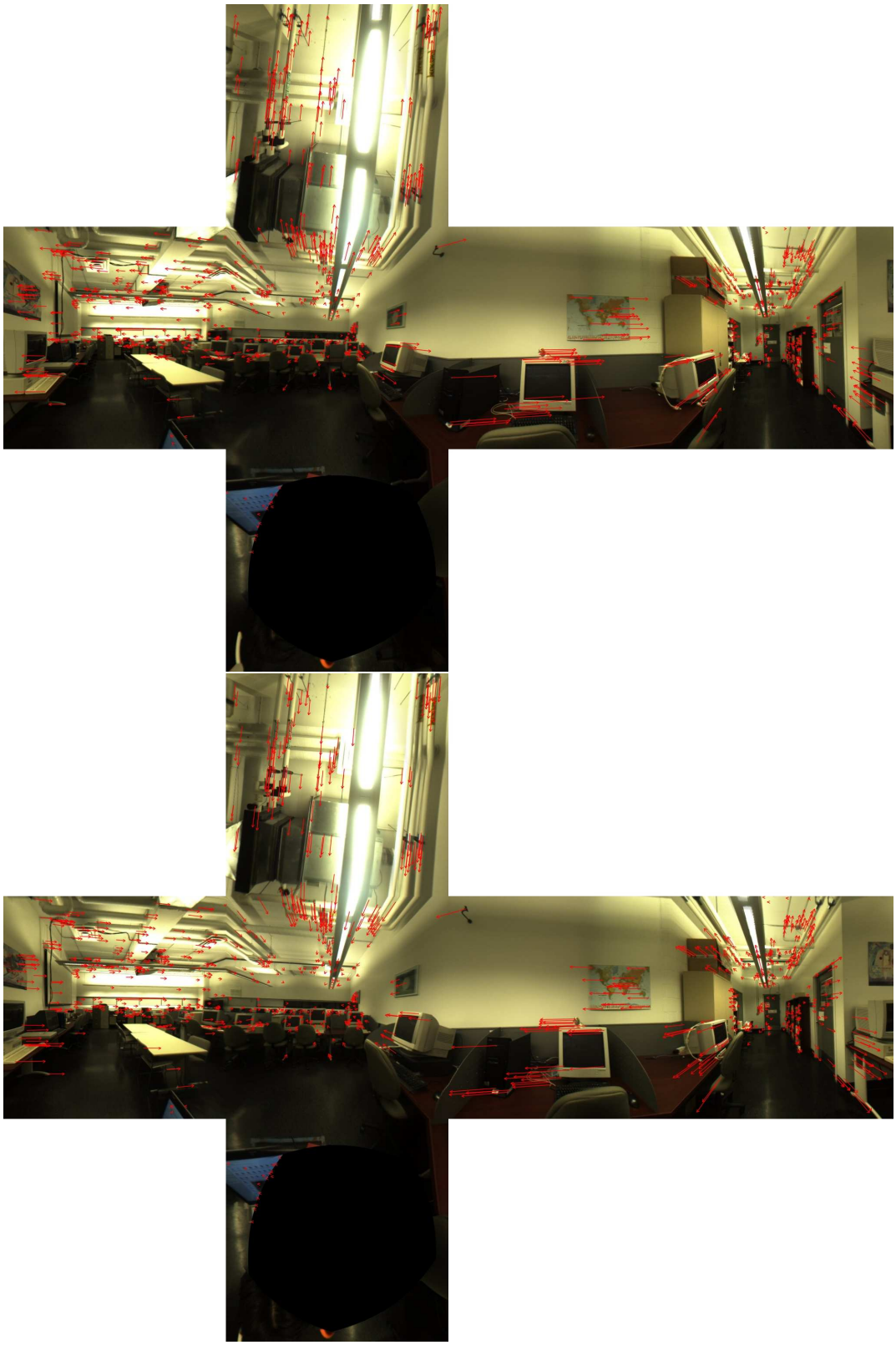


Figure 3.9: Feature displacements for two neighbour cubes: cube 1 and cube 2

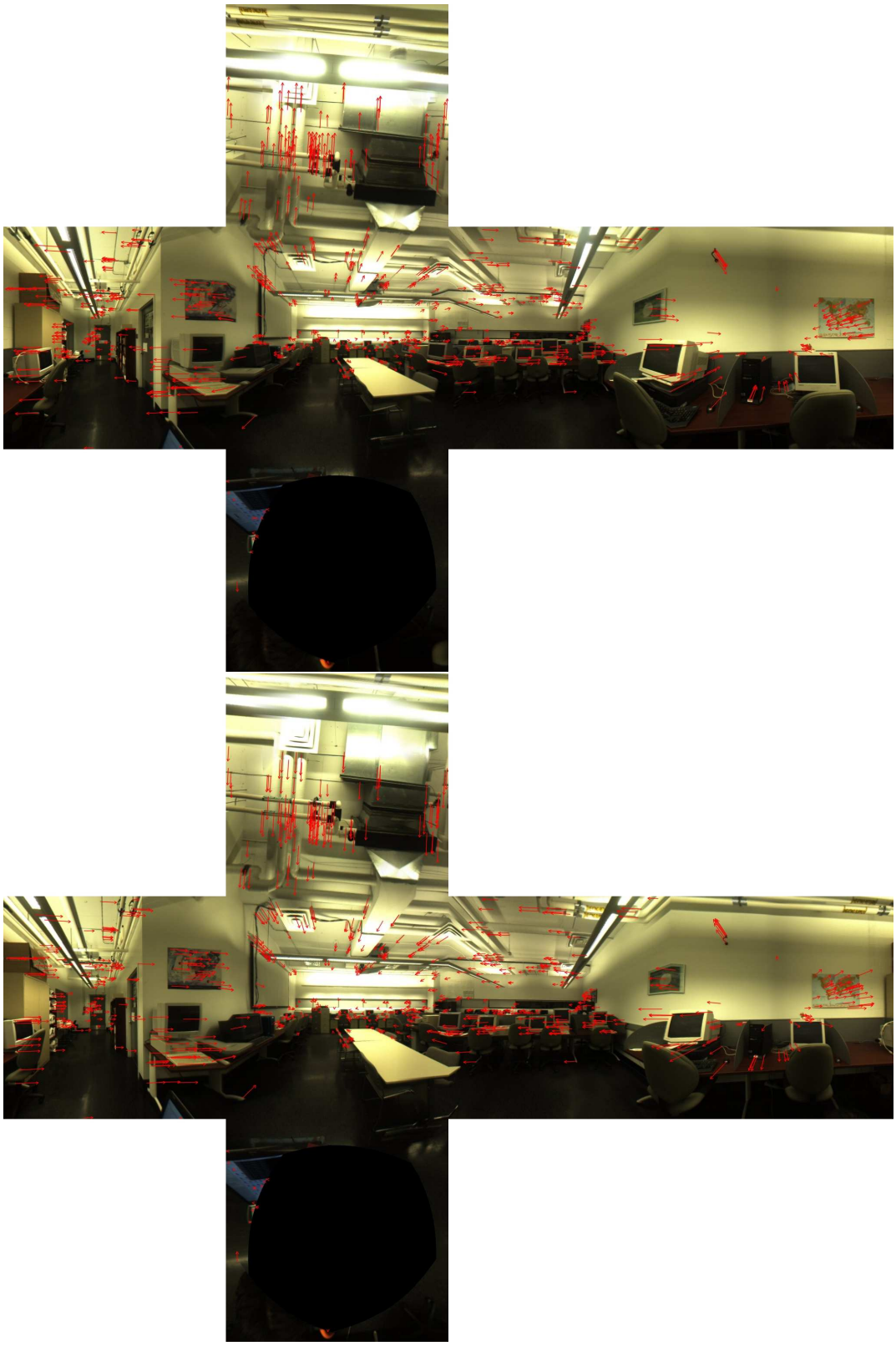


Figure 3.10: Feature displacements for two neighbour cubes: cube 3 and cube 4

There are a lot of approaches to solve optical flow problem in the literature. These approaches can be classified into several types. The first type, called *differential techniques*, compute velocity from spatiotemporal derivatives of image intensity [34, 46]. The second type of approaches use *region-based matching* to find best matches between image regions at different time [2, 63]. The third class is based on the output energy of velocity-tuned filters in the Fourier domain [32]. These approaches are called *energy-based methods* or *frequency-based methods*.

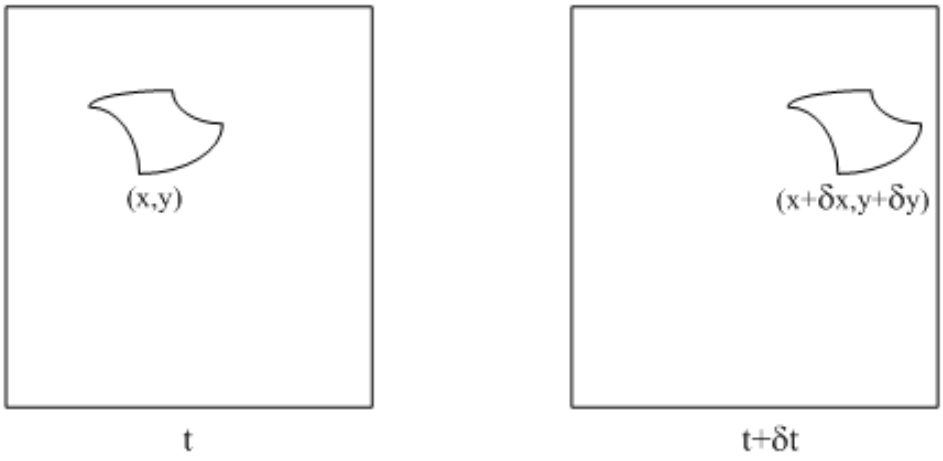


Figure 3.11: The image area at position (x, y, t) is relocated at (x+δx, y+δy, t+δt)

Generally, there are three assumptions for optical flow calculation:

- There are only rigid transformations for all objects in the scene.
- There is no occlusion involved.
- There are no illumination variations.

All these assumptions will make sure that objects in the image at time t will still be the

same in the image at time $t + \delta t$, see Figure 3.11. This can be represented as:

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \tag{3.1}$$

After performing a 1^{st} order Taylor series expansion and get rid of High Order Terms, we get:

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t \tag{3.2}$$

From Equation3.1 and Equation3.2, we have:

$$\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = 0$$

and then:

$$\frac{\partial I}{\partial x}v_x + \frac{\partial I}{\partial y}v_y + \frac{\partial I}{\partial t} = 0 \tag{3.3}$$

Here $v_x = \frac{\delta x}{\delta t}$ and $v_y = \frac{\delta y}{\delta t}$ are the x and y components of optical flow or image velocity. By using I_x, I_y and I_t to represent $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$, we finally rewrite the Equation3.3 as:

$$(I_x, I_y)\begin{pmatrix} v_x \\ v_y \end{pmatrix} = -I_t \tag{3.4}$$

more compactly as:

$$\nabla I \cdot \vec{v}^T = -I_t \tag{3.5}$$

This is a one equation with two unknowns. Therefore it does not have an exact solution. Lucas and Kanade [46] use a weighted least-square fit of local first-order cal-

culation to provide an additional constraint by minimizing the residual function ϵ in the neighborhood window. This can be represented as following equation:

$$\epsilon(v_x, v_y) = \sum_{x,y\in\Omega} W^2(x,y)[\nabla I(x,y,t)\cdot\vec{v}^T + I_t(x,y,t)]^2 \tag{3.6}$$

where Ω is a image neighborhood of size $(2\omega_x + 1)(2\omega_y + 1)$, typically, $\omega_x, \omega_y \in \{2,3,4,5,6,7\}$, and $W(x,y)$ denotes a window function that gives more influence to constraints at the centre of the neighbourhood than those at the periphery. For more mathematical expressions, readers can refer to [6].

Although proposed 20 years ago, Lucas and Kanade's approach can give very good optical flow results. In *Performance of Optical Flow Techniques* [5], Barron et al. evaluated it as: "the most reliable were the first-order, local differential method of Lucas and Kanade."

Many new approaches for computing optical flow have appeared in the literature. Shi-Tomasi's optical flow algorithm [61] extends Lucas-Kanade's Newton-Raphson style search methods to work under affine image transformations. Their method works good for relatively small feature displacements, but fails at large displacements and deteriorates for rotation on the image plane. The spline-based image registration technique [65] uses coarse-to-fine image registration strategy to track features with larger displacements. Although all the new approaches claim to have better optical flow results, often their performances are only marginally better. When the motion is mostly translational between frames and translation is small, Lucas-Kanade's method still be one of the best and probably the most efficient algorithm.

For the navigation problem, we are only concerned with linear motion with no rotation. We are considering a small number of frames at a time, and image warping due to local image plane rotation is not expected. Therefore, we use Lucas-Kanade's optical

flow algorithm to solve our cube homing problem.

Warping scale

For *cube homing* problem, our objective is to warp the source cube so that it will approximate the destination cube as close as possible. Thus, we need to decide what warping scale we should take. We are only interested in very small translation, and the smaller the cube translation is, the fewer the pixel displacement is. Therefore, we require that the maximum pixel displacement between source cube image and destination cube image should be less than 80 pixels.

For forward warping, our homing processing is as follows:

1. Set the warping step $\Delta = 1$, and initial warping scale $t = 0$.
2. Set $t = t + \Delta$, and warp the source cube for t pixel(s).
3. Compute optical flow $\vec{V_t}(i) = (V_{tx}(i), V_{ty}(i))$ between warped cube and destination cube, where $i \in [1, N]$, $N = numbers\ of\ image\ pixels$.
4. Calculate average norm of optical flow $\widetilde{\|V_t\|} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{V_{tx}(i)^2 + V_{ty}(i)^2}$.
5. Repeat step2 until $t_{max} = 90$.
6. For $t \in [1, t_{max}]$, find *homing step* $T = t$ such that $\widetilde{\|V_T\|}$ is minimum.

After having *homing step* T, we can decide warping scales t_0 of an arbitrary warping position between source cube and destination cube as a function of $s \in [0, 1]$:

$$t_0 = s \cdot T \tag{3.7}$$

Figure 3.12: Two neighbouring cube images with relatively large translation(top and bottom faces cropped)

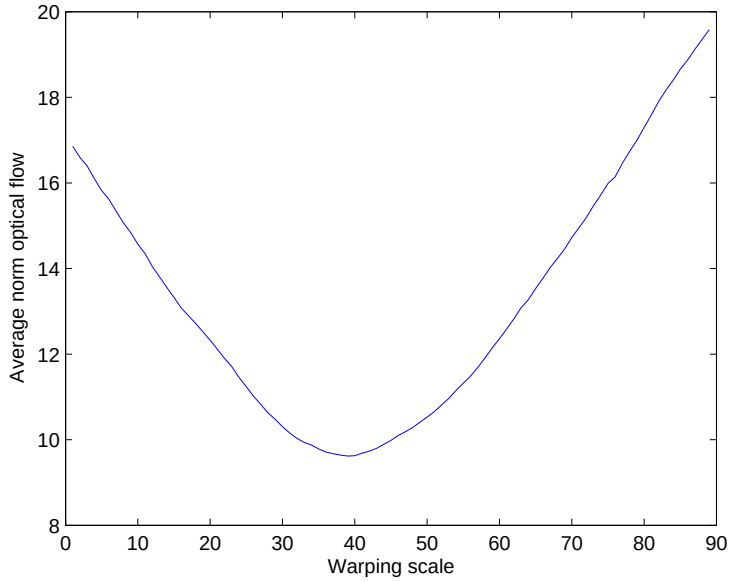


Figure 3.13: The average norm optical flow of different warping scales

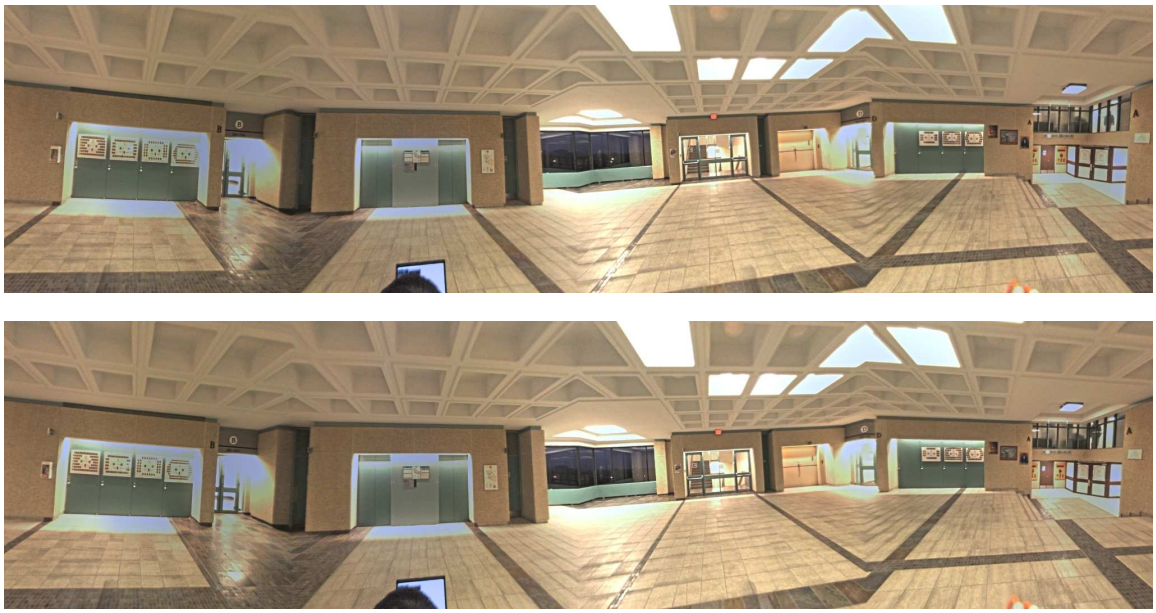


Figure 3.14: Two neighbouring cube images with relatively small translation(top and bottom faces cropped)

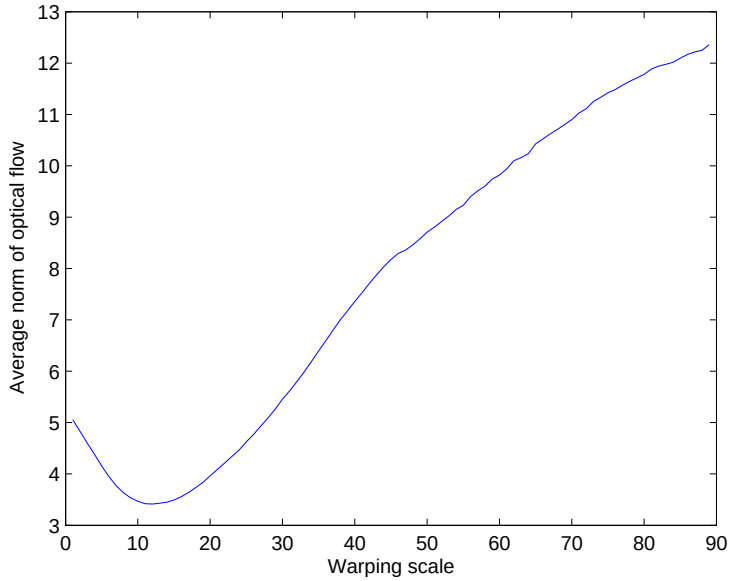


Figure 3.15: The average norm optical flow of different warping scales

Figure 3.12 are two cube images with relatively large translation (top and bottom faces cropped). Their average norm optical flows, calculated through above homing processing, are shown in Figure 3.13. As expected, this is similar to what we have analyzed before, see Figure 3.8. In this experiment, the source cube is warped one pixel at a time for 89 different warping scales. The optical flows between every warped image and destination image are computed and plotted. From the plot, we can easily find that *homing step* $T = 42$ when $\|\widetilde{V_T}\|$ is minimum.

For relatively small image translation (see Figure 3.14), Figure 3.15 shows similar result as Figure 3.13. In such case, the *homing step* T is equal to 12. Although both experiments show the same result, the minimum values of their $\|\widetilde{V_T}\|$ (average norm of optical flow) are different. For large image translation , the minimum $\|\widetilde{V_T}\|$ is equal to 9.2, whereas for small image translation, it is equal to 3.4. This result is consistent with our assumption: our cube warping model simulates the small cube translation. The smaller the cube translation is, the more accurately our warping model approximates the real cube moving.

3.3.3 Algorithm overview

Our objective is to generate novel view of an arbitrary location between two neighbouring cubes. First, let us normalize the translation between two neighbouring cubes as 1. Translation $s \in [0, 1]$ is an arbitrary location from the source cube (see Figure 3.16). As we stated previously, our warping models work well only for small cube translation. Therefore, if $s \in [0, 0.5]$, we warp source cube forward to approximate novel views. If $s \in (0.5, 1]$, we warp destination cube backward.

The algorithm for generating a novel cube view between two neighbouring cubes is as follows:

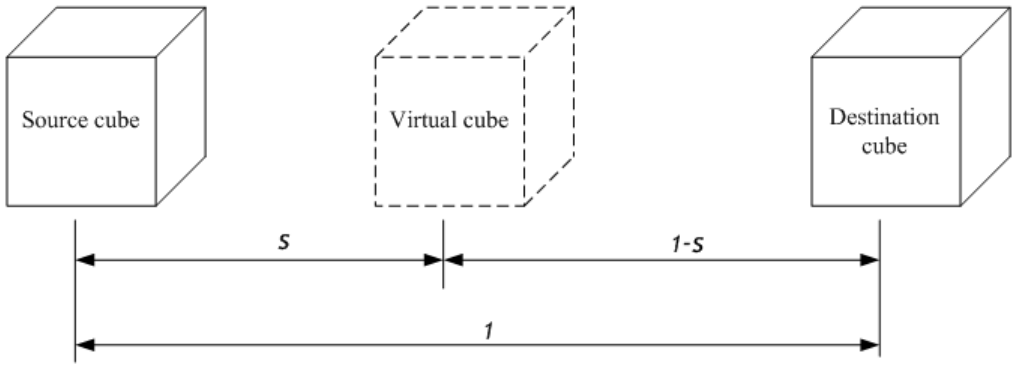


Figure 3.16: The cube distance

1. Using the previously stated *cube homing* method to compute the *homing step* T.

2. If $s \in [0, 0.5]$, warping source cube forward to approximate novel views. The warping step is: $t_0 = s \cdot T$

3. If $s \in (0.5, 1]$, warping destination cube backward to approximate novel views. The warping step is: $t_0 = (1 - s) \cdot T$

3.3.4 Algorithm cost

The computation costs and communication costs for navigation through cube warping are very low. Table 3.1 shows the computation costs of our algorithms. The cube image resolution for our experiment is: 6 faces$\times 512 \times 512$. For any two neighbour cubes, we use optical flow algorithms to compute their *warping scale*. The computation time for this process is 18.85 seconds in an AMD 64x2 1.6GHz, 1024MB memory laptop computer. Since we are only interested in how much *warping scale* we need for *cube homing*, this process can be pre-computed. Therefore, the only on-line computation is cube warping process, which generates virtual cubes and takes 0.25 seconds. Such on-line warping

operation can be processed on remote client computers.

The communication costs are shown in Table 3.2. For remote navigation through cube warping, the clients need to download our *cube warping* program (104KB) and Intel OpenCV .dll files (2.12MB). It only needs extra one byte for the transmission of pre-computed *warping scale*, which is a integer in the range of $[1, 89]$, between any two neighbour cubes. After the transmission of one cube and *warping scale* , the client computer can generate novel virtual cubes while waiting for new neighbour cube data. This actually alleviates communication burden.

Table 3.1: Computation Cost

Pre-processing	Warping scale pre-computation	18.85 seconds
On-line precessing	Cube warping running time	0.25 seconds

Table 3.2: Communication Cost

Warping applications	Warping program OpenCV .dll files	104KB 2.12MB
Warping scale between two cubes	1 byte	

3.4 Simulation results

We present here two experiment results performed from two pairs of neighbour cube images: cube5, cube6 and cube7, cube8. Figure 3.17 shows two precaptured cube images with no rotation. These two images have raltively larger translation(70 Centimeter VS 35 Centimeter) than those two images shown in Figure 3.19.

First, we use *cube homing* method (See Section 3.3.2) to compute the *homing step* T. For cube5 and cube6, the computed *homing step* $T_1 = 42$, and for cube7 and cube8, $T_2 = 12$. Then for translation $s_1 \in [0, 0.5]$ between cube5 and cube6, we forward warp cube5 with the *warping step* $t_1 = s \cdot T_1$, whereas for translation $s_1 \in (0.5, 1]$, we backward

warp cube6 with the *warping step* $t_1 = s \cdot T_1$. We perform the same experiment with the cube7 and cube8.

Figure 3.18 shows the experiment results of cube warping for cube5 and cube6. From top to bottom, the images (top and bottom faces cropped) are: cube5, forward-warped cube5 with $s_1 = 0.25$ and $t_1 = 10$, forward-warped cube5 with $s_1 = 0.5$ and $t_1 = 21$, backward-warped cube6 with $s_1 = 0.5$ and $t_1 = 21$, backward-warped cube6 with $s_1 = 0.75$ and $t_1 = 10$, and cube6. From the resulting images we can see that there are smooth transitions between cube5 and cube6. For halfway view between cube5 and cube6, the image generated from forward-warping of cube5 is very close to the image generated from backward-warping of cube6.

The experiment results of cube warping for small translation, cube7 and cube8, are shown in Figure 3.20. From top to bottom, the images (top and bottom faces cropped) are: cube7, forward-warped cube7 with $s_1 = 0.25$ and $t_1 = 3$, forward-warped cube7 with $s_1 = 0.5$ and $t_1 = 6$, backward-warped cube8 with $s_1 = 0.5$ and $t_1 = 6$, backward-warped cube8 with $s_1 = 0.75$ and $t_1 = 3$, and cube8. The resulting images show the similar results as the cube warping of cube5 and cube6. As expected, two halfway views between cube7 and cube8, one view generated from forward-warping of cube7 and another view generated from backward-warping of cube8, are closer than those generated from cube5 and cube6. This result is consistent with our previous analysis: our cube warping model works well with small cube translations.

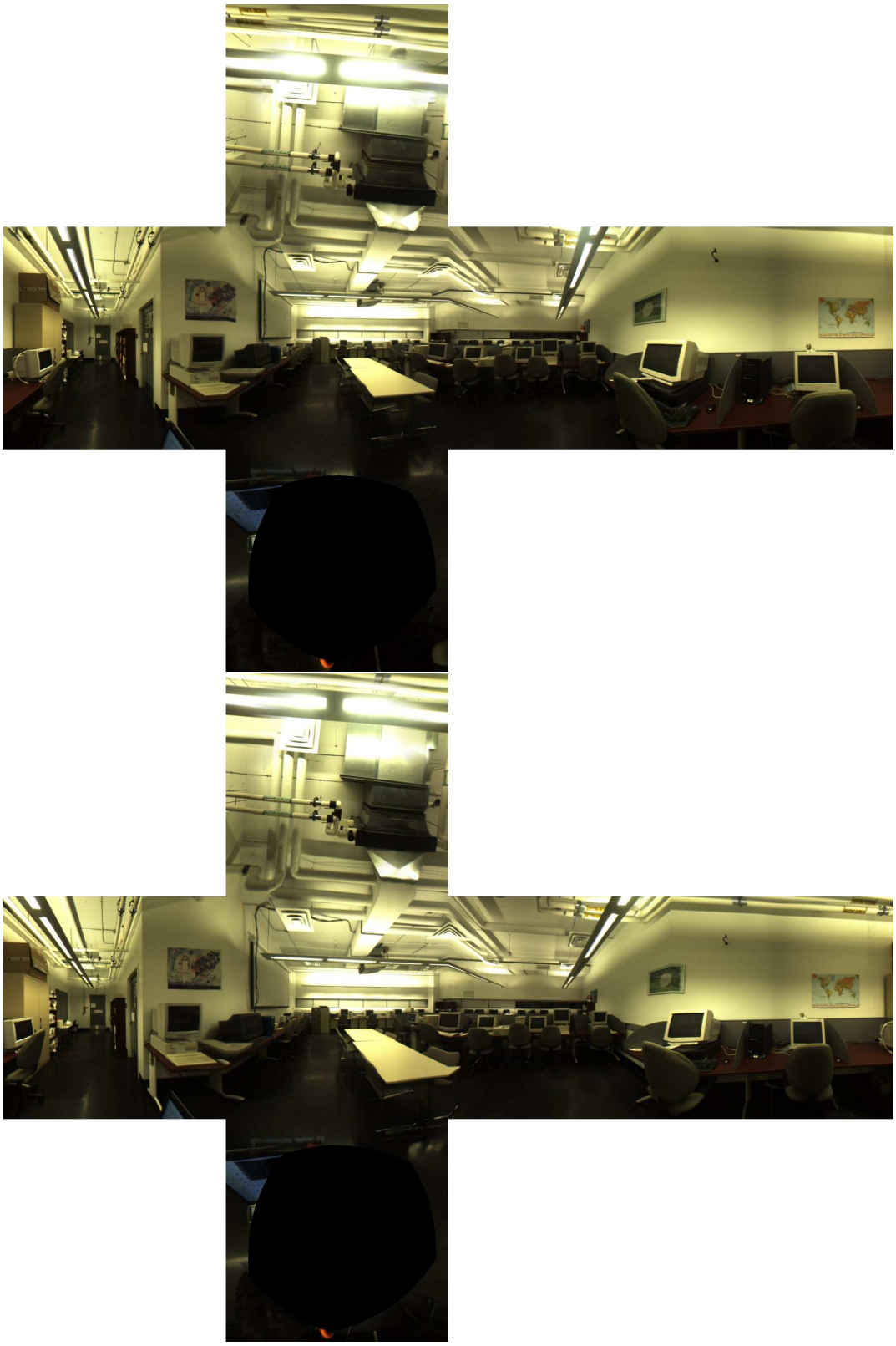


Figure 3.17: Two original cubes with relatively larger translation: cube 5 and cube 6

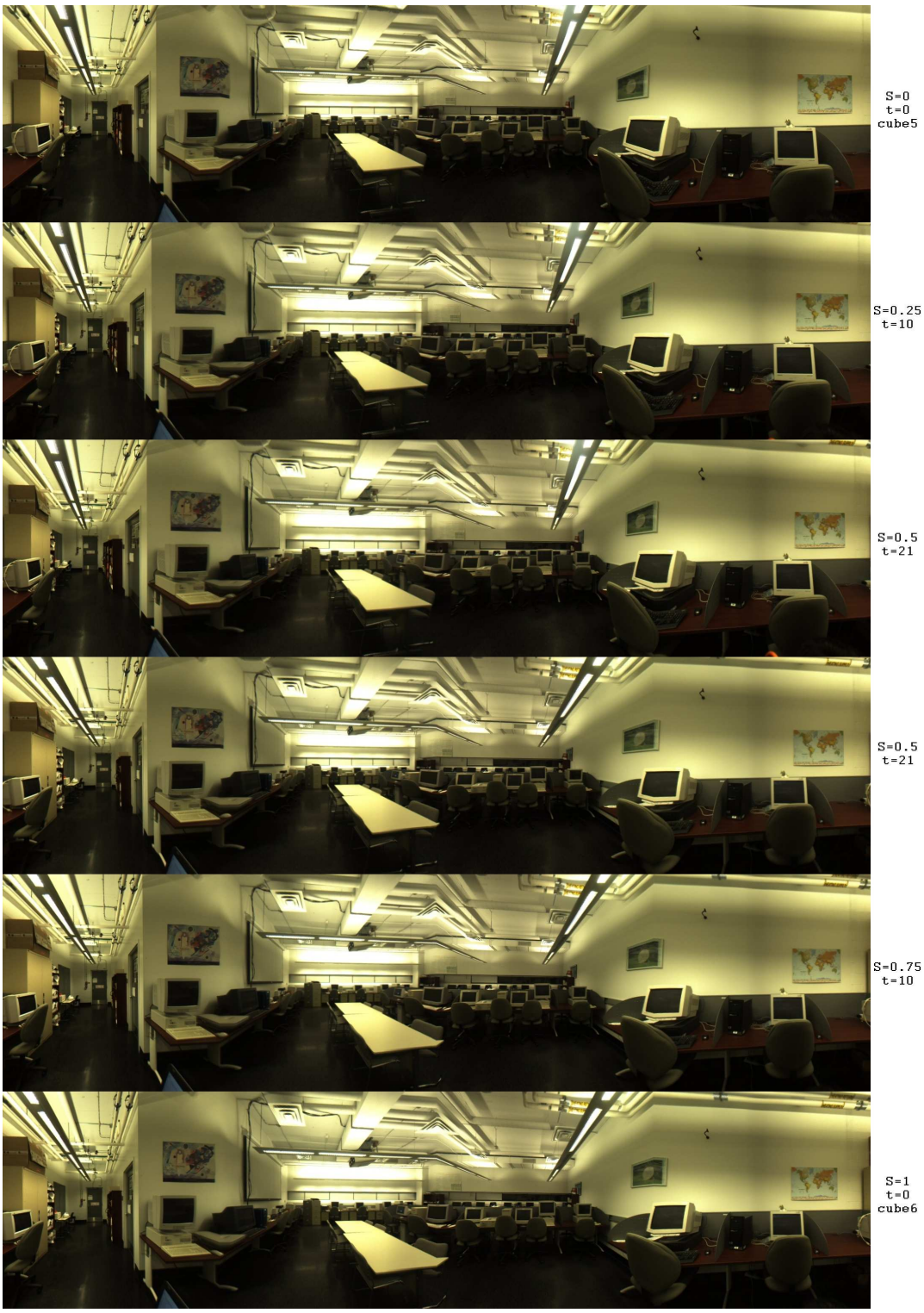


Figure 3.18: Two neighbouring cube images and their in-between warped images(top and bottom faces cropped)

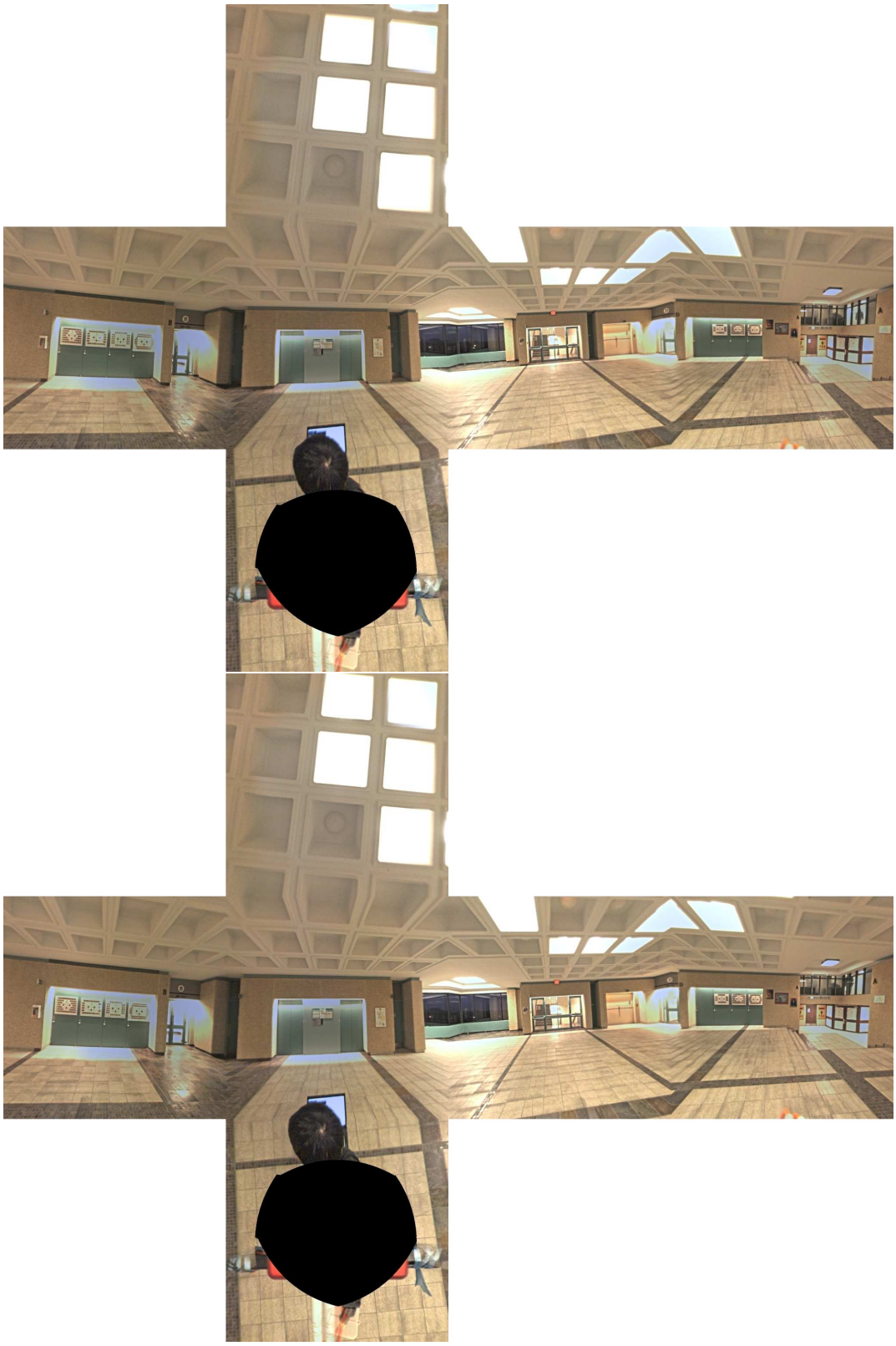


Figure 3.19: Two original cubes with relatively smaller translation: cube 7 and cube 8

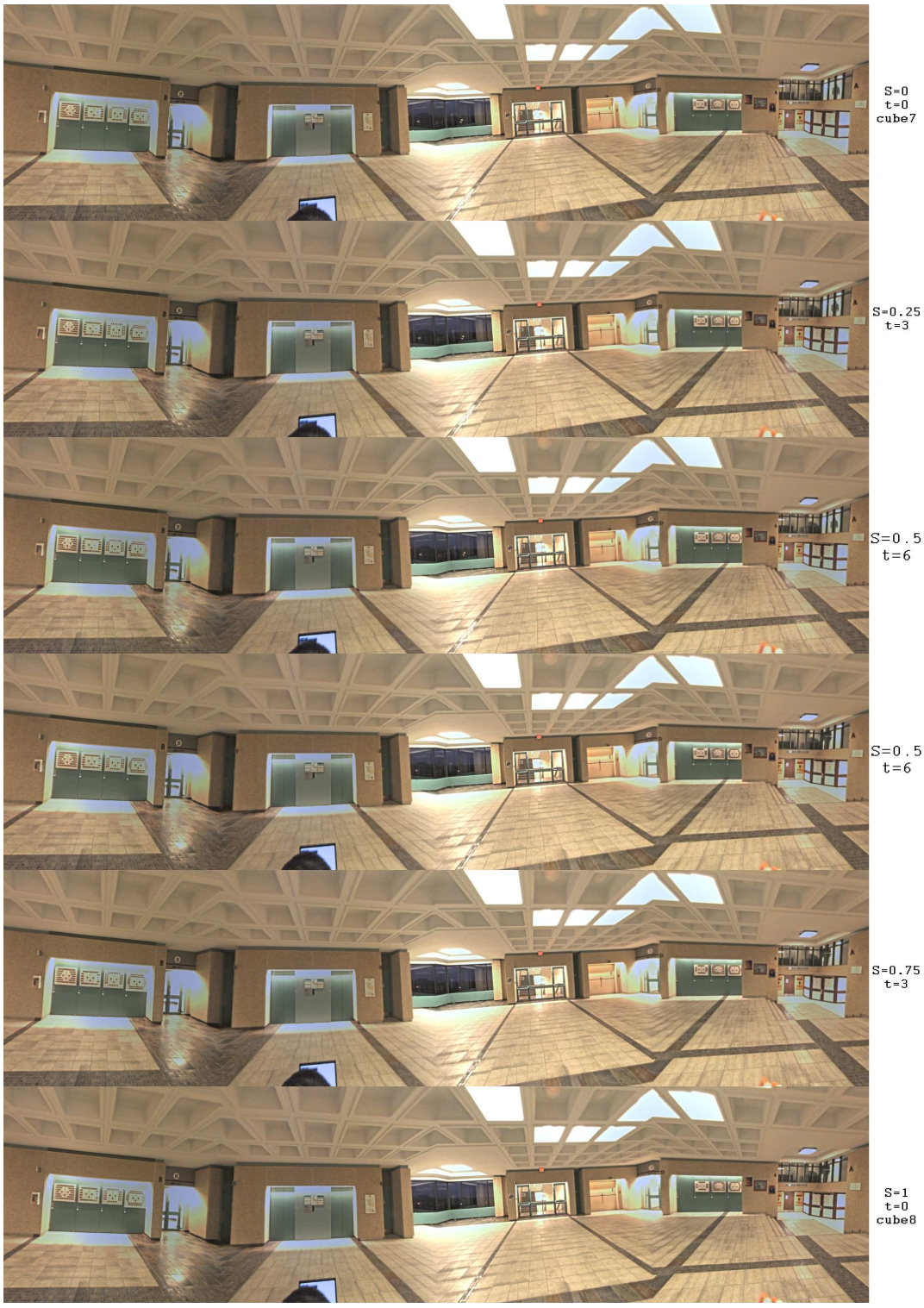


Figure 3.20: Two neighbouring cube images and their in-between warped images(top and bottom faces cropped)

3.5 Discussion and conclusion

3.5.1 Assumption

The method makes an equal distance assumption: all the objects have same distance from the location of snapshot. Such assumption has been used by others [9, 22].

At first sight, the equal distance assumption seems not to be realistic. However, if the translation is small between two neighbouring cubes, the difference of the largest pixel displacement and the smallest one will be small. Therefore, the resulting error remains small for our cube warping algorithm, which uses identical pixel displacement models to approximate cube translations.

The experiment results show that our cube warping algorithm works well under small translations. It has been observed that for translations up to 40 centimeters in normal circumstances, our method can produce smooth in-between images. However, under certain condition, we must limit the translations under 30 centimeters or less. Such condition is: some objects have very large distance from the camera whereas others have very close distance. For such condition, the difference of the largest pixel displacement and the smallest one will be relatively large. Thus, smaller translations could reduce the difference, and as a result, reduce the resulting errors.

3.5.2 Conclusion

This chapter presented a new novel view generation technique that incorporates the model of pixel displacements between aligned cubes into a cube warping model. By making our "same distance" assumption, we find that we are able to generate good in-between novel images for small cube translations.

Our method, although with simplified models, has following advantages:

- It is a fast, on-line algorithm, which can run on client computers.
- The very low computation and communication costs of the method can alleviate the burden of the high definition cube images on storage and network transmissions and loading time.
- Our pixel displacement model provide a good solution to the greatest difficulty of cubic representation, namely estimating image flow fields across the boundaries between faces and at corners.

However, our method is just an approximation of real cube walkthrough. Its applications are limited in the situation where small cube translations are involved.

Chapter 4

Cube Interpolation: Multiple-Node Navigation

4.1 Introduction

In the previous chapter, we proposed a fast, online algorithm for cube navigation. Although the computation and communication cost is very low and the resulting novel views look surprisingly realistic, its applications are limited due to the following reasons:

- Since cube warping uses only one cube to approximate walkthrough and involves no interpolation from two or more cubes, it works well only for small translation (less than 40 centimeters).
- By cube warping, it is only practical to simulate walkthrough between two cubes, in particular, with no rotation. Therefore the camera motion is restricted to a straight line.
- It is difficult, if not impossible, for cube warping to perform navigations in a global configuration, namely generating an arbitrary virtual view among a set of reference

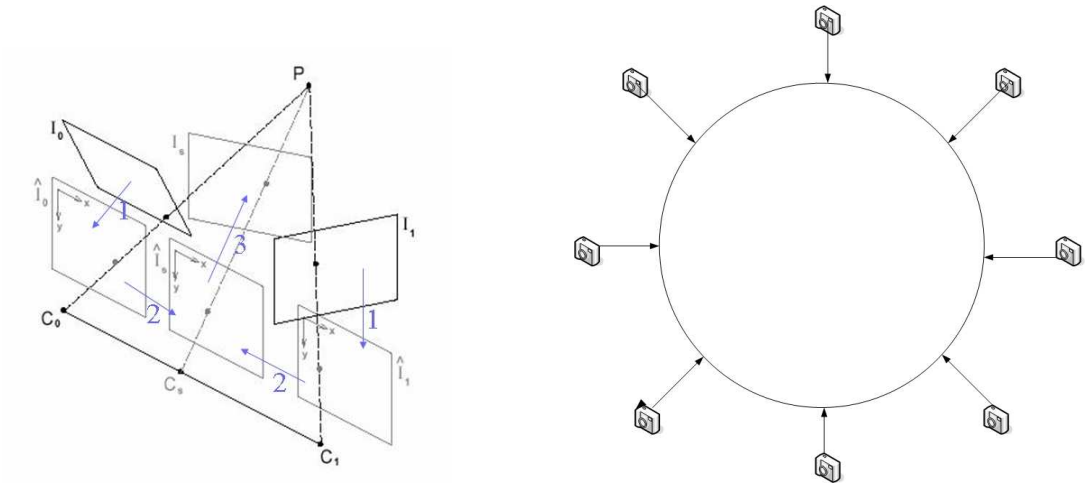
cubes.

In order to acquire seamless visualization of environment from different viewing positions and orientations, it is desirable to generate virtual images for an arbitrary position with no limitation on gaze directions given a set of reference views.

4.2 Related work

Visual navigation is an intensively researched area, and there are various image-based rendering techniques in the literature for generating novel views from sampled images. These methods can be classified into two categories: (i) viewing with restrained viewpoints; (ii) viewing with arbitrary viewpoints.

4.2.1 Viewing with restrained viewpoints



(a) View morph (reproduced from[59]) (b) MCOP (reproduced from[19])

Figure 4.1: Viewing with restrained viewpoints of two different algorithms

Image-based rendering methods can also be grouped into reconstruction-based methods and interpolation-based methods, and most interpolation-based methods are belong

to the category of "Viewing with restrained viewpoints". These methods try to generate points of lifelike novel views directly by interpolating the corresponding pixels of the input images. Even with photorealistic novel views and low computation costs, interpolation-based methods [11, 59, 42, 69] have the same limitation as our *cube warping* method in that the new viewpoint has to lie on the straight line connecting the projection centers of reference viewpoints.

One typical interpolation-based technique, Seitz and Dyer's *view morphing* [59] approach produces novel images of any viewpoint on the line linking two optical centers of the reference cameras. It employs a three-step algorithm to guarantee the intermediate view being geometrically correct or *3D shape preserving*. A *prewarp* stage is applied to rectify two reference images before intermediate *morphing* stage if they are not parallel. Finally, a *postwarp* stage is followed to neutralize the prewarping effects. Although the images produced are geometrically correct and appear strikingly lifelike, the new viewpoint is constrained to a straight line. This is shown in Figure 4.1(a). The new viewpoint is limited along the line linking two reference camera centers C_0 and C_1.

Another type of rendering methods can provide a richer user experience by allowing the observers move freely in a circular region. The *Multi-Center-Of-Projection (MCOP)* [55] samples the modeling images by placing the camera around the interested objects (shown as 4.1(b)). The virtual view is generated by interpolating these images. The navigation is restricted around a circular line with orientation of outside-in. By constraining camera movement along planar concentric circles, *Concentric Mosaics (CMs)* [62] can provide similar walkthrough experiences by compositing slit images taken at different locations of each circle.

4.2.2 Viewing with arbitrary viewpoints

Most reconstruction-based methods can provide a walkthrough experience with a user-specified rotation and translation. After a 3D reconstruction performed, it is a just a problem of projection and texture transfer to generate a novel view with an arbitrary viewpoint. Although reconstruction-based methods can be used to render nearly all viewpoints, computer vision techniques such as feature correspondence or stereo must be employed to solve the very difficult problem of acquiring dense or quasi-dense correspondence maps.

In [48], a generalized disparity value associated with each point in the reference image is computed first. This disparity value is then used to determine an image-warping equation that maps the visible points in a reference image to their correct positions in any desired view. *Layered depth images (LDI)* [60] method constructs "multiple overlapping layers" by using stereo techniques. LDI stores not only what is visible in the input image, but also what is behind the visible surface. To rendering an arbitrary novel view, it is only need to warp a single image, in which each pixel consists of a list of depth and color values.

Plenoptic modeling [49] can allow rendering from arbitrary viewpoints without explicit 3D reconstruction. After cylindrical panoramas composed, it computes stereo disparities between cylinder pairs, and then project disparity values to an arbitrary viewpoint. Dornaika [15] applies the invariance of the parallax field to achieve a user-specified view synthesis. His approach recovers the parallel field from the computed dense or quasi-dense correspondences first. Then the invariance of the parallax field is exploited to warp the reference image into novel view. Laveau and Faugeras [40] employ the epipolar constraints to perform a raytracing like search of corresponding pixels in reference images for novel view pixels. The dense correspondences are also computed in

their approach.

All these methods, with 3D reconstruction explicitly or implicitly, can synthesize novel view with an arbitrary rotation and translation. However, they all require solving the difficult dense or quasi-dense feature correspondence problem to extract disparity or depth values. Dense correspondences are hard to compute automatically, especially when the reference images have large difference in rotation and scale due to viewing orientations and zooming or large baseline separations. Sometimes it is almost impossible to compute dense correspondences due to the lack of texture regions in real image.

As for cubic panoramas, the correspondence problems are even more significant. While cubes can be easily stored and rendered by computers, the greatest difficulty of cubic representation is estimating image flow fields across the boundaries between faces and at corners. Therefore, we need to find a new algorithm to provide unconstrained cubic navigations without the computation of dense correspondences.

Our approach is similar to that of [40] in that we use a ray-tracing like algorithm. For every pixel in the new target image, a search is performed to locate the corresponding pixels in reference images. However, instead of computing dense correspondences, we use *colour invariance constraints* to guide the searches. In addition, we use multiple cubic panoramas for more accurate sence reconstruction.

4.3 The algorithm

In this section, we give details of the principle of the approach.

4.3.1 Basic idea

Our goal is to generate the photorealistic novel cubic views for virtual navigation. Given a set of precaptured cubic panoramas, we are interested in yielding arbitrary novel views so that we can achieve seamless visualization of environment from arbitrary viewing positions and orientations.

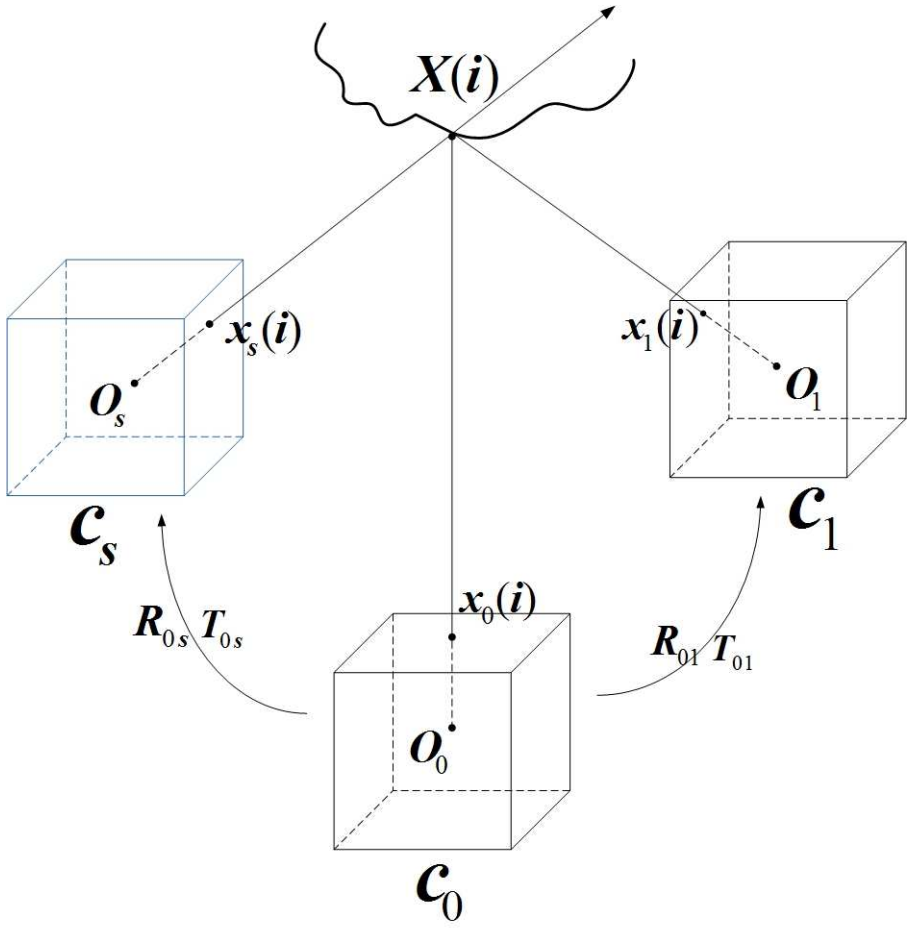


Figure 4.2: Novel view generation: a raytracing like approach

Our approach is straightforward and well-known in image synthesis: the scanning must take place directly in the new target image. The raytracing like methods follow the optical ray coming from every pixel of target view into the world instead of projecting a

world point onto the image. For every pixel in the target image, a search is performed to locate the corresponding pixels in reference images. The search is guided by colour invariance constraints.

Let us consider the example of the case where two cubes are available (see Figure 4.2). We have two reference cubes, cube C_0 and cube C_1, and the world frame is attached to the frame of cube C_0. The Euclidean transformation between the world (cube C_0) and C_1 coordinate frames is: $\mathbf{R}_{01}$ and $\mathbf{T}_{01}$. We need to generate the novel view C_s, which has an Euclidean transformation of $\mathbf{R}_{0s}$ and $\mathbf{T}_{0s}$ from the world frame. Our method is as follows:

For an image pixel $\mathbf{x}_s(i)$ of target cube C_s, we trace the optical ray $\mathbf{O}_s\mathbf{x}_s(i)$ into the world in order to find 3D point $\mathbf{X}(i)$, the intersection of the ray with the objects of the environment. If no occlusion involved (often the case for cubic panoramas), $\mathbf{x}_0(i)$ and $\mathbf{x}_1(i)$, the projection of the 3D point $\mathbf{X}(i)$ into cube C_0 and cube C_1 respectively, should have same colours. This colour consistency information is used to guide the search for the depth value of the 3D point $\mathbf{X}(i)$.

4.3.2 Choosing an arbitrary novel view

To set up the novel view, we must choose the new viewpoint and the new retinal plane first. For cubic panoramas, the process of choosing the new viewpoint and the new retinal plane is simple and intuitive. As noted before, a cubic panorama has six non-overlapping identical faces. Each face may be regarded as a image plane of a standard pinhole camera with 90° field of view, and all the cameras which take the six face images are identical and centered at the same optical center, which is also the cube center. This means the new retinal plane is always fixed with the six faces of the cube at the new viewpoint. Therefore, we only need to set up the new viewpoint and cube orientation. This can

be easily decided by its *rotation matrix* $\mathbf{R}$ and *translation vector* $\mathbf{t}$ related to the world frame, which is often attached with one of the reference cubes.

4.3.3 Colour consistency

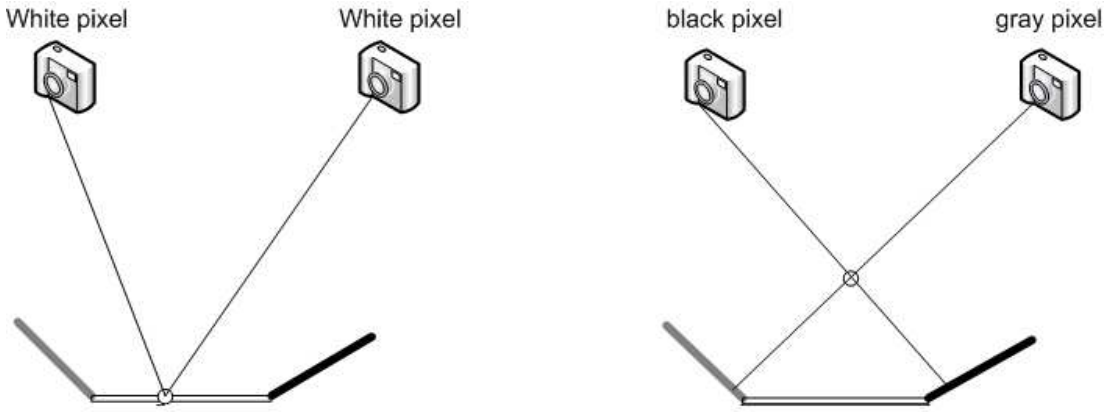


Figure 4.3: Colour consistency (reproduced from [64]): On the left, the pixels from two images have the same colours at a point on a surface. On the right, the pixels from two images show inconsistent colours at a point not on the same surface.

Colour consistency, introduced by Seitz et al.[58], is widely used in the techniques for volumetric scene reconstruction [58, 14, 37, 16, 39]. As shown in Figure 4.3, it is used to differentiate surface points from others in a scene. It is assumed that the scene is completely composed of rigid Lambertian surfaces under constant illumination. If two pixels show inconsistent colours, they must be projected from different scene points.

The colour consistency of a set of pixels can be defined as the maximum of absolute difference of colour channels between all pair of pixels [16]. Let $\mathbf{X}$ be a 3D point, and $R_i(\mathbf{X})$, $G_i(\mathbf{X})$, $B_i(\mathbf{X})$ are the three colour channels of visual information at the projection of $\mathbf{X}$ on view i, we have:

$$|\gamma_i R_i(\mathbf{X}) - \gamma_j R_j(\mathbf{X})| + |\gamma_i G_i(\mathbf{X}) - \gamma_j G_j(\mathbf{X})| + |\gamma_i B_i(\mathbf{X}) - \gamma_j B_j(\mathbf{X})| < \Theta \qquad (4.1)$$

with

$$\gamma_i = 1/\left(R_i(\mathbf{X}) + G_i(\mathbf{X}) + B_i(\mathbf{X})\right) \tag{4.2}$$

The threshold Θ is applied to decided if the pixels are the projection of the same scene point $\mathbf{X}$. To reduce the effects of the illumination variations, the components of chromaticity are used in Equation 4.1.

Another method to measure the colour consistency of a given 3D point is by computing standard deviation of its projected pixel colours[54, 39]. Let $\bar{R}(\mathbf{X})$, $\bar{G}(\mathbf{X})$, $\bar{B}(\mathbf{X})$ be the three colour channels of visual information at $\mathbf{X}$ averaged over n views, we can compute the deviation of view i over this average as:

$$d_i(\mathbf{X}) = \sqrt{\left(R_i(\mathbf{X}) - \bar{R}(\mathbf{X})\right)^2 + \left(G_i(\mathbf{X}) - \bar{G}(\mathbf{X})\right)^2 + \left(B_i(\mathbf{X}) - \bar{B}(\mathbf{X})\right)^2} \tag{4.3}$$

This deviation will be low if all the cameras see the same surfaces point $\mathbf{X}$. Otherwise, cameras viewing different points of the scene surface will result in a large deviation.

4.3.4 Implementation 1: Brute-force depth searching

Our first approach is trying to find depth information with exhaustive searching. Given a image point of a novel cube, we transform it into a 3D vector $\mathbf{x}_s(i)$ in the cube face first (For details, please refer to Appendix C). As shown in Figure 4.2, we trace the optical ray $\mathbf{O}_s\mathbf{x}_s(i)$ into the world so that the ray intersects with the objects of the environment at 3D point $\mathbf{X}(i)$. This 3D point can be expressed as:

$$\mathbf{X}(i) = \lambda_s(i)\,\mathbf{x}_s(i) \tag{4.4}$$

Where $\lambda_s(i)$ is depth (up to scale) of 3D point $\mathbf{X}(i)$.

The brute-force depth searching algorithm is as follows:

`For` $\lambda_s(i) = \lambda_{min}$; $\lambda_s(i) < \lambda_{max}$; $\lambda_s(i) = \lambda_s(i) + step$;

`compute` $\mathbf{X}(i) = \lambda_s(i)\,\mathbf{x}_s(i)$

`project` $\mathbf{X}(i)$ `to cube` C_0, C_1,...,C_m, `compute` $d_k(\mathbf{X}(i))$ `with Equation`(4.3)

`if` $\sum d_k(\mathbf{X}(i)) <$ `threshold, then` $\lambda_s(i)$ `is right depth, break`

`else choose the pixels with minimal` $\sum d_k(\mathbf{X}(i))$

Ideally, $\sum d_k(\mathbf{X}(i)) = 0$, $\lambda_{min} = 0$, $\lambda_{max} = \infty$, and `step` should be set to an infinitly small increment. However, according to our experiments, if we set $\lambda_{min} = 0.0005$, $\lambda_{max} = 2$ (the depth is up to scale) and `step` $= 0.0002$, the true depth will fall within the searching range.

After the depth $\lambda_s(i)$ is found, we can project the $\mathbf{X}(i) = \lambda_s(i)\,\mathbf{x}_s(i)$ to all the input cubes in the set. This will result in a set of projected image points, namely, $x_0(i)$, $x_1(i)$, ... ,$x_m(i)$ of input cubes C_0, C_1, ... ,C_m, respectively. Then the colour values of novel view pixel $\mathbf{x}_s(i)$ can be interpolated with the input image points $x_0(i)$, $x_1(i)$, ... ,$x_m(i)$.

To illustrate our brute-force depth searching method, we performed a simlulaton test and the results are shown in Figure 4.4. In our test, we used five cubes: four cubes as input cubes and one cube as assumed "virtual" cube. We extracted their *rotation matrixs* and *translation vectors* pairwise first. After that a image point, marked with a cross in the figure, is chosen from "virtual" cube. Then the brute-force searching is performed and the searching ranges are shown with a searching line in all four input cubes of the set.

As shown in Figure 4.4, the ranges of brute-force searching are very broad. This is especially true when the objects are close to cameras, such as the searching ranges of cube *a* and cube *b* in the figure. The broader the searching ranges are, the more

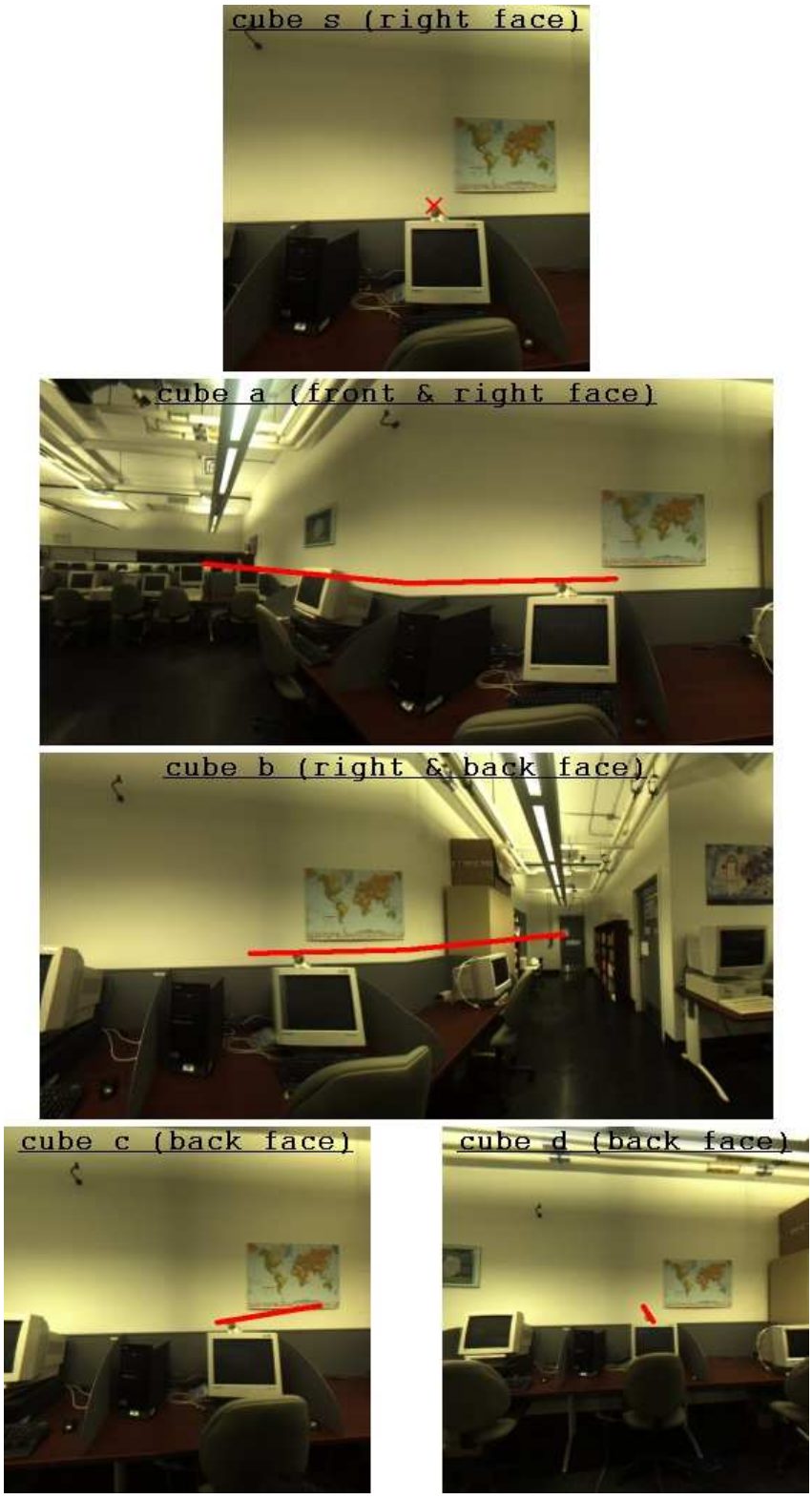


Figure 4.4: Brute-force depth searching: A point is chosen from "virtual" cube *s*. Then the brute-force searching is performed and the searching ranges are shown in all input cubes of the set: cube **a**, cube **b**, cube **c** and cube **d**

"same" colour pixels there are in the searching ranges of input cubes, and the higher the probability is to find the wrong pixels in the input cubic images.

Another problem with brute-force depth searching is its extravagant computational costs. With brute-force searching, it takes several seconds to generate a depth value for only one pixel of novel view. This is apparently impractical to generate a novel image with 2048*1536 pixels.

4.3.5 Implementation 2: Depth searching guided with sparse reconstruction

As mentioned above, the method with *brute-force depth searching* partly failed to generate good results. This blind searching has a very high computational cost and often ends up with finding the wrong pixels in the input cubic images. Therefore, we need to narrow down the searching range to speed up the searching process and reduce the chance of getting wrong depth results.

We improve our method by guiding the searching with sparse reconstruction. Firstly, we find sparse feature matches pairwise from the input cubes of the set. Then we construct the 3D points with the matches. In order to get depth information as dense as possible, we found an augmentation method by transferring the constructed 3D points from all the input cubes into one cube frame and computing the depth values from these 3D points. These computed depth information can be transformed into the novel view frame and used to guide the searching for right depth. These procedures are shown in Figure 4.5.

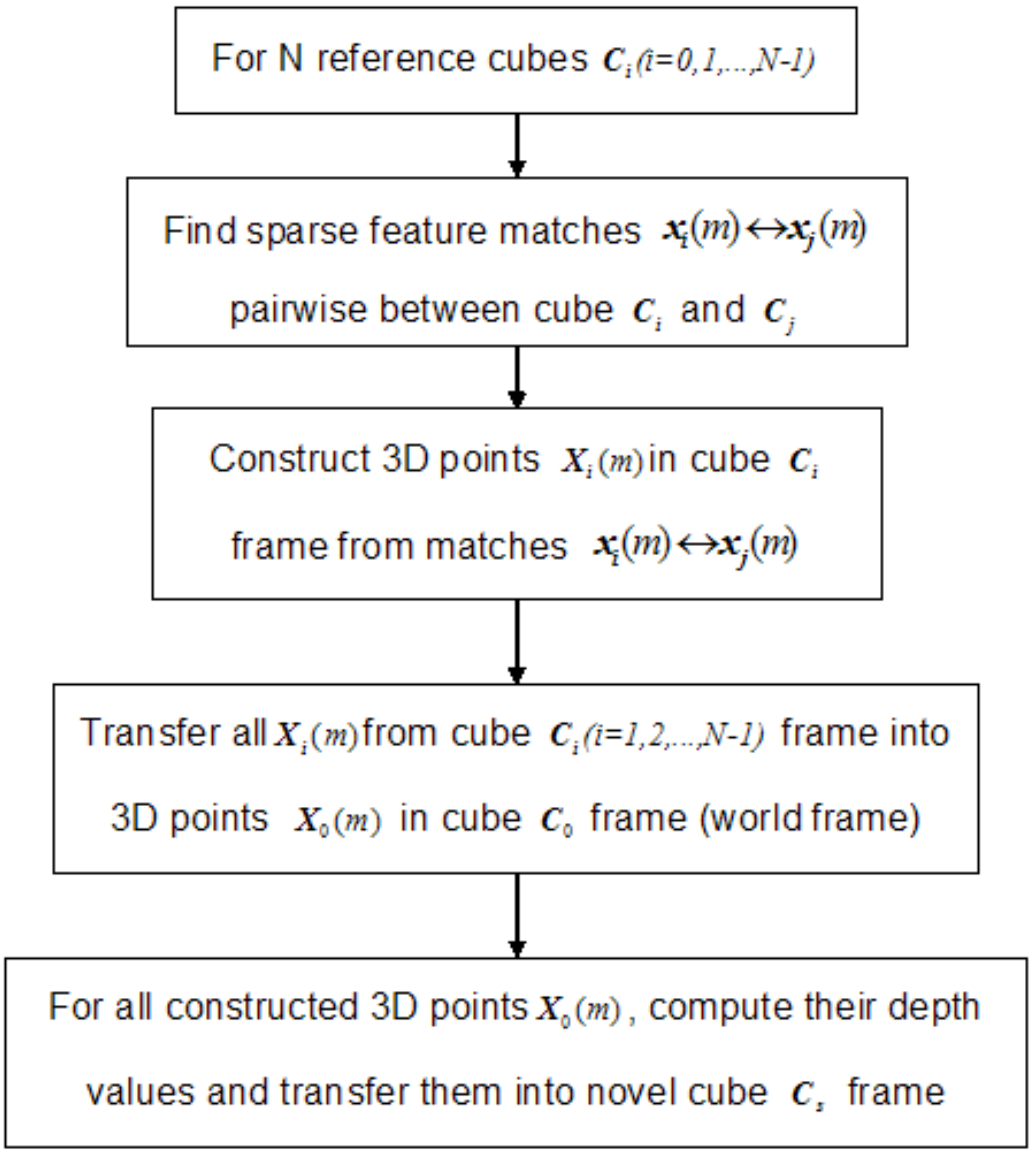


Figure 4.5: Guided depth searching with sparse 3D reconstruction

Sparse reconstruction and its augmentation

For guided depth searching, we need to compute feature correspondences first, the denser the better. Since dense and precise feature-based reconstruction is very difficult to acquire, we use sparse feature matches to compute sparse reconstruction and apply *transfer* techniques (See Section 2.5) to augment sparse reconstruction.

For a set of cubes, we attached the world frame with the frame of cube C_0. The Euclidean transformation from the the frame of cube C_i into the that of cube C_j is denoted as $\mathbf{R}_{ji}$, $\mathbf{t}_{ji}$. We also express the depth information of the j^{th} pixel ($\mathbf{x}(j)$) of cube C_i as $\lambda_i(j)$, or more concisely as $\lambda(i)$ if there is no confusion involved. The 3D object point related to depth $\lambda_i(j)$ is represented as $\mathbf{X}_i(j)$.

Our sparse reconstruction and its augmentation can be computed as follows:

- Use the method of Chapter 2 to find matches and compute $\mathbf{R}_{ij}$, $\mathbf{t}_{ij}$ pairwise for any two cubes of the set.

- Given $\mathbf{x}_j(i) \longleftrightarrow \mathbf{x}_k(i)$, the i^{th} correspondence of cube C_j with the cube C_k in the set, triangulate $\mathbf{X}_j(i)$, the 3D point (up to scale) in the frame of cube C_j.

- Transfer all computed 3D points $\mathbf{X}_j(i)$ of cube C_j frame into cube C_0 frame (world frame) with following equation:

$$\mathbf{X}_0(i) = \mathbf{R}_{0i}\mathbf{X}_j(i) + \mathbf{t}_{0i}$$

- Get rid of same 3D points from $\mathbf{X}_0(i)$, then compute the depth values $\lambda_0(i)$ of all the 3D points from $\mathbf{X}_0(i)$ with following steps:

  1. Intersect 3D point $\mathbf{X}_0(i) = (X_0(i), Y_0(i), Z_0(i), 1)^T$ with cube C_0 to acquire $\mathbf{P}_0(i) = (x_0(i), y_0(i), z_0(i), 1)^T$, the 3D point on the face (image plane) of

cube C_0. For details, please refer to Appendix B.

2. Compute the depth value as: $\lambda_0(i) = X_0(i)\,/x_0(i) = Y_0(i)\,/y_0(i) = Z_0(i)\,/z_0(i)$

Guided depth searching

After we found relatively dense image points $\mathbf{x}_0(i)$ and their depths $\lambda_0(i)$ of one cube (often attached with the world frame), we can project these points and depths to the novel view cube C_s, and obtain a set of image points $\tilde{\mathbf{x}}_s(i)$ and their depths $\tilde{\lambda}_s(i)$ in cube C_s. Since these points are relatively dense and are often feature points of edges and corners, their depths can be used to guide the searching for right depths of other image points near them.

Referring to Figure 4.2, the guided depth searching algorithm is as follows:

`project` $\mathbf{X}_0(i) = \lambda_0(i)\,\mathbf{x}_0(i)$ `to novel view` C_s; `get` $\tilde{\mathbf{x}}_s(i)$ `and` $\tilde{\lambda}_s(i)$

`For` $i = 1$; $i \leq$ `number of` $\tilde{\mathbf{x}}_s(i)$; $i = i + 1$;

`let` $\lambda_s(i) = \tilde{\lambda}_s(i)$, `compute` $\mathbf{X}(i) = \lambda_s(i)\,\mathbf{x}_s(i)$

`project` $\mathbf{X}(i)$ `to cube` C_0, C_1,...,C_m, `compute` $d_k(\mathbf{X}(i))$ `with Equation`(4.3)

`if` $\sum d_k(\mathbf{X}(i)) <$ `threshold, then` $\lambda_s(i)$ `is right depth, break`

`else choose the pixels with minimal` $\sum d_k(\mathbf{X}(i))$

To further narrow down the searching range, we can sort these image points and their depths into six groups according to which cube face they belong to. Then the searching are performed within six different faces of the novel cube, guided with corresponding depth information.

A similar experiment was performed to illustrate our guided depth searching approach. For comparison, the experiment was set up with the same cube sets and procedures as the *brute-force depth searching*. The only different is that the searching is

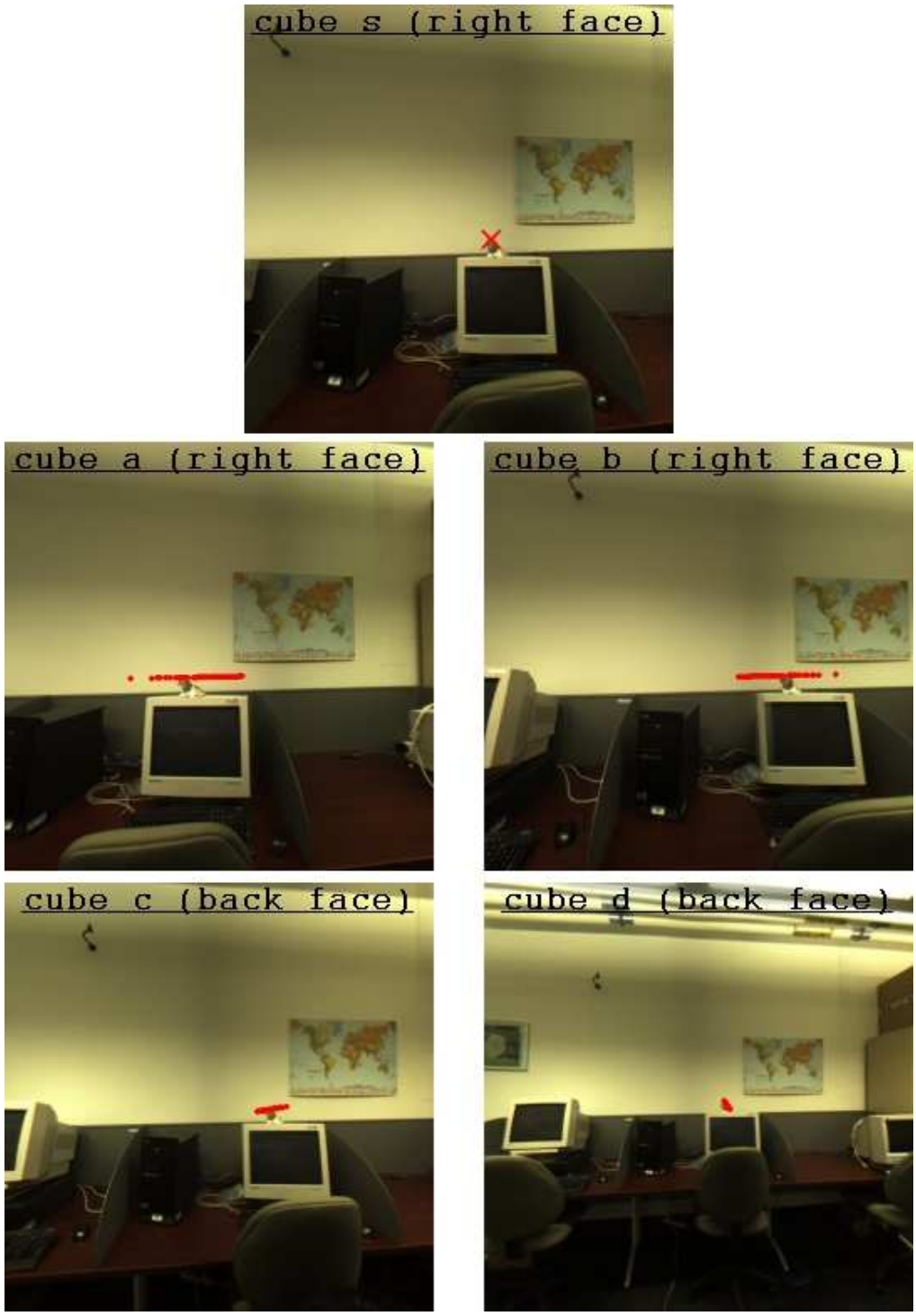


Figure 4.6: Guided depth searching: A point is chosen from "virtual" cube *s*. Then the guided searching is performed and the searching ranges are shown in all input cubes of the set: cube **a**, cube **b**, cube **c** and cube **d**

guided with known depth information.

The simulation results are shown in Figure 4.6. Compared with Figure 4.4, the searching ranges are far smaller than that of the *brute-force depth searching*. Therefore, the searching guided with computed depth information can reduce the chance of getting wrong depth results.

The *guided depth searching* also improve the searching speed dramatically. To generate novel cube view from 4 input cubes, the computation time for the two depth searching methods is shown in Table 4.1. The cube image resolution for our experiment is: 6 faces$\times 512 \times 512$, and the experiment is processed in an AMD 64x2 1.6GHz, 1024MB memory laptop computer.

Table 4.1: Computation costs of two depth searching methods

Method	*Brute-force depth searching*	*Guided depth searching*
Computation costs	26 hours 33 minutes 3 seconds	1 hours 23 minutes 38 seconds

4.3.6 Occlusion and disocclusion

All IBR systems must deal with the problems of occlusion and disocclusion. An occlusion occurs when a visible surface in some input images becomes occluded in other input images or output image. This often results in one type of situation: *folds*. Traditional solution for *folds* problems is using Z-buffer techniques, providing depth information is available. Some algorithms even force the input views within very small translations so that visibility ambiguity does not pose a serious problem.

Another visibility issue is disocclusion problem. A *hole* happens in output image where part of scene is seen by output image, but not by the input images. To fill in holes, most commonly used methods include interpolating with neighbouring pixels of output images, or using redundant input images for better visibility.

Cubic panoramas can provide a 360-degree field of views plus 90-degree"UP" and "DOWN" view. Therefore, the visibility problems are highly ameliorated because of the use of cubic panoramas. However, due to the view positions and camera orientations, there are still some occlusions and disocclusions involved. Our solutions to these problems is using oversampling. That is we use redundant cubes to provide better surface coverage.

To generate a novel pixel from a set of N input cubes, we apply our *guided depth searching* method to find the depth for which the best colour consistency is obtained among the N cubes. Then the input cube with the pixel colour that differs the most from mean colour value is eliminated. (See Equation4.3). The novel pixel is finally interpolated with N-1 pixels of the input cubes.

4.4 Experiments

A number of experiments have been performed to test our cube interpolation algorithm. In our experiments, we used 4 pre-captured cubes to generate virtual cubes. In order to compare the virtual cube with real captured cube, we interpolated the virtual cube with the *rotation matrix* and *translation vector* of a real cube. If our algorithm works well, the virtual cube should be the same as the real cube.

For the first experiment, we used 4 indoor cubes, *cube 1, cube2, cube3 and cube4* (shown on Figure 4.7), as input views. The largest translation among these input cubes is about 1 meter. The world frame is attached with the frame of cube 1. We also used the *rotation matrix* and *translation vector* (related to world frame) of a real cube, *cube a* (shown on Figure 4.8), to generate virtual cube. Therefore, this virtual cube should have the same viewpoint and orientation as *cube a*.

Figure 4.7 shows the four input indoor cubes. The virtual cubes generated from these four cubes is shown as the top image on Figure 4.8. The result as a whole is quite

good considering the complexity of the scene and relative large translation. However, there are some small reconstruction errors, especially on the "computer monitor" on the right face of the virtual cube. The reason is that the "monitor" is very close to camera, which results in a large searching range. As stated previously, the closer the objects are to camera, the broader the searching ranges, and the higher the probability is to reconstruct pixels with wrong colours. As expected, we also noticed that the virtual cube appears to be the same as the real pre-captured *cube a* (shown on the bottom of Figure 4.8).

The next experiment shows the cube interpolation results of outdoor cubes with large translation. The experiment is set up the same as the first one but differ only by the translations of input cubes. The largest translation among these outdoor input cubes is about 10 meter, which is a very severe condition for image interpolation. Compared with the last experiment, the scene is even more complex. The four input outdoor cubes are shown on Figure 4.9, and the virtual cube is shown as the top image on Figure 4.10. The experiment shows the similar results as last one. However, the accuracy is not as good since the translation is too large. For example, there are very large reconstruction errors for bikes in the scene. The main reason is the large translations among input cubes result in the very small resolution of bikes in *cube 6* and *cube 8* shown on Figure 4.9. Also, the homogeneous colours in the scene (the colours of the floor and the main building are almost same) put more challenges on our method.

To test the *brute-force depth searching* method, more experiments on this method are also performed and the results are shown on Figure 4.11. As expected, the reconstruction errors are too big and the virtual cubes appear to be non-realistic.

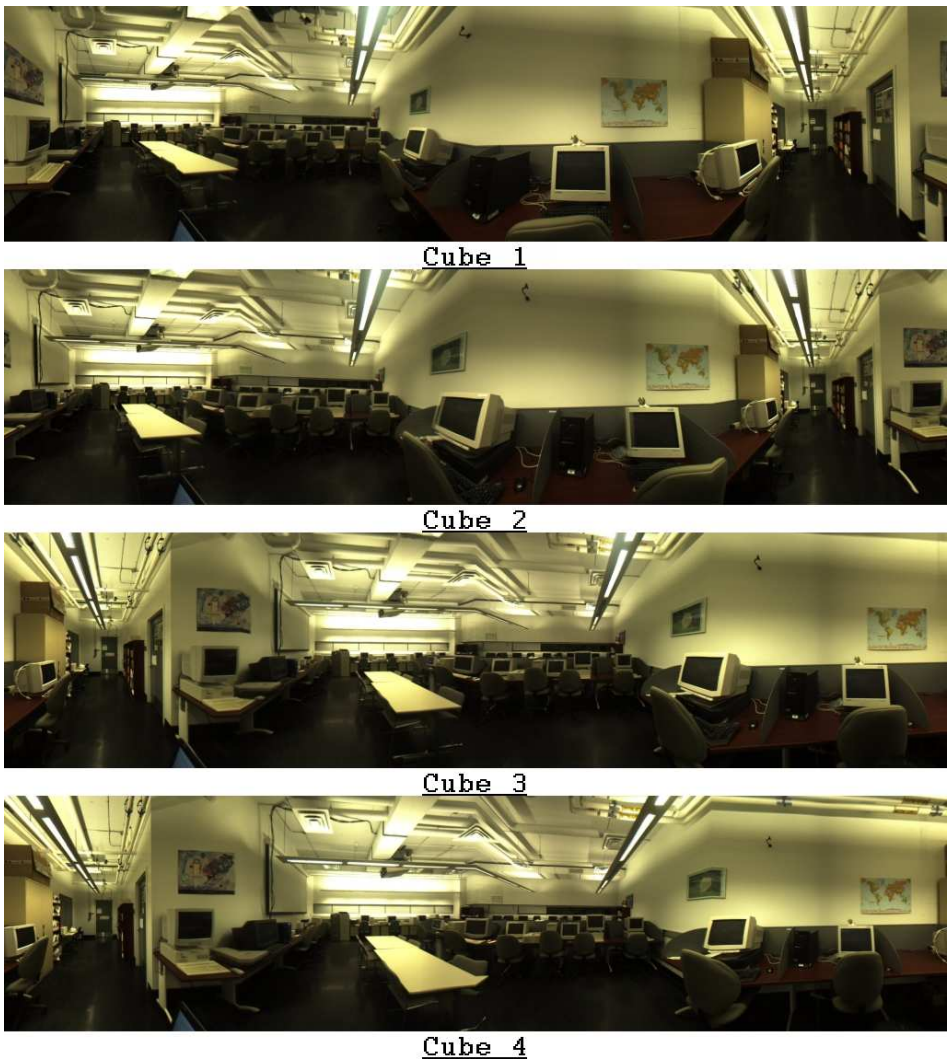


Figure 4.7: Indoor cube sequence (top and bottom faces not shown) used to generate virtual cubes.

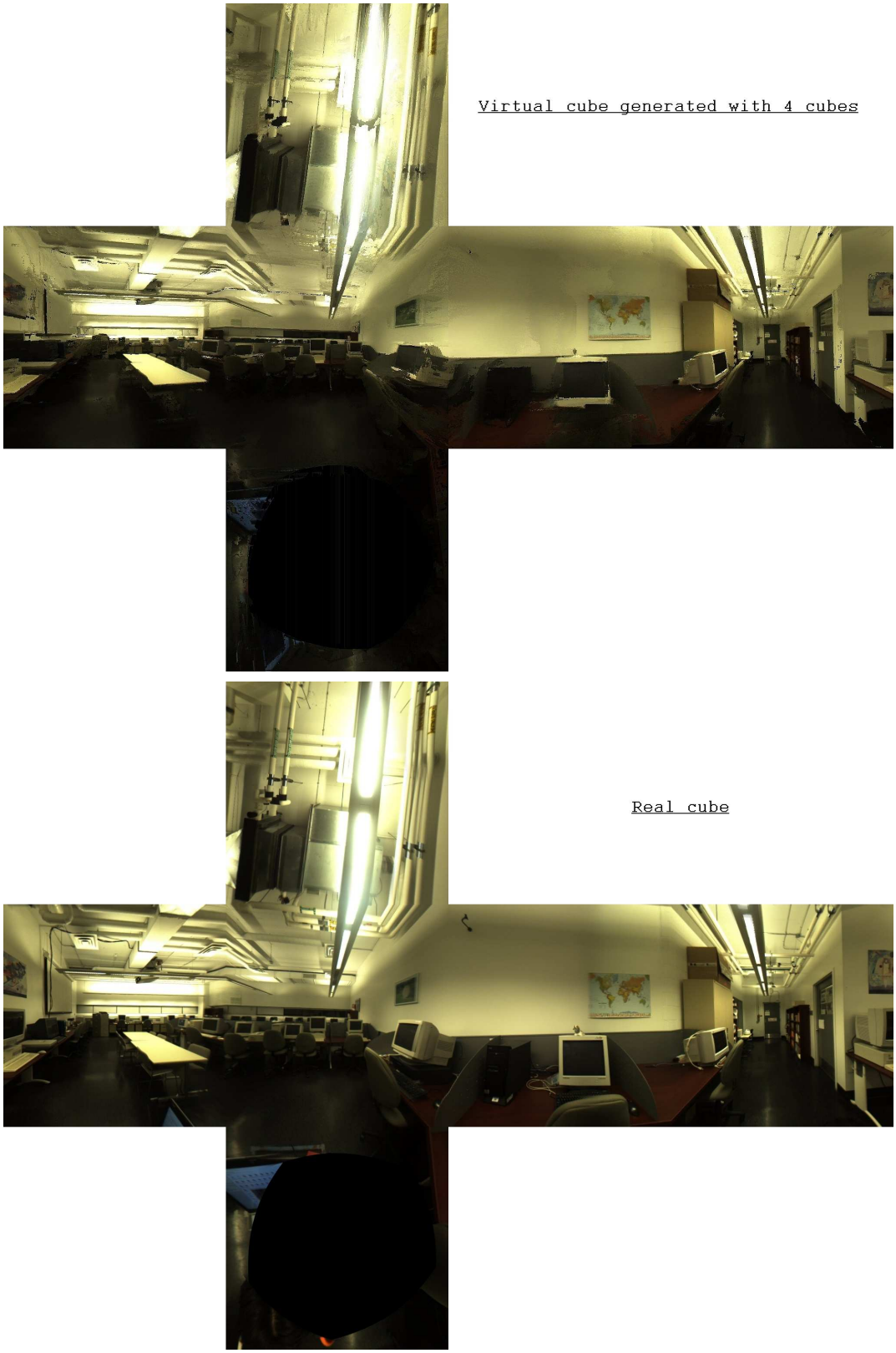


Figure 4.8: Virtual cube Vs. real cube for indoor cubes: the top cube is a virtual view generated from 4 cubes shown on Figure 4.7. This virtual cube is designated to produce the same view as the bottom real cube

Figure 4.9: Outdoor cube sequence (top and bottom faces not shown) used to generate virtual cubes.

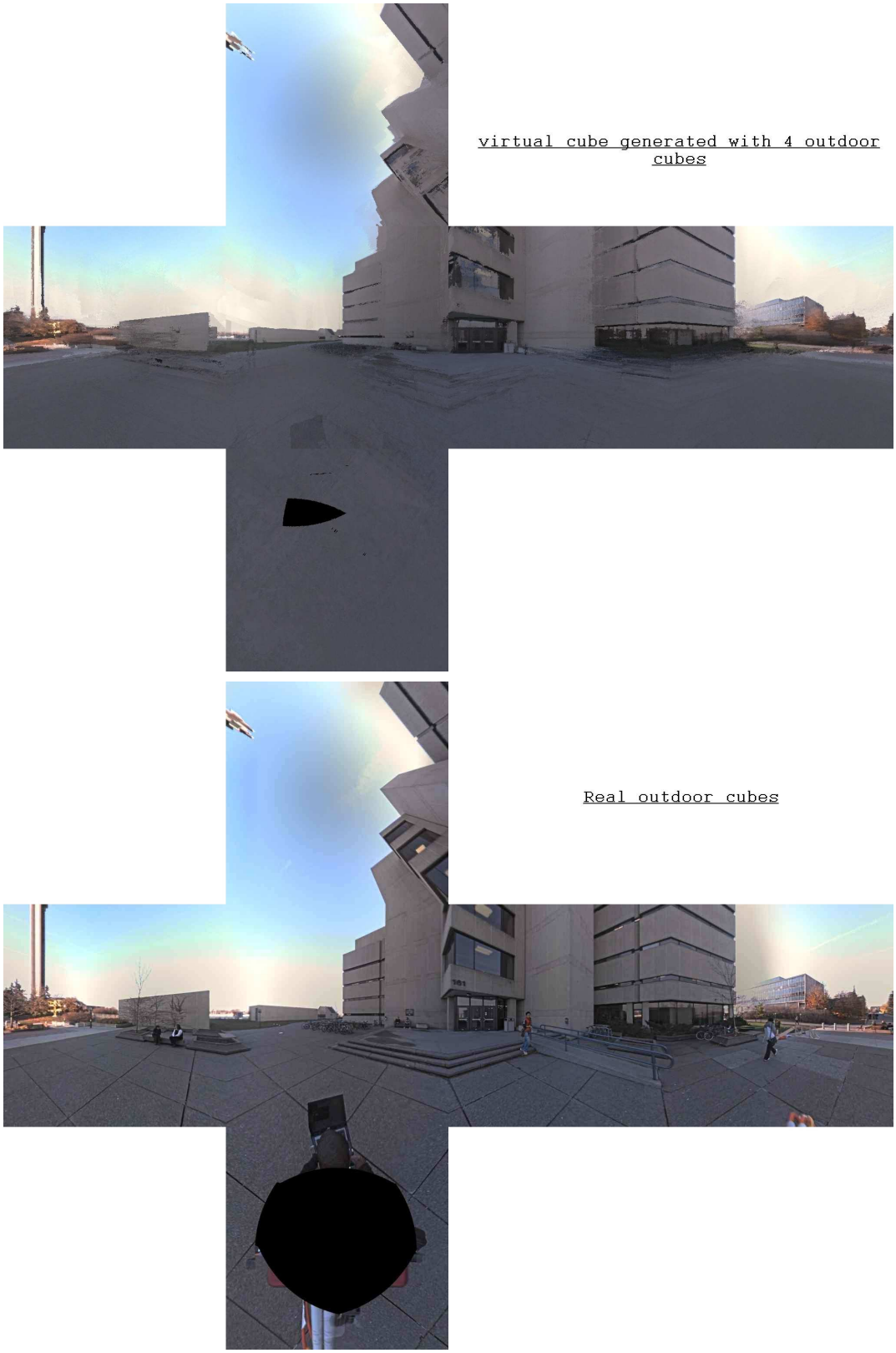


Figure 4.10: Virtual cube Vs. real cube for outdoor cubes: the top cube is a virtual view generated from 4 cubes shown on Figure 4.9. This virtual cube is designated to produce the same view as the bottom real cube

Figure 4.11: Virtual cubes generated with brute-force depth searching. Compared with virtual cubes generated with guided depth searching shown on Figure 4.8 and 4.10 , there are much more reconstruction errors.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this thesis, panorama view syntheses for image-based navigation have been studied for the purpose of performing virtual navigations of remote environment with an image-based rendering(IBR) system. In particular, cubic panorama has been introduced and its geometry as well as mathematical models have been discussed. Two algorithms have been proposed for generating virtual cubic panorama views.

Cubic panorama is a representation of panorama images with a set of six planar projections in the form of a cube. A cubic panorama can be modeled with a panorama image taken with six identical ideal pinhole cameras whose optical center are all fixed at the cubic center. All these cameras have 90° field of view, with non-overlapping image planes. According to this model, there are 36 non-independent essential/fundamental matrices between two cubes. However, only one of these matrices is independent and from which the others can be computed. Therefore, only one essential matrix is needed to express the epipolar geometry of two cubes, and a cube can be treated as a whole and not as a multi-sensor-camera system. In addition, cubic panoramas have implicit

intrinsic matrix (See Appendix D). It is not necessary to take full calibration procedures (i.e. only computing the essential matrix is needed) for 3D reconstruction.

In our *cube warping* algorithm, we proposed a fast, full-automatic method for cube view synthesis. Our method is based on image warping. First, a simplified model of cube pixel displacements is constructed to simulate a walkthrough from one cube to another. Second, the optical flow techniques are used to decide the "warping scales". Then the warping model based on the pixel displacements is applied to warp a input cube to approximate a real cube navigation. Although the approximate model is adopted, the experiment results show that our approach works well under small translations. Despite the limitation of its applications, the *cube warping* method has following strengths: (i) ability to produce photorealistic novel view; (ii) real-time novel view image synthesis regardless of the scene complexity; (iii)very low computation and communication costs.

In our *cube interpolation* algorithm, we presented an efficient method for view interpolation from multiple cube views. The main strength of our approach is the ability to control the location and orientation of the novel views with $\mathbf{R}$, $\mathbf{t}$ and synthesize arbitrary viewpoint views far away from (big translations) the input reference cubes. Instead of attempting to adopt traditional dense reconstruction approaches, the method try to reconstruct colours with colour invariant constraints. By designing a guided depth raytracing like searching strategy, the method can generate a novel scene view with maximized photo consistency. The solutions to the visibility issues were also provided in the text. Despite the high computation expense and small reconstruction errors, the *cube interpolation* method can produce complex virtual cube views for an arbitrary position with no limitation on gaze directions given a set of reference views, and therefore can acquire seamless visualization of environment from different viewing positions and orientations.

In comparison of two algorithms, please refer to the following table:

Table 5.1: Comparison of two algorithms

Algorithm	*Cube warping*	*Cube interpolation*
Computation costs	< 1 seconds	Several hours
Viewpoint	Limited on a straight line	No limitation
Results	Photorealistic, but approximate	With artifacts, but accurate

5.2 Future work

The problem of panorama view syntheses for image-based navigation, however, is far from solved. Future work could include the following improvements:

- Instead of only estimating cube epipolar goemetry pairwise with our feature matching approach, use bundle adjustment method to recover essential matrices in a global consideration.

- For *cube warping* algorithm, rectify and align the cubic panoramas with the method of [36] before cube warping to free our method's limitation on aligned cubes.

- For *cube interpolation* method, instead of searching the depth of one pixel at a time, try to perform a search on a window of the target image to locate the corresponding pixels in reference images.

- For guided depth searching, use segmentation techniques to lower computation costs and improve searching accuracy near region boundaries.

Appendix A

Cube Face Rotation Matrices

Cubic panoramas are very suitable for 3D reconstruction because of their implicit calibration and cube face relationships. Kangni and Laganière have given a good analysis of cube geometry in [36]. This and following appendices is partially based on their discussion.

A cubic panorama is made of six identical faces. Each of them can be seen as a image plane of a standard pinhole camera with 90° field of view. We name the six faces as: up, left, front, right, back, down, and label each face of the cube as F_i, for $i \in \{\mathtt{U}, \mathtt{L}, \mathtt{F}, \mathtt{R}, \mathtt{B}, \mathtt{D}\}$, with $\mathtt{U}$ standing for the up face, $\mathtt{L}$ standing for the left face and so on. As shown in Figure2.1(b), the cube reference frame is chosen as follows: the original point is located at the center of the cube with the x axis pointing to the "right" face, the y axis toward "down" face and the z axis toward the "front" face.

Since all the camera optical centers are the same at the cube center, the relationship of the six faces with the frame in Figure2.1(b) can be simply expressed as a rotation matrix. The rotation matrix $\mathbf{R}_i$ for $i \in \mathtt{U}, \mathtt{L}, \mathtt{F}, \mathtt{R}, \mathtt{B}, \mathtt{D}$ mentioned in the text can be expressed as follows:

$$\mathbf{R}_{\mathtt{U}} = \mathbf{R}_x(\frac{\pi}{2}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{R}_{\mathtt{L}} = \mathbf{R}_y(-\frac{\pi}{2}) = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{R}_{\mathtt{F}} = \mathbf{R}_x(0) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}_{\mathtt{R}} = \mathbf{R}_y(\frac{\pi}{2}) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{R}_{\mathtt{B}} = \mathbf{R}_y(\pi) = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

$$\mathbf{R}_{\mathtt{D}} = \mathbf{R}_x(-\frac{\pi}{2}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$$

Appendix B

Transformation of Arbitrary 3D Point and Cube Face 3D Point

Because cubic panoramas are omnidirectional, given an arbitrary 3D point $\mathbf{P} = (X, Y, Z)$, we can find its projection $\mathbf{P}_0 = (x_0, y_0, z_0)$ onto one of the cube faces. On the contrary, given any 3D point $\mathbf{P}_0 = (x_0, y_0, z_0)$ on cube face, we have a line of 3D point along vector $\mathbf{P}_0$. This line of 3D point can be expressed as $\mathbf{P}_i = \lambda_i * \mathbf{P}_0 = \lambda_i * (x_0, y_0, z_0)$. Here λ_i is depth scale.

The projection of an arbitrary 3D point with cube is equal to simply finding the intersection of this point with one of the cube faces.

B.1 Basic geometry: point, line, plane in 3D space

R. Harley and A. Zisserman have given an excellent discussion of point, plane, plane and quadrics in [30]. Based on their discussion, we adopt following basic geometry for our future cube analysis.

B.1.1 points

A point $\mathbf{P}$ in 3D-space IP^3 has 3 degree of freedom, and its homogeneous representation is $\mathbf{P} = (P_1, P_2, P_3, P_4)^T$. $P_4 = 0$ represent homogeneous points at infinity. If $P_4 \neq 0$, a homogeneous 3D points are often represented as $\mathbf{P} = (X, Y, Z, 1)^T$.

B.1.2 planes

A plane in IP^3 may be expressed as

$$aX + bY + cZ + d = 0$$

It also has 3 degree of freedom. The homogeneous representation of the plane in 3D space may be written as: $\mathbf{\Pi} = (a, b, c, d)^T$.

A point $\mathbf{P}$ is on the plane $\mathbf{\Pi}$ if

$$\mathbf{\Pi}^T\mathbf{P} = 0 \tag{B.1}$$

or

$$aP_1 + bP_2 + cP_3 + dP_4 = 0$$

B.1.3 lines

A line can be defined by the intersection of two planes or the join of two points. It has 4 degree of freedom in IP^3 space. To represent an object with 4 degrees of freedom, we need a homogeneous 5-vector. The problem is we can not find simple operations for a 5-vector with the 4-vector representation of points and planes.

Plücker matrices provide a good solution to this problem. Here a line in 3D space is represented by a four-by-four skew-symmetric homogeneous matrix. In particular, the line joining two 3D points $\mathbf{P}_1$, $\mathbf{P}_2$ can be represented by the skew-symmetric matrix $\mathbf{L}$

as

$$\mathbf{L} = \mathbf{P}_1\mathbf{P}_2^T - \mathbf{P}_2\mathbf{P}_1^T \tag{B.2}$$

For example, given a point $\mathbf{P} = (1, 2, 3, 1)^T$, the line joining it with original point $\mathbf{P}_0 = (0, 0, 0, 1)^T$ is represented as

$$\mathbf{L} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -2 \\ 0 & 0 & 0 & -3 \\ 1 & 2 & 3 & 0 \end{bmatrix}$$

The intersection of the line $\mathbf{L}$ with the plane $\mathbf{\Pi}$ is a point $\mathbf{P}$:

$$\mathbf{P} = \mathbf{L}\,\mathbf{\Pi} \tag{B.3}$$

In the case of a cube of side 512, the line $\mathbf{L}$ of above example intersect with "front" face plane $\mathbf{\Pi}_\mathrm{F} = (0, 0, 1, -256)^T$, which is $(Z = 256)$, at point:

$$\mathbf{P}_0 = \mathbf{L}\,\mathbf{\Pi}_\mathrm{F} = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -2 \\ 0 & 0 & 0 & -3 \\ 1 & 2 & 3 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ -256 \end{bmatrix} = \begin{bmatrix} \frac{1}{3} \times 256 \\ \frac{2}{3} \times 256 \\ 256 \\ 1 \end{bmatrix}$$

B.2 3D point and cube face intersection

Given an arbitrary 3D point on any object, its image in cube may be simply acquired by projecting this 3D point with one of the six cube faces.

B.2.1 Line equation for a 3D point vector

The line equation of a point vector $\mathbf{P} = (x, y, z, 1)^T$ is the line joining it with original point $\mathbf{P}_0 = (0, 0, 0, 1)^T$:

$$\mathbf{L} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} [x \;\; y \;\; z \;\; 1] - \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} [0 \;\; 0 \;\; 0 \;\; 1] = \begin{bmatrix} 0 & 0 & 0 & -x \\ 0 & 0 & 0 & -y \\ 0 & 0 & 0 & -z \\ x & y & z & 0 \end{bmatrix} \tag{B.4}$$

B.2.2 face plane equation

For the reference shown in Figure2.1(b), since all the faces are perpendicular to one of the X, Y, Z axis, their equations are simple. Given a cube of size d, its face equations are given below:

$$\text{“}up\text{”}\ face\ Y = -\frac{d}{2}: \qquad \mathbf{\Pi}_{\mathtt{U}} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \frac{d}{2} \end{bmatrix}$$

$$\text{“}left\text{”}\ face\ X = -\frac{d}{2}: \qquad \mathbf{\Pi}_{\mathtt{L}} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \frac{d}{2} \end{bmatrix}$$

$$\text{“}front\text{”}\ face\ Z = \frac{d}{2}: \qquad \mathbf{\Pi}_{\mathtt{F}} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ -\frac{d}{2} \end{bmatrix}$$

$$
\begin{aligned}
\text{“}right\text{”}\ face\ X = \frac{d}{2}: \qquad & \mathbf{\Pi}_{\mathtt{R}} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ -\frac{d}{2} \end{bmatrix} \\
\text{“}back\text{”}\ face\ Z = -\frac{d}{2}: \qquad & \mathbf{\Pi}_{\mathtt{B}} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \frac{d}{2} \end{bmatrix} \\
\text{“}down\text{”}\ face\ Y = \frac{d}{2}: \qquad & \mathbf{\Pi}_{\mathtt{D}} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ -\frac{d}{2} \end{bmatrix}
\end{aligned} \tag{B.5}
$$

B.2.3 3D vector and face point conversion

There is only 90° field of view for each face camera. An arbitrary point vector in space can only intersect one point on one cube faces. After having equation of 3D line and face plane, we can easily find this point on cube (of size d) face with following strategies:

1. Use **Equation(B.3)** to compute intersection of 3D point (**Equation(B.4)**) with all six faces (**Equation(B.5)**)

2. From the 6 computed intersecting points, eliminate four points which have absolute coordinate value bigger than $d/2$

3. The remaining two points are from two opposite faces. Find the coordinate with absolute value of $d/2$. Eliminate last point if the sign of the coordinate is different

with the sign of corresponding coordinate of original 3D space point

Appendix C

Transformation of Face 3D Vector and Face Image Point

Given a cube of side d under the frame in Figure2.1(b), we want to convert between cube face $3D$ vector and cube face $2D$ image coordinates. We observed: for right and left face, $x = \pm\frac{d}{2}$; for down and up face, $y = \pm\frac{d}{2}$; for front and back face, $z = \pm\frac{d}{2}$. This gives us a group of simple transformation matrices $\mathbf{T}_i$, for $i \in \{\mathtt{U}, \mathtt{L}, \mathtt{F}, \mathtt{R}, \mathtt{B}, \mathtt{D}\}$, to convert between cube face $3D$ vector and cube face $2D$ image coordinates.

$$\mathbf{T}_{\mathtt{U}} = \begin{bmatrix} 1 & 0 & -\frac{d}{2} \\ 0 & 0 & -\frac{d}{2} \\ 0 & 1 & -\frac{d}{2} \end{bmatrix}$$

$$\mathbf{T}_{\mathtt{L}} = \begin{bmatrix} 0 & 0 & -\frac{d}{2} \\ 0 & 1 & -\frac{d}{2} \\ 1 & 0 & -\frac{d}{2} \end{bmatrix}$$

$$\mathbf{T}_{\mathtt{F}} = \begin{bmatrix} 1 & 0 & -\frac{d}{2} \\ 0 & 1 & -\frac{d}{2} \\ 0 & 0 & \frac{d}{2} \end{bmatrix}$$

$$\mathbf{T}_{\mathtt{R}} = \begin{bmatrix} 0 & 0 & \frac{d}{2} \\ 0 & 1 & -\frac{d}{2} \\ -1 & 0 & \frac{d}{2} \end{bmatrix}$$

$$\mathbf{T}_{\mathtt{B}} = \begin{bmatrix} -1 & 0 & \frac{d}{2} \\ 0 & 1 & -\frac{d}{2} \\ 0 & 0 & -\frac{d}{2} \end{bmatrix}$$

$$\mathbf{T}_{\mathtt{D}} = \begin{bmatrix} 1 & 0 & -\frac{d}{2} \\ 0 & 0 & \frac{d}{2} \\ 0 & -1 & \frac{d}{2} \end{bmatrix} \tag{C.1}$$

For a cube face image $2D$ point $\mathbf{p} = (x, y, 1)^T$ and a cube face $3D$ vector $\mathbf{P} = (X, Y, Z)^T$, the conversion function are:

$$\mathbf{p} = \mathbf{T}_i \mathbf{P}$$

or

$$\mathbf{P} = inv(\mathbf{T}_i)\mathbf{p}$$

inv() means inverse matrix.

Appendix D

Cube Intrinsic Matrix

As mentioned in appendixA, cubic panoramas have implicit calibration parameters. It is not necessary to take full calibration procedures for 3D reconstruction (up to scale).

A cubic panorama is made of six identical faces. Each of them can be seen as a image plane of a standard pinhole camera with 90° field of view. All the six cameras are centered at the same camera center, which is also the cube center. In the case of a cube of side d with the frame shown in Figure2.1(b), the image plane is at a distance $\frac{d}{2}$ from camera center, and the principal point is always at $\left(\frac{d}{2}, \frac{d}{2}\right)$ of image plane. Thus the cube intrinsic matrix may be written:

$$\mathbf{K} = \begin{bmatrix} \frac{d}{2} & 0 & \frac{d}{2} \\ 0 & \frac{d}{2} & \frac{d}{2} \\ 0 & 0 & 1 \end{bmatrix} \tag{D.1}$$

Appendix E

3D Reconstruction: Linear Triangulation

This appendix discusses how to compute the position of a scene point in 3D space given its image in two views and the camera projection matrices of those views. We will describe a simple 3D reconstruction method: linear triangulation. This appendix is partially based on the discussion of [30].

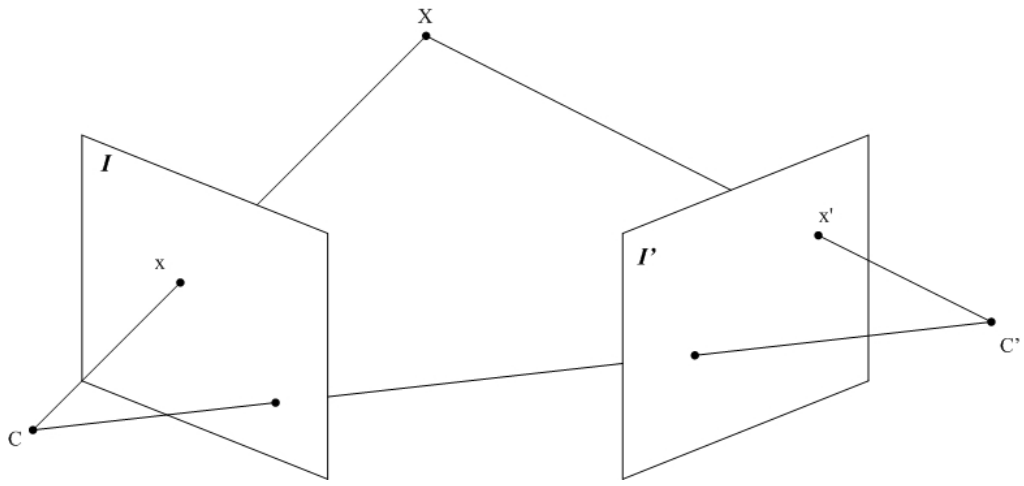


Figure E.1: 3D reconstruction by linear triangulation

As illustrated in FigureE.1, a 3D point $\mathbf{X} = (X, Y, Z, 1)^T$ projects to the two images, I and I', at image points $\mathbf{x} = (x, y, 1)^T$ and $\mathbf{x}' = (x', y', 1)^T$. We assume that cameras are calibrated. Thus, camera projection matrices, $\mathbf{P}$ and $\mathbf{P}'$, are available.

For two images, we have following measurements:

$$\begin{aligned} \mathbf{x} &= \mathbf{P}\mathbf{X} \\ \mathbf{x}' &= \mathbf{P}'\mathbf{X} \end{aligned} \tag{E.1}$$

Then, we can use cross product, $\mathbf{x} \times (\mathbf{P}\mathbf{X}) = 0$ and $\mathbf{x}' \times (\mathbf{P}'\mathbf{X}) = 0$, to get three equations (up to scales) for each image point, of which two are linearly independent. For example, $\mathbf{x} \times (\mathbf{P}\mathbf{X}) = 0$ can be written as:

$$\begin{aligned} x\left(\mathbf{p}^{3T}\mathbf{X}\right) - \left(\mathbf{p}^{1T}\mathbf{X}\right) &= 0 \\ y\left(\mathbf{p}^{3T}\mathbf{X}\right) - \left(\mathbf{p}^{2T}\mathbf{X}\right) &= 0 \\ x\left(\mathbf{p}^{2T}\mathbf{X}\right) - y\left(\mathbf{p}^{1T}\mathbf{X}\right) &= 0 \end{aligned} \tag{E.2}$$

Where $\mathbf{p}^{iT}$ is the row vector of the i^{th} row of the $\mathbf{P}$. These equation are linear in the components of $\mathbf{X}$. Therefore, EquationE.1 can be composed into a linear equation: $\mathbf{A}\mathbf{X} = 0$, with

$$\mathbf{A} = \begin{bmatrix} x\mathbf{p}^{3T} - \mathbf{p}^{1T} \\ y\mathbf{p}^{3T} - \mathbf{p}^{2T} \\ x'\mathbf{p}'^{3T} - \mathbf{p}'^{1T} \\ y'\mathbf{p}'^{3T} - \mathbf{p}'^{2T} \end{bmatrix} \tag{E.3}$$

This is a linear equation with a total of four equations in four homogeneous unknows. It is easy to find the solution to this equation by using Direct Linear Transformation (DLT) algorithm (see [30]).

For above linear triangulation algorithm, it is assumed that camera matrices are

known exactly and the errors in the measured image points $\mathbf{x}$ and $\mathbf{x}'$ are negligible. However, since there are errors in the measured image points as well as camera matrices, the simple triangulation by back-projecting rays from the measured image points will not intersect in general. This means there will be no 3D point $\mathbf{X}$ which exactly satisfies $\mathbf{x} = \mathbf{P}\mathbf{X}$, $\mathbf{x}' = \mathbf{P}'\mathbf{X}$.

To solve such problems, there are some more sophisticated algorithms which are projective-invariant and can minimize reprojection errors. For more details, please refer to [30]. Nevertheless, the linear triangulation method described above can generate good result.

Appendix F

Glossary of Terms

geometry-based rendering (GBR) A rendering approach in which objects and environments are modeled and rendered with geometric primitives.

image-based rendering (IBR) A rendering approach in which objects and environments are modeled and rendered with image data instead of geometric primitives.

node One pre-captured cell of cubic panorama image in the navigation grid. Navigators can walk-through from one node into another node.

reference image The prestored real image at the node of grid. It is used to interpolate novel views.

cube homing The precess of warping one cube to approximate another cube.

equal distance assumption All the environment objects are assumed to have same distance from the location of snapshot.

transfer For a set of images, given the position of a point in one (or more) image(s), determine the positions in all other images of the set.

Bibliography

[1] Internet raytracing competition. `http://www.irtc.org/`.

[2] P. Anandan. A computational framework and an algorithm for the measurement of visual. *International Journal of Computer Vision*, 2(3):283–310, Jan, 1987.

[3] A. A. Argyros, K. E. Bekris, and S. C. Orphanoudakis. Robot homing based on corner tracking in a sequence of panoramic images. *CVPR*, 02:3, 2001.

[4] H. S. Baird. Document image defect models and their uses. *Proc., IAPR 2nd Int'l Conf. on Document Analysis and Recognition*, 1993.

[5] J.L. Barron, D.J. Fleet, S.S. Beauchemin, and T.A. Burkitt. Performance of optical flow techniques. *CVPR*, 92:236–242, 1992.

[6] J.L. Barron and N.A. Thacker. Tutorial: Computing 2D and 3D optical flow, 2005.

[7] T. Beier and S. Neely. Feature-based image metamorphosis. In *SIGGRAPH92*, volume 26, pages 35–42, 1992.

[8] L. Bergman, H. Fuchs, E. Grant, and S. Spach. Image rendering by adaptive refinement. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 29–37, 1986.

[9] D. Binding and F. Labrosse. Visual local navigation using warped panoramic images. In *Proceedings of Towards Autonomous Robotic Systems*, University of Surrey, Guildford, UK, 2006.

[10] D. Bradley, A. Brunton, M. Fiala, and G. Roth. Image-based navigation in real environments using panoramas. In *IEEE Int. Workshop on Haptic Audio Visual Environments and their Applications*, pages 103 – 108, October 2005.

[11] S. Chen and L. Williams. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, volume 27, pages 279–288, 1993.

[12] S. E. Chen. QuickTime VR-an image-based approach to virtual environment navigation. In *Computer Graphics (SIGGRAPH'95)*, pages 29–38, Augest 1995.

[13] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright. Simplification envelopes. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 119–128, 1996.

[14] W. B. Culbertson, T. Malzbender, and G. Slabaugh. Generalized voxel coloring. In *ICCV '99: Proceedings of the International Workshop on Vision Algorithms*, pages 100–115, September 1999.

[15] F. Dornaika. View synthesis from two uncalibrated images. In *ICIAP '01: Proceedings of the 11th International Conference on Image Analysis and Processing*, page 284, 2001.

[16] P. Eisert, E. Steinbach, and B. Girod. Multi-hypothesis volumetric reconstruction of 3-D objects from multiple calibrated camera views. In *Proc. International Con-*

ference on Acoustics, Speech, and Signal Processing (ICASSP'99), pages 3509–3512, Phoenix, USA, 1999.

[17] O. Faugeras and Q-T. Luong. *The geometry of multiple images*. Cambridge University Press, ISBN: 0262062208, first edition, 2001.

[18] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[19] S. Fleishman, B. Chen, A. Kaufman, and D. Cohen-Or. Navigating through sparse views. In *VRST '99: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 82–87, 1999.

[20] W. Forstner and E. Gulch. A fast operator for detection and precise location of distinct points, corners and centres of circular features. *Intercommission Conference on Fast Processing of Photogrammetric Data*, pages 281–305, 1987.

[21] M. Franz, B. Scholkopf, and H. Bulthoff. Homing by parameterized scene matching. In *Proc. 4th Europ. Conf. on Artificial Life (to appear)*, 1997.

[22] M. Franz, B. Scholkopf, H. Mallot, and H. Bulthoff. Where did i take that snapshot? scene-based homing by image matching. 79:191–202, 1998.

[23] M. Garland and P. S. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *IEEE Visualization '98*, pages 263–270, 1998.

[24] S. Genc; and V. Atalay. Texture extraction from photographs and rendering with dynamic texture mapping. In *ICIAP '99: Proceedings of the 10th International Conference on Image Analysis and Processing*, page 1055, 1999.

[25] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54, New Orleans, LA, USA, 1996.

[26] N. Greene. Environment mapping and other applications of world projections. *IEEE Comput. Graph. Appl.*, 6(11):21–29, 1986.

[27] N. Greene. Hierarchical polygon tiling with coverage masks. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 65–74, 1996.

[28] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[29] R. Hartley. Theory and practice of projective rectification. *Int. Journal Computer Vision*, 35(2):115–127, 1999.

[30] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[31] T. He, L. Hong, A. Varshney, and S. W. Wang. Controlled topology simplification. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):171–184, 1996.

[32] D. J. Heeger. Model for the extraction of image flow. *Journal of the Optical Society of America A*, 4:1455–1471, aug 1987.

[33] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 577–584, Los Angeles, California, 2005.

[34] B. K. P. Horn and B. G. Schunck. Determining optical flow. 17:185–204, 1981.

[35] S. B. Kang and R. Szeliski. 3-D scene data recovery using omnidirectional multibaseline stereo, 1997.

[36] F. Kangni and R. Laganière. Epipolar geometry for the rectification of cubic panoramas. In *CRV '06: Proceedings of the The 3rd Canadian Conference on Computer and Robot Vision*, page 70, 2006.

[37] K. N. Kutulakos and S. Seitz. What do N photographs tell us about 3D shape? In *Technical Report TR680, Computer Science Dept., U. Rochester*. January 1998.

[38] Viva Lab. Virtual navigation in image-based representations of real world environments. 2006. `http://www.site.uottawa.ca/ research/viva/projects/ibr/`.

[39] R. Laganière, H. Hajjdiab, and A. Mitiche. Visual reconstruction of ground plane obstacles in a sparse view robot environment. *Graphical Models*, 68(3):282–293, 2006.

[40] S. Laveau and O. Faugeras. 3-D scene representation as a collection of images and fundamental matrices. In *Proceedings of the 12th IAPR International Conference*, volume 1, pages 689–691, Oct. 1994.

[41] M. Levoy and P. Hanrahan. Light field rendering. In *SIGGRAPH '96*, pages 31–42, New Orleans, LA, USA, 1996.

[42] M. Lhuillier and L. Quan. Image interpolation by joint view triangulation. In *Proceedings of IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 139–145, Fort Collins, CO, USA, 1999.

[43] A. Lippman. Movie maps: An application of the optical videodisc to computer graphics. In *Computer Graphics (SIGGRAPH'80)*, pages 32–43, 1980.

[44] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, September 1981.

[45] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[46] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI81*, pages 674–679, 1981.

[47] D. Luebke and C. Georges. Portals and mirrors: simple, fast evaluation of potentially visible sets. In *SI3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 105–ff., Monterey, California, United States, 1995.

[48] L. McMillan. *An image-based approach to three-dimensional computer graphics*. PhD thesis, University of North Carolina at Chapel Hill, Apr 1997.

[49] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. *Computer Graphics*, 29(Annual Conference Series):39–46, 1995.

[50] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Proceedings of the 8th International Conference on Computer Vision, Vancouver, Canada*, volume 2, pages 525–531, 2001.

[51] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27(10):1615–1630, 2005.

[52] G. E. Miller, S. E. Hoffert, E. Chen, D. Patterson, S. Blackketter, S. A. Rubin, D. Applin, and J. Hanan Yim. The virtual museum: Interactive 3D navigation of a multimedia database. *The Journal of Visualization and Computer Animation*, 3(3):183–197, 1992.

[53] E. Ofek, E. Shilat, A. Rappopport, and M. Werman. Highlight and reflection independent multiresolution textures from image sequences, March-April 1997.

[54] A. Prock and C. Dyer. Towards real-time voxel coloring. In *Proc. Image Understanding Workshop*, pages 315–321, 1998.

[55] P. Rademacher and G. Bishop. Multiple-center-of-projection images. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, volume 32, pages 199–206, 1998.

[56] T. Rofer. Controlling a wheelchair with image-based homing. In *Proceedings of the AISB Symp. on Spatial Reason. in Mobile Robots and Animals*, Manchester University, UK, 1997.

[57] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172, 2000.

[58] S. Seitz and C. Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proc. CVPR*, pages 1067– 1073, June 1997.

[59] S.M. Seitz and C. R. Dyer. View morphing. In *SIGGRAPH96*, pages 21–30, 1996.

[60] J. Shade, S. Gortler, L. He, and R. Szeliski. Layered depth images. In *SIGGRAPH*, pages 231–242, July 1998.

[61] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, Jun 1994.

[62] H. Shum and L. He. Rendering with concentric mosaics. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 299–306, 1999.

[63] A. Singh. An estimation-theoretic framework for image-flow computation. In *Third international conference on computer vision*, pages 168–177, 1990.

[64] G. Slabaugh, B. Culbertson, T. Malzbender, and R. Schafer. A survey of methods for volumetric scene reconstruction from photographs. In *Proc. Int'l Workshop Volume Graphics*, pages 81–100, June 2001.

[65] R. Szeliski, S.B. Kang, and H. Shum. A parallel feature tracker for extended image sequences. In *SCV95*, page 5A Motion II, 1995.

[66] R. Szeliski and H.Y. Shum. Creating full view panoramic image mosaics and environment maps. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 251–258, 1997.

[67] E. Vincent and R.Laganière. Matching with epipolar gradient features and edge transfer. In *Proc. IEEE Int. Conf. Image Processing*, volume 1, pages 277–280, Barcelona, Spain, Sept, 2003.

[68] X. Wang, J. Lim, R. T. Collins, and A. R. Hanson. Automated texture extraction from multiple images to support site model refinment and visualization. In *Winter School of Computer Graphics 1996*, 1996.

[69] T. Werner, R. D. Hersch, and V. Hlavac. Rendering real-world objects using view interpolation. In *ICCV '95: Proceedings of the Fifth International Conference on Computer Vision*, page 957, 1995.

[70] G. Wolgerg. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, CA, 1990.

[71] Y. Xiong and K. Turkowski. Creating image-based VR using a self-calibrating fisheye lens. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 237, 1997.

[72] 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. *Artificial Intelligence Journal*, 78:87–119, OCT 1995.