2006-08-14

Vision-Based Rendering: Using Computational Stereo to Actualize IBR View Synthesis

Kevin L. Steele
Brigham Young University - Provo

VISION-BASED RENDERING: USING COMPUTATIONAL STEREO TO ACTUALIZE IBR VIEW SYNTHESIS

by

Kevin L. Steele

A dissertation submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

Brigham Young University

December 2006

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a dissertation submitted by

Kevin L. Steele

This dissertation has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Parris K. Egbert, Chair
Date	Bryan S. Morse
Date	Thomas W. Sederberg
Date	Michael A. Goodrich
Date	Dan Ventura

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the dissertation of Kevin L. Steele in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date	Parris K. Egbert Chair, Graduate Committee

Accepted for the Department

Parris K. Egbert
Graduate Coordinator

Accepted for the College

Thomas W. Sederberg
Associate Dean, College of Physical and Mathematical Sciences

ABSTRACT

VISION-BASED RENDERING: USING COMPUTATIONAL STEREO TO ACTUALIZE IBR VIEW SYNTHESIS

Kevin L. Steele

Computer Science

Doctor of Philosophy

Computer graphics imagery (CGI) has enabled many useful applications in training, defense, and entertainment. One such application, CGI simulation, is a real-time system that allows users to navigate through and interact with a virtual rendition of an existing environment. Creating such systems is difficult, but particularly burdensome is the task of designing and constructing the internal representation of the simulation content. Authoring this content on a computer usually requires great expertise and many man-hours of labor.

Computational stereo and image-based rendering offer possibilities to automatically create simulation content without user assistance. However, these technologies have largely been limited to creating content from only a few photographs, severely limiting the simulation experience. The purpose of this dissertation is to enable the process of automated content creation for large numbers of photographs. The workflow goal consists of a user photographing any real-world environment intended for simulation, and then loading the photographs into the computer. The theoretical and algorithmic contributions of the dissertation are then used to transform the photographs into the data required for real-time exploration of the photographed locale.

This permits a rich simulation experience without the laborious effort required to author the content manually.

To approach this goal we make four contributions to the fields of computer vision and image-based rendering: an improved point correspondence methodology, an adjacency graph construction algorithm for unordered photographs, a pose estimation ordering for unordered image sets, and an image-based rendering algorithm that interpolates omnidirectional images to synthesize novel views. We encapsulate our contributions into a working system that we call Vision-Based Rendering (VBR).

With our VBR system we are able to automatically create simulation content from a large unordered collection of input photographs. However, there are severe restrictions in the type of image content our present system can accurately simulate. Photographs containing large regions of high frequency detail are incorporated very accurately, but images with smooth color gradations, including most indoor photographs, create distracting artifacts in the final simulation. Thus our system is a significant and functional step toward the ultimate goal of simulating any real-world environment.

ACKNOWLEDGMENTS

I would like to thank my dissertation advisor Dr. Parris Egbert for his support, advice, counsel, confidence, and patience. I also thank Dr. Bryan Morse and David Cline for their many helpful conversations and suggestions to my research. Above all I thank my wife Sandra for her endless encouragement, her positive perspective, and her willingness to live the student life for so long.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Computer graphics imagery (CGI) has progressed from a novelty in the 1960s to an indispensable technology in the 2000s. What was once a nascent academic niche is now a world-wide market with applications in simulation and training, medical visualization, defense and security, and entertainment. Furthermore, the ability to render CGI fast enough for smooth animation (at least 20-30 frames per second for "real-time" animation) has been an important goal from the outset due to the practical uses of such technology. For instance, real-time computer-generated flight simulation was initiated by General Electric and others in the mid-1960's [Schacter, 1981], a very early point in computer graphics history.

Many applications lend themselves well to real-time computer graphics, including the relatively new technology of virtual reality, wherein a user visually navigates and interacts with an artificial environment using the computer generated imagery as a navigational cue. In non-immersive virtual reality simulations[1] one typically uses an input device such as a keyboard, mouse, or joystick to indicate a desired route to follow or action to take. The computer simultaneously renders an image of what the user would see if the monitor were a window into the artificial environment. An important ability of such simulations is to place the user's viewpoint or virtual "eye" wherever the user desires in the artificial world and render the images from that vantage point.

[1] Non-immersive virtual reality usually refers to the use of low-end hardware such as a workstation or PC to compute and render the artificial world. Examples of this include consumer-grade driving and flight simulators and "first-person shooter" or avatar-based video games. High-end virtual reality installations generally employ more exotic hardware such as wired gloves, omnidirectional treadmills and immersive displays.

Virtual reality systems typically simulate *artificial* environments, worlds which have been imagined, designed and instantiated solely in the computer. While navigating artificial environments in this manner is interesting, navigating models of *existing* environments is a much more useful function. Modern flight simulators and architectural walkthrough systems fall into this category, as do many types of training simulators used by industry and the military. To develop these types of systems a designer must first create three-dimensional geometry that will represent objects, buildings and environmental features in the simulation. The creation process includes defining geometric entities such as points, lines, planes and polygons, coloring the entities using palettes and photographs, and placing sources of lighting to realistically illuminate the geometric creation. Once finished, the collection of geometric entities, sometimes referred to as "content," can be rendered in real-time by the computer for the simulation output. Unfortunately, these creational tasks are laborious, time consuming, and require very specialized training, and hence they are expensive to realize. Additionally, the resulting simulation output generally does not look identical to the real subject matter that was modeled, a disadvantage for some applications (Figure 1.1).

Another common type of virtual reality simulation is terrain visualization—*Google Earth*[2] is a well-known example of a terrain visualization product. The content for terrain visualization simulations is often created by overlaying satellite or aerial photography on geometric tessellations created from digital elevation models (DEMs). While the content creation process is much more automated, the simulation output quality usually suffers considerably when the user's viewpoint is close to ground level.

1.1 Intent

The goal of this dissertation is to eliminate the tedious content creation process from virtual reality simulations by constructing the content data automatically and exclusively from photographs. The resulting simplified workflow consists of a user photographing the real-world environment intended for simulation and then loading

[2] `http://earth.google.com`

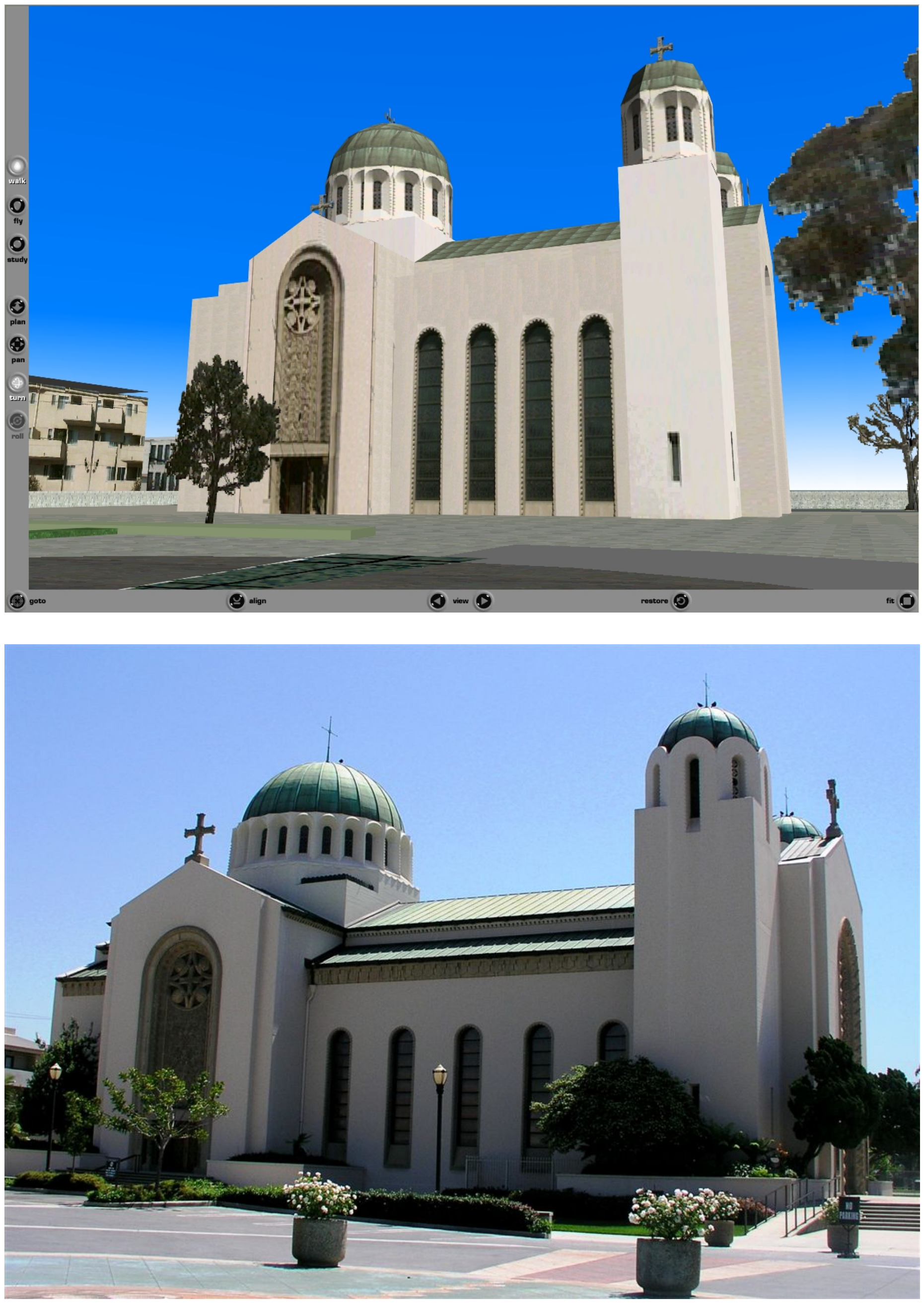


Figure 1.1: St. Sophia's Church in Los Angeles. The top image is a VRML model of the church (`http://oldcda.design.ucla.edu/CAAD/worlds.html`) rendered using a real-time VRML viewer. The bottom image is a photograph of the same church (`http://www.constantinepainting.com/gallery.html`). The difference in image quality is due to the simplified geometric model and rudimentary rendering technique required to meet the real-time performance constraint.

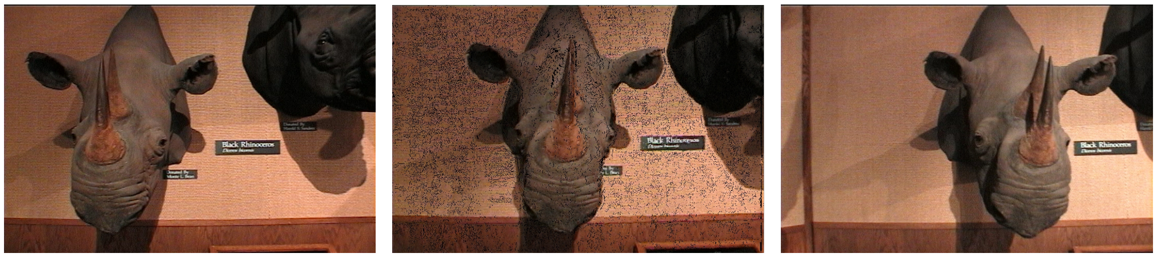


Figure 1.2: Image-based rendering creates images of novel viewpoints (middle) given a set of input images (left and right). This example uses an IBR technique called View Interpolation [Chen and Williams, 1993]. The black spots in the interpolated image are characteristic artifacts resulting from sampling rate differences between the input and novel viewpoints. Most IBR algorithms include methods to reduce artifacts inherent in their respective techniques.

the photographs into the computer. The theoretical and algorithmic contributions of the dissertation are then used to transform the photographs into data necessary for real-time exploration of the photographed locale. This frees the designer from the tedious task of creating content resembling the target location, and facilitates the system creation process. The approach we take to solve this problem is to employ image-based rendering.

1.2 Image-Based Rendering

Image-based rendering (IBR) is a relatively new sub-field of computer graphics. The goal is still to render images, but rather than rendering from a geometric scene description, the rendering is performed using metadata from other images, including photographs and video streams. IBR is often used to render the image content from a novel viewpoint that is not represented in the photographic input. For example, given two photographs of the rhinoceros head in Figure 1.2, a third image is rendered representing a viewpoint between the original two. We will give a brief background of current IBR techniques and then discuss the specific approach we use to address the content-creation problem.

1.2.1 IBR Background

This section will highlight the main contributions in image-based rendering. More thorough surveys can be found in Oliveira [2002] and Zhang and Chen [2003].

Image-based rendering was first introduced in a seminal paper by Chen and Williams [1993]. There had been prior work done with the intention of interactive walkthroughs of photographed locations, such as Lippman's Movie Maps [Lippman, 1980] and Miller's Virtual Museum [Miller et al., 1992]. However, Chen and Williams were the first to deconstruct the sampled contents of an image for later reconstruction to fit an objective function, in this case the simulation of a novel viewpoint. In their paper they present a method to generate images from novel viewpoints within a regularly spaced grid of input images, each with known camera and depth characteristics. To generate a new view, pixels from images at enclosing grid points are linearly interpolated to new (x, y) locations using a preprocessed mapping.

Many new algorithms to accomplish IBR-related goals soon followed. Chen [1995] reported a summary of Apple Computer's QuickTime VR technology, in which cylindrical panoramas are projected to a user-specified viewing plane in real-time. McMillan and Bishop [1995a] introduced the plenoptic function [Adelson and Bergen, 1991] in the context of computer graphics to provide a comparison framework. The plenoptic function is a seven dimensional function that describes the pencil of light rays visible from any point in space at any time and in any wavelength. A parameterized version of the function is given by:

$$p = P(\theta, \phi, \lambda, V_x, V_y, V_z, t) \tag{1.1}$$

where p is the radiant energy arriving at point (V_x, V_y, V_z) from direction (θ, ϕ) at time t and wavelength λ. The plenoptic function provides a convenient method to describe the reconstruction capabilities of various IBR algorithms.

1.2.2 Image-Based Rendering by Warping

McMillan and Bishop [1995a] presented a new image-based rendering system in which panoramic images are projected onto cylinders, pixel correspondences between cylinders are computed, and novel views between cylinders are generated. They also introduced generalized 3D warping functions that compute new pixel locations for images of arbitrary viewpoints [McMillan and Bishop, 1995b]. These functions were formalized by McMillan [1997] and form the basis of Image-Based Rendering by Warping (IBRW) algorithms.

Mark et al. [1997] utilize IBRW to create in-between frames for an interactive walkthrough system. They render key frames synthetically using traditional CGI to serve as reference images. Since IBRW is a much faster process, the reference images are warped to generate current views while the next reference location is predicted and rendered. This allows the system to achieve 60Hz rendering rates.

Shade et al. [1998] introduce *layered depth images* (LDIs), images whose pixel locations contain multiple color-depth pairs. Layered depth images can be warped to new viewpoint locations using IBRW. The multiple pixel layers help to reduce disocclusions, or "tears" in a re-sampled image resulting from lower sampling rates than in the original image.

A number of papers use LDIs and IBRW to improve the efficiency of larger systems. Popescu et al. [1998] use LDIs and IBRW to optimize architectural walk-throughs of large scene databases. They divide their scenes into cells (rooms and hallways) and portals (doors and windows). To compute an image of an arbitrary viewpoint in the scene, they render all geometry that is contained within the same cell as the viewpoint, but they use pre-computed LDIs for portals of the same cell.

Chang et al. [1999] use a hierarchical space partitioning scheme, the *LDI tree*, to efficiently render from arbitrary viewpoints. They place an octree around the scene and construct LDIs at leaf nodes by warping the reference images to the leaf node viewpoint. The depth of each path in the octree is set so the sampling rate of the LDI matches that of the reference images.

McAllister et al. [1999] present an end-to-end system for acquiring depth images using a custom-built laser range device. The images are preprocessed to identify potential disocclusion artifacts and tiled to facilitate real-time rendering. Tiles are rendered using 3D warping on custom-built rendering hardware.

1.2.3 Database-Driven IBR

Another class of IBR algorithms consists of storing in a large database hundreds or thousands of images taken from extremely close viewpoints, and retrieving selected pixels of each image to reconstruct novel views. Gortler et al. [1996] construct a *lumigraph* by reducing the plenoptic function to four dimensions—two position parameters and two direction parameters. They only consider rays leaving the convex hull of a bounded object, and parameterize the rays as 2D coordinates of two parallel planes facing the object. Using a calibrated camera, the authors capture images of an object sitting on a capture stage and "rebin" the source pixels to produce a uniform sampling of the object. A dense array of these images is used to reconstruct novel views of the object. Applying approximated geometric information to the reconstruction process mitigates ghosting artifacts arising from finite sampling rates.

In a closely related paper Levoy and Hanrahan [1996] propose *light field rendering*. The authors again reduce the plenoptic function to four dimensions, but they do so using a regular dense array of compressed images. Novel views are rendered by sampling the 4D function at locations coinciding with the sampling rays of the desired image, interpolating when necessary. Unlike the lumigraph, light fields do not use approximate geometry to reduce sampling artifacts, relying instead on sampling theory to sufficiently blur incorrect details. Isaksen et al. [2000] show how to reparameterize light fields to accomplish effects such as variable focus and depth of field, and passive autostereoscopic viewing.

Another database-driven IBR technique, *concentric mosaics* [Shum and He, 1999], consists of acquiring a set of concentric panoramas by rotating a slit camera along off-center circles. Rendering novel views of the free space within the concentric boundary consists of determining the column of pixels for each column of the desired view from the correct cylinder image and column within that cylinder (or interpolating from nearby neighbors). Vertical distortion can be alleviated by assuming depth information for pixels, columns, or cylinders. A latent network-based solution to concentric mosaics is given by Li et al. [2001].

Plenoptic stitching [Aliaga and Carlbom, 2001] provides a 4D parameterization of the plenoptic function to facilitate IBR walkthroughs constrained to a horizontal plane and fixed vertical viewing angle (always looking straight ahead). Omnidirectional video footage through the environment is stored in a database. Closed loop subsets of the footage are then used to render novel viewpoints within the loop. Neighboring loops are said to be “stitched” because there are no visual artifacts as viewpoints move from one loop to another.

1.2.4 Other IBR Techniques

A few other IBR techniques illustrate the variety, scope, and possibility of IBR as a rendering medium. Debevec et al. [1996] present *view-dependent texture mapping* which combines images from widely spaced viewpoints into textures for rendering recovered models of the image contents. During the rendering phase the texture maps are blended based on the gaze angle of the new viewpoint.

Buehler et al. [2001] describe an approach to generalize current IBR algorithms. Their *unstructured lumigraph* algorithm relies on a camera blending field to blend images of a scene taken from arbitrary locations. Because the cameras can be closely-spaced or widely-separated, their algorithm generalizes the two approaches of view-dependent texture mapping and light field rendering, and smoothly transitions between these two extremes. They also contribute a set of listed goals for IBR, which their algorithm was directly designed to meet.

Recently there has been interest in approximating the plenoptic function on the surface of objects, reducing its dimensionality from seven to six dimensions. Wood et al. [2000] introduce *surface light fields*, functions that assign a color to each ray originating from the surface of an object. Surface light fields outperform other IBR methods on objects with highly specular BRDFs (Bidirectional Reflectance Distribution Functions).

1.3 Image-Based Modeling

Image-based modeling (IBM) is a closely related field to image-based rendering. While image-based rendering is concerned with synthesizing novel views of existing images, the goal of image-based modeling is to generate 3D models from image data. These models can then be used in a traditional CGI rendering pipeline alongside synthetic CGI elements.

In one of the earliest forms of IBM Debevec et al. [1996] employed *photogrammetric modeling* to create a simplified model of a photograph's contents. In their system the user manually associates straight line segments in the photograph with edges of geometric primitives that will eventually define the scene geometry. A minimization procedure then finds the camera calibration and geometric primitive scales that best project the line segments onto the edges.

The *Tour Into the Picture* [Horry et al., 1997] algorithm consists of a user-specified scene model, including a vanishing point, imposed on a photograph. The photograph is warped to new views based on the model to simulate navigation within the photograph. This is considered image-based modeling rather than rendering because an actual geometric scene representation is derived.

More recently, Rusinkiewicz et al. [2002] proposed a method to iteratively acquire the geometry of a hand-held object. They project structured light patterns onto the object and compute range images from the projections. A user slowly turns the object to expose all sides, and the range images are merged in real-time to form a 3D model. This method allows the user to observe holes or deficiencies in the model and move the object to fill them.

Image-based modeling is an exciting and active research area. Unfortunately current IBM methods rely on user intervention to help construct the object geometry.[3] A feature of this dissertation is to meet its goals automatically, with no user intervention. We therefore focus our research on image-based rendering approaches.

1.4 View Interpolation

Given the ability of image-based rendering to generate images of arbitrary viewpoints, an obvious goal is to extend the simulation beyond a few photographs to

[3] An active area of computer vision research is *structure from motion* which attempts of retrieve scene geometry from photographs with no user intervention.

hundreds of photographs of large environments. In a recent attempt, Uyttendaele et al. [2004] show an interactive system in which omnidirectional video footage is captured and processed offline. During simulation time, a user can walk through a virtual environment along the set paths of the captured video, and can change the viewing direction at any point along a path.

Because the user of this system is essentially viewing stored video footage, this method is a database solution to image-based rendering. Database solutions such as this, and including lumigraphs, light fields, and concentric mosaics, are size-limited. Comparing *view interpolation* [Chen and Williams, 1993] to *light field rendering* (and assuming that light fields were 5D rather than 4D to give it similar navigational ability), if the horizontal sampling rate of a light field is n times that of view interpolation, its storage requirements are n^3 times more than view interpolation. This severely restricts the ability of database-driven IBR to handle large-scale environments. For this reason, we choose view interpolation methodologies.

Chen and Williams [1993] first introduced view interpolation, but interpolation by image warping was formally generalized by McMillan [1997]. McMillan gives a set of equations that will map a pixel from one image to a location of another image with a user-specified viewpoint. The derivation of the warping equation helps us to understand both the power and the limitations of IBRW.

1.4.1 Derivation of Warping Equation

The first step in the derivation is to define a basis for the pinhole camera model used in image acquisition. Figure 1.3 illustrates graphically the vectors used to form the basis. When an image coordinate (u, v) is projected onto the basis $\mathbf{P}$, the result is a vector $\bar{d}$ from the camera center to the location of the coordinate in world space:

$$\bar{d} = \begin{bmatrix} d_i \\ d_j \\ d_k \end{bmatrix} = \begin{bmatrix} a_i & b_i & c_i \\ a_j & b_j & c_j \\ a_k & b_k & c_k \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{1.2}$$

To construct the warp equation, two camera bases are related in terms of a

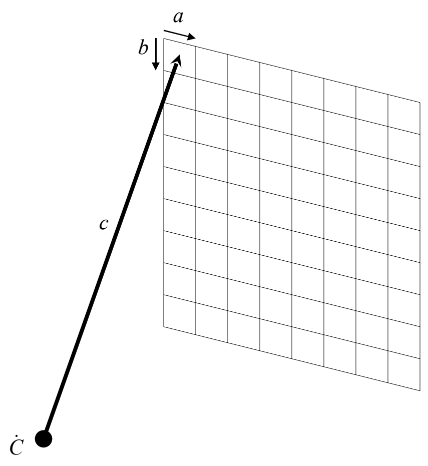


Figure 1.3: A pinhole camera model is adopted for IBRW. $\dot{C}$ is the camera location, a and b are vectors defining the unit directions of the image axes, embedded in world space, and c is the vector from the camera center to the image origin in world space. A computational basis is formed as the matrix $[\bar{a}\ \bar{b}\ \bar{c}]$.

common world space point $\dot{X}$, as in Figure 1.4. Let

$$\bar{x}_1 = \begin{bmatrix} u_1 \\ u_1 \\ 1 \end{bmatrix}, \quad \bar{x}_2 = \begin{bmatrix} u_2 \\ u_2 \\ 1 \end{bmatrix}. \tag{1.3}$$

Then given two camera bases $\mathbf{P_1}$ and $\mathbf{P_2}$,

$$\dot{X} = \dot{C}_1 + t_1 \underbrace{\mathbf{P_1}\bar{x}_1}_{\text{ray}_1} = \dot{C}_2 + t_2 \underbrace{\mathbf{P_2}\bar{x}_2}_{\text{ray}_2}. \tag{1.4}$$

Equation 1.4 employs simple vector addition to point from the camera center to the world space point $\dot{X}$, equating the expressions from both bases. Rearranging terms yields

$$\frac{t_2}{t_1}\mathbf{P_2}\bar{x}_2 = \frac{1}{t_1}(\dot{C}_1 - \dot{C}_2) + \mathbf{P_1}\bar{x}_1 \tag{1.5}$$

where t_1 and t_2 are scaling factors for the rays emanating from the camera centers. The ratio $\frac{t_2}{t_1}$ is now dropped from the equation, making the Euclidean ray $\mathbf{P_2}\bar{x}_2$ a projective entity, and as such it has an unknown but non-zero scaling factor. This

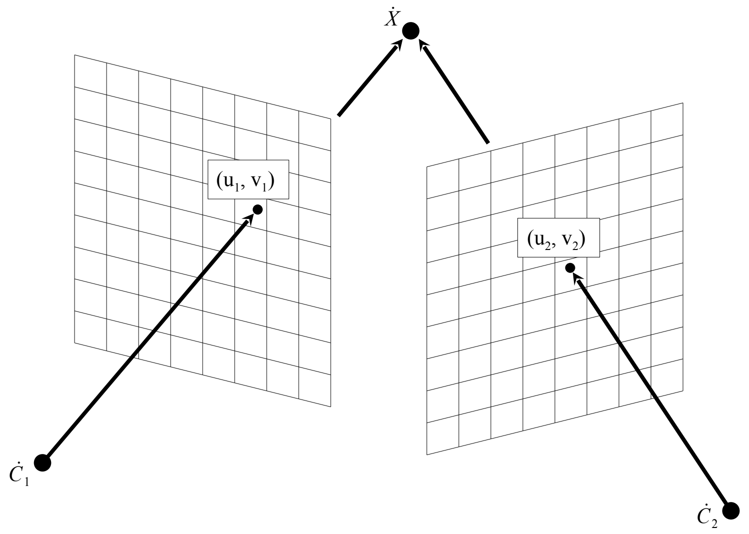


Figure 1.4: To derive the warp equation, two camera bases pointing to a common 3D location $\dot{X}$ are equated.

is consistent with the projective space of the pinhole camera model where image coordinates are actually homogeneous coordinates, having arbitrary scale. The term $\frac{1}{t_1}$ is known as generalized disparity, equivalent to one over the depth at a given pixel, and is referred to as $\delta(u, v)$.

Using basic linear algebra, the equation can now be rearranged to form the warp equation:

$$\begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \bar{a}_1 \cdot (\bar{b}_2 \times \bar{c}_2) & \bar{b}_1 \cdot (\bar{b}_2 \times \bar{c}_2) & \bar{c}_1 \cdot (\bar{b}_2 \times \bar{c}_2) & (\dot{C}_1 - \dot{C}_2) \cdot (\bar{b}_2 \times \bar{c}_2) \\ \bar{a}_1 \cdot (\bar{c}_2 \times \bar{a}_2) & \bar{b}_1 \cdot (\bar{c}_2 \times \bar{a}_2) & \bar{c}_1 \cdot (\bar{c}_2 \times \bar{a}_2) & (\dot{C}_1 - \dot{C}_2) \cdot (\bar{c}_2 \times \bar{a}_2) \\ \bar{a}_1 \cdot (\bar{a}_2 \times \bar{b}_2) & \bar{b}_1 \cdot (\bar{a}_2 \times \bar{b}_2) & \bar{c}_1 \cdot (\bar{a}_2 \times \bar{b}_2) & (\dot{C}_1 - \dot{C}_2) \cdot (\bar{a}_2 \times \bar{b}_2) \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ 1 \\ \delta(u_1, v_1) \end{bmatrix} \quad (1.6)$$

Given two camera centers $\dot{C}_1$ and $\dot{C}_2$ and their corresponding bases $[\bar{a}_1\ \bar{b}_1\ \bar{c}_1]$ and $[\bar{a}_2\ \bar{b}_2\ \bar{c}_2]$, an image coordinate (u_1, v_1) in image 1 having disparity $\delta(u_1, v_1)$ is mapped in image 2 to the location (u_2, v_2).

1.4.2 Warping Equation Implications

This analysis provides us with some insight regarding view interpolation via IBRW. Given an accurate camera model for a photograph and per-pixel generalized disparity, we can warp the image to resemble what would be seen from any viewpoint in the same world space as the original. This is a powerful capability that could be exploited for large-scale environment visualization.

A difficulty arises from the fact that the images are point-sampled, and the warp equation operates on individual pixels. As points are transformed to new locations, sampling irregularities occur resulting in un-sampled gaps in the final image that need to be filled to avoid rendering artifacts. Another problem that occurs is related to the scale-independence of the homogeneous image coordinates. Since the $\frac{t_2}{t_1}$ term was removed from the warp equation, there is no notion of depth for the warped pixel. Thus depth order cannot be resolved in the warped pixels from the equation alone. McMillan [1997] provides a solution by imposing a warp order on the source image pixels, referred to as *occlusion compatible ordering.* However, no solution exists for warping more than one image to the same camera basis while maintaining an occlusion compatible ordering for all pixels. This precludes the use of many images in an IBRW simulation.

Unfortunately, there is a much larger problem with using IBRW for large environment simulation—obtaining the requisite parameters $\dot{C}$ and δ in the warp equation. Since our intent is to utilize many photographs in our view interpolation, we must determine the camera center $\dot{C}_i$ in a common space for each photograph, and the generalized disparity $\delta(u_i, v_i)$ for each pixel of each photograph. The image-based rendering literature is scarce on suggesting methods to obtain these quantities. However, we found the field of computer vision rich with viable solutions.

1.5 Computer Vision

Researchers in computer vision have been investigating the problems mentioned above for a long time. The problem of determining the camera parameters is known as camera calibration, and the problem of finding per-pixel depth is equivalent to the dense correspondence problem in computer vision. We will briefly outline the state of the research for each of these problems.

1.5.1 Camera Calibration

Much of computer graphics and vision research involves simplifying the sampling device to a pinhole camera model. This model consists of an infinitely small focal point (the pinhole) through which all light rays emanating from a scene pass to strike the imaging surface (film or CCD array). In this model all points in the scene are in perfect focus. The parameters of the model are of two classes—*intrinsic* or internal parameters defining the optical properties of the camera, and the *extrinsic* or external parameters defining the geometric properties. The intrinsic parameters consist of a principle point (p_x, p_y), which is the center of projection in the image, the focal length f, the number of pixels per unit distance in image coordinates (m_x, m_y), and a skew parameter s. The extrinsic parameters consist of a translation and rotation to place the camera in the same space as the scene. The extrinsic parameters are sometimes called the *pose* of the camera.

Collectively the intrinsic parameters are represented by a calibration matrix $\mathbf{K}$:

$$\mathbf{K} = \begin{bmatrix} m_x f & m_x s & m_x p_x \\ 0 & m_y f & m_y p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1.7}$$

The external parameters are represented by a 3×3 rotation matrix $\mathbf{R}$ and a translation vector $-\mathbf{C}$ where $\mathbf{C}$ is the camera location in scene space. Taken as a linear transform, the calibration parameters map 3D points to the 2D image plane of the camera:

$$\mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{R} & -\mathbf{RC} \end{bmatrix} \mathbf{X} \tag{1.8}$$

where $\mathbf{X}$ is a scene point $(x, y, z, 1)^\mathsf{T}$ in homogeneous coordinates and $\mathbf{x}$ is the projected point $(x, y, w)^\mathsf{T}$ in image space.

1.5.2 Intrinsic Calibration

Camera calibration consists formally of determining both intrinsic and extrinsic camera parameters for a given 3D scene and an image of that scene. However in practice one usually refers to calibration as finding the intrinsic parameters only, and the extrinsic parameters (camera pose) are found later in a separate step. Intrinsic calibration parameters can be estimated from the contents of an image if 2D image coordinates can be associated with known 3D scene points, and there are very stable algorithms for performing this estimation. A common technique is to photograph a *calibration object*, which is often a checkered planar surface or box, and then to either manually or automatically associate a number of image points with their corresponding points on the object. Intrinsic parameter estimation can then be performed using methods such as that proposed by Zhang [2000]. These parameters are equivalent to the camera basis $\mathbf{P}$ in Equation 1.2, if the basis were embedded in its own local space rather than world space.

In computer vision the pinhole camera model is expanded slightly to account for radial distortion, the non-linear warping of pixel samples along radial lines intersecting the principle point, and vignetting, the darkening of a photograph at its extremities when taken with a short focal length. Both artifacts are caused by finite apertures and imperfections in real lenses, and are corrected for prior to camera calibration.

1.5.3 Extrinsic Calibration

The problem of extrinsic calibration, or pose estimation, is closely related to intrinsic calibration. Solutions in the literature are available for a single image, for two images (stereo), or for three or more images (multi-view) [Hartley and Zisserman, 2004]. A typical stereo vision solution amongst practitioners is to establish feature correspondences between two images, such as points in the 3D scene that are common to both photographs, and to use the features as constraints on a set of linear equations to compute the *essential matrix*, a 3×3 matrix of rank 2 representing the geometric relationship between camera poses. The rotation and translation of the second camera relative to the first can then be extracted from the essential matrix using factorization; for more details see Longuet-Higgins [1981], Huang and Faugeras [1989], Hartley [1997], and Ma et al. [2004]. The camera pose estimates could be used in IBRW as parameters to the warp equation: the camera positions are the camera centers $\dot{C}_1$ and $\dot{C}_2$ from Equation 1.4, and the camera rotations transform the camera bases $\mathbf{P_1}$

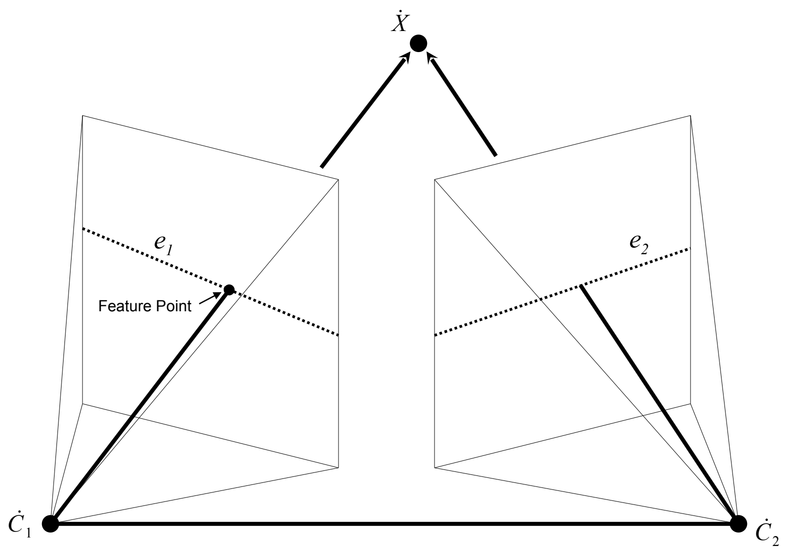


Figure 1.5: When two images exist of a 3D object, features in one image are searched in the second image along epipolar lines. In this figure the camera locations $\dot{C}_1$ and $\dot{C}_2$ together with the 3D point $\dot{X}$ define an *epipolar plane*. The intersection of the epipolar plane with both images form epipolar lines e_1 and e_2 respectively, shown as dotted lines in the figure. When a feature point in image 1 is searched for in image 2, the search is constrained along e_2 because even though the exact position of $\dot{X}$ is not known, its projection in image two is guaranteed to be on e_2.

and $\mathbf{P_2}$ in Equation 1.4 from their local spaces to a common world space.

1.5.4 Dense Correspondence

The dense correspondence problem for finding per-pixel depth is a special case of the general *correspondence problem*: given an image and a geometric or photometric feature in the image such as a point, line, curve, area, texture, or abstracted object, find the same feature in another image—essentially a search problem. In a stereo image configuration where two photographs have been acquired of the same 3D scene or object, features in a reference image are searched for in a query image along epipolar lines, as in Figure 1.5. If the two images are stereo rectified, their corresponding epipolar lines fall along identical scan lines in the image, reducing the search to horizontal image rasters.

Two important criteria to the problem are the choice of feature representation and the choice of an objective function whose minimum indicates a correct feature match between images. For instance, points are commonly represented by a window of pixels surrounding the image point to be found, and a typical objective function to compare two such windows for similarity is to accumulate the squared differences or normalized cross-correlation of individual pixels between windows. A more recent trend is to represent feature points in an image with affine- and photometrically-invariant region descriptors [Tuytelaars and Van Gool, 2000; Mikolajczyk and Schmid, 2002; Lowe, 2004] and to compare the descriptors using a scale-invariant metric such as the Mahalanobis distance.

Dense correspondence is simply finding point feature matches for most or all pixels between two or more images. The correspondence problem, be it for a sparse or a dense set of features, is challenging for several reasons:

1. The objects projecting to pixels in one image may not even be present in another image, or they may be occluded.

2. Photometric differences between images complicate comparison; this could happen when photographs were taken at different times, at largely different angles, or with two different cameras.

3. Manifestations of the *aperture problem* occur because of the finite-sized acquisition instrument (the aperture in a camera for instance). Examples include images with repeated patterns, images with little high-frequency detail such as photographs of a blank wall, or images with sampling rate differences such as two photographs taken with different focal lengths (zoom factors).

Despite these difficulties in solving the correspondence problem many advances have been made. Scharstein and Szeliski [2002] provide an excellent survey on various dense correspondence algorithms, and Sun et al. [2005] show a recent contribution.

1.5.5 Structure From Motion

With a correct camera calibration and dense correspondence, computing per-pixel depth for each pixel of each image in an image set is straightforward, and

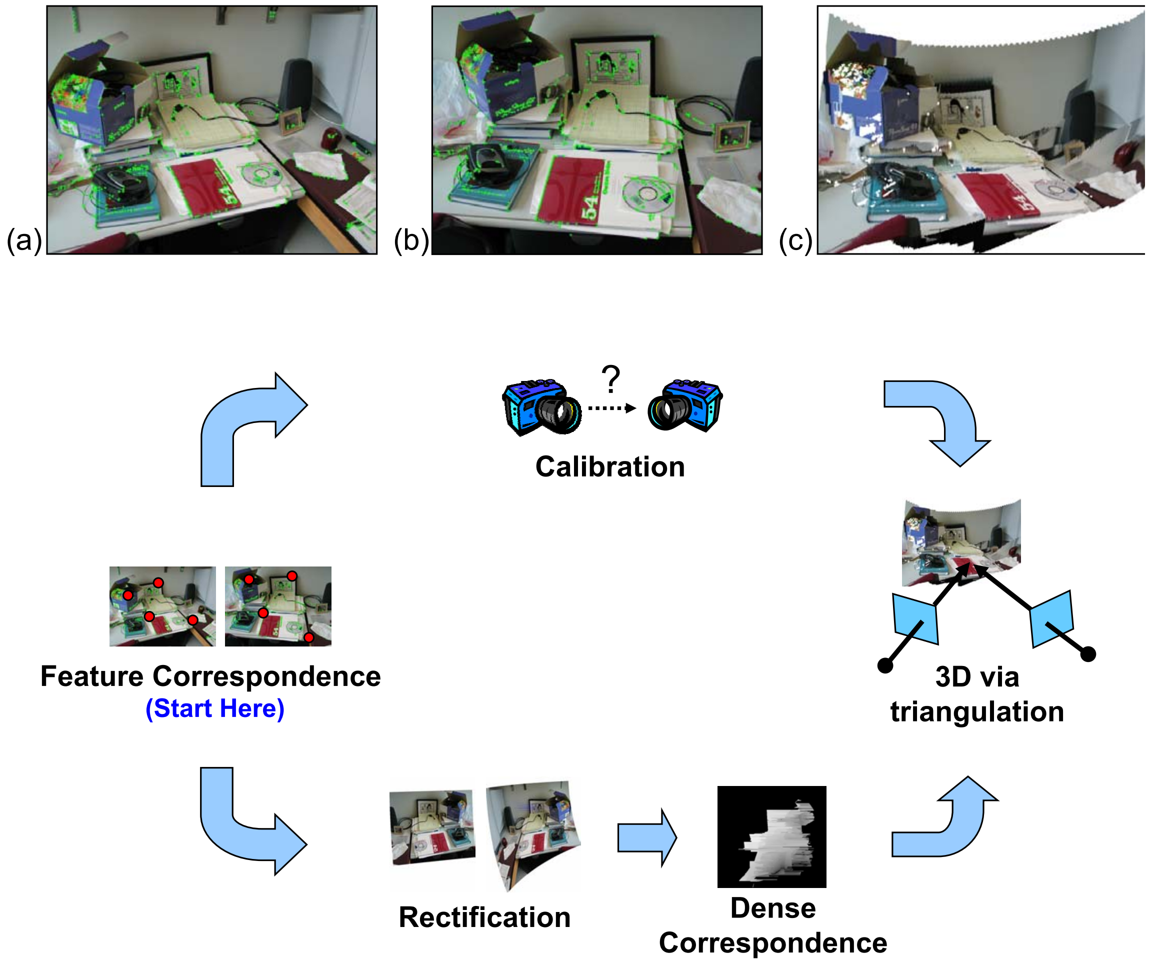


Figure 1.6: A structure-from-motion pipeline to re-create the 3D structure that is seen in a pair of photographs. The top row of images shows two photographs (a) and (b) of a desktop environment, taken from slightly different positions. Image (c) shows the derived 3D structure re-projected to a lower vantage point. The process begins by identifying a sparse set of point correspondences between images (a) and (b). Several hundred point correspondences are usually necessary in each image, and four correspondence examples are marked in the diagram at the "Start Here" label. Having obtained the sparse set of correspondences, the process diverges into two sub-processes: camera calibration and image rectification followed by dense correspondence estimation. The set of sparse correspondences is used in both sub-processes to solve the problems at these stages. In the camera calibration stage the camera pose of image (b) is estimated relative to image (a). In the rectification stage both images are re-sampled to align their epipolar lines with common image rasters, and then the dense correspondence for each pixel in image (a) is estimated. Given calibration and dense correspondence, a 3D point structure is finally estimated via triangulation of each pixel pair.

the original 3D locations of image features can be inferred. This process is known in the computer vision literature as *structure-from-motion*, and coincides with the image-based modeling goals stated in Section 1.3. Figure 1.6 illustrates one possible computer vision "pipeline" that will take images as input and give an estimate of the 3D structure as output. This process is harmonious with the goals of this dissertation, and we adopt this methodology as our modus operandi for creating the content for virtual reality simulations.

Unfortunately, there exist severe limitations in the robustness of the algorithmic solutions to the calibration and correspondence problems. These limitations ultimately stem from the specific problems listed in Section 1.5.4. For instance, if the initial set of sparse correspondences is corrupted by noise or non-linear inaccuracies, then the camera pose estimate will be incorrect to some degree, which in turn litters the estimated 3D structure with inaccurate geometry (a close look at image (c) of Figure 1.6 reveals many such artifacts). While camera pose estimation is largely robust for two or three cameras, the aforementioned limitations preclude the use of current computer vision methods for the pose estimation of extremely large camera sets and thus restrict our goal of real environment simulation.

1.6 Dissertation Contribution

The purpose of this dissertation is to contribute theoretic and algorithmic research elements to the fields of computer vision and image-based rendering in order to advance the goal of completely automating geometric content creation for virtual reality simulations of existing environments. This specifically includes addressing the deficiencies in camera pose estimation for large collections of photographs, and proposing a novel image-based rendering method to accommodate the calibrated camera network. To frame and validate our research contributions we also develop an end-to-end system that will actualize the workflow proposed in Section 1.1 for a limited class of environments. Due to our heavy use of and contributions to computational stereo,[4]

[4] "Computational stereo is broadly defined as the recovery of the three-dimensional characteristics of a scene from multiple images taken from different points of view."— Barnard and Fischler [1982]

we have termed our system *Vision-Based Rendering*, emphasizing its image-based rendering roots and its indispensable reliance on computer vision.

1.6.1 Chapter Descriptions

Chapters 2–5 are papers that have been published or are currently submitted for review. Each paper proposes solutions to a specific need that will advance the achievement of our stated goals.

Chapter 2, "Correspondence Expansion for Wide Baseline Stereo," addresses the unreliability of camera pose estimation when computing the pose from few point correspondences. In this chapter we outline an algorithm to expand the number of point correspondences between two images while still maintaining an accurate epipolar geometry[5] between the images.

Chapter 3, "Histogram Matching for Camera Pose Neighbor Selection," discusses the need for a camera adjacency graph when dealing with hundreds of related photographs of an environment. The adjacency graph is an essential data structure for the task of computing pose estimates for many photographs. We present methods to construct the adjacency graph based on histogram distances.

Chapter 4, "Minimum Spanning Tree Pose Estimation," gives an optimal method to compute the camera pose estimates for a large set of input photographs. The pose order is based on a traversal of the MST of the adjacency graph, and we propose an effective cost function that measures the accuracy of the 3D reconstruction of an image pair.

Chapter 5, "Omnidirectional View Interpolation of Unstructured Photographs," is our contribution to image-based rendering. In this chapter we present our rendering solution that provides six degrees of navigational freedom to a user when exploring the virtual rendition of an existing environment. Given the pose estimation results of the previous chapters, we create omnidirectional images at the node points of a Delaunay decomposition of the viewpoint convex hull. The omnidirectional images

[5] The epipolar geometry is a rigid-body constraint between two images that guarantees a projective transform between their respective camera poses.

are interpolated using a non-linear Barycentric weighting to produce novel viewpoints of the input set.

In Chapter 6 we describe our system as a whole and discuss its performance and storage considerations, the limited class of environments for which it works well, and the conditions under which it fails. We also discuss the specific contributions of the dissertation, and offer conclusions to the body of research and observations of future directions to take with this project.

Chapter 2

Correspondence Expansion for Wide Baseline Stereo

Kevin L. Steele and Parris K. Egbert, "Correspondence Expansion for Wide Baseline Stereo," In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Vol I, pp.1055-1061, 2005.

Abstract

We present a new method for generating large numbers of accurate point correspondences between two wide baseline images. This is important for structure-from-motion algorithms, which rely on many correct matches to reduce error in the derived geometric structure. Given a small initial correspondence set we iteratively expand the set with nearby points exhibiting strong affine correlation, and then we constrain the set to an epipolar geometry using RANSAC. A key point to our algorithm is to allow a high error tolerance in the constraint, allowing the correspondence set to expand into many areas of an image before applying a lower error tolerance constraint. We show that this method successfully expands a small set of initial matches, and we demonstrate it on a variety of image pairs.

2.1 Introduction

Reliable feature matches between wide baseline image pairs are important for many stereo algorithms in computer vision. Typical feature types include points, lines, curves, and textured regions. Correct feature correspondences enable stereo camera calibration and structure-from-motion algorithms, and permit robust estimation of epipolar geometries between two or more images. Epipolar geometries in turn facilitate further feature matching, image rectification, and the finding of dense image correspondences.

Finding a sufficiently large number of correct feature correspondences between image pairs can determine the success or failure of stereo algorithms that rely on plentiful matches. It is therefore important to be able to generate many correct, high confidence matches from images in a reasonable amount of time. While there exist many wide baseline matching algorithms, most address the problem of finding matches independently; fewer use existing matches in a guided search for more [Matas et al., 2002; Pritchett and Zisserman, 1998; Schaffalitzky and Zisserman, 2001].

The most common method for finding point correspondences between two wide baseline images is to first identify points on each image that lend themselves well to matching. Such *interest points* often have characteristics such as high intensity variance and anisotropic texturing in surrounding pixels. Two sets of interest points are then tentatively matched to one another by finding similar feature vectors between points, yielding a set of putative matches. The feature vectors often include correlation measures [Faugeras, 1993], and geometric and photometric invariances [Mindru et al., 1999].

If each image of the pair has N interest points, the matching complexity is $O(N^2)$. Unfortunately, this common N^2 method of finding a set of feature matches will usually result in many mismatches, due mainly to sampling noise, lighting changes, and foreshortening effects. A robust epipolar geometry estimator such as RANSAC [Fischler and Bolles, 1981] is frequently used as a final step to eliminate the outlying matches. However, if there is a high percentage of incorrect matches given to the estimator, RANSAC executes very slowly. Even worse, matches failing the epipolar constraint are simply discarded, greatly reducing the size of the final correspondence set. A contribution of our work is that we exploit the near-correctness of these discarded matches, locally adjusting their positions to conform to the constraint (see Section 2.3.4).

Our goal is to aggressively search for additional matches using existing matches as starting locations, and to make the final correspondence set as large as possible while preserving the accuracy of its member matches.

2.1.1 Related Work

Many robust methods exist to create point matches for wide baseline stereo. Baumberg [2000] finds affine-invariant features by extracting the relative skew, stretch, and rotation from interest point neighborhoods, and matches points with similar image structure. Applying the extracted affine transformation to the sampling window reduces the number of incorrect matches in the final set of correspondences. Tuytelaars and Van Gool [2000] use local affine and photometric invariant features of the points to facilitate matching. An elliptical region surrounding an interest point is examined to find its generalized color moments, which comprise the invariant feature vector. Interest points are matched according to feature vector similarity. Schaffalitzky and Zisserman [2001] find texture region matches using affine and photometrically invariant descriptors. Their method is statistically insensitive to the shape of the region, yielding a more stable match descriptor than point-based matches provide. Mikolajczyk and Schmid [2002] find affine-invariant feature points by first detecting multi-scale Harris corners, and then use these points in an iterative procedure until the points converge to affine invariance. Both the relative scale and the shape of the point neighborhoods are recovered simultaneously.

Matas et al. [2002] find corresponding regions using an improved similarity measure that adds a voting scheme to the commonly-used Mahalanobis distance. Their method also improves on the large-scale invariance of Mindru et al. [1999]. Ferrari et al. [2003] propose a method to utilize multiple images (more than two) to establish point correspondences between all images. We focus on using only two images, and our method for correspondence expansion is independent of the type of comparison function used to score the fitness of a match. In fact, any method previously used to identify feature matches can be leveraged to initialize our algorithm, and correspondence expansion can easily be appended to any existing matching scheme to increase the number of final matches.

Several methods have been proposed that use known matches to guide the search for additional point matches. Lourakis et al. [1998] find point and line feature correspondences on a common plane by using a randomized search strategy to find an initial set of point and line matches. They then use the derived homography of three lines to verify the point locations and to predict the location of further matches. Their method relies on the presence of planar features, while our algorithm makes no assumptions on geometric properties. Pritchett and Zisserman [1998] compute local homographies at existing matches to guide the search for new matches. They first use existing homographies to predict match locations and then employ a hierarchical approach to create new homographies to carry out additional searches. Our algorithm uses local affine transformations to guide searches, rather than homographies. Matas et al. [2002] improve their number of matched regions by finding affine transformations of correspondences that survive a preliminary RANSAC cull. They then include those portions of regions whose transformed correlation are above a pre-selected threshold in a second RANSAC cull. This roughly doubles or triples the number of correspondences from simply using a single RANSAC prune. A major contribution we make beyond both Pritchett and Zisserman [1998] and Matas et al. [2002] is to apply an epipolar constraint at each iteration, rather than once, as described in Section 2.3.2, enabling many more correct matches to be found.

Ferrari et al. [2004a,b] introduce a method that selects an initial set of feature correspondences as anchors to an iterative exploration of the surrounding image areas. The resulting correspondence set is able to detect matches between images exhibiting non-rigid deformations, and they use this ability as input to an object recognition system. Our proposed method is similar to that of Ferrari et al. in that we iteratively augment and constrain our expanding match set (see Section 2.3). However we employ a novel use of the epipolar constraint (Section 2.3.2) that allows a high growth rate while retaining an approximate epipolar geometry, then refine the final match set to a correct epipolar geometry upon completion. This is in contrast to the method of Ferrari et al. in which they purposefully avoid epipolar constraints in order to permit non-rigid deformations.

Finally, Schaffalitzky and Zisserman [2001] improve on the number of matched regions they find by matching pixels within matched regions. This approach is very successful in generating large numbers of correctly matched points. Their method is limited, however, to isotropic texture regions. In contrast, we are not limited by any statistical property of the input images.

2.1.2 Contribution

In this paper we present a new method that expands an initial set of wide baseline correspondences by an iterative two-step process. We do not propose a new method of establishing initial feature matches. Rather, we propose a novel technique to iteratively grow a set of correspondences outward from a small initial set of matched points.

We use a local affine transform approximation to predict search locations near existing matches. Newly found matches are incrementally added to the correspondence set, and the expanded correspondence set is refined using a high error tolerance RANSAC measure. The result is an expanding set of high confidence correspondences that "grow" outward from existing matches, as shown in figure 2.1. We discuss the details of our algorithm in the next sections. Section 2.2 briefly describes the initialization of the algorithm, Section 2.3 discusses how the process iteratively expands the correspondence set, Section 2.4 shows the results of the method, and we conclude in Section 2.5.

2.2 Initial Correspondence Set

Prior to employing the correspondence expansion algorithm, we must have a potentially small set G of putative matches between two source images I_1 and I_2. These matches need not all be correct. For correspondence expansion to work, at least one match needs to be correct. The more correct matches contained in G, the faster the algorithm will perform. Note that a single initially correct match is a necessary, but not sufficient condition, i.e., given at least one correct match, the

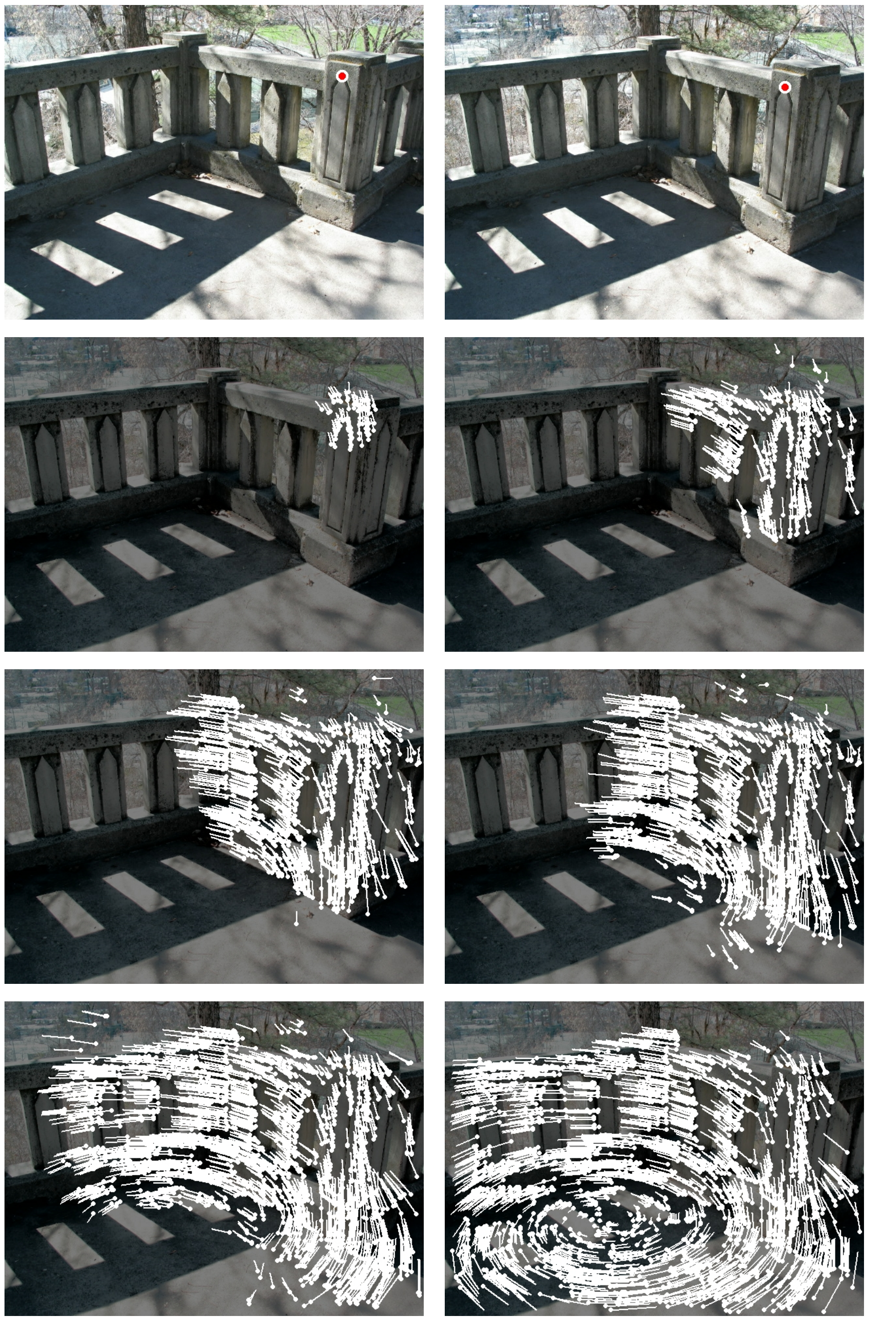


Figure 2.1: An example of correspondence expansion occurring at each step of the iterative algorithm. The top two images of a wall are the left and right images to be matched. One initial point was selected by a user, shown as a red and white circle in the top two images. As matches are found and added to the correspondence set, the matches "grow" outward from the original point, and incorrect matches quickly disappear. Point correspondences are shown as white lines in the image sequence. The last image in the sequence shows the final correspondence set containing 1,925 matches. The lower six images have been darkened to better highlight the correspondences.

algorithm provides no guarantee that matches will be expanded. We have found in practice that this is of no practical concern, since generally there are several correct matches in an initial correspondence set, all of which tend to expand quickly as the algorithm proceeds.

As discussed in the introduction, there have been many wide baseline feature correspondence algorithms proposed over the past several years (see Ferrari et al. [2003] and Goedemé et al. [2004] for references to more algorithms), and any of these could be used to create G. These matches could also be input by a user if desired. While not novel, we briefly mention the initialization procedure we used to create G.

We start by detecting Harris corners [Harris and Stevens, 1988] in each of the two source images I_1 and I_2. We employ the often-used $O(N^2)$ scheme of comparing each corner point detected in I_1 to every corner point detected in I_2. We also determine the relative local rotation between image patches surrounding the corner points, and following Dufournaud et al. [2000] we attempt to match at several resolutions to find a characteristic scale between image regions.

We measure similarity by taking the sum of the squared differences between pixels in the local image region, and assign as matches point pairs with the highest similarity. Since the matching assignment may be a many-to-one mapping, point pairs with the highest similarity are bi-directionally checked, making it an $O(N^3)$ procedure. Matches passing the bi-directional comparison check are finally added to G. While not optimal, for a small number of initial corner points the computation time is negligible when run on a modern processor, exploiting a strength of our algorithm of not needing many initial matches.

2.3 Guided Matching

With a set G of initial matches, we employ our correspondence expansion algorithm to grow the set to include additional matches. The expansion algorithm is iterative and adds matches to the correspondence set at each step. There are two parts to each iterative step: *aggregation* and *constraint*. In the aggregation step, we use the current set of matches as seed points to "grow" additional matches that are nearby, adding the new matches to the current set. In the constraint step, we constrain the newly-enlarged correspondence set to an epipolar geometry, so that when the points in the set are used as seed points in the next iteration, they will have a higher likelihood of growing correct matches. Important to the success of the algorithm is the need for a high error tolerance on the epipolar geometry, as will be explained in Section 2.3.2.

2.3.1 Aggregation

Before beginning the iterative cycle, we detect a set of several thousand Harris corners P_1 in I_1. These serve as interest points which will be matched together with locations in I_2 to form correspondences as the iteration proceeds. We also maintain a current correspondence set C which is initialized to the original match set G.

For each point P_i in P_1, we find the nearest point c_{1i} in C; c_{1i} has already been matched to a point c_{2i} in I_2. To quickly find nearest points, Voronoi maps over I_1 and I_2 are constructed for the points in C. We use graphics hardware to quickly build Voronoi diagrams by rendering cones into two depth buffers, one each for I_1 and I_2 [Kenneth E. Hoff et al., 1999; Woo et al., 1997]. The cones are centered at each point in C and have a finite base. The colors of the rasterized cones determine the identity of the Voronoi regions. In this way Voronoi regions can be looked up from a 2D location in constant time. Figure 2.2 shows an example of the Voronoi regions for an image.

Having found the closest matched point c_{1i} in C to the unmatched feature point P_i, we compute an affine transformation that maps the image region surrounding c_{1i} to the region surrounding c_{2i}. Baumberg observed that small planar surface patches undergo affine transformations when seen from different viewpoints [Baumberg, 2000], and that non-planar smooth patches can successfully be approximated by planar surface patches for correlation. Rather than estimate the whole affine transform, which given the match location in C amounts to finding four parameters, we only consider the local rotation and scale.

Rotation is estimated by using the best correlation from a small set of candidate

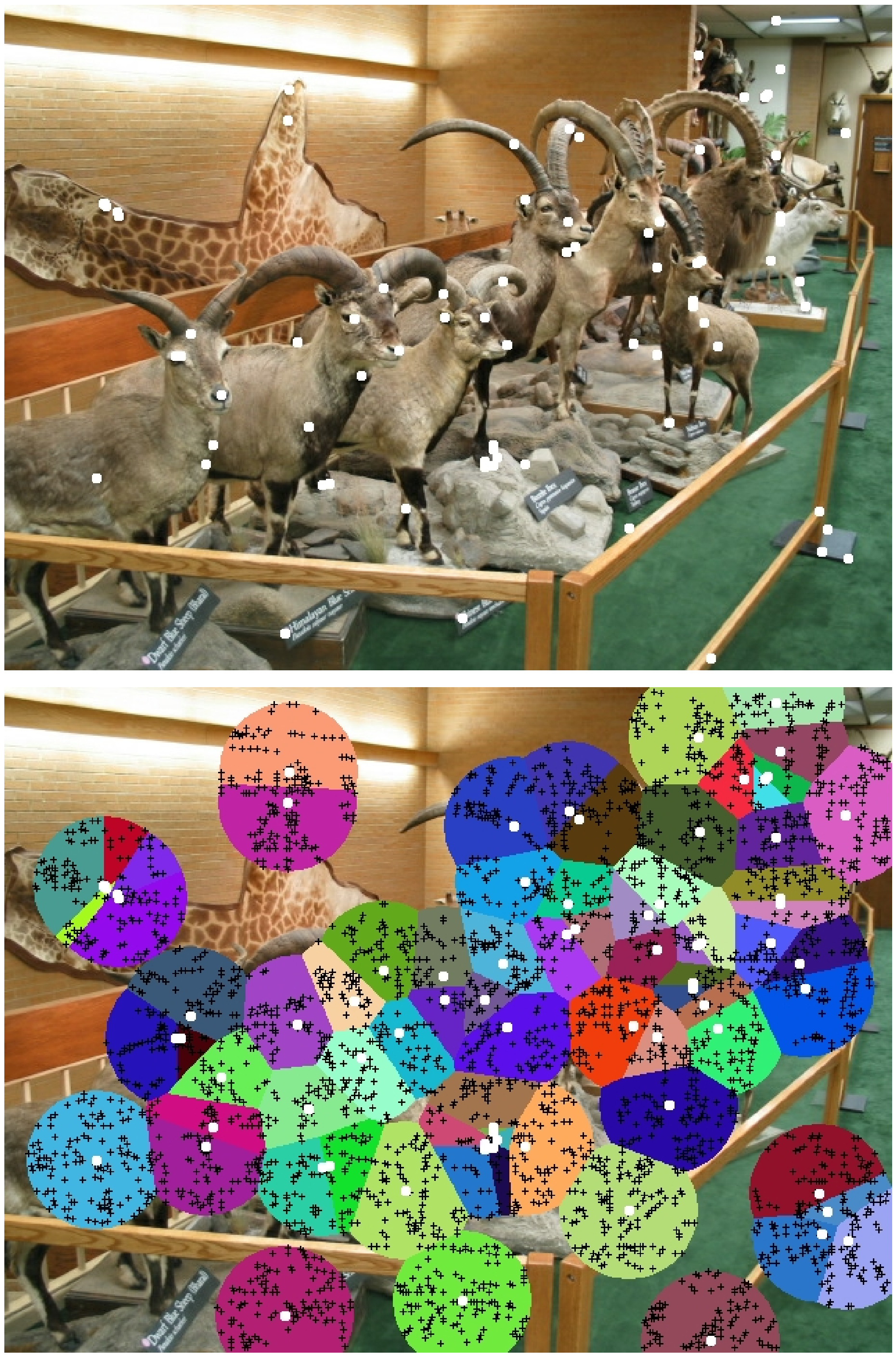


Figure 2.2: Voronoi regions for a set of existing correspondences. The top image of a taxidermy display contains a set of points, highlighted in white, matched to points in another image of the same scene (not shown). The bottom image shows the same set of points surrounded by their corresponding Voronoi regions–each separate region is highlighted in a different shade. The black Harris corners within each Voronoi region will potentially be matched using the local rotation and scale of the nearest existing match.

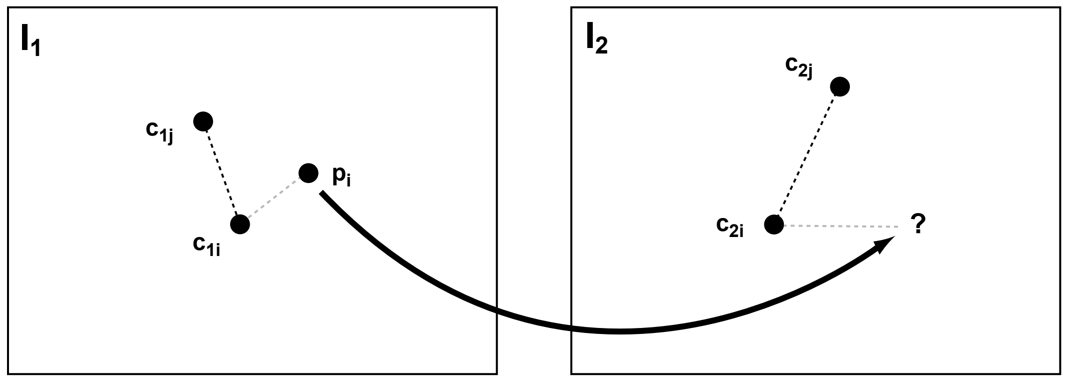


Figure 2.3: The local transform is used to predict new match locations. The feature point p_i was detected using the Harris corner detector, and its closest existing matching point c_{1i} provides a local rotation and scale to guide the search for a match for p_i in I_2. The local rotation is found by maximizing the correlation from a set of candidate rotations. To find the local scale around c_{1i}, its closest match c_{1j} is located, then the scale to transform p_i to I_2 is computed as the ratio of the distances: $|c_{2i} - c_{2j}| \,/\, |c_{1i} - c_{1j}|$.

rotations. A pre-computed lookup table is used to accelerate the rotated locations of each pixel in the window. We could use nearby matches to estimate the rotation, or even the full affine transformation. However, local 2D rotations differ greatly across the image due to the projection of 3D camera rotation, so matches not in the immediate neighborhood of c_{1i} yield incorrect rotations. Local scale is less susceptible to the effects of 3D camera rotation, so we compute it directly from a nearby match. We first find the closest match c_{1j} to c_{1i} in C, then using these two existing matches the local scale is the ratio of their distances (Figure 2.3).

Once the local rotation and scale are estimated, the feature point p_i in I_1 is transformed to a new location p' in I_2 (not necessarily a corner point). A steepest-ascent hill-climbing strategy is used to find the best match in I_2—the correlation at p' is compared with the correlation at all the pixel neighbors, and p' is moved until a local maximum is reached.

With the putative match identified, the whole process is reversed, where the inverse rotation and scale are used to transform p' back to I_1 to predict the original location of p_i in I_1. Again the hill-climbing strategy is used to find the best match p'' in I_1. If the original feature point p_i and the point p'' are within a threshold distance (we use 1 pixel), then the match $[p_i, p']$ is considered valid and added to the set C of current correspondences, and p_i is removed from the set P_1 of feature points.

After all feature points in P_1 have been processed in this manner, C will potentially be much larger. The rate at which matches are added to C depends on the size of the cones determining the Voronoi regions, the number of Harris corners detected within the Voronoi region of each existing match, and the heterogeneity of the texture surrounding the detected corners. In images containing a large portion of high frequency detail, we have found that C increases in size by 50% to 200% at each iteration until the saturation point is reached (discussed in Section 2.3.3).

2.3.2 Constraint

It is imperative that C contain many correct matches, since these are used to seed the growth of additional matches in subsequent iterations. To further ensure that most or all matches in C are correct, they are constrained to an epipolar geometry. It has been well demonstrated that a robust epipolar geometry can be determined from a set of putative matches using RANSAC [Fischler and Bolles, 1981; Ma et al., 2004]. RANSAC is an iterative algorithm, and the number of iterations needed can be automated as shown in Hartley and Zisserman [2004]. The generation of the epipolar geometry using this algorithm also serves to effectively segment, or cull, correct correspondences from incorrect, outlying correspondences. It is for this second purpose that we employ the RANSAC algorithm.

Even though the RANSAC algorithm is robust, care must be taken to avoid too many incorrect matches in the input set, as the number of iterations required will quickly grow very large. For instance, given an input set with an estimated 75% outliers, the number of iterations required to ensure correct segmentation of inliers from outliers with 95% probability will be about 50,000, as given by the following equation from Hartley and Zisserman [2004]:

$$N_{iterations} = \frac{\log(1 - .95)}{\log(1 - (1 - \epsilon)^7)} \quad | \quad \epsilon = .75 \tag{2.1}$$

When generating the consensus sets during RANSAC culling, we intentionally use a high inlier error tolerance—matches within five pixels of their epipolar lines are considered inliers. This is an important aspect of our algorithm. Though it results in a less accurate epipolar geometry, it permits many more matches to be added to C. By including more matches in this way even if they are slightly incorrect, we speed up the inlier/outlier segmentation considerably, giving the algorithm a fast iteration cycle, and more importantly, improving the ability to grow more matches in the next iteration. Thus, at this stage of the algorithm, the set C temporarily contains a large number of incorrect matches due to the high error tolerance. However, as a result, the algorithm is able to create an average of 50% more final correct matches in our test images than it does by using a low RANSAC error tolerance, such as 0.5 or 0.1 pixels. This is because the high error tolerance permits matches to expand into regions of I_1 and I_2 that otherwise would have contained fewer candidate matches.

Inlier matches that survive the epipolar constraint are kept in the set C of current correspondences. Matches that fail are removed from the set, and each point c_{1i} from the failed matches are placed back in the set P_1 of feature points for future matching consideration.

It is important to note that we do not use the epipolar constraint to guide the search for new matches, as it has been used historically. Doing so would potentially contaminate the correspondence set with false matches following an incorrectly estimated epipolar geometry. Rather, as explained in Section 2.3.1, we use an approximated affine transform to guide the search, and we utilize the epipolar constraint to refine the augmented correspondence set along the way.

2.3.3 Saturation

The correspondence expansion iterations are allowed to proceed to a saturation point, when no additional matches are added to C in the aggregation step. This occurs when all the feature points in P_1 either have been matched or have no correlating matches that can be found. We have observed that during the aggregation and constraint phases, the size of C may occasionally drop slightly as the matches it contains shift to a more accurate epipolar geometry. Immediately following such adjustments, the size of C usually increases dramatically since the increased accuracy will admit more correct matches. To permit these desirable fluctuations, we allow the size of C to drop a pre-determined number of times (we use three) before terminating the iteration cycle. This conservative termination criterion tends to allow the correspondence expansion to advance into most areas of I_1 and I_2 that would have been reached by an unbounded number of iterations. Our experience has shown that when the expansion covers most of the image area, succeeding operations such as pose estimation using the point matches are more accurate than when the expansion reaches saturation only on localized regions of the images.

2.3.4 Final Guided Matching

At its saturation point, the current correspondence set C contains many more matches than it did initially; however, the matches do not adhere closely to the epipolar constraint as a result of the high error tolerance permitted earlier in the algorithm. As a final step, we wish to impose a tighter epipolar constraint to ensure a correct set of matches. Unfortunately, applying the constraint would eliminate many matches in C that are close to correct but are far enough away from their epipolar lines to fail the epipolar constraint.

Rather than discard all of these near-correct matches, we adjust their matched positions prior to the final constraint application. We use a simple guided search strategy that, in contrast to Section 2.3.2, does use the epipolar constraint as a guide. Matched points c_{2i} from C in I_2 are projected orthogonally to their corresponding epipolar lines, then their counterpart points c_{1i} are matched to points in I_2 along short segments of the epipolar lines. Those matches with better correlations replace the old matches, while those not having better correlations are discarded. Finally, the set C is constrained to an epipolar geometry with a low error tolerance of .5 pixels

Im#	Initial Matches	Final Matches	Time In Secs.	Iterations
1	53	144	33	11
2	25	189	29	14
3	29	270	20	19
4	37	480	28	17
5	53	531	46	14
6	53	531	55	15
7	35	607	43	13
8	52	661	39	13
9	63	818	37	11
10	98	849	48	11
11	27	871	29	15
12	100	908	42	12
13	46	1208	34	13
14	81	1307	25	10
15	58	1401	36	18
16	109	1916	33	13
17	66	1942	43	17

Table 2.1: Results from the correspondence expansion algorithm, sorted by the number of final matches found. The initial matches were computed using the initialization procedure of Section 2.2. The Final Matches column reports the number of matches found after expansion. The average expansion time in seconds for all images in the table is 36.5 seconds; this time does not include finding the initial match set. The average number of iterations is 14. All tests were run on a 3.2 GHz Pentium 4 CPU.

to create the final correspondence set.

2.4 Results

We tested the correspondence expansion algorithm on images acquired from a variety of indoor and outdoor environments. Starting with a few dozen matches, our algorithm performed extremely well, expanding the correspondence set by approximately 3 to 30 times its original size. We also found that there were very few incorrect final matches.

To measure the accuracy of the matches, we hand-picked a small set of correspondences in each image pair to compute an accurate fundamental matrix **F**. We

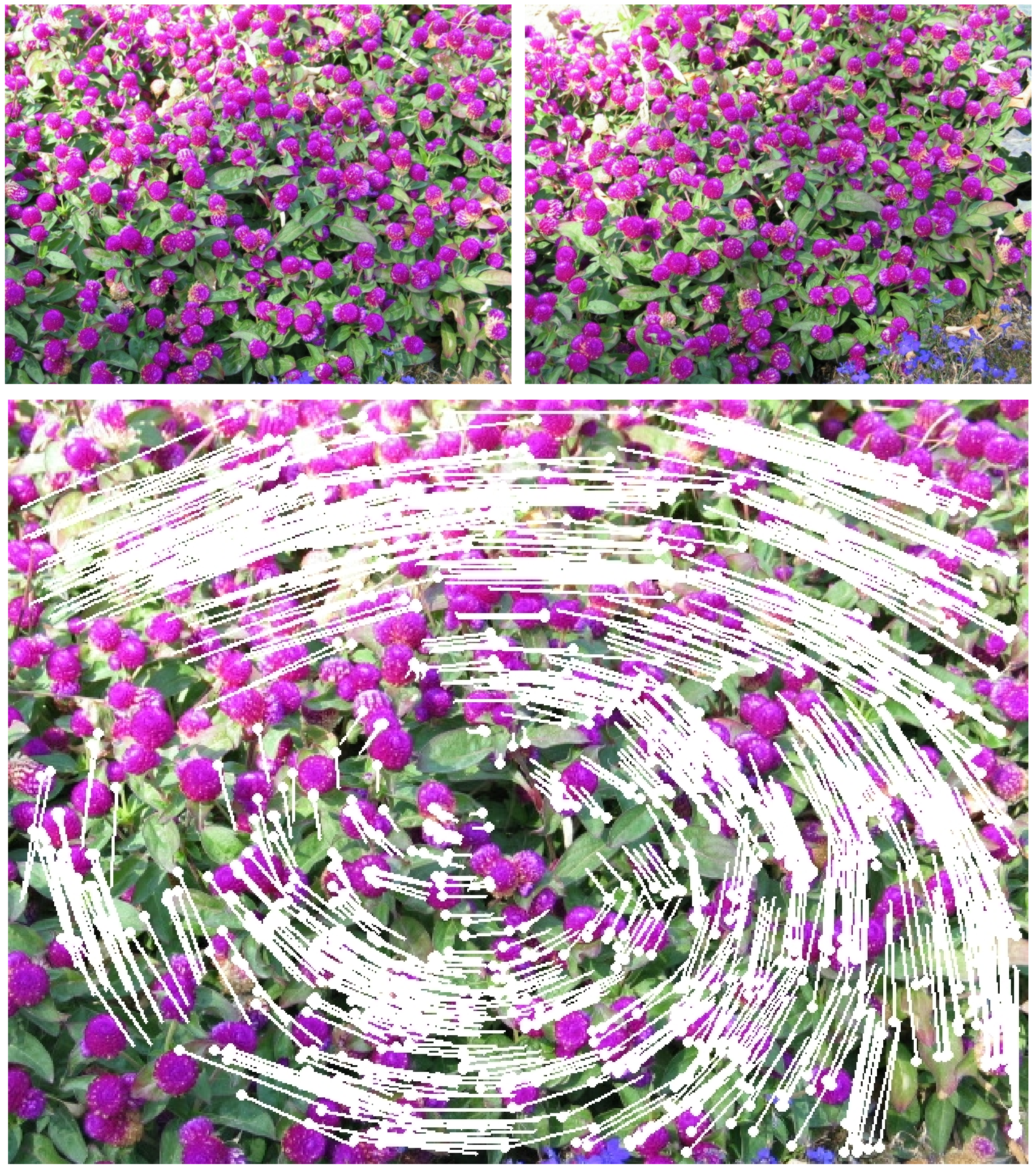


Figure 2.4: Flowerbed image pair. This correspondence set was expanded from 35 initial matches to 931 final matches. The top images are the original pair, and the bottom image illustrates the final correspondences with lines.

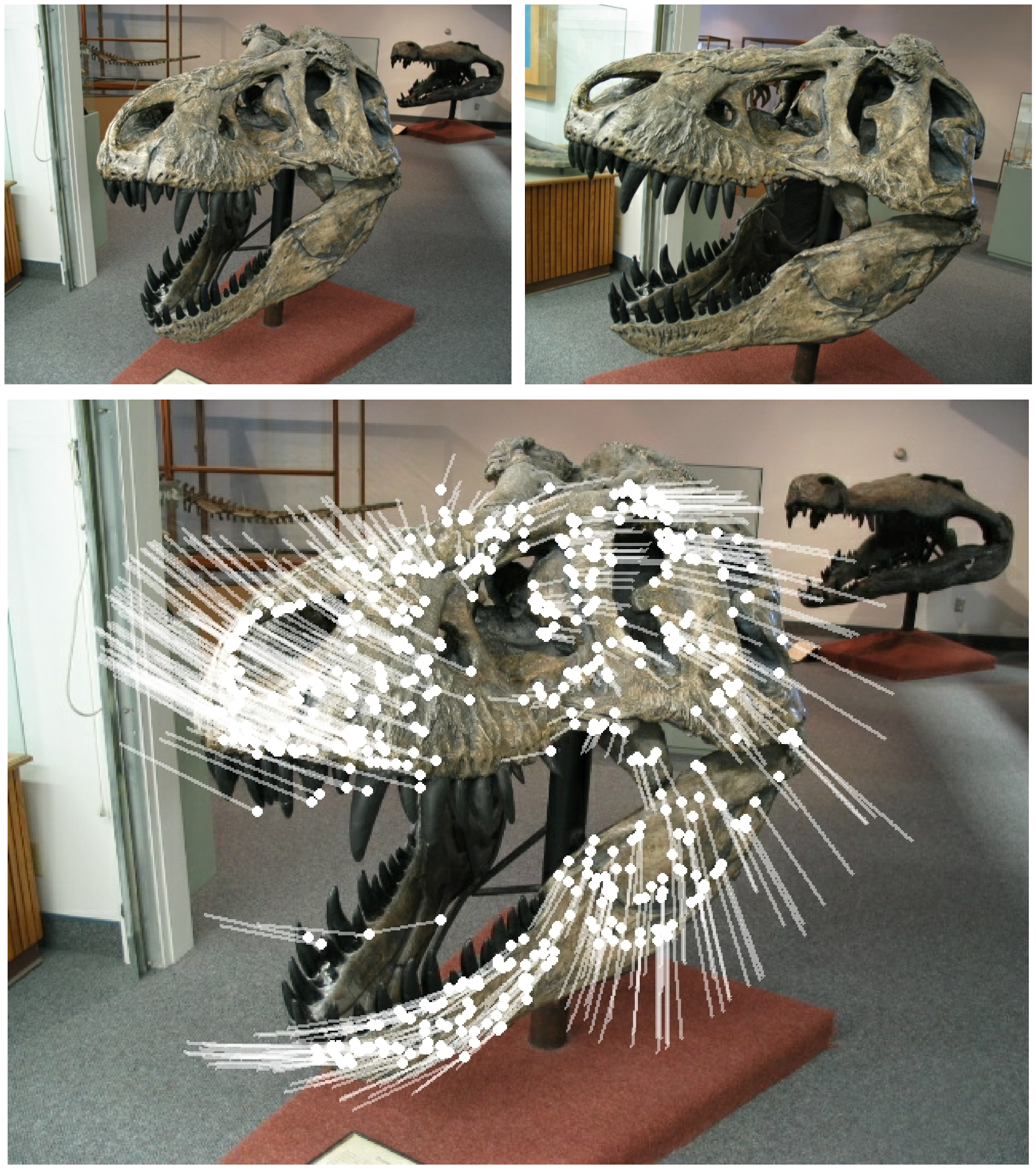


Figure 2.5: T-Rex skull image pair, expanded from 30 initial matches to 532 final matches. Correspondences are shown with white lines.

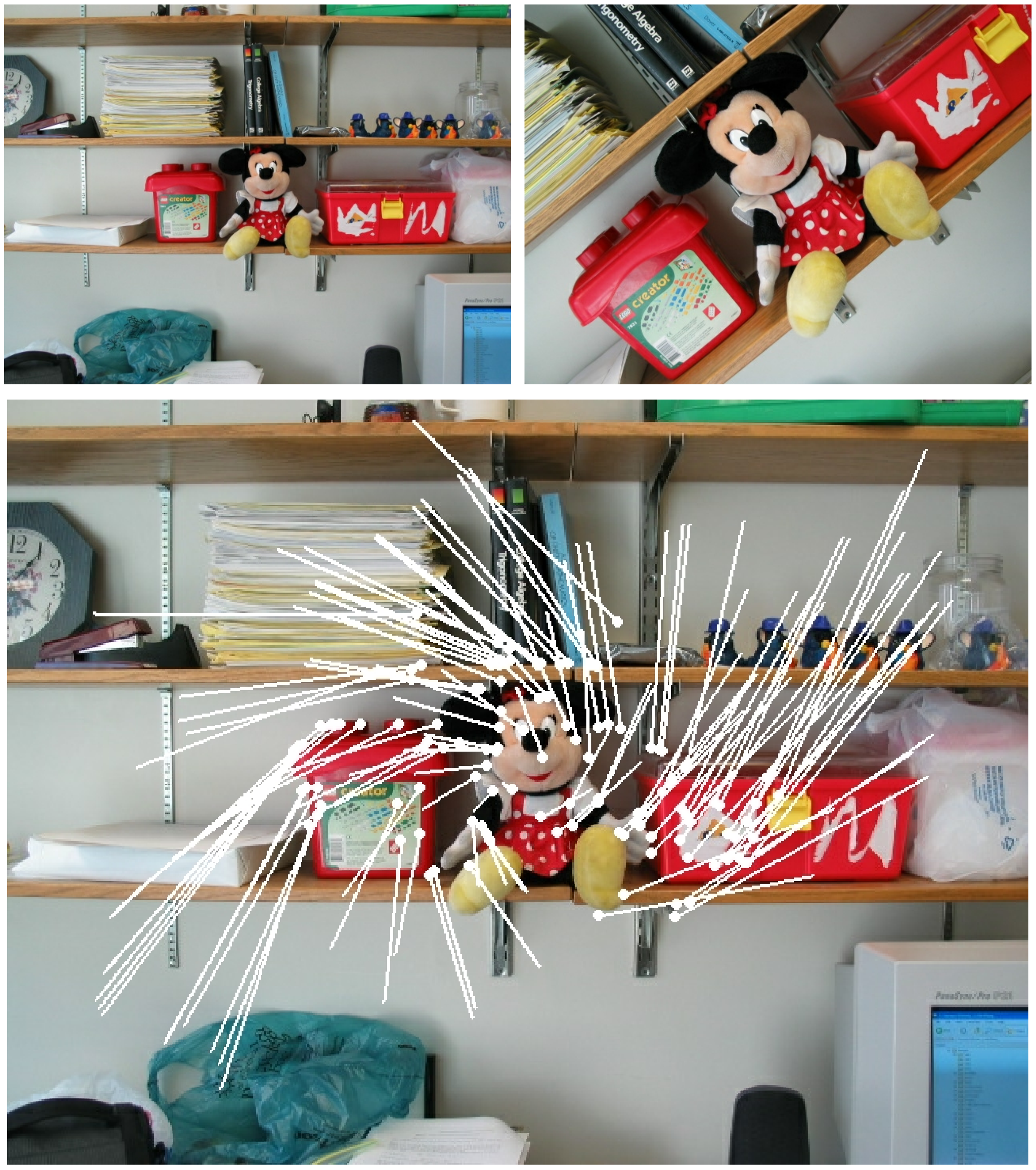


Figure 2.6: Shelf image pair exhibiting scale change and camera cyclo-rotation. This set was expanded from 19 initial matches to 112 final matches.

then started the expansion algorithm from a different, small set of automatically derived initial matches, and tested the expanded set against $\mathbf{F}$ to measure the match distances from their respective epipolar lines. We measured this accuracy on a set of 17 image pairs. The average error for the hand-picked correspondences was 0.766 pixels, and the average error for the expanded set of correspondences was 1.862 pixels, with a standard deviation of 1.18. This indicates a high accuracy for the expanded correspondence set, considering that the point matches are not made to sub-pixel precision. Note also, that this measure does not check for mismatches which lie along correct epipolar lines.

Table 2.1 reports the number of matches found, the expansion run-time in seconds, and the number of iterations needed for the 17 images used in the accuracy check. Figures 2.4, 2.5, and 2.6 show examples of the expansion algorithm finding matches in image pairs.

2.5 Conclusion and Future Work

We have presented a method to expand an initial set of wide baseline correspondences to many times its original size. Using an iterative two-step process, we first aggregate additional matches around existing "seed point" matches. An approximate affine transformation (translation, rotation, and scale) maps the points of the seed match and is used to predict new match locations. Second, we constrain the aggregated matches to a high error tolerance epipolar constraint using RANSAC. These steps are iterated until no matches are added to the expanded set. Using correspondence expansion successfully yields many more wide baseline matches than are obtained using previous methods alone. Our algorithm does not replace previous matching algorithms; rather it augments existing methods as a "post-process" to increase the number of final, high quality correspondences.

We presently do not consider photometric differences between most image pairs while matching. To match images with significant lighting change, we perform intensity histogram equalization in YIQ space as a preprocess to correspondence expansion. While this reduces photometric differences and performs fairly well in practice, we would like to use an invariant measure. A possible solution is to use color moments as in Mindru et al. [1999]; Tuytelaars and Van Gool [2000], however any local measure will impose a large computational burden on our algorithm. A pre-processed contrast and brightness normalization procedure such as in [Sand and Teller, 2004] may be a more efficient approach.

Chapter 3

Histogram Matching for Camera Pose Neighbor Selection

Kevin L. Steele, Parris K. Egbert, and Bryan S. Morse, "Histogram Matching for Camera Pose Neighbor Selection," *Third International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT '06)* `http://ieeexplore.ieee.org/xpl/conhome.jsp?punumber=1000000.`

Abstract

A prerequisite to calibrated camera pose estimation is the construction of a camera neighborhood adjacency graph, a connected graph defining the pose neighbors of the camera set. Pose neighbors to a camera C are images containing sufficient overlap in image content with the image from C that they can be used to correctly pose C using structure-from-motion pose estimation techniques. In a video stream, the camera neighborhood adjacency graph is often a simple connected path; frames are only posed relative to their immediate neighbors.

We propose a novel method to build more complex camera adjacency graphs that are suitable for posing large numbers of wide- and narrow-baseline images. We employ Content-Based Image Retrieval techniques to identify similar images likely to be graph neighbors. We also develop an optimization to improve graph accuracy that is based on an observation of common camera motions taken when photographing with the intent of structure-from-motion. Our results substantiate the use of our method for determining neighbors for pose estimation.

3.1 Introduction

Camera pose estimation, the recovery of a camera's extrinsic parameters, is a long-studied problem in computer vision, and researchers have generated significant results and some robust algorithms [Hartley and Zisserman, 2004]. In many pose-estimation algorithms, image features such as points and lines are matched between image pairs, triplets, or sequences and the matches are used to compute the camera pose. A prerequisite to this feature matching is the identification of image pairs or an image neighborhood defining similar images with which to match features. These image neighbors can be expressed as an undirected adjacency graph [Teller et al., 2003], where nodes of the graph are cameras and their respective images, and edges infer proximity between cameras whose view frusta overlap to include common scene structure. Edges essentially connect cameras that have a high likelihood of successful pose estimation due to the overlapping image content (see Figure 3.1 for an example).

Many pose-estimation algorithms depend on the transitive correctness of successive camera poses:

$$(A \xrightarrow{\mathrm{p}} B) \wedge (B \xrightarrow{\mathrm{p}} C) \Rightarrow (A \xrightarrow{\mathrm{p}} C), \tag{3.1}$$

where $A \xrightarrow{\mathrm{p}} B$ is a binary relation between camera pair $\{A, B\} \in \mathbf{S}_{\text{cams}}$ (the set of all input cameras) indicating that camera B is correctly posed relative to camera A. Since there is often an explicit ordinality to the pose estimation of a camera sequence, a poor estimate early in the chain can propagate large pose errors. It is therefore important that the camera neighborhood adjacency graph be as connected as possible (Figure 3.1). Our objective in this paper is to propose a novel process of camera neighbor selection for construction of the adjacency graph.

The emergence of inexpensive and high-quality digital cameras and camcorders, together with the improved ability to quickly move image and video content into computer memory has enabled many applications of 3D reconstruction and visualization. It is increasingly desirable to construct dense camera networks of hundreds of cameras for visualization and reconstruction purposes. However, given the limitations of current pose estimation algorithms, especially with wide-baseline cameras,

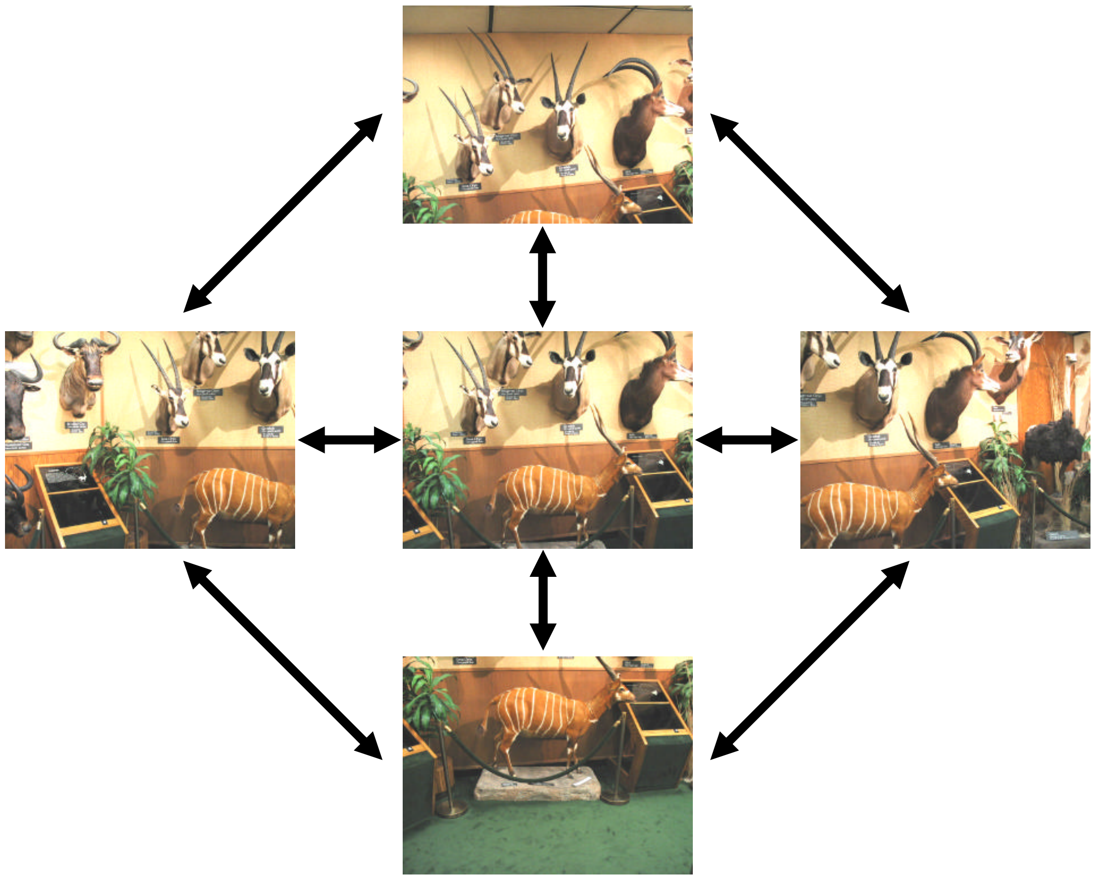


Figure 3.1: A camera neighborhood adjacency graph. These five images contain varying amounts of overlap of a museum taxidermy display. Images containing significant overlap are bi-directionally connected in the graph. Thus, the center image has four neighbors, and its camera should be posed relative to these four neighboring cameras.

creating the prerequisite adjacency graph is difficult. Current algorithmic deficiencies include the inability to involve all close camera neighbors in an initial pose estimate while excluding images that do not overlap at all. Another deficiency is the lack of a coherent method to include all types of footage, e.g., video streams and still images, simultaneously in the pose estimates.

In this paper we propose a novel and efficient way to create the camera neighborhood adjacency graph in the presence of hundreds of input images by using methods from content-based image retrieval systems. We use color histograms to identify similar images as candidate camera neighbors, and partial histogram functions (defined in Section 3.4) to more accurately determine neighbors given constraints unique to the purpose of pose estimation. Our method has the ability to find accurate camera neighbors for large numbers of input images without imposing a specific pose order (at the expense of an $O(n^2)$ algorithm), and makes no distinction between image input formats (still images or video streams).

3.2 Background

Often the adjacency graph creation for a pose solution is done by hand in order to exploit known matching characteristics and optimize the quality of the pose estimation. When few images are to be matched, neighbor selection is trivial. In this section we describe past methods in determining pose-neighbor selection. Often the selection process is implicit to a pose-estimation algorithm, and an adjacency graph is not explicitly constructed.

Teller et al. [2003] create omni-directional images of outdoor urban environments for pose estimation. They determine camera neighbors by taking the k-nearest neighbors of an initial adjacency graph constructed from GPS sensor data acquired at the physical camera location. Most other methods (including ours) attempt to build the adjacency graph exclusively from image content rather than utilizing external sensors.

Much recent work has focused on using video streams in the matching process [Fitzgibbon and Zisserman, 1998; Koch et al., 1999a,b; Lhuillier and Quan, 2002; Nistér, 2000; Sainz et al., 2003], in which features (points, lines, etc.) are identified in an initial frame and then tracked through subsequent frames until the features are no longer identifiable. New features are typically added throughout the tracking process so that at any given frame of the stream many features exist between the current frame and its immediate predecessor and successor. Thus an image's match (and pose) neighbors are the frames immediately preceding or following it in the video stream, and the corresponding adjacency graph degenerates to a path graph, i.e. a path containing all the nodes of the graph (see (a) in Figure 3.2). All algorithms that operate on sequences of images produce similar degeneracies, regardless of input format. For instance, Lhuillier and Quan [2002] use still images rather than a video stream, but they nonetheless enforce an ordering on the input images to define match partners. Sainz et al. [2003] designed their calibration solution to include video sequences, manually tracked sequences, and still images. However, they still require an imposed ordering on all input images. In contrast, our method makes no assumption on the input order and produces adjacency graphs in which most or all correct world-space neighbors are detected and included, not simply those nearby in a linear sequence.

Algorithmic variants include matching to images several frames separated (two, three, or more frames ahead or behind the current frame) to improve matching characteristics such as sharpness or depth variance [Nistér, 2001]. The first attempts to depart from the conventional sequential ordering requirement are those of Koch et al. [1999a,b], who proposed the method of sweeping a camcorder over a viewpoint surface in a zigzag pattern to construct a *viewpoint mesh*, a 3D polygonal mesh whose vertices are the viewpoints of the reconstructed cameras. Rather than restricting their pose neighbors to frames before and after a current frame, they exploit the zigzag nature of the sweeping pattern to find additional neighbors. Their method backtracks at each frame to examine the 3D locations of previously posed cameras—any prior cameras within a distance threshold are included as neighbors to the current frame. Figure 3.2 (b) shows an example of their method. In [Koch et al., 1999b] the authors predict a very coarse estimate of an unknown camera pose by first computing the fundamental matrix F of an image pair from feature matches. They use the epipole

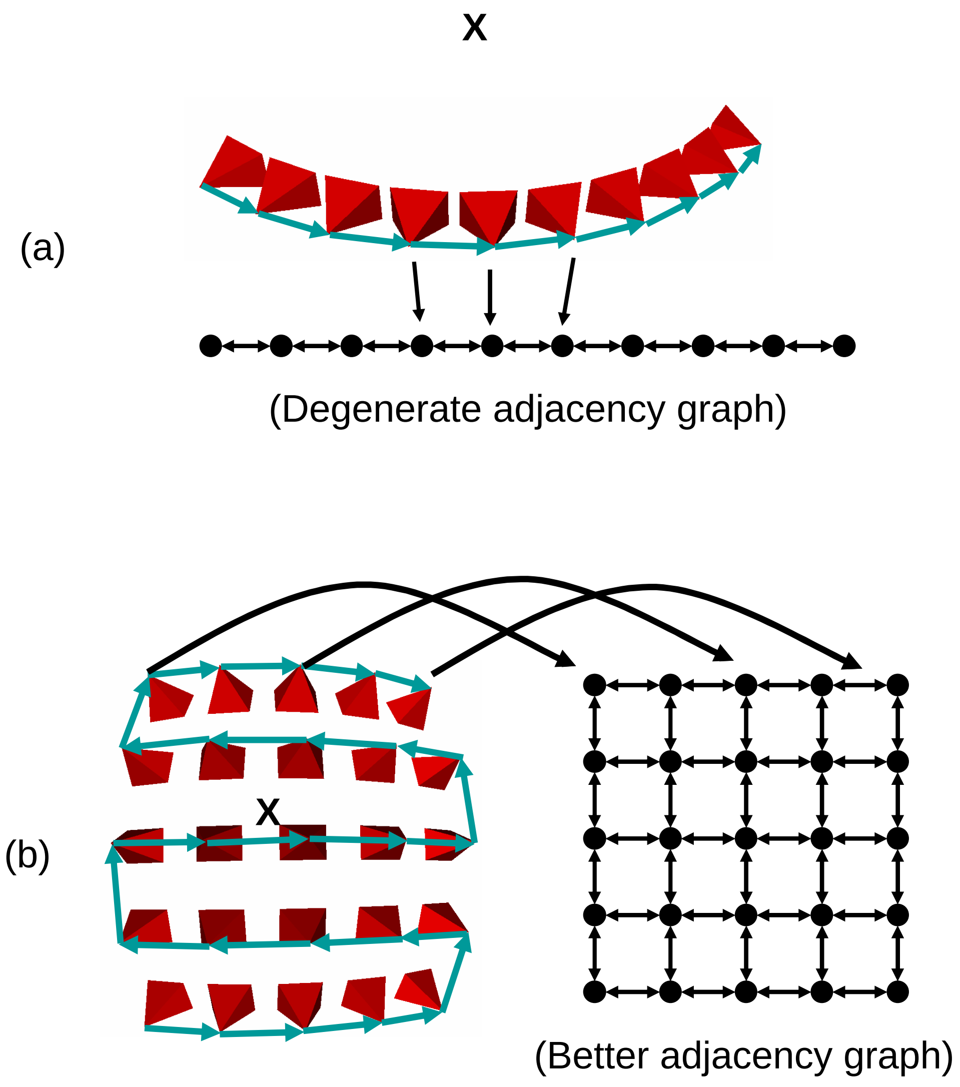


Figure 3.2: Adjacency graphs created from an implicit ordering of the input cameras. The pyramids in (a) and (b) show the camera locations of two hypothetical input video streams pointing toward a central point (X). The input for (a) is a one-dimensional tracked sequence about X. Most current algorithms construct the (degenerate) adjacency graph using immediate frame neighbors, as shown with the arrows. The input for (b) is a viewing sphere [Koch et al., 1999a] from a zigzag about X. Looking beyond the immediate frame sequence [Koch et al., 1999a,b], one can create better graph configurations with more pose neighbors, and consequently more accurate pose estimates.

extracted from F to predict the new pose direction, and the residual correspondence error of the rectified image pair[1] to predict the distance from the pose-partner. Given this coarse pose estimate, they can use a world-space proximity measure to determine camera neighbors from previously posed cameras. Thus the authors are able to implicitly build a more complete, non-degenerate camera neighborhood adjacency graph. This method is similar to ours in that it attempts to build a non-degenerate adjacency graph. However, their video stream must be centered on one central object of the scene, since the coarse pose estimate cannot account for camera rotation. Our method is able to deal with arbitrary camera motion around any part of the scene, provided there is sufficient overlap of scene content in the input set.

Several recent contributions associate overlapping images using local image features. Brown and Lowe [2003] and Brown and Lowe [2005] extract SIFT features from each image and build a k-d tree containing all features from all the images of the input set. Images of the same objects or scenes are identified by searching the k-d tree for the nearest neighbors of a given query feature found on the object of interest. This allows the implicit creation of an adjacency graph, which the authors use to determine the metric pose of each image through bundle adjustment.

Schaffalitzky and Zisserman [2002] find geometrically and photometrically invariant feature descriptors in all images and store them in a BSP tree. Feature vectors within a threshold distance are considered matches, and an adjacency graph in the form of an explicit spanning tree is constructed. Brown and Lowe [2003, 2005] and Schaffalitzky and Zisserman [2002] use local image features to identify adjacent images. Their algorithmic complexity is linear in the number of images, plus a non–linear (though small) tree searching component.

In this paper we propose the use of content-based image retrieval (CBIR) techniques for the task of image neighbor determination for camera pose estimation. By using CBIR to build the camera neighborhood adjacency graph, rather than relying on implicit input order, we remove the requirement of a sequentially-ordered input

[1]For efficiency the authors use a linear affine mapping as an approximation to the projective rectification.

set. The input set can thus be seen as an image database from which to draw pose neighbors for a given image. Queries made on the database have the constraint that all returned images are neighbors in the adjacency graph.

3.3 Content-Based Image Retrieval

Content-based image retrieval is a set of techniques for retrieving images from a database using features automatically derived from the images themselves. The features used to query CBIR databases often include color histogram content, texture, color location, shape and image composition. CBIR has received widespread research attention, and a number of general-purpose CBIR search engines exist—IBM QBIC [Flickner et al., 1995], MIT Photobook [Pentland et al., 1994], and Berkeley Blobworld [Carson et al., 1999] to name a few.

Image retrieval in the larger context is concerned with finding the images in a database that are *semantically* relevant or similar to a query image. Often the relevant images are of the same class or category, such as "all brown dogs" or "all persons on a beach." In the context of pose estimation, however, we are concerned with finding *locationally* relevant images, or images of the same part of a scene as the query image. This puts a large constraint on typical CBIR usage, and restricts the useful feature types.

We use color histograms as the feature type to match images. The color histogram is a widely-used feature representation for image retrieval [Rui et al., 1997]. Its advantages include invariance to rotation and translation of the image content. One major drawback of using color histograms for image retrieval is that they discard any spatial information in the image content. For example, a histogram of an outdoor scene loses all information that the blue sky is at the "top" of the image, complicating queries for more blue sky images. However, as will be discussed in Section 3.4, this weakness is not a disadvantage when using histogram matching for camera neighbor selection.

We transform RGB image color to the HSV color space and make comparisons using the hue component exclusively. A simple L_1-distance comparison function between two hue histograms H and I works well in practice:

$$d(H, I) = \sum_{k=1}^{n} |H_k - I_k| \tag{3.2}$$

where n is the number of color bins. Given a query image histogram H, and a database image histogram I, the histogram distance $d(H, I)$ represents the dissimilarity between the two images. For the L_1-distance metric, $d(H, I)$ is precisely the number of pixels that differ in hue in both images. For each query image, we can sort the remaining images in our input set (the database) on increasing values of d, then threshold either the number of neighbors or the value of d to determine the actual neighbors of the query image.

By performing histogram matching using all images in the input set as independent query images, we can build a directed adjacency graph. If the degree of each node is constant, as would be the case in a typical implementation, then the graph is directed due to the lack of a symmetric binary relation in the set of all neighbors, i.e. I_1 having neighbor I_2 does not imply I_2 has neighbor I_1. For many pose estimation algorithms, a directed adjacency graph is sufficient. If an undirected graph is necessary, it can be assumed from the directed graph with the allowance of a non-uniform threshold (for example, each node may have differing numbers of neighbors).

3.4 Optimization

We now propose a novel optimization that improves the accuracy of the adjacency graph as constructed in Section 3.3. The optimization is based on an observation of typical camera motions made when photographing scenes for eventual structure-from-motion applications:

Observation. *When photographing for the purpose of later 3D reconstruction or visualization, we have a tendency to follow specific camera-motion patterns—we tend either to rotate about a subjects up-axis, or to track (translate) horizontally or vertically past a subject.*

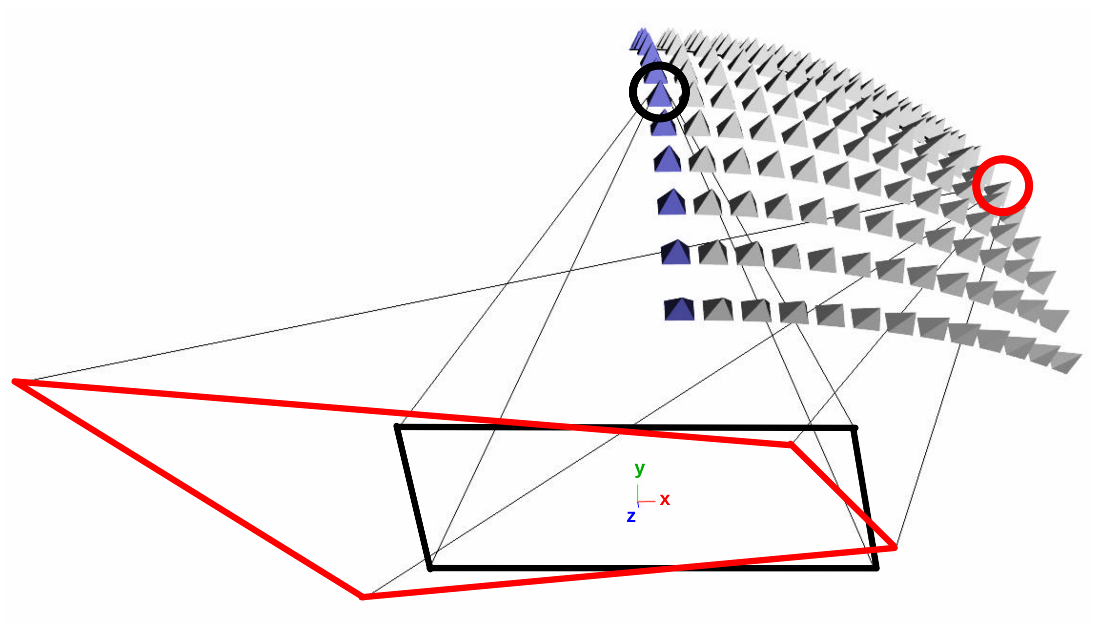


Figure 3.3: Pyramids representing camera view frusta. Each of the darker blue cameras in the left column has been rotated to the right about the origin in 12 increments of 3°; each row of the pictured matrix of viewpoints comprises a rotational set $\mathbf{S}_{\mathrm{rc}}$. Each set has also been rotated 5° up from the previous set to represent varying grazing angles of the geometry with the cameras. The view frusta for all cameras intersect the ground plane $y = 0$ forming individual quadrilaterals. The quads for two such cameras (circled) are outlined in bold red and black lines. Note that the overlap between the two quads is exactly the geometric scene content shared between the two images of the circled cameras. The image of the overlap in this figure is shown as the shaded area of Figure 3.4.

We also tend to avoid extreme or arbitrary camera motions such as cyclo-rotation, panning (except in the case of creating panoramas), diagonal tracks, large rotations and large tracks. While we haven't performed user studies on the validity of this observation, we believe it to be a fair characterization based on our own camera motion patterns and those observed in the computational stereo literature.

This observation will guide the development of our algorithm. Let the ground plane $y = 0$ be a coarse approximation of the geometry of interest in a structure-from-motion application. If we map the typical camera motions taken from the observation, we get a set of translated and rotated camera positions centered on a point, e.g., the

Euclidean origin. Figure 3.3 illustrates candidate sets of rotated cameras. The set of rotations can be expressed by

$$\mathbf{S}_{\mathrm{rc}} = \bigcup_{\theta=\alpha}^{\beta} \mathbf{M}(\theta, \mathbf{t})\mathbf{P} \tag{3.3}$$

where $\mathbf{S}_{\mathrm{rc}}$ is the set of homogeneous camera projection matrices representing the rotated cameras, $\mathbf{P}$ is the homogeneous camera matrix of an initial camera, $\mathbf{M}(\theta, \mathbf{t})$ is the parameterized matrix that rotates the initial camera by θ about a point on the principal axis t units from the camera center, and α and β are the limits of rotation. $\mathbf{M}(\theta, \mathbf{t})$ is expressed in Equation (3.4) as the composition of the individual affine transformations in (3.5). These matrices translate the camera's rotational center to the origin, rotate the camera about the positive y-axis, and then translate it back again.

$$\mathbf{M}(\theta, \mathbf{t}) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & -t\sin\theta \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & t - t\cos\theta \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -t \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.5}$$

The intersections of the view frusta with the ground plane form quadrilaterals, as shown in Figure 3.3. Let q_i be the image of the quadrilateral formed by an initial camera C_i, and q_r be the image of the quadrilateral formed by a rotated camera C_r, where C_i and C_r are cameras whose defining matrices $M_i, M_r \in \mathbf{S}_{\mathrm{rc}}$. The image q_i can be re-projected to C_r by a homography $\mathbf{H}$. This re-projection $\hat{q}_i$ is the original quadrilateral as seen from the vantage point of C_r. The overlap of the re-projected quad $\hat{q}_i$ with q_r is precisely that geometry that is seen from both cameras. Figure 3.4 illustrates the overlap of $\hat{q}_i$ with q_r for the array of cameras in Figure 3.3.

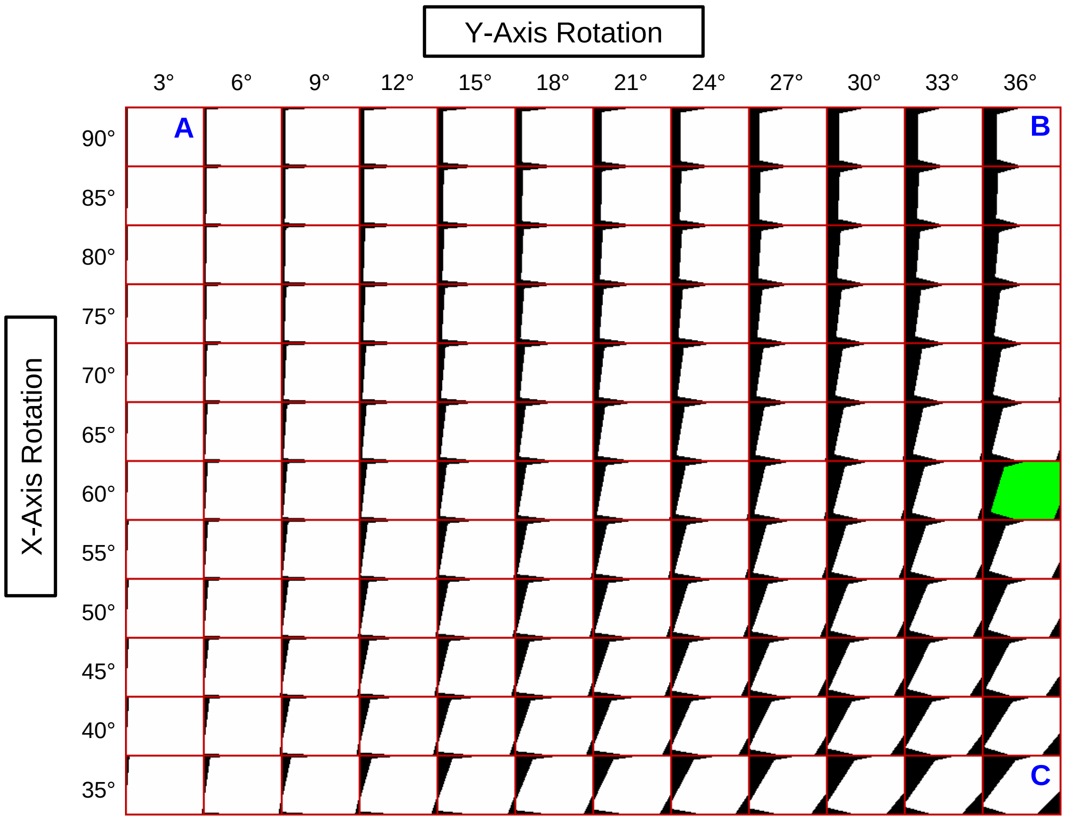


Figure 3.4: Grid of quadrilateral re-projections. Each square of the grid shows the re-projection of an initial camera's quad to a rotated camera of the same set. The white area in each square is the overlap of the re-projected quad with the quad of the corresponding rotated camera. It is easily seen that the coverage pattern shifts to the right (the square labeled B) as the cameras increase in their y-axis rotation, and the pattern rotates clockwise as the camera sets decrease in their x-axis rotation (the square labeled C). This suggests the search patterns illustrated in Figure 3.5. The green shaded area in the figure corresponds to the quadrilateral overlap from Figure 3.3.

In the context of pose estimation, the overlap represents the image content from which to derive useful features for feature matching. The quality of the matches directly determines the success of the pose estimation. Since larger image overlaps imply larger quantities of matches, we would like to quantify the amount of overlap between a query image and all other images, and assign neighbors to the query image from among those with the most overlap. Herein lies the optimization: we can improve the accuracy of the adjacency graph by detecting this overlap, then computing color histogram comparisons only on the overlapping portions of images rather than on entire images.

Examining the family of overlap patterns created from typical camera motions suggests an efficient method to determine the correct overlap. Consider the example of the two cameras labeled A and B in the top corners of Figure 3.4. Camera B is horizontally rotated 36° about a point visible to both cameras. The overlap pattern seen at label B in Figure 3.4 is approximately a right-shift of the image contents. If the scene contains any depth variation, there will be parallax in the shift, but for moderate depth variation the shift still maintains most of the color content. If we compute a color histogram on only the shifted portion of image B, and its equal but opposite shift (we denote as the *conjugate shift*) in image A, we exclude from the histograms those pixels that are not likely to be shared between the images. Using these partial histograms, the distance function of Equation (3.2) will yield a more accurate comparison of these two images.

Since we do not know in advance the amount of rotation (if any) between two cameras, we can parameterize the shift and perform an iterated search for the smallest histogram distance. The histogram distance function then becomes

$$d(H_\phi, I_{\phi^*}) = \frac{1}{|\phi|} \sum_{k=1}^{n} |H_{\phi,k} - I_{\phi^*,k}| \tag{3.6}$$

where H_ϕ is the partial histogram of the query image given a shift pattern, I_{ϕ^*} is the partial histogram of the test image given the conjugate shift pattern, and $|\phi|$ is the size in number of pixels of the partial histogram. The function ϕ returns the set of pixels of a specific shift parameter, and ϕ^* the set of pixels of the conjugate shift.

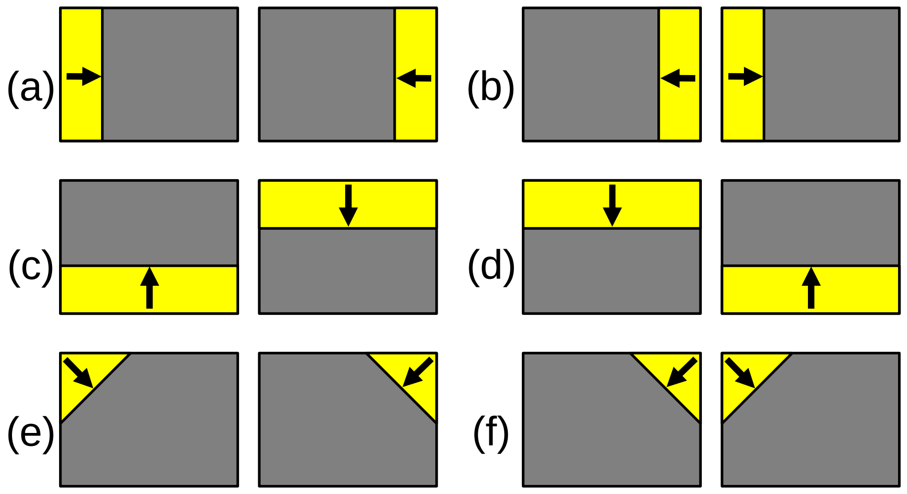


Figure 3.5: Histogram search coverage patterns—each pair consists of a search pattern and its conjugate. We propose six pairs of search patterns (a-f) that follow the coverage patterns in Figure 3.4. Pattern (a) proceeds left-to-right in one image and right-to-left in the other image. Patterns (b-d) proceed similarly according to the arrows. Patterns (e) and (f) proceed from the corners as illustrated to detect low-grazing angle rotation, for example the pattern labeled C from Figure 3.4. Note in this case that we do not need to proceed from the bottom corners for two reasons: the effect of perspective decreases the overlap much more at the top corners, and camera angles from the underside of surfaces are not included in the list of common camera motions.

The distance is weighted by the size of the partial histogram to give distance per unit pixel. The histogram distance of the best overlap is given by

$$d_{\min} = \min_{l} d(H_{\phi(l)}, I_{\phi^*(l)}) \tag{3.7}$$

where l is the shift parameter. In the above example, l iterates left-to-right across the columns of image B, $H_{\phi(l)}$ computes the histogram of the pixels to the right of column l in image B, and $I_{\phi^*(l)}$ computes the histogram of the pixels to the left of column $(ImageWidth - l)$ in image A.

This process can be repeated using other shift patterns that represent the common camera motions. We use six different shift patterns, given in Figure 3.5, and take the minimum $d_{\min}$ of the six patterns to be the final partial histogram distance between two images. Figure 3.6 shows the outlines of an optimal partial histogram region from an image pair.

The algorithmic complexity of partial histogram comparison is similar to that of using global histograms. Computing the global histogram is $O(n)$ in the number of pixels, and computing partial histograms is also $O(n)$ for each shift pattern since, for each value of the shift parameter l, we simply add a line of pixel values to the histogram data computed for the previous value. However, while comparing histograms is $O(n)$ in the number of hues chosen for both global and partial histograms, we make m more comparisons for partial histograms, $m = ImageWidth$, $m = ImageHeight$, or $m = ImageWidth + ImageHeight$ depending on the shift pattern. We have observed this commensurately increased running time in practice.

3.5 Results

We tested our optimized and non-optimized adjacency graph building algorithms using four sets of wide-baseline images. Sample images from each of the four sets are shown in Figure 3.7. We use precision vs. recall comparisons to measure the accuracy of the set $\mathbf{S}_i$ of computed neighbors for each image. To use this measure, we also constructed the set $\mathbf{T}_i$ of "true" neighbors for each image. Precision measures

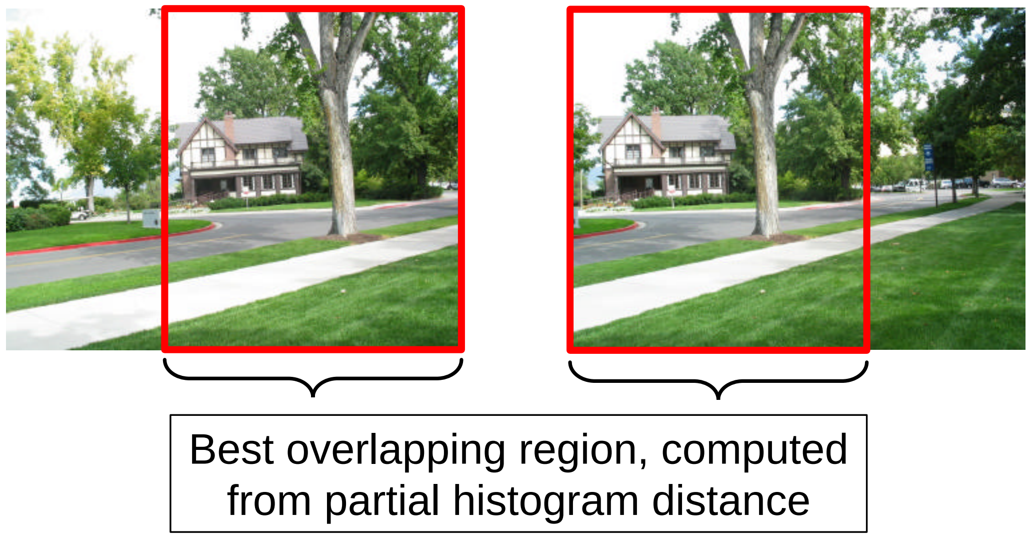

Best overlapping region, computed from partial histogram distance

Figure 3.6: Image pair showing optimal partial histogram regions. The partial histogram distance function found the minimum at column 219 (640×480) in the left image using coverage pattern (b) from Figure 3.5.

the percentage of $\mathbf{S}_i$ in $\mathbf{T}_i$, or $|\mathbf{S}_i \bigcap \mathbf{T}_i|/|\mathbf{S}_i|$. Recall measures the percentage of $\mathbf{T}_i$ in $\mathbf{S}_i$, or $|\mathbf{S}_i \bigcap \mathbf{T}_i|/|\mathbf{T}_i|$.

To automatically construct the set $\mathbf{T}_i$ we utilized feature correspondences that are often-times used to estimate internal and external camera parameters. We computed an accurate set of point matches and their connecting vectors between all pairs of input images. The proportion of image overlap for each pair was determined from its average matching vector. Since low variances of point match vectors correlate well with close image neighbors, we rejected outlying images with no overlap by thresholding on σ^2. Finally, we constructed $\mathbf{T}_i$ for each image from the images exhibiting the most overlap.

We use Equation (3.7) to build the candidate neighbors for each image in the test sets. For a given query image, the minimum histogram distances are computed to all other images in the set. The distances are sorted, and the test images corresponding to the smallest n distances are kept as neighbors to the query image. We measure the precision/recall accuracy for values of $n = [1..k-1]$, k being the size of the test set.

Results of the four test cases are given in Figure 3.8. Each data point on the four graphs represents the average precision and recall of the computed neighbor sets for a specific n number of neighbors. The graphs show a typical inverse relationship between precision and recall [Chowdhury, 2004], and in all four cases using partial histogram distances improves the precision accuracy from simply using the global histogram function in Equation (3.2). Partial histogram distances improve precision on average by 26.9% for five neighbors, and by 24.3% for ten neighbors. Table 3.1 lists percentage improvements for each of the four test cases.

Percentage Improvement

	5 Neighbors	10 Neighbors
Museum	35.8%	36.2%
office	24.0%	24.1%
outdoor1	24.7%	23.7%
outdoor2	23.0%	13.3%

Table 3.1: Improvement of precision accuracy for using partial histogram distances rather than global histogram distances.

3.6 Summary and Conclusions

We have shown a novel and efficient method to create the camera neighborhood adjacency graph for use in pose estimation without having to first find feature matches between two images—a potentially time-consuming process for wide-baseline images [Goedemé et al., 2004]. In addition, our method can process sequential and non-sequential image collections simultaneously, such as images from still cameras and video streams from camcorders. An ordering such as a least-cost Hamiltonian path could later be imposed on the adjacency graph for use in existing sequential pose-estimation algorithms, although we foresee hierarchical or other non-sequential posing schemes to ultimately be more efficient.

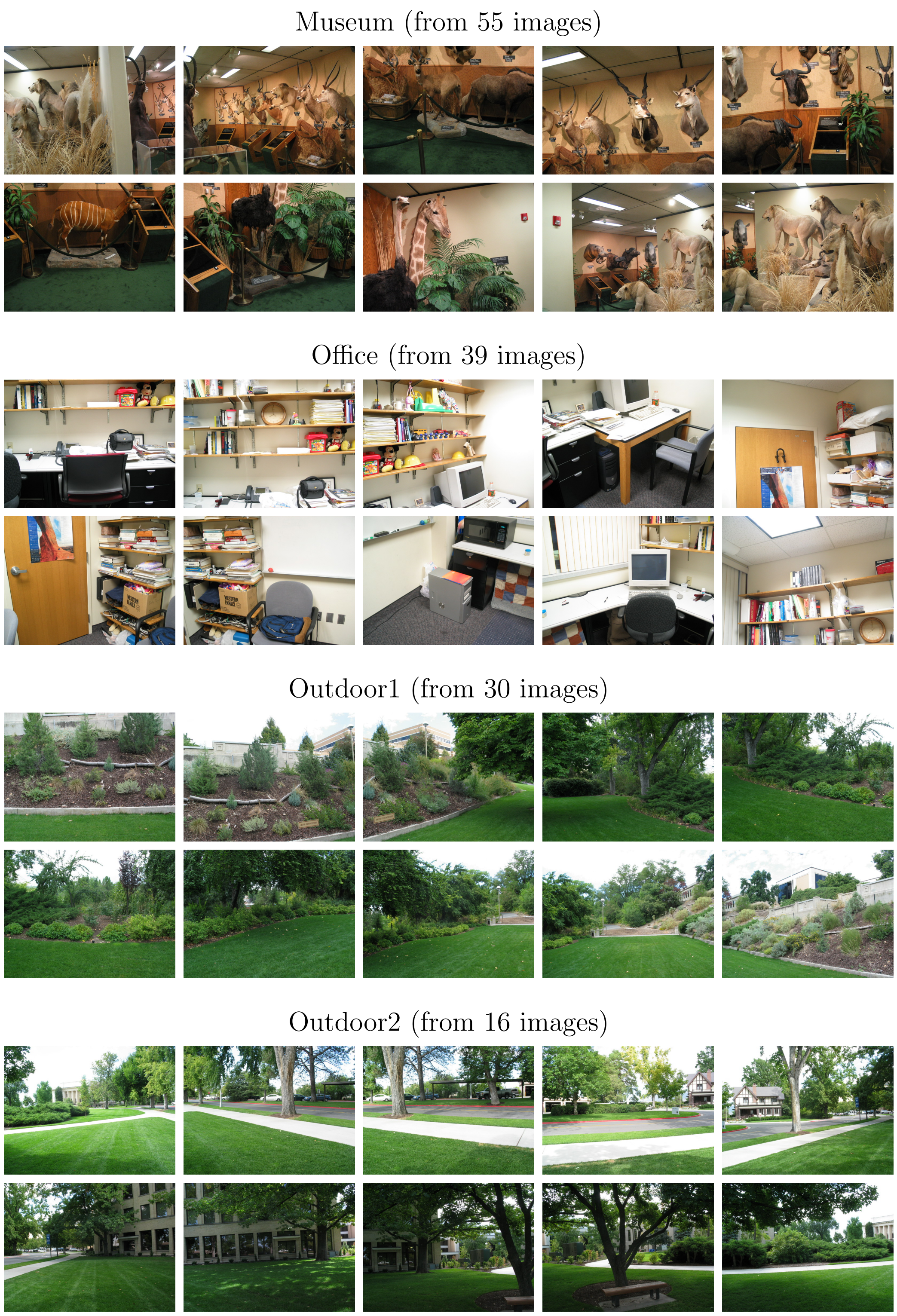


Figure 3.7: Representative images from four test sets. The images from both the Museum and Office sets have camera motions containing significant horizontal and vertical rotation, while Outdoor1 has horizontal and some vertical rotation, and Outdoor2 has only horizontal rotation.

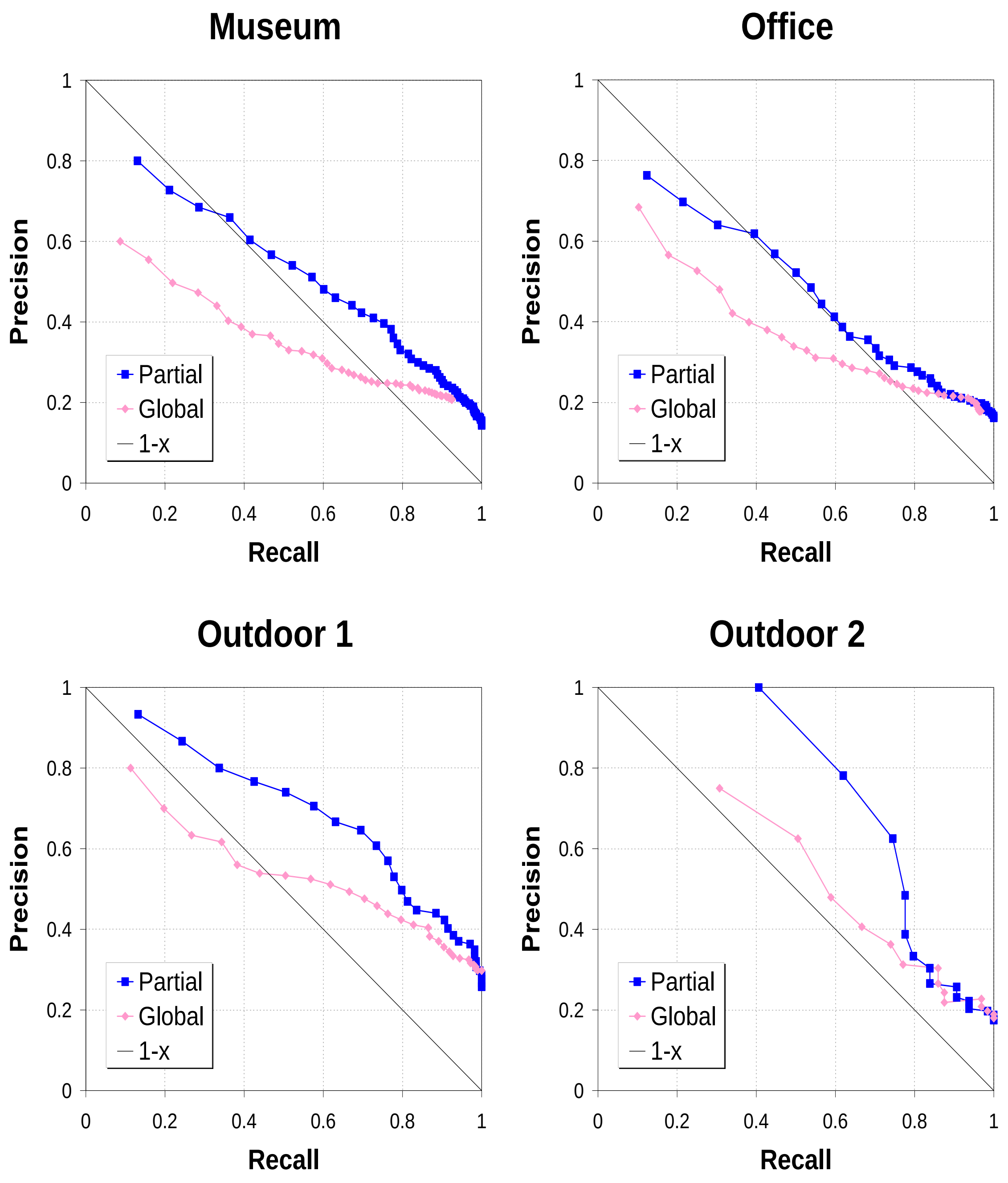


Figure 3.8: Precision/recall graphs from the four test suites (museum, office, outdoor1 and outdoor2). The lighter pink plot in each graph shows the precision/recall accuracy for histograms taken from the whole images (global histogram). The darker blue plots show the accuracy for partial histograms. In all test cases the partial histogram comparisons improved the accuracy significantly.

We have also proposed an optimization for improving the accuracy of the adjacency graph by computing partial histograms, i.e., histograms on just the overlapping portions of image pairs. The optimal overlapping portion is found as the minimum of a distance function that computes partial histograms on a pre-determined pattern of image overlaps. The pre-determined pattern comes from an observation of common camera motions used when photographing content for visualization or 3D reconstruction.

Our method works well for the image sets we have shown in this paper and for many other image sets we have tested. Since our algorithm requires each image in a set to act as the query image, and each query image is compared to all other images in the set, it is an $O(n^2)$ algorithm. In practice one can perform the algorithm on sets containing hundreds of images without undue computation time on a modern processor. However, for sets of extremely large size (thousands of images) it may be beneficial to hierarchically cluster the input images as they are matched. Traditional clustering methods could be used for this using coarse histogram content as a cluster feature.

Chapter 4

Minimum Spanning Tree Pose Estimation

Kevin L. Steele and Parris K. Egbert, “Minimum Spanning Tree Pose Estimation,” *Third International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT ’06)* `http://ieeexplore.ieee.org/xpl/conhome.jsp?punumber=1000000`.

Abstract

The extrinsic camera parameters from video stream images can be accurately estimated by tracking features through the image sequence and using these features to compute parameter estimates. Long video sequences have been posed in this manner. However, large sets of still images cannot be posed using the same strategy because wide-baseline correspondences are not as robust as narrow-baseline feature tracks. Moreover, video pose estimation requires a linear or hierarchically-linear ordering on the images to be externally calibrated (posed), reducing the image matches to the neighboring video frames.

We propose a novel generalization to the linear ordering requirement of video pose estimation by computing the Minimum Spanning Tree of the camera adjacency graph and using the tree hierarchy to determine the pose order for a set of input images. We validate the pose accuracy using an error metric that is functionally independent of the posing process. Because we do not rely on feature tracking for generating feature correspondences, our method can use internally calibrated wide- or narrow-baseline images as input, and can pose images from multiple video streams without special pre-processing to concatenate the streams.

4.1 Introduction

External camera calibration consists of determining the external or extrinsic parameters of a camera matrix P, which are the parameters defining the camera location and orientation relative to a world coordinate frame. Much effort in computer vision has gone into developing stable methods of estimating the external camera parameters in projective and metric spaces; see Hartley and Zisserman [2004] for a rigorous treatment and a compilation of references. A related body of work describes the process of auto-calibration, the automated estimation of a camera's internal parameters from a collection of uncalibrated images [Armstrong et al., 1994; Faugeras et al., 1992; Pollefeys et al., 1998; Triggs, 1997]. Auto-calibration is often performed simultaneously with external calibration (pose estimation).

More recently research has focused on the use of video streams as input to pose estimation problems [Fitzgibbon and Zisserman, 1998; Koch et al., 1999a,b; Lhuillier and Quan, 2002; Nistér, 2000; Sainz et al., 2003]. Large numbers of cameras can be successively posed by tracking features through a sequence of video frames and using those features to pose cameras relative to their predecessors in the sequence. Dense pose estimation of this sort is an important precursor to reconstruction and visualization applications. The advantage to using video streams as an input to pose estimation, rather than taking still images of the same structure, is that the correspondence problem is simpler to resolve in a narrow-baseline setting. Hence, feature matches between image pairs and image triplets are more accurate, improving the pose estimation accuracy.

However, several disadvantages exist to using video streams. Video pose estimation uses an implicit ordering to determine pose order—frames are posed relative to immediate or close predecessors in the image sequence. Most algorithms are unable to exploit out-of-sequence image matches that would otherwise improve an estimate. It is also problematic to combine multiple video streams of a scene, since features are not propagated from one stream to another. We would also like the ability to pose large numbers of wide-baseline images that cannot be matched using robust feature tracking.

In this paper we propose a generalization to the sequential ordering scheme required by video pose estimation. Rather than posing cameras in a linear ordering, we utilize the camera adjacency graph [Teller et al., 2003] to determine the best images from which to extract match features for pose estimation. We compute the Minimum Spanning Tree (MST) of the adjacency graph to determine the pose order for the set of input images, and validate the pose accuracy using a novel error metric that is functionally independent of the posing process.

The contributions of our proposed method are that it can utilize both narrow- and wide-baseline images as input, it can include multiple video streams, and it can determine an optimal set of posed images to use as match generators in computing the next estimate. Our method has produced reliable pose estimates in scenes of over one hundred wide-baseline images without using feature tracking as a correspondence solution.

4.2 Related Work

The goal of dense pose estimation (the estimation of many related camera poses) has been addressed almost exclusively in the context of narrow-baseline imagery from video stream input. This is due largely to the existence of highly robust solutions to the correspondence problem in the narrow-baseline setting, where feature tracking can take a predominant role [Tomasi and Kanade, 1992]. In this section we review the work on dense pose estimation from video streams (video pose estimation).

Fitzgibbon and Zisserman [1998] present a method to track features through an open or closed sequence of video frames, and use successive frame triplets to estimate trifocal tensors. They then hierarchically combine the tensors to build a reconstruction within a common world frame. While the tensor hierarchy promotes reliable 3D structure throughout the sequence to aid in matching, the matching order is still essentially linear in that images are matched to preceding or succeeding video frames. Nistér [2000] presents a generalization of Fitzgibbon and Zisserman [1998] where registration of leaf-level trifocal tensors is delayed until the tensor hierarchy is complete. A set of spanning tensors (wide tensors) are chosen from the hierarchy to represent the entire sequence, from which intermediate views are registered and the structure is triangulated. In this way unnecessary video frames can be discarded.

Still images have also been used for closed loops [Lhuillier and Quan, 2002], but a linear ordering on the input images is still enforced, and a "quasi-dense" feature set is used to compute the fundamental matrices and trifocal tensors, thus partially avoiding the difficulty of posing wide-baseline images. Steedly et al. [2003] use tracked features for video pose estimation, then the posed sequence is partitioned into rigid body clusters to simplify the bundle adjustment phase.

There have been several attempts to utilize feature matches outside the conventional video frame order [Koch et al., 1999a,b]. Koch et al. [1999a] sweep a camcorder over the object of interest in a zigzag fashion, and construct a 3D polygonal mesh whose vertices are the viewpoints of the reconstructed cameras. Rather than restricting their fundamental matrix computations to the preceding frames, they exploit the zigzag nature of the sweeping pattern to find additional images with which to match. Their method backtracks at each frame to examine the 3D locations of previously posed cameras—any prior cameras within a distance threshold are used to update the current pose estimate.

In another work Koch et al. [1999b] sequentially predict a coarse pose estimate for the next video frame by using the epipole extracted from the fundamental matrix F to predict the new pose direction, and the residual correspondence error of the rectified image pair to predict the distance from the previous pose. Given this coarse pose estimate, they use a world-space distance threshold to find additional images to update the pose estimate. A deficiency of this method is that the video stream must be centered on one central object in the scene, since the coarse pose estimate cannot account for camera rotation.

While these methods can successfully pose many frames in their video sequences, they are still restricted to single streams, or streams that have been modified to permit concatenation. We propose a unifying solution to both still image and multiple video stream pose estimation that does not require the robustness of inter-frame feature tracking.

Some related work has been done to create 2D topologies (connected graphs) for image mosaics [Marzotto et al., 2004; Sawhney et al., 1998]. Sawhney et al. [1998] present a method to find the connected graph relating all the images of a mosaic or panorama. Their choice of graph edges depends on two competing goals: to connect images with the best overlap, and to connect images that will best improve the global registration accuracy. Marzotto et al. [2004] build upon the method of Sawhney et al. [1998] by creating a spanning tree of the 2D topology whose edges are determined as the minimum of the normalized distance between image centroids projected onto the mosaic being constructed.

Both Marzotto et al. [2004] and Sawhney et al. [1998] illustrate the application of graph construction to optimize image/camera placement. However, their application is specifically developed for the purpose of mosaic creation, where their cost function is related to image registration and their camera placement is about a common optical center. Our contribution, in contrast, is to use graph construction to determine the optimal ordering for camera placement in general position, requiring a novel cost function. Specifically, our cost function uses validation on the 3D reconstruction to accurately determine edge inclusion, as developed in Section 4.3.1.

4.3 Minimum Spanning Tree

The basic data structure we use is the camera adjacency graph [Teller et al., 2003], an undirected graph whose nodes are cameras and their respective images, and whose edges infer geometric proximity between the cameras. Two nodes sharing an edge in the adjacency graph imply that the view frusta of the corresponding cameras overlap to include common scene structure, and thus the images share some amount of content. Teller et al. [2003] construct each node of the camera adjacency graph from the k nearest neighbors taken from GPS sensor data acquired at the physical camera location. For traditionally-acquired camcorder or still camera imagery, the adjacency graph could be constructed by determining the quantity or quality of feature matches between image pairs; edges in the graph indicate large numbers of accurate feature correspondences. Alternatively, we construct our graph based on the amount of image

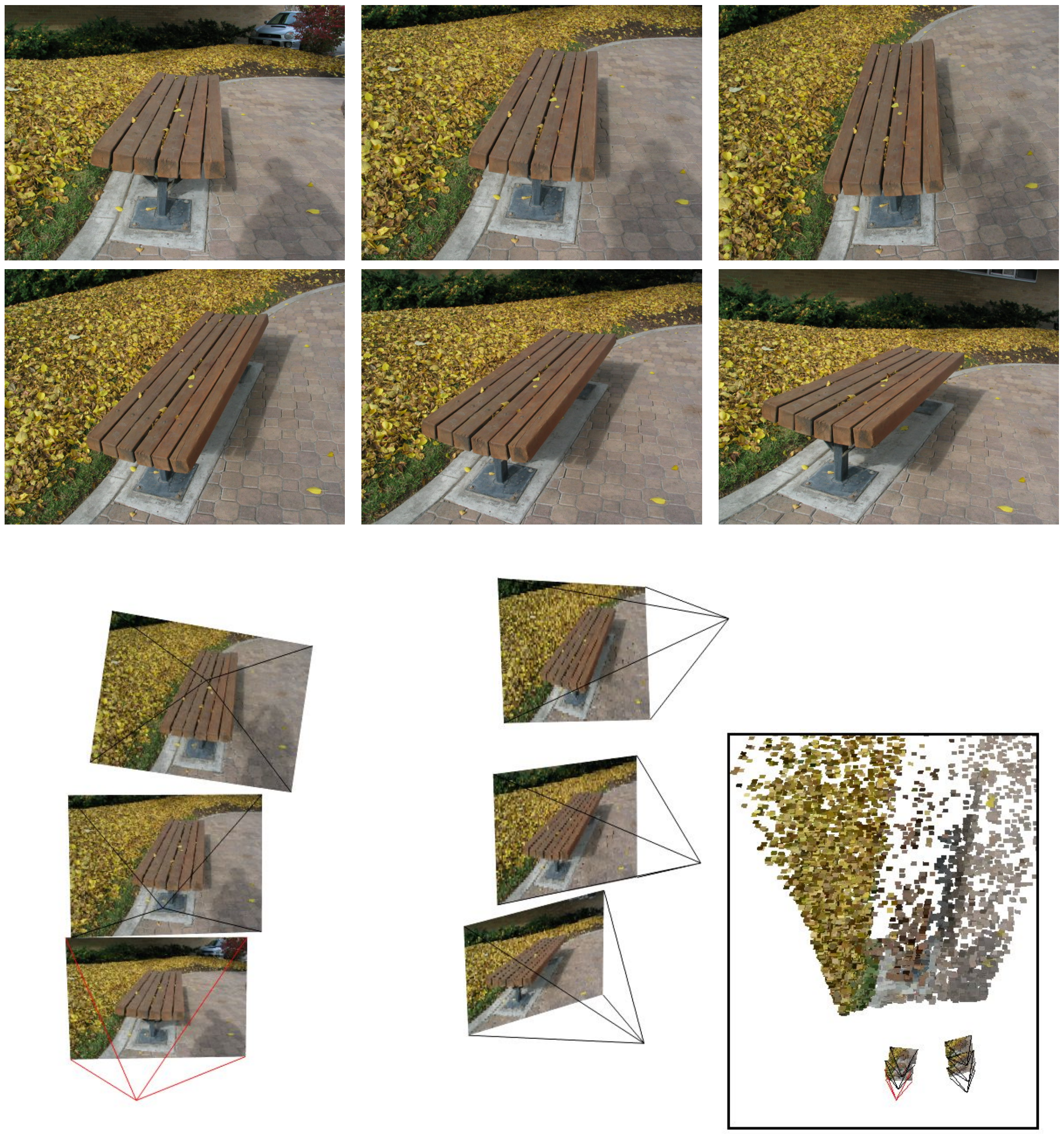


Figure 4.1: Six images of a park bench. The pyramids show the position and orientation of the cameras after being posed using the MST algorithm. The inset shows a small amount of the 3-D reconstruction.

overlap between image pairs, determined from color histogram comparisons [Rui et al., 1997]. From each node in the graph (each image of the input set) we add edges to the n nodes closest in histogram distance.

Our primary contribution is to remove the linear pose-order requirement by using the Minimum Spanning Tree of the adjacency graph to determine pose order. We start by specifying the root node, either manually or heuristically (a node attached to the smallest-weighted edge, for instance). We then proceed by using Prim's algorithm to construct the MST—nodes are iteratively added to the tree in the order of increasing edge weights [Cheriton and Tarjan, 1976]. The edge weights are the histogram distances between nodes, and the camera pose order is the order in which nodes are added. Figure 4.1 illustrates a simple input set and its pose estimates, and Figure 4.2 shows the corresponding adjacency graph and MST.

The final spanning tree is guaranteed to be optimal in minimizing the total edge cost. We wish to transfer this optimality to the process of pose neighbor selection so that finding the MST means finding the optimal ordering. We define this optimality to be the following: At each step of the tree creation, the node added is precisely that camera which

(a) contains the most image overlap (the minimum histogram distance) with some node in the tree, and

(b) maintains a scene reconstruction consistent with that offered by the current tree (see Section 4.3.1).

By choosing the camera order based on maximum overlap we pre-condition incoming nodes to have a high likelihood of correct pose estimation. However, it is not sufficient to simply pose cameras in the order of maximum image overlap. In practice, the histogram distance estimator for image overlap is not perfect and will result in occasional outliers. Additionally, there will be noise and occasional outliers in the correspondence as well, which propagates errors to the pose estimate computed from them. Therefore we need to add a validation of the pose estimate that is independent

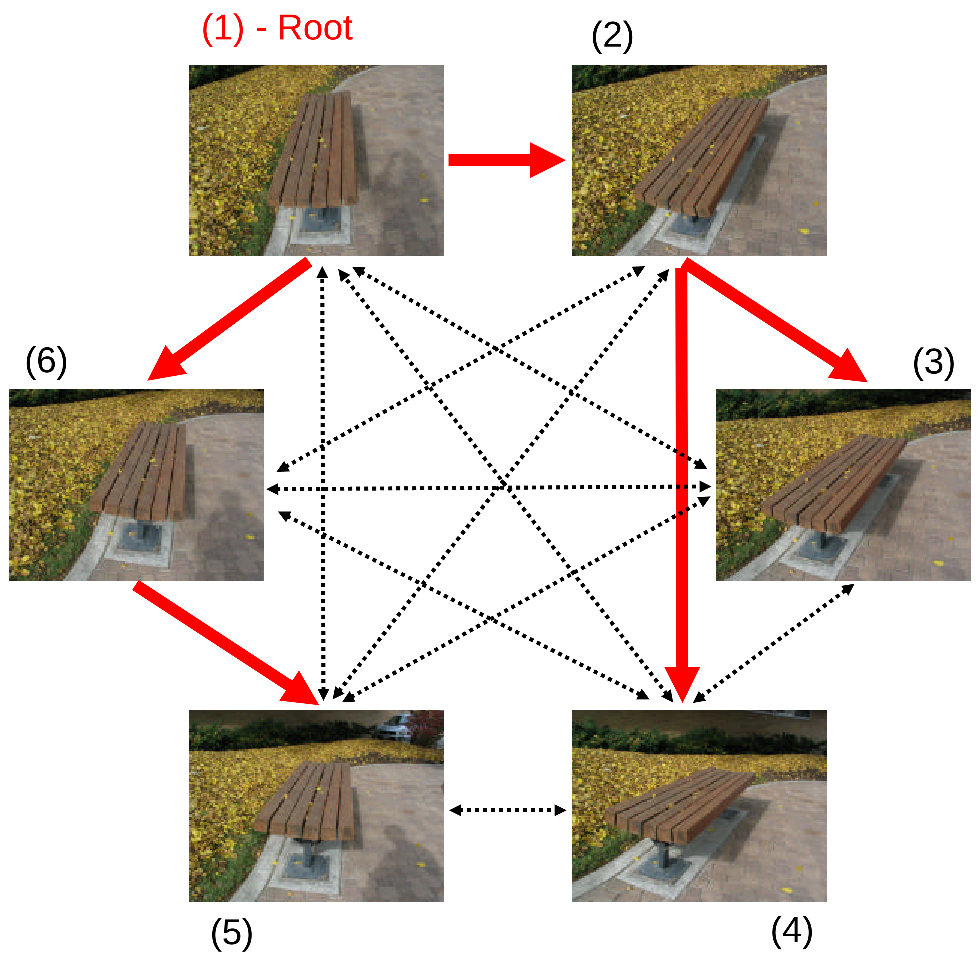


Figure 4.2: Camera adjacency graph and MST of Figure 4.1. The images in this figure are spatially oriented as shown by the posed pyramids in Figure 4.1. The edges of the adjacency graph are small dotted lines—in this simple example the graph is completely connected. The minimum spanning tree was constructed with image (1) as the root node, and the edges are marked with thick solid lines.

of both the adjacency graph edge weights (histogram distance) and correspondence set noise.

4.3.1 Pose Validation

We validate a new pose estimate by comparing the scene structure contributed by the new pose to the scene structure provided by its parent in the MST (we use calibrated cameras in our MST pose estimation algorithm, so there is no need to upgrade the reconstruction from projective to metric). Rather than compare points triangulated from the correspondence set, we compare dense stereo correspondence between images. This results in a richer pose comparison and thus a more accurate error estimate. Scharstein and Szeliski [2002] offer a good survey of dense correspondence methods. Given a pose candidate C_{new}, its parent C', and its grandparent C'', we compute depth maps D_1 and D_2 by triangulating dense stereo between C' and C'', and between C' and C_{new} respectively. Note that D_1 and D_2 are both computed from the viewpoint of C' to make depth comparisons meaningful. We define the reconstruction similarity S between two depth maps as the sum of the Gaussian of depth differences:

$$S(D_1, D_2) = \sum_{p \in P} e^{-(D_{1p} - D_{2p})/2\sigma_p^2} \tag{4.1}$$

where P is the set of all pixel locations in the depth maps D_1 and D_2. We choose σ separately for each pixel to be the camera space inter-pixel distance at the specified depth:

$$\sigma_p = \frac{D_p}{f_{C'} \sqrt{\|\mathbf{P}_{C'} - p\|^2 + f_{C'}^2}}, \qquad p \in P \tag{4.2}$$

where $f_{C'}$ is the focal length of C' and $\mathbf{P}_{C'}$ is the principal point of the image from C'. Choosing σ separately for each pixel provides a more uniform depth comparison by factoring out projective scaling. As each new pose C_{new} is added to the MST, its similarity S to the current reconstruction is retained as an attribute of C_{new}. The similarity attribute is undefined for the first two nodes of the MST since they do not have grandparents, so when the third node is added to the MST, its similarity is propagated up as a special case.

In the ideal case where both the pose estimate and the dense stereo correspondence are perfect, the similarity measure is equal to the number of pixels that overlap between the three input images. This can be seen by considering three perfectly matched points from the input images of C'', C', and C_{new}. The two resulting triangulated 3-D points will coincide, and thus the difference in depth distances $D_{1p} - D_{2p}$ will be zero for those points. The summation $\sum e^0$ from Equation 4.1 will then be equal to the number of common pixels from the three images.

Realistically there will be two error classes that commonly arise—dense correspondence error and pose error. Correspondence error typically arises in regions of low frequency, and hence good matches generally occur on edges, corners, and areas of high texture frequency [Pritchett and Zisserman, 1998]. Given a correct pose estimate, the correspondence error will be less in areas of high detail, and the similarity measure will correspond to the number of shared pixels with accurate point matches. This will be a reasonably high value given sufficient image detail (see Table 4.1 for typical values). Pose error arises in the absence of accurate point matches in the pose estimation process, and a large pose error yields a very low similarity measure. Thus the reconstruction similarity S is a valid discriminator of correct or near-correct pose.

We consider a new pose to be valid if its similarity S is at least half that of its parent node. This constraint invalidates poses that are structurally inconsistent with valid MST nodes, and we have found this threshold to work well in practice. This condition can arise when a node's correspondence set has too much noise or too many outlying matches, or if the node has an incorrect neighbor in the adjacency graph.

When a node is invalidated by failing the similarity comparison, it is not added to the MST, and the algorithm proceeds with the next node. The failed node can be added at a later stage of the algorithm, but it must be added with a different parent. In this way the pose order will be optimal with respect to both optimality properties (a) and (b).

4.3.2 Noise and Outlier Resolution

To estimate camera pose P relative to the parent node, we use the eight-point algorithm to first estimate the essential matrix E from point matches, then we recover rotation and translation from E [Ma et al., 2004]. Since no perfect point correspondence algorithm exists, there is always some amount of noise in the point matches, which transfers to the pose estimate. Worse, there may sometimes be severe outliers in the point matches that will lead to a completely erroneous pose. While the validation step of section 4.3.1 will detect most such pose errors, we can improve the chances of finding an initially correct estimate by first observing the effect of noise propagation in the eight-point algorithm.

The effect of correspondence noise in posing can be illustrated by treating the pose location as a continuous random variable X. [1] If we estimate pose from random subsets of the point correspondence set, we generate a population of pose locations $\mathbf{X_p} = \{x_1, x_2, \ldots, x_n\}$. By using a density estimator such as Parzen windows, we define a likelihood function for a 3D location x:

$$L(x) = \frac{1}{n}\sum_{i=1}^{n} W\left(\|x - x_i\|\right) \tag{4.3}$$

and a probability density function for the random variable X:

$$\rho(x) = \frac{L(x)}{\int_\omega L(x)\,dx}, \quad \omega = \mathbb{R}^3. \tag{4.4}$$

For the kernel W we use a Gaussian function with σ a constant factor of the desired pose baseline (for instance, if the baseline is set to 1, $\sigma = .1$).

Note that this formulation is not a RANSAC procedure, in which subsets of sampled data are iteratively and independently chosen to robustly fit a model to the sampled data. While both RANSAC and Equation 4.4 employ randomly chosen subsets of sampled data, we are simply using the random subsets of 2D point correspondences as input to the eight-point algorithm to establish hypothetical camera

[1] Camera rotation could also be used as a random variable, but since we lack a Euclidean distance measure between any two rotations for use in a smoothing function, i.e., $(R_1 - R_2) \in SO(3)$ cannot be mapped to $\mathbb{R}^1$, it is more difficult to estimate density from a population of rotations.

poses, and then using these discrete pose locations to define a continuous function having higher probabilities near clusters of hypotheses.

The shape of the pdf indicates the stability of the given correspondence set. A narrow, single-modal function is desirable and will generally yield a correct pose estimate. A multi-modal function is indicative of extreme outliers in the correspondence set. We take the pose to be the element of the population that maximizes the pdf:

$$P = \arg\max_{x \in \mathbf{X_p}} \rho(x). \tag{4.5}$$

This effectively chooses the most probable pose in the population as the correct one. In practice we do not need to evaluate the denominator in Equation 4.4 since we only need to find a maximum. Figures 4.3 and 4.4 illustrate two examples of noisy poses and our method of selecting the most probable pose from the pdfs.

If the density function is multi-modal, resulting from match outliers for instance, then the maximum argument associates P with the mode of maximum density; see Figure 4.5 for an example. This will be correct only if the majority of the point correspondences are inliers. While this will be the case most of the time when using a robust point correspondence algorithm, it will occasionally fail. To further reduce the effect of outlier-propagated error, we augment the node selection portion of Prim's algorithm from Section 4.3 with pose density estimation. Rather than posing a new node only with its intended parent in the MST, we additionally pose it with the k nearest neighbors of the intended parent, constrained to similar gaze directions, creating a population of candidate poses. We then use density estimation again to determine the most probable pose, setting σ to a factor of the Euclidean distance between the intended parent and the intended grandparent. In practice this avoids nearly all pose errors resulting from outlying point correspondences. Any remaining erroneous poses are culled by validation, yielding very stable pose estimation.

4.4 Results

We verified the stability of MST pose estimation using several sets of wide-baseline images taken from a still digital camera. Figures 4.6, 4.7, and 4.8 show

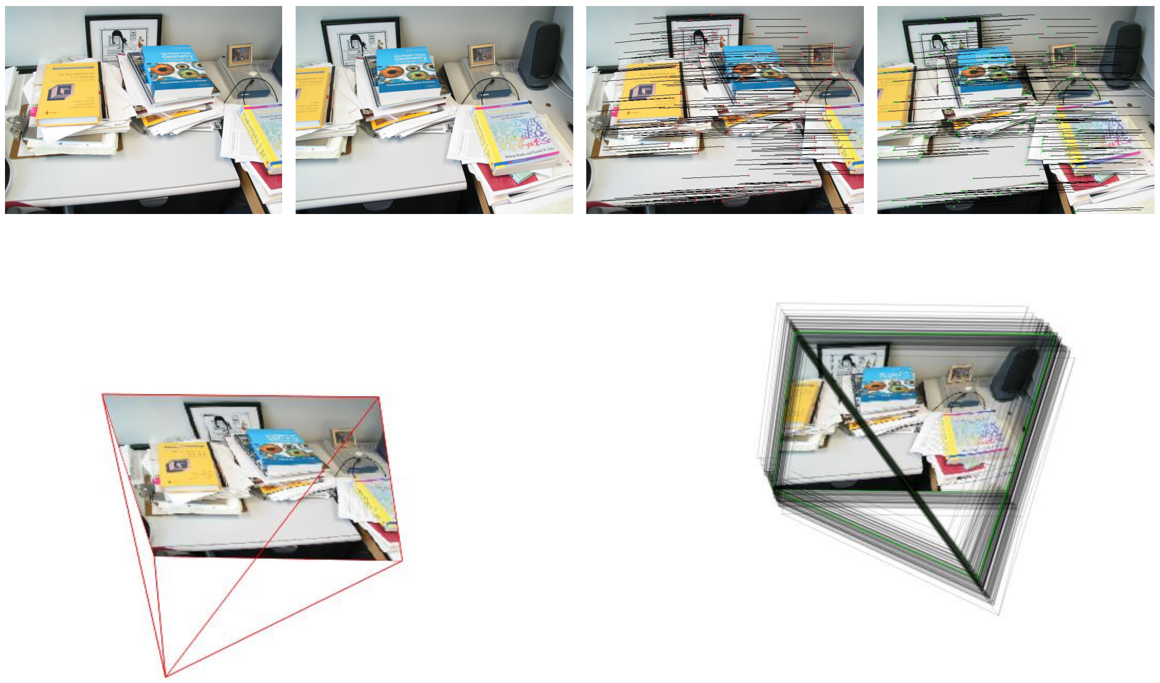


Figure 4.3: An image pair of a cluttered desk (upper left pair). The upper right pair of images illustrates a portion of the point correspondences used to generate the pose estimate. The images were taken by hand attempting to restrict the camera motion to horizontal translation only, and the right image was posed relative to the left. The left wireframe pyramid shows the left camera pose defined to be at the world-space origin, and the cluster of wireframe pyramids on the right shows the population of pose estimates from which the most probable pose was selected using Equation 4.5. Both the left camera and the final selected pose have their images texture mapped into the respective pyramids. Note that due to the large amount of detail in the images and hence the large number of accurate point correspondences, the cluster distribution is small relative to the baseline.

	Hydrant	Skull	Lions
$\bar{S}$	15402.7	9769.8	15791.8
σ	4219.9	4580.3	3586.9

Table 4.1: Mean and standard deviation of the similarity measure S for the examples in Figures 4.6 (fire hydrant), 4.7 (fossilized skull), and 4.8 (lion display). The values of $\bar{S}$ roughly correspond to the average number of consistently reconstructed 3-D points in image triplets as nodes are added to the MST, and thus can be comparatively used to indicate good pose.

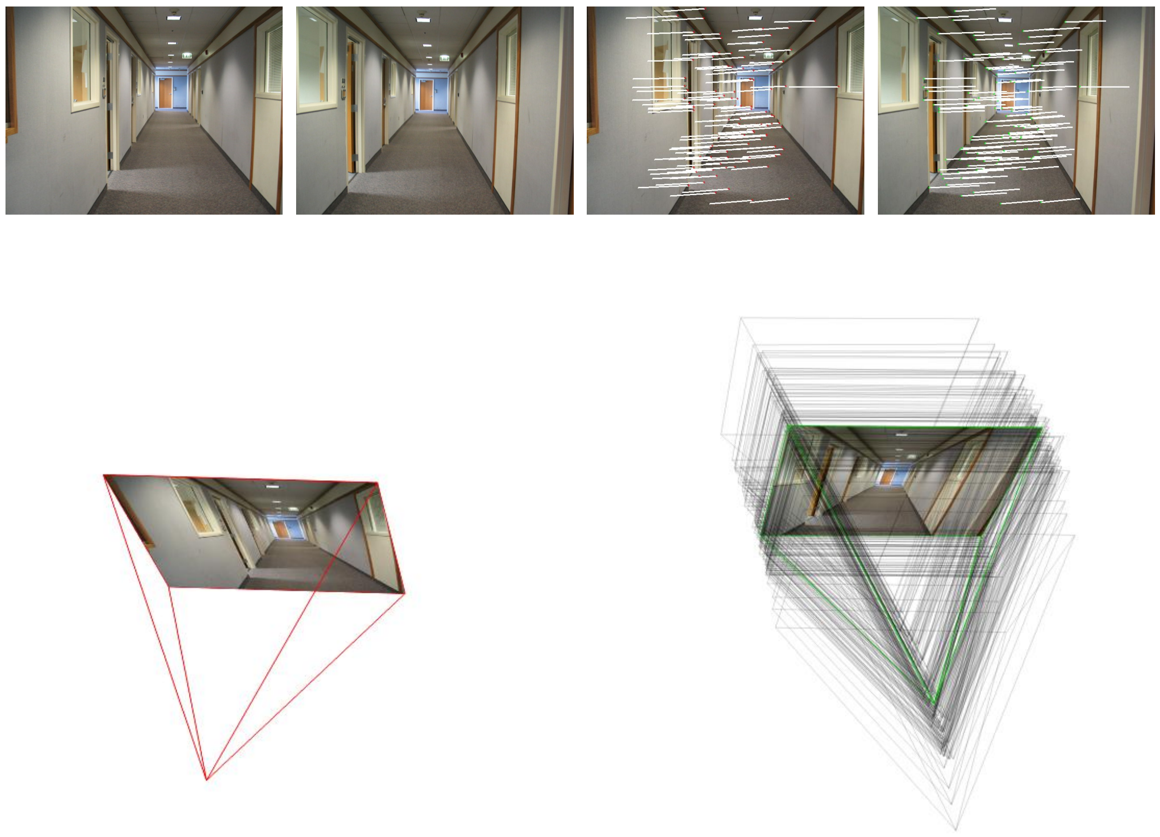


Figure 4.4: An image pair of a hallway in the same configuration as Figure 4.3. Both the left camera and the final selected pose have their images texture mapped into the respective pyramids. Note that these images have less high-frequency detail, so the point correspondences are noisier, resulting in a larger cluster distribution than that of Figure 4.3.

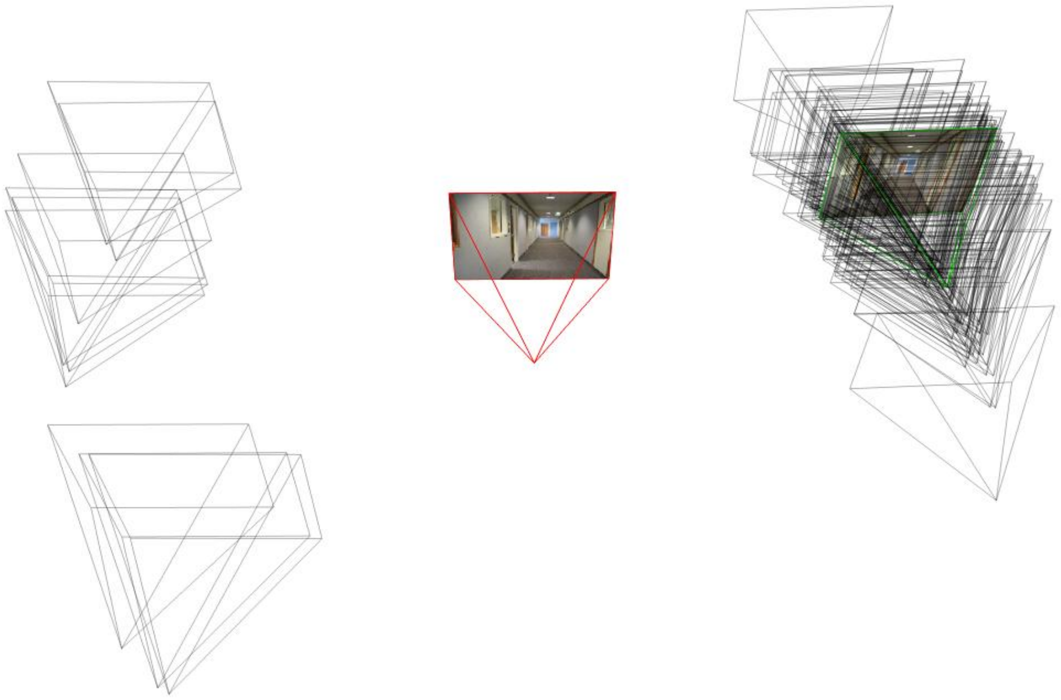


Figure 4.5: An example of a multi-modal density function. This image pair is the same as in Figure 4.4, but the cameras are posed using a smaller number of point correspondences as input to the eight-point algorithm [Ma et al., 2004]. With a higher probability of choosing outlying matches, the pyramid cluster distribution is large and the density function is multi-modal. An accurate pose is selected because in the correspondence set there is a larger proportion of correct point matches than outlying matches.

the pose estimates of three sets of input images. Each image set was taken by hand with a still digital camera. The examples show ten sample images from the input set (each image is 640x480 pixels) and pyramids representing the position and orientation of the final pose estimates. One image in each set was defined to be located at the world-space origin, and all remaining images in each set were posed in the same space using the MST pose estimation algorithm. We have listed the average values of the similarity measures for each example in Table 4.1.

The cameras in the examples are posed fairly accurately, as shown in the figures by the positions of the pyramids and the sparse 3-D reconstruction. However, it is difficult to determine the exact accuracy of the pose estimates without measuring the extrinsic camera calibration using an external verification setup such as a gantry or robotic arm while photographing a scene. Fortunately, if the eventual goal of camera calibration is to perform 3-D reconstruction or visualization where the success of the application is measured by the accuracy of the 3-D content, the similarity measure of Section 4.3.1 is applicable and by definition is directly related to the accuracy of the pose estimation.

4.5 Summary and Conclusions

We have proposed a novel method, MST pose estimation, to estimate the extrinsic camera parameters, or pose parameters, for large collections of images. Our method generalizes on current methods which use narrow-baseline feature tracking to robustly estimate point correspondences and camera pose in a linear or hierarchically-linear order (imposed by the linear nature of the video stream). MST pose estimation finds the Minimum Spanning Tree of the camera adjacency graph and uses the tree node hierarchy to determine pose order. This enables pose candidates to be matched against a much larger number of images than just the immediate predecessors in linear video streams. We lose the robustness from narrow-baseline matching algorithms but gain in the ability to pose generalized input: still images with multiple video streams.

To compensate for the reduced robustness of point correspondences, we proposed a validation method based on reconstruction similarity to quantify the pose correctness. We additionally outlined a novel noise error compensation technique that reduces pose error propagated from correspondence noise. This technique is based on interpreting a population of pose estimates as a probability density function and using density estimation to retrieve the most probable pose from the population. Together with pose validation, these techniques enable robust pose estimation for large collections of wide-baseline images.

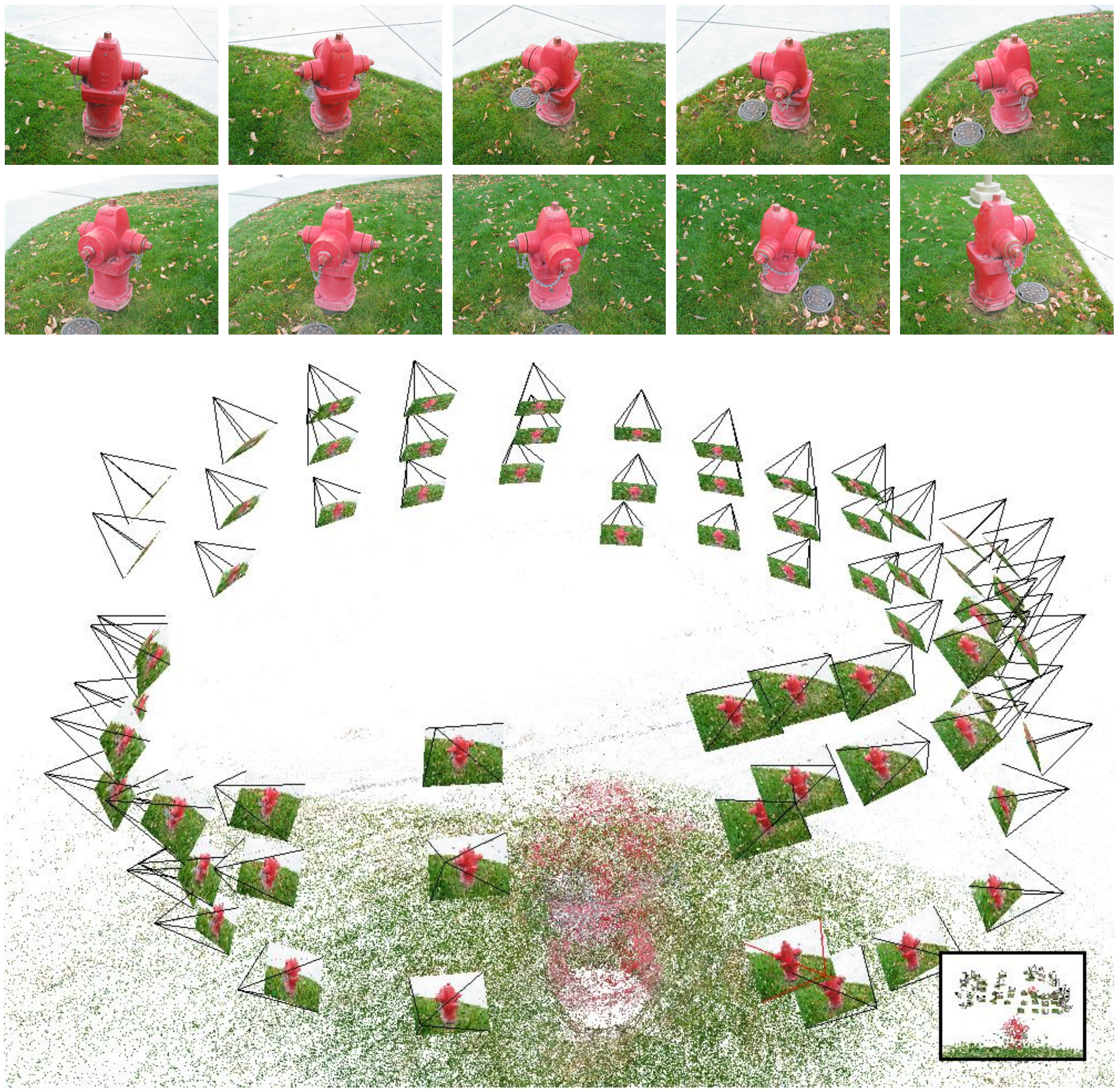
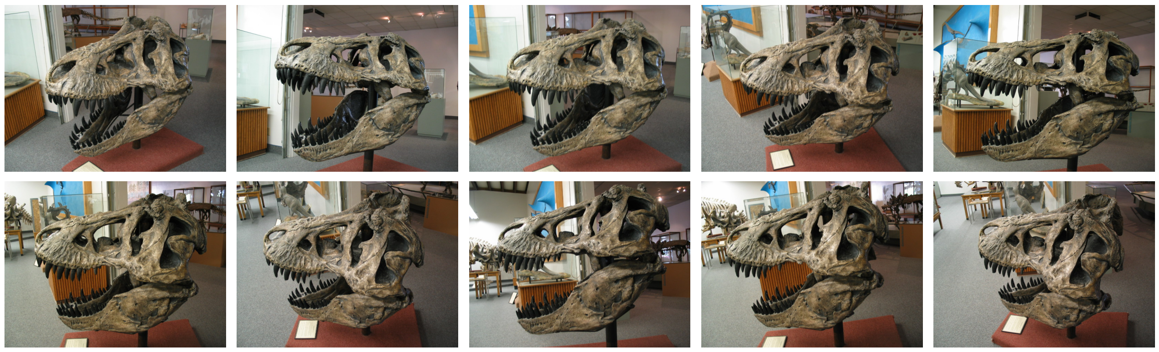


Figure 4.6: Images of a fire hydrant from an input size of 69 images. The pyramids illustrate the pose estimates for each image. The inset shows the pose estimates from a viewpoint close to ground level. The 3-D point clouds in the background of this figure and of Figures 4.7 and 4.8 are shown only to illustrate the general position of the estimated poses relative to the scene structure, and are not attempts to accurately reconstruct the 3-D geometry.



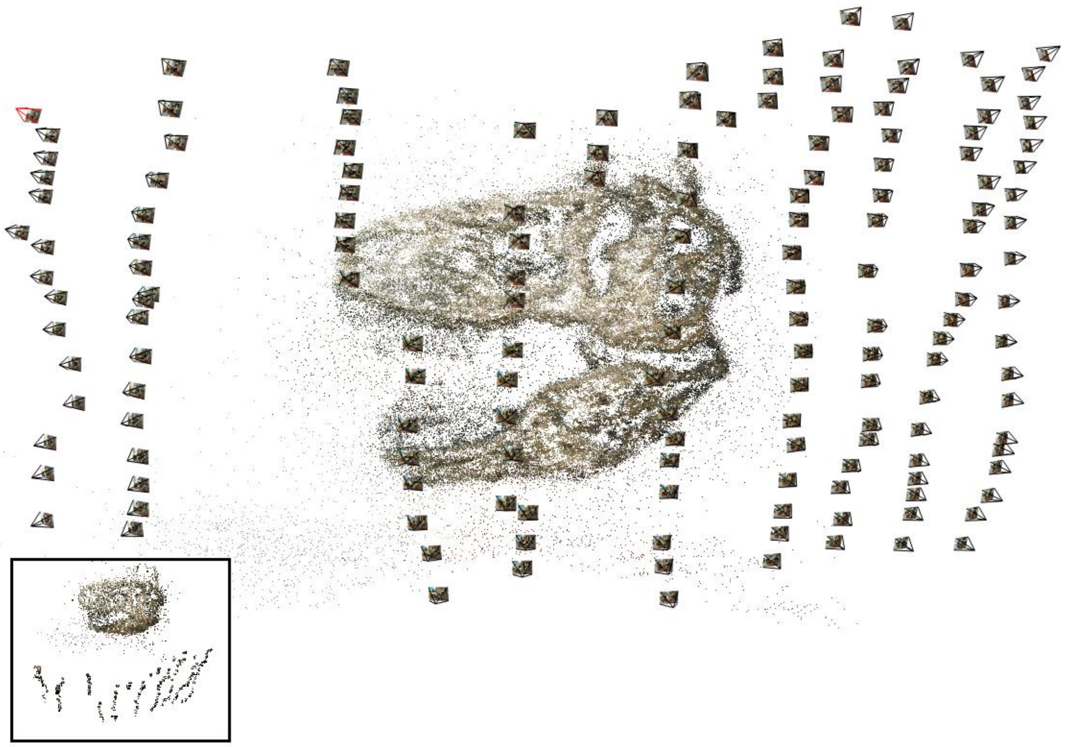


Figure 4.7: Images of a fossilized skull specimen from an input size of 160 images. The pyramids illustrate the pose estimates for each image, and the inset shows the pose estimates from a higher viewpoint. The pyramids in this figure are necessarily small in order to show each image from the complete set.

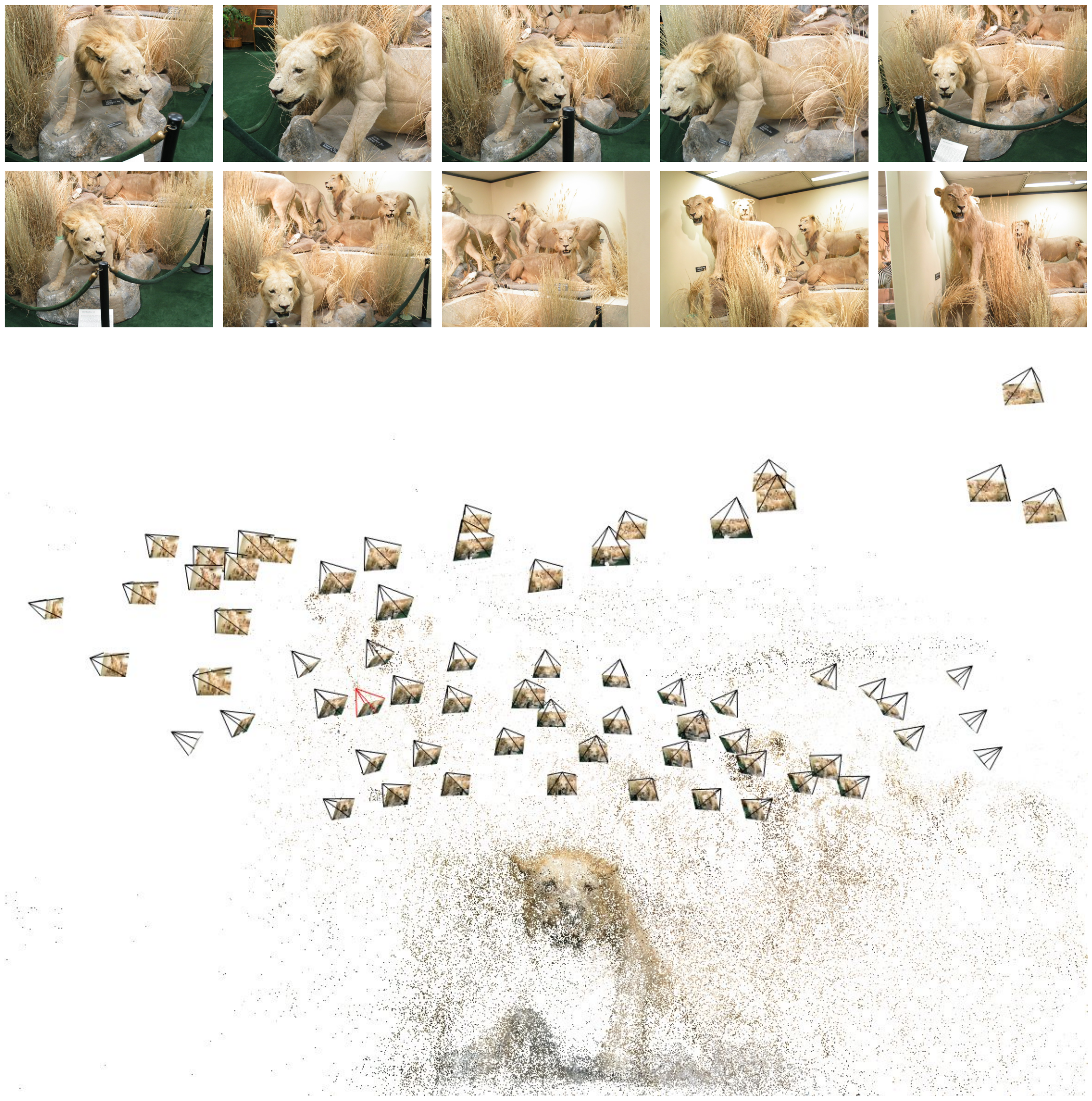


Figure 4.8: Images and pose estimates of a taxidermy display from an input size of 67 images. The lion shape evident in the reconstructed point cloud corresponds to the lion displayed in the top row of images.

Chapter 5

Omnidirectional View Interpolation of Unstructured Photographs

Kevin L. Steele and Parris K. Egbert, "Omnidirectional View Interpolation of Unstructured Photographs," The contents of this chapter will be submitted to *Graphics Interface 2007*.

Abstract

We present a view interpolation method for photographs taken with hand-held consumer cameras. Given pose and per-pixel depth estimates for each image in a sparse collection, our scalable algorithm synthesizes views between existing images. View interpolation occurs within a tetrahedral decomposition of the image viewpoint convex hull, where each viewpoint is a node in a Delaunay complex. There are no constraints on the location or orientation of the input photographs or the synthesized viewing parameters, allowing six degrees of navigational freedom.

Our primary contribution is a novel method of creating omnidirectional images at each node that are well suited to interpolating unconstrained sets of photographs. Our method preserves the sampling rate of each original image, and it does not require a panoramic sampling about the node center. We also propose an image quality function that defines and estimates the distance between a synthesized image and a target photograph. We use the quality function as an error metric to compare variations of our omnidirectional interpolation algorithm.

5.1 Introduction

View interpolation is a form of image-based rendering in which an image is synthesized to resemble an intermediate viewpoint between two or more input images. Usually, though not always [Seitz and Dyer, 1996], the change in viewpoint between input images is a rigid-body transformation, and novel synthesized views are rendered from viewpoints that are linear combinations of these transformations.

The use of view interpolation of CGI images as a means to navigate virtual environments was made feasible by Shade et al. [1996] and Chang et al. [1999]. However, the real potential of Image-Based Rendering (IBR) techniques is in their use with photographs rather than with CGI. Images synthesized from IBR manipulation of photographic content has high impact and can deliver compelling results. Particularly, the use of view interpolation of photographs or video for virtual navigation of a real environment is an important goal with many applications in simulation, education, and entertainment. Unfortunately, realizing this goal has been elusive for two reasons: the difficulty of determining a camera's external calibration or pose, and the greater difficulty of determining per-pixel depth information in photographs.

In fact, the computer vision community has researched these two problems for many years. The external calibration of large camera networks has recently become viable as a solution to the pose estimation problem [Brown and Lowe, 2005; Schaffalitzky and Zisserman, 2002]. Per-pixel depth estimation strategies, also known as the stereo correspondence problem, have been surveyed in Scharstein and Szeliski [2002]. Recent results in segmentation-based stereo allow multiple views (more than two) to contribute to the solution, and produce reasonable depth maps [Tao et al., 2001; Zitnick et al., 2004].

With the advances in computer vision addressing the shortcomings of IBR, view interpolation of large photographic collections is becoming feasible. Following this trend, we propose a view interpolation algorithm for unstructured photographs, or photographs taken from arbitrary positions without the aid of a rig or gantry, i.e. by using a hand-held consumer- or prosumer-grade still digital camera. Our algorithm assumes accurate internal and external calibration and per-pixel depth information for all input photographs. Given an input set, we interpolate images located at the vertices of tetrahedra taken from a Delaunay partitioning of the viewpoint convex hull.

Our primary contribution in this paper is a novel method of creating omnidirectional images at each camera viewpoint that are well suited to view interpolation. Our method preserves the sampling rate of each original image; each image that is associated with a given camera viewpoint is retained in its entirety in the resulting omnidirectional image. The omnidirectional images utilize as much information as possible from the original input set.

5.2 Background

View interpolation first appeared in the graphics literature in an innovative paper by Chen and Williams [1993]. The authors presented a method for generating images of novel viewpoints from existing synthetic images arranged in a grid pattern. Pixels in the source images were linearly interpolated in real-time to new locations using a preprocessed mapping between existing viewpoints. The mapping was later generalized [McMillan and Bishop, 1995b] by relaxing the regular grid constraint of the source viewpoints, and is generally referred to as Image-Based Rendering by Warping [McMillan, 1997].

At approximately the same time the computer vision community became interested in view interpolation of photographs. Laveau and Faugeras [1994] presented a limited method to interpolate photographs using the fundamental matrices between image pairs. Werner et al. [1995] introduced a method to synthesize arbitrary viewpoints using linear combinations of source photographs. Both methods required dense pixel correspondences between source image pairs.

Virtual walkthroughs have received attention from both graphics and vision researchers. Aliaga and Carlbom [2001] record closed loops of omnidirectional video and render novel views from frames enclosing the new viewpoint. Gotz et al. [2002] propose a database organization that is optimized for image-based walkthroughs. Columns of image data are grouped into epipolar plane images which are easily compressed.

IBR reconstruction is performed by retrieving the appropriate columns of image data as needed by novel viewpoints. Li et al. [2001] show an algorithm to transmit pictures, panoramas, and concentric mosaics [Shum and He, 1999] over low-bandwidth network connections by employing selective retrieval and caching strategies. Tomite et al. [2002] interpolate images taken from an omnidirectional camera to render planar interactive walkthroughs. Our work is similar in intent to Tomite et al. [2002], however their emphasis is in the interpolation methodology, while ours is on the creation of omnidirectional images from planar images.

5.3 Omnidirectional Image Creation

In addition to providing view interpolation between the viewpoints of an input set of photographs, we wish to enable arbitrary viewing orientation between viewpoints as well. However, each input photograph samples only a small amount of the viewing sphere surrounding the viewpoint. We will therefore create an omnidirectional image at each original camera viewpoint, retaining the pixel samples of the original photograph in the new image.

The use of omnidirectional imagery for rendering is not new. Teller et al. [2003] also create omnidirectional images from photographs, but for each omnidirectional image they synthesize, the constituent photographs are all taken close to a common center of projection which facilitates the creation process. Due to the unconstrained nature of the input set, our input photographs do not have a common center of projection, and so we must re-project the pixel samples to the viewpoint of the input photograph for which an omnidirectional image is being synthesized.

We create an omnidirectional image for a camera by first constructing a cubical environment map [Greene, 1986] centered on the camera viewpoint and oriented along the primary axes. We convert the pixels of all input images, including those of the image associated with the given camera, to 3D points in a common world space, then re-project them to the environment map. Pixels are converted to 3D points in world

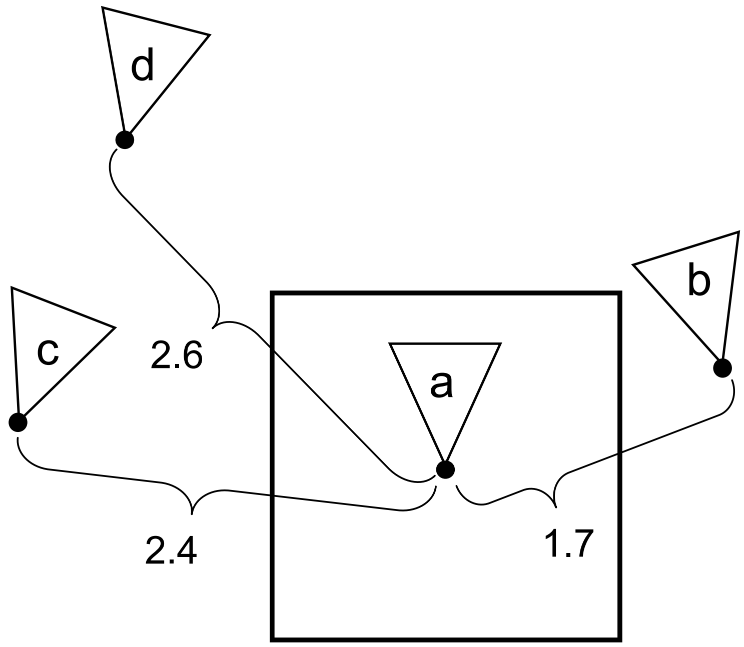


Figure 5.1: Cameras **a**, **b**, **c**, and **d** all have images acquired through the view frustums shown. An environment map is centered on the viewpoint of camera **a**. The distances of cameras **b**, **c**, and **d** to the environment map center are 1.7, 2.4, and 2.6 units respectively. The distance of camera **a** to the environment map center is 0 units, so its pixels are re-projected first, followed by the pixels of camera **b**, camera **c**, then camera **d**.

space by projecting them from their respective camera centers:

$$P_{world} = C + \frac{(p_{xy} - C)}{\|p_{xy} - C\|} \, d_{xy} \tag{5.1}$$

where C is the camera center of the pixel, p_{xy} is the pixel location in world space, d_{xy} is the depth at pixel $\langle xy \rangle$, and P_{world} is the 3D point in world space.

5.3.1 Pixel Re-Projection Order

The environment map maintains all information from the re-projected points, including color and world-space position. Rather than treat the map faces as depth maps, we treat them as write-once maps; the first point projected to a given pixel is kept, and all others are discarded. The order of re-projection is therefore important. We order the point re-projections by the distance between a 3D point's camera viewpoint and the environment map center of projection. Since the environment map is

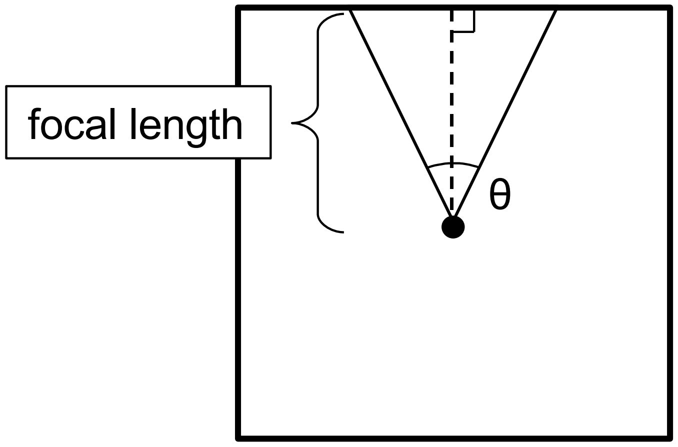


Figure 5.2: Top view of a camera frustum and environment map centered on the camera viewpoint. A camera with no rotation will coincide with the front map face. If the horizontal resolution is 640 pixels and the horizontal field of view θ is 40°, then the camera focal length in pixel units is $\frac{640}{2}/\tan\frac{40}{2} \approx 879$, and the resolution of each map face would be 1758×1758 pixels.

centered on a camera viewpoint, the pixels of that camera are re-projected first, thus retaining the original image in the environment map (see Figure 5.1).

In order to retain the sampling rate of the original image, the environment map should match the image resolution as closely as possible. This can be achieved by setting the resolution of each map face to twice the camera focal length in pixel units. Figure 5.2 illustrates the relationship geometrically.

The advantage to using write-once maps rather than depth maps is that the write-once depths are more uniformly consistent with whole photographs. If depth maps were used, the resulting depths in a given environment map face would be corrupted with depth noise that inevitably occurs in dense depth estimation. This depth noise favors the heterogeneous distribution of source pixels throughout the map, i.e. any given pixel's neighbors are likely to have originated with different source photographs, resulting in a noisy environment map. Write-once maps ensure that the

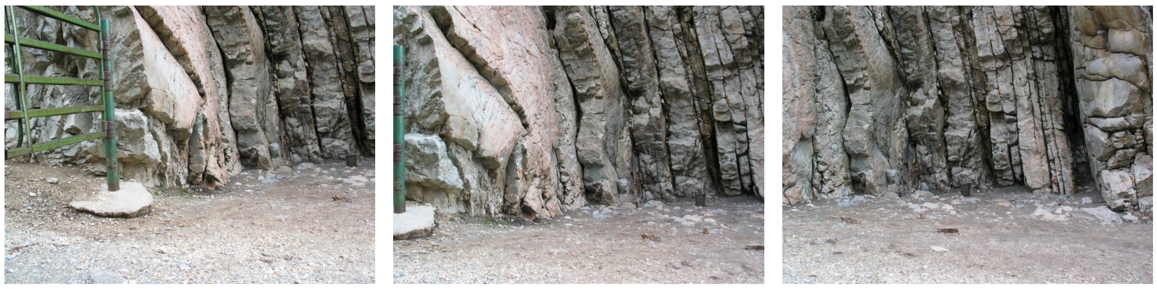


Figure 5.3: Three images of a rock façade from the original input set of 99 images.

source pixels from large portions of a photograph will be transferred as a block to the environment map, maintaining a truer rendition of the original photograph. However, this potentially creates erroneous environment map segments in the presence of large photograph depth errors or occlusion differences between viewpoints.

Once the environment map is constructed, the images of each face could be warped [McMillan and Bishop, 1995b] to generate new viewpoints nearby. However, warping requires per-pixel generalized disparity—equivalent information to per-pixel depth. Rather than warp the environment map faces to render new views, we will simply use the stored locations and colors of the original 3D points, and render these points using the traditional graphics pipeline. The omnidirectional image thus contains a subset of all reconstructed points, and the write-once mapping policy ensures that the subset comes from cameras located closest to the viewpoint. The image is therefore the most accurate estimation of an omnidirectional sampling at that viewpoint. Figure 5.3 shows three images from an example input set of 99 images, Figure 5.4 shows one face of an environment map after re-projecting all pixels of all images onto it, and Figure 5.5 shows the map centered on a camera viewpoint.

5.4 View Interpolation

Given omnidirectional images for each viewpoint in the input set, we define the navigational boundaries as the convex hull of the set of all viewpoints. Note that we are not constrained to planar movement as previous walkthrough methods have been. The virtual camera's location and orientation are unconstrained within the boundary of the viewpoint convex hull, permitting six degrees of navigational freedom.

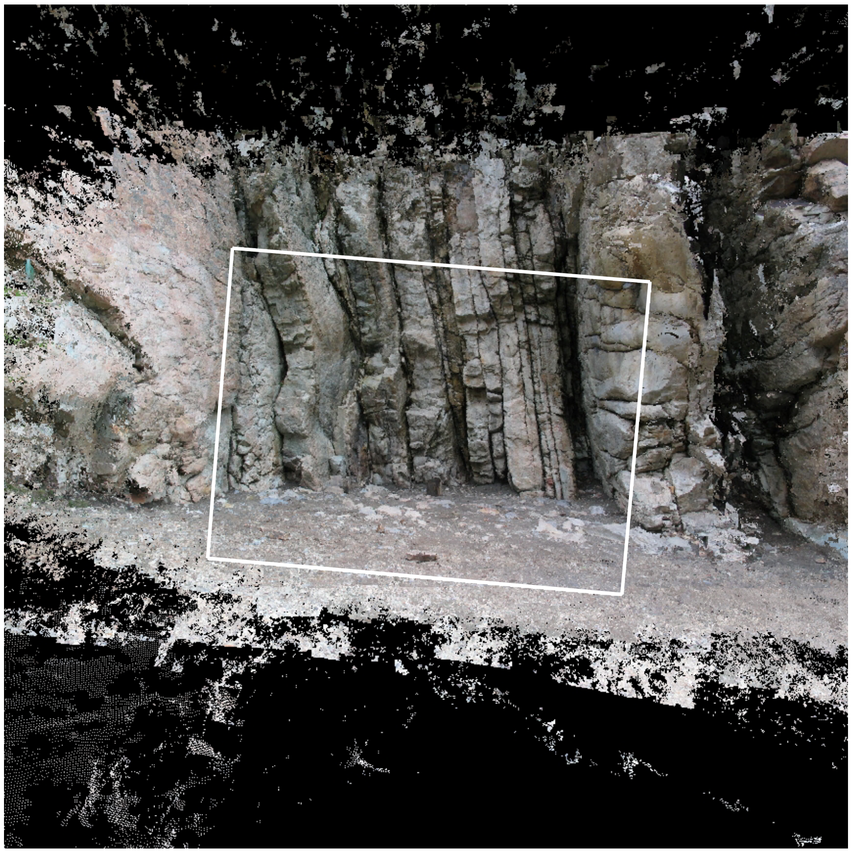


Figure 5.4: The front face of an environment map after all pixels of the input set images have been re-projected. The pixels belonging to the camera centered at the environment map were re-projected before all other cameras, and are outlined in white.

The convex hull is partitioned into a 3D Delaunay complex consisting of adjoining irregular tetrahedra. Figure 5.6 shows the camera poses and Delaunay partitioning of the input set.

Our view interpolation occurs within the four vertices of a tetrahedron. The 3D points of each omnidirectional image are rendered to the viewing plane of a novel viewpoint. Four such intermediate images are created from the four tetrahedral vertices and are blended using the Barycentric coordinates of the novel viewpoint as blending weights.

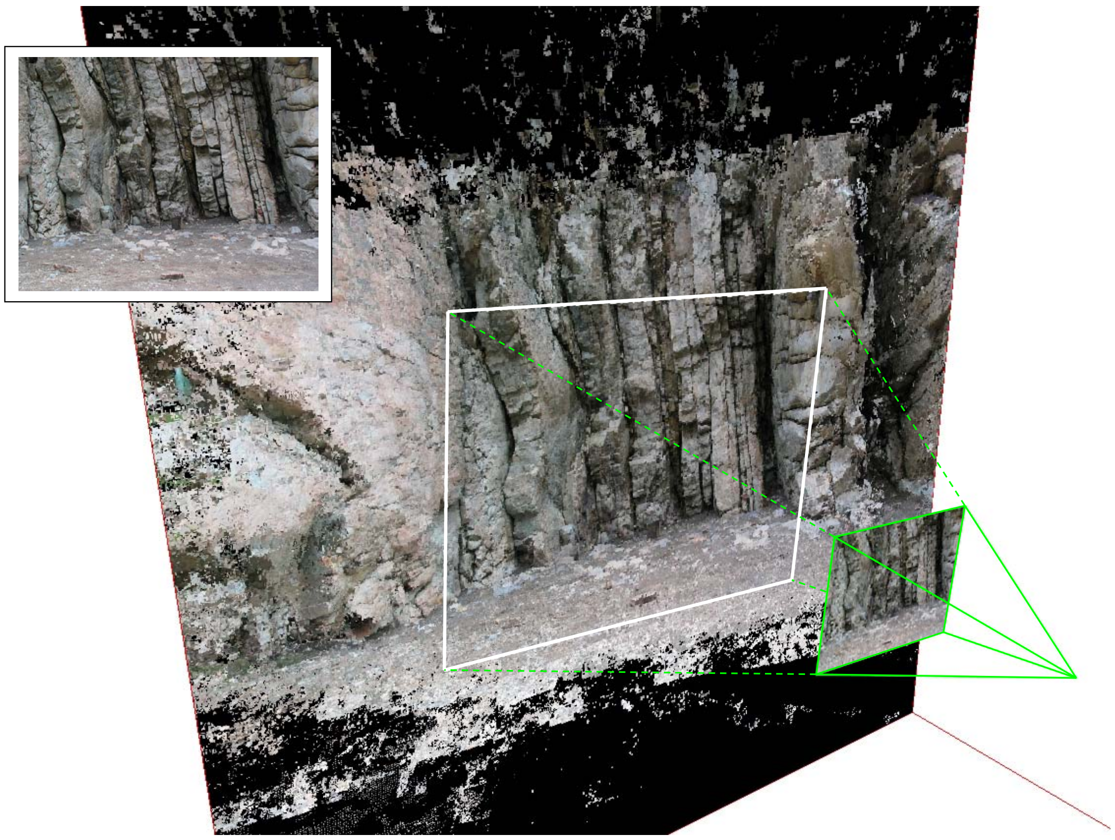
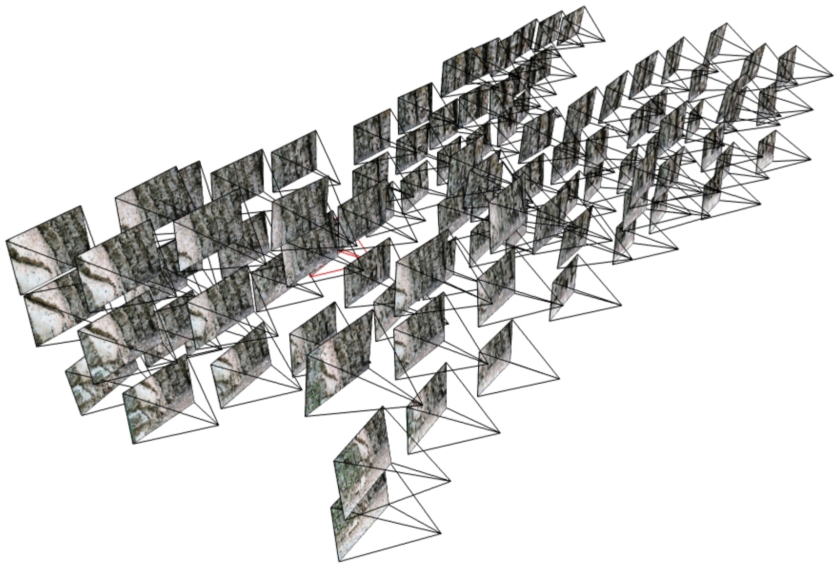


Figure 5.5: The front face (from Figure 5.4) of an environment map centered on a camera viewpoint whose view frustum is shown in green. The image pixels are projected to 3D points, then re-projected onto the face of the environment map.

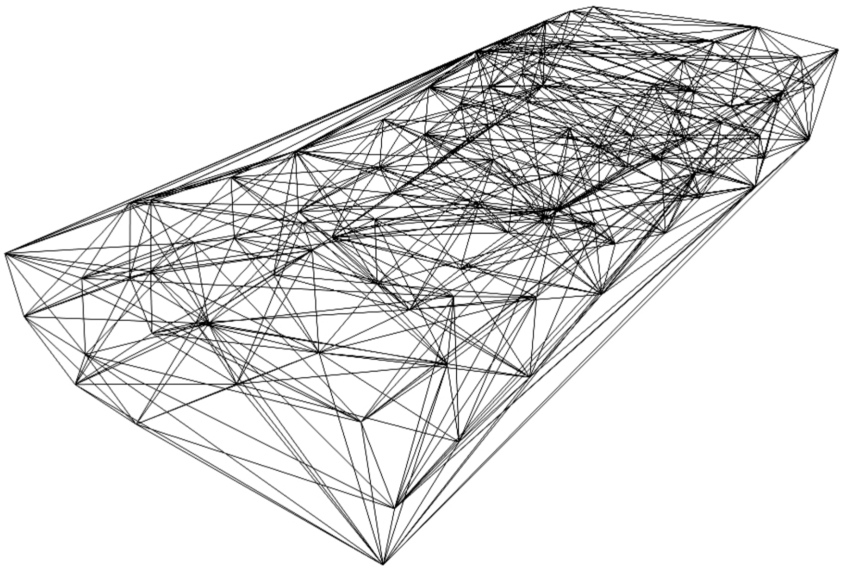
The novel viewpoint within a tetrahedron creates four sub-tetrahedra when combined with the four original vertices. The Barycentric coordinates of the novel viewpoint within an irregular tetrahedron are computed as the ratio of volumes of sub-tetrahedra to the volume of the enclosing tetrahedron.

5.5 Analysis

Using the view interpolation strategy of Section 5.4, we are able to synthesize images of arbitrary viewpoints within the simulation boundary of the viewpoint convex hull. In a simulation setting this enables unconstrained navigational freedom for exploring the virtual rendition of the input photographic content. However, it is not enough to simply render synthetic images that look convincing or acceptable to the user. It is important to measure the success of our view interpolation method



(A)



(B)

Figure 5.6: (A) The pyramids show the position and orientation of each camera from the input set. (B) The Delaunay partitioning of the viewpoint convex hull.

with a quantifiable error metric in order to unambiguously determine its visual performance. In this paper we will quantify both the re-projection error when creating omnidirectional images and the interpolation error during view synthesis. We will combine these two errors into one error metric that measures the quality of the final interpolated image.

In general it is difficult to quantify the correctness of a synthesized image because of the many subjective factors that contribute to the meaning of “correctness.” In terms of image-based rendering and photographic view interpolation, a possible correctness requirement for a synthesized view is that it be identical to some photograph that could be taken with a real camera at the same viewpoint with the same viewpoint parameters (orientation, focal length, and sampling rate). We can further define a quality spectrum to measure how close a synthetic image is to being correct. We will adopt a simple error metric to measure the closeness of a synthesized image to an ideal image—the average Euclidean distance in RGB space between counterpart pixels:

$$E(I_{synth}, I_{ideal}) = \frac{1}{WH} \sum_{x=1}^{W} \sum_{y=1}^{H} \| I_{synth}(x,y) - I_{ideal}(x,y) \| \tag{5.2}$$

where I_{ideal} is an ideal image (an original photograph), I_{synth} is the synthesized image at the same viewpoint, $I(x,y)$ is the RGB color at image coordinate (x,y), and (W,H) are the image dimensions of both images.

The error function E will be zero for identical synthetic and ideal images, and in the worst case some maximum M for the distance between white and black images:

$$M = 2^b\sqrt{3}, \tag{5.3}$$

where b is the image’s bit depth per channel.

The error function quantifies how close a synthesized image is to the ideal image. We would wish this comparison to reflect how a human would judge the quality of an interpolated view. If a user judges image A to be closer than image B to photograph P, then $E(A,P) < E(B,P)$. Even though we cannot duplicate subjective human judgments of image quality, we still find the error metric a useful tool to compare algorithmic variations of the interpolation methods.

Given the error metric E, we can evaluate the quality of the final interpolated image. If we could acquire photographs from known positions and orientations within the interpolation boundary, we could compare any synthesized view directly with an acquired view. However, measuring external camera calibration is a difficult procedure that usually requires using a controlled gantry, and it is subject to mechanical inaccuracies that would interfere with the error metric. Instead we take error measurements only at the viewpoints of the input image set. Since we have a priori knowledge of the input set external calibration, we can synthesize views at exactly the same position and orientation as each camera in the set. For each error measurement E_A to be taken at viewpoint A from the input set, we rebuild the Delaunay complex *without* including the viewpoint A as a Delaunay site. We then synthesize a novel view A' at the viewpoint A and compare it with the input photograph P_A originating at A. The error function $E(A', P)$ defines the accuracy of the interpolated image compared to the original photograph.

5.5.1 Algorithmic Variations

The raw values of the error function E are not instructive as standalone values. However, we can compare values of E between different methods of synthesizing interpolated views to get relative error values, and then draw conclusions on the merits of each variation accordingly. We will examine six variations on the basic algorithm presented in Section 5.3 by varying two parameters: the resolution of the environment maps used to create omnidirectional images, and the rendering point size used for view synthesis.

Recall that the resolution of each environment map face was computed to be twice the focal length of the camera at the map center in order to maintain the sampling rate of the original photograph. Relaxing this constraint by reducing the resolution introduces artifacts into the synthesized image, but can improve performance by requiring fewer points being rendered per map face.

A visible artifact of reduced map resolution is the introduction of gaps or holes in the synthesized image where no 3D points were projected (see the right side images of Figures 5.8 and 5.9). These gaps can be reduced by using a rendering point size larger than a pixel, effectively "filling in" the gaps with redundant color. We can measure the effects of reduced map resolution and increased render point size via the error function E.

We perform error analysis on the following six variations in map resolution and pixel point size:

Resolution Scale	Point Size
1 ($R = 2f$)	1 pixel
1 ($R = 2f$)	2 pixels
1 ($R = 2f$)	3 pixels
1 ($R = 2f$)	4 pixels
$^2/_3$ ($R = 4f/3$)	2 pixels
$^1/_2$ ($R = f$)	2 pixels

Table 5.1: Algorithmic variations used for comparison. R is the map resolution for each face, and f is the camera focal length.

For each algorithmic variation we use E to compare the synthesized interpolated image with the original camera image at each viewpoint of the input set. Figure 5.7 shows the comparison of the six variations. The graph illustrates the average pixel distance for all pixels in an image pair except those pixels coinciding with gaps in the synthesized image.

It is evident from the graph that the best variation is the full resolution map with a pixel point size of 1. However, there may be advantages to using poorer quality variations, as explained in Section 5.6. Figures 5.8, 5.9, and 5.10 show examples of synthesized images using the four of the six algorithmic variations.

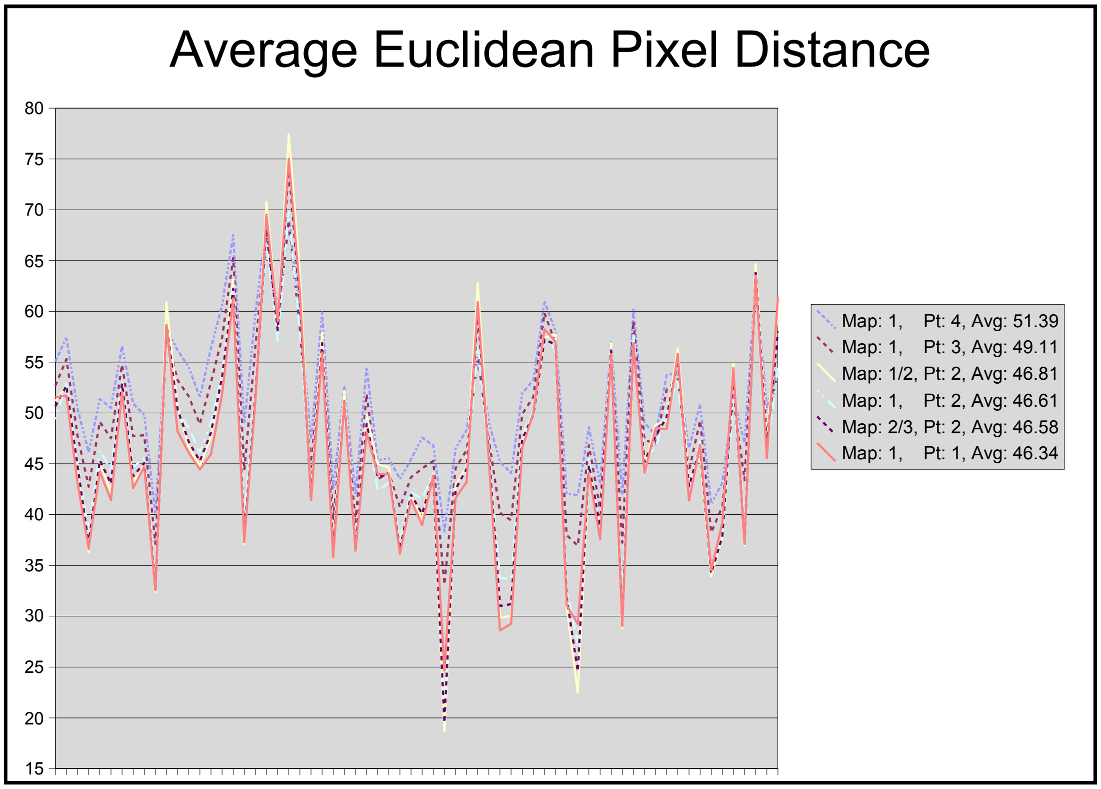


Figure 5.7: Comparison of algorithmic variants. The vertical axis is the value of the error function E, and each data point on the horizontal axis is an evaluation of the error function at a specific input viewpoint (66 total). The higher lines (in the graph and the legend) indicate less-accurate images. The lowest line in the graph, corresponding to the full-resolution map with point size 1, indicates the most accurate set of synthesized images.

5.6 Discussion

We tested our omnidirectional images and view interpolation method using a custom viewer that virtually navigates the environment captured by the input photographs. Figure 5.11 shows two input images from one of the tetrahedra and the interpolated image created at the tetrahedron center.

Since the resolution of the environment maps is large, each omnidirectional image contains hundreds of thousands of 3D points. Given a large number of input photographs, we found it impractical to pre-load omnidirectional image points into display lists. Instead we load the points as needed from disk during run-time, which does not incur too large a penalty. For any viewpoint within a tetrahedron, the 3D points attached to a single vertex are loaded from disk as a block. As the viewpoint moves from a tetrahedron to a neighboring tetrahedron, only one vertex changes at a time, requiring the loading of the new vertex's 3D points.

Some hardware systems may have disk access restrictions that prohibit loading a block of 3D points in real-time. Using lower environment map resolutions result in fewer 3D points in each omnidirectional image, and hence smaller block sizes to load. It may also be desirable to use larger point sizes to render the 3D points in each block to reduce gap artifacts in the synthesized image. The analysis of Section 5.5.1 helps us to understand the image quality tradeoffs for lower resolutions and larger point sizes.

While the viewpoint is within a tetrahedral boundary, our view interpolation algorithm runs at 15-20 fps on a 3GHz P4. As future work we hope to implement a predictive caching strategy to load omnidirectional images from disk based on the user viewpoint movement.

We also hope to further investigate the pixel re-projection order criterion in Section 5.3.1. It may be beneficial to consider not only distance from the omnidirectional center, but also angle differences between cameras. In this way we could favor images that are farther from the center of re-projection, but that are acquired from closer angles and hence have a more accurate sampling of the subject material.

5.7 Conclusion

We have introduced a novel method of creating omnidirectional images that exploit the characteristics of pose- and depth-enhanced photography. The images are amenable to tetrahedral view interpolation and permit six degrees of navigational freedom within the environment described by the set of input photographs. We have also proposed a simple view interpolation strategy that is straightforward to implement. We believe that current and future advances in dense camera calibration and per-pixel depth estimation will promote applications such as the virtual navigation of large-scale photographic content.

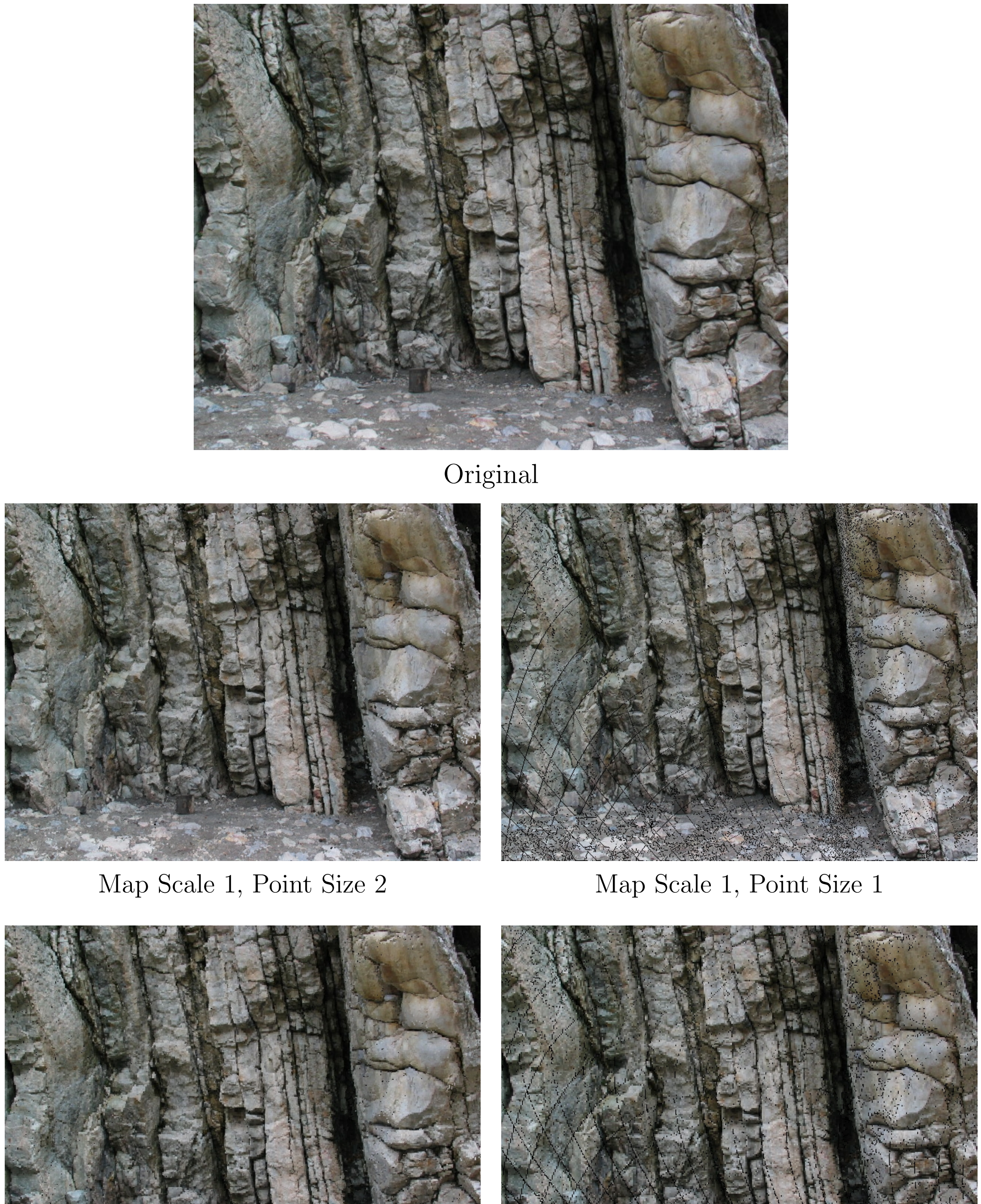
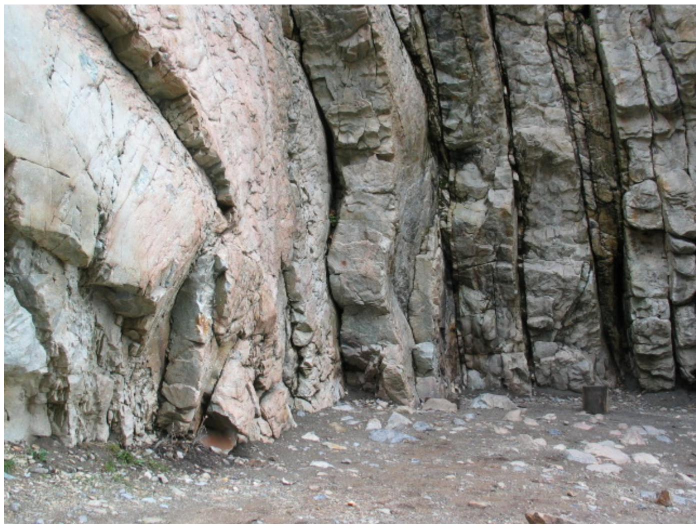
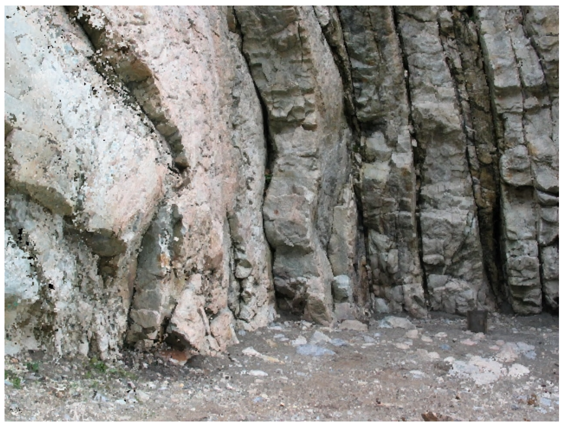
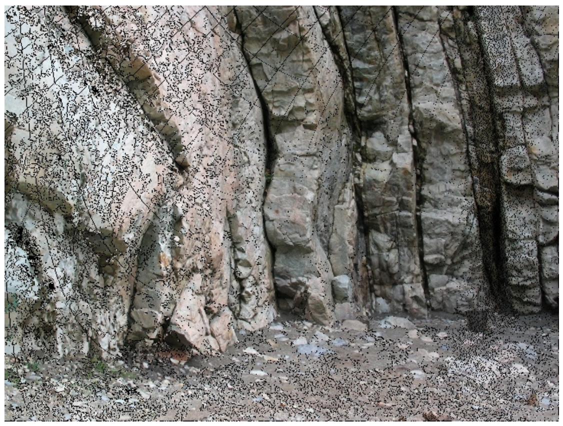
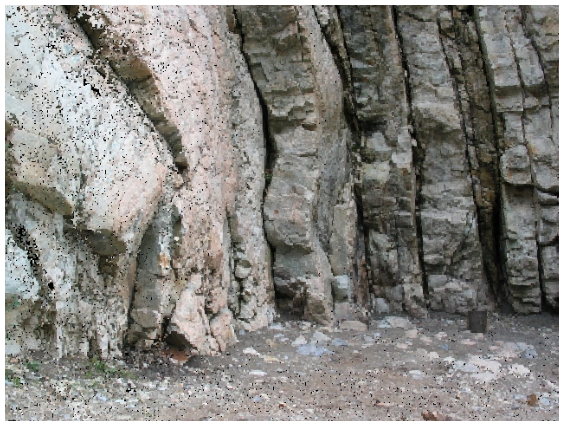

Original
Map Scale 1, Point Size 2
Map Scale 1, Point Size 1
Map Scale ⅔, Point Size 2
Map Scale ½, Point Size 2

Figure 5.8: The top image is the original photograph that was removed for comparison. The remaining four images show the interpolated results for various environment map resolutions and render point sizes. The dark curves seen in the right two images result from aliasing due to a higher sampling rate in the reconstruction than in the original photographs.


Original


Map Scale 1, Point Size 2


Map Scale 1, Point Size 1


Map Scale ⅔, Point Size 2

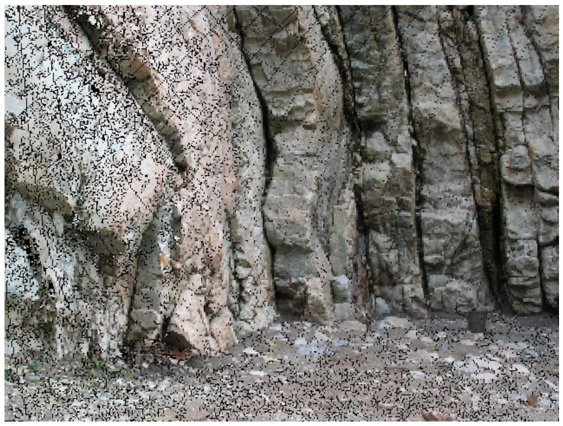
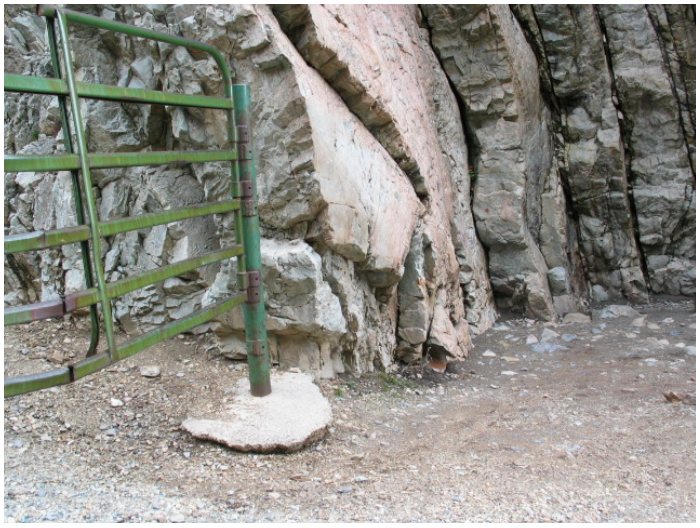
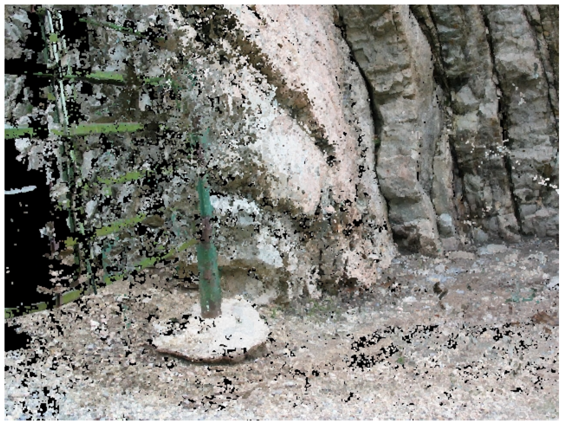

Map Scale ½, Point Size 2

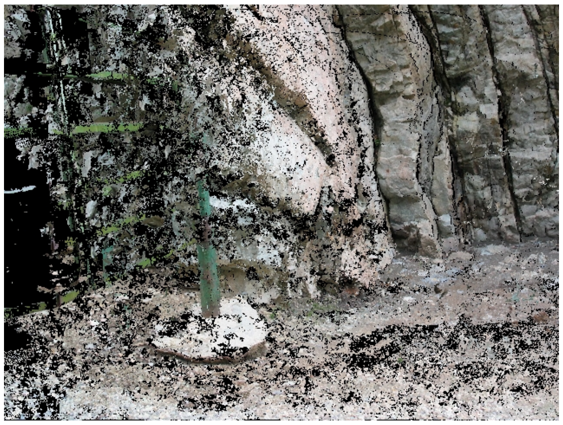
Figure 5.9: The top image is the original photograph that was removed for comparison. The remaining four images show the interpolated results for various environment map resolutions and render point sizes. The dark points and curves seen in the right two images result from aliasing due to a higher sampling rate in the reconstruction than in the original photographs.



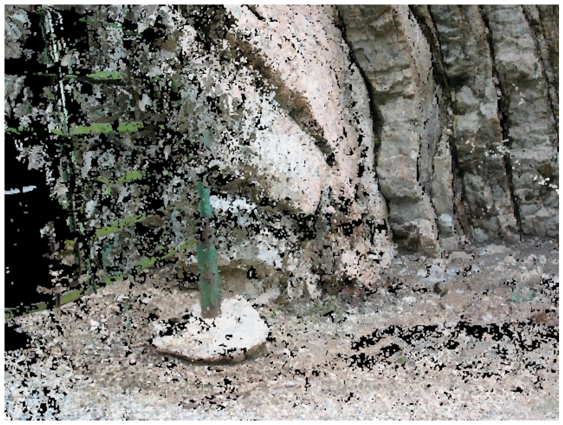
Original



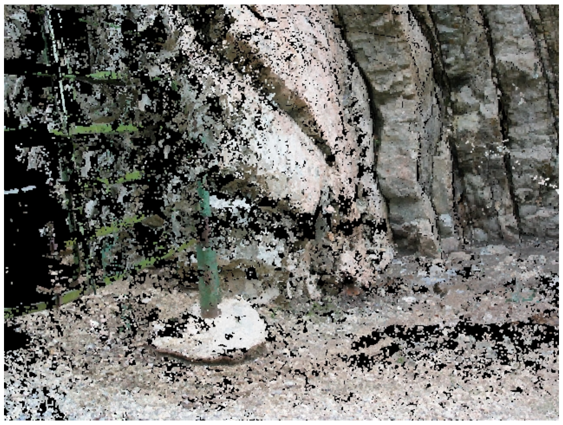
Map Scale 1, Point Size 2



Map Scale 1, Point Size 1



Map Scale $\frac{2}{3}$, Point Size 2



Map Scale $\frac{1}{2}$, Point Size 2

Figure 5.10: The top image is the original photograph that was removed for comparison. The remaining four images show the interpolated results for various environment map resolutions and render point sizes. The noise prevalent throughout the synthesized images is the result of incorrect dense depth estimates.

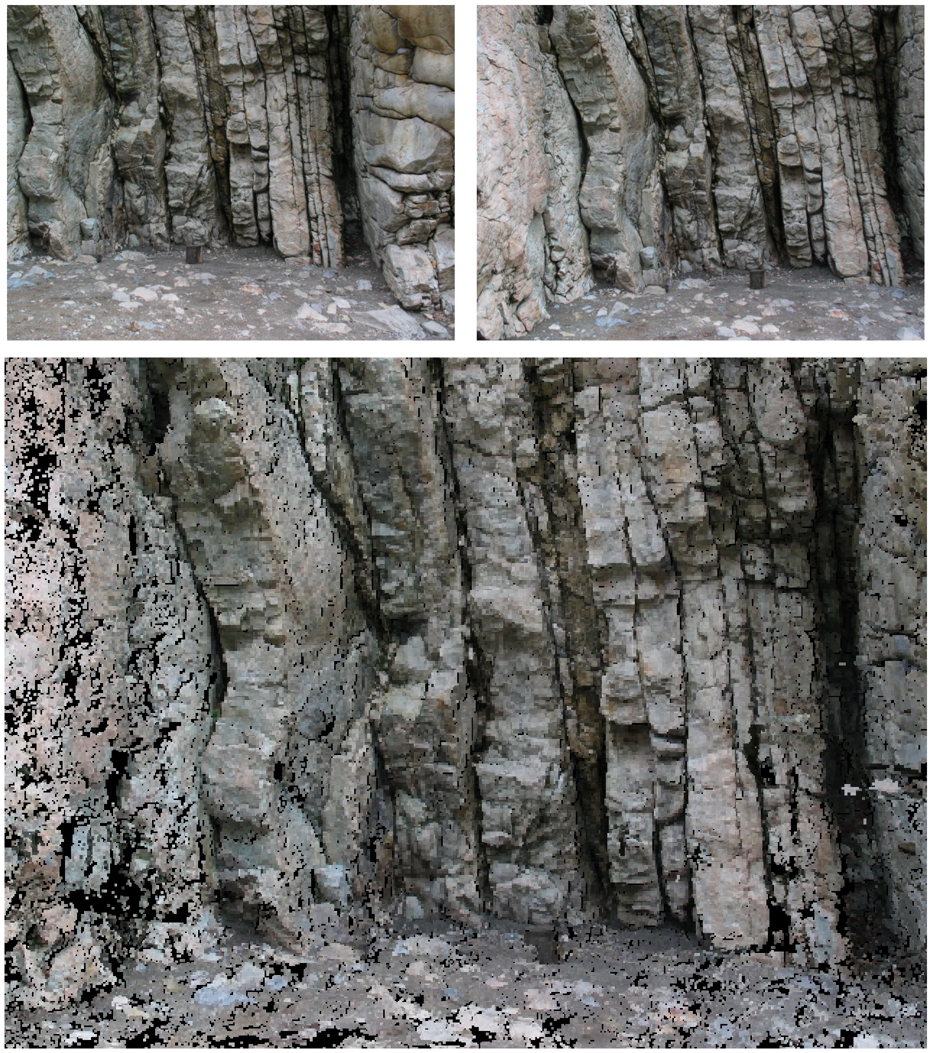


Figure 5.11: The top two images are photographs from the input set that are part of the omnidirectional images of two tetrahedral vertices. The bottom image is an interpolated view from the center of the tetrahedron. The black artifacts on the image periphery result from per-pixel depth estimate errors.

Chapter 6

Conclusions

This dissertation has presented an end-to-end system that allows a user to input a set of photographs acquired by an off-the-shelf still digital camera, and then to interactively explore the photographic environment with unconstrained navigation. In this chapter we will describe the system's functional components and how the preceding chapters fit together as a whole.

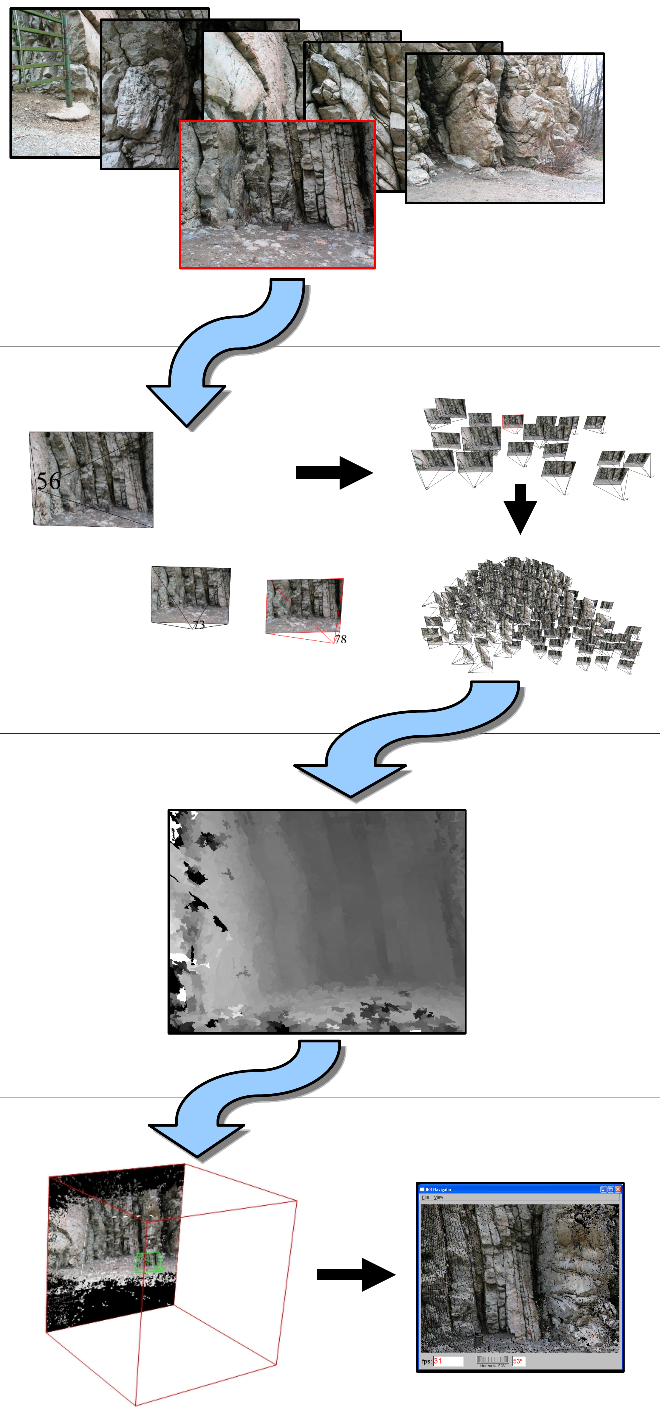
Our current system is operative for content creation, but it is unfortunately very limited in the types of environments it can presently simulate. We will therefore discuss the limitations of the developed system and characterize the primary sources of error that lead to current system deficiencies, and we will place the system limitations into a framework of future research. It is our hope that further work will extend the system capabilities to enable simulation of a much more general class of environments.

6.1 VBR System

Our Vision-Based Rendering system consists of four major stages: photographic input, camera pose estimation, dense depth correspondence estimation, and omnidirectional image creation. Figure 6.1 illustrates these four steps, and the sequence can be thought of as a "pipeline" where each step is dependent on the completion of its predecessor.

6.1.1 Photographic Input

As a first step, the user must decide on the environment to be simulated. We have found that acquiring several hundred photographs of an enclosed space such as a large



1. Acquire Photography

- Input photos to system
- Select one image to be the world-space origin (lower center, outlined in red)

2. Pose Cameras

- Create adjacency graph (Chapter 3)
- Find point matches using Correspondence Expansion (Chapter 2)
- Use MST Pose Estimation (Chapter 4)

3. Estimate Dense Depth

- Utilize many nearby cameras for more accurate depth estimation
- Use modified version of Zitnick et al. (2004)

4. Create Omni-directional Images

- Store each omnidirectional image as a separate block on disk (Chapter 5)
- Load blocks as needed during simulation run-time

Figure 6.1: Vision-Based Rendering System Pipeline

room to be sufficient. Furthermore, any prominent features should be photographed from many different angles and distances to ensure that interest points have a high sampling rate.

Most of the photo acquisition for this dissertation was done using a Canon PowerShot G3, a high-quality consumer camera. We shot at 640×480 with a fixed aperture and shutter speed (in manual mode). This resolution was used to keep the correspondence expansion time (see Chapter 2) to under 30 seconds per image. We usually kept the ISO speed at 100, and chose the aperture and shutter speed to yield a reasonable brightness while in the acquisition setting. While photographing we took moderate care to avoid motion blur, but we did not use a tripod, flash or any other accessories.

In our system it is important to maintain a constant zoom on the digital camera during acquisition, as will be explained in Section 6.1.2. We generally chose a zoom setting midway between the focal length extremities of the lens, and keep that zoom setting for the duration of a photography session.

Not all input photographs need be acquired during the same session, so long as the same camera settings are used and the scene content and lighting has not changed significantly. We found that outdoor scenes often look better in diffuse lighting, i.e. on a cloudy day, but this can be challenging if the cloud cover is variable.

From the acquired photographs the user selects one image to be the world-space origin. Its camera location will be at Euclidean coordinates $(0, 0, 0)$ and have no rotation. We found it helpful to choose an image in a central location of the input set to reduce cumulative pose error, but also one that also has a substantial number of images overlapping in content.

Once the input images have been acquired by the user and an initial image chosen to be the origin, the user is finished and the VBR system automatically processes the images to compute the final data files necessary for virtual exploration of the image content.

6.1.2 Camera Pose Estimation

We use the MST pose algorithm as detailed in Chapter 4 to estimate the camera poses for the input set. As images are added to the set of posed cameras, their pose is estimated by computing the essential matrix[1] between an image pair from point correspondences, then recovering translation and rotation from the essential matrix using singular value decomposition (the rotation is an orthonormal matrix and the translation is a vector in skew-symmetric matrix form). We use the eight-point algorithm from Ma et al. [2004] to compute the essential matrix. However, since we create a population of pose estimates (and hence essential matrices) from which to create a probability density function (Section 4.3.2), we need as many accurate point correspondences as possible. Prior to computing any essential matrices, we establish and store in a database a set of highly accurate point correspondences between image pairs using the correspondence expansion algorithm of Chapter 2.

Since we are computing essential matrices rather than fundamental matrices[2], we need to provide the camera calibration for each input image. With high-end cameras, much of the internal calibration is nearly ideal; the pixels are square, the principle point is very close to the center of the image, and the skew is essentially zero. The only quantity remaining is the focal length, which we extract from the EXIF headers in the JPEG images output from our (and any) digital camera. This focal length is accurate to two significant figures in millimeters, and is sufficiently accurate for our system.

To avoid having to extract the focal length for every input image we require a constant zoom during the acquisition session. This also reduces problems that may arise from the limited focal length precision. A popular alternative is to employ *autocalibration*, where a camera's internal calibration is derived automatically from point correspondences between several images. This is an area of active research in computer vision, beginning with Faugeras et al. [1992]. See Ronda et al. [2004]

[1] The essential matrix E is a 3×3 rank two matrix that encodes the geometric relationship (relative position and rotation) between two calibrated cameras.

[2] The fundamental matrix F is an essential matrix with internal calibration K factored in; e.g. $F = K^{-\top} E K^{-1}$. Given two corresponding points x and x', $x'^{\top} F x = 0$.

for recent results. Autocalibration is not a fully mature process and requires careful attention to the specific types of camera motions, so we avoid it by requiring a constant zoom constraint. In practice this is not restrictive.

6.1.3 Dense Depth Correspondence Estimation

Dense correspondence, or stereo correspondence, has received much research attention largely because of its difficulty. Scharstein and Szeliski [2002] provide a good survey on algorithms to date. Much of the difficulty stems from the finite aperture problem, specifically the ambiguity inherent in matching textureless regions between two images. A recent approach to solving the problem is to match unstructured segments rather then geometric features such as points or lines. Image segments are easily defined and consist of groups of neighboring pixels that have similar color or some other attribute. Segmentation-based matching is more robust to mismatch error, and has met with some success [Tao and Sawhney, 2000; Tao et al., 2001; Zitnick et al., 2004].

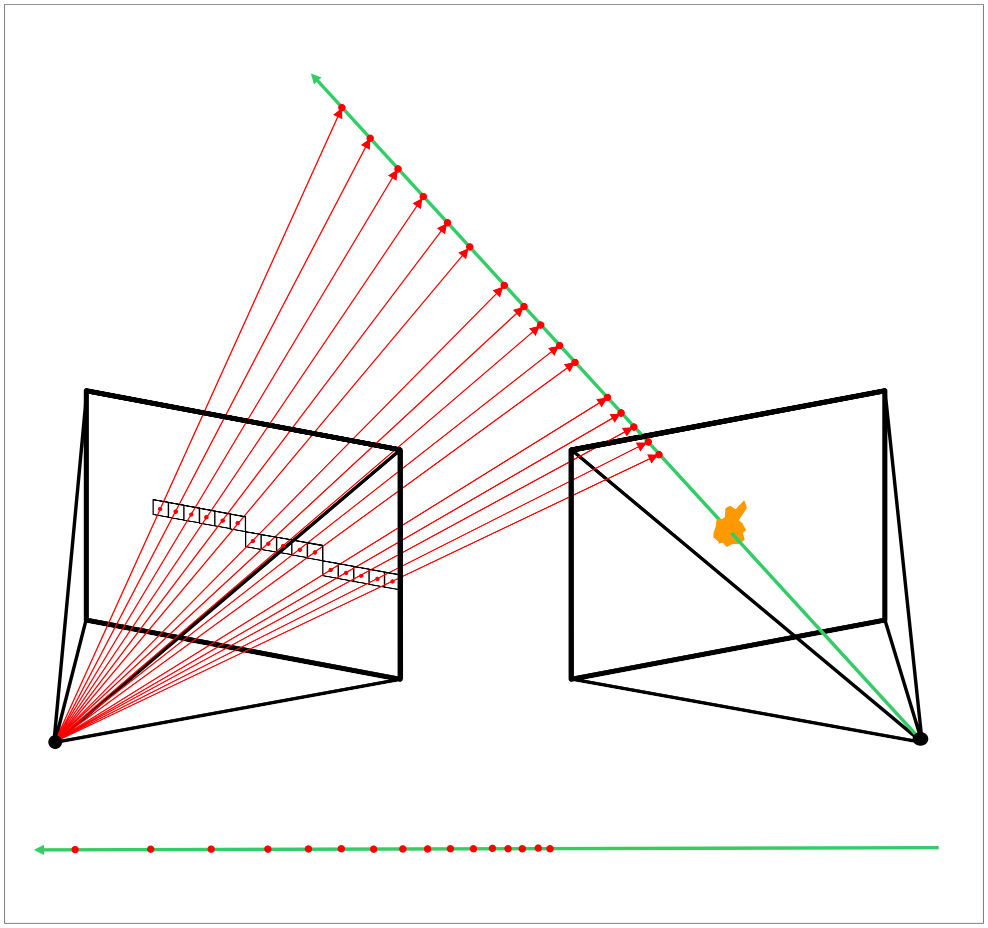
We use a modified version of the segmentation-based stereo algorithm presented by Zitnick et al. [2004]. In their paper the authors describe an approach to estimate dense correspondence between an image and two neighboring images. They define a discrete probability density function for each segment of an image in which the pdf domain is the set of possible pixel disparities a segment can have. The image and its left and right neighbors must be rectified (all possible matches are on common rasters between the images), and the segment disparity is the distance in pixels that the segment would need to shift left or right in order to coincide with its counterpart segment in another image. The pdf of each segment is iteratively modified according to a set of constraints. The authors give a smoothness constraint in which a segment's disparity must be similar to that of neighboring segments of the same image based on color differences, and a consistency constraint in which a segment's disparity must be similar to that of segments it projects to on neighboring images. After a number of iterations the disparity at the pdf maximum is defined to be the final disparity for a segment. The segmentation is then relaxed and a per-pixel disparity smoothing is performed across the entire image.

While this method produced good results in Zitnick et al. [2004], it has a few restrictions that limit its use in our system. The use of scalar disparities and the rectified image requirement work well with one or two neighboring images, but we wish to leverage many neighboring cameras when estimating dense depth to improve the correspondence accuracy. Since our cameras are in general position (arbitrary 3D locations and orientations), they cannot easily be rectified to a canonical stereo configuration.[3] Rectification requires either a planar homography (a one-to-one mapping between two planes in $\mathbb{R}^3$) or an algorithmic equivalent, and rectifying many cameras to a common reference image loses nearly all useful matching segments in the rectified images.

To use the depth estimation algorithm of Zitnick et al. we generalize the pdf domain from disparity to 3D depth, maintaining a set of candidate depths for each segment in an image rather than a set of disparities. The candidate depth set is constructed by first rasterizing in another image the epipolar line associated with the segment center, and then back-projecting the pixels of the rasterized line onto a ray emanating from the first image's camera center through the segment center (see Figure 6.2). Each point of closest intercept on the ray is a viable depth. For each segment of an image we accumulate all viable depths contributed from all neighboring cameras, and then discard depths that project to redundant pixels. Figure 6.2 illustrates the candidate depth gathering process.

We iteratively modify the generalized pdf domain using the smoothness and consistency constraints of the original paper. The depth at each segment that maximizes the pdf is retained as the final depth for the segment. Stage 3 of Figure 6.1 shows an example of a dense depth map created using this method. The artifacts along the left and bottom image borders in the figure illustrate the difficulty of obtaining completely correct depth estimates.

[3] A canonical stereo configuration consists of two cameras with identical internal and external calibration parameters, except that one camera's position differs in the local x-axis.



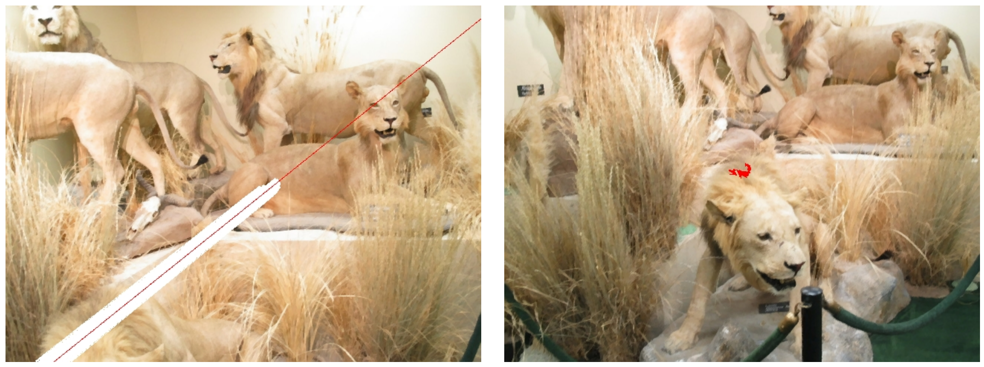


Figure 6.2: The top diagram shows a segment in the right camera and its rasterized epipolar line in the left camera. Each pixel of the epipolar line is back-projected to the segment's ray to generate a set of candidate segment depths. The horizontal green line shows the relative separation of the depths in the set. The bottom photographs illustrate the same process. A segment is red-highlighted in the right image. Its epipolar line is drawn as a thin red line in the left image, and the segment projections that yield viable depths in the candidate depth set are shown as a thick white line.

6.1.4 Omnidirectional Image Creation

With camera pose and per-pixel depth estimates computed for each input image, we create omnidirectional images centered on each camera viewpoint using the methods in Chapter 5. Each omnidirectional image is comprised of a collection of 3D points and their respective colors, and is stored on disk as a block for on-demand runtime loading. We also store on disk the Delaunay partitioning of the viewpoint convex hull. These files constitute the output of our VBR system.

We created a real-time viewing application to demonstrate the results of the VBR system. The viewer allows a user to visually explore the virtual photographic contents of the input set. It does so by receiving navigational input from the user to position a virtual "eye" with location and orientation viewing parameters, and then rendering a synthetic image in real-time. The image resembles how a photograph would appear if acquired with the same parameters. If the virtual viewing parameters happen to coincide with those of an existing photograph given to the system in stage 1, then the synthesized view will be identical to the original photograph. If the "eye" is elsewhere within the Delaunay partitioning, an interpolated view is synthesized.

As a preliminary step, the viewer loads the Delaunay partitioning information from disk and creates a polygonal representation of all tetrahedra from the partitioning. The polygon colors are encoded with tetrahedral identifiers. For each frame during simulation time, an off-screen render of the polygonal representation is performed using the input viewing parameters and a 1-pixel wide color buffer. The resulting color encodes the identity of the tetrahedron enclosing the viewpoint. This off-screen render operation is performed very quickly in GPU hardware by using display lists to keep the polygons on the GPU, and by using such a small render buffer.

Given the tetrahedron ID, the four omnidirectional images enclosing the virtual viewpoint are loaded from disk (if not already resident in a cache). A novel view is then synthesized from the four omnidirectional images using our view interpolation method given in Section 5.4. This iterative process of tetrahedron identification followed by image synthesis is performed for each frame in real-time, giving the application user the sensation of virtually navigating the simulated photographic content.

6.1.5 System Results

Synthesized image results from the VBR system are shown in Figures 5.11, 6.3, 6.4, and 6.5. The system can successfully integrate and interpolate image collections with certain characteristics such as high frequency detail and moderately complex scene geometry. However it fails to correctly estimate camera pose and dense depth maps for many types of indoor images, and distracting artifacts are consequently generated in such simulations. Sections 6.3.1 and 6.3.2 discuss the nature of these errors in greater detail.

6.2 Dissertation Contributions

This dissertation contributes theoretical and algorithmic advances to the fields of computer vision and graphics. We list here the specific contributions and their references within this document:

- We define an algorithm to expand a small set of existing point correspondences between two wide-baseline images to a much larger set while maintaining the epipolar constraint (Section 2.3).
- We define an algorithm to build a camera neighbor adjacency graph from a collection of unordered photographs using partial color histograms (Section 3.4).
- We give an explicit and optimal ordering to the construction of the adjacency graph in the form of the Minimum Spanning Tree (Section 4.3).
- We give a cost function used in constructing the Minimum Spanning Tree that measures the reconstruction accuracy between two pairs of wide baseline images. The cost function is independent of the method used to determine pose order (Section 4.3.1).
- We give a model that describes how correspondence noise transfers to a camera pose estimate, and we provide a method to estimate the optimal pose in the presence of correspondence noise and match outliers (Section 4.3.2).

- We generalize the segmentation-based dense correspondence algorithm of Zitnick et al. [2004] to use many neighboring cameras in arbitrary, non-rectified configurations (Section 6.1.3).
- We define an algorithm to transform still photographs with distinct focal points into omnidirectional images that maintain the sampling rate of the original primary image (Section 5.3).
- We define an image quality function that ranks interpolated images by their distance to a target photograph (Section 5.5).
- We propose a view interpolation algorithm that permits six degrees of navigational freedom (Section 5.4).
- We construct a system to automatically input a set of photographs and build the necessary data structures for unconstrained virtual navigation of the photographic content (Section 6.1).

In any large research project a list of specific contributions represents only the "tip of the iceberg" when compared to the total amount of research performed. Failed attempts to solve a particular problem often go undocumented, yet provide key insights and intuitions to the real solutions. As an example, we first attempted to expand the match correspondence set of Section 2.3 with a logically small epipolar error tolerance (.5 pixels). Only marginally successful, we discovered upon investigation that the expansion could not proceed into most portions of an image because of the extreme accuracy of the epipolar constraint. This observation led to the insight and novel usage of a relaxed epipolar constraint (5+ pixels) during the aggregation phase (see Section 2.3.1). Thus the contributions listed above are not only the result of inductive and deductive problem solving, but of trial and error processes as well.

6.3 System Performance and Future Research

With our VBR system we are able to automatically create simulation content from a large unordered collection of input photographs, a major achievement in image-based rendering. However, there are presently severe restrictions in the type of image content our system can accurately simulate. Photographs containing large regions of high frequency detail, Lambertian lighting, and moderately complex geometry are incorporated into simulation content very accurately. These include images of outdoor and natural phenomena such as rock formations, soil, rubble, trees, flora, and vegetation. Images with low frequency detail or smooth color gradations, such as painted walls and objects, or scenes with high reflectivity like a tiled floor or metal sculptures do not integrate into existing content very well and create distracting artifacts in the final simulation. Most indoor photographs fall into this category.

6.3.1 Camera Pose Error

There are two primary reasons for these limitations in our present system. The first is pose estimation error propagated from incorrect point correspondences. Correspondence error resulting from smooth color ambiguity is manifest in the form of match outliers (extreme point mismatches obvious to the human eye) and small mismatches of several pixels or less. While our correspondence expansion method of Section 2.3 enforces a rigid epipolar constraint on the final set of correspondences, there is room along epipolar lines for marginal super-pixel error. Given a poor initial correspondence set, it is also possible to inadvertently enforce an incorrect epipolar geometry as the expansion proceeds. In either case, the resulting set of near-correct point matches propagate to a near-correct, though not sufficiently correct, pose estimate. Our robust method of optimizing a pose estimate from a noise-corrupted pose population (Section 4.3.2) helps reduce the error, but that method assumes a zero-median noise distribution, which is not often possible for indoor photographs. The interpolation artifacts produced by mismatch error come in the form of ghosting or synthesized "double exposures." Figure 6.3 shows two examples of ghosting caused by incorrect pose estimates.

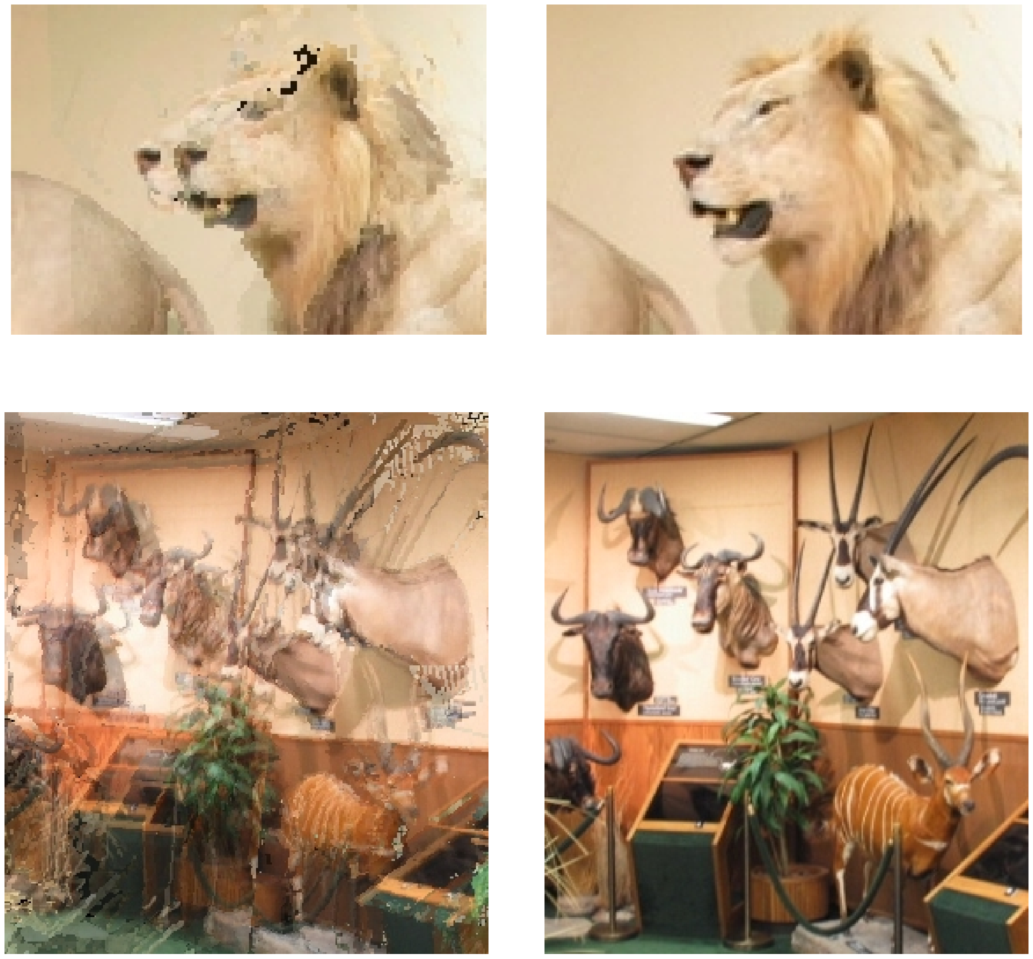


Figure 6.3: Example of ghosting artifacts resulting from incorrect pose estimates. The images on the left are interpolated from four omnidirectional images, where one or more of the camera poses are not correctly aligned with the remaining poses. The counterpart images on the right are from the original photographs to illustrate the intended synthesized images.

A more robust feature type would improve the matching characteristics and hence the camera pose estimates. Nistér [2000] uses lines in addition to point features to match image triplets by computing their tri-focal tensors, the three-image epipolar equivalent to fundamental matrices. As future work we propose adding richer feature sets such as line segments to the correspondence algorithms. This alone should improve the pose estimation for indoor photographs, since indoor scenes generally contain large numbers of straight line segments.

6.3.2 Dense Depth Error

The second primary reason for our system limitations is incorrect dense depth estimation. We compute dense depth maps for all input images by generalizing the segmentation-based algorithm of Zitnick et al. [2004] (see Section 6.1.3). Segmentation-based stereo improves dense depth maps over older stereo methods such as dynamic programming on raster spans [Cox et al., 1996], but it is still very error prone in the presence of photometric ambiguity like solid colors and smooth color changes. Additionally, all dense stereo methods fail along image borders when there is insufficient information from neighboring images to resolve depth. For example, note the erroneous depth values in the borders of the depth map shown in Stage 3 of Figure 6.1. In that example, there were not enough images below and to the left of the image in question to find correspondences and compute accurate depth.

Incorrect depth estimates are manifest as "floating" geometry in the final simulation. Depth errors create incorrect geometry which is often disconnected from the correct reconstruction. Such errors appear to float in front of the scene being synthesized. Unfortunately, depth error is more common and more difficult to resolve than pose estimation error, and its artifacts are more distracting to the simulation user. Figure 6.4 shows examples of geometric artifacts caused by incorrect depth estimates.

Dense depth estimation suffers from the same ultimate problems as camera pose estimation—finding point matches between two or more images. There is much research currently being performed in the field of dense stereo correspondence. Most of the research focuses on using belief propagation, graph cuts, or combinations of the two [Tappen and Freeman, 2003], but less research is focused on using dense multi-view correspondence to compute depth from more than two images. As future work on this system, we suggest utilizing a recent dense stereo method such as belief propagation to reduce the amount of geometric error in the synthesized images.

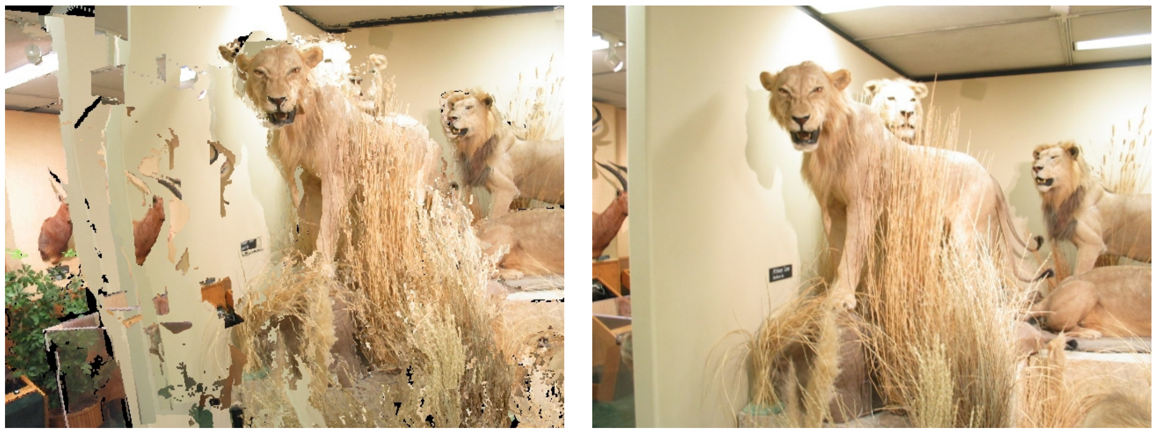


Figure 6.4: Example of geometric artifacts resulting from incorrect depth estimates. The left image is interpolated from the reconstructed geometry of several input images. Note the “floating” geometry in front of the lioness, image segments originating from the left border of the input image. The right image is one of the input images used to generate the synthesized view.

6.3.3 Successful Synthesis

Given the limitations we have described, our system currently cannot simulate many indoor and outdoor environments that we had originally hoped to model. However it can simulate environments whose input photographs have enough high frequency detail to mitigate the error conditions from Sections 6.3.1 and 6.3.2. For example, we can create a fairly accurate simulation from the input images referred to in Figure 4.7 because of the amount of detail in the photographic content. Figure 6.5 shows six images synthesized from the simulation. Note the relative absence of ghosting or floating geometry artifacts.

Another example of a successful simulation is shown in Figure 5.11. In this case, photographs of a rock outcropping are very accurately transformed into the simulation content because the photographic detail and uniform lighting permitted precise camera pose and dense depth estimation.

6.3.4 Other Performance Considerations

If we consider the system as a whole—correspondence discovery and expansion, camera pose estimation, dense depth computation, and omnidirectional image

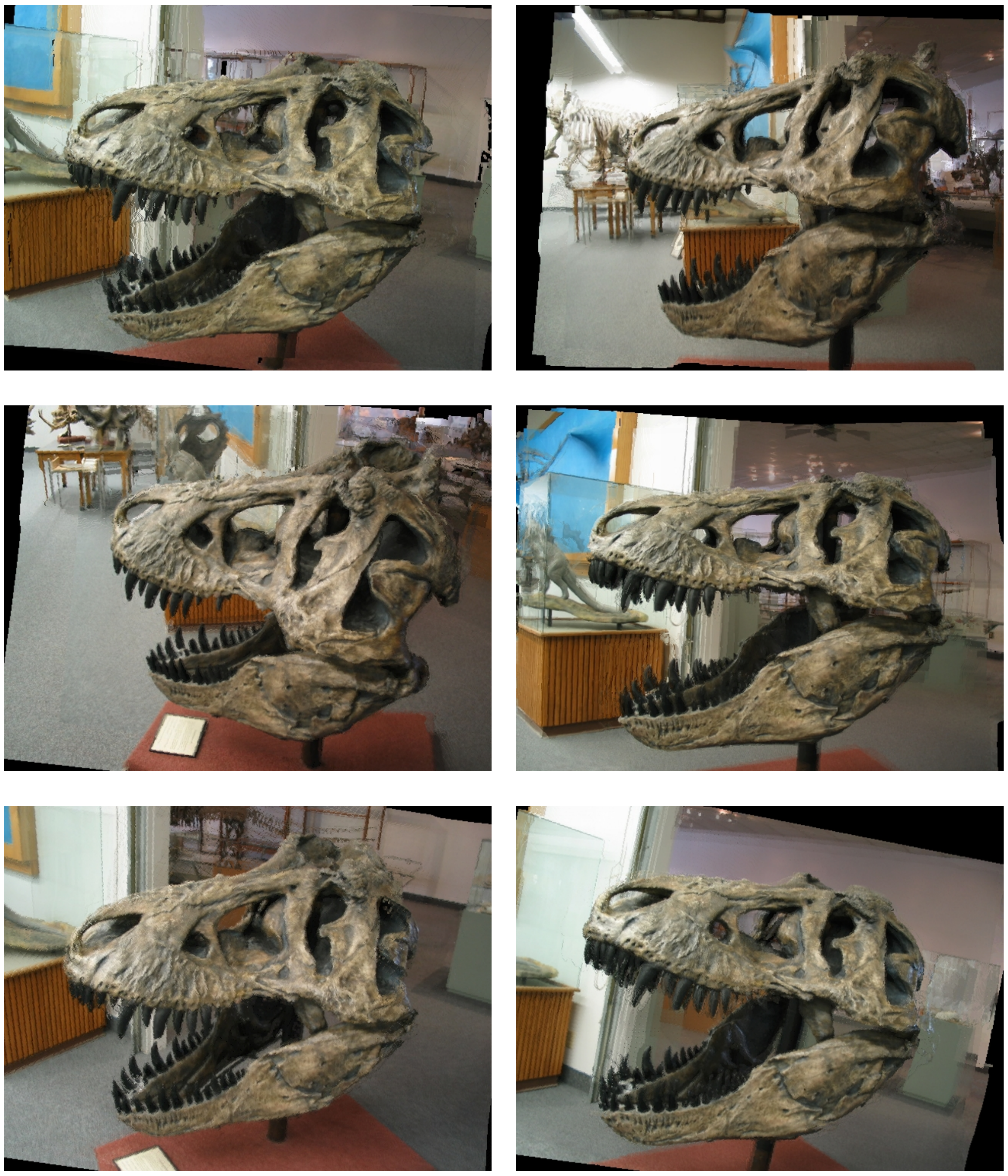


Figure 6.5: Six images of a t-rex skull, each synthesized by interpolating the geometry of three neighboring input photographs. The background in each image has some artifacts, but the skull specimen itself is cleanly rendered.

creation—the major computational bottleneck is in finding correspondences. Our system takes 20–30 seconds on a 3.2 GHz P4 to find an ample set of accurate correspondences between two wide-baseline images. Given the considerable number of correspondence sets we generate in order to pose a large image collection, this pair-wise runtime not only affects the total system runtime for content creation, but it inhibits algorithmic exploration and development of all parts of the system. We suggest as one of the first considerations for future work to investigate alternative (faster) methods to find accurate wide-baseline correspondences. Unfortunately accuracy cannot be sacrificed for speed, since correspondence accuracy is essential for all succeeding system phases. Using region-based matching [Matas et al., 2002] may be a worthwhile starting point.

We have also observed that photographs taken with smaller baselines, i.e. that are more closely spaced, integrate better into simulation content with fewer objectionable artifacts than photographs with larger baselines. This is an obvious consequence of a specific matching algorithm characteristic: it is much easier to find accurate pair-wise correspondences between small baseline images than it is for wide baseline images. More accurate correspondences yield improved pose and depth estimates, and hence fewer simulation artifacts. Since adding images to the simulation content increases database size, the necessary tradeoff is to sacrifice simulation quality (in terms of rendering artifacts) for database size. Understanding this tradeoff and its implications would be a useful area for further study.

Another area of future work is to incorporate individual frames of video footage into the set of input images. Shooting video is much easier and faster than snapping hundreds of still images, but often the images from video suffer from photographic artifacts resulting from poorer lens and CCD quality. Care would need to be taken to avoid motion blur and lighting changes; Nistér [2001] offers a good starting point.

Finally, one should consider whether it is better to simulate photographic content using a view synthesis system as we have, or to simply create geometry using structure-from-motion techniques. We chose to use view interpolation rather than reconstruction because it is easier to hide the error artifacts inherent to geometric reconstruction when we retain the original photographic imagery. Image-based modeling and reconstruction strategies must also resolve the problem of geometric data size—each photograph potentially contributes a 3D point from each pixel. Handling a database of millions or billions of such points, and transforming them into more general geometric entities such as polygons is a very difficult problem.

Almost all of the problems associated with both approaches can be traced back to the correspondence problem—given two images, which portions match? A general and robust solution to this formidable problem will solve most if not all of the difficulties in computational stereo and reconstruction, and enable many exciting and useful applications that have yet to be seen.

Bibliography

Adelson, E. and Bergen, J. The plenoptic function and the elements of early vision. In Landy, M. and Movshon, J. A., editors, *Computational Models of Visual Processing*, chapter 1. MIT Press, Cambridge, Mass., 1991.

Aliaga, D. G. and Carlbom, I. Plenoptic stitching: a scalable method for reconstructing 3D interactive walkthroughs. In *Proc. 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*, pages 443–450, 2001.

Armstrong, M., Zisserman, A., and Beardsley, P. Euclidean reconstruction from uncalibrated images. In *Proc. British Machine Vision Conference (BMVC '94)*, pages 509–518, 1994.

Barnard, S. T. and Fischler, M. A. Computational stereo. *ACM Computing Surveys*, 14(4):553–572, 1982.

Baumberg, A. Reliable feature matching across widely separated views. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR '00)*, pages 774–781, 2000.

Brown, M. and Lowe, D. G. Recognising panoramas. In *Proc. International Conference on Computer Vision (ICCV '03)*, pages 1218–1225, 2003.

Brown, M. and Lowe, D. G. Unsupervised 3D object recognition and reconstruction in unordered datasets. In *Proc. Fifth International Conference on 3-D Digital Imaging and Modeling (3DIM 2005)*, pages 56–63, 2005.

Buehler, C., Bosse, M., McMillan, L., Gortler, S., and Cohen, M. Unstructured lumigraph rendering. In *Proc. 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*, pages 425–432, New York, NY, USA, 2001. ACM Press.

Carson, C., Thomas, M., Belongie, S., Hellerstein, J. M., and Malik, J. Blobworld: A system for region-based image indexing and retrieval. In *Proc. 3rd Int. Conference on Visual Information and Information Systems (VISUAL '99)*, pages 509–516, London, UK, 1999. Springer-Verlag.

Chang, C.-F., Bishop, G., and Lastra, A. LDI Tree: A hierarchical representation for image-based rendering. In *Proc. 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*, pages 291–298, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

Chen, S. E. Quicktime VR: an image-based approach to virtual environment navigation. In *Proc. 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*, pages 29–38, New York, NY, USA, 1995. ACM Press.

Chen, S. E. and Williams, L. View interpolation for image synthesis. In *Proc. 20th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '93)*, pages 279–288, New York, NY, USA, 1993. ACM Press.

Cheriton, D. and Tarjan, R. Finding minimum spanning trees. *SIAM Journal of Computing*, 5:724–742, 1976.

Chowdhury, G. G. *Introduction to Modern Information Retrieval, 2nd Ed.* Facet Publishing, London, 2004.

Cox, I. J., Hingorani, S. L., Rao, S. B., and Maggs, B. M. A maximum likelihood stereo algorithm. *Computer Vision and Image Understanding*, 63(3):542–567, 1996.

Debevec, P. E., Taylor, C. J., and Malik, J. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *Proc. 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pages 11–20, New York, NY, USA, 1996. ACM Press.

Dufournaud, Y., Schmid, C., and Horaud, R. Matching images with different resolutions. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR '00)*, pages 612–618, 2000.

Faugeras, O. *Three-Dimensional Computer Vision: A Geometric Viewpoint.* MIT Press, 1993.

Faugeras, O., Luong, Q., and Maybank, S. Camera self-calibration: theory and experiments. In *Proc. European Conference on Computer Vision (ECCV '92)*, pages 321–334. Springer-Verlag, 1992.

Ferrari, V., Tuytelaars, T., and Van Gool, L. Wide-baseline multiple-view correspondences. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR '03)*, pages 718–725, 2003.

Ferrari, V., Tuytelaars, T., and Van Gool, L. Simultaneous object recognition and segmentation by image exploration. In *Proc. European Conference on Computer Vision (ECCV '04)*, volume 1, pages 40–54, 2004a.

Ferrari, V., Tuytelaars, T., and Van Gool, L. Integrating multiple model views for object recognition. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR '04)*, pages 105–112, 2004b.

Fischler, M. A. and Bolles, R. C. RANdom SAmple Consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

Fitzgibbon, A. W. and Zisserman, A. Automatic camera recovery for closed or open image sequences. In *Proc. 5th European Conference on Computer Vision-Volume I (ECCV '98)*, pages 311–326, London, UK, 1998. Springer-Verlag.

Flickner, M., Sawhney, H., Niblack, W., Ashley, J., Huang, Q., Dom, B., Gorkani, M., Hafner, J., Lee, D., Petkovic, D., Steele, D., and Yanker, P. Query by image and video content: The QBIC system. *Computer*, 28(9):23–32, 1995.

Goedemé, T., Tuytelaars, T., and Van Gool, L. Fast wide baseline matching for visual navigation. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR '04)*, pages 24–29, 2004.

Gortler, S. J., Grzeszczuk, R., Szeliski, R., and Cohen, M. F. The lumigraph. In *Proc. 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pages 43–54, New York, NY, USA, 1996. ACM Press.

Gotz, D., Mayer-Patel, K., and Manocha, D. IRW: An incremental representation for image-based walkthroughs. In *ACM Multimedia '02*, 2002.

Greene, N. Environment mapping and other applications of world projections. *IEEE Computer Graphics and Applications*, 6(11):21–29, 1986.

Harris, C. and Stevens, M. A combined corner and edge detector. In *Proc. Fourth Alvey Vision Conference*, pages 147–151, 1988.

Hartley, R. and Zisserman, A. *Multiple View Geometry in Computer Vision, Second Edition*. Cambridge University Press, ISBN: 0521540518, 2004.

Hartley, R. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI '97)*, 19(6):580–593, 1997.

Horry, Y., Anjyo, K.-I., and Arai, K. Tour into the picture: using a spidery mesh interface to make animation from a single image. In *Proc. 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*, pages 225–232, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

Huang, T. and Faugeras, O. Some properties of the e matrix in two-view motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI '89)*, 11(12):1310–1312, 1989.

Isaksen, A., McMillan, L., and Gortler, S. J. Dynamically reparameterized light fields. In *Proc. 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*, pages 297–306, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

Kenneth E. Hoff, I., Keyser, J., Lin, M., Manocha, D., and Culver, T. Fast computation of generalized voronoi diagrams using graphics hardware. In

Proc. 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99), pages 277–286, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

Koch, R., Pollefeys, M., Heigl, B., Van Gool, L., and Niemann, H. Calibration of hand-held camera sequences for plenoptic modeling. In *Proc. International Conference on Computer Vision (ICCV '99)*, pages 585–591, 1999a.

Koch, R., Pollefeys, M., and Van Gool, L. Robust calibration and 3D geometric modeling from large collections of uncalibrated images. In *DAGM*, 1999b.

Laveau, S. and Faugeras, O. 3-D scene representation as a collection of images. In *Proc. International Conference on Pattern Recognition (ICPR '94)*, pages 689–691, 1994.

Levoy, M. and Hanrahan, P. Light field rendering. In *Proc. 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pages 31–42, New York, NY, USA, 1996. ACM Press.

Lhuillier, M. and Quan, L. Quasi-dense reconstruction from image sequence. In *Proc. 7th European Conference on Computer Vision-Part II (ECCV '02)*, pages 125–139, London, UK, 2002. Springer-Verlag.

Li, J., Tong, Y., Wang, Y., Shum, H.-Y., and Zhang, Y.-Q. Image-based walkthrough over the internet. In *International Workshop on Very Low Bitrate Video Coding (VLBV '01)*, 2001.

Lippman, A. Movie-maps: An application of the optical videodisc to computer graphics. In *Proc. 7th Annual Conference on Computer Graphics and Interactive*

Techniques (SIGGRAPH '80), pages 32–42, New York, NY, USA, 1980. ACM Press.

Longuet-Higgins, H. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(10):133–135, 1981.

Lourakis, M., Halkidis, S., and Orphanoudakis, S. Matching disparate views of planar surfaces using projective invariants. In *Proc. British Machine Vision Conference (BMVC '98)*, pages 94–104, 1998.

Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

Ma, Y., Soatto, S., Kosecka, J., and Sastry, S. S. *An Invitation to 3-D Vision*. Springer, 2004.

Mark, W. R., McMillan, L., and Bishop, G. Post-rendering 3D warping. In *Proc. 1997 Symposium on Interactive 3D Graphics*, pages 7–16, 1997.

Marzotto, R., Fusiello, A., and Murino, V. High resolution video mosaicing with global alignment. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR '04)*, pages I: 692–698, 2004.

Matas, J., Chum, O., Urban, M., and Pajdla, T. Robust wide baseline stereo from maximally stable extremal regions. In *Proc. British Machine Vision Conference (BMVC '02)*, pages 384–396, 2002.

McAllister, D. K., Nyland, L., Popescu, V., Lastra, A., and McCue, C. Real-time rendering of real world environments. In *Proc. Eurographics Workshop on*

Rendering Techniques, pages 145–160, 366, 1999.

McMillan, L. *An Image-Based Approach to Three-Dimensional Computer Graphics.* PhD thesis, University of North Carolina, April 1997. URL `http://www.cs.unc.edu/$\sim$ibr/pubs/mcmillan-diss/mcmillan-diss.pdf`.

McMillan, L. and Bishop, G. Plenoptic modeling: an image-based rendering system. In *Proc. 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*, pages 39–46, New York, NY, USA, 1995a. ACM Press.

McMillan, L. and Bishop, G. Head-tracked stereoscopic display using image warping. *Proceedings of SPIE*, 2409:21–30, 1995b.

Mikolajczyk, K. and Schmid, C. An affine invariant interest point detector. In *Proc. 7th European Conference on Computer Vision-Part I (ECCV '02)*, pages 128–142, London, UK, 2002. Springer-Verlag.

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

Mindru, F., Moons, T., and Van Gool, L. Recognizing color patterns irrespective of viewpoint and illumination. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR '99)*, pages 368–373, 1999.

Nistér, D. Frame decimation for structure and motion. In *Second European Workshop on 3D Structure from Multiple Images of Large-Scale Environments (SMILE*

'00), pages 17–34, London, UK, 2001. Springer-Verlag.

Nistér, D. Reconstruction from uncalibrated sequences with a hierarchy of trifocal tensors. In *Proc. 6th European Conference on Computer Vision-Part I (ECCV '00)*, pages 649–663, London, UK, 2000. Springer-Verlag.

Oliveira, M. M. Image-based modeling and rendering techniques: a survey. *RITA - Revista de Informtica Terica e Aplicada*, 9(2):37–66, 2002.

Pentland, A., Picard, R., and Sclaroff, S. Photobook: content-based manipulation of image databases. *Proc. SPIE*, 2185:34–47, 1994.

Pollefeys, M., Koch, R., and Van Gool, L. Self calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *Proc. 6th International Conference on Computer Vision (ICCV '98)*, pages 90–96, 1998.

Popescu, V., Lastra, A., Aliaga, D., and de Oliveira Neto, M. Efficient warping for architectural walkthroughs using layered depth images. In *Proceedings of the Conference on Visualization (VIS '98)*, pages 211–215, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

Pritchett, P. and Zisserman, A. Wide baseline stereo matching. In *Proc. 6th International Conference on Computer Vision (ICCV '98)*, pages 754–760, 1998.

Ronda, J. I., Valdés, A., and Jaureguizar, F. Camera autocalibration and horopter curves. *International Journal of Computer Vision*, 57(3):219–232, 2004.

Rui, Y., Huang, T. S., and Chang, S.-F. Image retrieval: past, present, and future. In *International Symposium on Multimedia Information Processing*, 1997.

Rusinkiewicz, S., Hall-Holt, O., and Levoy, M. Real-time 3D model acquisition. *ACM Transactions on Graphics (SIGGRAPH '02)*, 21(3):438–446, 2002.

Sainz, M., Susin, A., and Bagherzadch, N. Camera calibration of long image sequences with the presence of occlusions. In *Proc. IEEE International Conference on Image Processing (ICIP '03)*, pages I: 317–320, 2003.

Sand, P. and Teller, S. Video matching. Technical report, Massachusetts Institute of Technology, 2004. URL `http://graphics.csail.mit.edu/~sand/vid-match/tech-report.html`.

Sawhney, H. S., Hsu, S., and Kumar, R. Robust video mosaicing through topology inference and local to global alignment. In *Proc. 5th European Conference on Computer Vision-Volume II (ECCV '98)*, pages 103–119, 1998.

Schacter, B. Computer image generation for flight simulation. *IEEE Computer Graphics and Applications*, 1(4):29–68, 1981.

Schaffalitzky, F. and Zisserman, A. Viewpoint invariant texture matching and wide baseline stereo. In *Proc. International Conference on Computer Vision (ICCV '01)*, pages 636–643, 2001.

Schaffalitzky, F. and Zisserman, A. Multi-view matching for unordered image sets, or "How do I organize my holiday snaps?". In *Proc. 7th European Conference on Computer Vision-Part I (ECCV '02)*, pages 414–431, 2002.

Scharstein, D. and Szeliski, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, 2002.

Seitz, S. M. and Dyer, C. R. View morphing. In *Proc. 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pages 21–30, New York, NY, USA, 1996. ACM Press.

Shade, J., Lischinski, D., Salesin, D. H., DeRose, T., and Snyder, J. Hierarchical image caching for accelerated walkthroughs of complex environments. In *Proc. 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pages 75–82, New York, NY, USA, 1996. ACM Press.

Shade, J., Gortler, S., wei He, L., and Szeliski, R. Layered depth images. In *Proc. 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*, pages 231–242, New York, NY, USA, 1998. ACM Press.

Shum, H.-Y. and He, L.-W. Rendering with concentric mosaics. In *Proc. 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*, pages 299–306, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

Steedly, D., Essa, I., and Dellaert, F. Spectral partitioning for structure from motion. In *Proc. International Conference on Computer Vision (ICCV '03)*, pages 996–1003, 2003.

Sun, J., Li, Y., Kang, S. B., and Shum, H.-Y. Symmetric stereo matching for occlusion handling. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR '05)*, pages 399–406, Washington, DC, USA, 2005. IEEE Computer Society.

Tao, H. and Sawhney, H. S. Global matching criterion and color segmentation based stereo. In *Fifth IEEE Workshop on Applications of Computer Vision*

(WACV '00), pages 246–253, 2000.

Tao, H., Sawhney, H. S., and Kumar, R. A global matching framework for stereo computation. In *Proc. International Conference on Computer Vision (ICCV '01)*, volume I, pages 532–539, 2001.

Tappen, M. F. and Freeman, W. T. Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters. In *Proc. Ninth IEEE International Conference on Computer Vision (ICCV '03)*, pages 900–907, Washington, DC, USA, 2003. IEEE Computer Society.

Teller, S., Antone, M., Bodnar, Z., Bosse, M., Coorg, S., Jethwa, M., and Master, N. Calibrated, registered images of an extended urban area. *International Journal of Computer Vision*, 53(1):93–107, 2003.

Tomasi, C. and Kanade, T. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992.

Tomite, K., Yamazawa, K., and Yokoya, N. Arbitrary viewpoint rendering from multiple omnidirectional images for interactive walkthroughs. In *Proc. International Conference on Pattern Recognition (ICPR '02)*, pages 987–990, 2002.

Triggs, W. Auto-calibration and the absolute quadric. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR '97)*, pages 609–614, 1997.

Tuytelaars, T. and Van Gool, L. Wide baseline stereo based on local, affinely invariant regions. In *Proc. British Machine Vision Conference (BMVC '00)*, pages 412–422, 2000.

Uyttendaele, M., Criminisi, A., Kang, S. B., Winder, S., Szeliski, R., and Hartley, R. Image-based interactive exploration of real-world environments. *IEEE Computer Graphics and Applications*, 24(3):52–63, May/June 2004.

Werner, T., Hersch, R. D., and Hlaváč, V. Rendering real-world objects using view interpolation. In *Proc. International Conference on Computer Vision (ICCV '95)*, pages 957–962, June 1995.

Woo, M., Neider, J., and Davis, T. *OpenGL Programming Guide*. Addison Wesley, second edition, 1997.

Wood, D. N., Azuma, D. I., Aldinger, K., Curless, B., Duchamp, T., Salesin, D. H., and Stuetzle, W. Surface light fields for 3D photography. In *Proc. 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*, pages 287–296, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

Zhang, C. and Chen, T. A survey on image-based rendering - representation, sampling and compression. Technical Report AMP 03-03, Carnegie Mellon University, June 2003.

Zhang, Z. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI '00)*, 22(11):1330–1334, 2000.

Zitnick, C. L., Kang, S. B., Uyttendaele, M., Winder, S., and Szeliski, R. High-quality video view interpolation using a layered representation. *ACM Transactions on Graphics (SIGGRAPH '04)*, 23(3):600–608, 2004.