Image and Depth Coherent Surface Description

Pragyana K. Mishra

CMU-RI-TR-05-15

Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy*

March 2005

Thesis Committee:
Omead Amidi, *co-chair*
Takeo Kanade, *co-chair*
Martial Hebert
Peter Allen, Columbia University

Keywords: 3-D Modeling, triangulation, object representation, mesh fairing, image coherence, principal component analysis, 3-D textures, eigentexture, multiple view geometry, epipolar constraint.

To my parents

Acknowledgments

This thesis would not have been possible without the guidance and remarkable patience of my advisers–Dr. Omead Amidi and Professor Takeo Kanade.

It is difficult to overstate my gratitude to Takeo for his wisdom, insight, and drive. He taught me the value of perseverance, hard work, and in-depth research. He is truly the *O'Sensei*. I remain indebted to Omead for his profoundly optimistic attitude and confidence in me. His ideas, technical acumen, support, and energy were essential in allowing me to successfully complete this thesis. I cannot thank him enough.

I could not have wished for better advisers and teachers; if I were to do this all over again, I would do just the same.

I am deeply grateful to Professor Martial Hebert for his wise advice, support, and technical discussions.

I gratefully acknowledge the enthusiastic and thoughtful comments of Professor Peter Allen.

I thank Dr. Jonas August for his rigorous mathematical analysis and equally impassioned political discussions.

Thanks to Dr. Ryan Miller for his camaraderie and encouragement. I thank Todd Dudek and all others for making the Helicopter Laboratory a great place to work.

A very special thank you to Louise Ditmore and Suzanne Lyons Muth.

I wish to acknowledge the support and tremendous help of Professor Matt Mason and Professor Bob Colins.

I thank my classmates, Owen Carmichael and Alvaro Soto in particular, for their tremendous help and encouragement. Thanks also to Daniel Huber, Raju Patil, and Sanjiv Kumar for their friendship, criticism, and valuable feedback. I thank Nicolas Vandapel for helping me collect numerous data-sets.

And then there are all the other people who have made Pittsburgh a very sunny and special place over all those years: Sayam Sengupta, Hasnain Lakdawala, Arindam Chowdhury, Rajesh Shenoy, Dushyanth Narayanan, David Petrou, Sourav Ghosh, Bikram Baidya, Soshi Iba, Sanjeev Swarup, David LaRose, and Kelly Martin. I thank them all.

I remain indebted to to my brother, Pradyumna, for his constant love, support, words of encouragement, and his regular visits to Pittsburgh to cheer me up. I dedicate this thesis to my mother, Archana, and my father, Prasanta, for their unending patience, motivation, and love. Their many sacrifices and unflinching courage has been a great source of strength all through these years. I could not have asked for much more.

I cannot end without thanking my past teachers, Professor Prabir K. Biswas, Professor Ajoy K. Ray, Siddhartha Guha, and Alok Biswas, without whom I would not have accomplished much.

In the following pages I offer nothing more than simple facts, plain arguments, and common sense; and have no other preliminaries to settle with the reader, than that he will divest himself of prejudice and prepossession, and suffer his reason and his feelings to determine for themselves.

- Thomas Paine, *Common Sense*

Abstract

Description of a surface involves accurate modeling of its geometrical and textural properties. The choice of a surface description depends on both the observations we obtain from the scene and the level of modeling we seek. It can be a piecewise-linear approximation of surface geometry and 2-D texture or a dense point-based approximation of fine-scale geometry and albedo. In both cases, modeled properties of the unknown surface have to best explain observations of the surface such as images, depth samples, or appearance primitives. Re-projections of the 3-D model should be image coherent in that they best account for what is observed in images. Depth values of the modeled structure, as viewed from sensors, should be coherent with those of the real surface.

In many computer vision and graphics applications, a surface is described using a texture-mapped mesh of simplicial elements, such as triangles or tetrahedra. The mesh is a piecewise-linear approximation of surface geometry. Vertex points of the mesh are derived from observed 2-D image features or dense 3-D depth data. However, these vertices do not usually sample the surface at its critical points, such as corners and edges. This results in poor modeling of the surface, both geometrically and texturally. We show that the model can be refined or faired by relocating vertices of the mesh to the desired critical points. Relocating a vertex uses texture of the mesh faces common to the vertex. We introduce the idea of EigenFairing, an image-coherent mesh fairing algorithm that exploits the distance-from-feature-space of textured faces to incrementally refine a mesh to best approximate the modeled surface. EigenFairing couples geometrical properties of a 3-D model with its textural properties.

A piecewise-linear approximation of a surface does not suffice for modeling accuracy because most surface patches in natural scenes are curved or have fine-scale geometry, e.g., 3-D texture. Points on a surface, as observed in multiple images, are related to each other via the Epipolar constraint. Pixel intensities along epipolar lines for neighboring points that form a 3-D surface patch are used to generate a search space–an isodepth map–to which the surface patch belongs. The 3-D patch can be modeled by a slice of pixel intensities and depth values carved out of the isodepth map. We show that this slice is constrained by the texture of the patch as viewed in multiple images, a property we call the Epitexture constraint. Depth values carved out by the Epitexture constraint are coherent with depth samples of the surface as well. Depth samples of the surface can be either in the form of dense range data or derived from a set of images using stereo or structure-from-motion algorithms. The Epitexture constraint can be combined with texture and structure properties of natural surfaces. These properties regularize the slice by imposing a priori constraints. Regularization is essential if high-quality 3-D models are to be computed from 2-D images.

We show modeling results of several natural scenes. Results obtained for a variety of scenes, ranging from building facades to highly cluttered natural environments, demonstrate that the use of image and depth coherence yields high fidelity models.

Table of Contents

1 Introduction

The goal of Computer Vision is to derive a description of a three-dimensional scene in terms of its observed two-dimensional images. It focuses on finding a relationship between the three-dimensional (3-D) surface of the scene or object, and a set of two-dimensional (2-D) images of the same surface. The relationship has been elusive because of two reasons: First, the third dimension or depth of the scene is lost in the imaging process. Second, we are trying to understand and describe the 3-D surface using a combination of geometric and surface functions. Both the geometric and surface function data are derived from two-dimensional images. What makes the problem more interesting is that the two-dimensional image data we have are discretized as pixel intensities, both in the imaging plane and along the intensity scale. In addition to this, depth data, which is used for the geometric description, is a set of discrete point-samples of the surface.

Problems in Computer Vision have, so far, focused on solving parts or interesting facets of the relationship between the unknown 3-D surface and its observed 2-D images. While calibration, Stereo and Structure-from-Motion algorithms have solved for the 3-D geometrical structure of the observed surface or external parameters of the imaging system, problems such as reflectance modeling, segmentation, detection, grouping and object recognition have primarily focused on the part that involves pixel intensities of images and their relationship to a surface property.

The two research areas involve two different paradigms of conceiving the unknown 3-D textured surface of the object–paradigms that have been termed as model-based and view-based representations, or primitive-based and appearance-based approaches (Hebert et al., 1994). Researchers have also attempted to combine the two representations to solve various problems (Allezard et al., 2000) (Buker and Hartmann, 2000) (Chen and Sara, 2000). This is a more holistic approach of solving most vision problems as the two representations are in fact, inseparable from each other when it comes to describing the geometric and surface-function properties of a 3-D surface.

In biological vision, 3-D cues of scene description are learnt in early stages of development using a binocular or compound visual system. During latter stages of life, one can easily recognize 3-D objects and navigate in 3-D environments using monocular vision. Of course, one has access to the 3-D cues or descriptors learnt *a priori* by then (Edelman et al., 2002). Not to anthropomorphize computer vision, but this hints at a very important paradigm of vision; 3-D structure-based geometric attributes, and appearance cues that are parametric descriptors that are based on image intensities are coupled together to refine and *regularize* our description of a 3-D scene. How does one couple 3-D geometric attributes with 2-D image-based cues?

Yet another salient but so far unnoticed aspect of the relationship is that the 2-D images used to describe the surface are discretized both spatially in the imaging plane and functionally along the intensity scale as pixels. Moreover, a 3-D model or any description of the 3-D surface can only be done as discrete elements whether it be dense point clouds, polygonal meshes or spline patches, associated with textural and reflectance attributes. This leads to an interesting question; what is the best way of describing the surface of the 3-D object as discrete elements that, at the same time, best explains depth and image data? In simpler words, how do we represent the 3-D model in a manner that is coherent with observed surface structure and pixel intensities?

It might now seem that the first issue of the coupling of geometric attributes and textural cues is not related to the issue of what discrete elements the 3-D model is represented as. The question we would like to ask is: how do we represent the attributes and cues in the first place? If the 3-D model is a set of discrete elements, any cue or attribute will be a function of those discrete elements as well. Moreover, any coherence of the 3-D model with depth and image data can only be measured using these attributes or cues obtained from the generated 3-D model, and the given depth and image data–all of which are discrete elements.

Let us say we have found these attributes and cues from images, depth values and the 3-D model. In fact, there will be different sets of attributes or cues, each of which is a function of depth, pixel intensities and local shape parameters. These attributes can include principal components of image patches, textural or color statistics of local regions, surface orientation, edge statistics –just to name a few. Instead of attempting to solve a *bigger* problem of finding out the most compact and efficient set of attributes and cues for surface description from images, we concentrate on solving a small but significant part of the problem– finding the description, using a dense set of points or piecewise linear elements, that best accounts for observed data. Our goal, therefore, is to determine

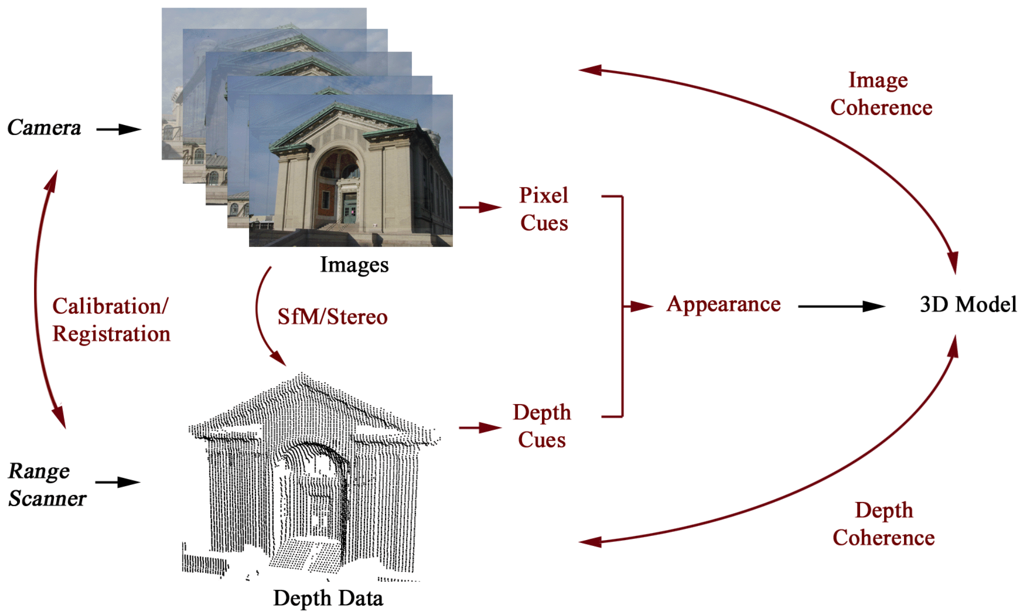


Fig. 1.1. High-resolution images from a camera or a set of cameras combined with depth data are used to generate descriptive 3-D models that best explain both pixel and depth data. Depth data can either be collected by a range-scanner or derived from images using SfM, stereo or any 3-D reconstruction algorithm. Pixel cues and depth cues derived from raw images or depth data collectively form appearance. One can explicitly derive a 3-D model of the scene or a view-dependent representation of the scene from appearance. The resulting 3-D model or any 3-D representation of the scene best accounts both for what is observed in images and geometrical structure as captured by the depth data. In other words, any 3-D description of the scene is coherent with its images and depth as viewed from the sensors.

a refined 3-D model that best describes the real surface, given depth and image data. How does one derive a coherence measure to compare the generated 3-D model with the observed data?

This thesis attempts to answer all these queries. It presents a unified approach for describing a 3-D surface by means of a 3-D model from depth values and images, that is most coherent with both data. The model, at the same time, is regularized using depth and texture priors to generate a high-quality description.

1.1 Context and Motivation for 3-D Surface Description

There has been a growing interest within robotics and humanoid-research, computer graphics and animation, architecture and design, manufacturing and rapid-prototyping communities in being able to represent and store the shape and appearance of physical objects. The 3-D model which captures various attributes of the object is an enriched description. The description is based on real and physical properties of the object that include geometry and shape, texture and color, reflectance, shading and any view-dependency. These properties that are primarily concerned with the external surfaces of objects collectively constitute *appearance*.

Representing and storing appearance of real physical objects has several important applications. Some important applications include product design, environment and event recording and simulation, reverse engineering, rapid prototyping, architectural archiving and preservation, virtualized flythroughs creation of models for movie making and Ethernet transmission. A far more significant contribution of representing appearance is being able to efficiently store and retrieve the way 3-D objects appear when viewed in two-dimensional images. Remembering events, recognizing familiar objects and scenes, recovering unknown environment structure and self-pose are important functionalities for designing a machine that can both perceive and conceive.

Considerable advances have been achieved in perception or sensing of the environment. Optical sensing technology, both passive and active, can register images and scene depth. Passive sensing such as cameras employ visible light to sense and pixelize light rays reflected by objects. Active sensing such as laser range scanners can register scene depth and shape of objects. A combination of both (Miller et al., 1999) can yield depth and color of point samples of the scene. However, the goal is to produce a seamless, enriched and geometric representation of the externally visible surface of the object.

Range scanners that digitize the shape of the visible surface of the object outputs a range image–a rectangular lattice of pixels, each of which contains the distance of a point or a surface patch, called the footprint of the scan, on the object's surface from the sensor. A camera, on the other hand, yields high-resolution imagery that registers the light reflected from the objects surface. Let us now assume that we have a hypothetical sensor, a combination of the range scanner and the camera, that yields high-resolution images with depth values associated with each and every pixel. This is precisely the problem that Stereo and SfM algorithms have tried to solve. Now, does the output of this sensor solve most problems of computer vision? Can we effectively register and recover the appearance of objects for the applications mentioned earlier?

The answers to the above questions lie somewhere deeper–at a level of conception beyond the perceived layers of pixels with depth or voxels with color. The answers lie in how appearance is defined and encoded. Surfaces of objects are manifolds; they are contour surfaces and water tight, continuous and having no holes or self-intersections.

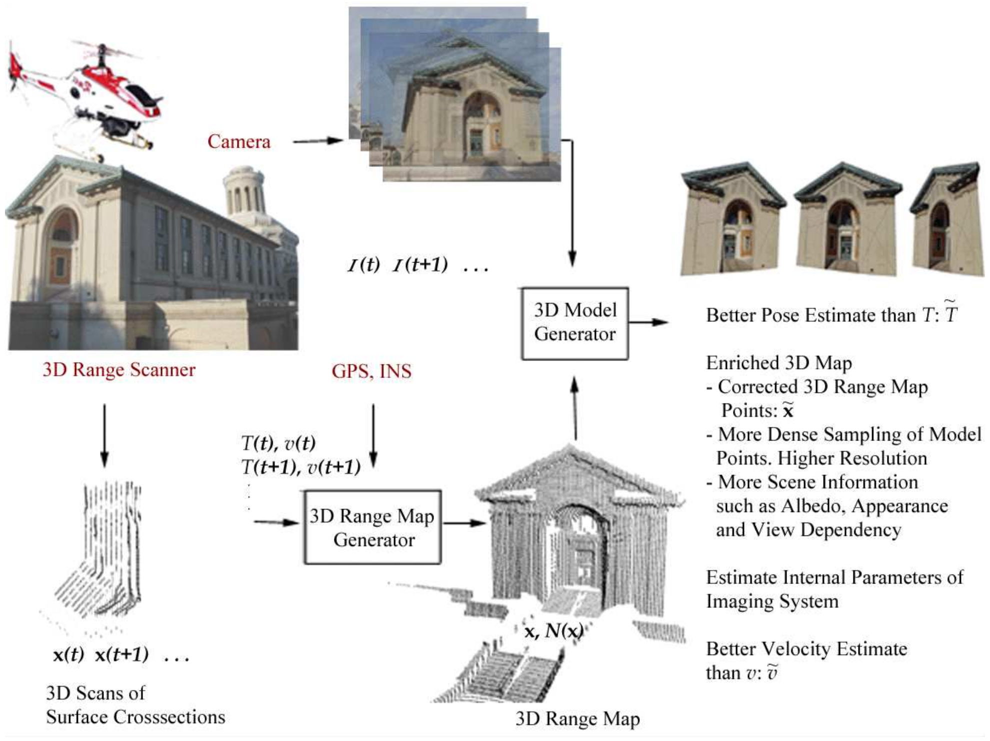

Camera
$I(t)$ $I(t+1)$ ...
3D Model Generator
Better Pose Estimate than T: $\tilde{T}$
Enriched 3D Map
- Corrected 3D Range Map Points: $\tilde{\mathbf{x}}$
- More Dense Sampling of Model Points. Higher Resolution
- More Scene Information such as Albedo, Appearance and View Dependency
Estimate Internal Parameters of Imaging System
Better Velocity Estimate than v: $\tilde{v}$
3D Range Scanner
GPS, INS
$T(t)$, $v(t)$
$T(t+1)$, $v(t+1)$
3D Range Map Generator
$\mathbf{x}$, $N(\mathbf{x})$
$\mathbf{x}(t)$ $\mathbf{x}(t+1)$ ...
3D Scans of Surface Crosssections
3D Range Map

Fig. 1.2. A 3-D range map consists of point samples of a 3-D surface. An *enriched* 3-D surface model is a discretized surface mesh that approximates the actual surface as close as possible and has albedo, appearance and view-dependency information. The 3-D *Model Generator* creates the surface model by combining images and the 3-D range map in a coherent manner.

Appearance of a surface patch is a function defined over a manifold in 3-D space. Researchers have, so far, dealt with discrete surface data or discretized version of the the manifold, primarily due to the sensor technology, and the manner of storing information as digitized elements. Of course, there is no way of getting around digitization or discretization. However, there has been no concerted effort to address the effect of discretization and determined its relationship to the actual smooth manifold while modeling appearance.

Once we have the discrete elements that approximate the actual surface and its properties, it is important to understand how well those elements collectively form a model that best accounts for observed data. The accuracy of the approximation depends on the sizes, shapes and attributes of the elements. A quality measure that compares the 3-D model against observed image and depth data can be used to refine the elements. In the process of refinement, the the approximation of the real surface is improved, both geometrically and texturally.

1.2 Problem of Surface Description

Let us explain the problem of surface description using an example. Figure 1.3 shows a 3-D surface of a building that we seek to model. In addition to albedo or color variations, a surface has two different types of physical geometry: structure as in shape and fine-scale corrugations. Consider the facade of the building; it has distinct corners that are at critical points of the surface as well as surface corrugations due to layers of bricks on the wall. The critical points of the surface, which are located at corners or edges and distinct ends of structure, are not always captured by range scanners due to their physical minuteness. Notice the corners and spiked ends of the cornice–the green decorative border–of the roof. Sometimes they do not appear in images due to occlusion or clutter such as the ones where the cornice intersects the gray wall. The other class of surface structure consists of fine-scale geometry or corrugations.

Three-dimensional surface corrugations on globally smooth surfaces give rise to brightness modulations of intensity patterns. Systematic variations of such 3-D image textures are a function of illumination and viewpoint. As pointed out by (Dana et al., 1999), there are essentially two different types of *texture*. 2-D image textures are due to flat pattern or albedo variations on smooth surfaces, such as painted designs on the inner dome of the facade and shadows. Such textures are used in “shape-from-texture” and “texture-mapping” techniques, which are results of the fact that the texture itself has no 3-D details but effects of albedo variations. However, majority of textures in images of natural scenes are due to the 3-D geometrical details of surfaces that can only be resolved visually in images. Such 3-D textures exist even if the surface has uniform albedo, such as the brick columns of the facade and the gable, which is the triangular area between the outer curve of the dome and the cornice.

2-D textures provide image features from which one can draw inferences about the structure or shape of the global surface. These 2-D texture features along with other image features that correspond to surface discontinuities are often used for reconstructing the 3-D geometry of the surface using Stereo or SfM. The surface discontinuities do not include fine-scale geometric details that give rise to 3-D textures. The 3-D fine-scale structure of the surface is, therefore, ignored during the reconstruction process. 3-D textures depend strongly of illumination and viewpoints and do not provide invari-

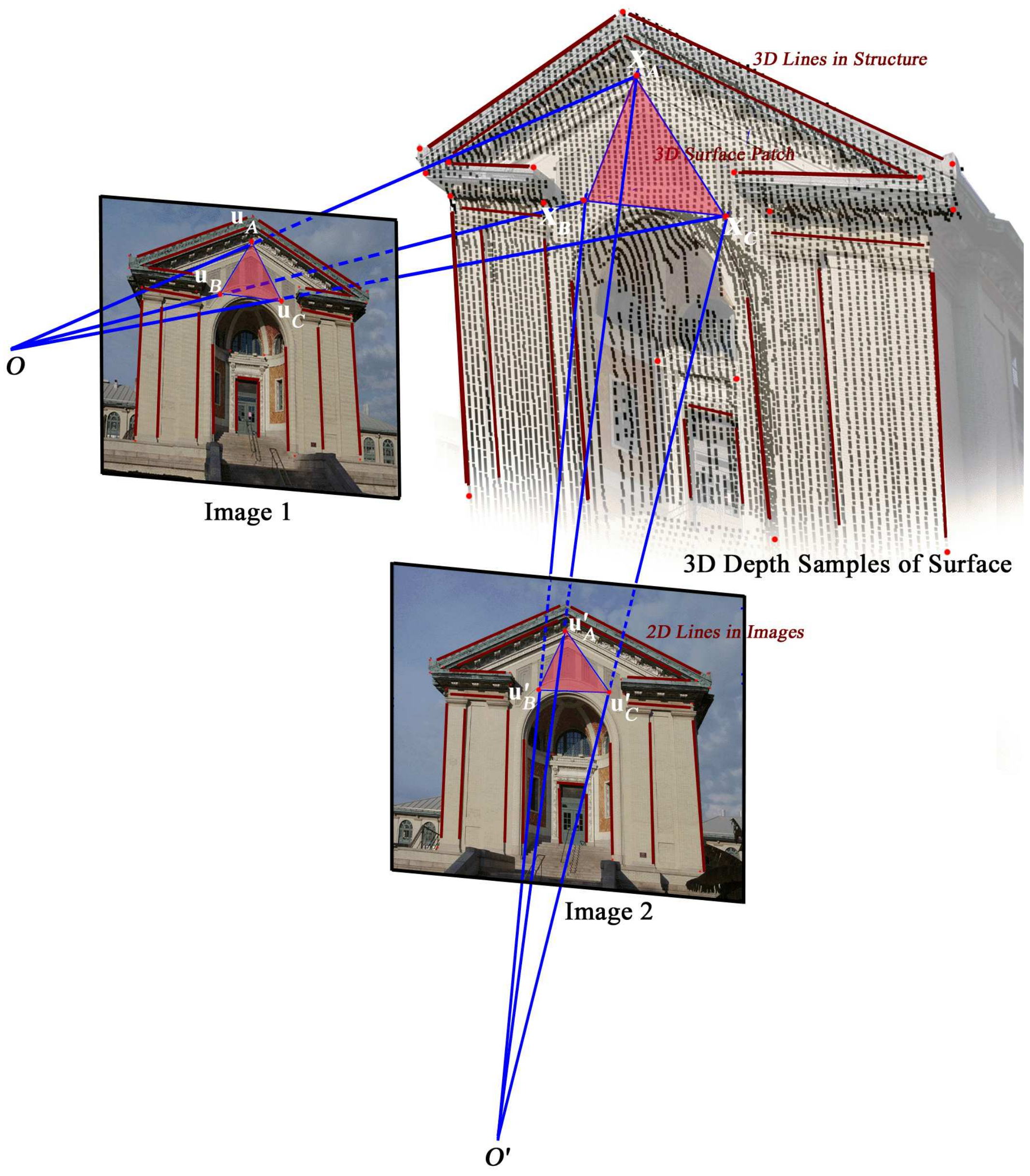


Fig. 1.3. The texture of the 3-D surface patch $X_A X_B X_C$ is projected into the images as the red triangular areas of pixels: $\mathbf{u}_A\mathbf{u}_B\mathbf{u}_C$ and $\mathbf{u}'_A\mathbf{u}'_B\mathbf{u}'_C$. Images and point samples of the 3-D structure or depth data are given. The sets of red points in the two images are related to each other via Epipolar geometry. However, the description of the surface requires more than just point samples of depth and 2-D images with feature points that satisfy Epipolar constraints.

ant features. They do not provide image features that can be easily detected and reliably tracked. On the other hand, they do provide information concerning local illumination and the nature of 3-D surface details and corrugations.

When a mesh is generated to approximate a surface, we triangulate 3-D points that correspond to features from 2-D textures and 3-D critical points as viewed in images. At that level it is not expedient or even practically feasible to go for the level of detail in modeling surface geometry due to 3-D textures. This is primarily because of the fact that any optimization process to model fine details is bound to get trapped in local extrema.

There exists many possible triangulations of a given set of 3-D points, each producing a different faceted surface. What makes the problem more interesting is that the 3-D points themselves may not be the best candidates for vertices of the triangulation. 2-D textures do not have any information of the critical points. 2-D image features do not always correspond to corners and edges. Dense range data miss these points as well. So, even if we figured out the best triangulation of 3-D points, we cannot guarantee that edges of the mesh are aligned along corresponding edges of the surface.

Let us now assume that we have the best possible piecewise-planar approximation of the surface. Consider the surface patch $X_A X_B X_C$ in Figure 1.3. It does not model the grooves and corrugations that appear on the gable. Moreover, the mesh face is a planar approximation of the surface patch, which could have been curved. If we now move to a point-based representation of the patch, we face a new set of problems. Treating a 3-D scene as a cloud of features, having no relationship to each other, can lead to contradictory results whereby incorrect features yield isolated features under the surface of an object. These can only be detected once the surface is itself recovered. Such surface contradictions are especially common at edges of occlusion, clutter, or other surface discontinuities.

There exists a geometry-based relationship between views and structure of the surface patch. Views of the surface patch, as in image areas $\mathbf{u}_A \mathbf{u}_B \mathbf{u}_C$ and $\mathbf{u}'_A \mathbf{u}'_B \mathbf{u}'_C$, register how the surface curvature and 3-D texture varies across the patch. Local geometry *modulates* local texture of the patch, and any projection of the textured patch into images registers this process.

The problem can be stated as: How does one combine surface curvature, texture, and the multiple-view relationship between images to model the surface? What guides the 3-D modeling process?

1.3 Problem Statement

The problem is formalized as follows.

Given information:

– Image data as discrete pixel elements and depth data as discrete point samples of the unknown 3-D surface. Depth data could be derived from images using Stereo or SfM. Image pixels in a small neighborhood are related to each other by ways of texture.
– A set of common textures and shapes that occur in natural scenes. This gives us prior knowledge of depth and texture in local areas of the scene that can be used to generate a high-quality descriptions.

The objective:

– To determine the refined 3-D model that best explains, or is most coherent to, observed image and depth data.

In order for the above objective to be achieved, it is important that we ask the following questions:

– How do we compare the 3-D model and the observed data? What is coherence?
– What is the best piecewise linear approximation–mesh–that can be generated from 2-D texture features and samples of depth data?
– How can the mesh be refined to model fine-scale geometry of the scene?
– What are the constraints on points that form a local surface patch?
– Can modeling of a surface patch be refined or regularized by prior knowledge of texture and structure?
– How does the surface patch account for observed image and depth data?

1.4 Contributions of the Thesis

In this thesis, we unify the use of 3-D meshes to integrate information from multiple images with that of using multiple cues. We view this specific framework in which multiple cues are integrated together to be an important contribution. The integration of multiple cues to describe a surface in form of a 3-D model that is most coherent with observed multiple images and depth data.

The significance of the research presented in this thesis is believed to be four-fold:

1. We will exploit coherence to model a 3-D surface from observed images and depth. The generated 3-D model best accounts for the observed image and depth data and closely approximates the real surface at the same time. Texture and geometrical attributes of a 3-D model are interdependent; one cannot achieve significant improvement in one attribute without improving the other.
2. We will show the importance of sampling a surface at its geometrical discontinuities such as edges and corners. Point samples that form vertices of a mesh–a piecewise-linear approximation of a surface–can be relocated to these discontinuities by means of texture.
3. A mesh based on prominent features in 2-D image textures and 3-D depth data can be further refined to account for 3-D textures–variations in image intensities due to macroscopic structural details. The mesh is transformed into a point-based model to accommodate fine-scale geometry.
4. Points in a local patch of a surface satisfy the multiple-view geometrical constraint–the epipolar constraint–between images. This constraint can be combined with texture and depth priors to generate a high quality model, which is coherent with observed data at the same time.

1.5 Thesis Outline

The focus of this thesis is the study of a technique, called *image and depth coherence*, for modeling surfaces by using textured meshes. The idea of surface modeling using geometrical, textural and reflectance properties has been discussed in Chapter 2. The chapter briefly surveys some of the previous research of describing surfaces, either by model-based or image-based approaches. The two approaches aim at efficient organization of data to describe a surface or object. They create a formal description of reality by encoding information either in an object-centric or viewer-centric way. The surface descriptors that are often encoded via these representations–surface structure, texture and appearance–are discussed.

Chapter 2 also explains image-based refinement techniques that have been used to improve the quality of 3-D models. In particular, it discuses various techniques in Computer Graphics such as image-driven mesh simplification and silhouette preservation. Other algorithms based on image consistency and view dependency that aim at refining textured meshes using multiple images have been discussed.

The third chapter focuses on the data that are used to describe surfaces–images and depth. It brings out the concept of cues in relation to these data sets and a method of combining the multi-modal cues. It presents the Crofton's theorem in Integral geometry and how the theorem can be used to combine various cues to form appearance. This forms the first part of the *image and depth-coherent* approach. Appearance, as defined here, is a compact description of a local patch of surface that can be used for deriving a coherence measure between the actual surface and its model.

Coherence compares the appearance of a surface with that of the model. The actual surface is not known and, therefore, any coherence measure can only compare the model description with the observed data. One such coherence measure based on the appearance and its representation in eigenspace. Minimizing the distance between the observed appearance and its reconstruction from eigenspace, also known as the distance from feature space (DFFS), leads to a better approximate of the surface by the model.

Chapter 4 introduces depth coherence whereby the coherence with respect to depth data is used to model surface geometry. Depth coherence is an extension of point and line-based epipolar geometry to derive a 3-D surface patch-based constraint–the Epitexture constraint. The Epitexture constraint uses more than just depth samples; it exploits the textural property of a surface patch and can model patches that have high textural detail. A salient point of depth coherence is that it is more than a constraint on a set of point features across images in a sequence. It is based on pixel intensities and corresponding depth values.

The Epitexture constraint allows us to combine the relationship between epipolar lines for neighboring points, which form a 3-D surface patch, with texture and structure properties by way of priors. We start from a textured mesh face, which is a planar approximation of the surface patch, and use epipolar geometry to build a search space to which the patch belongs. Texture and structure priors are then used to carve out the 3-D surface patch. The resulting surface patch, at the same time, best explains what is observed in images. We show results for a few natural scenes using this approach.

Besides depth coherence, there exists coherence between the model and observed images. Re-projection of the 3-D model into the images is used to enforce this coherence. In case of a triangulated mesh, textures are associated with faces of the mesh to model a surface. However, faces of the mesh are mere planar approximations of the surface patch they model. Point-based epipolar constraint applies to 3-D points or vertices of the mesh and their re-projections into the image plane. This does not guarantee that piecewise linear faces of the mesh formed by the vertices are good approximations of the surface patch. When the vertices do not lie at critical points of maximum curvature or discontinuities of the real surface, faces of the mesh do not lie close to the modeled surface. This results in textural artifacts, and the model is not perfectly coherent with a set of actual images–the ones that are used to texture-map its mesh.

Chapter 5 explains a technique for perfecting the 3-D surface model by repositioning its vertices so that it is coherent with a set of observed images of the object. The textural artifacts and incoherence with images are due to the non-planarity of a surface patch being approximated by a planar face, as observed from multiple viewpoints. Image areas from the viewpoints are used to represent texture for the patch in eigenspace. The eigenspace representation captures variations of texture, which we seek to minimize. A coherence measure based on the difference between the face textures reconstructed from eigenspace and the actual images is used to reposition the vertices so that the model is improved or faired. This technique of model refinement is termed EigenFairing, by which the model is faired, both geometrically and texturally, to better approximate the real surface.

Results for real data have been detailed in Chapter 6. Image and depth coherence have been used to model real scenes using data collected from different sensors. The Epitexture constraint as well as EigenFairing have been used to create and refine 3-D models of large architectural structures.

The main shortcoming of our three-dimensional refinement approach is that texture and geometric features pose a restriction on the extent to which a surface model can be refined. The main concern is about the effect of quantization error in sensor data. Although most numerical algorithms are affected by quantization errors, there are fundamental reasons why geometry and texture refinement of surface descriptions are particularly susceptible. One of the fundamental reasons lies in the very nature of imaging process where the spatial quantization of pixel coordinates and intensity quantization couple to pose a limit on the degree of refinement. In most numerical algorithms, errors due to quantization or roundoff can be minimized if not eliminated by careful numerical analysis. However, this cannot be done in case of pixel-quantization as any knowledge of the surface is absent and therefore, any relation between the surface geometry and texture as they appear on an image plane cannot be determined. As a result of this, errors due to pixel-coordinate and intensity quantization that depend on surface geometry and texture are difficult to analyze. This has been discussed in *Appendix A*.

1.6 Notation Standard

We now introduce the notation used throughout this thesis. We seek to standardize this notation for future work as well.

Symbol	Definition
$\mathbf{u}$	Pixel coordinate; $\mathbf{u} = [u\ v]^T$
$\mathbf{x}$	3-D point coordinate; $\mathbf{x} = [X\ Y\ Z]^T$
f	Focal length
O	Camera center
λ	Depth of a point with respect to a camera; $\lambda = Z/f$, $\lambda = Z$ for $f = 1$
T	World-camera transformation (3×4); $\lambda[\mathbf{u}^T\ 1]^T = T\mathbf{x}$
L	Line in 3-D space
l	Epipolar line
I	Image
$\mathbb{I}$	Set of images; $\mathbb{I} = \{I_1, I_2, ..., I_n\}$
$\boldsymbol{u}$	Pixel coordinate in a fixed area that an image area is warped to
$\mathcal{F}$	Mesh face
$\mathcal{V}$	Mesh vertex
$\mathcal{E}$	Mesh edge
$\mathcal{T}$	Surface texture or reflectance
$\mathit{\Gamma}$	image or depth descriptor
D_{ij}	Isodepth map of image i in image j
e_p	Pixel re-projection error
$\mathbb{T}$	Learnt textures or texture data-base
ρ	Robust error norm
$\mathbf{H}$	Hessian
$\mathbf{g}$	Gradient

2 Related Work

This chapter presents work related to our image and depth coherent surface description framework. At first, we discuss two most prevalent paradigms of surface representation: image-based and model-based. This leads to a brief discussion of our descriptors that use surface structure, texture, and appearance. We categorize prior surface representation research with respect to surface descriptors. We then present existing research on how the descriptors can be refined once they have been determined. These refinement methods typically exploit consistency of the descriptor with respect to observed image data for different viewpoints. While a structure descriptor is 3-D in nature and the texture descriptor is primarily 2-D, an appearance descriptor lies somewhere in between the structure description and the texture representation. The last section presents existing strategies by which 3-D range data have been combined with 2-D images to generate descriptive surface models.

2.1 Surface Representation Methods

Finding a description of a surface in terms of its structure and reflectance properties from multiple images has received meticulous attention (Barrow and Tenenbaum, 1978; Marr, 1982; Terzopoulos, 1988; Okutomi and Kanade, 1993; Grimson and Huttenlocher, 1992). Barrow and Tenenbaum's *Intrinsic Images* (Barrow and Tenenbaum, 1978) and Marr's $2\frac{1}{2} - D$ *Sketch* (Marr, 1982) are $2\frac{1}{2} - D$ view-centered surface representations. Marr's $2\frac{1}{2} - D$ *Sketch* is based on rough distances to surface patches and their orientations. (Koenderink and van Doorn, 1979) used differential geometry to relate their ideas to Gestalt theories of perception.

Many have used single sources of information, such as sequences of range data or intensity images (Asada et al., 1992; Hung et al., 1991), stereo (Diehl and Heipke, 1992; Kaiser et al., 1992; Witkin et al., 1987), and shading (Hartt and Carlotto, 1989; Horn, 1990; Terzopoulos et al., 1987). Some have combined different sources of information to describe surfaces, such as shading and texture (Choe and Kashyap, 1991), focus, vergence angle, stereo, and calibration (Abbott and Ahuja, 1993; Fine and Jacobs, 1999), texture, motion, and depth (Jacobs, 1999), shading, stereo and edge information (Bulthoff and Mallot, 1987). shading and stereo (Leclerc and Bobick, 1991; Cryer et al., 1993).

Full 3-D representations have been extensively used to describe surfaces. 3-D surface meshes (Vasilescu and Terzopoulos, 1992; Vemuri and Malladi, 1991), parameterized surfaces (Stockely and Wu, 1992; Lowe, 1991), local surfaces (Ferrie et al., 1990; Fua and Sander, 1992), particle systems (Szeliski et al., 1993), volumetric models (Pentland, 1990; Terzopoulos and Metaxas, 1991; Pentland and Sclaroff, 1991).

For 3-D representations, a variety of single image cues have been used for their reconstruction. Silhouettes (Liedtke et al., 1991), image features (Cohen et al., 1992; Tomasi and Kanade, 1992; Wang and Wang, 1990; Terzopoulos, 1986), range data (Whaite and Ferrie, 1991), stereo (Fua and Sander, 1992), and motion (Szeliski, 1991). Using full 3-D representations, (Heipke, 1992) integrates stereo from areas of varying albedo and shading from areas of constant albedo.

2.2 Surface Descriptors

Representing surfaces, either by an image-based or model-based representations, necessitates certain primitives that describe surfaces or surface descriptors. While a model-based representation is, for the most part, based on shape-based primitives or an explicit three-dimensional description of the surface. However, there exists no particular primitive that can fully capture the shape of every object. The assumption that is often violated is that there exist primitives that support a stable decomposition of shape or structure of a surface. In contrast, image-based approaches do use a set of images to derive an implicit description of a surface. Aspect graphs that focus on image geometry, photometric, color, or texture information are a few descriptors often used by image-based techniques. However, these approaches depend on views of the images and incorporate certain degree of structure information. In our opinion, the only approach that is purely image-based is the one on image-based rendering using image priors, proposed by (Fitzgibbon et al., 2003). The following sections present a brief survey of related work, categorized by descriptors that we have focused on.

2.2.1 Structure

Estimating 3-D surface structure of a rigid surface from 2-D images is one of the most popular problems in computer vision. Structure-from-motion and stereo algorithms have used multiple view geometry to derive point structure from a set of corresponding features in an images sequence (Hartley and Zisserman, 2000; Faugeras et al., 2001). Salient features—usually points at intensity discontinuities or high curvature points as elucidated by lighting—are detected and tracked across images. It is from these features that the unknown structure as well as the camera parameters (external and if needed internal) are estimated. There is certain interaction between the multiple-view geometry estimation and the feature tracking; the multiple view constraints can be used to regularize feature tracking.

If the extracted features are image points, as is often the case, then the estimated 3-D structure is a 3-D point cloud. In other words, structure is reduced to points whereupon an estimate of camera parameters and 3-D point structure can be achieved. However, the camera parameters and the 3-D points structure can then be used to estimate a more detailed or full structure. All in all, SfM and stereo algoritms utilize layers of approximated models or problems, even more so, since solving for structure with points also uses hierarchies of models, for example factorization methods. Factorization algorithms (Kanade and Morris, 1998; Sturm and Triggs, 1996; Tomasi and Kanade, 1992) enable a the SfM problem to be solved without an initial guess needed, as in the non-linear bundle adjustment approaches (Slama, 1980; Triggs et al., 2000). Together with the eight-point methods for estimating the fundamental matrix (Hartley, 1997; Longuet-Higgins, 1981) the factorization methods are sometimes referred to as direct methods. Another limitation surmounted is the ability to perform 3-D reconstruction without known internal camera parameters, i.e. auto-calibration (Faugeras et al., 1992; Hartley et al., 1999; Heyden and Astrom, 1997; Pollefeys et al., 1999). However, relaxing the constrains also increases the number of ambiguities and degenerate configurations. Ambiguities within structure from motion have also been identified and studied (Buchanan, 1988; Carlsson and Weinshall, 1998; Sturm, 1997).

As for point-based 3-D reconstruction process, the main problem faced is that of the unreliability of the feature tracking and correspondence. So much effort has gone into developing robust statistical methods for estimating multiple view relations, such that outliers of the feature tracking could be identified and dealt with. To mention a few, there s the issue of robustly estimating the epipolar geometry (Torr et al.,

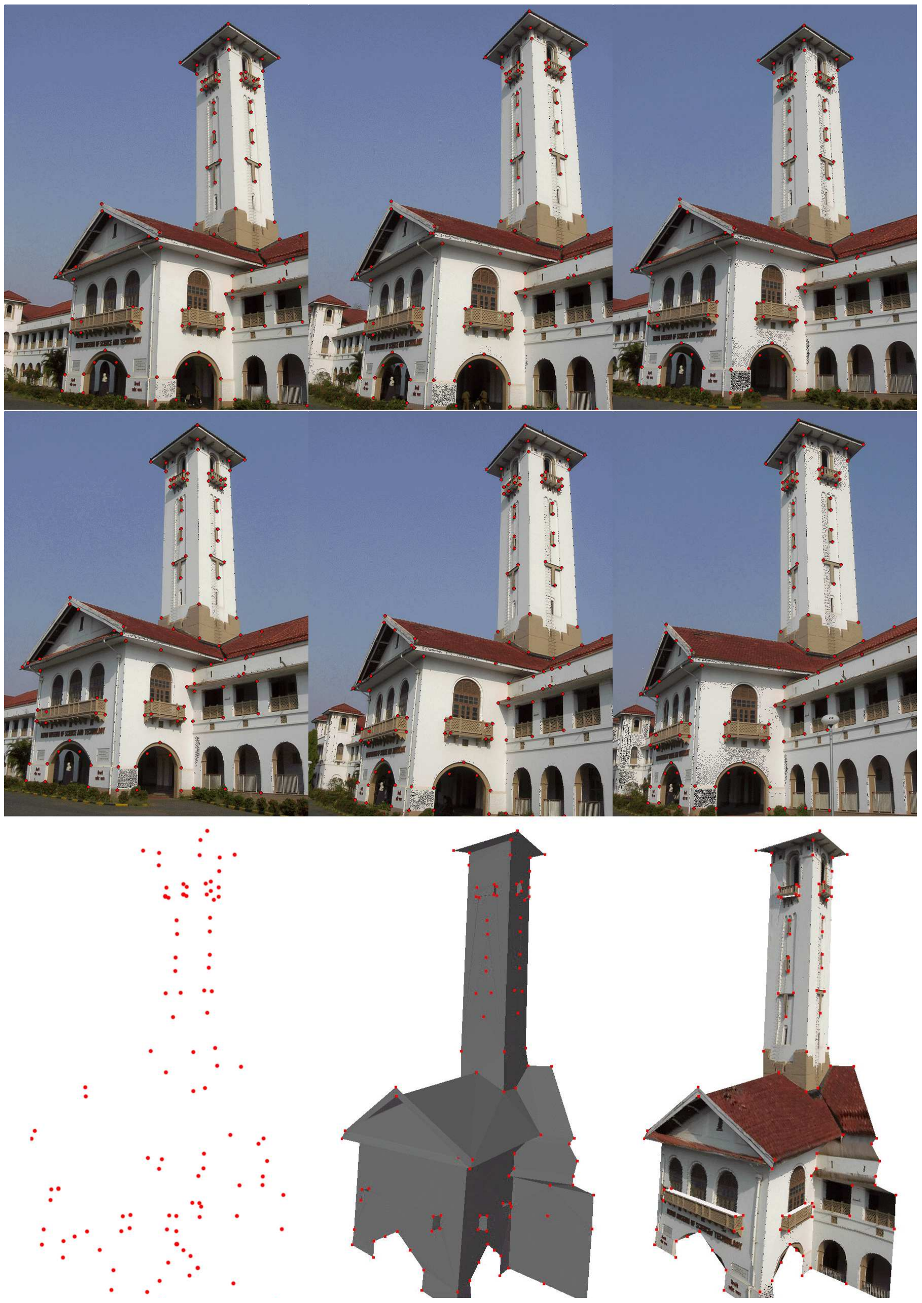


Fig. 2.1. Image features are used to reconstruct the 3-D points that are samples of the surface. The convergent bilinear algorithm proposed by Mahamud et al. (2001) is used for the point-based reconstruction. The data set consists of 102 feature points in 18 images (six of which are shown above), each image being 800 × 600 pixels in size. At convergence, the final average re-projection error was approximately 0.6 pixel.

1998; Torr and Murray, 1993), which sparked the popularity of the RANSAC algorithm (Fischler and Bolles, 1981) and incited the development of extending the idea to three and four view geometry (Faugeras, 1995; Hartley, 1995; Shashua and Werman, 1995; Triggs, 1995).

2.2.2 Texture

Texture adds a new quality to 3-D structure making it more similar to the real surface and more adequate for perception and understanding. It makes a more enriched description of the surface. A general photo-realistic description includes view-dependency. It consists of two distinct components: texture which is view-independent and (specular) reflectance which depends on illumination direction and viewpoint (Kurazume et al., 2002). In our framework, we will focus on texture (sometimes referred to as the diffuse reflection component) for photo-realistic description of a surface. In doing so, we will bring out the inter-dependence of texture and structure descriptors. Accurate and photo-realistic texturing enables geometrical analysis of structure in regions that are not spatially informative. Structural detail of such regions might not be captured well from given viewpoints.

(Pentland et al., 1994) describe use of view-based eigenface models to represent a variety of viewpoints. The Eigentexture method (Nishino et al., 2001) encodes appearance variations on the same physical patch of an object under various viewing conditions. It compresses appearances of a patch by using the eigenspace method and renders new views of a patch from its eigenspace representation. View-based representation in eigenspace has also been used in tracking rigid and articulated objects (Black and Jepson, 1998). It has been shown that for diffuse surfaces of arbitrary texture, the first five components of eigenspace explain most of the image variation (Epstein et al., 1995). In our research, we will use an eigenspace representation of dimensionality five to refine the mesh that approximates a 3-D surface.

2.2.3 Appearance

Principal component analysis has not only been used to parameterize texture, but also shape and appearance. Two popular models for the purpose of shape and appearance representation and modeling are Active Shape Models (ASM) (Cootes et al., 1995; Hill et al., 1996) and Active Appearance Models (AAM) (Cootes et al., 2001; Edwards et al., 1999). In ASM, the local appearance model, represents local statistics

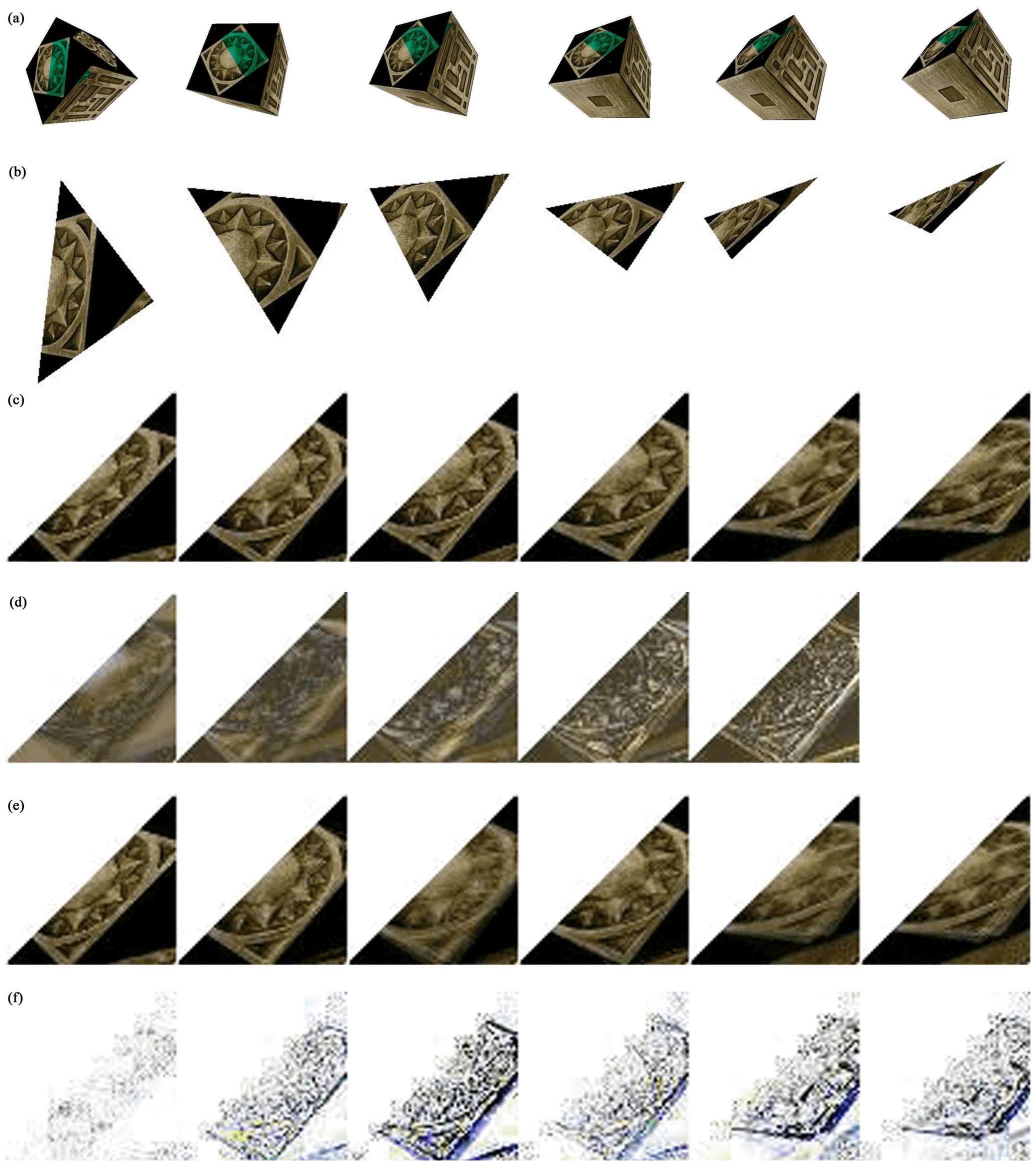


Fig. 2.2. (a) A surface patch, highlighted in green, is viewed at different angles in a set of images. (b) Image areas corresponding to the same surface patch are warped into a fixed image area as shown in (c). Any variation in these image areas can be attributed to either reflectance or any other view-dependent surface descriptor. (d) The first five components of the eigenspace decomposition of these image areas captures the minute variations as discussed in the eigentexture method by Nishino et al. (2001). (e) Reconstruction of the same image areas from the first five components. (f) The error between the original image areas and the reconstructed ones. Minimizing this error can lead to better approximation of surface descriptors such as texture or reflectance.

around selected features or landmarks and finds the best candidate point for each landmark in the image. The solution space is confined by a properly trained global shape model. For an accurate modeling of local feature, ASM gives accurate results in shape localization. AAM, on the other hand, combines constraints on both shape and texture in representing appearance of the surface. In the optimization two linear relationships are assumed: the linear mapping from appearance variation to texture variation, and the linear mapping of texture variation to position variation. The optimization is guided by minimizing the texture reconstruction error. ASM has better accuracy in shape localization and AAM has lower texture representation error. Moreover, they have individual limitations; ASM uses only local information, which makes the solutions often locally optimal and largely dependent on how good the initialization is, and AAM, the simplified linear relationships make the solutions much affected by the illumination. In addition, training an AAM model is complex and time consuming. While ASM is less sensitive to illumination variation because its local appearance is based on local features or landmarks, AAM is very sensitive to perturbations in illumination. We would like to combine the strong attributes of both methods: the flexibility of not having to initialize the shape description accurately and the texture-driven robustness to illumination variation. Our framework is similar in idea to the texture-constrained shape description proposed by (Cootes and Taylor, 2001).

2.3 Surface Descriptor Refinement

Once a surface descriptor is estimated, it is often the case that it does not precisely explain what is being observed in images or depth data. Many vision algorithms have focused on this issue; they improve the quality of the surface description by comparing it to observed data. In many cases, surface structure derived from triangulation of point-data leaves a lot of scope for improvement as the triangulated mesh can be refined to better approximate the surface. Researchers in the graphics community have approached this problem from a mesh generation and simplification standpoint. While some have compared rendered images of 3-D models with original images, others have exploited silhouettes and specular highlights of models to simplify the models. Another class of algorithms, primarily in computer vision, have used images as well as surface geometric constraints such as smoothness to refine 3-D models.

2.3.1 Structure Refinement: Image-driven Mesh Simplification

The problem of mesh refinement given a set of 3-D point samples of an object is similar to the problem of mesh simplification. In computer graphics, the problem of mesh simplification focuses on simplifying the geometric representation of a scene to reduce its rendering cost, while preserving visual fidelity. However, a recent paper (Williams et al., 2003) argues that imperceptible simplification is flawed, and that if one cannot afford to render the detailed original model, one can't afford to render an indistinguishable approximation. Having said this, many simplification algorithms proposed by the computer graphics community are guided by image properties, and therefore close in spirit to our work.

3-D surface meshes that constitute models have been simplified using purely geometric properties. Positions of vertices, edges, and faces are used by many researchers to decide how close a model is to the original. (Garland and Heckbert, 1998) refine the model by repositioning vertices using surface properties such as color, surface normals, and texture coordinates. Their mesh simplification algorithm never accesses the pixels of the single-image texture and merely updates the texture coordinates in the image. Some others use (Erikson and Manocha, 1999) use point clouds in color space or texture coordinate space to simplify models using vertex merge. These methods have focused on geometric fidelity while preserving appearance related surface properties during simplification.

2.3.2 Structure Refinement: View-dependent Silhouette Preservation

Another class of algorithms have included view-dependent heuristics to drive mesh simplification. For example, view-dependent silhouette preservation (Luebke and Erikson, 1997), a vertex clustering simplification algorithm, enforces a tighter screen-space threshold for silhouette regions than interior regions. Presence and movement of specular highlights across a surface provide important clues to its shape, and therefore, view-dependent simplification algorithms (Xia and Varshney, 1996; Klein and Schilling, 1999) preserve detail where highlights are likely to appear.

Image-driven mesh simplification (Lindstrom and Turk, 2000) uses multiple images to decide which portions of a model to simplify through the edge collapse operator. Edges are collapsed and replaced by vertices to make incremental changes to a model. It compares images of an original model against those of a simplified model using the root-mean-squared difference of luminance channels to determine the cost of an edge

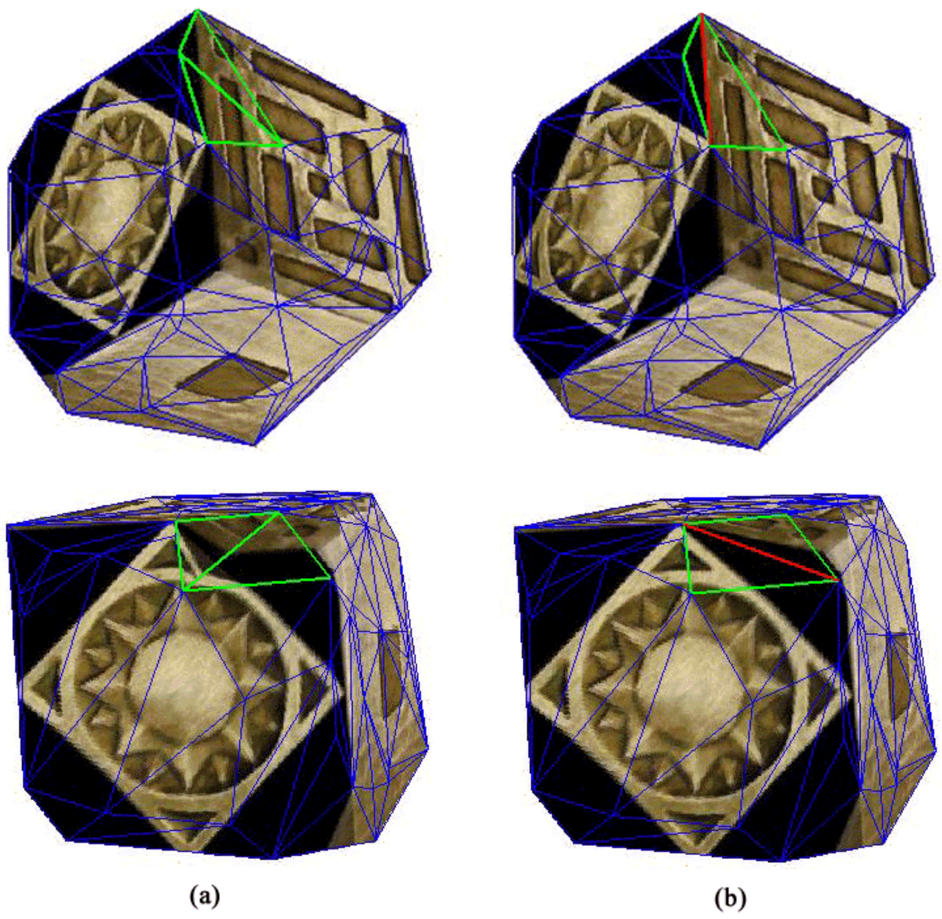


Fig. 2.3. Image-consistent surface triangulation proposed by Morris and Kanade (2000) uses edge-swaps to refine a mesh that approximates surface structure. Edge-swapping, at the same time, removes textural artifacts and thereby leads to better approximation of surface texture. The triangulation in (a) shows two adjacent triangles with green edges. The common edge, however, does not lie close to the real edge of the surface and this leads to a poor approximation of the surface by faces of the mesh. Swapping the edge generates a better surface triangulation and removes the existing textural artifacts. In (b), the swapped edge is shown in red.

collapse. The approach has the important benefit that simplification is ultimately guided not by geometric error, nor by some combination of geometric and shading attribute error, but by an estimation of what effect the simplification will have on the rendering from other view-points.

2.3.3 Structure and Texture Refinement: Image Consistency

A similar multi-view approach, image-consistent surface triangulation (Morris and Kanade, 2000), searches for the best triangulation among many possible candidates that best explains observed images. The initial mesh of a point-cloud is refined through edge-swaps to best account for images of the same object. The surface is modeled by a texture-mapped mesh and reprojected onto the images to yield the re-projection error. This error is minimized over swaps of edges of the mesh. The mapping of 2-D texture onto the surface is not isometric, which means that the associated affine transformation is not the same for all faces of the mesh (Schilling and Klein, 1998). In absence of the knowledge of the true surface, the mesh faces are texture-mapped by affine warping of triangles instead of perspectively projecting the images onto the surface. This simplification works only if the surface patch has enough texture and, at the same time, is close to the planar mesh face that approximates it. In areas of little or no texture, swapping edges often produces complicated overlapping mesh faces.

Other algorithms Fua and Leclerc (1996) Nobuhara and Matsuyama (2003) recover surface shape and reflectance properties from multiple images by deforming a 3-D representation. However, these methods optimize complicated objective functions that combine several image and geometric-based constraints.

2.4 Combining Structure from Range Data and Images

Previous approaches of combining range data that captures structure and image data have focused on recovering the transformation between an image and the range model coordinates by finding a set of correspondence matched between features, mostly linear edges in the two coordinate frames. Images have textural information as well as structure while range data captures surface geometry or structure. However, there has been no work that directly combines structure from the two sources of information.

(Allen and Stamos, 2000) have segmented range scans into *planar* regions to determine 3-D linear features. These are matched to linear features found in an image by a *user* and registration between range and image data is done by the algorithm proposed by (Kumar and Hanson, 1994). In subsequent research, (Allen and Stamos, 2001) have included clustering of these linear features into sets of parallel lines or rectangles to register images with the model. At first, they have extracted the vanishing point in an

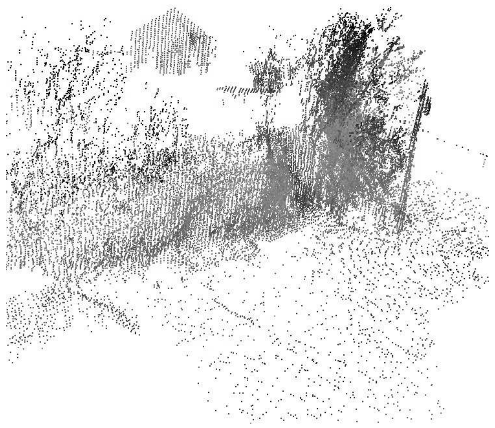


Fig. 2.4. Noise is caused by quantization of a 3-D point-coordinate as well as inaccuracy in spatial distribution of the mesh density (topology).

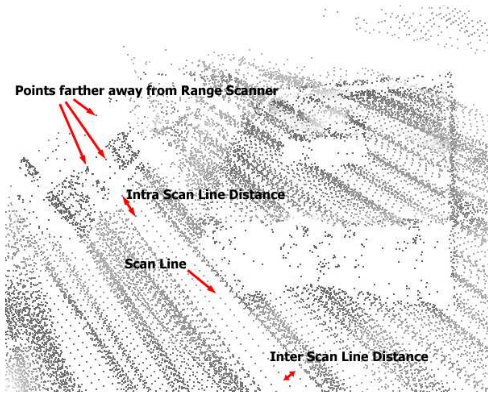


Fig. 2.5. Anisotropic range data where the inter and intra-scanline distances and associated errors vary over the map.

image to determine parallel lines and then used it to undo the perspective effect and map quadrangles to rectangles in a model.

(Baltzakis et al., 2002) have inferred 3-D structure information based on both visual and range data. The range measurements are grouped into line segments that are used to determine the motion of the robot. The robot motion is in turn used to find the 3-D structure from images using epipolar geometry. The camera is precisely calibrated with the laser range scanner beforehand.

(Zhao and Shibasaki, 2001, 2000) have used a similar setup where the CCD camera and the range scanner are calibrated with respect to each other. The rotating axis of the scanner is set vertical to the ground surface and range points are projected on to a horizontal plane (Z-image). Line segments are extracted from a set of Z-images to determine the transformation between a set of range points. The approach fails in presence of non-vertical planar features or lack of distinct linear features for registration; e.g. buildings partially hidden by large trees.

(Kurazume et al., 2002, 2001) have used reflectance images to register 3-D range maps with color images. Reflectance images are aligned with range maps as they are obtained by the same optical device. The reflectance images are similar to intensity images as they capture surface roughness and have similar edges. The edges are aligned using the M-estimator.

The approaches mentioned so far have something in common: extract features in range and image data and determine correspondence between the two sets of features to

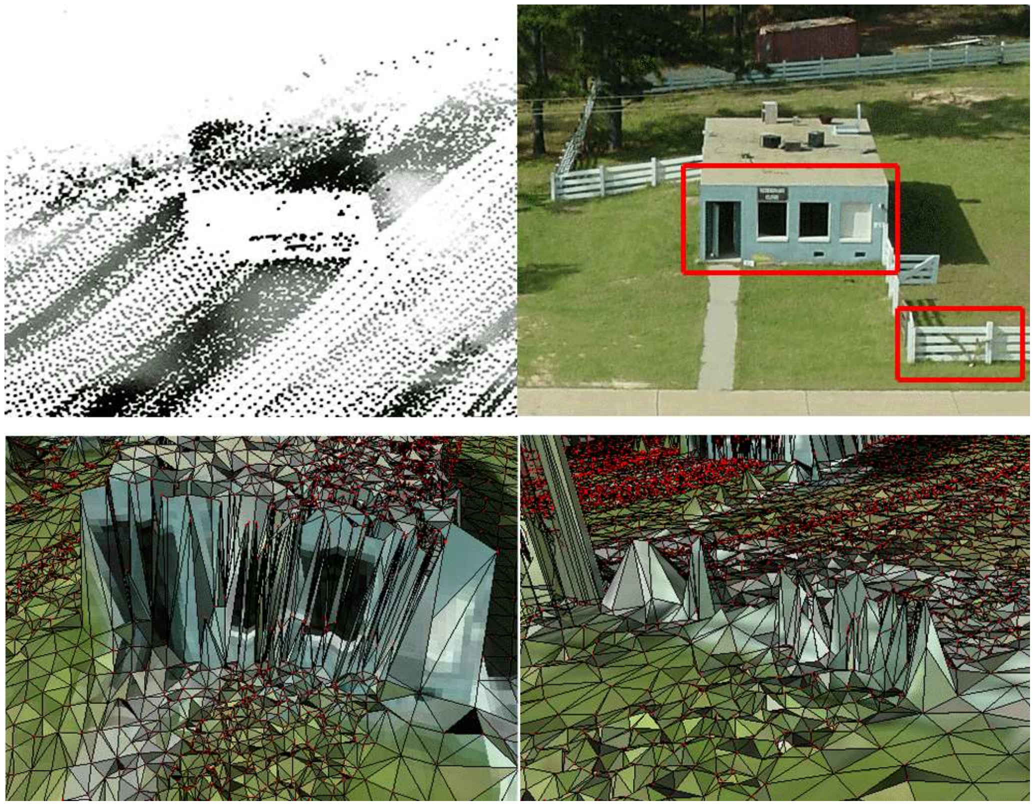


Fig. 2.6. The top row has a range map on left and its aligned image on right. Range data and images are combined to generate textured 3-D models. Range data points are often triangulated to generate a mesh onto which textures from images are mapped. However, range data has noise and cluttered regions that lack consistent edges and structure. Rendered images of the 3-D model as shown in the bottom row show the poor structural modeling of the house and the picket fence.

extract the transformation. Before good correspondences can be estimated, a reasonable estimate of the transformation is needed. On the other hand, if reliable correspondences, are not available how can the transformation be determined?

The issue of finding correspondences has two sources of errors. How well do edges extracted from an image relate to edges in a range data? In other words, are the edges stable under various lighting conditions and changes in albedo. A range model in most cases, does not have any information about surface albedo or lighting condition. If an edge in an image does not accurately show a surface property of a model under a given lighting condition, it is hard to determine correspondences between the two. An environment full of shadows or texture and reflectance variations like arrays of windows, and a forest of trees are few such examples.

The second source of error comes from the very nature of range data. The dense 3-D range point-cloud suffers from noise or errors in geometry. This noise is caused by the quantization of the numerical point coordinate data as well as inaccuracy in spatial distribution of the mesh density or topology [Figure 2.4]. Moreover, range data often misses out regions of the surface, especially at the edges and other critical points. This is attributed to absence of suitable viewpoints of the range sensor, noise, occlusions or abrupt discontinuities in the surface. An actual surface is approximated by a meshed triangulation of the 3-D range points and the absence of clear edges in the range data leads to poor modeling of the surface [Figure 2.6].

Can the model be always described as a collection of edges? Smooth scenes like fields and facades of large building pose a problem. Extremal mesh or lines (Thirion, 1996) can be determined but would those correspond to their image counterparts? Scenes like forest of trees do not have distinct edges or useful silhouettes. In the presence of a large number of features like edges, it is difficult to determine correspondences. Epipolar constraint can be used to prune the space of potential correspondences but the problem of finding reliable correspondences in reasonable time still remains.

In many applications, range data is often collected from mobile robots. Due to the very nature of combining scans of cross-sections of the surface from a moving platform, additional errors in positions of 3-D points creep in, as can be seen in Figure 2.5. The presence of partially resolved 3-D surfaces as well as point-samples of regions that lack definite structure, for example foliage makes registration of a model with images a difficult task. At the same time, surface generation and refinement using the 3-D point-cloud becomes impossible because of a large number of newly revealed or occluded faces in each iteration. The discontinuous and *holey* nature of range data introduces large faces in a mesh that have high re-projection error when compared to images. It is therefore, important that the generated surface model concentrate on regions where the data captures more coherent structure. This makes the refinement of surface geometry and physical data on the geometry by measuring their consistency with images feasible.

3 Surface Description

The view of a surface as seen in a 2-D image changes as its viewpoint varies. There are three approaches of determining this variation and the new appearance corresponding to the particular viewpoint. The first approach explicitly computes a 3-D description of the surface and generates a new view by rendering a full 3-D model [Figure 3.1]. The second is based on introducing non-linearities into a sequence of 2-D images, interpolating between the set of 2-D images, or use image-based priors in regularizing the solution [Figure 3.2. The description of surface geometry is implicit in this case and pixels of the new or synthesized view are derived from the rays sampled by the pixels of given images. geometry is used to guide the selection of color at each pixel of the new image. A third approach is to use shape and texture variability encoded as appearance. In this technique, the description of the surface in the new view is such that it best explains observed data. The description of the surface is such that color of pixels or texture guide surface geometry and view-dependent surface geometry is used to guide the selection of color at each pixel in the new image.

In any case, the description or encoding of appearance has to be consistent or coherent with images, depth data, or any other observed data such as surface normal or curvature. The appearance, which represents both shape variation and texture of the region covered by the model, can be used to generate full synthetic images of modeled objects. The difference between the synthesized image and the observed image, and any inconsistency between the predicted depth and observed depth can be used to update the appearance encoding. This framework can be categorized into four parts.

- Deriving a set of descriptors from image and depth data.
- Combining various descriptors to form an appearance model.
- Rendering a 3-D model for synthesized images corresponding to new viewpoints from an appearance model.
- Determining the coherence of the physical 3-D model with observed depth data, and the coherence of the new synthesized images with observed images.

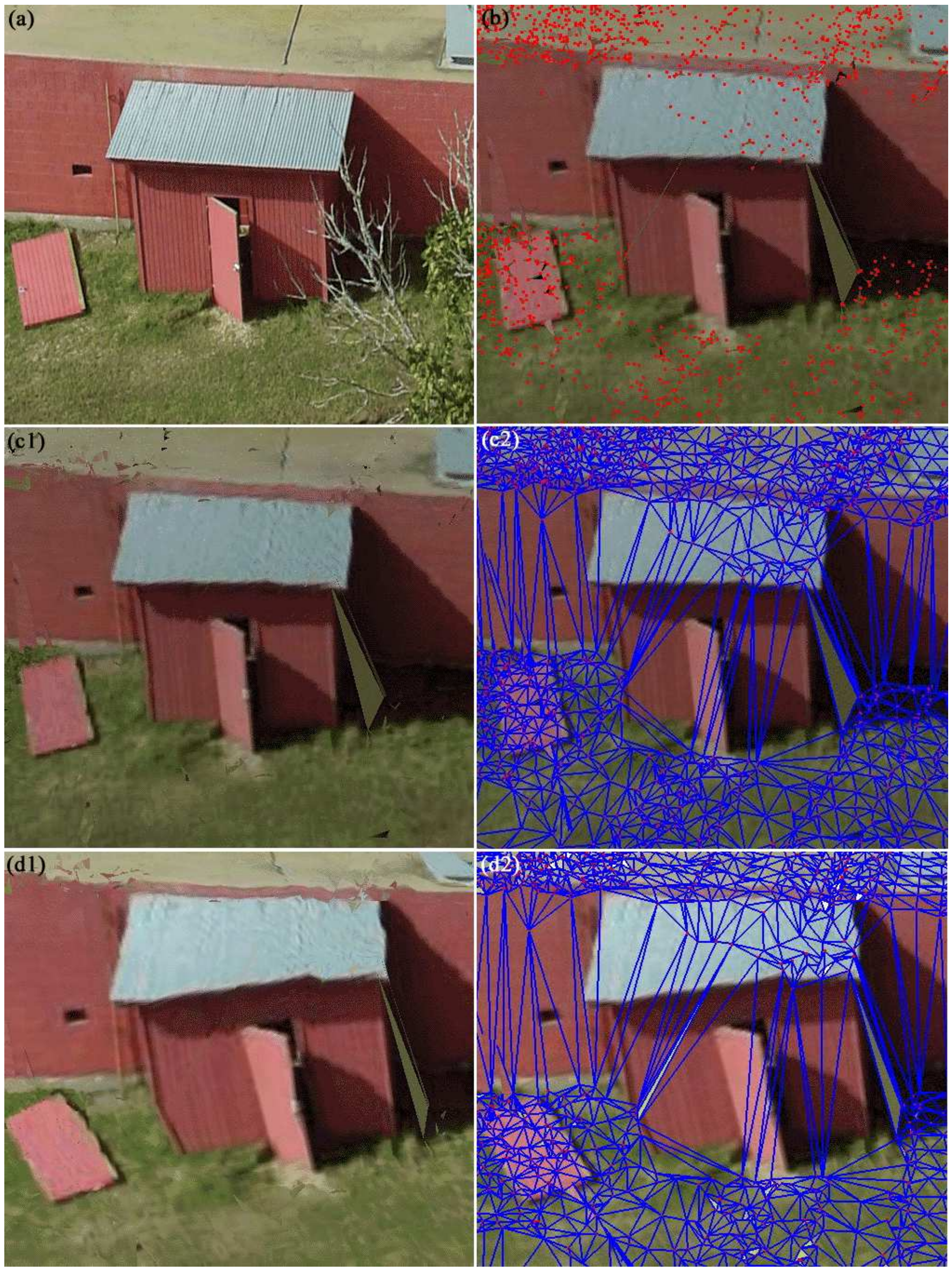


Fig. 3.1. Model-based Rendering: (a) The original image as observed from a particular viewpoint, (b) Depth data superimposed on the rendering of the modeled surface from the same viewpoint, (c1) Rendered image of the model, (c2) The underlying mesh of the 3-D model, (d1) Rendered image from a slightly different viewpoint, and (d2) The corresponding mesh as viewed from the new viewpoint.

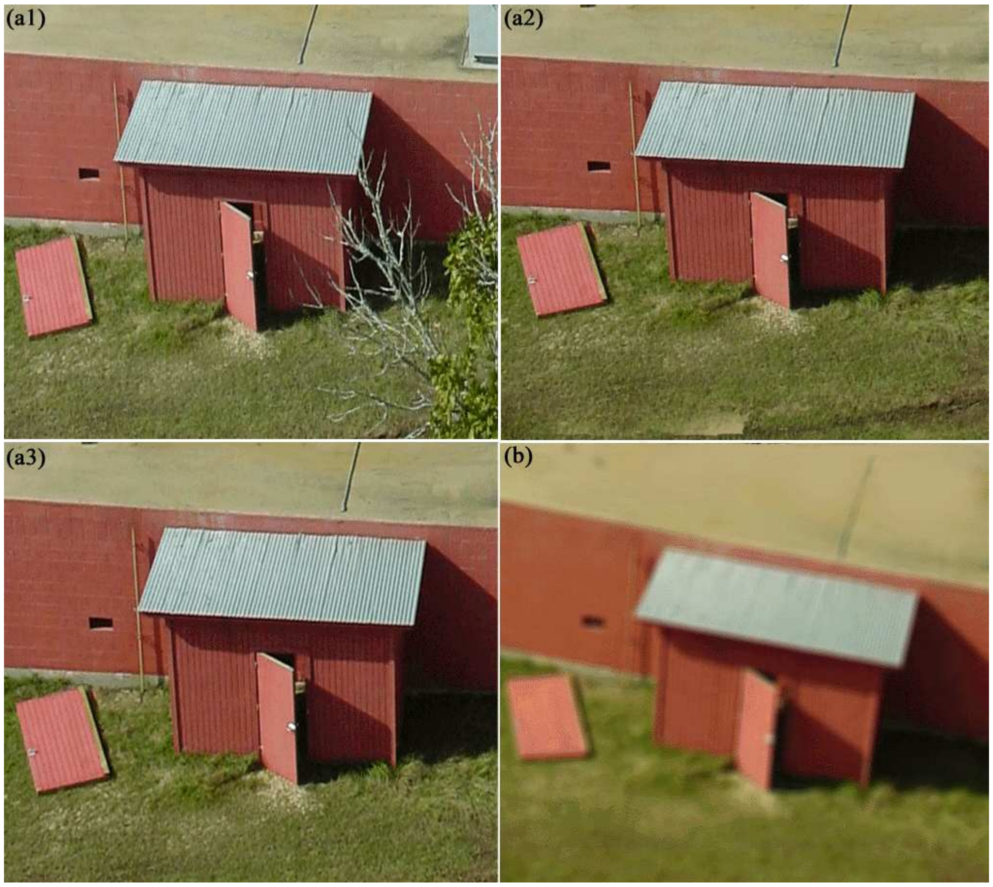


Fig. 3.2. View-based Rendering: (a1, a2, a3) Three from a set of 8 images taken from different viewpoints, and (b) Image rendered from a new viewpoint using the image-based rendering technique proposed by (Fitzgibbon et al., 2003). The new viewpoint is around 16 degrees apart from the closest viewpoint in the image sequence.

3.1 Image and Depth Descriptors

The process of imaging can be interpreted as a mapping of the scene into a set of images of much lower dimensionality. Two-dimensional images are perceptions of the three-dimensional world. The operations involved in the world-to-image mapping are projection and sampling. Nonetheless, this mapping results in the loss of information more than just depth. This refers to the fact that there are many configurations of the world, not merely related to structure or shape, that could give rise to an observed set of images.

In describing the world in term of images, we are making an attempt to reverse the world-to-image mapping. The description of the world that we seek, depends on the goal of our perceptual process. In any case, the number of possible descriptions of the world that could give rise to the observed image and depth data is more than one. Vision algorithms resolve this ambiguity by imposing constraints of one form or another on the set of descriptions.

Geometric constraints relate corresponding features, lines through those features, and planes formed by those lines in form of the trilinear tensor equations for three views or epipolar constraints for two views (Hartley and Zisserman, 2000; Faugeras et al., 2001). These constraints relate the depth of a points to their projections in different views. The constant brightness constraint assumes that the brightness of the corresponding point does not change between views. Similarly, the surface smoothness constraint assumes that surfaces do not have any discontinuities in depth (Marr and Poggio, 1979) (Grimson, 1983). The Lambertian assumption (Horn, 1977)

The world-to-image mapping is made invertible by imposing enough constraints on the space of possible surface descriptions. Violation of a constraint is supplemented by new constraints that allow a unique description to be obtained. However, all these constraints or assumptions are additional information, often *ad hoc* in nature, and further removed from the whole sensory and perceptual process of images and depth. We assume that the visual process can be inverted by taking into consideration certain assumptions or constraints that bear little relevance to the process of projection, sampling and conception of visual descriptors.

What constraints or assumptions, based on the nature of the perceived world and the manner it is sensed and perceived, can be derived in a visual process?

It is not possible to obtain universally-applicable constraints. Nevertheless, we seek constraints that are statistically true. Each of these constraints acts on raw sensory data to yield a cue. If one constraint or assumption is violated in a visual process, the cue obtained thereof is invalid. In order to obtain a domain-independent visual process such as inverting the world-to-image mapping, it is necessary to use a set of descriptors. In the presence of a set of different descriptors, the dependence of the inverse mapping on an invalid cue through the violation of an constraint is reduced.

Descriptors encode local shape of structures, their spatial relationships, the colored albedo and its interaction with incident light, all of which constitute appearance. To reiterate, constraints act upon the raw sensory data, both depth and images, to yield descriptors. Various depth and image descriptors are combined to form appearance.

An interesting question regarding descriptors is that whether descriptors are viewpoint invariant.

Use statistics of local regions of data, both from 2-D images and 3-D range data to derive descriptors that describe appearance of the scene. Registration (between inter-modal data) and matching (between data of same modality) use measures or metrics based on these regional statistics. Appearance coherent descriptors are more complete descriptors than mere geometrical features of range data sets and intensity-based features of images.

3.2 From Images and Depth to Appearance

Most appearance models in vision are simple in their definitions. The simplicity can be ascribed to the complexity in dealing with full 3-D models that consist of two disparate elements: structure and surface texture or reflectance. Further, the two elements are coupled together in their interaction with incident light and viewing direction, to form image pixels. Therefore, a definition of appearance makes a few simplifications regarding geometrical structure, surface property, imaging model, or lighting. For example, templates or exemplars are appearance models that consist of one or more 2-D maps of pixels from images. This definition of appearance captures both textural properties and spatial properties of the object. Nevertheless, it fails to account for significant changes in shape due to varying viewpoint or object rotation. It has, therefore, been combined with deformation models to account for shape variations.

Appearance has been modeled as a linear combination of components that capture small variations in appearance, lighting, and low-dimensional perturbations. The model uses pixel values around features that depend on edge orientations, curvatures and other local surface attributes such as texture. Most appearance models to date, are based on 2-D maps or parameterizations of pixel regions, and fail to model diverse or complex appearances. This is primarily due to the fact that these models do not exploit the spatial arrangement of pixel values among different regions, that correspond to different textured patches of the object. That is to say, modeling appearance by using a set of templates or image basis functions does not consider the spatial relationship between various image and shape features that form descriptors.

3.2.1 A Single Image or Depth Descriptor

Before we delve deep into the problem of inferring and estimating a global measure such as coherence, by combining estimates from several descriptors, we shall determine the dependence of the measure on a single cue. The cue could be based on color, depth or any attribute of a projected point. We shall now derive this salient relationship between a global measure and estimates from a single descriptor. The relationship has the same form as the Crofton's Theorem in integral geometry.

A set of points in an image along with a specific attribute, say intensity values at the points or a function of the intensities thereof, in a local neighborhood form a descriptor. In other words, we define a *descriptor* to be the set of points along with their attribute in a local neighborhood. Say, a descriptor $\mathit{\Gamma}$ be defined over N points $\mathbf{q}_1, \mathbf{q}_2, ...\mathbf{q}_N$ and let H be a function which depends on the positions of these N points and the corresponding attribute values. We would like to relate a small change in the descriptor $\mathit{\Gamma}$ to a small change change in the expectation of the function. Let $\mathit{\Gamma}'$ be the set of points slightly smaller than $\mathit{\Gamma}$ and contained in $\mathit{\Gamma}'$. Denote by $\mathit{\Delta\Gamma}$ the part of $\mathit{\Gamma}$ not in $\mathit{\Gamma}'$. If $\mathit{\Delta\Gamma}$ is small enough that the expectation of two or more of $\mathbf{q}_i$ falling in $\mathit{\Delta\Gamma}$ can be neglected,

$$\begin{aligned} E[H] \cong E[H|\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_N \in \mathit{\Gamma}']E[\mathbf{q}_1, \mathbf{q}_1, ..., \mathbf{q}_N \in \mathit{\Gamma}']+ \\ \sum_{j=1}^{N} E[H|\mathbf{q}_j \in \mathit{\Delta\Gamma}, \mathbf{q}_i \in \mathit{\Gamma}' \forall i \neq j]E[\mathbf{q}_j \in \mathit{\Delta\Gamma}]E[\mathbf{q}_i \in \mathit{\Gamma}' \forall i \neq j] \end{aligned} \tag{3.1}$$

Let C, C' and $\mathit{\Delta}C$ denote the quantitative measures (e.g., area, volume, etc.) of $\mathit{\Gamma}$, $\mathit{\Gamma}'$ and $\mathit{\Delta\Gamma}$ respectively. Since the points are random in $\mathit{\Gamma}$,

$$E[\mathbf{q}_j \in \mathit{\Delta\Gamma}] = \mathit{\Delta}C/C \tag{3.2}$$

$$E[\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_N \in \mathit{\Gamma}'] = \left[C'/C\right]^N \tag{3.3}$$

Therefore,

$$\begin{aligned} E[H]C^N \cong E[H|\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_N \in \mathit{\Gamma}']{C'}^N + \\ NE[H|\mathbf{q}_1 \in \mathit{\Delta\Gamma}, \mathbf{q}_2, ..., \mathbf{q}_N \in \mathit{\Gamma}']{C'}^{N-1}\mathit{\Delta}C \end{aligned} \tag{3.4}$$

Since the descriptor $\mathit{\Delta\Gamma}$ is taken to be small enough that the higher order terms of $\mathit{\Delta\Gamma}$ can be neglected,

$$C^N = (C' + \mathit{\Delta}C)^N \cong {C'}^N + N(C')^{N-1}\mathit{\Delta}C \tag{3.5}$$

Substituting in equation (3.4),

$$C'(E[H]-E[H|\mathbf{q}_1, \mathbf{q}_2, .., \mathbf{q}_N \in \Gamma']) \cong N\Delta C(E[H|\mathbf{q}_1 \in \Delta\Gamma, \mathbf{q}_2, .., \mathbf{q}_N \in \Gamma'] - E[H]) \tag{3.6}$$

If ΔC becomes infinitesimal and Γ' is so close to Γ that C' approaches C, the difference $E[H] - E[H|\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_N \in \Gamma']$ is a small increment $\delta E[H]$ and ΔC becomes δC. In such a situation, $E[H|\mathbf{q}_1 \in \Delta\Gamma, \mathbf{q}_2, ..., \mathbf{q}_N \in \Gamma']$ becomes the expectation $E[H|\mathbf{q}_1]$ of H when one of the random points is on the boundary $\delta\Gamma$ of Γ. This is the Crofton's theorem on "local probability" Crofton (1885),

$$\delta E[H] = \frac{N(E[H|\mathbf{q}_1 \in \delta\Gamma] - E[H])}{C}\delta C \tag{3.7}$$

3.2.2 Combining Multiple Descriptors

The integration of information coming from multiple descriptors or modalities is a fundamental problem of any perceptual process. Pixel statistics, segmented color regions, motion detection or optical flow, shape and stereopsis have been used as visual descriptors for integrating information (Clark and Yuille, 1990).

The visual system obtains information about depth from direct methods such as binocular stereo, observer motion or object motion. In other cases, it calculates depth via secondary descriptors such as shading, texture gradients, shadows, occlusion, familiarity, 3-D interpretation of line drawings and parallax. However, combination of various descriptors with depth descriptors to infer or estimate some relevant global parameter has not been pursued in depth. The global parameter could be pose of the scene with respect to the visual system or coherence of a 3-D model with respect to observed data.

The goal of our research is to develop a combination framework that incorporates descriptor reliability and regional information to derive appearance measures from multiple descriptors. It uses appearance measures to account for observed data.

Estimation of a global parameter such as pose of a viewer with respect to the scene or coherence of a rendered image with an actual image of the scene, involves multiple descriptors. Visual perception involves multiple descriptors because different parts of the scene offer different descriptors. This is primarily attributed to variations in texture, or surface reflectance and geometrical properties. Individual descriptors are descriptors of appearances of local regions and therefore, depend on local shape. Some regions may have textural variations while some others may have shape that lead shading variations. to Other factors responsible for multiple descriptors in visual perception are clutter, Multiple descriptors vary in the degree to which they are used in estimating a global parameter. Weights assigned to estimates derived from a single descriptors reflect its reliability and *informativeness* in the scene and its viewing condition. Let us consider a few examples. Shading and shadow descriptors are weighted more at places where textural variations are small. Descriptors from distinct textural patterns are weighted more than shading descriptors. Motion and depth descriptors have approximately equal weightage at close distances, but motion descriptors are weighted more at far distances, presumably due to distance limits on binocular disparity Johnston et al. (1994). Descriptors based on edges due to surface discontinuities and occlusion are given more weightage for discriminating between small rotations and translations. Experiments have found that descriptor weighting are sensitive to image manipulations; if a descriptor is weakened, such as by adding noise, then the uncontaminated descriptor is utilized more in making judgment.

The multiple descriptors considered for combination need not always be in the same dimension. While depth of a point is a scalar, color has three components, edges can be parameterized by length and orientation, and point statistics are multi-dimensional. Any formulation derived for combining various descriptors, therefore, has to account for the difference in dimensions of the descriptors.

Suppose for $j = 1, .., k$, $\varGamma_j$ is a *descriptor* in n_j-dimensional Euclidean space, with content C_j, and that N_j points are chosen at random from $\varGamma_j$. Let μ be the expected value of some function of the $\sum_{j=1}^{k} N_j$ points which depends only on intrinsic properties of the points such as local surface properties, and their relative positions (local structure) and not on the descriptors $\varGamma_j$ or the positions of the points relative to them. Then

$$d\mu = \sum_{j=1}^{k} N_j \frac{(\mu_j^* - \mu)}{C_j} dC_j \tag{3.8}$$

where $d\mu$ is the increment in μ obtained by increasing $\varGamma_j$ by an infinitesimal increment dC_j, and μ_j^* is the expected value of the function when one point is chosen at random in the boundary in $\varGamma_j$, N_j-1 points are chosen at random in $\varGamma_j$ and N_i points are chosen at random in $\varGamma_i$ for $i \neq j$, $i = 1, 2, , k$, and dC_j/dT is the derivative of the content of the descriptor C_j with respect to transformation.

The above derivation is known as the generalized Crofton's theorem or the Ruben-Reed extension to Crofton's theorem Ruben and Reed (1973) . It holds true for *any* continuous probability distribution of the random points, not just for an uniform distribution. The most important effect of this generalization is that the distribution of the attribute as well as errors in the position of the points that constitute the descriptor do not change the theorem. Further, this generality allows it to be applied to other random geometrical elements (such as line segments, flats, ellipses) which can be represented as points that form an appropriate descriptor. The same relationship holds, but the measure μ has to be interpreted accordingly.

A salient aspect of formulation (3.8) is that each domain Γ_j can be defined in n_j-dimensional Euclidean space. This allows each descriptor to be parameterized in its own special way i.e., different number of variables. In other words, each local region in a model or an image can be described by a set of variables specific to the descriptor. Individual descriptors are calculated over local regions, and the global measure depends on each descriptor. μ_j^* captures the characteristics of a a particular local region varies while descriptors from others regions are kept constant.

This formulation is ideal for combining the statistics of range data Huang et al. (2000) and those of natural images Huang and Mumford (1999) Grenander and Srivastava (2001) . The problem of course, lies in the definition of the distance measure or metric and the interpretation of the boundary measure. In natural scenes such as foliage, 2-D joint statistics of Haar wavelet coefficients or joint distribution of intensities at two neighboring points in a domain can be used. Color information from image as well as range data can also be used to define descriptors if available.

Another salient aspect of calculating the measure over various descriptors is that regional statistics of images or models can be easily considered. Individual descriptors are calculated at local regions, and the global measure depends on each descriptor. μ_j^* captures the characteristics of a a particular local region varies while descriptors from others regions are kept constant.

3.3 Surface Description using a Texture-Mapped Mesh

3.3.1 Triangulated Mesh of Depth Data

We want to derive a surface description from multiple images and depth data. The depth data or shape can be extracted from the images themselves, either through stereo, shading or SfM. How does one describe or represent a surface in the form of a 3-D model? We need a surface description that can be used to generate images of the surface from arbitrary viewpoints, taking into account surface discontinuities, occlusions, self-shadowing, specularities and other viewpoint-dependent effects. It is clear that a single view-based representation does not suffice this need. Inverting the world-to-image mapping requires a model-based description.

Among the many model-based descriptions possible, the one we seek should be able to represent any surface, closed or open, and of arbitrary genus. Second, it should be relatively simple to generate an instance of a surface from depth maps or point samples of the surface. Third, new images of the surface using model-based rendering should be computationally feasible. This necessarily means that the parametric description of the surface should correspond to the actual 3-D shape of the surface. The true shape of the surface is unknown. However, the object-based description should explain best what is observed in images and depth data. That is to say, the description is *coherent* with the observations of the unknown surface. This coherence leads to the close correspondence of the descriptive model and the actual surface. Finally, the parameters specifying the model should be a compact and collectively complete in regards to the description of the surface. By varying the parameters, the coherence between the descriptive model and the actual surface can be varied.

A regular 3-D triangulation is one such surface description that meets the above mentioned criteria. It is a set of vertices, edges, faces, and an albedo or reflectance map associated with each face. In regards to modeling surface shape and reflectance, the set is a compact and complete description of the actual surface. It is easy to generate a texture-mapped 3-D triangulation from a set of images and 3-D points extracted as depth. Textured triangular faces are particularly simple to be rendered for generating images, occlusions and shadows.

Standard triangulation algorithms can be used to generate a 3-D mesh from points. (Fua and Sander, 1992) (Szeliski and Tonnesen, 1992). Further, initial meshes can be refined with edge swaps, vertex removal and other mesh operators.

3.3.2 Texture-mapping a Triangulated Mesh

The 3-D triangulated mesh represents the shape of the surface. It is an model-based representation of the surface. Modeling the appearance of the surface requires that points on the mesh be associated with a reflection function. The reflection function maps the wavelength distribution and orientation of a light source, the normal of the surface, and the viewing direction into the color of an image pixel at a point. Like shape, it is unknown as well. The exact function is difficult to model. However, most surface reflectance functions over a surface patch can be modeled using one or few parameters (Horn, 1977).

The Lambertian reflectance function that has a single parameter is one of such functions. The single parameter, albedo, is the ratio of reflected to incident light intensity. The reflected light intensity as viewed in images is independent of viewpoint. As a result of this generalization, image intensity can be used to compute albedo. In this thesis, we have confined our focus on surface properties to albedo or textural attribute.

3.4 Coherence of a Texture-mapped Mesh

In order to describe a surface of an object or scene, we must model both its shape and its surface reflectance or texture. Shape variations, texture variations and any interdependence between them have to be represented. By ‘texture’ we refer to the pattern of color across a surface patch. While shape characterizes the geometry of the surface, texture represents a physical property of the surface that accounts for the amount of incident light reflected. Shape and texture of a surface patch when viewed under a lighting condition constitutes appearance.

Shape and texture are straight forward to describe and represent. On the other hand, appearance is a combination of shape and texture variations, and can be represented in many ways. Moreover, the representation of appearance depends on surface patches it collectively attempts to model. Natural objects and scenes consist of various surface patches that vary both in their local shape and texture properties. Any attempt in representing such scenes or objects by a fixed set of appearance models is both inadequate and superficial.

The necessity of representing the surface of an object by appearance that is collectively based on different surface patches is clear. Depth and pixel data corresponding to each patch are used to form a statistic called a cue. Cues from different surface patches are combined to form a global measure of the object such as appearance.

The mapping between appearance of a textured patch and its representation in eigenspace is not completely understood. How can a surface patch be well represented in feature-space based on its view? How can its appearance from a new view be estimated from its principle components (i.e., projection onto its eigenspace)?

3.4.1 Coherence using Eigenspace Representation

Determination of appearance coherence requires that we establish a compact parameterization of appearance variability. We will show that a considerable gain in accuracy of modeling a surface can be achieved by selecting an appropriate representation of appearance. The representation of appearance should capture any variability in the modeled object properties, both shape and texture. It is this variability that can be effectively used in determining coherence of the modeled properties with observed data. It is, therefore, extremely important that modeling of appearance variability be done in an effective and compact manner. Moreover, models can be compared and fitted to new images in a fraction of a second, given a reasonable initialization by exploiting the approximate prior knowledge about the local nature of the optimization space.

The variability of appearance can be modeled using Principal Component Analysis (PCA), i.e., an eigenspace representation of the dispersions of shape and texture.

3.4.2 Minimizing the Coherence Measure

If we have a 3-D model that captures the geometry of the actual object accurately, there are many methods to measure texture coherence. The simplest measure is the color difference between a pixel of the re-projected image of the model and the actual image of the surface. This is, of course, a very local measure since it uses one sample per point of texture and very sensitive to noise. More robust measures use several samples for a single point by considering a neighborhood of pixels. A common criterion is the normalized cross-correlation between the two neighborhoods, one that belongs to the re-projected image and the other that belongs to the actual image. This measure compares the intensity distributions within the neighborhoods. The criterion is more robust to failures in the Lambertian assumption for the surface.

If an accurate 3-D model is unknown, we can recover it by maximizing the geometrical and textural coherence for a given set of views of the surface. Both the coherence measures put together constitute appearance coherence. Fua and Leclerc (1996) have linearly combined the two measures. The first component is a measure of the deformation of the surface from a nominal shape, and independent of the images. In our case, we refer to this component as geometrical coherence or depth coherence. The second component depends on images, and is the one that drives the reconstruction process. This component is similar to the textural coherence that we have defined in our research. However, our approach uses both geometrical and textural coherence to drive the model reconstruction process. We believe that both geometrical structure and surface properties such as texture are coupled together while reconstructing an accurate textured 3-D model. This necessitates the use of a model reconstruction and refinement framework that exploits both structural and textural coherence and any correlation between them. This combined coherence measure is based on the model's appearance and is termed as *appearance coherence* in our research.

Algorithms for modeling an object by maximizing the texture criterion have used two different optimization strategies. The first class of algorithms use texture similarity to evaluate a current model. If the measure is improved by deforming the model locally, then the model is refined and the process iterated as in (Fua, 1997; Fua and Leclerc, 1995) or level-set based methods (Colosimo et al., 2001; Faugeras and Keriven, 1998) where a volumic band is searched around the current model. The search remain locally dependent on the current model. The algorithm can fail due to the presence of local extrema during the texture coherence optimization. The second class of optimization algorithms tests all possible configurations. In order to be robust, the algorithms accumulate the coherence values into a 3-D grid by using a voting scheme as in (Esteban and Schmitt, 2003; Nobuhara and Matsuyama, 2003; Medioni et al., 2000). This approach is very robust to highlights and specularities. However, there exists a elegant method based on appearance coherence that combines descriptors from both geometrical structure and texture to accurately model a surface.

4 Depth Coherence

3-D vision consists to two distinct modalities of information: image pixels and depth. However, image pixels and depth share a relationship with each other–both are descriptors of the same object. The idea of describing an object that is consistent with pixels in observed images is well understood. Descriptions of object should be consistent with previously viewed images, new synthesized images or image-based cues already learnt by or built in our visual system. However, there exists yet another coherence constraint that has to be satisfied as well. This coherence measure is based primarily on depth cues or the structural geometry of the scene. Such surface constraints are exploited in stereo matching (Pollard et al., 1985; McLauchlan et al., 1991; McLauchlan and Murray, 1995). Dense stereo algorithms Scharstein et al. (2001) implement the depth constraint as optimizing a smoothness term. The surface smoothness term and temporal coherence are used to solve the aperture problem in optical-flow algorithms as well Barron et al. (1994). Another assumption based on the physical geometry of the scene involves rigidity of the scene, i.e., the camera moves in a stationary scene. The rigidity assumption is sufficient to solve the feature correspondence problem Fitzgibbon and Zisserman (1998) based on the fundamental matrix and tri-focal tensor—geometric constraints derived from two or three images of a static scene.

Using independent features based on surface smoothness, temporal coherence, or rigidity constraints, or any combination thereof, scales well to complex scenes that are rich in textural or geometrical features. However, this point-based representation loses out on a dense and enriched description of a 3-D scene. 3-D scenes consist of surface patches corresponding to various objects, and a cloud of point-features cannot describe these patches intoto, and at the same time, capture any relationship of the points lying between the point features. Treating a 3-D scene as a cloud of features, having no relationship to each other, can lead to contradictory results whereby incorrect features may yield isolated features under the surface of objects, which can only be detected once the

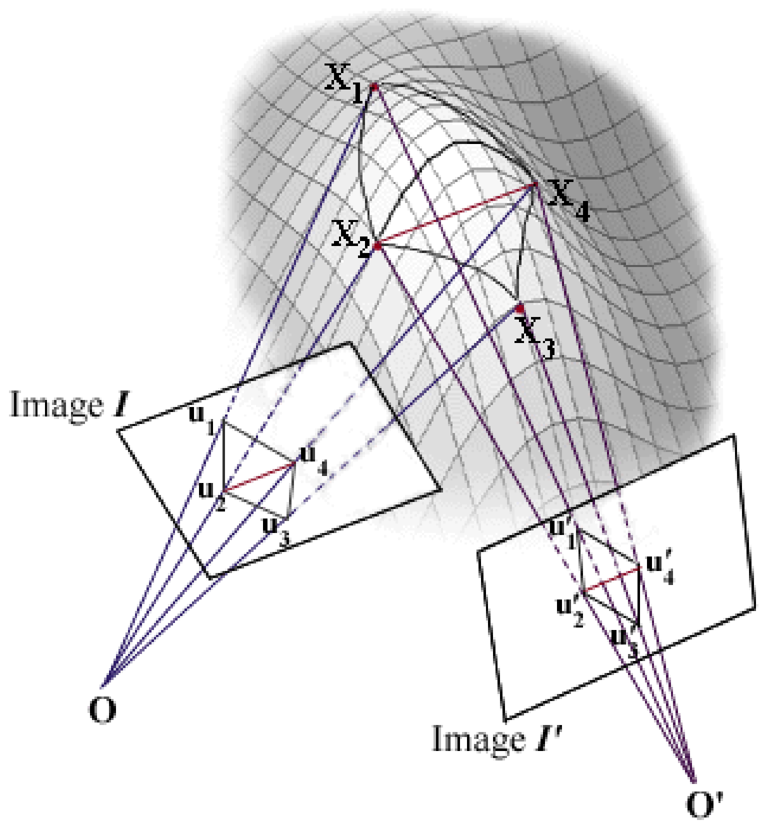


Fig. 4.1. A curved surface patch $\mathrm{X}_1\mathrm{X}_2\mathrm{X}_3\mathrm{X}_4$ is viewed in images $\mathbf{I}$ and $\mathbf{I}'$. Projection of the 3-D features that define the patch, $\mathrm{X}_1, \mathrm{X}_2, \mathrm{X}_3, \mathrm{and X}_4$, are points $\mathbf{u}_1, bu_2, \mathbf{u}_3, andbu_4$ in image $\mathbf{I}$ and $\mathbf{u}'_1, bu'_2, \mathbf{u}'_3, andbu'_4$ in image $\mathbf{I}'$. The 3-D features are triangulated to approximate the mesh. Image areas within the projected triangulation depend on the view-dependent relationship between surface depth (curvature), texture, and pose of the images with respect to the surface patch.

surfaces are themselves recovered. These surface contradictions are especially common at edges of occlusion, clutter, or other surface discontinuities.

The idea of describing a surface patch with point features can in fact be extended to representing a surface patch with dense set of points that not only capture changes in surface curvature but also surface texture. A sparse set of point features is often triangulated to generate a mesh to approximate a curved patch. Having smaller mesh faces or a

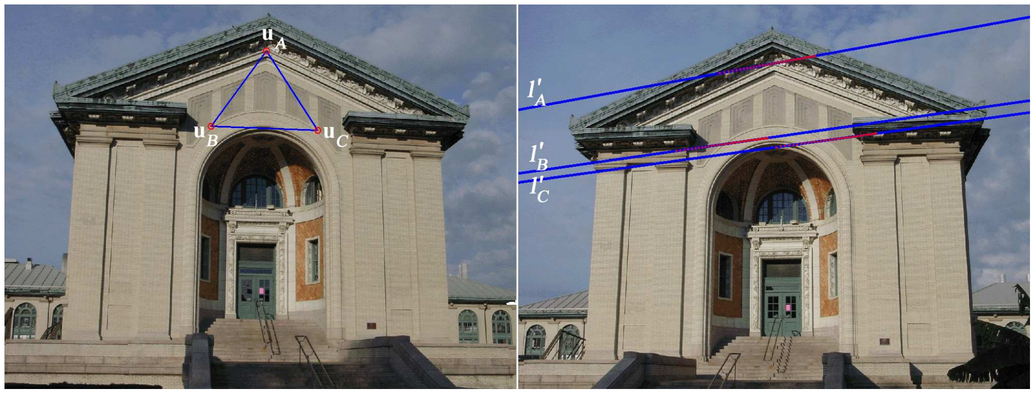


Fig. 4.2. For points $\mathbf{u}_A$, $\mathbf{u}_B$, and $\mathbf{u}_C$ in the left image, l'_A, l'_B, and l'_C are the epipolar lines in the image on right. The points in the right image corresponding to $\mathbf{u}_A$, $\mathbf{u}_B$, and $\mathbf{u}_C$ lie somewhere on the epipolar lines. The red dots on the epipolar lines are points sampled at regular depth intervals. The pixel intensities at points along the epipolar lines can be used to determine correspondences in the two images. Smoothness of the local patch formed by the 3-D points, poses an additional constraint that can be used to guide the search for correspondences.

refined mesh adds new points to the set of features. One may recursively refine the mesh faces to better approximate the curved surface patch. A better strategy of achieving such a close approximation would be to directly derive a dense set of points that models the patch with low reconstruction error.

In order to achieve a low reconstruction error, one may exploit the geometry-based relationship between views and depth of the surface patch. Views of the surface patch register how the surface curvature varies across the patch. In fact, local curvature *modulates* local texture of the patch, and any projection of the textured patch into images registers this process [Figure 4.1]. The problem can now be stated as: How does one combine surface curvature, texture, and the multiple-view relationship between images to model the surface? In this chapter, we extend the the point-based epipolar constraint to a surface patch-based constraint to derive a more complete 3-D description of a surface by encompassing the smoothness, temporal coherence, and rigidity constraints. We incorporate surface patches into existing frameworks of feature-based stereo and structure-from-motion algorithms.

4.1 Depth and Epipolar Geometry

The *epipole* is the point of intersection of the line joining the optical centers — the *baseline* — with the image plane. The epipole is the projection in one camera of the optical center of the other camera. The *epipolar plane* is the plane defined by a 3-D point and the optical centers or, equivalently, by an image point and the optical centers. The line of intersection of the epipolar plane with the imaging surface is the *epipolar line*. It is the projection in one camera of a ray passing through the optical center and an image point in the other camera [Figures 4.2]. In other words, a point in one image generates a line in the other on which its corresponding point must lie. The search for correspondences is thus reduced from a region to a line. This relationship, also called the epipolar constraint, between points in two images corresponding to the same 3-D point arises because the image points, the 3-D point, and the optical centers are coplanar.

The epipolar constraint is a purely geometrical in the sense that it relates the coordinates of the image points. However, the search for correspondence between two image points often uses pixel intensities at the image points and intensities along their epipolar lines (Wexler et al., 2003; Brandt, 2004). In addition to pixel-intensities along epipolar lines, one can also exploit an addition constraint that is posed by the structure of the scene. Light rays leaving each 3-D feature and converging on each camera center can be adjusted optimally with respect to both feature coordinates and camera positions. Since the structure is rigid, all of the 3-D features and their projections onto the imaging planes share a global relationship. This is also termed as *bundle adjustment* whereby a jointly optimal 3-D structure and camera parameters are estimated using a global constraint that quantifies model fitting error. But again, bundle adjustment is just another large sparse geometric parameter estimation problem, the parameters being 3-D feature coordinates, camera poses, and calibration Triggs et al. (2000).

4.2 Creating Isodepth Maps using Epipolar Geometry

If we were to extend bundle adjustment to a dense set of points on a particular surface patch, what additional constraint would we need? Applying bundle adjustment to a set of point coordinates that are close to one another is difficult as the model fitting error is sensitive to small perturbations in point coordinates. Minimizing re-projection error of such a set of points that lie on a continuous surface patch, that is smooth or has small structural variations, does not lead to a unique solution (up to a scalar factor). In some dense set of points, also called *critical surfaces*, images allow at least two distinct epipolar matrices.

Extending bundle adjustment to a dense set of points can exploit the intensities of pixels at the points. However, exploiting pixel intensities of points on a 3-D surface patch

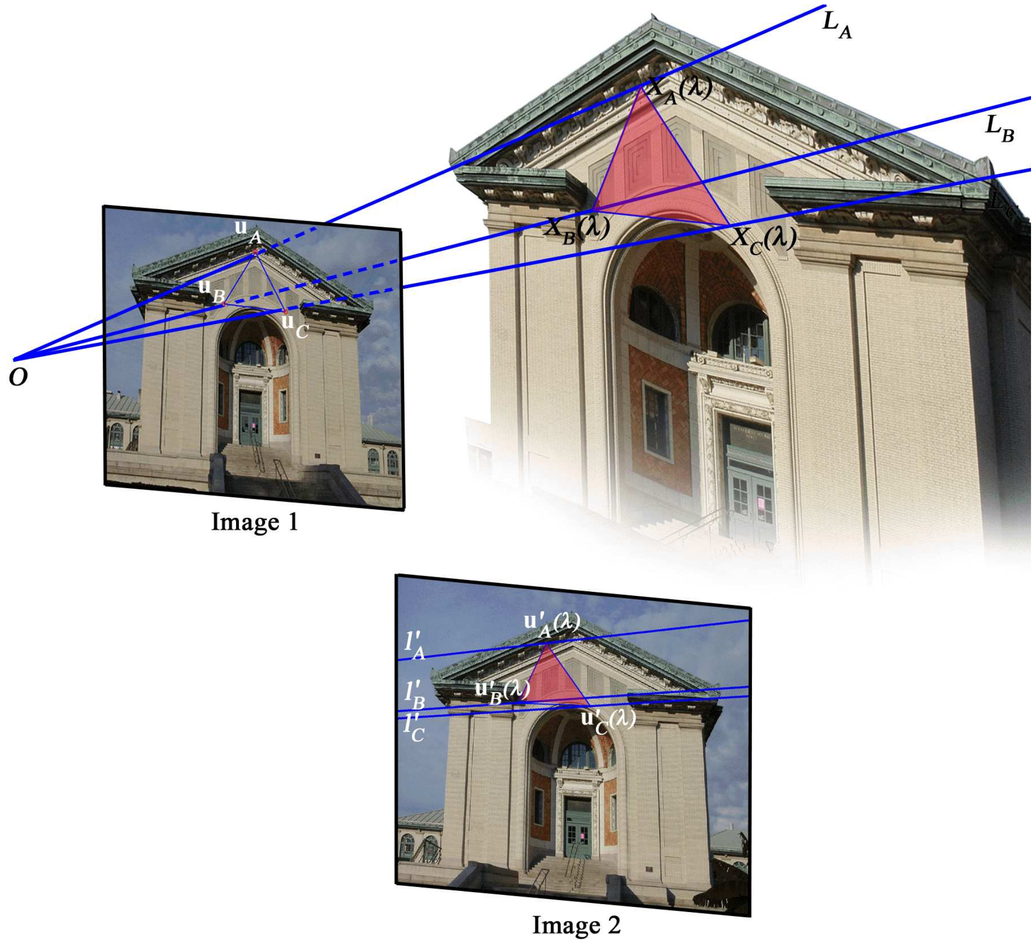


Fig. 4.3. l'_A, l'_B, and l'_C are the epipolar lines in Image 2 corresponding to fixed points $\mathbf{u}_A$, $\mathbf{u}_B$, and $\mathbf{u}_C$ in Image 1. The epipolar lines are projections of the 3-D rays L_A, L_B, and L_C defined by the camera center O and the points in Image 1. Points $X_A(\lambda)$, $X_B(\lambda)$, and $X_C(\lambda)$ on those 3-D rays are at depth λ from the camera center. Therefore, the triangular patch (in red) with $X_A(\lambda)$, $X_B(\lambda)$, and $X_C(\lambda)$ as vertices is at the same depth. It can also be visualized as the equi-depth cross-section of the tetrahedral cone formed by O and the fixed image points, $\mathbf{u}_A$, $\mathbf{u}_B$, and $\mathbf{u}_C$. The projection of the triangular patch into Image 2 is the triangle formed by $\mathbf{u}'_A(\lambda)$, $\mathbf{u}'_B(\lambda)$, and $\mathbf{u}'_C(\lambda)$ — projection of points $X_A(\lambda)$, $X_B(\lambda)$, and $X_C(\lambda)$ in Image 2.

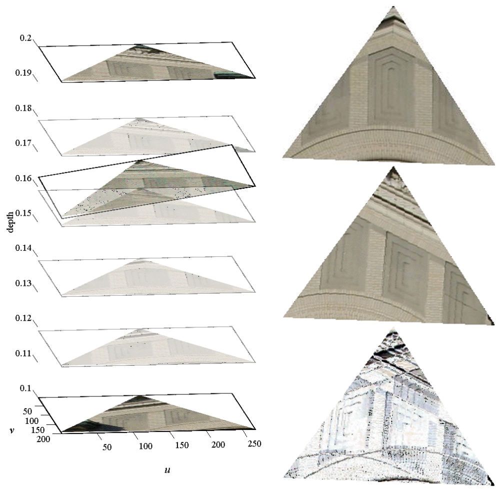
as viewed in images requires a representation whereby the epipolar geometry between views of the patch can be determined. Let us illustrate this using a simple example. Consider a triangular and planar surface patch in 3-D. In Figure 4.3, this patch is denoted by $X_A X_B X_C$. Its projection into two images yields two triangular image areas of pixel intensities—$\mathbf{u}_A\mathbf{u}_B\mathbf{u}_C$ in Image 1 and $\mathbf{u}'_A\mathbf{u}'_B\mathbf{u}'_C$ in Image 2. The image areas are related to each other via an affine transform. Points on the 3-D surface patch lie at different depth values with respect to a camera. Consider a hypothetical plane that is parallel to the imaging plane. Points on the plane are at the same depth from the camera's optical point. We call this plane an isodepth plane. In that case, the points $X_A(\lambda)$, $X_B(\lambda)$, and $X_C(\lambda)$ lie on an isodepth place at depth λ. In other words, 3-D epipolar rays corresponding to three points on the image intersect the isodepth plane and form a triangular 3-D patch. This can be visualized as the equi-depth cross-section of the tetrahedral cone formed by the optical center, O, and the three image points— $\mathbf{u}_A$, $\mathbf{u}_B$, and $\mathbf{u}_C$—as seen in Figure 4.3.

Projection of the 3-D triangular patch formed by the isodepth plane and the three points $X_A(\lambda)$, $X_B(\lambda)$, and $X_C(\lambda)$ onto the imaging planes yields triangular image areas. A point of the actual surface that lies on the 3-D triangular patch projects to corresponding pixels that lie in these image areas. Under the Lambertian assumption, pixel intensities at the projected points should be the same as they correspond to the same 3-D point. If we stacked all the triangular image areas, one on another, based on depth, points on the surface patch would correspond to different depth values based on the structure of the patch. Of course, the intensities of the points can be used to determine where they are located.

Stacking the image areas determined at different depths creates a 3-D search space of pixel intensities, that we term as an *Isodepth Map*. Figure 4.4 shows an *Isodepth Map* created by stacking the image areas $\mathbf{u}'_A(\lambda)\mathbf{u}'_B(\lambda)\mathbf{u}'_C(\lambda)$ at varying depth λ. The Isodepth Map is, therefore, a 3-D volume of intensity values sampled from Image 2 that correspond to a planar patch parallel to Image 1 at varying depth. It is formed by the projection of an equi-depth plane to one camera at different depths into another camera. Therefore, there exist two isodepth maps between two cameras; each one being the isodepth map with respect to one camera into the other. Let us denote a isodepth map between camera i and j as D_{ij}. The equi-depth planes are parallel to the image in camera i and projected into image j to sample pixels for the isodepth map. For N equi-depth planes or values of λ, this can be represented by:

$$D_{ij} = \{I_j(\lambda_k) \mid k = 1, 2, ...N\} \tag{4.1}$$

The two isodepth maps between camera i and j are, therefore, D_{ij} and D_{ji}. The slices of both maps correspond to the same surface patch and are related to each other by a rigid transformation (translation and rotation) that is the same as the transformation between the cameras i and j.


(a) Planar Slice

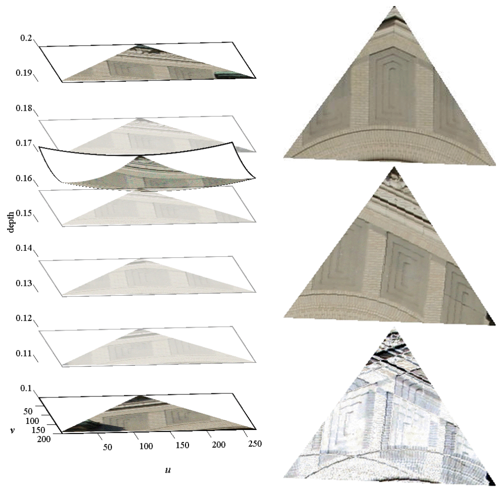

(b) Curved Slice

Fig. 4.4. The Isodepth Map in Image 2 corresponding to the fixed triangular patch in Image 1 is shown on the right of each sub-figure. The fixed triangular image area in the top right is the area in image 1 formed by the triangle $\mathbf{u}_A$, $\mathbf{u}_B$, and $\mathbf{u}_C$. The image underneath it is the image formed by slicing the Isodepth Map. The difference between the image area from Image 1 and the sliced image from the Isodepth Map in Image 2 is shown at the bottom-left corner of each sub-figure.

4.3 Slicing Isodepth Maps

Points of the actual 3-D surface that is not planar being modeled lie at different depths with respect to the cameras. The only case where all the points of the 3-D surface lie at the same depth is that of a planar patch which is parallel to the imaging plane. In all other cases the points of a 3-D patch correspond to different depth values and, therefore, lie on different equi-depth planes. Equivalently, the points correspond to different depths in the isodepth map. If we assume that a patch is continuous, its points form a continuous surface or slice in the isodepth map.

Consider a planar patch which is not parallel to the imaging plane. For an equi-depth plane at a given depth value the planar patch and the equi-depth plane intersect in a line segment. The projection of this line segment onto another image is another segment that samples a line of intensity values. In the isodepth map these intensity values lie at the same depth. As the equi-depth plane moves through different depths it scans the 3-D planar patch linearly. The projected line segments sample lines of intensity values. These collectively form a plane in the isodepth map as seen in Figure 4.4(a).

For a curved but continuous surface patch, 3-D points corresponding to the surface get mapped to a continuous slice of the isodepth map. Intensity values carved out of the isodepth map by this slice correspond to texture on the patch. Each point in 3-D structure gets mapped to a point in the isodepth map, both in geometrically and texturally. The equi-depth plane that each point lies on gets mapped to a corresponding depth plane in the isodepth map, thereby mapping the geometrical structure. The intensity value at the point in the isodepth map corresponds to the texture value of the point on the surface. In addition, points in the isodepth map that correspond to the continuous surface patch form a continuous slice. By a continuous surface patch we refer to a neighborhood of 3-D points that do not have discontinuities due to occlusions, holes, or clutter. Figure 4.4(b) shows a continuous but curved slice of the isodepth map.

For an isodepth map, D_{ij}, let $I_p(D_{ij})$ denote the slice through D_{ij} that yields a 2-D image of intensity values. The slice that we are looking for is the one that corresponds to the surface patch being viewed. Points of this slice have depth values that constitute structure of the surface with respect to camera i. Pixel intensities of the volume that determine how this slice is determined come from image j.

4.4 Depth Coherence with Isodepth Map Slices

For a Lambertian surface, the texture as determined by the isodepth map slice would be the same as the texture of the patch as viewed in the image to which the equi-depth planes of the isodepth map are parallel. The 2-D image of intensity values as determined by the slice I_p should be the same as the image area of the surface patch in image i. In Figure 4.3, the image area $\mathbf{u}_A\mathbf{u}_B\mathbf{u}_C$ in Image 1 and the slice of its isodepth map, $D_{2\,1}$, in Image 2 should be the same. The image area $\mathbf{u}_A\mathbf{u}_B\mathbf{u}_C$ is shown in Figure 4.5(b). Two slices of the isodepth map are shown in the same figure. The slice determined by a plane parallel to image 1 at a depth of $\lambda = 0.1790$ return an image as seen in 4.5(c1). It is different when compared to image area $\mathbf{u}_A\mathbf{u}_B\mathbf{u}_C$ because the structure as determined by the planar slice is not close to the actual surface. However, the slice at a depth of $\lambda = 0.1384$ returns an image of intensity values that is very close to the image area $\mathbf{u}_A\mathbf{u}_B\mathbf{u}_C$ in Image 1. Therefore, depth values corresponding to the slice are closer to the actual surface. We refer to this property as depth coherence, that is to say, the structure and texture as computed by the slice are coherent with each other.

The coherence between the pixel intensities at the sliced depth values and the image area that the isodepth map is based on can be used to determine the slice. In Figure

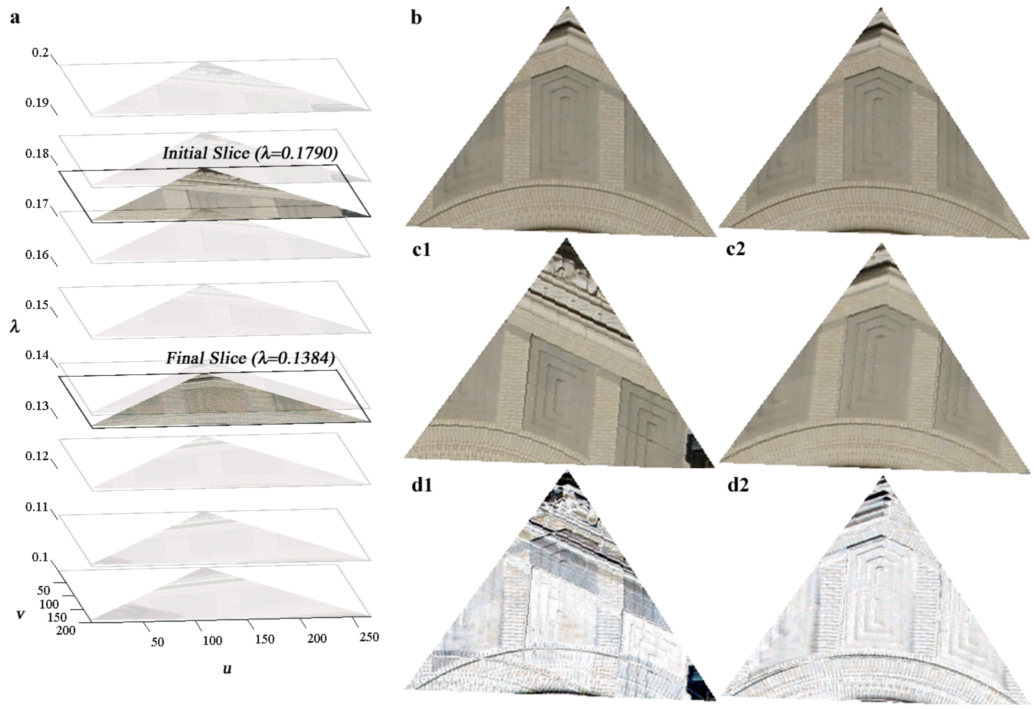


Fig. 4.5. (a) The planar slice of the Isodepth Map corresponding to an initial depth of $\lambda = 0.1790$ and the final slice at $\lambda = 0.1384$. (b) The fixed image area corresponding to the triangular patch in Image 1. (c1) The image area corresponding to the initial slice of the Isodepth Map as shown in (a) (d1) The difference between the two. (c2) The final slice image. (d2) The difference of the fixed triangular pixel area in Image 1 and the final slice of the Isodepth Map.

4.5(d1,d2) and Figure 4.6(d1,d2), the difference between the fixed triangular pixel area in Image 1 and the pixel area determined by the slice of isodepth map is used to determine how the isodepth map is sliced. For the case in Figure 4.5 the slice is only allowed to vary in depth and is always horizontal, i.e., all points on the slice have the same depth. Minimizing the error between the sliced image and the fixed image areas leads us to a slice at a depth that best approximates the actual 3-D surface patch. It can be visualized as an orthographic approximation of the 3-D surface, i.e., a planar patch that is parallel to the image plane and best approximates the 3-D surface patch.

Let us derive a simple formulation for determining the slice that returns an pixels that are closest to the fixed image area. For an image i, let $I_{i\mathcal{F}}$ denote the fixed triangular image area for a surface patch $\mathcal{F}$, and $D_{ij\mathcal{F}}$ the isodepth map between image i and image j. D_{ij} is constructed by taking equi-depth planes parallel to image i and stacking the image areas projected into image j. Pixels in the image area $I_{i\mathcal{F}}$ are represented by

$\boldsymbol{u}$. For depth coherence, the objective function to be minimized is:

$$E(\lambda) = \sum_{\boldsymbol{u}_{\mathcal{F}}} \rho\big([D_{ij\mathcal{F}}(\lambda)|_{I_p}](\boldsymbol{u}) - I_{i\mathcal{F}}(\boldsymbol{u})\big) \tag{4.2}$$

where $I_{i\mathcal{F}}$ the error norm ρ defined over residual pixel-error over the image areas. We determine the depth λ for which the sliced image area $[D_{ij\mathcal{F}}(\lambda)|_{I_p}]$ of the depth map $D_{ij\mathcal{F}}$ is closest to the fixed image area $I_{i\mathcal{F}}$. To achieve this, we minimize E with respect to λ using an iterative gradient descent algorithm: at each step m we take the actual estimate $\lambda^{(m)}$ and calculate a proposed update $\delta\lambda^{(m)}$. If the step is successful, the proposed depth is accepted and λ is updated as $\lambda^{(m+1)} = \lambda^{(m)} + \delta\lambda^{(m)}$. Otherwise, a more conservative update $\delta\lambda^{(m)}$ is calculated, and the step repeated. For a gradient descent with feedback step size adjustment, the update rule is:

$$\delta\lambda^{(m)} = -\mu \sum_{\boldsymbol{u}_{\mathcal{F}}} \dot{\rho}\big(e_p(\boldsymbol{u})\big) \; . \; \frac{\partial[D_{ij\mathcal{F}}(\lambda)](\boldsymbol{u})}{\partial\lambda} \tag{4.3}$$

$\dot{\rho}$, proportional to the influence function, is the derivative of the error norm ρ with respect to the residual pixel error, $e_p(\boldsymbol{u}) \equiv [D_{ij\mathcal{F}}(\lambda)|_{I_p^{(m)}}](\boldsymbol{u}) - I_{i\mathcal{F}}(\boldsymbol{u})$, for the slice $I_p^{(m)}$ in the m-th iteration. After a successful step, μ is multiplied by μ_f, otherwise it is divided by μ'_f. For the result shown in Figure 4.5, we had $\mu = 10^{-6}$, $\mu_f = 10$, and $\mu'_f = 20$. To increase robustness and reject outliers, the ρ-function must be able to reject outliers; that is, it should increase less rapidly than e_p^2. In the above depth estimation case, we have considered the Lorentzian norm as our error norm: ρ is defined over pixel error e_p in the image areas. Given a scale factor, σ, that controls the convexity of the norm, we have:

$$\rho(e_p) = \log\left[1 + \frac{1}{2}\left(\frac{e_p}{\sigma}\right)^2\right] \quad , \quad \dot{\rho}(e_p) = \frac{2e_p}{2\sigma^2 + e_p^2} \;\; , \text{ and } \quad \ddot{\rho}(e_p) = \frac{2(2\sigma^2 - e_p^2)}{(2\sigma^2 + e_p^2)^2}$$

When the absolute value of the gradient magnitude increases beyond a threshold determined by the scale factor σ, its influence is reduced; it is also referred to as a redescending influence function. If any local intensity error e_p has a large magnitude then the value of $\dot{\rho}(e_p)$ will be small and have little effect on the update of λ. The scale factor was set to 10 for the depth estimation of the patch in Figure 4.5. The depth of the initial slice, $\lambda = 0.1790$ converged to a value of $\lambda = 0.1384$ in 20 iterations. As can be observed in Figure 4.5(d1, d2), the absolute error between $I_{i\mathcal{F}}$ and the slice of the isodepth map reduced considerable for the depth transition.

The error in Figure 4.5(d2) is not as low as it should be. It can be observed that the error image is high at the edges of the patch as compared to the pixel error at the

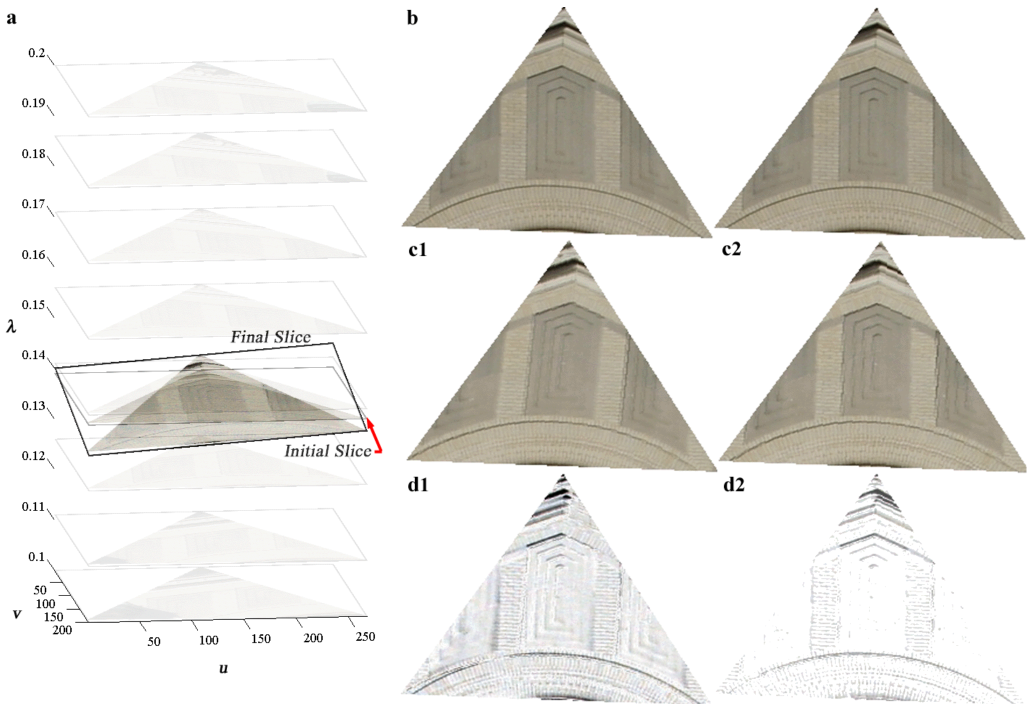


Fig. 4.6. (a) The planar slice of the Isodepth Map corresponding to an initial depth of and the final slice. (b) The fixed image area corresponding to the triangular patch in Image 1. (c1) The image area corresponding to the initial slice of the Isodepth Map as shown in (a) (d1) The difference between the two. (c2) The final slice image. (d2) The difference of the fixed triangular image area in Image 1 and the final slice of the Isodepth Map.

center. This is due to the fact that the actual 3-D surface patch, though planar, is not parallel to the image plane. This necessitates extra two degrees of freedom in allowing the isodepth map to be sliced at an angle. The depth and orientation of the surface patch determines how the isodepth map is sliced [Figure 4.6]. Starting with a horizontal slice at the depth estimated in the previous step, we allow the planar slice to have an orientation with respect to the horizontal plane; in this case, we parameterize the slice I_p as $\lambda = a_\lambda + a_u(u - u_0) + a_v(v - v_0)$. a_λ is the depth of the center or the average depth of the slice and u_0 and v_0 are coordinates of the center of the slice. The parameters a_u and a_v determine the orientation of the slice with respect to the two axes.

The error function E in equation 4.2 is now a function of $\mathbf{a} = (a_\lambda,\ a_u,\ a_v)$. For the optimization algorithm we need to calculate the partial derivatives of $E(\mathbf{a})$, as they form the gradient $\mathbf{g_a}$ and the Hessian matrix $\mathbf{H_a}$. We obtain the first partial derivatives for calculating the gradient vector $\mathbf{g_a} = (\ \partial E/\partial a_\lambda\ \ \partial E/\partial a_u\ \ \partial E/\partial a_v\ )^T$ as

$$\frac{\partial E}{\partial a_k} = \sum_{\boldsymbol{u}_{\mathcal{F}}} \dot{\rho}\big(e_p(\boldsymbol{u})\big) \ \frac{\partial [D_{ij\mathcal{F}}(\lambda)](\boldsymbol{u})}{\partial \lambda} \frac{\partial \lambda}{\partial a_k} \qquad \text{for} \qquad k \in \{\lambda, u, v\} \tag{4.4}$$

For $\lambda = a_\lambda + a_u(u - u_0) + a_v(v - v_0)$, we have

$$\frac{\partial \lambda}{\partial a_\lambda} = 1 \ , \quad \frac{\partial \lambda}{\partial a_u} = (u - u_0) \ , \quad \text{and} \quad \frac{\partial \lambda}{\partial a_v} = (v - v_0)$$

The $\{k, l\}$-th element of the Hessian $\mathbf{H_a}$ are the second partial derivatives:

$$\frac{\partial^2 E}{\partial a_k \partial a_l} = \sum_{\boldsymbol{u}_{\mathcal{F}}} \left(\ddot{\rho}\big(e_p(\boldsymbol{u})\big) \ \left(\frac{\partial [D_{ij\mathcal{F}}(\lambda)](\boldsymbol{u})}{\partial \lambda} \right)^2 + \dot{\rho}\big(e_p(\boldsymbol{u})\big) \ \frac{\partial^2 [D_{ij\mathcal{F}}(\lambda)](\boldsymbol{u})}{\partial \lambda^2} \right) \frac{\partial \lambda}{\partial a_k} \frac{\partial \lambda}{\partial a_l} \tag{4.5}$$

The Newton-Raphson method uses the values of $\mathbf{a}^{(m)}$, $\mathbf{g}_{\mathbf{a}}^{(m)}$, and $\mathbf{H}_{\mathbf{a}}^{(m)}$ calculated at step m to estimate the next set of parameter values, $\mathbf{a}^{(m+1)}$;

$$\mathbf{a}^{(m+1)} = \mathbf{a}^{(m)} - (\mathbf{g}_{\mathbf{a}}^{(m)})^{-1}.\mathbf{H}_{\mathbf{a}}^{(m)} \tag{4.6}$$

where $-(\mathbf{g}_{\mathbf{a}}^{(m)})^{-1}.\mathbf{H}_{\mathbf{a}}^{(m)}$ is referred to as the Newton direction. Figure 4.6 shows the slice of the isodepth map orienting itself to minimize the pixel error starting from a horizontal position at a depth of 0.1384, i.e., $\mathbf{a}^{(0)} = \begin{pmatrix} 0.1384 & 0 & 0 \end{pmatrix}$. The derivatives $\frac{\partial [D_{ij\mathcal{F}}(\lambda)](\boldsymbol{u})}{\partial \lambda}$ and $\frac{\partial^2 [D_{ij\mathcal{F}}(\lambda)](\boldsymbol{u})}{\partial \lambda^2}$ are calculated by convolving the isodepth map with a Gaussian derivative kernel. The method will converge in a few iterations only if the following Taylor series approximation is valid;

$$E(\mathbf{a}) \approx E(\mathbf{a}^{(m)}) + \left(\mathbf{a} - \mathbf{a}^{(m)}\right)^T .\mathbf{g}^{(m)} + \frac{1}{2}\left(\mathbf{a} - \mathbf{a}^{(m)}\right)^T .\mathbf{H}_{\mathbf{a}}^{(m)}.\left(\mathbf{a} - \mathbf{a}^{(m)}\right) \tag{4.7}$$

for a point $\mathbf{a}$ in a suitable neighborhood of the current point $\mathbf{a}^{(m)}$. The gradient $\mathbf{g}_{\mathbf{a}}^{(m)}$ and the Hessian matrix $\mathbf{H}_{\mathbf{a}}^{(m)}$ are evaluated at $\mathbf{a}^{(m)}$. Even though convergence may seem to be fast, the performance of the method is dependent on the positive definiteness of the Hessian. For the method to converge towards a minimum, the Newton direction needs to be a direction of descent which requires

$$\mathbf{g}_{\mathbf{a}}^{(m)}.\left(\mathbf{a}^{(m+1)} - \mathbf{a}^{(m)}\right)^T < 0 \tag{4.8}$$

Inserting equation 4.6 yields

$$\left(\mathbf{a}^{(m+1)} - \mathbf{a}^{(m)}\right).\mathbf{H}_{\mathbf{a}}^{(m)}.\left(\mathbf{a}^{(m+1)} - \mathbf{a}^{(m)}\right)^T < 0 \tag{4.9}$$

The inequality is satisfied at all points for which $\left(\mathbf{a}^{(m+1)} - \mathbf{a}^{(m)}\right) \neq 0$ if $\mathbf{H}_{\mathbf{a}}^{(m)}$ is positive definite. The further $\mathbf{a}^{(m)}$ is from the solution, the worse the quadratic approximation in equation 4.7 gets, which can result in the Hessian not being positive definite. The method is then no longer guaranteed to proceed to a minimum; it may end up at any other critical point, whether it be a saddle point or a maximum point. Starting at a depth estimate close to the average depth of the patch assured the positive definiteness of the Hessian for evaluating the orientation of the patch. For most patches, the intensity-based error function converged to a minimum within 10 iterations. However, starting with the average depth of the slice as an initial guess assures the Hessian being positive definite during optimization only if the surface patch is close to a planar approximation. For many patches, that were continuous but curved or had distinct geometrical features within, the error between the intensity slice and the fixed image did not converge to a minimum and the orientation estimate was trapped in a local minimum.

4.5 An Illustration

The last section discussed methods to determine depth and orientation of a surface patch by slicing the isodepth map. The isodepth map was sliced by using a planar section at a depth and an orientation that correspond to the average depth and pose of the patch with respect to the camera. This works well as long as the patch can be approximated by a planar face. However, many surfaces in the real world have curvature and cannot be approximated by planar faces of a mesh. One of the ways of dealing with high curvature patches is by subdividing a face into smaller faces so that each new face lies closer to the actual surface patch being modeled. This subdivision can be iteratively done until the faces are small enough and lie close to the surface patch. This method motivates us to use a similar strategy for carving out a slice in an isodepth map.

Let us illustrate this by considering a curved patch. Figure 4.7 shows a textured patch on a sphere that we wish to model from a set of images. Pixels of the texture shown in the top left corner of the image are projected on to a spherical surface. Images of the textured patch are created for different camera poses. The best possible triangulation considering the four corners of the patch X_A, X_B, X_C, and X_D as vertices is the set of triangles $X_A X_B X_C$ and $X_A X_C X_D$. However, the planar approximation by faces of the mesh does not model the curved nature of the surface patch. We would like to assign every pixel a depth value as well as interpolate between those pixels to generate a dense

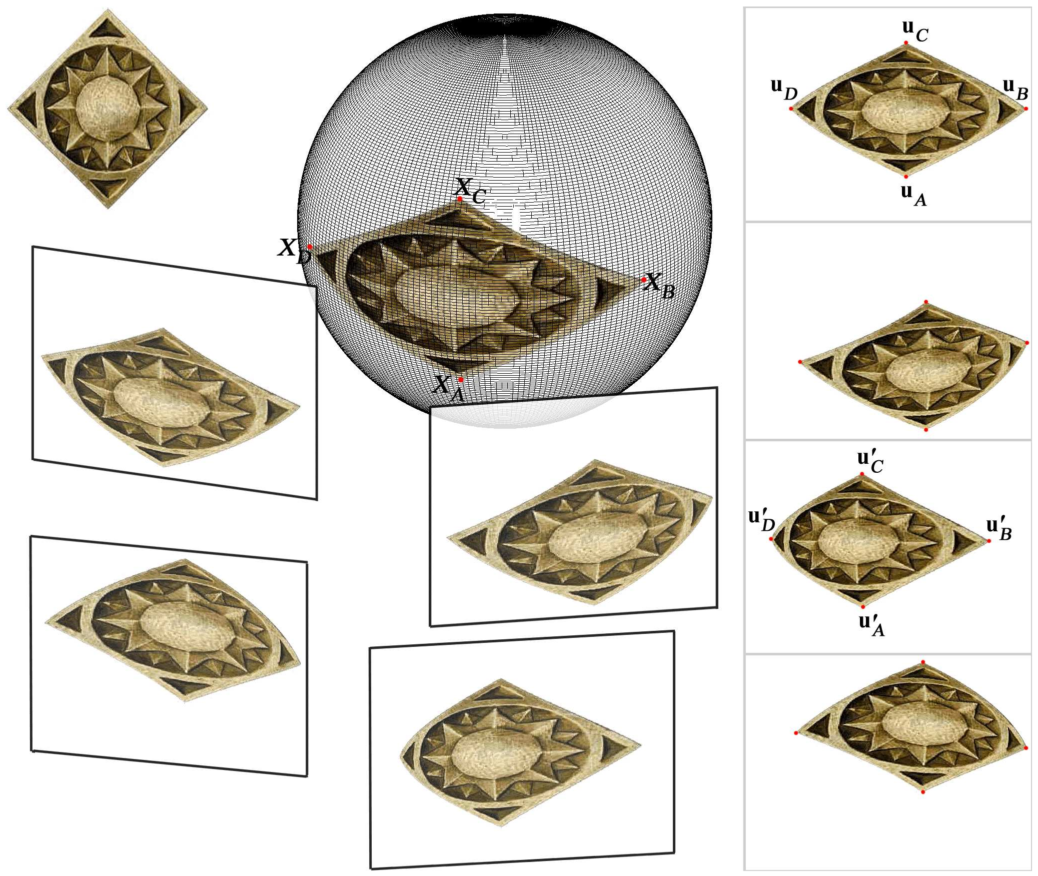


Fig. 4.7. The texture shown on the upper left corner is projected onto a sphere. Images of the textured sphere are taken from different views. Four of the images are shown on the right column. The triangular patches $X_A X_C X_D$ and $X_A X_B X_C$ are reconstructed from these images.

set of points with associated intensities. Every point in the set is assigned a depth and an intensity value.

For sake of illustrating, let us consider a single patch that is observed in two images [Figure 4.8]. The image area $\mathbf{u}_A\mathbf{u}_B\mathbf{u}_C$ in Image i corresponds to the surface patch $X_A X_C X_D$ in Figure 4.7. We denote this image area by $I_{i\mathcal{F}}$ for mesh face $\mathcal{F}$ that approximates the patch. Points on the planar mesh face will be perturbed to new 3-D positions that best capture the curved nature of the surface patch. The epipolar lines corresponding to $\mathbf{u}_A$, $\mathbf{u}_B$, and $\mathbf{u}_C$ are l'_A, l'_B, and l'_C in Image j. The triangular image area $\mathbf{u}'_A(\lambda)\mathbf{u}'_B(\lambda)\mathbf{u}'_C(\lambda)$ corresponding to the equi-depth plane at depth λ is extracted from the image by triangulating the points $\mathbf{u}'_A(\lambda)$, $\mathbf{u}'_B(\lambda)$, and $\mathbf{u}'_C(\lambda)$ on the epipolar lines. The points on the epipolar lines are taken at discrete and fixed intervals of depth.

The image areas are affine warped to $I_{i\mathcal{F}}$ and then stacked onto one another to form the isodepth map of the patch in Image j with respect to Image i: $D_{ij\mathcal{F}}$. In this illustration, we have considered a depth interval of 10^{-5} units of focal length. The depth samples vary between a fixed interval of preset values such that every pixel in Image j that could possibly belong to the patch is covered by at least one equi-depth image plane. In other words, we allow the equi-depth plane to range between an estimated depth interval between which points of the 3-D surface patch lie. For constructing the isodepth map $D_{ij\mathcal{F}}$ shown in Figure 4.8, the depth λ ranges from 0.00512 to 0.00522. A depth interval of 10^{-5} gives us 11 discrete equi-depth planes including the first and last planes. Once a rough depth estimate of the 3-D patch is determined, we can refine this range and consider smaller depth intervals.

4.6 Using Priors for Slicing Isodepth Maps: Epitexture Constraint

The isodepth map $D_{ij\mathcal{F}}$ has to be sliced at depth values $\boldsymbol{\lambda}$ such that the image intensities carved out by the slice match the image area $I_{i\mathcal{F}}$. The objective, therefore, is to maximize the probability of $P(\boldsymbol{\lambda}|I_{i\mathcal{F}})$. For a given image area $I_{i\mathcal{F}}$, Bayes rule is used to calculate the *a posteriori* probability of the depth slice $\boldsymbol{\lambda}$;

$$P(\boldsymbol{\lambda}|I_{i\mathcal{F}}) \propto P(I_{i\mathcal{F}}|\boldsymbol{\lambda})P(\boldsymbol{\lambda}) \tag{4.10}$$

$P(\boldsymbol{\lambda})$ is the prior on $\boldsymbol{\lambda}$, and $P(I_{i\mathcal{F}}|\boldsymbol{\lambda})$ measures the likelihood that the image area $I_{i\mathcal{F}}$ would be observed if $\boldsymbol{\lambda}$ consists of the actual depth values for the surface patch. This probability can be maximized either by Simulated Annealing Kirkpatrick et al. (1983) or Iterated Conditional Modes Besag (1986). The corresponding depth slice $\boldsymbol{\lambda}$ is the *maximum a posteriori* (MAP) estimator of the actual structure of the patch. In other words, it is the posterior probability of $\boldsymbol{\lambda}$, the shape or structure of the patch, given the corresponding image area $I_{i\mathcal{F}}$.

The brightness or texture value of a 3-D point at depth λ when projected into the image plane is often assumed to be corrupted by Gaussian white noise with mean zero and some non-zero variance ν^2. In a more general case, let us model the noise as being drawn from a distribution with a density function of the form $exp\left\{\frac{-\rho(e_p)}{2\nu^2}\right\}$ where $\frac{1}{2\nu^2}$ is the width of the distribution and $e_p \equiv [D_{ij\mathcal{F}}(\lambda)] - I_{i\mathcal{F}}$. We will have $\rho(e_p) = e_p^2$ for a Gaussian distribution. The likelihood at pixel coordinate ***u*** is

$$P(I_{i\mathcal{F}}(\boldsymbol{u})|\lambda) = exp\left\{\frac{-1}{2\nu^2}\ \rho\big([D_{ij\mathcal{F}}(\lambda)](\boldsymbol{u}) - I_{i\mathcal{F}}(\boldsymbol{u})\big)\right\} \tag{4.11}$$

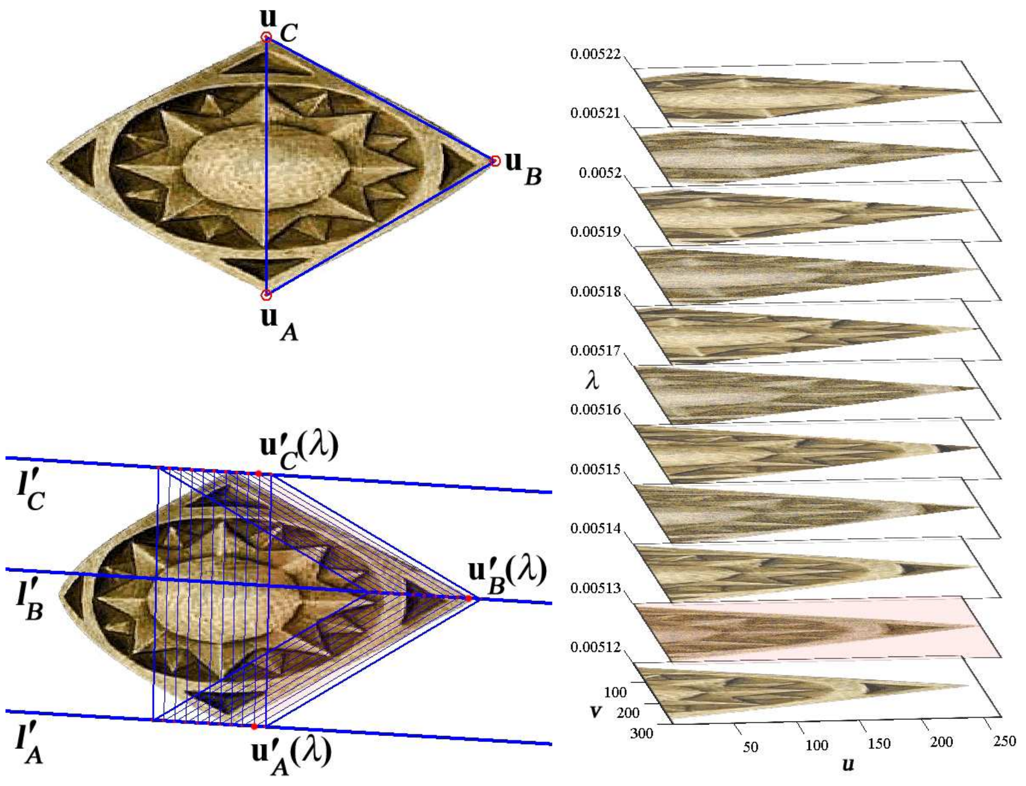


Fig. 4.8. $\mathbf{u}_A$, $\mathbf{u}_B$, and $\mathbf{u}_C$ are points in Image i triangulating the image area $I_{i\mathcal{F}}$. Their corresponding epipolar lines in Image j are l'_A, l'_B, and l'_C. $\mathbf{u}'_A(\lambda)$, $\mathbf{u}'_B(\lambda)$, and $\mathbf{u}'_C(\lambda)$ are points on the epipolar lines at depth λ. The triangular area $\mathbf{u}'_A(\lambda)\mathbf{u}'_B(\lambda)\mathbf{u}'_C(\lambda)$ is the projection of the equi-depth plane in Image j that corresponds to the image area $I_{i\mathcal{F}}$ $\mathbf{u}_A\mathbf{u}_B\mathbf{u}_C$. The equi-depth image areas are warped onto $I_{i\mathcal{F}}$ and stacked according to depth. This forms the isodepth map $D_{ij\mathcal{F}}$. The planar slice in red corresponds to the image area $\mathbf{u}'_A(\lambda)\mathbf{u}'_B(\lambda)\mathbf{u}'_C(\lambda)$ at $\lambda = 0.00513$.

In addition to the Lambertian assumption, there is yet another assumption made in the above formulation; brightness or texture value at a 3-D point is independent of texture values at its neighboring points. Most surfaces in the real world have some form of texture and there is a degree of correlation between brightness values at points in a neighborhood. This could be addressed using a texture prior which we will discuss later.

4.6.1 The Depth Prior

The term $P(\boldsymbol{\lambda})$ encapsulates our knowledge about the surface continuity or smoothness. It relates the depth value at a particular point to depth values in the neighborhood of that point, i.e., arrangement of depth values within a patch. This spatial relationship gives us a

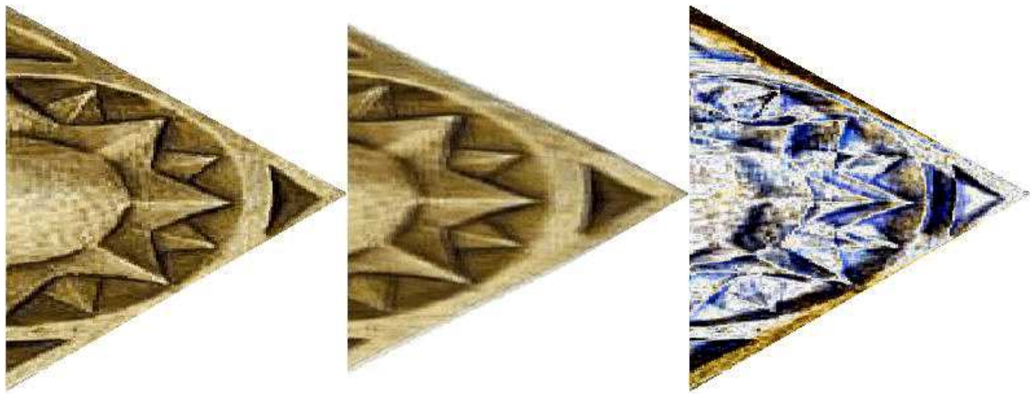


Fig. 4.9. On left is the image area $\mathbf{u}_A\mathbf{u}_B\mathbf{u}_C$ from Image 1. The image in the center is a planar slice of the isodepth map at $\lambda = 0.00513$. The difference between the two is shown on right.

way to deal with ambiguity that is inherent in the problem of 3-D surface reconstruction from 2-D images. In order to make a reasonable judgment of what we see in images, we have to regularize possible interpretations by applying some depth constraint or *a priori* knowledge of what the depth values within a neighborhood should be. We term this additional *a priori* information as the depth prior. Some could also see this as the shape or structure prior.

The depth prior—that a point on the slice is connected to its neighbors—is incorporated as a Markov Random Field (MRF). MRF is in itself a conditional probability model, where the probability of a depth value at a point depends on those in its neighbors. A Gaussian MRF is a frequent choice for a prior model, but the quadratic penalty in the Gaussian often causes excessive smoothing of edges. Several prior models have been developed which include desirable edge-preserving or sharper surface properties, and which maintain convexity in their log prior densities (Stevenson et al., 1994; Bouman and Sauer, 1993; Lange, 1990) . Provided we choose one of these models, the MAP problem is also convex, and iterations converge to the unique global minimum solution. We use the Generalized Gaussian MRF (GGMRF) model (Bouman and Sauer, 1993) with the density function

$$P(\boldsymbol{\lambda}) = exp\left\{-\frac{\mu^r}{r}\sum_{\{\boldsymbol{u}_k,\boldsymbol{u}_l\}\in\mathcal{N}} b_{kl}|\lambda_{\boldsymbol{u}_k} - \lambda_{\boldsymbol{u}_l}|^r\right\} \tag{4.12}$$

where $\mathcal{N}$ is the set of all neighboring points pairs of image coordinates $\boldsymbol{u}$, b_{kl} is the coefficients linking points $\boldsymbol{u}_k$ and $\boldsymbol{u}_l$, μ is a scale parameter, and $1 \leq r \leq 2$ is a parameter which controls the smoothness of the patch. This model includes a Gaussian MRF for

$r = 2$, and an absolute-value potential function for $r = 1$. In general, smaller values of r allow a more curved slice to form in the isodepth map. Larger values of r lead to a smoother patch without abrupt discontinuities.

The MAP estimate of depth values, that form a slice of the isodepth map, results from maximizing the following

$$\begin{aligned}\hat{\boldsymbol{\lambda}}_{MAP} &= \arg\max_{\boldsymbol{\lambda}}\ P(\boldsymbol{\lambda}|I_{i\mathcal{F}}) = \arg\max_{\boldsymbol{\lambda}} P(I_{i\mathcal{F}}|\boldsymbol{\lambda})P(\boldsymbol{\lambda}) \\ &= \arg\max_{\boldsymbol{\lambda}} \left[\log P(I_{i\mathcal{F}}|\boldsymbol{\lambda}) + \log P(\boldsymbol{\lambda})\right]\end{aligned} \tag{4.13}$$

Combining the depth prior from equation 4.12 with the likelihood expression in equation 4.11 yields the cost function

$$\hat{\boldsymbol{\lambda}}_{MAP} = \arg\min_{\boldsymbol{\lambda}} \left[\frac{1}{2\nu^2}\,\rho\big([D_{ij\mathcal{F}}(\boldsymbol{\lambda})] - I_{i\mathcal{F}}\big) + \frac{\mu^r}{r}\sum_{\{\boldsymbol{u}_k,\boldsymbol{u}_l\}\in\mathcal{N}} b_{kl}|\lambda_{\boldsymbol{u}_k} - \lambda_{\boldsymbol{u}_l}|^r\right] \tag{4.14}$$

Equation 4.14 may be minimized by using a local iterative technique such as iterated condition modes (ICM) proposed by (Besag, 1986). ICM is a deterministic optimization algorithm that iteratively minimizes the cost function at each pixel coordinate $\boldsymbol{u}$. The discrete Gauss-Seidel-like optimization technique accepts configurations that only decrease the cost function. Each pixel coordinate is sequentially considered for optimizing its depth while depth values at the neighboring coordinates are held fixed. The minimization of the cost function at each pixel coordinate $\boldsymbol{u}$ in the isodepth map with respect to the depth value at that coordinate, $\lambda(\boldsymbol{u})$, results in the following local computation

$$\hat{\lambda}_{MAP}(\boldsymbol{u}) = \arg\min_{\lambda(\boldsymbol{u})} \left[\frac{1}{2\nu^2}\,\rho\big([D_{ij\mathcal{F}}(\lambda)](\boldsymbol{u}) - I_{i\mathcal{F}}(\boldsymbol{u})\big) + \frac{\mu^r}{r}\sum_{\boldsymbol{u}_l\in\mathcal{N}(\boldsymbol{u})} b_l|\lambda(\boldsymbol{u}) - \lambda_{\boldsymbol{u}_l}|^r\right] \tag{4.15}$$

where $\mathcal{N}(\boldsymbol{u})$ is the neighborhood of $\boldsymbol{u}$.

4.6.2 Optimization

For $r = 2$, the depth prior is a Gaussian. In that case, a solution for equation 4.15 can be computed by searching for a zero in the derivative of the cost function. The cost function may be analytically differentiated with respect to $\lambda(\boldsymbol{u})$ to yield the expression

$$\frac{1}{2\nu^2}\,\dot{\rho}\big(e_p(\boldsymbol{u})\big)\frac{\partial[D_{ij\mathcal{F}}(\lambda)](\boldsymbol{u})}{\partial\lambda} + \mu^2\sum_{\boldsymbol{u}_l\in\mathcal{N}(\boldsymbol{u})} b_l\big(\lambda(\boldsymbol{u}) - \lambda_{\boldsymbol{u}_l}\big) = 0$$

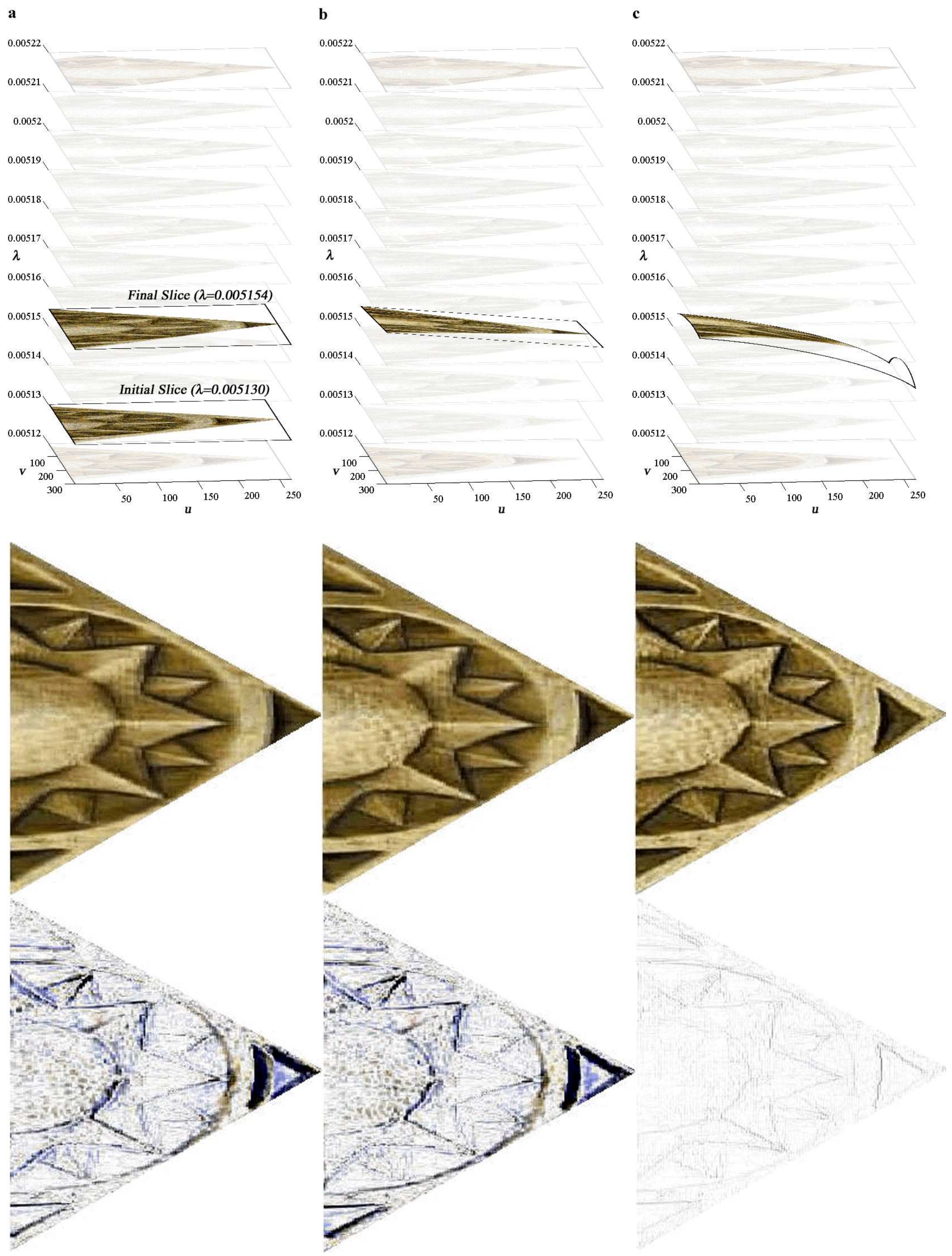


Fig. 4.10. (a) Depth Optimization. The initial slice of the depth map is a plane at a fixed depth $\lambda = 0.005130$. Optimizing for depth we obtain the final slice at $\lambda = 0.005154$. The final sliced image area is shown under the isodepth map. The difference of the image area from Image 1 and the final slice is on the bottom row. (b) Orientation Optimization. Starting from the final slice of the isodepth map in (a), we optimize for orientation of the plane that best approximates the curved patch. Points on the final plane are not at the same depth any more. (c) Shape Optimization. Points on the oriented plane are allowed to move out of the plane. However, the points are constrained such that the resulting slice is smooth. The image area corresponding to the final slice and the pixel error are shown underneath.

The ICM algorithm is implemented by sequentially updating depth at each pixel coordinate. The depth values in the neighborhood $\mathcal{N}(\boldsymbol{u})$ of pixel coordinate $\boldsymbol{u}$ are used to update the depth $\hat{\lambda}(\boldsymbol{u})$ by minimizing the MAP cost function.

$$\hat{\lambda}(\boldsymbol{u}) = \frac{-\frac{1}{2\nu^2\mu^2}\,\dot{\rho}\big(e_p(\boldsymbol{u})\big)\frac{\partial[D_{ij\mathcal{F}}(\lambda)](\boldsymbol{u})}{\partial\lambda}\big|_{\lambda(\boldsymbol{u})} + \sum_{\boldsymbol{u}_l\in\mathcal{N}(\boldsymbol{u})} b_l\lambda_{\boldsymbol{u}_l}}{\sum_{\boldsymbol{u}_l\in\mathcal{N}(\boldsymbol{u})} b_l} \tag{4.16}$$

As seen above, $\log\big(P(\lambda)\big)$—the log of depth prior—is quadratic if the prior is Gaussian. If the likelihood is also Gaussian, that is if $\rho(e_p) = e_p^2$, the MAP estimate corresponds to the minimum mean-squared-error estimate.

If the depth prior is modeled using a non-convex function, local maxima will exist for the MAP cost function. The cost function may be non-convex unless relatively low weight is applied to the prior portion of the cost. If the total cost function is non-convex, the global optimization required in the MAP estimation cannot be exactly computed. In addition to the difficulty in computing MAP estimates for non-convex priors, the solutions are also inherently discontinuous with respect to the data. If $\rho(e_p)$ is a convex function of e_p, a strictly convex $\log\big(P(\lambda)\big)$ will insure strict convexity of $P(\lambda|I_{i\mathcal{F}})$ in λ. Therefore, strict convexity of $|\lambda(\boldsymbol{u}) - \lambda_{\boldsymbol{u}_l}|^r$ will generally insure stability of the MAP estimate. For $r = 1$, the depth prior is Laplacian and its corresponding log component in the cost function are not strictly convex. The MAP estimate in such a case is not guaranteed to be well posed and stable. Moreover, the absolute-value cost component is not differentiable and we cannot compute the derivative of the cost function with respect to λ. The MAP estimate in this case is

$$\hat{\lambda}_{MAP}(\boldsymbol{u}) = \arg\min_{\lambda(\boldsymbol{u})}\left[\frac{1}{2\nu^2}\,\rho\big([D_{ij\mathcal{F}}(\lambda)](\boldsymbol{u}) - I_{i\mathcal{F}}(\boldsymbol{u})\big) + \mu\sum_{\boldsymbol{u}_l\in\mathcal{N}(\boldsymbol{u})} b_l|\lambda(\boldsymbol{u}) - \lambda_{\boldsymbol{u}_l}|\right] \tag{4.17}$$

The log of depth prior $\sum_{\boldsymbol{u}_l\in\mathcal{N}(\boldsymbol{u})} b_l|\lambda(\boldsymbol{u})-\lambda_{\boldsymbol{u}_l}|$ achieves a minimum value at the weighted median of the neighboring depth values. The weighted medium is the value, $\hat{\lambda}_D$, for which the total weight of depth values greater than $\hat{\lambda}_D$ is close to the total weight of depth values less than $\hat{\lambda}_D$. However, the log likelihood component of the cost function also depends on depth and this renders the numerical minimization difficult.

While the mean is associated with the Gaussian distribution that often appears naturally in practical problems, the median is related to the Laplacian distribution, which has heavier tails and can often provide a sharper feature. In other words, when $r = 1$ a slice with a sharp feature is no more costly than a smooth slice. Squaring the difference of neighboring depth points applies too high a penalty to sudden changes in depth or sharp features whereas an absolute-value potential function does not distinguish between a sudden change or a slow graduation of depth within a patch. Nevertheless, estimating the MAP slice of an isodepth map using an absolute-value function is difficult because local operations of updating pixel depths generally do not converge to the global MAP estimate. The log depth prior $\sum_{\boldsymbol{u}_l \in \mathcal{N}(\boldsymbol{u})} b_l |\lambda(\boldsymbol{u}) - \lambda_{\boldsymbol{u}_l}|$ has discontinuities and local operations get stuck at induced peak optima or the non-differentiable edges. The slice using a Gaussian prior ($r = 2$) suffers from smoothing of edges as an effect of noise suppression. Using $r = 1$ can describe sharp surface boundaries but at the expense of failing to attain global minimum and presence of substantial noise artifacts. This problem can be solved by carefully choosing a surface patch for modeling using an isodepth map such that the surface patch does not have sharp features.

When $1 < r < 2$, the depth prior and therefore, the cost function are differentiable. Differentiating equation 4.15, we have

$$\frac{1}{2\nu^2}\,\dot{\rho}\big(e_p(\boldsymbol{u})\big)\frac{\partial[D_{ij\mathcal{F}}(\lambda)](\boldsymbol{u})}{\partial\lambda} + \mu^r \sum_{\boldsymbol{u}_l \in \mathcal{N}(\boldsymbol{u})} \operatorname{sign}\left[\lambda(\boldsymbol{u}) - \lambda_{\boldsymbol{u}_l}\right] b_l \big|\lambda(\boldsymbol{u}) - \lambda_{\boldsymbol{u}_l}\big|^{r-1} = 0 \tag{4.18}$$

The root of equation 4.18 can be determined using the secant method or *regula falsi* also called the secant method. It is computationally inexpensive since the neighborhood $\mathcal{N}(\boldsymbol{u})$ of point $\boldsymbol{u}$ typically contains only a few pixels. In addition, the derivative of the isodepth map $\partial[D_{ij\mathcal{F}}(\lambda)](\boldsymbol{u})/\partial\lambda$ can be calculated once and stored in a lookup table. A full ICM iteration consists of applying a depth update at each pixel coordinate $\boldsymbol{u}$ in the isodepth map.

The selection of the weights b_l and the scale parameter μ control the relative influence of the prior and that of the likelihood. If μ is too large, the prior will have an over-smoothing effect on the slice. Conversely, if it is too small the MAP estimate may be unstable, reducing to the ML solution as μ tends to zero. The depth prior used in our algorithm included the 8 pixel-coordinate neighbors, with $b_l = 0.1036$ for the nearest vertical and horizontal neighbors on the pixel plane and $b_l = 0.1464$ for the diagonal neighbors. For the depth prior, we varied r from 1.6 to 2.0. The curved surface whose isodepth map is shown in Figure 4.10 was modeled using $r = 1.8$. The slice of the isodepth map shown in the same figure converged to the MAP estimate in 16 iterations.

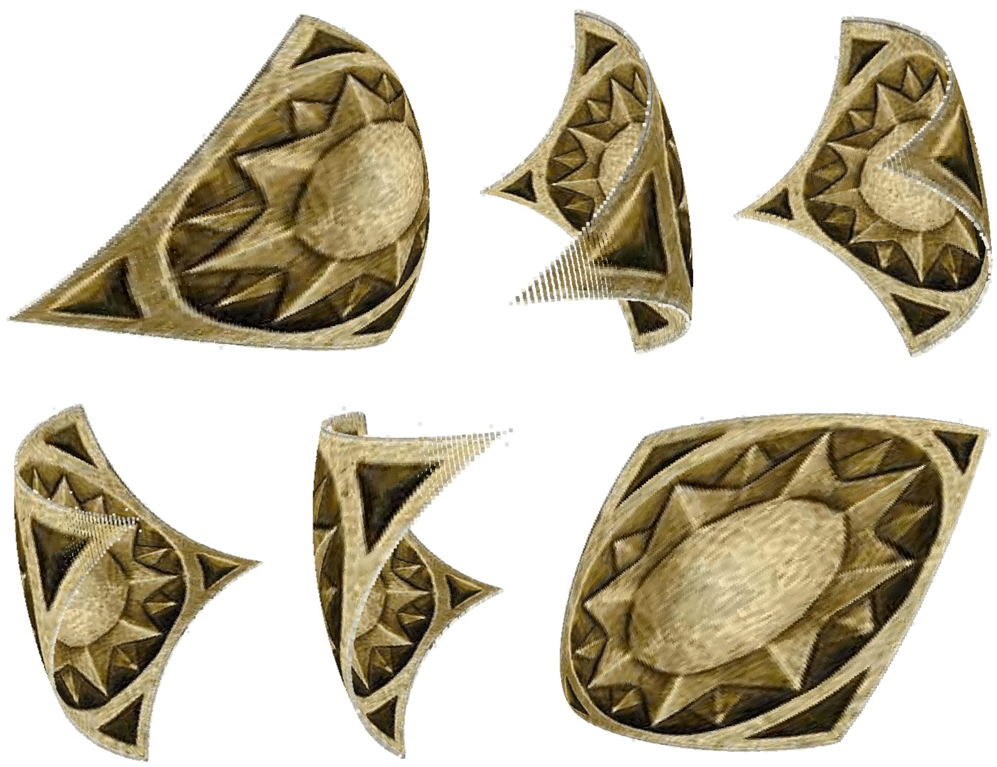


Fig. 4.11. Image renderings of the reconstructed surface patch.

3-D depth points carved out of the isodepth map for two patches $X_A X_B X_C$ and $X_A X_C X_D$ have been shown in Figure 4.11. The two patches were individually modeled using a Gaussian likelihood and a GGMRF depth prior with $r = 1.8$. The only discrepancies in the model are the edge effects where pixels of the background creep into the modeled surface and secondly, the slight overlap of two sets of 3-D points where the patches intersect.

4.6.3 The Texture Prior

The slice of an isodepth map gives us depth values as well as a set of intensity values associated with the depth values. The set of intensity values forms the texture of the patch. Depth values of points on the patch, $\boldsymbol{\lambda}$, have corresponding pixel intensities that constitute the texture of the patch, $\boldsymbol{\mathcal{T}}$. Incorporating the texture of the patch along with its depth, the objective is to maximize the probability of $P(\boldsymbol{\lambda}, \boldsymbol{\mathcal{T}} | I_{i\mathcal{F}})$. For a given image area $I_{i\mathcal{F}}$, Bayes rule is used to calculate the *a posteriori* probability of the depth slice $\boldsymbol{\lambda}$

and its associated texture $\boldsymbol{\mathcal{T}}$ as in equation 4.10;

$$\begin{aligned} P(\boldsymbol{\lambda}, \boldsymbol{\mathcal{T}} | I_{i\mathcal{F}}) &\propto P(I_{i\mathcal{F}} | \boldsymbol{\lambda}, \boldsymbol{\mathcal{T}}) P(\boldsymbol{\lambda}, \boldsymbol{\mathcal{T}}) \\ &= P(I_{i\mathcal{F}} | \boldsymbol{\lambda}, \boldsymbol{\mathcal{T}}) P(\boldsymbol{\lambda} | \boldsymbol{\mathcal{T}}) P(\boldsymbol{\mathcal{T}}) \end{aligned} \tag{4.19}$$

$P(\boldsymbol{\mathcal{T}})$ is the prior on texture. This prior encapsulates local textural properties within the patch and can be derived from learnt appearances of natural surface patches.

Given a small image pixel neighborhood around any particular pixel, we can learn a distribution for the pixel's intensity value by using pixel values at the centers of similar neighborhoods from a texture database. Each pixel $\boldsymbol{u}$ in the isodepth map has a neighborhood region $\mathcal{N}(\boldsymbol{u})$ consisting of pixel coordinates around it, but not itself. For the textural pattern, $[D_{ij\mathcal{F}}(\lambda)](\boldsymbol{u})|_{\mathcal{N}(\boldsymbol{u})}$, corresponding to the neighborhood $\mathcal{N}(\boldsymbol{u})$ in the depth map, we find the region in the set of all sampled textures that is closest to $[D_{ij\mathcal{F}}(\lambda)](\boldsymbol{u})|_{\mathcal{N}(\boldsymbol{u})}$. This texture, $\mathcal{T}$, is the one that is most similar to the textural pattern at $\mathcal{N}(\boldsymbol{u})$. Let us assume that the textural pattern $\mathcal{T}$ has been determined and the central pixel of $\mathcal{T}$ is $\boldsymbol{v}_c$. The pixel intensity at point $\boldsymbol{u}$ for depth λ is then assumed to be Gaussian distributed with mean equal to the intensity of central pixel $\mathcal{T}(\boldsymbol{v}_c)$. The pixel intensity at point $\boldsymbol{u}$, $[D_{ij\mathcal{F}}(\lambda)](\boldsymbol{u})$, is determined by the depth λ of that point in the depth map. The texture prior is now defined as

$$P(\mathcal{T}) = exp\left\{ -\mu_{\mathcal{T}} \left([D_{ij\mathcal{F}}(\lambda)](\boldsymbol{u}) - \mathcal{T}(\boldsymbol{v}_c) \right)^2 \right\} \tag{4.20}$$

where $\mathcal{T}$ is the textural pattern that is closest to the texture corresponding to $\mathcal{N}(\boldsymbol{u})$ and $\mu_{\mathcal{T}}$ is a normalizing constant.

How does one find out the texture that is most similar to the textural pattern of the patch surrounding the 3-D point which corresponds to the pixel $\boldsymbol{u}$? The texture prior should be independent of the window size for determining the texture in the point's neighborhood $\mathcal{N}(\boldsymbol{u})$. Any texture classification scheme should produce the same prior when applied to a small or large window with both windows centered at the same pixel. For a given texture database $\mathbb{T} = \{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_N\}$, we would like to find the texture $\mathcal{T}$ that is most similar to the textural pattern of the neighborhoods patch $\mathcal{N}(\boldsymbol{u})$. The set of points of the neighborhood patch $\mathcal{N}(\boldsymbol{u})$ includes all points in the window but excludes the very point at $\boldsymbol{u}$. This texture similarity is based on some perceptual distance d between the two textures–one that belongs to the pixel's neighborhood and the other $\mathcal{T}$ that is selected from the database of common textures. This idea of describing the texture of the pixel's neighborhood is similar to the texture synthesis problem (Efros and Leung, 1999).

$$\mathcal{T} = \underset{\mathcal{T}_k \in \mathbb{T}}{\arg\min}\ d\big(\ [D_{ij\mathcal{F}}(\lambda)](\boldsymbol{u})|_{\mathcal{N}(\boldsymbol{u})},\ \mathcal{T}_k\ \big) \tag{4.21}$$

One choice for d is the normalized sum of squared differences metric. The neighborhood $\mathcal{N}(\boldsymbol{u})$ of a pixel at $\boldsymbol{u}$ is modeled as a square window around that pixel coordinate. For a larger neighborhood $\mathcal{N}(\boldsymbol{u})$, the weight of differences of nearby pixel coordinates should be more than that of coordinates farther away from $\boldsymbol{u}$. The pixel intensity differences can be weighted by a two-dimensional Gaussian kernel centered at $\boldsymbol{u}$. An ideal source of the texture database $\mathbb{T}$ would be a large number of patches of natural images. The texture library could also be build of patches from the given images (Wexler et al., 2003) provided they are taken from varied viewpoints. In our case, we used image patches of several natural scenes as well as the input images to generate a database of common textures. The window size in all our implementations is 7×7 pixels.

4.6.4 Depth-from-Texture Prior

$P(\lambda|\mathcal{T})$ is the prior on depth for a given texture pattern in a neighborhood. This encodes the depth value at point $\boldsymbol{u}$ when the most similar texture pattern of points in its neighborhood $\mathcal{N}(\boldsymbol{u})$ is known. In the previous section we determined this texture $\mathcal{T}$ by considering the pixel intensities of the neighboring points in the window but excluding the intensity at point $\boldsymbol{u}$. The texture $\mathcal{T}$ determines the set of depth values that point $\boldsymbol{u}$ can take. This is on the lines of, but not identical to, the ideas used in many shape-from-texture algorithms such as vanishing-point perspective, surface markings, texture flows, spatial frequency modulations, affine deformations of textural pattern, foreshortening, or locally correct perspective cues. A new class of algorithms argue that texture elements are able to reveal the underlying shape of an object even though the texture itself lacks a pattern in any particular direction (Kim et al., 2004; Li and Zaidi, 2003; Zaidi and Li, 2002). Our approach to determine depth values for points of a surface patch, which collectively form shape, is based on the similar philosophy that texture of the patch determines the range of depth values these points can have. In other words, the value of r and μ_λ in equation 4.12 is determined by the texture of the neighborhood $\mathcal{N}(\boldsymbol{u})$.

How are the parameters μ_λ and r learnt for each texture in the database? Consider the prior term

$$P_{\mu_\lambda, r}(\lambda|\mathcal{T}) = \mu_\lambda^M exp\left\{ -\frac{\mu_\lambda^r}{r} \sum_{\{\boldsymbol{u}_k, \boldsymbol{u}_l\} \in \mathcal{N}} b_{kl} |\lambda_{\boldsymbol{u}_k} - \lambda_{\boldsymbol{u}_l}|^r \right\} \tag{4.22}$$

where M is the number of pixels in the neighborhood $\mathcal{N}$. We first derive the ML estimate of the parameter μ_λ. The normalized log-likelihood can be computed from equation 4.22 as

$$\log P_{\mu_\lambda, r} = -\frac{\mu_\lambda^r}{r} \sum_{\{\boldsymbol{u}_k, \boldsymbol{u}_l\} \in \mathcal{N}} b_{kl} |\lambda_{\boldsymbol{u}_k} - \lambda_{\boldsymbol{u}_l}|^r + M \log \mu_\lambda \tag{4.23}$$

Differentiating 4.23 with respect to $\mu_l ambda$ and equating it to zero yields the equation for $\hat{\mu_\lambda}$, the ML estimate of μ_λ;

$$\hat{\mu_\lambda}^r = \frac{M}{\sum_{\{\boldsymbol{u}_k, \boldsymbol{u}_l\} \in \mathcal{N}} b_{kl} |\lambda_{\boldsymbol{u}_k} - \lambda_{\boldsymbol{u}_l}|^r} \tag{4.24}$$

This is a very simple but interesting form; intuitively, consider the case where $\boldsymbol{\lambda}$ are i.i.d Gaussian random variables. For this case, $r = 2$, μ_λ^2 is the inverse of variance, and equation 4.24 reduces to $M / \sum_{k=1}^{M} \lambda_{\boldsymbol{u}_k}^2$. Substituting the value of μ_λ from equation 4.24 into 4.23 we have:

$$\log P_{\mu_\lambda, r} = -\frac{M}{r} - \frac{M}{r} \log \frac{\sum_{\{\boldsymbol{u}_k, \boldsymbol{u}_l\} \in \mathcal{N}} b_{kl} |\lambda_{\boldsymbol{u}_k} - \lambda_{\boldsymbol{u}_l}|^r}{M} \tag{4.25}$$

The ML estimate of r is then given by

$$\hat{r} = \arg \min_{r \in [1,\, 2]} \left\{ \frac{M}{r} + \frac{M}{r} \log \frac{\sum_{\{\boldsymbol{u}_k, \boldsymbol{u}_l\} \in \mathcal{N}} b_{kl} |\lambda_{\boldsymbol{u}_k} - \lambda_{\boldsymbol{u}_l}|^r}{M} \right\} \tag{4.26}$$

The above can be easily computed for a texture provided the variation of a point's height with respect to its neighbors is known. This falls in the domain of 3-D textures (Dana and Nayar, 1998, 1999; van Ginneken et al., 1999).

The values $\hat{\mu}_\lambda$ and $\hat{r}$ are thus learnt for each texture $\mathcal{T}$ in $\mathbb{T}$. A few textures from the Columbia-Utrecht Reflectance and Texture (CUReT) Database (Dana et al., 1997a,b, 1999) are shown in Figure 15. The depth-from-texture prior term is given by

$$P(\lambda|\mathcal{T}) = exp \left\{ -\frac{\hat{\mu_\lambda}^{\hat{r}}}{\hat{r}} \sum_{\boldsymbol{u}_l \in \mathcal{N}(\boldsymbol{u})} b_l |\lambda(\boldsymbol{u}) - \lambda_{\boldsymbol{u}_l}|^{\hat{r}} \right\} \tag{4.27}$$

Combining both priors, the minimization of the cost function at each pixel coordinate $\boldsymbol{u}$ in the isodepth map with respect to the depth value at that coordinate, $\lambda(\boldsymbol{u})$, is

$$\hat{\lambda}_{MAP}(\boldsymbol{u}) = \arg \min_{\lambda(\boldsymbol{u})} \Bigg[\frac{1}{2\nu^2} \rho\big([D_{ij\mathcal{F}}(\lambda)](\boldsymbol{u}) - I_{i\mathcal{F}}(\boldsymbol{u})\big) + \\ \hat{\mu}_\lambda^{\hat{r}} \sum_{\boldsymbol{u}_l \in \mathcal{N}(\boldsymbol{u})} b_l |\lambda(\boldsymbol{u}) - \lambda_{\boldsymbol{u}_l}|^{\hat{r}} + \mu_{\mathcal{T}} \left([D_{ij\mathcal{F}}(\lambda)](\boldsymbol{u}) - \mathcal{T}(\boldsymbol{v}_c)\right)^2 \Bigg]$$

(4.28)

The optimization algorithm iteratively minimizes the cost function at each pixel coordinate $\boldsymbol{u}$ in the depth map for the depth value at that point after determining the most similar texture $\mathcal{T}$ for its neighborhood $\mathcal{N}(\boldsymbol{u})$. With the texture $\mathcal{T}$ chosen from the database $\mathbb{T}$ is associated a set of values $\{\hat{\mu}_\lambda, \hat{r}\}$ that are learnt. $\{\hat{\mu}_\lambda, \hat{r}\}$ values for 61 textures of the CUReT database are shown in Table 6.1. A few textures are shown in Figure 15. The learning can also be done using depth and texture data that is available from dense cloud of range points or pre-determined textured models of natural scenes.

4.6.5 Algorithm Summary

The complete algorithm can be summarized as follows:

- For fixed number of iterations or until convergence,
  - Consider each face $\mathcal{F}$ of mesh and for the pair of images i and j:
    - Construct isodepth maps $D_{ij\mathcal{F}}$ and $D_{ji\mathcal{F}}$.
    - For every pixel coordinate $\boldsymbol{u}$ in the isodepth maps, consider its neighborhood $\mathcal{N}(\boldsymbol{u})$.
    - Find texture $\mathcal{T}$ from texture data base that is most similar to pixel intensity pattern carved out by $\mathcal{N}(\boldsymbol{u})$ in isodepth maps and calculate texture prior.
    - Use the learnt parameters $\{\hat{\mu_\lambda}, \hat{r}\}$ to compute shape-from-texture prior component of cost function.
    - Calculate likelihood component of cost function that enforces image coherence.
    - Iterate for all pixel coordinates within face $\mathcal{F}$.
  - Repeat for all faces of mesh.
- Iterate for whole mesh again.

5 Image Coherence

Depth coherence is also sensitive to discontinuities or sudden changes in curvature within the patch. Depth coherence models a surface patch accurately only if the region of interest is small enough such that the depth variation within the patch is small and does not include sharp features. The absolute-value log depth prior $\sum_{\boldsymbol{u}_l \in \mathcal{N}(\boldsymbol{u})} b_l |\lambda(\boldsymbol{u}) - \lambda_{\boldsymbol{u}_l}|$, as explained in the last chapter, can be used to model sudden changes, sharp features or discontinuities within a patch. However, the MAP estimate for that prior is not stable and any local pixel-based operation does not converge to the global MAP estimate. Therefore, the 3-D surface patch being modeled should be close to its planar approximation that is used as an initial estimate of structure. Any discontinuities in the surface structure has to be modeled by choosing the patches such that the surface discontinuities are at the edges of patches. In other words, the surface has to be segmented such that individual patches have their edges aligned along the discontinuous edges of the surface. This allows us to start with a good guess of the structure or a planar approximation of each 3-D surface patch.

In many graphics and vision applications, a surface is often modeled as a triangulated mesh of 3-D points and textures associated with faces of the mesh. The 3-D points could be either sampled from range data or derived from a set of images using a stereo or Structure-from-Motion algorithm. When the points do not lie at critical points of maximum curvature or discontinuities of the real surface, faces of the mesh do not lie close to the modeled surface. This results in textural artifacts, and the model is not perfectly coherent with a set of actual images—the ones that are used to texture-map its mesh. In order to apply depth coherence to refine a model, we need to start with a mesh that is the best possible planar approximation of the surface. This can be achieved by enforcing image coherence prior to depth coherence.

In this chapter, we present a technique for perfecting the 3-D surface model by repositioning its vertices so that it is coherent with a set of observed images of the object. The textural artifacts and incoherence with images are due to the non-planarity of a surface

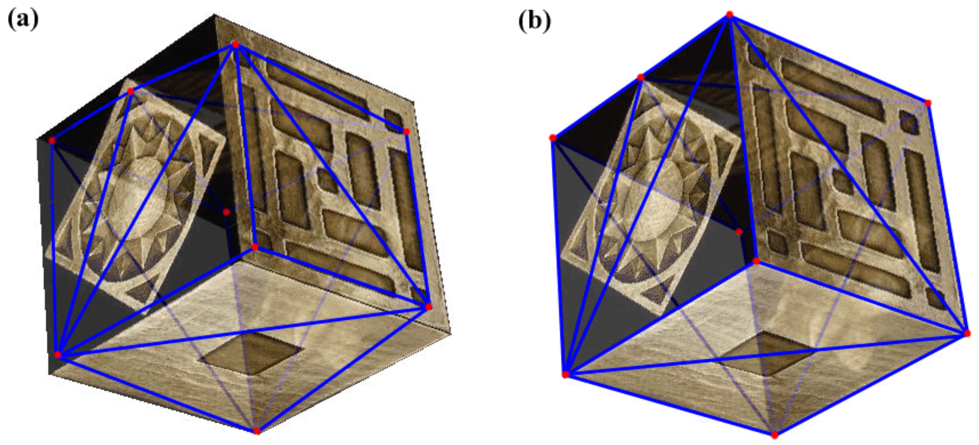


Fig. 5.1. (a) 3-D points used for creating the model do not lie at geometrical extremities or points of discontinuities of the cube's surface. Faces of the mesh with such 3-D points as vertices are poor approximation of the real surface. (b) For the new vertex positions, the faces of the mesh lie on the surface of the cube, thus creating a good model.

patch being approximated by a planar face, as observed from multiple viewpoints. Image areas from the viewpoints are used to represent texture for the patch in eigenspace. The eigenspace representation captures variations of texture, which we seek to minimize. This refinement of a segmentation of the surface corresponds to moving along the discontinuous edges of the log likelihood function.

An image coherence measure based on the difference between the face textures reconstructed from eigenspace and the actual images is used to reposition the vertices so that the model is improved or faired. We refer to this technique of model refinement as EigenFairing, by which the model is faired, both geometrically and texturally, to better approximate the real surface. The resulting surface approximation is ideal for enforcing depth coherence afterwords.

5.1 Image Consistent Modeling

A model of a real surface is often represented as a mesh of 3-D points. The 3-D points could be sampled from range data or derived from a set of images using a Stereo or SfM algorithm. The points form vertices of a triangulated mesh, faces of which are texture-mapped to model the surface. This representation has proven to be effective, as a significant number of surfaces can be modeled by a mesh of planar faces. Most sensed 3-D data, however, do not always lie at the discontinuities, such as corners and

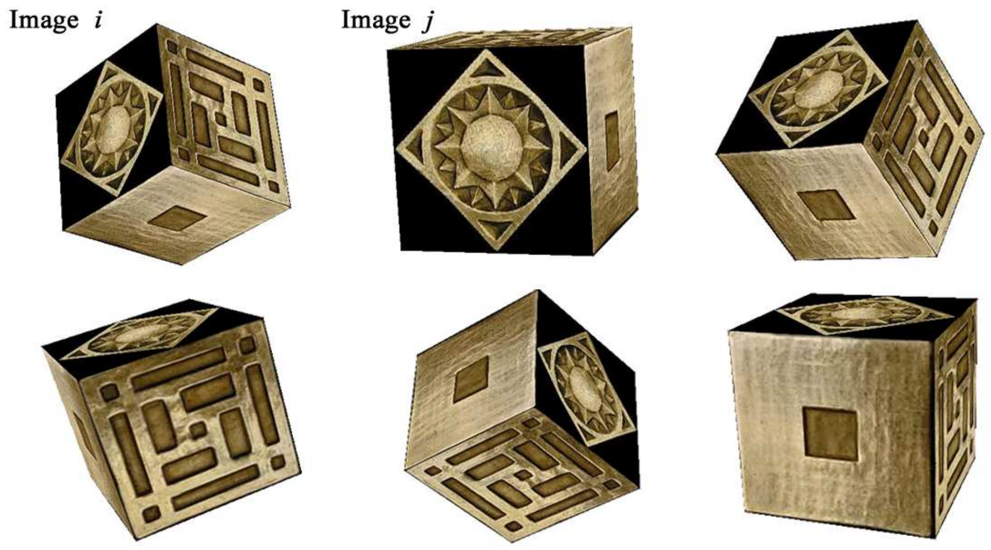


Fig. 5.2. Images capture the view-dependency of a surface. Textures across a surface patch depend on the positions of point features.

edges of the real surface[Figure 5.1]. Laser scanned data misses out on these strategic but spatially minute points. Features selected and tracked using local pixel intensities in images for stereo or SfM algorithms do not always correspond to points of discontinuity of the observed surface. They might not be visually distinct as well. This fact leads to a poor approximation of the surface by faces of the mesh. The model, as represented by the mesh, can be refined or *faired* by relocating the vertices, such that the planar faces lie closer to patches of the real surface they approximate. We refer to this process of refining the model as *fairing*.

What information could be used to *fair* the model? The real surface is unknown to us, but we have images of the same. Images are used to texture-map faces of the mesh to generate a 3-D model. The albedo or texture corresponding to the surface patch is being viewed in a set of images. The mesh face that approximates the surface patch can be re-projected back into the images to yield triangular image areas. A set of these image areas corresponding to the same surface patch have information of how good its planar approximation is, and can be used for fairing the model.

Had the surface patch been planar, the image areas corresponding to its planar approximation would be the same when warped into a fixed triangular area [Figures 5.3 and 5.4]. As the real world is not built of simplicial elements, the variations of texture in the image areas can be attributed to the non-planarity of the surface patch. We would

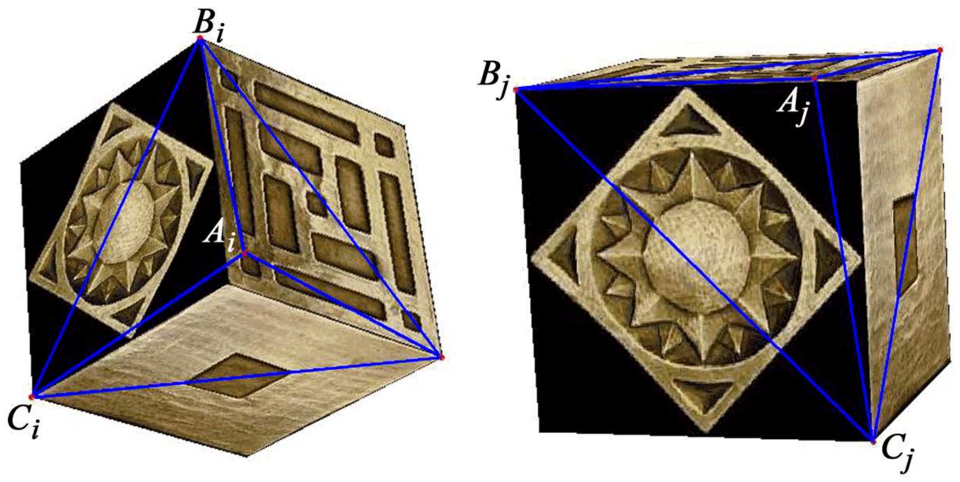


Fig. 5.3. The triangular image patches, corresponding to a single mesh face, depend on the degree of non-planarity of the actual 3-D-surface patch from the viewpoints. In the above Image patch $A_iB_iC_i$ in the i-th image and $A_jB_jC_j$ in the j-th image correspond to mesh face ABC. The image patches differ because A does not lie on an edge or corner, and the mesh face does not lie on the actual 3-D surface of the object.

like to get the best possible representation of texture from the image areas by encoding the textural variations.

Given a set of image areas, a compact representation of these variations can be derived in terms of a small number of orthogonal basis images. This representation, also called an eigenspace decomposition, encodes all variations of the patch texture, as observed from different viewpoints Kurazume et al. (2002) Nishino et al. (2001).

We reposition the vertices of the mesh such that these textural variations are minimized. In the process, the faces of the mesh better approximate the real surface in the geometrical sense. Also, the texture-mapped 3-D mesh, that is the 3-D model, appears to be as close as possible to the actual images when observed from the same camera viewpoints. This *coherence* between the actual image area and the textural representation of the corresponding surface patch can be quantified as the distance-from-eigenspace of the image area. The better a triangular face approximates the surface patch geometrically, the smaller is the variation of appearance in images, and the smaller is the distance-from-eigenspace of the image-areas corresponding to the patch. We refer to this idea of 3-D model refinement as *EigenFairing*.

5.2 Refinement of Model Description

The Eigentexture method Kurazume et al. (2002) encodes appearance variations on the same physical patch of an object under various viewing conditions. It compresses appearances of a patch by using the eigenspace method and renders new views of a patch from its eigenspace representation. View-based representation in eigenspace has also been used in tracking rigid and articulated objects Black and Jepson (1998). It has been shown that for diffuse surfaces of arbitrary texture, the first five components of eigenspace explain most of the image variation Epstein et al. (1995). We will use an eigenspace representation of dimensionality five to refine the mesh that approximates a 3-D surface.

The problem of model improvement, given a set of 3-D range data of an object, has been studied extensively. In the absence of multiple images, mesh simplification from a single image Garland and Heckbert (1998) has been used to reposition vertices of a texture-mapped mesh by minimizing error metrics based on the vertex positions, surface normals, and other surface properties such as color and texture. The mesh simplification algorithm never accesses the pixels of the single-image texture and merely updates the texture coordinates in the image. On the other hand, image-driven mesh simplification Lindstrom and Turk (2000) uses multiple images to decide which portions of a model to simplify through the edge collapse operator. It compares images of an original model against those of a simplified model using the root-mean-squared difference of luminance channels to determine the cost of an edge collapse.

Image-consistent surface triangulation Morris and Kanade (2000) searches for the best triangulation among many possible candidates that best explains observed images. The initial mesh of a point-cloud is refined through edge-swaps to best account for images of the same object. The surface is modeled by a texture-mapped mesh and re-projected onto the images to yield the re-projection error. This error is minimized over swaps of edges of the mesh. In absence of the knowledge of the true surface, the mesh faces are texture-mapped by affine warping of triangles instead of perspectively projecting the images onto the surface. This simplification works only if the surface patch has enough texture and, at the same time, is close to the planar mesh face that approximates it. In areas of little or no texture, swapping edges often produces complicated overlapping mesh faces.

The planarity of a surface patch as compared to a mesh face, i.e. perspective distortion, depends on the position of the vertices of the mesh. 3-D points obtained from image features using SfM or stereo do not always correspond to physical corners or edges.

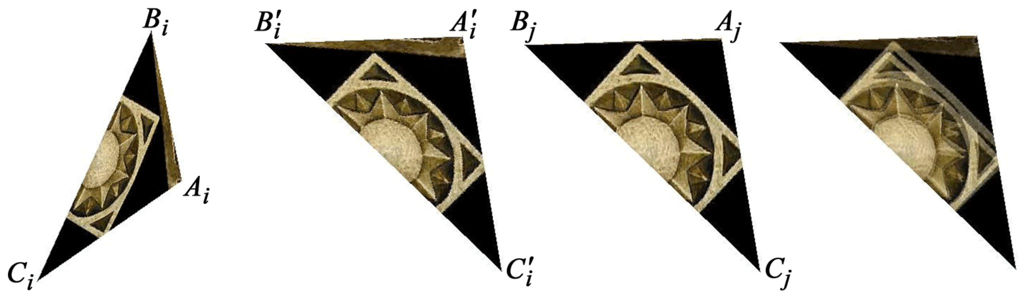


Fig. 5.4. Image patches $A_iB_iC_i$ and $A_jB_jC_j$ in two images correspond to the same 3-D mesh face. Affine warping of $A_iB_iC_i$ into $A_jB_jC_j$ yields image patch $A'_iB'_jC'_j$. The texture within the two patches, $A'_iB'_iC'_i$ and $A_jB_jC_j$, are not the same. Note the shadowy edges in their superimposition (average) on the extreme right.

Scanning the surface structure by a range scanner has very few 3-D points corresponding to geometrical extremities. Reed and Allen (1999) have used range scans from multiple views to model objects at their extremeties. One of the critical components of model refinement that is missing from image-based object modeling is to move the vertex points so that they lie at the extremities of object geometry.

5.3 Image Coherence

If the surface patch being approximated by the mesh face is not planar and has perspective distortion, how can texture of the face be best represented, given the triangular pixel-areas in multiple images corresponding to it? The triangular image patches depend on viewpoints of the camera as well as degree of non-planarity of the 3-D-surface patch [Figure 5.3]. If the 3-D-surface patch is non-planar, affine warping of triangular image patches onto each other does not align the texture within [Figure 5.4]. Estimating texture corresponding to the mesh face that approximates the 3-D-surface patch, by weighted-average of pixels from different images Morris and Kanade (2000), leads to blurring of the estimated texture. Since image-consistency is sensitive to textural variations, averaging of affine-warped image areas leads to complicated overlapping surfaces and chances of the refinement algorithm being trapped in local minima. Moreover, textural variations of image patches corresponding to the same 3-D-surface patch can be exploited for refining the mesh. Image coherence considers this *intra-image* textural variation, as well as consistency of estimated texture, when compared to actual images.

For a given set of image areas corresponding to a single 3-D-surface patch, image coherence constructs a small set of basis images that best captures the variations in tex-

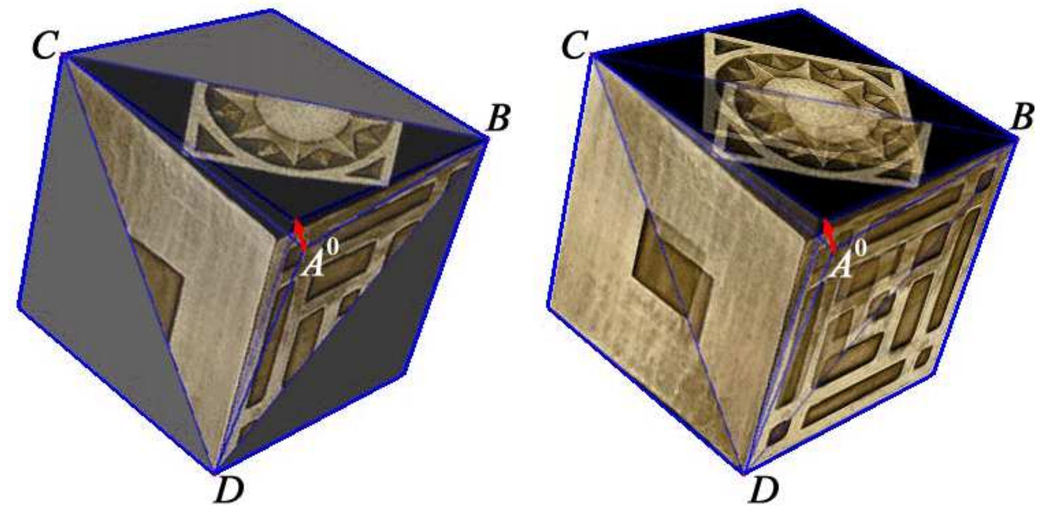


Fig. 5.5.

ture [Figure 5.6]. These basis images form a view-based representation of the texture of the patch. Each triangular image area is affine warped to a fixed triangular area called the *cell image* Nishino et al. (2001). A column vector of size $3N_{pix} \times 1$ is constructed by raster-scanning each color cell image that has N_{pix} pixels. Using these column vectors for a set of n image areas corresponding to a patch, we create a matrix ***X*** from these column vectors. Matrix ***X*** has a dimension of $3N_{pix} \times n$. The Singular Value Decomposition of ***X*** into a $3N_{pix} \times n$ orthogonal matrix ***U***, a $n \times n$ diagonal matrix Σ, and a $n \times n$ orthogonal matrix $\boldsymbol{V}^T$ can be written as

$$\boldsymbol{X} = \boldsymbol{U}\Sigma\boldsymbol{V}^T \tag{5.1}$$

The columns of ***U***, or the left singular vectors, represent the principal components of the cell images. The diagonal entries of Σ have the singular values in decreasing order and the rows of $\boldsymbol{V}^T$ or the right singular vectors have the coefficients required to extract each column of ***X*** in terms of the principal components. Although all the n dimensions are necessary to accurately represent each cell image, a few basis vectors are sufficient for reconstructing the cell images to a good degree of appearance.

Since each cell image is created by affinely warping the triangular image areas corresponding to a patch, the number of basis vectors to adequately represent its appearance depends on the planarity of the patch. This is valid, of course, assuming that the patch has Lambertian reflective properties. The more planar a patch is, the better is its approximation by the mesh face, the lower is its perspective distortion as seen from the image-

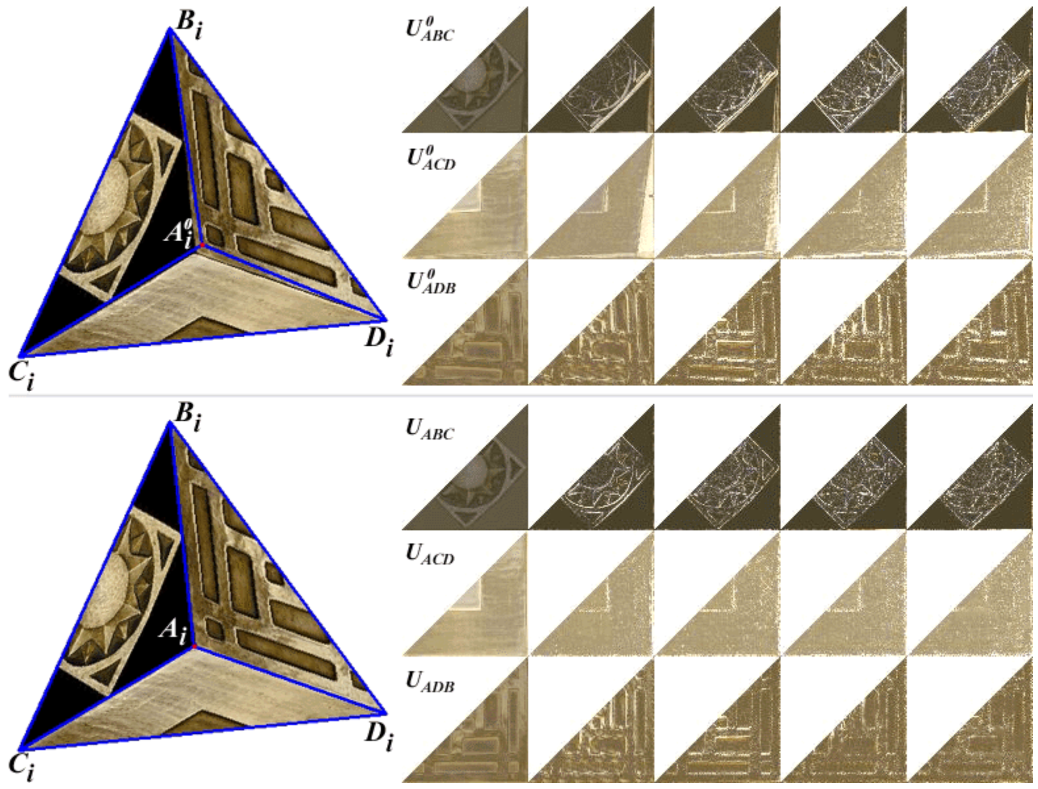


Fig. 5.6. For the position of vertex A on left, the first five components of the eigenspace representation of texture for each face are shown on the right.

viewpoints, and therefore, the lower is the number of basis vectors needed to represent the texture of the patch. Considering the first k principal components corresponding to the k largest singular values, the basis image-set is $\boldsymbol{U}_{\mathcal{F}} = [U_{\mathcal{F}1}, U_{\mathcal{F}2}, ..., U_{\mathcal{F}k}]$ for face $\mathcal{F}$. Let $U_{\mathcal{F}}\mathbf{c}_i$ denote the reconstructed cell image corresponding to the i^{th}-image patch. The set of scalar values, $\mathbf{c}_i = [c_{i1}, c_{i2}, ..., c_{ik}]$, is computed by taking the dot product of the cell image and basis images, $\boldsymbol{U}_{\mathcal{F}}$. Thus, $U_{\mathcal{F}}\mathbf{c}_i$ is constructed by a linear combination of the k basis images;

$$U_{\mathcal{F}}\mathbf{c}_i = \sum_{m=1}^{k} c_{im} U_{\mathcal{F}m} \tag{5.2}$$

An image coherent representation is the best possible approximation, $U_{\mathcal{F}}\mathbf{c}_i$ over all image patches, i.e., for $i = 1, 2, ..., n$. For a Lambertian surface, such a representation reflects the accuracy of approximation of the textured patch by the mesh geometrical element. Image coherence is based on the closeness of these two approximations: the

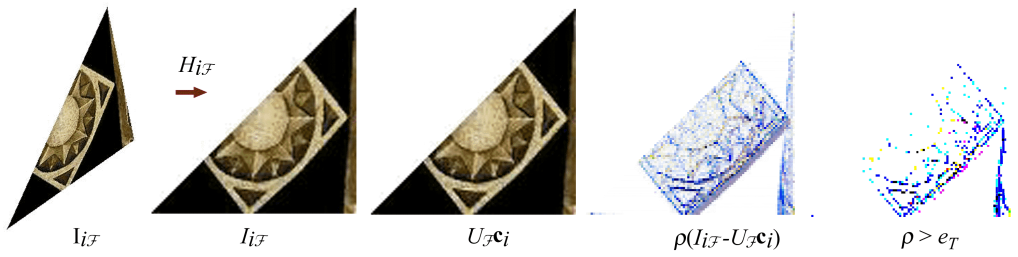

$H_{i\mathcal{F}}$
$\mathrm{I}_{i\mathcal{F}}$
$I_{i\mathcal{F}}$
$U_{\mathcal{F}}\mathbf{c}_i$
$\rho(I_{i\mathcal{F}}\text{-}U_{\mathcal{F}}\mathbf{c}_i)$
$\rho > e_T$

Fig. 5.7. The triangular image patch $\mathrm{I}_{i\mathcal{F}}$ in the i-th image corresponding to face $\mathcal{F}$ is warped to a fixed cell image $I_{i\mathcal{F}}$. The reconstructed cell image using the first five components of eigenspace is $U_{\mathcal{F}}\mathbf{c}_i$. The Geman-McClure norm between $I_{i\mathcal{F}}$ and $U_{\mathcal{F}}\mathbf{c}_i$ for σ=100, and the spatial distribution of outliers ($e_T = \sigma/\sqrt{3}$) are on the right.

textural approximation as captured by the basis images, and the geometrical approximation of the physical 3-D patch by the planar face of the mesh. An image coherent mesh refinement technique minimizes the error in the approximation of the surface patch by the linear mesh element by minimizing the error in representation of surface texture by the basis images.

5.4 An Illustration

At first, we illustrate an EigenFairing process for a *unit* cube with textures on three faces visible in 12 images. The vertex A, common to the three faces as shown in Figure 5.3, is faired such that it corresponds to the actual corner of the cube.

5.5 Model Fairing

Given a set of basis texture-images $\boldsymbol{U}_{\mathcal{F}} = [U_{\mathcal{F}1}, U_{\mathcal{F}2}, ...U_{\mathcal{F}k}]$ corresponding to a face $\mathcal{F}$, we can reconstruct the image patch, $U_{\mathcal{F}}\mathbf{c}_i$, in the i-th image. For image coherence, the objective function to be minimized for the set of image patches, $I_{i\mathcal{F}}$ is

$$E(\mathbf{c}) = \sum_{i=1}^{n} \sum_{\boldsymbol{u}_{\mathcal{F}}} \rho(I_{i\mathcal{F}}(\boldsymbol{u}) - [U_{\mathcal{F}}\mathbf{c}_i](\boldsymbol{u})) \tag{5.3}$$

for a sequence of n images and a error norm ρ defined over residual pixel-error in cell images. Note that $\boldsymbol{u}_{\mathcal{F}}$ is the set of cell pixel coordinates over which the residual pixel-error is summed. Instead of using an exhaustive search technique around each vertex in 3-D space, we can formulate an iterative search method.

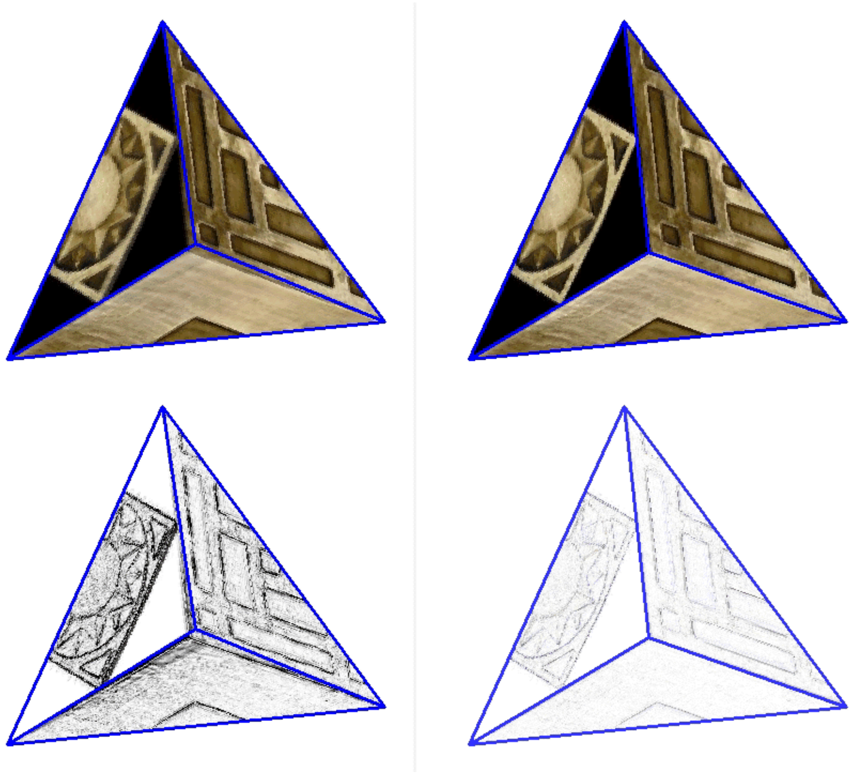


Fig. 8 a. Re-projected images of the model and re-projection error are shown above. The images on left correspond to the initial vertex position while the images on right correspond to the EigenFaired vertex.

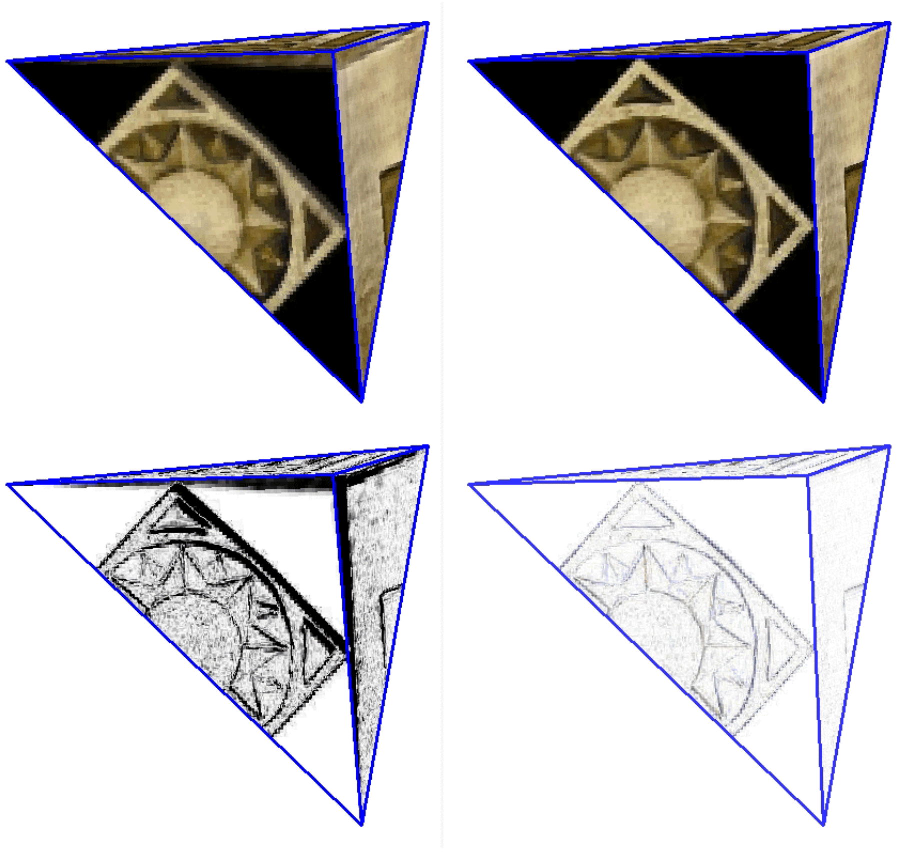


Fig. 8 b. The re-projected images and re-projection error before and after EigenFairing for another view.

5.5.1 Vertex Displacement

Let $\boldsymbol{\eta} = [\eta_X \ \eta_Y \ \eta_Z]^T$ represent a 3-D displacement of a vertex. The faired vertex point $\tilde{\mathbf{x}}_{\mathcal{V}}$ is

$$\tilde{\mathbf{x}}_{\mathcal{V}} \leftarrow \mathbf{x}_{\mathcal{V}} + \boldsymbol{\eta} \tag{5.4}$$

The goal is to simultaneously find the coefficients $\mathbf{c}$ and displacement vector, (η_X, η_Y, η_Z), that minimize the objective function of the residual error,

$$E(\mathbf{c}, \boldsymbol{\eta}) = \sum_{i=1}^{n} \sum_{\boldsymbol{u}_{\tilde{\mathcal{F}}}} \rho(I_{i\tilde{\mathcal{F}}}(\boldsymbol{u}) - [U_{\tilde{\mathcal{F}}}\mathbf{c}_i](\boldsymbol{u})) \tag{5.5}$$

for the faces, $\tilde{\mathcal{F}}$, constructed from the new vertices, $\tilde{\mathbf{x}}_{\mathcal{V}} \leftarrow \mathbf{x}_{\mathcal{V}} + \boldsymbol{\eta}$. This optimization algorithm interleaves two sub-problems. The first sub-problem is to minimize $E(\mathbf{c}, \boldsymbol{\eta})$ with respect to $\mathbf{c}$ while the vertex, $\mathbf{x}_{\mathcal{V}}$, is kept fixed. This is the same as the eigentexture method Nishino et al. (2001) discussed in the last section.

The second sub-problem is to minimize $E(\mathbf{c}, \boldsymbol{\eta})$ with respect to the *fairing* parameters or displacement $\boldsymbol{\eta}$, this time with the coefficients $\mathbf{c}$ held fixed. The image patch $I_{\mathcal{F}}$ corresponding to face $\mathcal{F}$ gets warped to $I_{\tilde{\mathcal{F}}}$ corresponding to the new face, $\tilde{\mathcal{F}}$. For a given set of basis images, $\boldsymbol{U}_{\mathcal{F}} = [U_{\mathcal{F}1}, U_{\mathcal{F}2}, ..., U_{\mathcal{F}k}]$ for face $\mathcal{F}$, we have to determine a new set of image patches and corresponding cell images, $I_{i\tilde{\mathcal{F}}}(\boldsymbol{u})$, for $i = 1, 2, ..., n$ such that the following is minimized:

$$\sum_{i=1}^{n} \sum_{\boldsymbol{u}_{\tilde{\mathcal{F}}}} \rho(I_{i\tilde{\mathcal{F}}}(\boldsymbol{u}) - [U_{\mathcal{F}}\mathbf{c}_i](\boldsymbol{u})) \tag{5.6}$$

Due to the warping of the image patch, cell pixels at $\boldsymbol{u}$ get displaced to new cell image coordinates, $\boldsymbol{u} + \upsilon_{i\mathcal{F}}(\boldsymbol{u}, \boldsymbol{\eta})$, for displacement, $\boldsymbol{\eta}$. For a given pixel-coordinate displacement function, $\upsilon_{i\mathcal{F}}(\boldsymbol{u}, \boldsymbol{\eta})$, the new cell image,

$$I_{i\tilde{\mathcal{F}}}(\boldsymbol{u}) = I_{i\mathcal{F}}(\boldsymbol{u} + \upsilon_{i\mathcal{F}}(\boldsymbol{u}, \boldsymbol{\eta})) \tag{5.7}$$

Ideally, we should have:

$$I_{i\mathcal{F}}(\boldsymbol{u} + \upsilon_{i\mathcal{F}}(\boldsymbol{u}, \boldsymbol{\eta})) = [U_{\mathcal{F}}\mathbf{c}_i](\boldsymbol{u}) \tag{5.8}$$

Equation (5.8) states that there are pixel displacements, $\upsilon_{i\mathcal{F}}(\boldsymbol{u}, \boldsymbol{\eta})$, that, when applied to the image patch $I_{i\mathcal{F}}$, make $I_{i\mathcal{F}}$ look like some image reconstructed from the eigenspace. Rewriting the left hand side of Equation (5.8) using a first order Taylor series expansion,

$$I_{i\mathcal{F}}(\boldsymbol{u}) + \nabla I_{i\mathcal{F}} \cdot \upsilon_{i\mathcal{F}}(\boldsymbol{u}, \boldsymbol{\eta}) = [U_{\mathcal{F}}\mathbf{c}_i](\boldsymbol{u}) \tag{5.9}$$

We can now define the residual pixel-error in cell images, e_c, as

$$e_c \equiv \nabla I_{i\mathcal{F}} \cdot \upsilon_{i\mathcal{F}}(\boldsymbol{u}, \boldsymbol{\eta}) + (I_{i\mathcal{F}}(\boldsymbol{u}) - [U_{\mathcal{F}}\mathbf{c}_i](\boldsymbol{u})) \tag{5.10}$$

Summing e_c over all images, and cell pixels $\boldsymbol{u}_{\mathcal{F}}$ corresponding to face $\mathcal{F}$, the error function can now be written in term of the residual error as

$$E(\mathbf{c}, \boldsymbol{\eta}) = \sum_{i=1}^{n} \sum_{\boldsymbol{u}_{\mathcal{F}}} \rho\,(\,e_c\,) \tag{5.11}$$

5.5.2 Optimization

The minimization of $E(\mathbf{c}, \boldsymbol{\eta})$ with respect to $\boldsymbol{\eta}$ can be obtained using the Gauss-Newton algorithm. In the Gauss-Newton method, a search direction is computed using the gradient, and a first-order approximation to the Hessian for the given objective function $E(\mathbf{c}, \boldsymbol{\eta})$. The k-th element of gradient vector $\mathbf{g}$ is

$$\mathbf{g}_k = \sum_{i=1}^{n} \sum_{\boldsymbol{u}_{\mathcal{F}}} \dot{\rho}\,(\,e_c\,)\;\frac{\partial e_c}{\partial \eta_k} \qquad \text{for} \qquad k \in \{X, Y, Z\}. \tag{5.12}$$

where $\dot{\rho}$, also called the influence function, is the derivative of the error norm ρ with respect to the residual pixel error, and

$$\frac{\partial e_c}{\partial \eta_k} = \left(\nabla I_{i\mathcal{F}} \cdot \frac{\partial \upsilon_{i\mathcal{F}}(\boldsymbol{u}, \boldsymbol{\eta})}{\partial \eta_k} \right)$$

In this paper, we have considered the Geman-McClure norm as our error norm ρ defined over pixel error e_c in cell images. Given a scale factor, σ, that controls the convexity of the norm, we have:

$$\rho(e_c) = \frac{e_c^2}{\sigma + e_c^2}\ , \qquad \dot{\rho}(e_c) = \frac{2\sigma e_c}{(\sigma + e_c^2)^2}\ , \text{ and } \qquad \ddot{\rho}(e_c) = \frac{2\sigma(\sigma - 3e_c^2)}{(\sigma + e_c^2)^3}$$

The $\{k, l\}$-th element of the Hessian $\mathbf{H}$ is

$$\mathbf{H}_{kl} = \sum_{i=1}^{n} \sum_{\boldsymbol{u}_{\mathcal{F}}} \ddot{\rho}(e_c) \frac{\partial e_c}{\partial \eta_k} \frac{\partial e_c}{\partial \eta_l} \qquad \text{for} \qquad k, l \in \{X, Y, Z\} \tag{5.13}$$

The objective function E is convex when the Hessian $\mathbf{H}$ of E is positive definite. This condition is met if and only if the eigenvalues of the matrix $\mathbf{H}$ are positive. A positive

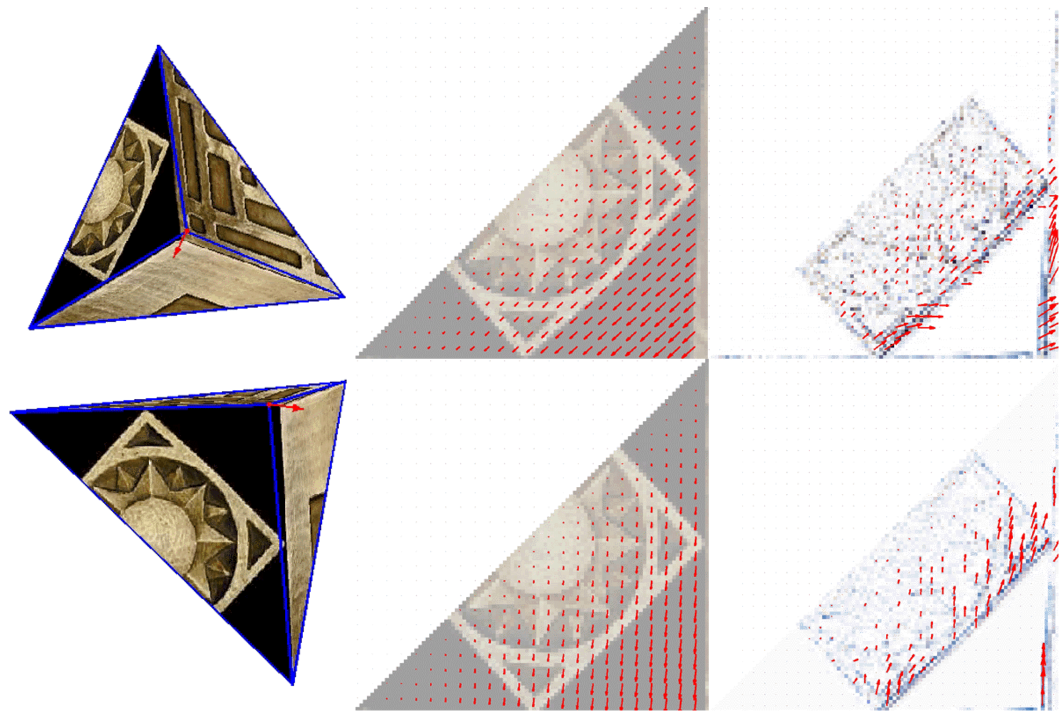


Fig. 9. A vertex displacement induces pixel displacements $\delta \mathbf{u}_{i\mathcal{V}}$ and $\delta \mathbf{u}_{i\mathcal{V}}$ shown as red arrows in the images on left. The pixel displacement is used to calculate the cell pixel-displacements $\upsilon_{i\mathcal{F}}$ and $\upsilon_{j\mathcal{F}}$ for the i-th and j-th cell images as shown in the middle column. Optical flow calculated using constant brightness constraint between the cell image and its eigenspace reconstruction, superimposed on their difference, is on the right.

definite Hessian indicates that the function has a unique optimum, in the local neighborhood, whereas a Hessian that has one or more eigenvalues zero will allow an entire manifold of solutions to minimize the objective function.

E is locally convex when $\ddot{\rho}(e_c) > 0 \ \forall \ e_c$. For the above norm, especially at small σ values, $\ddot{\rho}(e_c) = \frac{2\sigma(\sigma-3e_c^2)}{(\sigma+e_c^2)^3}$ can be negative, and therefore, one may not get a descent direction. As $\dot{\rho}(0) = 0$, $\ddot{\rho}(e_c)$ can be approximated by $\frac{\dot{\rho}(e_c)-\dot{\rho}(0)}{e_c-0} = \frac{\dot{\rho}(e_c)}{e_c}$ for small values of e_c Seber and Wild (1989). If $\ddot{\rho}(e_c)$ is substituted by this secant approximation $\frac{\dot{\rho}(e_c)}{e_c}$, which is positive everywhere, then the Gauss-Newton equations become

$$\sum_{l\in\{X,Y,Z\}} \left(\sum_{i=1}^{n} \sum_{\boldsymbol{u}_{\mathcal{F}}} \frac{\dot{\rho}(e_c)}{e_c} \frac{\partial e_c}{\partial \eta_k} \frac{\partial e_c}{\partial \eta_l} \right) \delta\eta_l = - \sum_{i=1}^{n} \sum_{\boldsymbol{u}_{\mathcal{F}}} \dot{\rho}(e_c) \frac{\partial e_c}{\partial \eta_k} \quad \text{for } k \in \{X,Y,Z\} \tag{5.14}$$

The fairing displacements, $\boldsymbol{\eta} = [\eta_X \ \eta_Y \ \eta_Z]^T$, are iteratively updated for step m as

$$\boldsymbol{\eta}^{(m+1)} = \boldsymbol{\eta}^{(m)} + \delta\boldsymbol{\eta}$$

for $\boldsymbol{\eta}^{(0)} = [\,0\;0\;0\,]^T$. After each vertex *fairing* step, the image patches are updated by considering the new position of the vertex. The new image patches are warped to their corresponding cell images. These cell images are used in the next optimization step. As the warping registers the image patches and the eigenspace, the approximation $[U_{\mathcal{F}}\mathbf{c}_i]$ continues to improve.

5.5.3 Cell-Pixel Displacement

The cell-pixel displacement function, $\upsilon_{i\mathcal{F}}(\boldsymbol{u}, \boldsymbol{\eta})$, in the i-th image corresponding to the face $\mathcal{F}$ has to be related to the vertex-displacement $\boldsymbol{\eta}$. The 3×4 projection matrix $\mathbf{P}$ for the i-th image is known, and let its p-th row be represented as $[\,\mathbf{r}_p \mid t_p\,]$ for $p = 1, 2$ and 3. The image coordinates, $\mathbf{u}_{i\mathcal{V}}$, corresponding to the 3-D vertex point $\mathbf{x}_{\mathcal{V}}$ is

$$\mathbf{u}_{i\mathcal{V}} = \begin{bmatrix} \frac{\mathbf{r}_1\mathbf{x}+t_1}{\mathbf{r}_3\mathbf{x}+t_3} & \frac{\mathbf{r}_2\mathbf{x}+t_2}{\mathbf{r}_3\mathbf{x}+t_3} \end{bmatrix}^T \tag{5.15}$$

The pixel displacement corresponding to a small change, $\boldsymbol{\eta}$, in the 3-D position of the vertex is

$$\delta\mathbf{u}_{i\mathcal{V}} = \begin{bmatrix} \frac{(\mathbf{r}_3\mathbf{x}+t_3)\mathbf{r}_1\boldsymbol{\eta}-(\mathbf{r}_1\mathbf{x}+t_1)\mathbf{r}_3\boldsymbol{\eta}}{(\mathbf{r}_3\mathbf{x}+t_3)^2} & \frac{(\mathbf{r}_3\mathbf{x}+t_3)\mathbf{r}_2\boldsymbol{\eta}-(\mathbf{r}_2\mathbf{x}+t_2)\mathbf{r}_3\boldsymbol{\eta}}{(\mathbf{r}_3\mathbf{x}+t_3)^2} \end{bmatrix}^T \tag{5.16}$$

This pixel displacement causes a change in the pixels of the cells. For pixels corresponding to face $\mathcal{F}$ in the i-th image, the affine transformation between the image patch and its cell is $H_{i\mathcal{F}}$ (a 2×3 matrix) is known.

$$\boldsymbol{u} = H_{i\mathcal{F}}\;[\mathbf{u}^T \;\; 1]^T \tag{5.17}$$

Assuming that $H_{i\mathcal{F}}$ does not change much with the small pixel displacement, the cell-pixel displacement function can now be related to the pixel displacement in the image as

$$\upsilon_{i\mathcal{F}}(\boldsymbol{u}_{\mathcal{V}}, \boldsymbol{\eta}) = H_{i\mathcal{F}}\;[\delta\mathbf{u}_{i\mathcal{V}}^T \;\; 0]^T \tag{5.18}$$

Once the cell-pixel displacement at the moving vertex is determined, the cell-pixel displacement over the entire cell can be calculated by interpolation, because the displacements at the other two *fixed* vertices and along the connecting edge are zero. The assumption here is that the cell-pixel displacements vary linearly across the cell image from the moving vertex to the other two vertices. We are trying to determine the location of the *faired* vertex that best approximates the surface patch with a planar face of

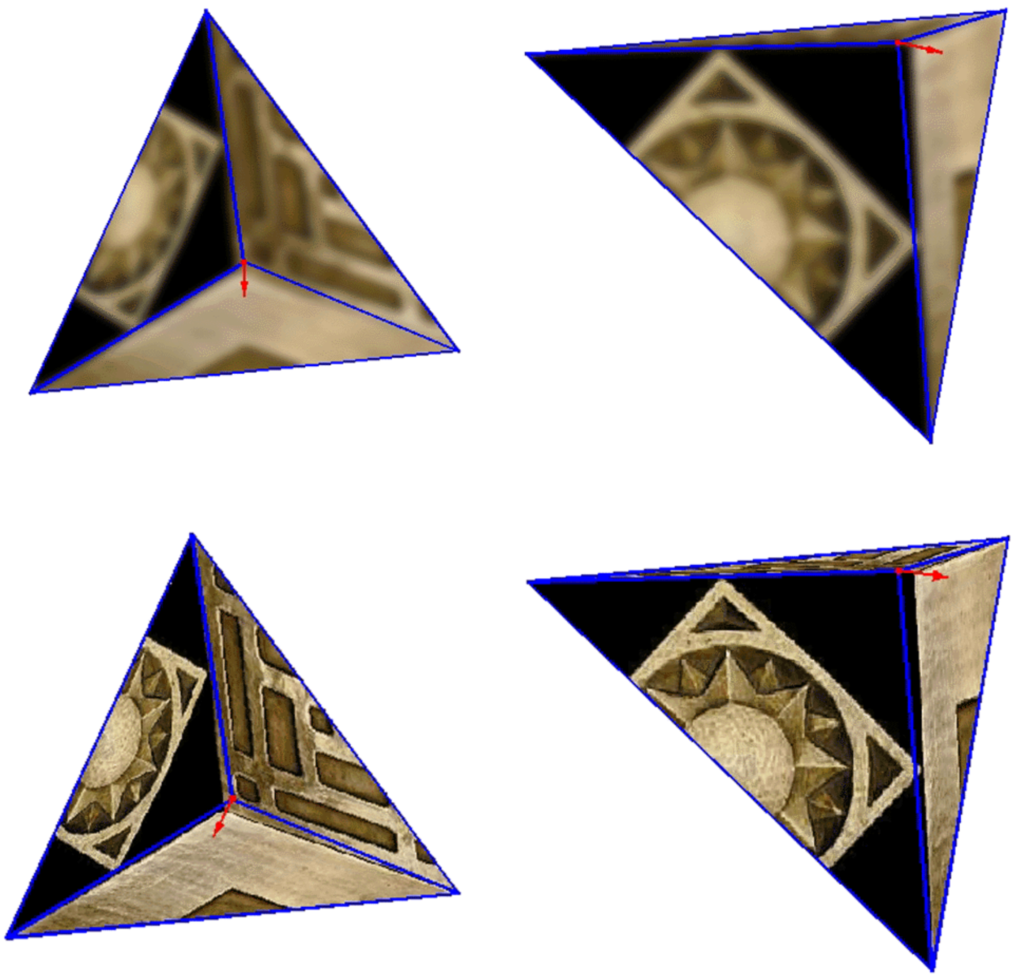


Fig. 10. At each iteration for updating the position of the vertex, a coarse-to-fine strategy is used to calculate the 3-D vertex displacement. The arrows show the displacement (magnified $20\times$) for two views in the 5-th iterative step.

the mesh. The same vertex location yields observed image-pixel displacements that are closest to linear variation of image-pixel displacements of a planar face. For the new position of the vertex, the cell-pixel displacements will correspond to a better approximation, and hence, a *more* linear variation of flow or pixel displacements across the cell image.

A vertex displacement during the fairing process is shown in Figure 9. The interpolated pixel-displacements in the cell images, for the same vertex displacement, are

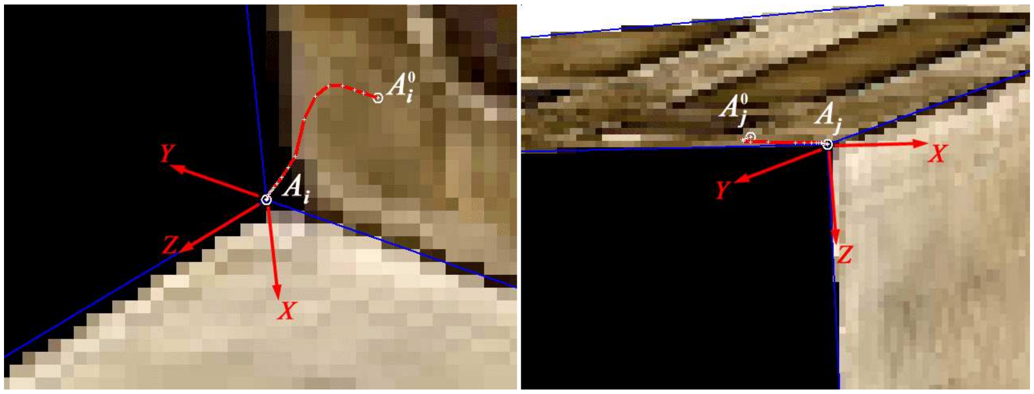


Fig. 11. Path of vertex A superimposed on images shown in the figure above. A_i^0 and A_j^0 correspond to the initial position of the vertex. A_i and A_j correspond to the 3-D point the vertex converges to.

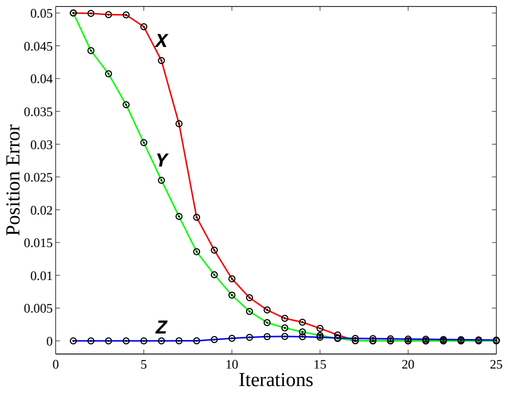


Fig. 12. Error in position of vertex for unit cube

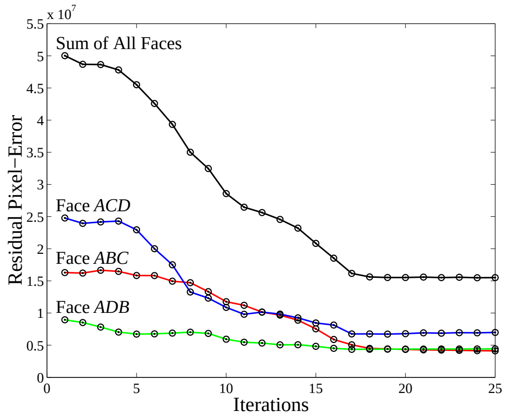


Fig. 13. Residual pixel-Error of common faces

shown in the second column of the same figure. The third column of Figure 9 shows the optical flow fields computed between the cell image and its eigenspace reconstruction. They point in the opposite direction to the pixel-displacements. The estimated vertex displacement is the change in vertex position for which induced pixel-displacements on faces best counter-balance flow-fields between the cell-images and their eigenspace reconstructions.

The cell size is 128×128 pixels. The path of the vertex, as it moves towards the actual corner, is displayed in Figure 11 for two views.

5.5.4 Algorithm Summary

The complete algorithm can be summarized as follows:

– For fixed number of iterations or until convergence,

  - Consider each vertex $\mathcal{V}$ in mesh and for all faces, $\{\mathcal{F}\}$, sharing the vertex:
    - Start with coarse level image patches in Gaussian pyramid corresponding to the faces.
    - Warp triangular image patches to cell images for all faces.
    - Get Eigenspace representation, $\mathbf{c}$, for each face.
    - For a fixed Eigenspace representation, determine vertex displacement, $\boldsymbol{\eta}$.
    - Project $\mathbf{c}$ and $\boldsymbol{\eta}$ to the next level of Gaussian pyramid.
    - Repeat
  - Update position of vertex, $\tilde{\mathbf{x}}_{\mathcal{V}} \leftarrow \mathbf{x}_{\mathcal{V}} + \boldsymbol{\eta}$.
  - Repeat for all vertices.
- Iterate for whole mesh again.

6 Results

This chapter uses the caveats delineated in the last two chapters to create descriptive 3-D models of real scenes. Image and depth coherence form the basis of both EigenFairing and the Epitexture constraint. In fact, both caveats are interrelated. The position of a 3-D point, be it a vertex of a mesh that approximates a surface or a point on the slice of an isodepth map, depends on the texture in the neighborhood of that point. Textural properties of a region as projected onto a set of images determine the geometrical positions of points that model it. EigenFairing repositions the vertex and refines the planar approximation by exploiting texture of the mesh faces common to that vertex. In the Epitexture constraint, the position of a point on the slice is determined by the texture in the point's neighborhood as well as the fact that points in a close region are connected to each other. The parameter r in the GGMRF model that determines the smoothness of the surface patch is determined by texture.

In many 3-D modeling applications, that use either dense stereo or any SfM algorithm, the visual quality of the model depends mostly on the collective texture of a region. Any model-based representation of a surface—mesh of vertices and textured faces, dense point clouds, layered depth images, Q-splats, Surfels or Radial Basis functions—has to approximate both texture and depth well as any inconsistency in either one is visually perceived as an aberration in the other. Texture and structure together effect view-dependent quality of a model. Image-based representations such as plenoptic modeling, lumigraph, light fields, or image-based priors have to capture the textural properties of the scene and reproduce them in such a way that view-dependent properties such as depth and surface smoothness or occlusions are visually coherent. Geometrical and textural properties of a surface region are inter-dependent and we have illustrated that coherence of one property can indeed be used to enforce coherence in the other.

Ideally, one would like to learn how to derive compact representations of texture and structure that occur in natural scenes, and subsequently, a coherence measure based on the two. However, we have focused on showing the inter-dependence between texture and structure and how their representations are refined by enforcing coherence to observed images and depth data. Coherence has been implemented by minimizing the distance-from-feature-space or a robust function of pixel-error between re-projection of the model and actual images of the scene. The distance-from-feature space has been used for data-sets that have more than five images.

In this chapter, we shall first use image coherence to generate a texture mesh of sparse point cloud and refine the mesh by removing faces that are not coherent with observed images and have high re-projection error. The initial mesh is built by finding the most coherent mesh faces and subsequently refined using edge swaps and face removal. Swapping edges to improve the mesh geometry and texture often gets stuck in local extrema points of the cost function. Many a times, the local extrema points are due to occlusions and existence of faces that really do not approximate the surface structure that well due to absence of well-positioned points as vertices. Faces that yield high re-projection error are removed. Once we have a rough model of the scene, we proceed to refine it using EigenFairing—repositioning vertices of the mesh to derive the best possible linear approximation of the scene.

The process of EigenFairing repositions vertices of the mesh by using texture such that planar mesh faces are close to the surface patch they approximate. However, most natural scenes have curved surfaces as well as patches that have small structural variations such as grooves, striations, and fine-scale geometry (Dana and Nayar, 1999). We model these details and non-planarity of surface patches using the Epitexture constraint. Epitexture constraint further allows us to use depth and image priors to improve view-dependent appearance of the model.

6.1 Image Coherence

In many vision applications, there is a need to build a descriptive model from a sequence of images captured from a mobile platform. Depth is computed by detecting and tracking image features and then using a stereo or SfM algorithm. This approach yields a set of 3-D points whose density depends on the number of reliable features tracked across images. The three image shown in Figure 1.a is taken from an aerial vehicle. There are 107 features detected and tracked using the Kanade-Lucas-Tomasi (KLT) algorithm (Tomasi and Kanade, 1991; Tomasi and Shi, 1994). The initial mesh is build by sequentially adding triangles to a mesh starting from the center of an image. At first, three neighboring points at the center of an image are chosen to form a mesh face. For each

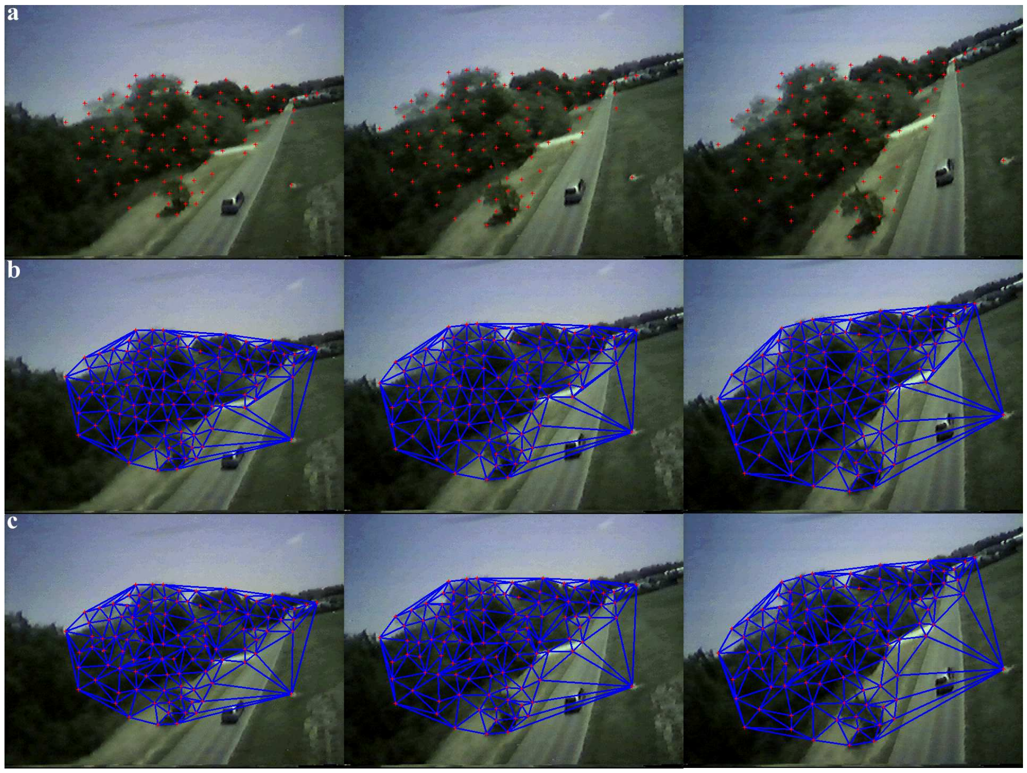


Fig. 1. a. Features detected and tracked in a sequence of aerial images, **b.** Initial mesh generated using the features, and **c.** Final mesh that is a result of refining the initial mesh using edge swaps. Notice that edges of the final mesh are aligned along the intersection of foliage and ground.

edge of this mesh face, a neighboring point is considered for the next face such that the face does not contain any other point. As there could be many such neighboring points, the face that has the lowest re-projection is chosen from the set of possible faces. The initial mesh thus generated has 200 faces and is shown in Figure 1.b. However, this mesh does not model the surface accurately, evidence of which can be clearly observed in Figure 2. The texture for each mesh face is the weighted average of image areas that the face projects into. The re-projection error for a particular image is the sum-of-squared-difference between the original image area and the re-projection of the textured mesh face. Edges of the mesh are swapped to create a better approximation of the model. The final mesh is shown in 1.c. This algorithm is the same as image-consistent triangulation proposed by (Morris and Kanade, 2000). The refinement algorithm is sensitive to texture and the positions of the 3-D point features. In regions of little or negligible texture, swapping edges may lead to complicated overlapping surfaces. The sky has no texture

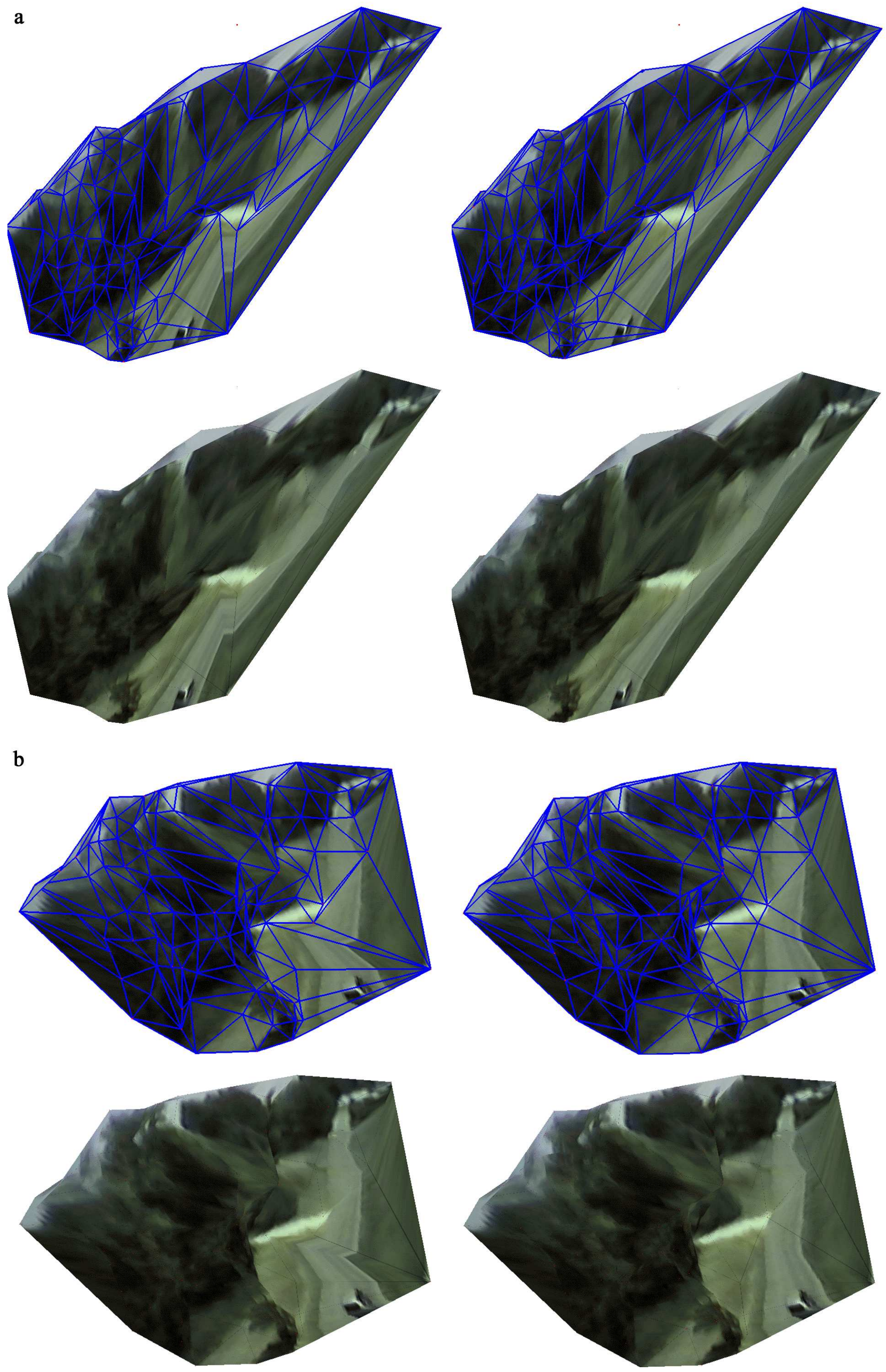


Fig. 2. Two rendered views of the model are shown in **a** and **b**. In each view, the column on left shows the initial model and the final model is on the right. The mesh shown in the top rows has textural artifacts that are clearly visible in the images on the second rows. **b.**

and the corresponding regions of the model have artifacts due to overlapping mesh faces which are more apparent in renderings of the model [Figure 2].

Re-projecting a model, be it a textured mesh or a dense point cloud with color, back into the images should be close to the actual images. For a given sequence of images, the texture attribute of the model is derived from corresponding image areas that a surface patch projects into. A compact representation of texture is computed from these image areas by using the principal components of their eigenspace decomposition. In case of a mesh, the triangular image areas that correspond to a particular mesh face can be warped to a fixed image area. The magnitude of the projection of the fixed image areas onto the eigenspace is a measure of how close the face is to the actual surface patch as viewed from the direction the image was taken. Pentland et al. (1994) define a distance-from-feature-space (DFFS) as the root-mean-square of the residual image, $I - [U_{\mathbf{g}}]$, and note that this error measure can be used for localization. Any variation in a sequence of cell images is due to local displacement between the actual surface and the planar face approximating it. A lower local displacement i.e., difference between the triangular face and the surface patch would necessarily mean that the eigenspace will have fewer but larger eigenvalues. The assumptions for all our data sets are that they are static scenes, lighting is fixed, and the surface has Lambertian reflectance properties. In the next example, we will show image coherent model refinement using edge swaps and face removal.

6.1.1 Image Coherent Edge Swaps

In this section, we provide results of a mesh triangulation refinement using image coherence on a real data set. Figure 3 shows a range scanned natural scene and a texture mesh generated from the range data and images. Vertices of the mesh were points sampled from the range data at geometrical extremities and edges. The remaining range data has not been used for modeling for this data. We will show how this discarded range data can be effectively used to generate a dense point cloud model using isodepth maps later. As of now, we select points in the range data that can be used as vertices of the textured mesh. The points are sampled at geometrical extremities by using the curvature consistency algorithm (Ferrie et al., 1993b,a; Sander and Zucker, 1990). The curvature consistency algorithm achieve a stable surface representation by iteratively minimizing a functional related to the satisfaction of local constraints on the curvature of the surface. The method uses a local surface model, which describes the local neighborhood around

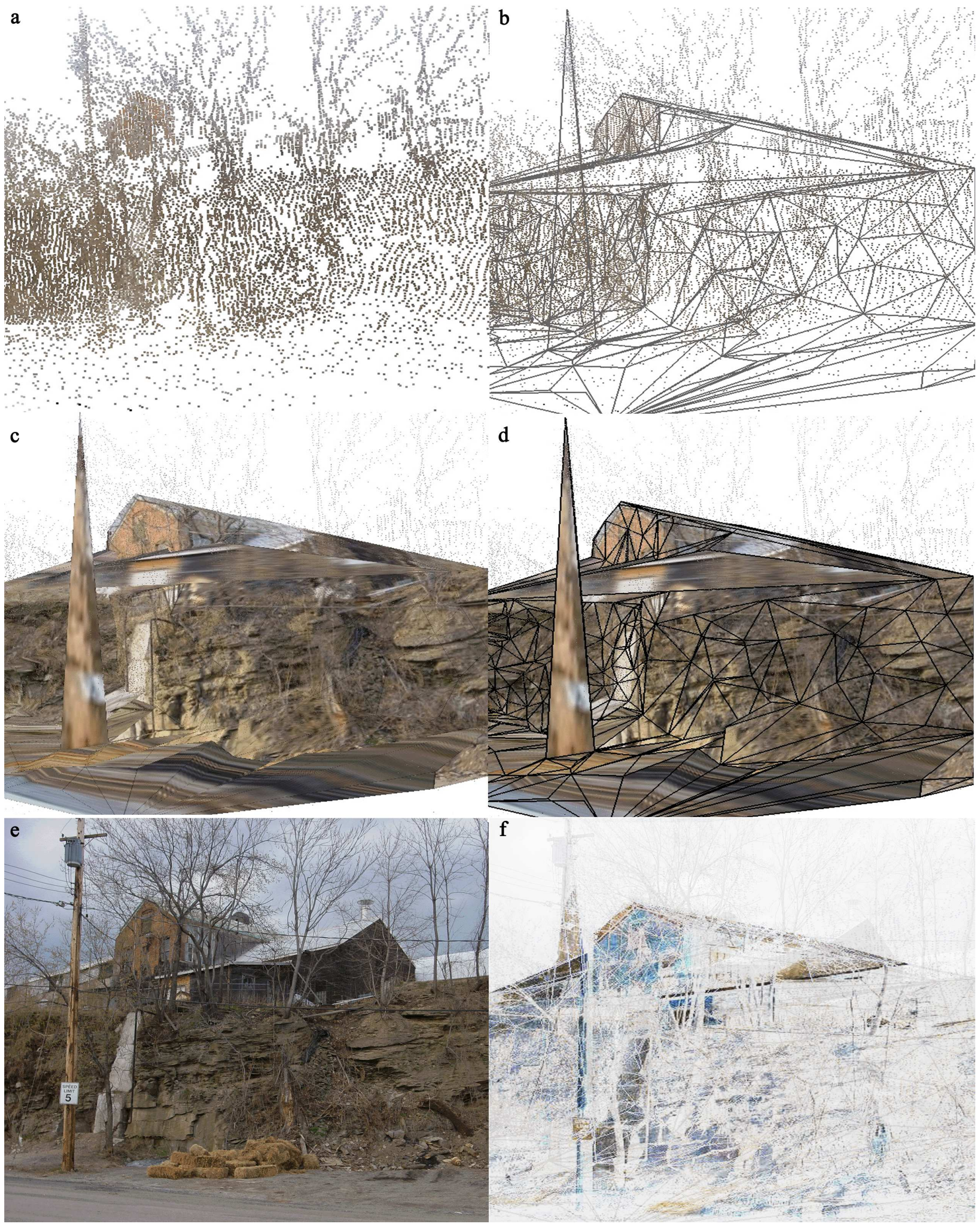


Fig. 3. a. The dense point-cloud from a range scanner, **b.** mesh generated from sampled range-data points, **c.** texture-mapped mesh, **d.** re-projected image from a particular view-point, **e.** image from the same view-point, and **f.** pixel re-projection error.

a particular 3-D point on the surface with a quadric patch. The information associated with each patch includes the location of the given point, the two principal directions of the paraboloid, the minimum and maximum curvatures along the principal directions, and a local coordinate frame– all of which are stored as the "augmented Darboux frame" for that point. After estimating the local surface geometry, we look for points where the mean curvature vanishes. The mean curvature is defined as the average of the principal curvatures. Krsek et al. (1997) mention that curves of inflection points may be the best distinguished features of a triangulation when the data has no sharp edges. In addition to this property, we detect where the surface is locally flat or has a low curvature in some direction. This is given by the zero-crossings of the Gaussian curvature. The Gaussian curvature is defined as the product of the two principal curvatures. There is much research done in segmenting 3-D data according to the mean "curvature-Gaussian curvature" map (Petitjean, 2002).

A 3-D surface model was constructed from 400 vertices that were sampled from the dense but noisy range data using the mean and Gaussian curvature. The underlying mesh consisted of 700 faces that was a result of the Delaunay triangulation of the vertices. The dense range data and the triangulation is shown in Figure 3. An eigenspace representation for texture is derived from the Image areas corresponding to a triangulated mesh face. Edges are swapped to minimize the distance-from-feature-space or maximizing image coherence. Sixteen images were used generate the final mesh. Figure 3.e shows the re-projection error. Regions of the surface that do not approximate the surface well have high pixel re-projection error. Pixel re-projection error is computed as the absolute value of pixel difference between the actual image and the re-projection of the model from the same view-point.

The curved surface of the ridge is well modeled by the final mesh. Large textural artifacts in the model are due to the absence of 3-D range data points for sampling vertices for mesh faces. Range data do not sample surfaces in regions that are occluded or cluttered. They often miss out edges and corners of the surface as well. Mesh faces build out of vertices that are sampled from range data, therefore, lead to poor approximation of the real surface. The large mesh face that approximates the lower part of the house is a poor approximation of the structures in that region. However, there is little that can be done to improve the model at this stage given that there is no range data for that region due to limited visibility of the scanner. By using calibrated cameras and known positions of 3-D features, the effect of limited sensor visibility and occlusions can be reduced by

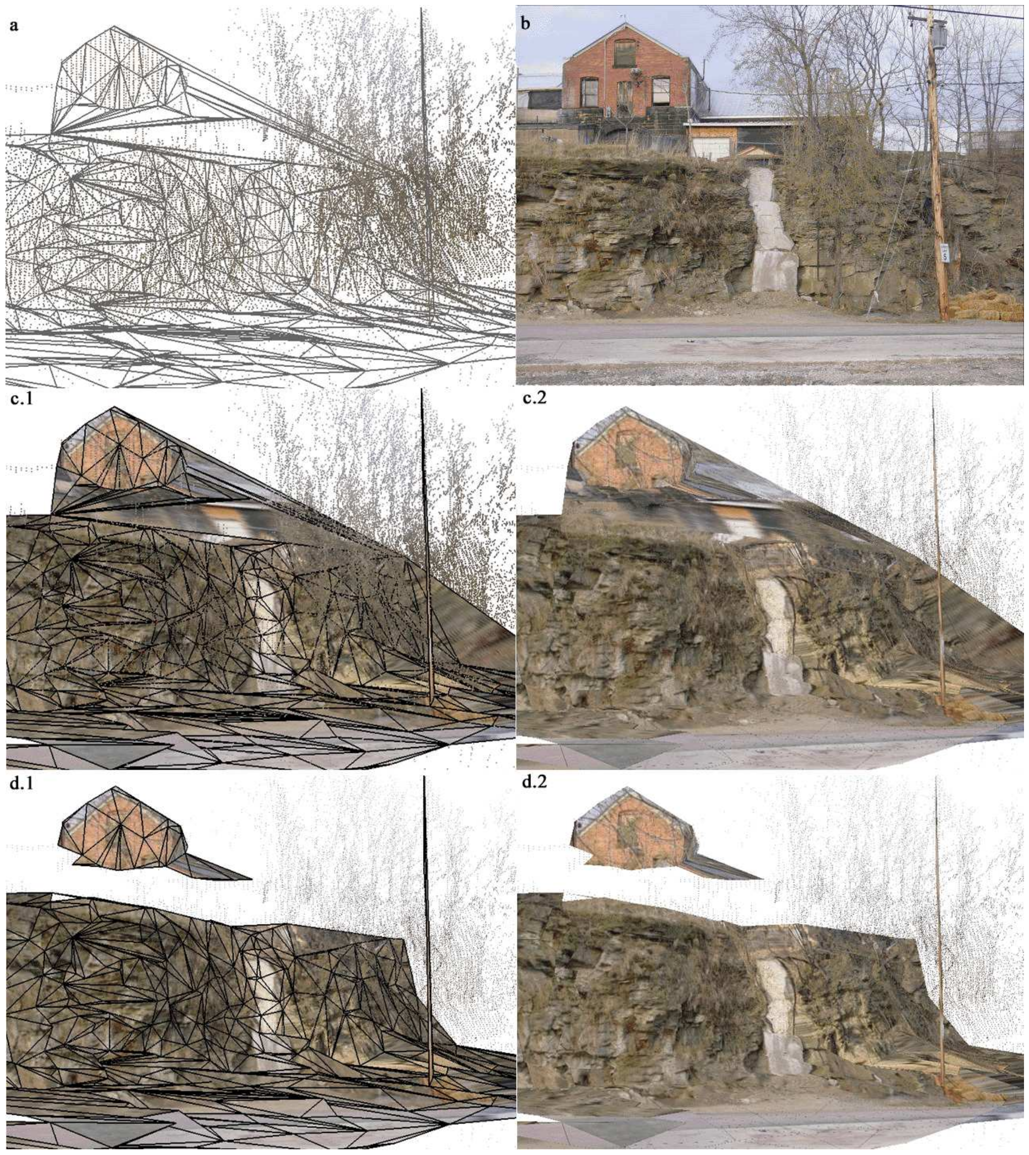


Fig. 4. **a.** The dense point-cloud and the mesh after edge swaps, **b.** the actual image, **c.1.** texture-mapped mesh, **c.2.** re-projection of the model into the image; notice the textural artifacts at mesh faces that do not approximate the scene well, **d.1.** texture-mapped mesh after removing faces with high re-projection error, and **d.2.** re-projection of the model after face removal.

careful surface selection. It is also possible to overcome this problem by relocating the mesh vertices to points or selecting more points within the mesh face.

The artifacts due to the electric pole that occludes regions in its shadow have been negligible due to the fact that the eigenspace representation captures the true texture of the surface and appropriately accounts for pixels that belong to the occluding edge. Swapping edges at occluding regions at times produce hidden faces which increase surface complexity without helping to explain images. In order to bias the edge-swapping strategy to simpler surfaces we assume that each triangle of the 3-D surface is visible in at least five images. We exclude potential swaps that produce faces which are only visible in four or less images. The single mesh face that models the electric pole results from removing faces that connect the pole to the surrounding areas–a strategy that we will discuss in the next section.

6.1.2 Image Coherent Face Removal

Natural scenes often consist of surfaces and objects that are not physically connected to each other. In addition to sudden changes in depth, there are discontinuities due to occlusions or shadow regions as viewed from sensors. Modeling these scenes with a single connected set of mesh elements or points lead to high textural artifacts and structural inconsistencies. One can, of course, give a possible description of a region that has not been completely viewed from sensor vantage points but with a change of viewing pose the description does not guarantee visual consistency. Moreover, a model with regions that poorly describe texture and geometry of the scene cannot be often refined as any coherence measure is saddled with local minima. It is, therefore, of up most importance that one starts with a rough model of the scene–an approximation that can be refined by using a optimization tool to maximize coherence.

Figure 4 shows a 3-D surface model created with edge swaps and face removal as mesh refinement operators. Faces that have a large re-projection error–difference between the eigenspace reconstruction and actual image–have been removed. Most faces that are removed this way belong to occlusion boundaries or silhouettes of surfaces. Thin, wiry, and branched structures that often give rise to such faces are abundant in natural scenes. Any mesh refinement strategy without the removal of faces that do not correspond to an actual surface generates complicated overlapping meshes that are not coherent with image or depth data. In addition to these factors, what makes face removal an important mesh refinement operator is that in many vision applications one would like to ascertain regions of the model where no real surfaces exist.

Image coherence can also be used for detecting any inconsistency in texture of faces in a mesh. In some cases, the mesh has thin elongated faces that correspond to thin structures such as electric poles or branches. These faces do not represent the full surface

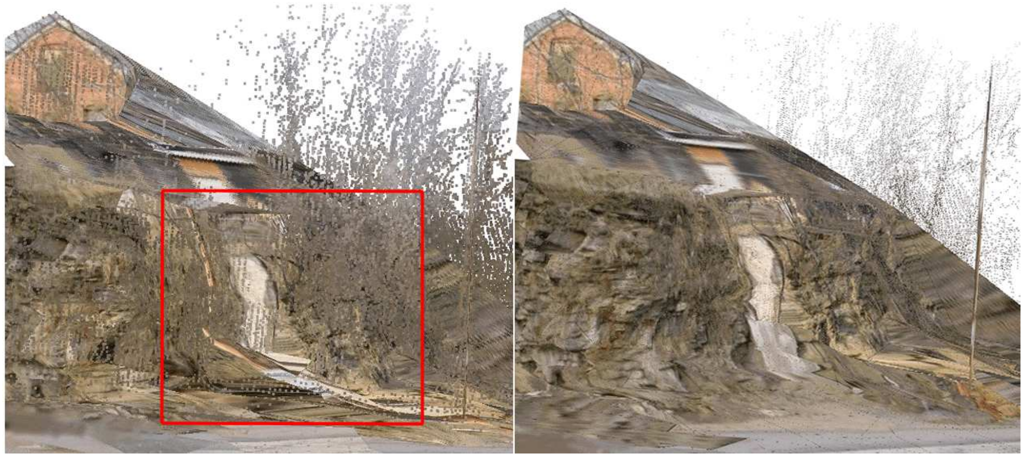


Fig. 5. A thin and elongated face occludes small areas of larger triangles of the mesh when viewed from a particular direction. On the left is a model where each face of the mesh is texture-mapped using a single image. In self-occluding models, “smears” of texture inconsistent with the model often get mapped onto the mesh. A better idea is to texture-map a face by the image cell that has the lowest distance-from-feature-space (DFFS). The artifacts are not present in the latter case.

structure. Determining occlusions by Z-buffering does not remove all pixels in images that correspond to these structures from being texture-mapped onto the mesh [Figure 5]. Finding the image patch that is most coherent i.e., has the highest projection into the eigenspace to texture-map the face removes this artifact.

6.1.3 Image Coherent Vertex Fairing: EigenFairing

Once we have generated and refined a mesh through edge swaps using image coherence, we are interested in refining if further by relocating the vertices of the mesh. Points that form vertices of the mesh are often not at critical points such as edges and corners of the surface. EigenFairing moves these vertices to those desired points based on coherence of texture of the mesh faces. We applied EigenFairing to real outdoor scenes and achieved significant improvement in the geometrical and textural quality of 3-D models.

Figure 6 shows another real data set where a SfM algorithm was used to derive the initial position of vertices from 18 images. 102 image features were manually selected and then tracked using the Kanade-Lucas-Tomasi algorithm across the image sequence. The convergent bilinear algorithm proposed by Mahamud et al. (2001) is used for reconstruction the 3-D positions of the features. The re-projection error of the 3-D points is 0.6 pixels. We did observe that small errors in detecting and tracking of image features lead to a slightly skewed reconstruction–a matter that we intend to investigate later. The point-

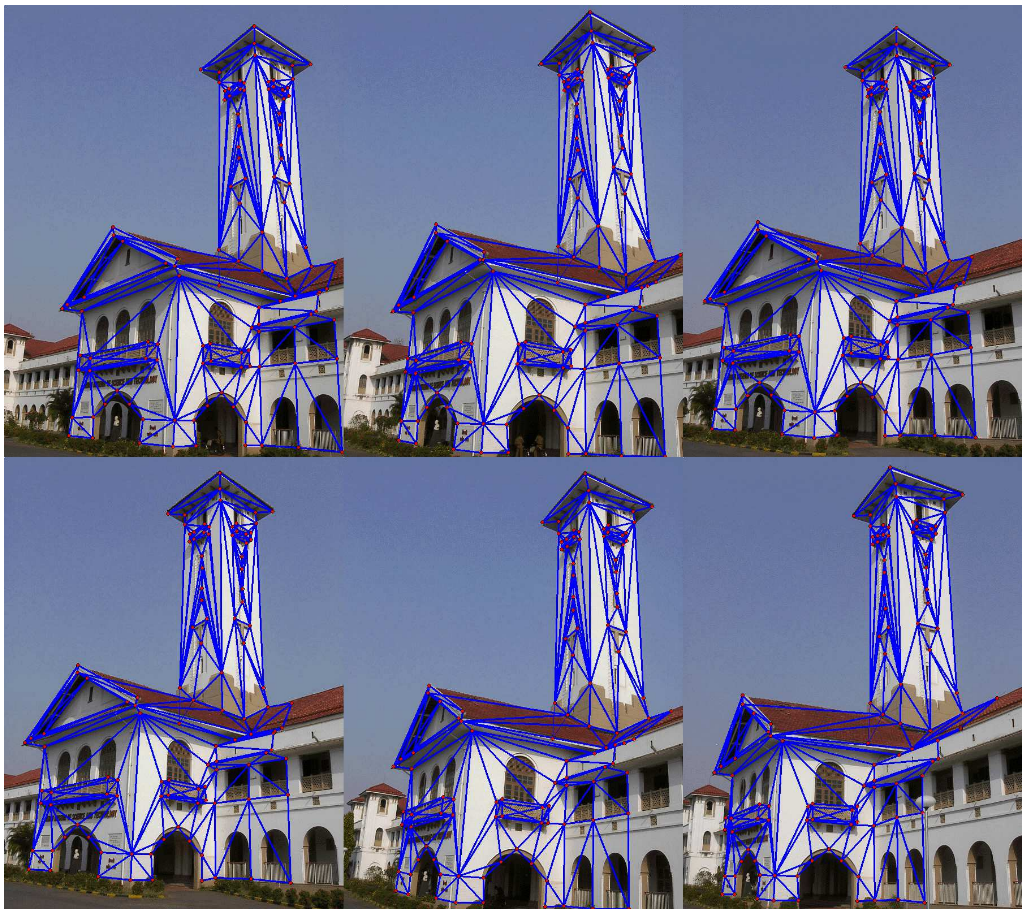


Fig. 6. Six out of the eighteen images used to extract the position of the points are superimposed with the initial mesh. The points, which are used as vertices of the mesh, do not exactly correspond to real corners and the edges thereof do not correspond to edges of the actual surface.

based reconstruction is triangulated using image-coherent triangulation which included face removal and edge swap operators. The mesh is overlaid on six images shown in 6. Apart from small errors in re-projections of the 3-D points, the points do not precisely correspond to actual corners or critical points of the surface–a property much needed for faces to lie close to the surface patches that they approximate.

EigenFairing the 3-D points, by minimizing the error between the re-projection of texture-mapped faces and the corresponding areas in actual images, we obtain new positions for the points as shown in Figure 7. The EigenFaired points better correspond to corners of the tower and roof. It is essential to have enough texture on each face to refine the position of the vertices; the clay tiles on the roof and brick patterns on the facades

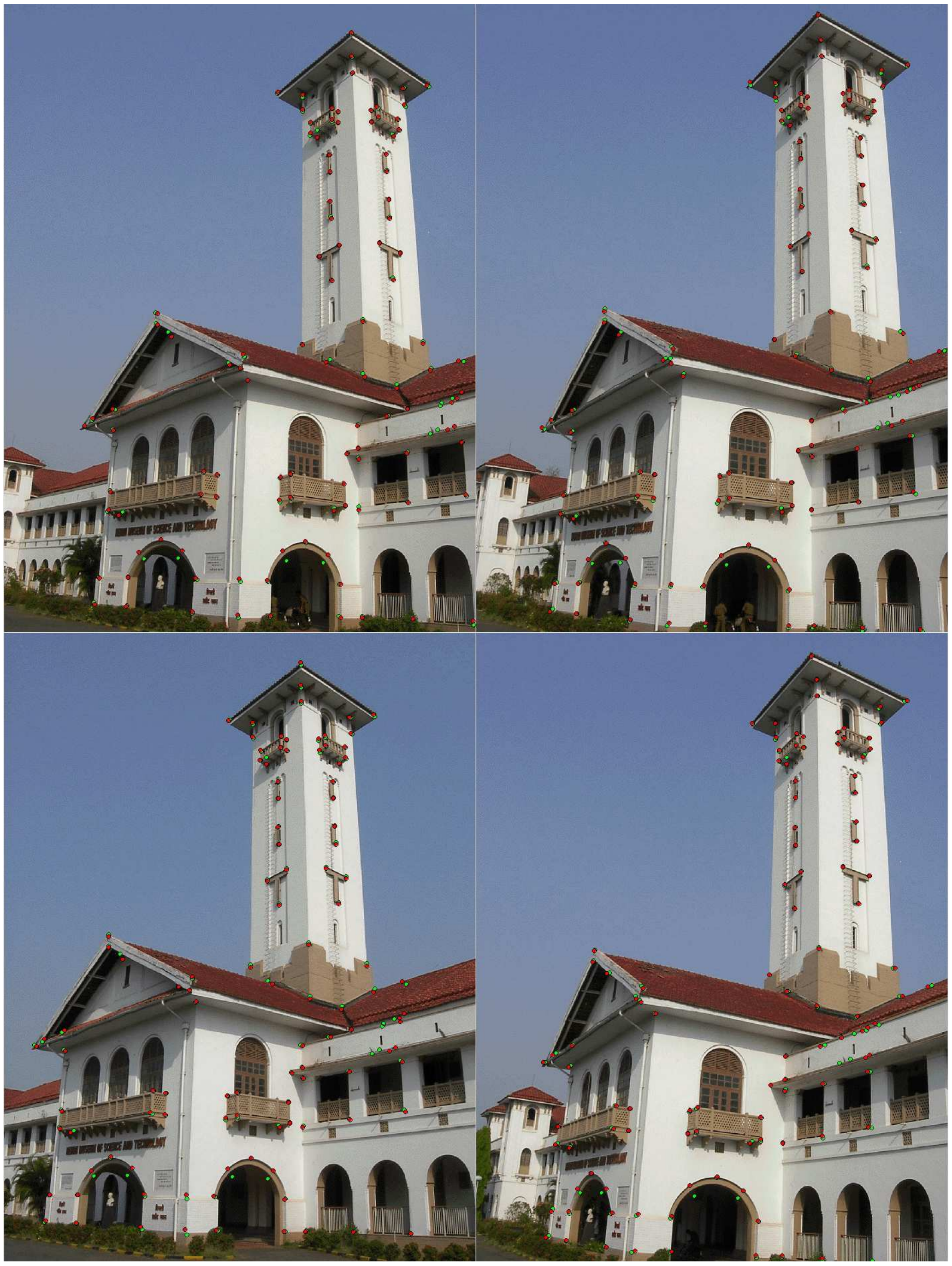


Fig. 7. The above images show the initial vertex positions in green and the faired vertices in red. The faired vertices accurately correspond to corners and edges of the architecture. Notice the points along the archways, roof and balcony edges.

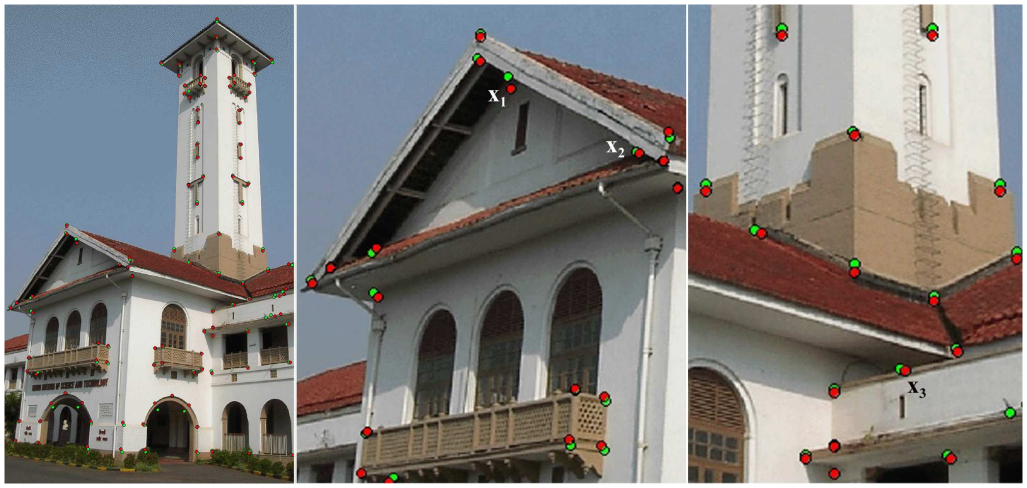


Fig. 8. Two closeups of the image on left show initial vertex positions in green and faired vertices in red. Vertex positions shown in red precisely correspond to surface discontinuities. Vertices $\mathrm{X_1, X_2, and X_3}$ are photometric features that do not correspond to real corners of the surface and faces common to the vertices do not lie on the surface. A photometric feature is often due to “proper T-junction” where a line terminates or is covered by a occluding boundary, resembling a “T”. The faired vertex positions no longer correspond to photometric features in images.

provide enough texture for minimizing the re-projection error. Points at the archways move inward so that the mesh edges are aligned along the 3-D edges. These points form a special set of mesh vertices as they lie at boundaries of the mesh. The faces of the mesh common to these boundary vertices lie on the surface before and after EigenFairing. However, we add a penalty to pixels close to edges that yield high re-projection error. These allows us to reposition the boundary vertices such that pixels with re-projection error are excluded from the final mesh. EigenFairing meshes at their boundaries using image coherence require heuristics that are based on identifying and minimizing outlier pixels.

A few corners or critical points of surfaces lie in regions that are occluded in the given set of images. EigenFairing successfully relocates vertices of the mesh to those points if the there is enough texture corresponding to the neighboring faces. The initial positions of those vertices are usually at T-junctions–the points of overlapping of occluded and occluding regions as visible in images. A proper T-junction occurs where a line segment lies on a surface in space that is occluded by another surface, whose occluding boundary contributes to form the “T”. These junctions lead to photometric features that are false corners. A sequence of corresponding photometric features tracked across the given set of images does not correspond to one particular point in 3-D as each image feature is

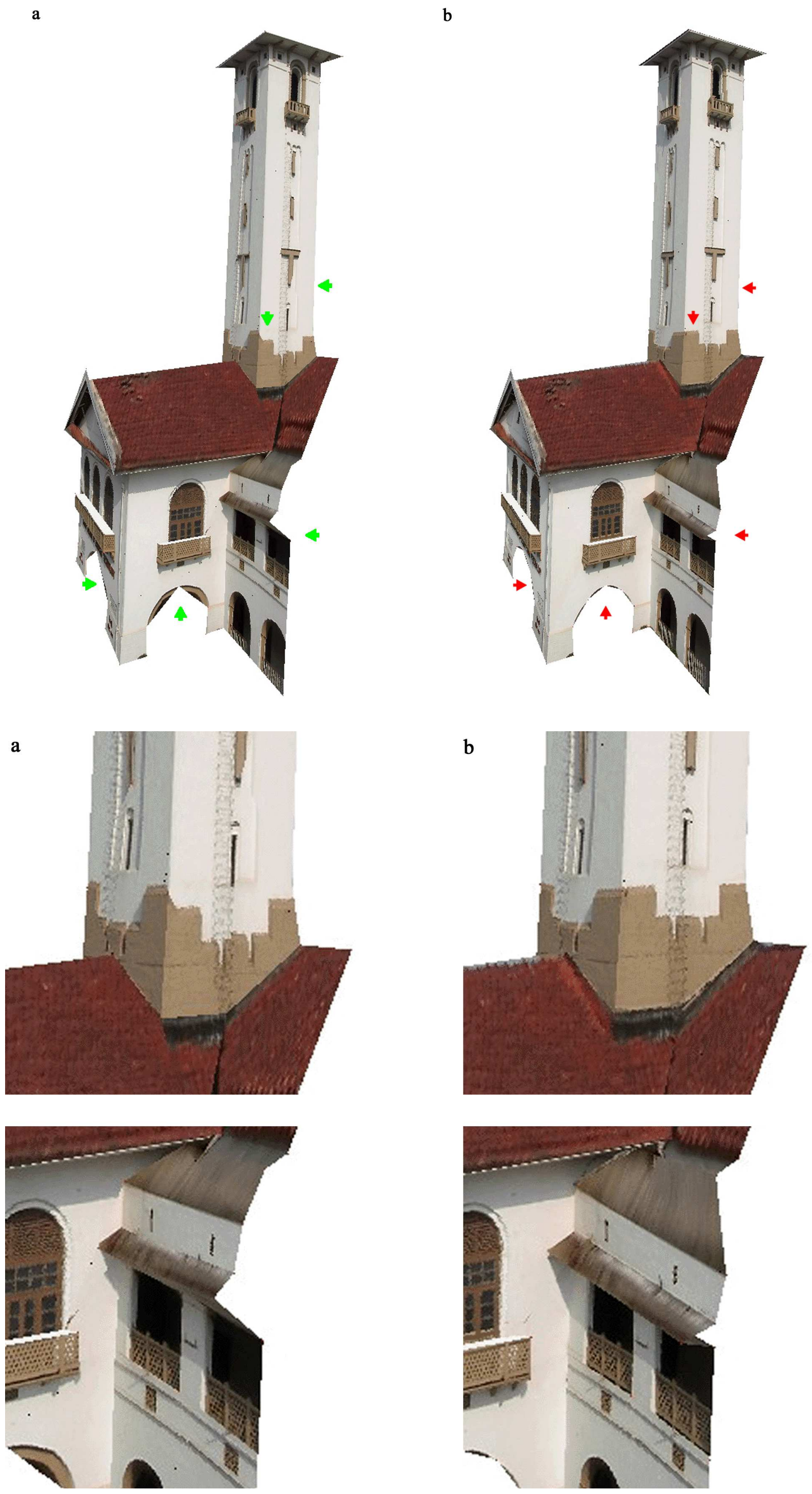


Fig. 9. The initial model and the EigenFaired model are shown in (a) and (b) respectively. The green and red arrows point out the presence and absence of modeling artifacts respectively. Notice the areas around the arches, window overhangs, and balcony edges. Close-ups delineate differences between the models.

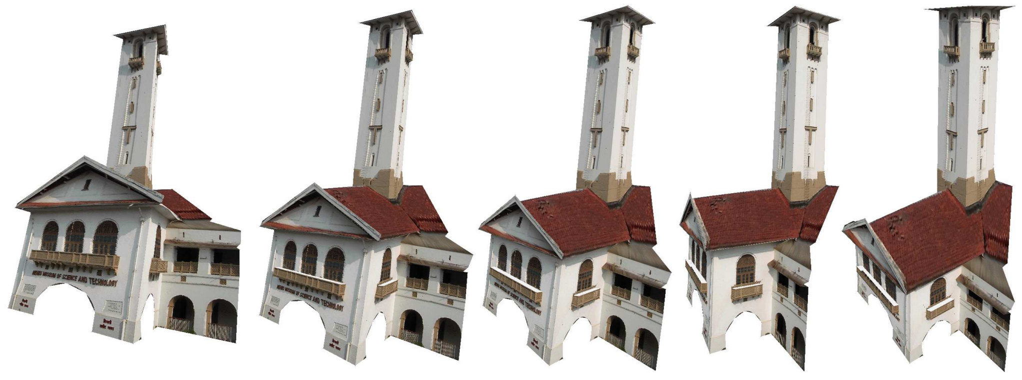


Fig. 10. New views rendered using the EigenFaired model.

view dependent. The 3-D point that is reconstructed using bundle adjustment or any other SfM algorithm from this sequence of photometric features does not correspond to a point on the surface. However, EigenFairing relocates such a 3-D vertex based on the fact that the texture in the neighboring region should be coherent with images. The vertex is positioned such that any approximation of the actual surface by the textured faces should be accurate, even if the new position of the vertex is in a shadow or occluded region. Figure 8 shows a few vertices that correspond to photometric features relocated such that they better correspond to corners.

In the previous example, the 3-D points were reconstructed from image features using SfM. Depth information for the scene was derived from images. The initial set of 3-D points can taken from a range scanner as well. Range data provides a dense point-cloud representation of structure. However, range scans often miss out corners and edges due to their physical minuteness. Any sampling of the range data does not yield vertices for a good planar approximation of the surface. The EigenFairing process can fair the surface structure by moving these samples of range data to the desired locations. The desired location, of course, do not exist in the dense range data.

In Figure 11(a) the initial set of 3-D points for generating the mesh is sampled from range data at points of geometrical extremities and discontinuities. The data was registered with a sequence of 22 images, three of which are shown in Figure 12, by selecting the closest points of correspondence. The set of 3-D points was triangulated using image-coherent triangulation. Nevertheless, points chosen as vertices do not accurately correspond to physical corners in images. Edges of the mesh are not aligned with edges of the facade as viewed in the images.

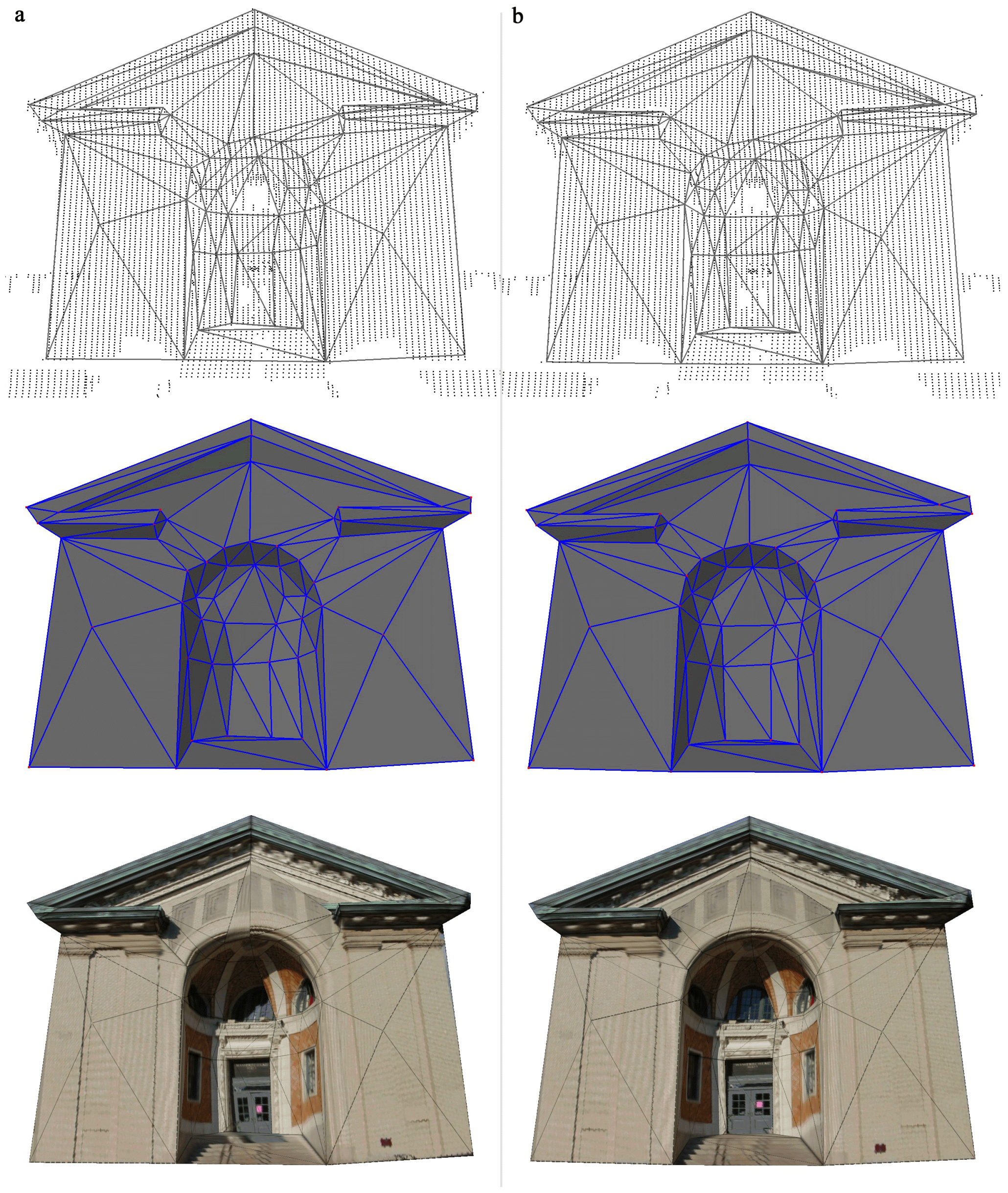


Fig. 11. The mesh shown in column (a) is generated from sampling range data at critical points. Structural and textural artifacts are clearly visible in the mesh and its texture-mapped model Compare the mesh in the middle row to the one on right. Column (b) on the right shows the EigenFaired mesh. Generating a mesh by selecting points of high curvature does not necessarily guarantee the best planar approximation of the surface.

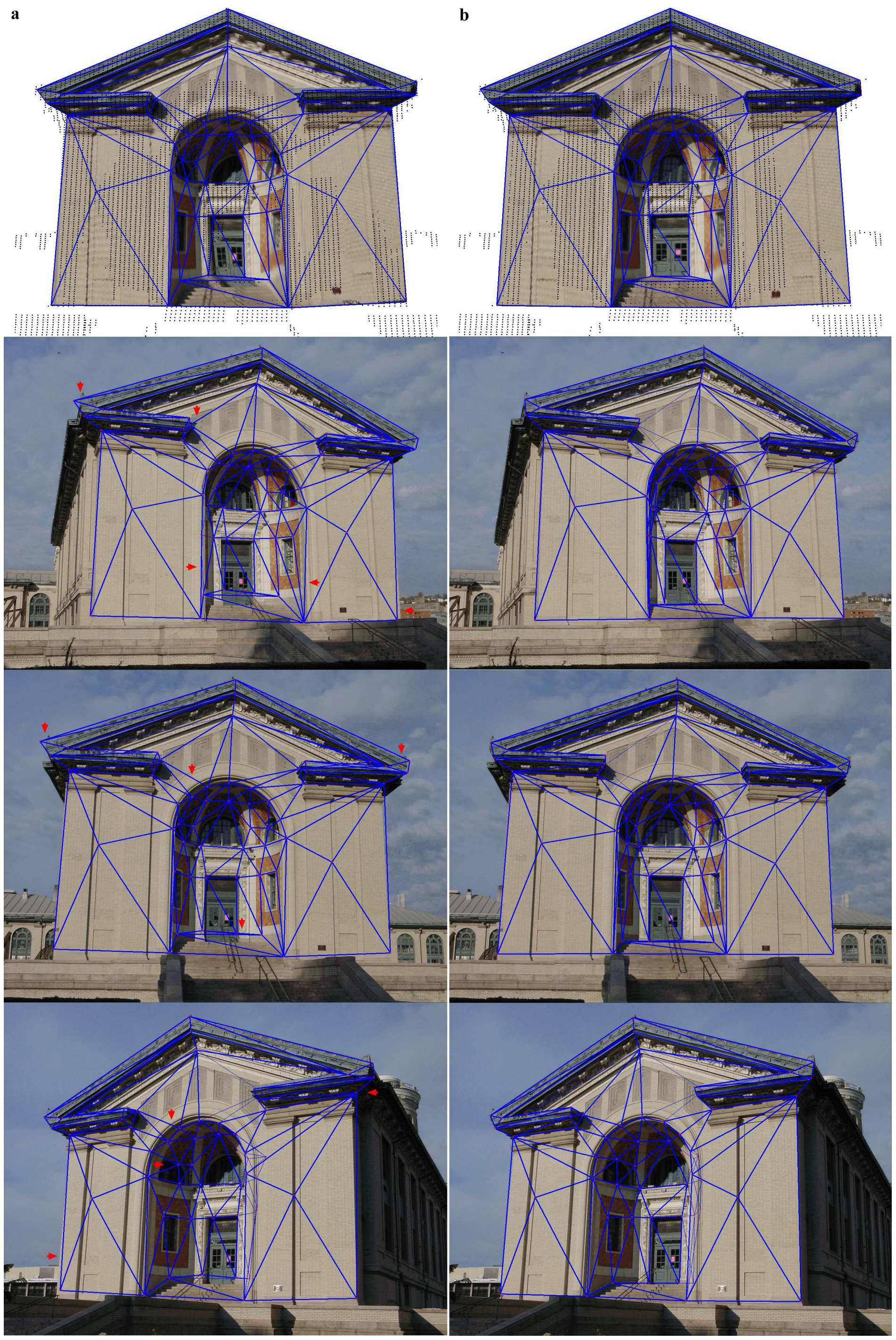


Fig. 12. On the first row, 3-D range data is superimposed on the texture-mapped models before and after EigenFairing. Range scans often miss out corners due to occlusions or their spatial minuteness. Selected vertices are not always best positioned for surface approximation, as seen on the inner dome. The edges are close to, but not perfectly aligned with the physical edges of the building as seen in the images. Column (a): Initial mesh. Column (b): The EigenFaired mesh. The red arrows point at mesh edges that have considerable inconsistencies with edges of the surface as viewed in images.

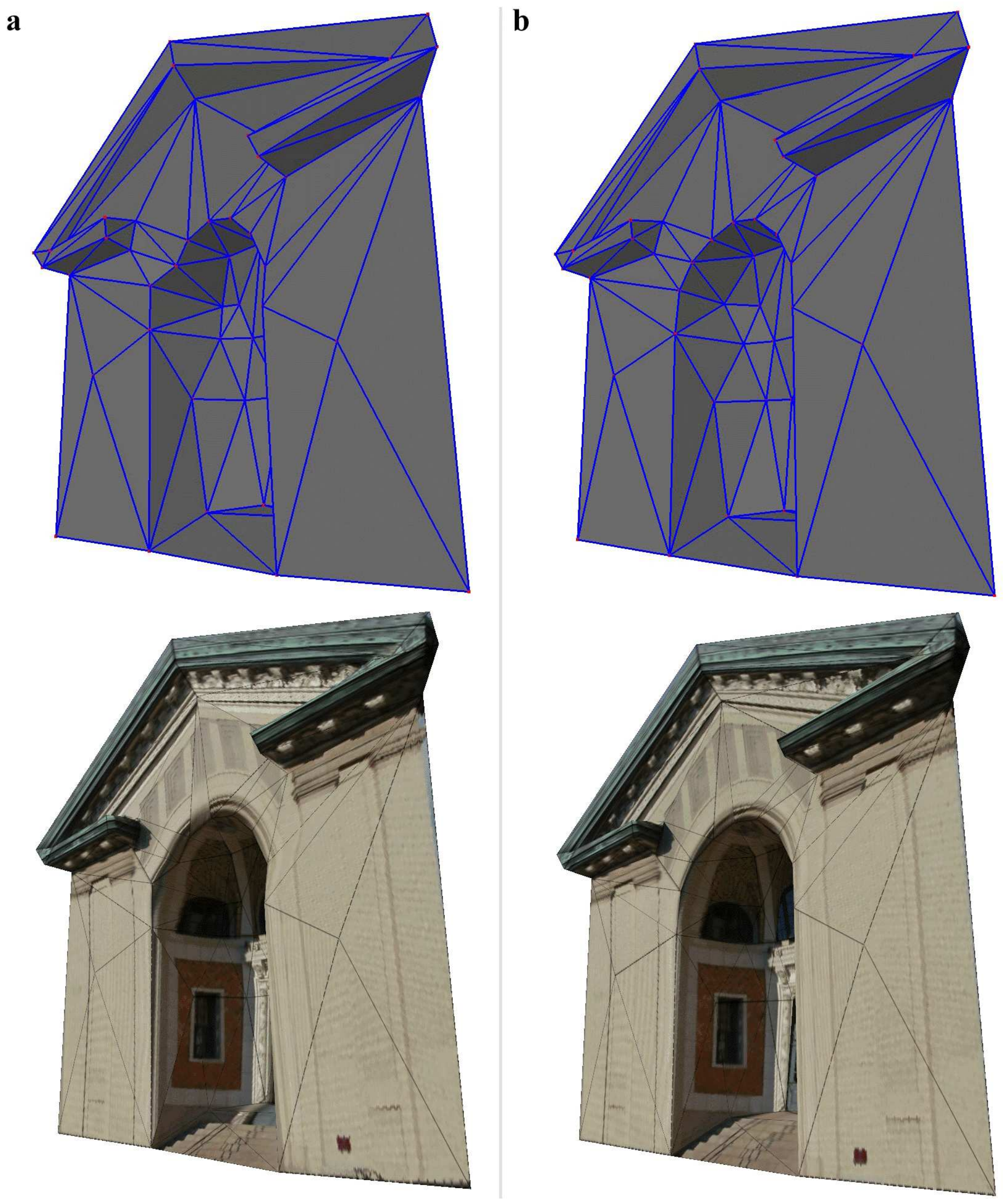


Fig. 13. Column **a.** The mesh and the texture-mapped model for the initial position of vertices. **b.** The EigenFaired mesh and model. Notice that the initial mesh has irregularities in structure along the arch and on the inner dome. The corresponding textural artifacts are clearly visible in the model. The EigenFaired mesh better approximates the geometrical structure of the building. Its texture-mapped model is devoid of artifacts at the same time.

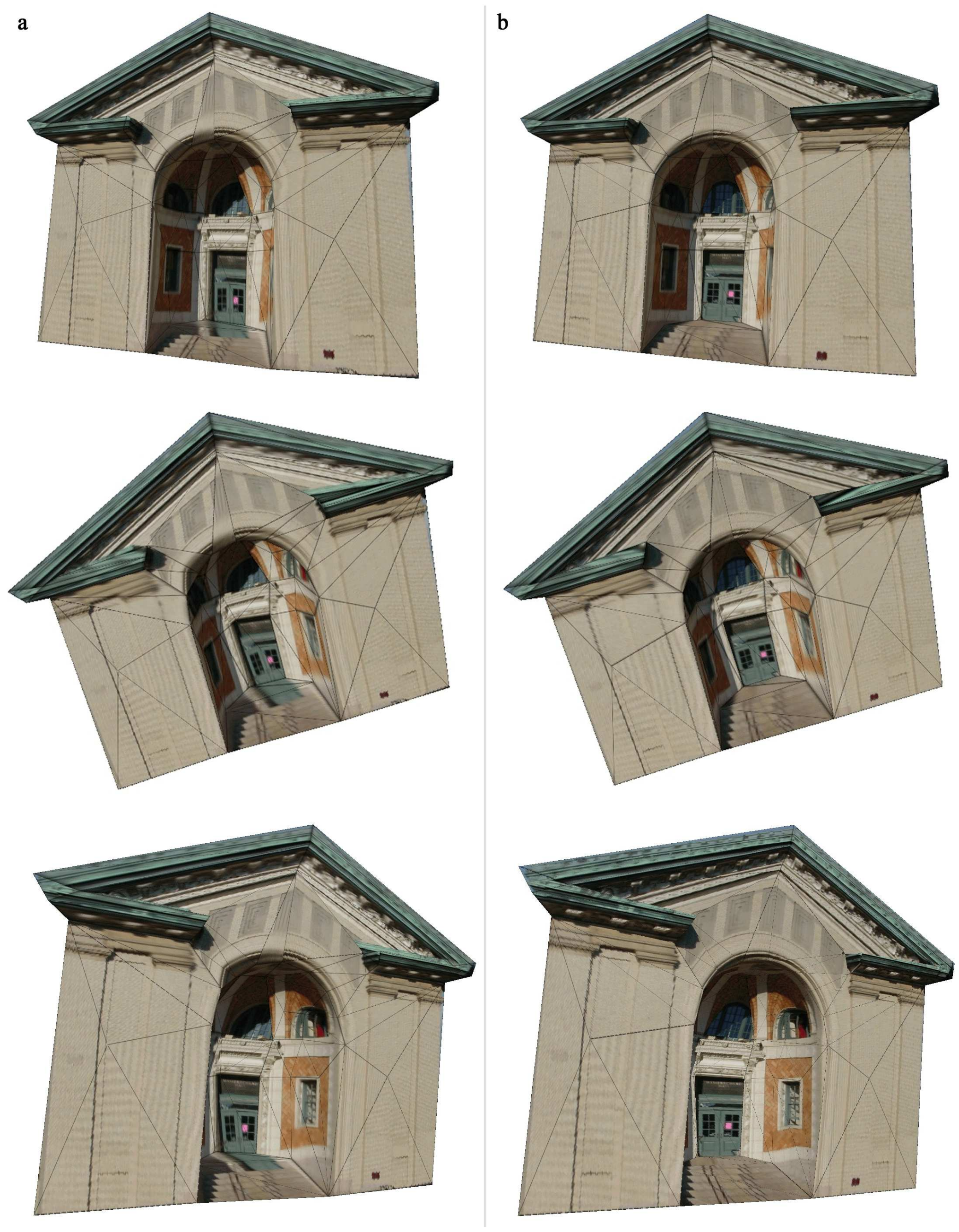


Fig. 14. Shown in column **a** are views of the model for the initial position of vertices. Views corresponding to the EigenFaired model are in column **b** on right. Note that the initial model has textural artifacts especially prominent along the curved dome, facade edges, and door steps. Artifacts in 3-D modeling are prominent when models are viewed from oblique angles. Renderings of the EigenFaired model, on the other hand, are devoid of artifacts.

Geometrical inconsistencies in modeling the mesh lead to textural artifacts [Figure 13 and 14]. Re-projections of the model into the images are computed from the eigenspace representation. For generating a texture map for a mesh face that can be viewed from any direction we use the weighted average of all the image areas corresponding to that face. We could also use the image area that has the strongest component in eigenspace to texture map the face. This image area is the one that best approximates the texture of the face when being viewed from that camera pose. For a planar surface patch, this image area corresponds to the corresponds to the one that has the largest projection area.

After EigenFairing the initial model, the 3-D vertices correspond to actual corners, and edges of the mesh are aligned with physical edges. Figure 11(b) shows projections of the EigenFaired mesh overlaid on images. Edges of the mesh, especially at the outer curve of the dome and sides of the columns exactly match with the corresponding line features in images. Geometrical inconsistencies, which are more conspicuous when the model is viewed at oblique angles, are absent. Texture on different mesh faces in flat areas such as the pediment– the triangular area (also called the gable) on the top part of the building–is well aligned as the mesh faces lie in the same plane [Figure 13]. The cornice–the green decorative border around the pediment just below the eaves of the roof–do not contribute texture smears to the gable in the background anymore [Figure 14, third row]. In the same figure, the protrusions that support the green cornice are more clearly visible in the EigenFaired model. Not only is the facade better modeled in a geometrical sense but is devoid of textural artifacts and smears.

6.2 Image and Depth Coherence

Once a piece-wise planar approximation has been found for the surface, we are interested in further refining it using the epitexture constraint. Improving the model further by accounting for finer details requires the use of texture and depth priors. Acquiring finer details in geometry is difficult due to resolution of the imaging system as well as occlusions of local patches. Constructing a finer mesh where a few mesh faces approximate a larger surface patch runs into the problem of a face being occluded, partially or fully, by other mesh faces. Therefore, the model has to be regularized by using depth and texture priors. The epitexture constraint allows us to apply this regularization and generates high-quality models.

Textured Material	$\hat{\mu}_\lambda$	$\hat{r}$
Peacock Feather	3.6383	1.83
Sponge	2.5524	1.80
Moss	3.3034	1.79
Aluminum Foil	5.9935	1.74
Limestone	10.1810	1.71
Orange Peel	2.9299	1.78
Lettuce Leaf	1.9154	1.67
Insulation	3.4668	1.77
Straw	3.5423	1.89
Soleirolia Plant	4.5473	1.68
Cracker B	2.7904	1.71
Concrete C	5.0568	1.80
Roof Shingle Zoomed	0.9084	1.63
Wood A	1.7293	1.91
Crumpled Paper	6.7874	1.72
Roof Shingle	2.2198	1.73
Salt Crystals	8.4129	1.76
White Bread	2.6431	1.74
Plaster B Zoomed	2.3631	1.85
Tree Bark	3.0355	1.68
Corn Husk	1.6202	1.95
Artificial Grass	1.2796	1.33
Styrofoam	7.1993	1.71
Leather	1.4674	1.63
Cracker A	4.0723	1.92
Slate B	5.7642	1.66
Slate A	14.9458	1.66
Human Skin	5.2270	1.97
Rug B	2.9524	1.67
Lambswool	2.8366	1.83
Felt	4.8278	1.84

Textured Material	$\hat{\mu}_\lambda$	$\hat{r}$
Rough Tile	1.4488	1.69
Rabbit Fur	4.0641	1.94
Stones	5.4577	1.69
Terrycloth	1.0931	1.71
Plaster B	3.9088	1.67
Rough Paper	1.5830	2.00
Loofah	7.4425	1.83
Concrete B	2.5754	1.74
Brown Bread	2.0260	1.64
Linen	6.0057	1.75
Wood B	2.0817	1.70
Brick B	3.3923	1.56
Polyester	2.9202	1.78
Quarry Tile	2.4707	1.74
Corduroy	4.2922	1.00
Frosted Glass	1.2742	1.98
Concrete A	1.8105	1.50
Cotton	4.9368	1.86
Cork	3.4865	1.63
Brick A	4.2827	1.64
Pebbles	3.1031	1.63
Polyester Zoomed	1.8949	1.62
Velvet	2.5634	1.57
Plaster A	7.9051	1.64
Rug A	4.9379	1.61
Sandpaper	5.8745	1.59
Rough Paper Zoomed	6.6496	1.51
Rough Plastic	3.4087	1.36
Painted Spheres	1.9147	1.00
Ribbed Paper	3.3985	1.00

Table 6.1. Generalized Gaussian MRF parameters for the depth-from-texture Prior for textures from the CUReT database.

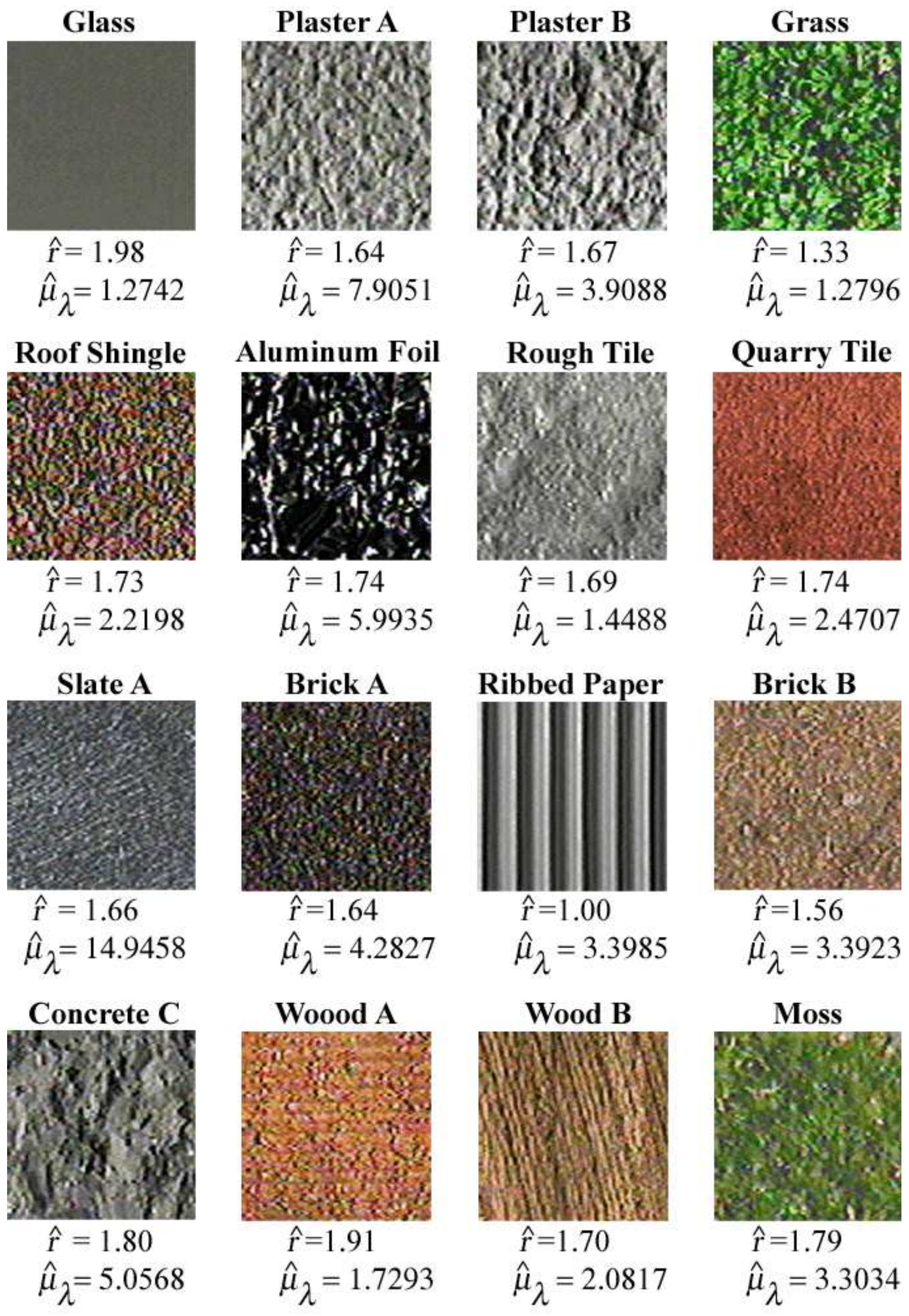

Glass
$\hat{r}$= 1.98
$\hat{\mu}_\lambda$= 1.2742
Plaster A
$\hat{r}$= 1.64
$\hat{\mu}_\lambda$ = 7.9051
Plaster B
$\hat{r}$= 1.67
$\hat{\mu}_\lambda$ = 3.9088
Grass
$\hat{r}$= 1.33
$\hat{\mu}_\lambda$ = 1.2796
Roof Shingle
$\hat{r}$= 1.73
$\hat{\mu}_\lambda$= 2.2198
Aluminum Foil
$\hat{r}$= 1.74
$\hat{\mu}_\lambda$ = 5.9935
Rough Tile
$\hat{r}$= 1.69
$\hat{\mu}_\lambda$ = 1.4488
Quarry Tile
$\hat{r}$= 1.74
$\hat{\mu}_\lambda$ = 2.4707
Slate A
$\hat{r}$ = 1.66
$\hat{\mu}_\lambda$= 14.9458
Brick A
$\hat{r}$=1.64
$\hat{\mu}_\lambda$ = 4.2827
Ribbed Paper
$\hat{r}$=1.00
$\hat{\mu}_\lambda$ = 3.3985
Brick B
$\hat{r}$=1.56
$\hat{\mu}_\lambda$ = 3.3923
Concrete C
$\hat{r}$ = 1.80
$\hat{\mu}_\lambda$= 5.0568
Woood A
$\hat{r}$=1.91
$\hat{\mu}_\lambda$ = 1.7293
Wood B
$\hat{r}$=1.70
$\hat{\mu}_\lambda$ = 2.0817
Moss
$\hat{r}$=1.79
$\hat{\mu}_\lambda$ = 3.3034

Fig. 15. Sixteen textures from the CUReT database and the associated GGMRF Parameters for the depth-from-texture prior.

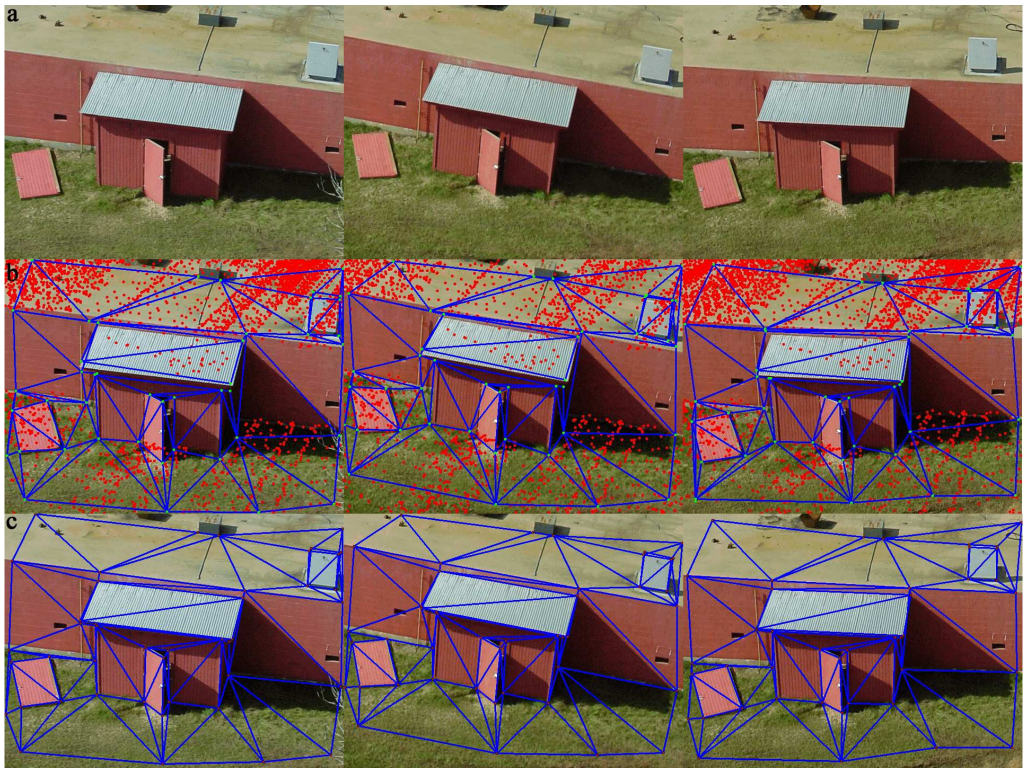


Fig. 16. Three of the set of six images are shown here. Row **a** shows the original images. In row **b** the images are superimposed with the range data collected from a helicopter flying over the scene. Notice that range data is absent for regions that are occluded from the view of the sensor. The mesh shown in the row is generated by selecting 3-D features from the range data and 2-D image features for regions that have substantial range data missing. The mesh is a close but not accurate approximation of the structure of the barn. Row **c** shows the EigenFaired mesh generated by moving the vertices of the initial mesh to accurately model the barn.

6.2.1 Epitexture Constraint using Depth and Texture Priors

Figure 16 shows three out of a set of six images. The range data that is superimposed on the images was scanned by a range scanner aboard an autonomous helicopter. Images of the same scene were registered with the images using manual correspondences. Parts of the scene are not scanned due to clutter, occlusion, and the flight pattern of the helicopter. Selecting 3-D points at surface discontinuities does not lead to a mesh that approximates the scene well. Therefore, we selected a few additional points based on image features to add to the set of vertices. Vertices were triangulated using image coherence. Even with the additional vertices the initial mesh does not model the surface accurately because the 3-D points from range data and image features do not correspond

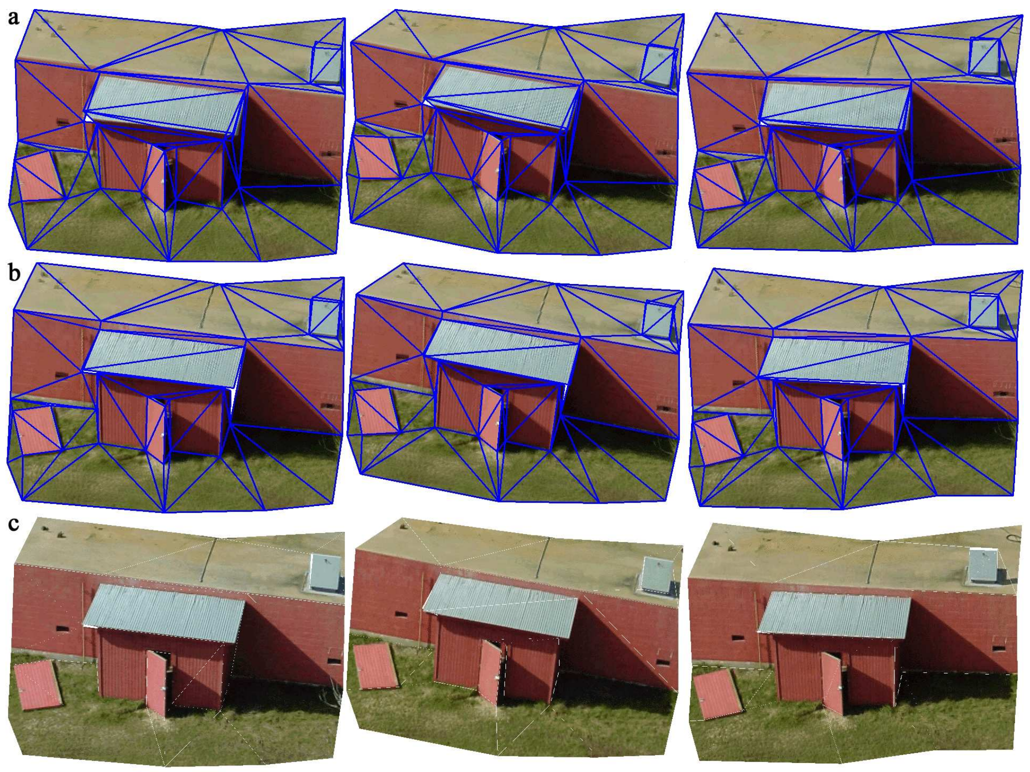


Fig. 17. Row (a) shows the re-projections of the initial model generated by image coherent triangulation. Re-projections of the EigenFaired mesh into the same images are shown in row (b) Notice the grassy ground is fuzzy as the planar mesh faces do not accurately model the curved nature of the surface. In row (c) the images are re-projections of the dense point-based model that is generated by the Epitexture constraint.

to strategic corners. EigenFairing the vertices improves the model considerably. Figure 17 shows re-projected images of the model while Figure 18 shows their absolute-value differences with the actual images. The EigenFaired model has clearer texture on the mesh and this is reflected in the re-projected images, especially on the corrugated roof and the textured ground.

The EigenFaired mesh still has imperfections in the sense that points within a face lie on a plane. The patch that is being approximated by the face might not be planar and have fine-scale geometry. We constructed isodepth maps for each mesh face by considering pair-wise images from the set of given images. Slices were determined in the isodepth maps to minimize the total re-projection error between the carved-out intensity values and the image area of the actual image.

The crucial issue in regularization of the model for generating fine details is the texture prior. Texture in the neighborhood of a point on the model also determines the range

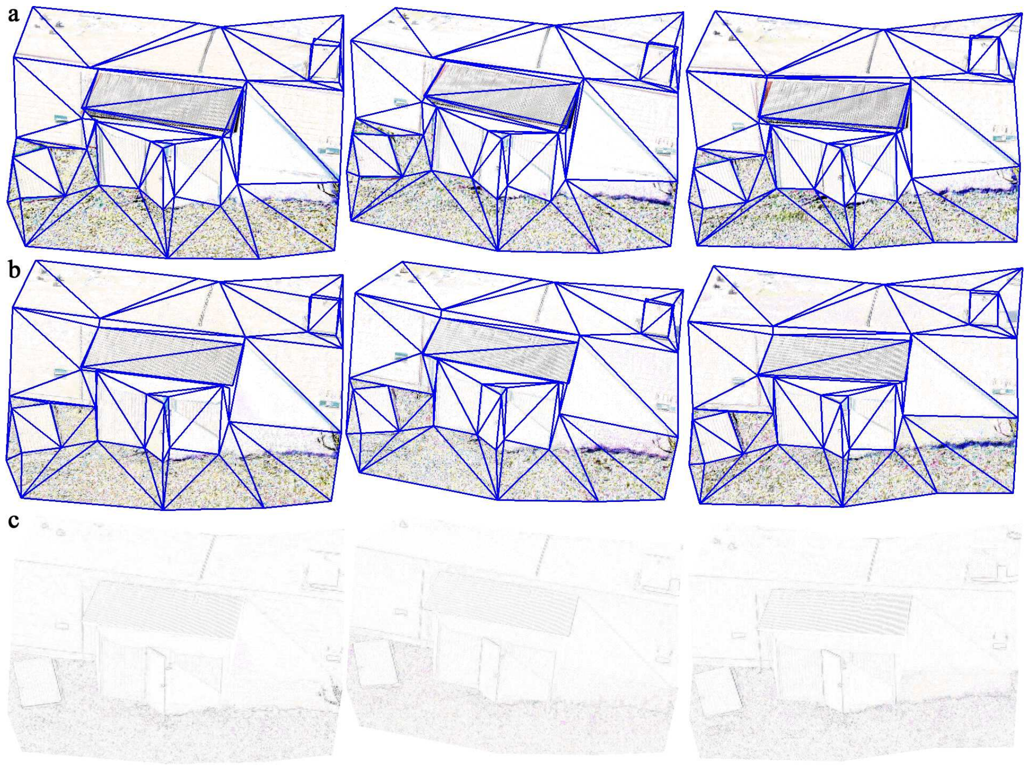


Fig. 18. The re-projection error of the models are shown in the three rows. Row **a** is the difference between re-projections of the initial mesh and the original images while row **b** is the re-projection error of the EigenFaired mesh. Row **c** shows the difference between the re-projections of the point-based model generated using the Epitexture constraint and the original images. The curved and "bumpy" nature of the grassy ground is better modeled by the point-cloud model that gives low re-projection error of pixels corresponding to points on the ground.

of depth values that the point can have in isodepth maps. Textures from the CUReT database (Dana et al., 1999; Dana and Nayar, 1999) were used for generating the texture and shape-from-texture priors. The parameters $\hat{\mu}_\lambda, \hat{r}$ for the GGMRF prior component in equation 4.27 are associated with each texture in the database [Table 6.1]. Figure 15 shows 14 textures from the texture database. The parameters determine the qualitative property of *patch roughness*. Roughness, as defined here, is purely a macroscopic property. It is not indications of the microscopic detail of a surface patch that is assumed to cause the patch to be locally Lambertian.

Starting with the EigenFaired mesh, we model details for points within each face from its isodepth maps. The re-projections of the point-based model into images is shown in Figure 17(c) and the corresponding re-projection error images is in Figure

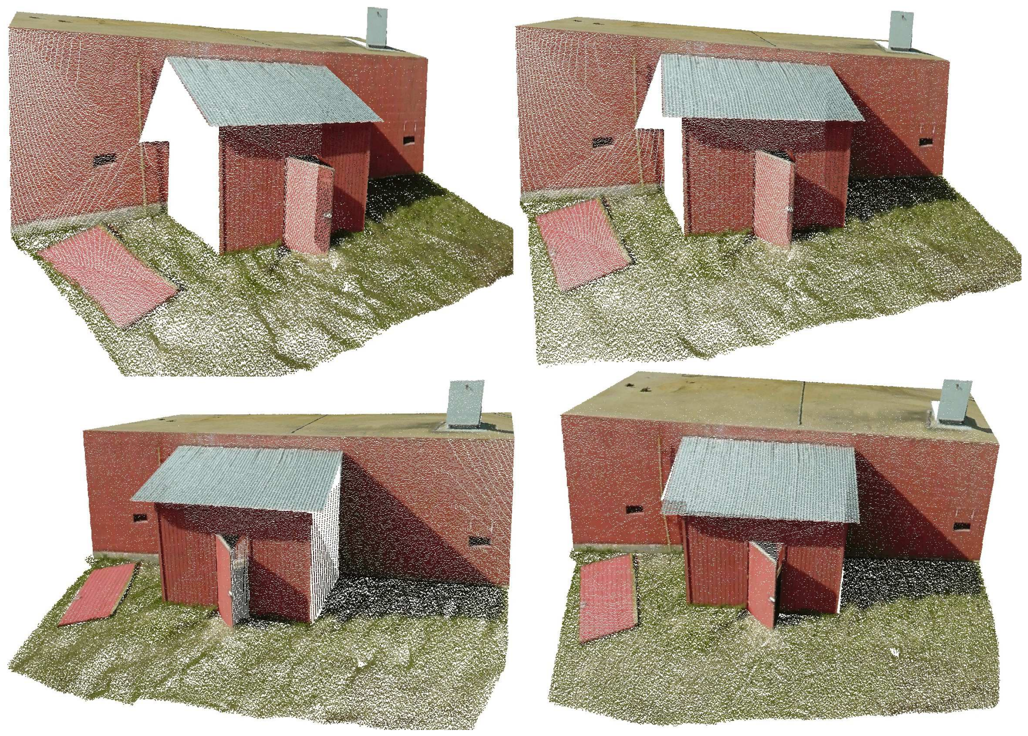


Fig. 19. New views of the dense point-cloud 3-D model. The 3-D model is created by carving out slices of isodepth maps for faces of the EigenFaired mesh using the Epitexture constraint. For the given proximity of rendering viewpoint and the size of points, the curvilinearity of patches can be clearly observed.

18(c). The corrugated roof, doors, and the textured grass have low re-projection error compared to their EigenFaired predecessors.

Certain artifacts are observed at edges of different patches because they have been carved out of different sets of isodepth maps. We applied boundary conditions in isodepth maps for adjacent patches. In the context of carving out slices, the constraints were imposed at the patch boundaries for depth values, tangents, and curvatures. This worked well for continuity and smoothness for adjacent patches; however, there are some points at those boundaries that overlapped. In addition, patches can be only implemented in a rectangular area and some points at the boundaries that did not belong to the patch crept into the final model. This problem is a result of the digitization of pixel coordinates that we discuss in *Appendix A: The Curse of the Pixel*.

Renderings of the final model are shown in Figure 20. The patch that approximates the door lying on the ground is planar when compared to it grassy surroundings. Surface depressions due to tire tracks and grass undulations have been modeled well due to the

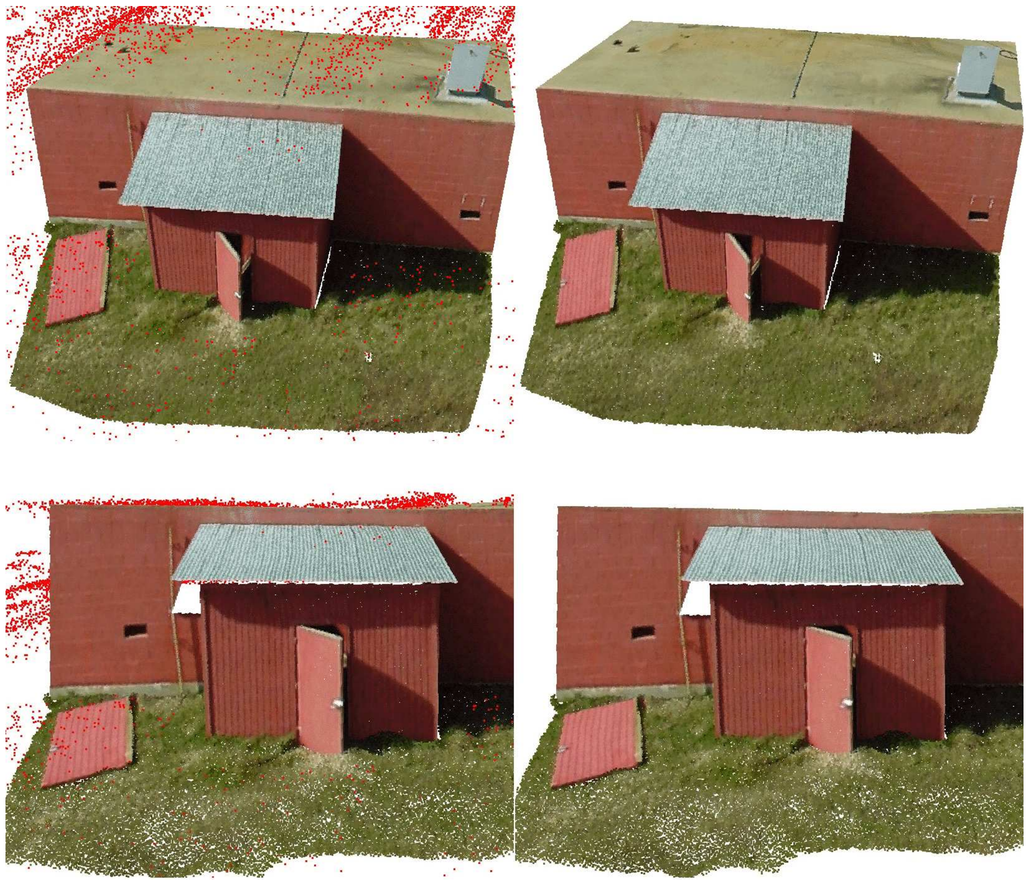


Fig. 20. On left are views of the model superimposed with the range data points. Range data is often cluttered or partially resolved in addition to noise present in 3-D positions. The slightly curved nature of the roof top cannot be clearly observed in the range data due to the presence of noise. However, the slowly varying curved attribute is well modeled by the dense 3-D point surface. The images of the model without range data are on right.

texture priors used in regularizing 3-D details. Figure 20 shows the dense range data superimposed on the model. Slowly-varying attributes of structure as well as sharp details have been captured by the model. The slightly curved nature of the roof and the ground are reflected in the range data. Sharp features at edges of the door and occluding parts of the wall and corrugated roof are well described by the model. Not only is the texture more coherent with actual images of the scene but the depth variations are consistent with observed range data as well.

7 Conclusion

In modeling a 3-D surface from a set of 2-D images we are making an attempt to reverse the scene-to-image mapping. This mapping is made reversible by imposing enough constraints on the space of possible models that best explain a set of images. One of the most important constraints is based on the epipolar geometry of sampled 3-D points on the surface as viewed in multiple images. However, this constraint does not include prior knowledge of surface structure or texture. Knowledge of structure and texture properties of local surface patches is essential for accurately modeling natural scenes. In this thesis, we have combined constraints between epipolar lines for neighboring points that form a 3-D surface patch with texture and structure properties by way of priors. We start from a textured mesh face, which is a planar approximation of the surface patch, and use epipolar geometry to build a search space to which the patch belongs. Texture and structure priors are then used to carve out the 3-D surface patch. Points of the resulting surface patch, in addition to satisfying the epipolar constraint, best account for what was observed in images–a property we have termed as the Epitexture constraint.

The textured mesh, which we initially start with, is build from triangulating a set of points. These points are derived from 2-D image features or points of discontinuities in dense 3-D range data. The triangulation best explains what was observed in images by maximizing image coherence. At this level of surface modeling, faces of the mesh are mapped with both 2-D and 3-D textures without making any distinction between the two. 3-D textures are results of fine-scale surface geometry and are modeled by using priors and the Epitexture constraint. Most 2-D image features or points of dense 3-D range data do not sample surfaces at their critical points–a requirement for generating a high-quality model. This is also a necessity for further refinement of the model by using the Epitexture constraint. The 3-D points serve as vertices of the mesh and are relocated to the desired critical points by exploiting the textures on faces of the mesh. We show that textural and geometric attributes of models are interdependent and it is indeed possible to use one attribute to refine the other. Modeling results for a few natural scenes, based on image and depth coherence, have been presented.

7.1 Summary of Contributions

The main contribution of this thesis is the confluence of image and depth coherence to describe 3-D scenes. The description of a 3-D scene was done by modeling its geometrical and textural details. The key idea behind image and depth coherence is the interdependence of surface geometry and surface texture. The following are the contributions of this thesis.

Coherence: We have used the coherence of a 3-D model with respect to observed images and depth to achieve an accurate description of the scene. The choice of a surface description depends on both the given data and the very level of description we seek. It can be a piecewise-linear approximation of surface geometry and 2-D texture or a dense point-based approximation of fine-scale geometry (3-D texture). In both cases, the 3-D model that describes the unknown surface has to account for any observations of the surface, be it images, depth, or appearance.

EigenFairing: Points used as vertices of a mesh that approximate a surface are derived from 2-D image features or discontinuities in dense 3-D depth data. However, these vertices do not usually sample the surface at its critical points such as corners and edges. This results in a poor modeling of the surface, both geometrically and texturally. We have shown that the model can be refined or faired by relocating vertices of the mesh to the desired critical points. Relocating a vertex was done by using texture on faces of the mesh common to the particular vertex.

Isodepth Maps and Epitexture Constraint: 3-D points observed in multiple images are related to each other via the Epipolar constraint. Pixel intensities along epipolar lines for neighboring points that form a 3-D surface patch can used to build a search space–an isodepth map–to which the surface patch belongs. This is particularly effective in modeling a patch that is not planar or has fine-scale geometry. The 3-D patch can be modeled by a slice of pixel intensities and depth values carved out of the isodepth map. We have shown that this slice is constrained by the texture of the patch as viewed in multiple images, a property we have termed as the Epitexture constraint.

Regularization using Texture and Depth Priors: We have shown that in addition to the Epitexture constraint, depth and texture priors can be effectively used to carve out slices of isodepth maps. These priors help in regularization that are essential if high-quality models are to be computed.

Combining Cues using Crofton's Theorem: We have proposed a variational approach whereby different descriptors or cues of an observed scene can be combined to form a global coherence measure. Appearance and other view-dependent attributes of a scene consisting of multiple objects can be modeled using such coherence measures.

7.2 Future Directions

This section sets directions for future work by proposing research issues of what, we believe, would make image and depth coherence essential for any 3-D surface modeling application.

Image-coherent Mesh Generation: The search for the best triangulation, both in terms of edges and position of vertices, depends on the error between the texture of the surface and that of the model. A cost function that is based on the texture-dependent re-projection error is optimized for image coherence. In low-contrast areas, such as uniform or little texture and washed-out specular highlights, the optimization proves to be difficult. For such a situation, heuristics based on shading and silhouettes can be used to provide additional information. These heuristics can also be used for the initial triangulation, which is important for convergence of the optimization process. Image coherence can be manually augmented with heuristics such as determining areas of specular highlights or adding tighter error thresholds for silhouettes. But, we would like to avoid ad hoc hand-tuned heuristics often used in many 3-D modeling applications.

We are interested in guiding the development of an initial mesh by reasoning directly from view-dependencies of the surface. Image coherence, as we have shown, can be used to develop the initial mesh. However, the initial mesh needs to be a close approximation of the surface, even in areas of occlusion and low-contrast textures. Finding the best piecewise-linear approximation of the surface helps in carving out the slice in isodepth maps to model fine-scale structure of the surface. It is, therefore, consequential to find the initial triangulation that converges to the point-based model in a reasonable amount of time without using complex heuristics.

Priors: Priors are essential for regularizing the image-to-surface inverse problem. Texture modeling of natural scenes requires a comprehensive library of exemplar image patches. It would be interesting to study the interdependence of texture and depth priors–an area that is being recently studied in terms of 3-D textures. We have used GGMRF's to derive a form for the depth prior; however, parameter estimation for GGMRF's is mathematically and numerically challenging. In addition, these parameters cannot be directly computed for a set of pixel areas taken from images: an approach that would be useful for adding new exemplars to our texture database. Knowledge of the most efficient forms of depth, texture, and depth-from-texture priors (efficient in terms of speed of learning, storage, update rate and feasibility of implementation) would be ideal for real-time vision. We believe that there are three priors involved in any visual process–texture, depth, and temporal priors. A promising research direction would be to model these priors and their interdependence.

Combining Multi-modal Cues: Image pixel intensities and depth values form different cues or descriptors for a visual system; pixel statistics, texture gradients and foreshortening, statistics of segmented color regions, optical flow, and local 3-D shape are a few examples of local cues or descriptors of a scene. We believe that a variational approach such as the Generalized Crofton's Theorem can combine several of these cues to form a global coherence measure. A global coherence measure can be effectively used in registration of images, recognition of discrete objects in a scene, and modeling appearance of 3-D scenes.

The Occluding and the Occluded: In many natural scenes, sudden changes in surface structure happen due to occlusions. As observed in many of our 3-D models, image coherence can detect these boundaries and locate false connections between the occluding and occluded surface patches. Mesh elements that do not correspond to real surface patches yield high re-projection error and are removed during the refinement procedure. However, it would be judicious to exclude these elements during mesh generation. This requires identifying pixels that belong to occluded and occluding patches at the overlapping boundary. Mesh elements modeling the two patches should have their edges aligned along the boundary instead of cutting across.

View-dependency of Eigentextures: We have not exactly been able to determine how the Eigentexture representation of a surface patch varies with view-point. We are interested in reconstructing the view of a patch from a new pose given its eigentexture representation. It would seem fruitful to investigate the relationship between Eigentextures of 3-D textures due to fine-scale geometry and its view-dependent appearance. To our knowledge, this relationship has not been studied yet.

7.3 Epilogue

In addition to albedo or color variations, a surface has structure as in shape and fine-scale surface corrugations. These are two levels of surface geometries. Take the example of a unplastered wall; it has distinct corners that are at critical points of the surface as well as surface corrugations due to layers of bricks. Three-dimensional surface corrugations on globally smooth surfaces give rise to brightness modulations of intensity patterns. Systematic variations of such 3-D image textures are a function of illumination and viewpoint–a property that we can exploit in isodepth maps. As pointed out by (Dana et al., 1999), there are essentially two different categories of *texture*. 2-D image textures are due to flat pattern or albedo variations on smooth surfaces, such as wallpapered or marbled surfaces, painted designs or prints, and shadows. Such textures are used in "shape-from-texture" and "texture-mapping" techniques, which are due to the fact that the texture itself has no 3-D details but effects of albedo variations. However, majority of textures in images of natural scenes are due to the 3-D details of surfaces that can only be resolved visually or by a camera. Such 3-D textures exist even if the surface has uniform albedo, such as plastered or brick walls, grass and foliage, gravel, and rocky surfaces.

2-D textures provide features from which one can draw inferences about the shape of the global surface. In this case the 3-D fine-scale structure of the surface is ignored. 3-D textures depend strongly of illumination and viewpoints and do not provide invariant features. On the other hand, they do provide information concerning local illumination and the nature of 3-D surface details and corrugations.

Other than 2-D textures, distinct 3-D surface discontinuities such as edges and corners often provide image features that capture structure or shape of the global surface. These surface discontinuities do not include fine geometric details that give rise to 3-D textures. In our research, we have made a clear distinction between the two– surface critical points and fine-scale surface geometry. However, our ultimate goal is to model both.

When a mesh is generated to approximate a surface, we use features from 2-D textures and 3-D critical points to generate the mesh. At that level it is not expedient to go for the level of detail in modeling surface geometry due to 3-D textures. This is primarily because of the fact that any optimization process to model fine details is bound to get trapped in local extrema.

Subdiving a mesh to model finer details is paradoxical. To what level would one sub-divide a mesh: to the level of 2-D textures or 3-D textures? In our opinion, a mesh should be fine enough to the extent of capturing 2-D textures and 3-D critical points. Any further , the mesh loses its significance of being a simplicial representation of the surface. Moreover, if a mesh has smaller and refined elements but does not accurately represent 3-D critical points at its vertices and 2-D textures on its faces, it is a poor approximation and an expensive model, in terms of generating, representing, and rendering new views of the model. A mesh that models a surface at the level of fine-scale geometry has artifacts due to fine-scale occlusions. Polygonal meshes created from dense range data are examples of this.

Would sub-dividing a mesh at regions without significant texture result in a better model or noticeable improvement in representing the surface? This query leads to yet another salient observation– one can accurately model a surface only to the degree or *content* of its 2-D and 3-D texture. The geometrical accuracy of a model depends on how textured the surface is. This is clearly observed in the process of EigenFairing which is more effective in regions that have enough texture to resolve geometrical inaccuracies of the model. Texture representation of a surface is dependent on illumination, viewpoint and the imaging process. In other words, there is a limit to the geometrical level of detail that a surface can be resolved in 3-D space. This limit depends on the degree of textural detail on the surface, the pixel size of the sensor and variation of the views of the surface. We can have smaller mesh faces provided we have finer texture within each face. The level of discretization and digitization of light patterns captured by an imaging sensor in form of pixels pose another limit on how fine the texture can be and, therefore, on how accurately a surface may be modeled. There is no end to the quest for perfection.

A The Curse of the Pixel

Interacting with a digital camera in computer vision research can, at times, be very unsettling. The light intensity pattern projected onto the imaging surface of a camera or an eye lies in the real continuum. Despite this, we are rudely awakened to the fact that we must, in practice, content ourselves with a pixelized form of the intensity pattern. This consequence of discrete and finite-precision representation introduces discretization and quantization errors into numerical computation. Moreover, the error is introduced right upon input of many vision algorithms.

This appended article separates and brings out two distinct errors involved in treating a light intensity pattern as a pixelized image. The first error, that we shall call the *Discretization* error, is due to spatial discretization or dividing the continuous pattern into an array of intensity values. The second form of error, the *Digitization* error, originates from the quantization or digitization of the intensity values in each cell of the image array. A pixel of an image, therefore, is a discrete cell with a quantized value of intensity. These two distinct errors depend on the pixel size. We shall show that the effects on the errors by varying the size of a pixel are complementary to each other.

A.1 Problem

A.1.1 The Problem of Discretization

Let us consider a single dimensional intensity signal, $I(x)$, that is continuously differentiable on the interval $[u_0, u_1]$. Define the operator D as

$$D(I) \equiv I'(u)$$

for some point u in $[u_0, u_1]$. Since x is discretized within the interval $[u_0, u_1]$, $D(I)$ can only be its numerical approximation;

$$D(I) = D_h + ch^r + o(h^r) \tag{A.1}$$

where D_h is our finite difference approximation, c is the asymptotic error constant, h is the pixel width, and r is the order of approximation [Conte and de Boor (1980), pg. 300].

The portion of error accounted for in equation (A.1) is called the *pixel discretization* error. It seems that we should be able to calculate $D(I)$ to any desired accuracy merely by calculating D_h for small enough h. However, the fact that all CCD cameras, digitizers and computers have fixed and limited number intensity levels for a pixel, together with loss of significance caused when nearly equal quantities are subtracted, combine to make high accuracy difficult to obtain. Indeed, for a system with a fixed pixel size and for a given intensity pattern, there is an optimum value of h below which the approximation becomes worse.

A.1.2 The Problem of Digitization

The additional error due to the finite set of pixels is referred to as the *pixel digitization* error. The dilemma lies in the fact that, while the discretization error goes to zero with h, the digitization error becomes unbounded. Subsequently, error terms such as that appearing in equation (A.1) present the illusion that unlimited accuracy is achievable simply by having finer pixels.

For the sake of definiteness, let us consider the central-difference formula:

$$D_h = \frac{I(u+h) - I(u-h)}{2h} \tag{A.2}$$

with $c = -I'''(u)/6$ and $r = 2$. If we account for the digitization error, $E_\pm$, incurred by rounding off intensity values, we will use the values $I(u+h) + E_+$ and $I(u-h) + E_-$ instead of $I(u+h)$ and $I(u-h)$. Therefore, $D_h(I)$, the computed derivative of the intensity function at u, is

$$\overline{D}_h(I) = \frac{I(u+h) + E_+ - I(u-h) - E_-}{2h} \tag{A.3}$$

A.1.3 The Two Errors

Equation (A.3) can be written as

$$D_h = \overline{D}_h(I) + \overline{E}_h \tag{A.4}$$

where the digitization error $\overline{E}_h = -\frac{E_+ - E_-}{2h}$. Substituting this in equation (A.1) yields

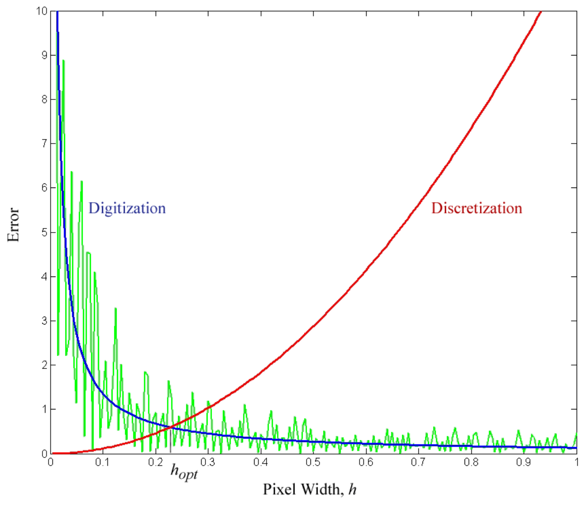


Fig. 1. The trade-off in pixel refinement. h_{opt}, the optimal pixel width, cannot be precisely determined.

$$D(I) = \overline{D}_h(I) + \overline{E}_h + ch^r + o(h^r) \tag{A.5}$$

where $c = -I'''(u)/6$ and $r = 2$. Thus, we determine that the actual error is composed of two quite dissimilar terms: a parabolic discretization error and a hyperbolic digitization error. This implies that there exists an optimum pixel width, h_{opt}, below which the total error increases. The perplexing facet of this phenomenon is that the precise determination of h_{opt} is impossible in practice.

Table A.1. Discretization and digitization errors for the derivative of the intensity function $I(u) = 127.5 * (sin(u) + 1)$ at $u = 1$.

h	$D_h(I)$	ch^2	$o(h^2)$	$\overline{D}_h(I)$	$\overline{E}_h$	$\|ch^2\| + \|\overline{E}_h\|$
1.280	51.55962	18.81117	-1.48224	51.56250	-0.00288	18.81404
0.640	64.28113	4.70279	-0.09538	64.84375	-0.56262	5.26541
0.320	67.71885	1.17570	-0.00600	67.18750	0.53135	1.70705
0.160	68.59500	0.29392	-0.00038	68.75000	-0.15500	0.44893
0.080	68.81509	0.07348	-0.00002	68.75000	0.06509	0.13857
0.040	68.87018	0.01837	-0.00000	62.50000	6.37018	6.38855
0.020	68.88395	0.00459	-0.00000	75.00000	-6.11605	6.12064
0.010	68.88740	0.00115	-0.00000	50.00000	18.88740	18.88854

Let us consider a numerical example. Consider a 1-D intensity function, $I(u) = 127.5 * (sin(u) + 1)$, incident on a 1-D pixel array. Values of the intensity function, as you may deduce, lie between 0.0 and 255.0. However, the values get rounded off to integers between 0 and 255 due to the CCD array or digitizer. Table A.1 shows $\overline{D}_h(I)$ — the derivative of the intensity function calculated at $u = 1$ using the rounded-off intensity values for different values of h. The exact derivative of the function at $u = 1$ is $D(I_{(u=1)}) = 68.88854$. The derivative, $D_h = (I(u+h)-I(u-h))/2h$, computed using the continuous values consists of the discretization errors; ch^2 where $c = -I'''/6 = 11.48142$ and $o(h^2)$.

With smaller pixel-width h, $D_h(I)$ gets closer to the real value of the derivative, $D(I_{(u=1)}) = 68.88854$. As the table shows, ch^2, the dominant component of the error goes to zero with h and the $o(h^2)$ component goes to zero faster. These components that together form the discretization error are unknown in a real situation. The table shows the variation of the calculable derivative $\overline{D}_h(I)$. $\overline{E}_h$, the digitization error increases in magnitude with decreasing h. The total error — summation of the digitization and discretization errors — at first decreases and then increases with finer pixelization. The errors are plotted in Figure 1. The optimal pixel width — the pixel width for which the total error is minimal — cannot be determined.

A.2 Solution using Richardson Extrapolation

Table A.2. Derivative of the intensity function $I(u) = 127.5 * (sin(u) + 1)$ at $u = 1$ computed using the Richardson Extrapolant. The actual value of the derivative, $I'(u = 1)$, is 68.88854

h	$D_h(I)$	$D_R \equiv \frac{4D_h - D_{2h}}{3}$	$R_h \equiv \frac{D_h - D_{2h}}{D_{h/2} - D_h}$	$\overline{D}_h(I)$	$\overline{D}_R \equiv \frac{4\overline{D}_h - \overline{D}_{2h}}{3}$
1.280	51.55962	63.81852	2.89	51.56250	63.80208
0.640	64.28113	68.52163	3.70	64.84375	69.27083
0.320	67.71885	68.86476	3.92	67.18750	67.96875
0.160	68.59500	68.88704	3.98	68.75000	69.27083
0.080	68.81509	68.88845	4.00	68.75000	68.75000
0.040	68.87018	68.88854	4.00	62.50000	60.41667
0.020	68.88395	68.88854	4.00	75.00000	79.16667
0.010	68.88740	68.88854	4.00	50.00000	41.66667

Fortunately, we can obtain an accurate derivative approximation without excessively refining the pixel. This is achieved by a formulation called the *Richardson Extrapolation*. Rather than derive a generalized form of this, we instead focus on the particular case of the central difference approximation of the first derivative.

Consider the particular central-difference formula where $c = -I'''(u)/6$ and $r = 2$. The constant c does not depend on h, and equation (A.1) reads

$$D(I) = D_h + ch^2 + o(h^2) \tag{A.6}$$

Substituting $2h$ for h in equation (A.6), we get

$$D(I) = D_{2h} + 4ch^2 + o(h^2) \tag{A.7}$$

Subtracting the above equation from equation (A.6) gives

$$ch^2 = \frac{D_h - D_{2h}}{3} + o(h^2) \tag{A.8}$$

The above equation states that, for sufficiently small h, the computable value

$$\frac{D_h - D_{2h}}{3}$$

is a good estimate for the unknown dominant error component ch^2. Note that the catch, of course, lies in the phrase “for sufficiently small h”.

If ch^2 is indeed the dominant error component, i.e., if the $o(h^2)$ is small compared to ch^2, then

$$ch^2 \approx \frac{D_h - D_{2h}}{3} \quad \text{and} \quad c\left(\frac{h}{2}\right)^2 \approx \frac{D_{h/2} - D_h}{3}.$$

Inserting the value of ch^2 in equation (A.6), we have

$$D(I) = D_R + o(h^2); \quad D_R \equiv \frac{4D_h - D_{2h}}{3} \tag{A.9}$$

D_R is the *Richardson extrapolant*, which constitutes a refined approximation so long as

$$R_h \equiv \frac{D_h - D_{2h}}{D_{h/2} - D_h} \approx 4$$

It can be shown that D_R is a higher-order approximation to $D(I)$ than is D_h [Conte and de Boor (1980), pg. 337]. Deriving such a higher-order approximation from two lower-order approximations is called *extrapolation to the limit*, or *to zero-grid size*.

The values computed by equations (A.8) and (A.9) are themselves subjected to the pixel-intensity digitization error, but, since we would presumably by using a larger value of h than for equation (A.3), the effect of the digitization error will be less pronounced. Substituting for the digitization error in equation (A.9), we have

$$D_R = \frac{4\overline{D}_h - \overline{D}_{2h}}{3} + \frac{4\overline{E}_h - \overline{E}_{2h}}{3} \tag{A.10}$$

The extrapolant, $\overline{D}_R \equiv \frac{4\overline{D}_h - \overline{D}_{2h}}{3}$, takes the digitization error into account. In the absence of the knowledge of the true intensity of light incident at pixels, D_h and D_R cannot be ascertained. In addition, R_h cannot be determined from digitized intensity values. The values that can indeed be calculated from digitized pixel intensities are $\overline{D}_h$ and $\overline{D}_R$.

Table A.2 shows the derivatives, D_R and $\overline{D}_R$, calculated using Richardson extrapolation. The derivative computed by D_R is closer than D_h to the actual derivative value at any pixel width h. However, the derivative value, $\overline{D}_R$, accounting for the pixel digitization error gets close to the actual value at $h = 0.080$ after which the digitization error takes over. At any pixel width above $h = 0.080$, $\overline{D}_R$ is a better derivative approximation than $\overline{D}_h$. For any pixel width smaller than $h = 0.080$, the Richardson extrapolant is as good an approximation as $\overline{D}_h$. In fact, at smaller pixel widths the opposite is true.

Bibliography

Abbott, A. and Ahuja, N. (1993). Active stereo: Integrating disparity, vergence, focus, aperture, and calibration for surface estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(10):1007–1029.

Allen, P. and Stamos, I. (2000). 3-d model construction using range and image data. In *CVPR 2000*, volume 1, pages 531–536.

Allen, P. and Stamos, I. (2001). Registration of 3d with 2d imagery in urban environments. In *The Eighth International Conference on Computer Vision*.

Allezard, N., Dhome, M., and Jurie, F. (2000). Recognition of 3d textured objects by mixing view-based and model-based representations. In *International Conference on Pattern Recognition*, volume 1, pages 960–963.

Asada, M., Kimura, M., Taniguchi, Y., and Shirai, Y. (1992). Dynamic integration of height maps into a 3d world representation from range image sequences. *International Journal of Computer Vision*, 9(1):31–53.

Baltzakis, H., Argyros, A., and Trahanias, P. (2002). Fusion of range and visual data for the extraction of scene structure information. In *International Conference on Pattern Recognition*, Quebec City, Quebec, Canada.

Barron, J. L., Fleet, D. J., and Beauchemin, S. S. (1994). Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77.

Barrow, H. and Tenenbaum, J. (1978). Recovering intrinsic scene characteristics from images. In *Computer Vision Systems*, pages 3–26.

Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society*, B-48:259–302.

Black, M. J. and Jepson, A. D. (1998). Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *Image and Vision Computing*, 26(1):63–84.

Bouman, C. and Sauer, K. (1993). A generalized gaussian image model for edge-preserving map estimation. *IEEE Transactions on Image Processing*, 2(3):296–310.

Brandt, S. S. (2004). On the probabilistic epipolar geometry. In *British Machine Vision Conference*, London, United Kingdom. British Machine Vision Association.

Buchanan, T. (1988). The twisted cubic and camera calibration. *Computer Vision Graphics and Image Processing*, 42(1):130–132.

Buker, U. and Hartmann, G. (2000). Object representation: on combining viewer-centered and object-centered elements. In *International Conference on Pattern Recognition*, volume 1, pages 956–959.

Bulthoff, H. H. and Mallot, H. A. (1987). Interaction of different modules in depth perception. In *Proc. of the 1st International Conference on Computer Vision*, pages 295–305.

Carlsson, S. and Weinshall, D. (1998). Dual computation of projective shape and camera positions from multiple images. *International Journal of Computer Vision*, 27(3):227–241.

Chen, C.-Y. and Sara, R. (2000). Integration of photometric stereo and shape from occluding contours by fusing orientation and depth data. In *Biologically Motivated Computer Vision, 10th International Workshop on Theoritical Foundations of Computer Vision*, volume 2032 of *Lecture Notes in Computer Science*, pages 251–269. Springer Verlag.

Choe, Y. and Kashyap, R. (1991). 3-d shape from a shaded textural surface image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):907–919.

Clark, J. J. and Yuille, A. L. (1990). *Data Fusion for Sensory Information Processing Systems*. Kluwer Academic Publishers, Boston, MA.

Cohen, I., Cohen, L., and Ayache, N. (1992). Using deformable surfaces to segment 3-d images and infer differential structures. *Computer Vision Graphics and Image Processing*, 56(2):242–263.

Colosimo, A., Sarti, A., and Tubaro, S. (2001). Image-based object modeling: A multiresolution level-set approach. In *International Conference on Image Processing*, pages II: 181–184.

Conte, S. and de Boor, C. (1980). *Elementary Numerical Analysis: An Algorithmic Approach*. McGraw-Hill, New York.

Cootes, T., Edwards, G., and Taylor, C. (2001). Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685.

Cootes, T. and Taylor, C. (2001). Constrained active appearance models. In *International Conference on Computer Vision*, pages I: 748–754.

Cootes, T., Taylor, C., Cooper, D., and Graham, J. (1995). Active shape models: Their training and application. *Computer Vision and Image Understanding*, 61(1):38–59.

Crofton, M. W. (1885). On local probability. *Encyclopaedia Britannica*, 19(9th. Edition):768–788.

Cryer, J., Tsai, P., and Shah, M. (1993). Integration of shape from x modules: Combining stereo and shading. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 720–721.

Dana, K., van Ginneken, B., Nayar, S., and Koenderink, J. (1997a). Reflectance and texture of real-world surfaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 151–157.

Dana, K., van Ginneken, B., Nayar, S., and Koenderink, J. (1997b). Reflectance and texture of real-world surfaces. In *DARPA97*, pages 1419–1424.

Dana, K. J. and Nayar, S. K. (1998). Histogram model for 3D textures. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Dana, K. J. and Nayar, S. K. (1999). 3D textured surface modeling. In *WIAGMOR Workshop CVPR*.

Dana, K. J., van Ginneken, B., Nayar, S. K., and Koenderink, J. J. (1999). Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics*, 18(1):1–34.

Diehl, H. and Heipke, C. (1992). Surface reconstruction from data of digital line cameras by means of object based image matching. *International Society for Photogrammetry and Remote Sensing*, 29 B3:287–294.

Edelman, S., Intrator, N., and Jacobson, J. S. (Second International Workshop, BMVC 2002). Unsupervised learning of visual structure. In *Biologically Motivated Computer Vision*, volume 2525 of *Lecture Notes in Computer Science*, pages 629–642. Springer Verlag.

Edwards, G., Cootes, T., and Taylor, C. (1999). Advances in active appearance models. In *International Conference on Computer Vision*, pages 137–142.

Efros, A. A. and Leung, T. K. (1999). Texture synthesis by non-parametric sampling. In *International Conference on Computer Vision*, pages 1033–1038.

Epstein, R., Hallinan, P., and Yuille, A. (1995). 5$\pm$2 eigenimages suffice: An empirical investigation of low-dimensional lighting models. *IEEE Workshop on Physics-based Modeling in Computer Vision*, pages 108–116.

Erikson, C. and Manocha, D. (1999). Gaps: general and automatic polygonal simplification. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 79–88. ACM Press.

Esteban, C. and Schmitt, F. (2003). Silhouette and stereo fusion for 3d object modeling. In *Conference on Recent Advances in 3-D Digital Imaging and Modeling*, pages 46–53.

Faugeras, O. (1995). Stratification of 3d vision: Projective, affine and metric representations. *Journal of the Optical Society of America*, 12(3):465–484.

Faugeras, O. and Keriven, R. (1998). Variational-principles, surface evolution, pdes, level set methods, and the stereo problem. *IEEE Transactions on Image Processing*, 7(3):336–344.

Faugeras, O., Luong, Q., and Maybank, S. (1992). Camera self-calibration: Theory and experiments. In *European Conference on Computer Vision*, pages 321–334.

Faugeras, O., Luong, Q.-T., and Papadopoulou, T. (2001). *The Geometry of Multiple Images: The Laws That Govern The Formation of Images of A Scene and Some of Their Applications*. MIT Press.

Ferrie, F., Lagarde, J., and Whaite, P. (1990). Recovery of volumetric object descriptions from laser rangefinder images. In *European Conference on Computer Vision*, pages 387–396.

Ferrie, F., Lagarde, J., and Whaite, P. (1993a). Darboux frames, snakes, and superquadrics: Geometry from the bottom up. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(8):771–784.

Ferrie, F., Mathur, S., and Soucy, G. (1993b). Feature extraction for 3d model building and object recognition. In *3D Object Recognition Systems*.

Fine, I. and Jacobs, R. A. (1999). Modeling the combination of motion, stereo, and vergence angle cues to visual depth. *Neural Computation*, 11(6):1297–1330.

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

Fitzgibbon, A., Wexler, Y., and Zisserman, A. (2003). Image-based rendering using image-based priors.

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

Fua, P. (1997). From multiple stereo views to multiple 3-d surfaces. *International Journal of Computer Vision*, 24(1):19–35.

Fua, P. and Leclerc, Y. (1995). Object-centered surface reconstruction: Combining multi-image stereo and shading. *International Journal of Computer Vision*, 16(1):35–56.

Fua, P. and Leclerc, Y. (1996). Taking advantage of image-based and geometry-based constraints to recover 3-d surfaces. *Computer Vision and Image Understanding*, 64(1):111–127.

Fua, P. and Sander, P. (1992). Segmenting unstructured 3d points into surfaces. In *European Conference on Computer Vision*, pages 676–680.

Garland, M. and Heckbert, P. S. (1998). Simplifying surfaces with color and texture using quadric error metrics. In *Conference on Visualization*, pages 263–269. IEEE Computer Society Press.

Grenander, U. and Srivastava, A. (2001). Probability models for clutter in natural images. *PAMI*, 23(4):424–429.

Grimson, W. (1983). Surface consistency constraints in vision. In *Computer Vision Graphics and Image Processing*, volume 24, pages 28–51.

Grimson, W. and Huttenlocher, D. (1992). Special issue on interpretation of 3d scenes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(2):97–307.

Hartley, R. (1995). A linear method for reconstruction from lines and points. In *International Conference on Computer Vision*, pages 882–887.

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

Hartley, R., de Agapito, L., Reid, I., and Hayman, E. (1999). Camera calibration and the search for infinity. In *International Conference on Computer Vision*, pages 510–515.

Hartley, R. and Zisserman, A. (2000). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049.

Hartt, K. and Carlotto, M. (1989). A method for shape-from-shading using multiple images acquired under different viewing and lighting conditions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 53–60.

Hebert, M., Ponce, J., Boult, T., and Gross, A. (1994). Report on the 1995 workshop on 3d object representations in computer vision. In *Object Representation in Computer Vision, International NSF-ARPA Workshop*, volume 994 of *Lecture Notes in Computer Science*. Springer Verlag.

Heipke, C. (1992). Integration of digital image matching and multi image shape-from-shading. *International Society of Photogrammetry and Remote Sensing XVII*, 29 B3:186–198.

Heyden, A. and Astrom, K. (1997). Euclidean reconstruction from image sequences with varying and unknown focal length and principal point. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 438–443.

Hill, A., Cootes, T., and Taylor, C. (1996). Active shape models and the shape approximation problem. *Image and Vision Computing*, 14(8):601–607.

Horn, B. (1977). Understanding image intensities. *Artificial Intelligence*, 8:201–231.

Horn, B. (1990). Height and gradient from shading. *International Journal of Computer Vision*, 5(1):37–76.

Huang, J., Lee, A., and Mumford, D. (2000). Statistics of range images. In *CVPR00*, pages I: 324–331.

Huang, J. and Mumford, D. (1999). Statistics of natural images and models. In *CVPR99*, pages I: 541–547.

Hung, Y., Cooper, D., and Cernuschi-Frias, B. (1991). Asymptotic bayesian surface estimation using an image sequence. *International Journal of Computer Vision*, 6(2):105–132.

Jacobs, R. A. (1999). Optimal integration of texture and motion cues to depth. *Vision Research*, 39:3621–3629.

Johnston, E. B., Cumming, B. G., and Landy, M. S. (1994). Integration of motion and stereopsis cues. *Vision Research*, 34:2259–2275.

Kaiser, B., Schmolla, M., and Wrobel, B. P. (1992). Application of image pyramid for surface reconstruction with fast vision. *International Society for Photogrammetry and Remote Sensing*.

Kanade, T. and Morris, D. D. (1998). Factorization methods for structure from motion. *Philosophical Transactions of the Royal Society of London, Series A*, 356(1740):1153–1173.

Kim, S., Hagh-Shenas, H., and Interrante, V. (2004). Conveying shape with texture: Experimental investigations of texture's effects on shape categorization judgments. *IEEE Trans. Vis. Comput. Graph.*, 10(4):471–483.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680.

Klein, R. and Schilling, A. G. (1999). Efficient rendering of multiresolution meshes with guaranteed image quality. *The Visual Computer*, 15(9):443–452.

Koenderink, J. and van Doorn, A. (1979). The internal representation of solid shape with respect to vision. *Biological Cybernetics*, 32:211–216.

Krsek, P., Pajdla, T., and Hlaváč, V. (1997). Estimation of differential structures on triangulated surfaces. In *21st Workshop of the Austrian Association for Pattern Recognition*, pages 157–164.

Kumar, R. and Hanson, A. (1994). Robust methods for estimating pose and a sensitivity analysis. *Computer Vision Graphics and Image Processing*, 60(3):313–342.

Kurazume, R., Nishino, K., Zhang, Z., and Ikeuchi, K. (2001). Mapping textures on 3d geometric model using reflectance image. In *Data Fusion Workshop in IEEE Int. Conf. on Robotics and Automation*.

Kurazume, R., Nishino, K., Zhang, Z., and Ikeuchi, K. (2002). Simultaneous 2d images and 3d geometric model registration for texture mapping utilizing reflectance attribute. In *Proc. of Fifth Asian Conf. on Comput. Vision*.

Lange, K. (1990). Convergence of em image reconstruction algorithms with gibbs priorsn. *IEEE Transactions on Medical Imaging*, 9(4):439–446.

Leclerc, Y. and Bobick, A. (1991). The direct computation of height from shading. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 552–558.

Li, A. and Zaidi, Q. (2003). Observer strategies in perception of 3-d shape from isotropic textures: developable surfaces. *Vision Research*, 43:2741–2758.

Liedtke, C., Busch, H., and Koch, R. (1991). Shape adaptation for modelling of 3d objects in natural scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 704–705.

Lindstrom, P. and Turk, G. (2000). Image-driven simplification. *ACM Transactions on Graphics*, 19(3):204–241.

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

Lowe, D. (1991). Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441–450.

Luebke, D. and Erikson, C. (1997). View-dependent simplification of arbitrary polygonal environments. *Computer Graphics*, 31(Annual Conference Series):199–208.

Mahamud, S., Hebert, M., Omori, Y., and Ponce, J. (2001). Provably-convergent iterative methods for projective structure from motion. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Marr, D. (1982). *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W.H. Freeman, SanFrancisco, California.

Marr, D. and Poggio, T. (1979). A computational theory of human stereo vision. In *Proceedings of Royal Society London*, volume B-204, pages 301–328.

McLauchlan, P., Mayhew, J., and Frisby, J. (1991). Stereoscopic recovery and description of smooth textured surfaces. *Image and Vision Computing*, 9(1):20–26.

McLauchlan, P. and Murray, D. (1995). A unifying framework for structure and motion recovery from image sequences. In *International Conference on Computer Vision*, pages 314–320.

Medioni, G., Lee, M., and Tang, C. (2000). *A Computational Framework for Segmentation and Grouping*. Elsevier Science.

Miller, J., O., A., Thorpe, C., and Kanade, T. (1999). Precision 3-d modeling for autonomous helicopter flight. In *Proceedings of International Symposium of Robotics Research (ISRR)*.

Morris, D. D. and Kanade, T. (2000). Image-consistent surface triangulation. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 332–338. IEEE Computer Society Press.

Nishino, K., Sato, Y., and Ikeuchi, K. (2001). Eigen-texture method: Appearance compression and synthesis based on a 3d model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1257–1265.

Nobuhara, S. and Matsuyama, T. (2003). Dynamic 3d shape from multi-viewpoint images using deformable mesh models. In *Proc. of 3rd International Symposium on Image and Signal Processing and Analysis*, pages 192–197.

Okutomi, M. and Kanade, T. (1993). A multiple-baseline stereo. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(4):353–363.

Pentland, A. (1990). Automatic extraction of deformable part models. *International Journal of Computer Vision*, 4(2):107–126.

Pentland, A., Moghaddam, B., and Starner, T. (1994). View-based and modular eigenspaces for face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 84–91.

Pentland, A. and Sclaroff, S. (1991). Closed-form solutions for physically based shape modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):715–729.

Petitjean, S. (2002). A survey of methods for recovering quadrics in triangle meshes. *ACM Computing Surveys (CSUR)*, 34(2):211–262.

Pollard, S., Mayhew, J., and Frisby, J. (1985). Pmf: A stereo correspondence algorithm using a disparity gradient limit. *Perception*, 14:449–470.

Pollefeys, M., Koch, R., and Van Gool, L. (1999). Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters. *International Journal of Computer Vision*, 32(1):7–25.

Reed, M. K. and Allen, P. K. (1999). 3-d modeling from range imagery: An incremental method with a planning component. *Image and Vision Computing*, 17(2):99–111.

Ruben, H. and Reed, W. J. (1973). A more general form of a theorem of crofton. *Journal of Applied Probability*, 10:479–482.

Sander, P. and Zucker, S. (1990). Inferring surface trace and differential structure from 3-d images. *PAMI*, 12(9):833–854.

Scharstein, D., Szeliski, R., and Zabih, R. (2001). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Proceedings of the IEEE Workshop on Stereo and Multi-Baseline Vision*, Kauai.

Schilling, A. G. and Klein, R. (1998). Texture dependent refinement of multiresolution meshes. Technical Report WSI–98–2, Wilhelm-Schickard-Institut für Informatik, University of Tübingen, Germany.

Seber, G. A. F. and Wild, C. J. (1989). *Nonlinear Regression*. John Wiley & Sons.

Shashua, A. and Werman, M. (1995). Trilinearity of three perspective views and its associated tensor. In *International Conference on Computer Vision*, pages 920–925.

Slama, C. (1980). *Manual of Photogrammetry*. American Society of Photogrammetry and Remote Sensing, Falls Church, Virginia, USA.

Stevenson, R. L., Schmitz, B. E., and Delp, E. J. (1994). Discontinuity preserving regularization of inverse visual problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(3):455–469.

Stockely, E. and Wu, S. (1992). Surface parameterization and curvature measurement of arbitrary 3-d objects: Five practical methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(8):833–840.

Sturm, P. (1997). Critical motion sequences for monocular self-calibration and uncalibrated euclidean reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1100–1105.

Sturm, P. and Triggs, B. (1996). A factorization based algorithm for multi-image projective structure and motion. In *Proceedings of the 4th European Conference on Computer Vision, Cambridge, England*, volume 1065 of *Lecture Notes in Computer Science*, pages 709–720.

Szeliski, R. (1991). Shape from rotation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 625–630.

Szeliski, R. and Tonnesen, D. (1992). Surface modeling with oriented particle systems. *Computer Graphics*, 26(2):185–194.

Szeliski, R., Tonnesen, D., and Terzopoulos, D. (1993). Modeling surfaces of arbitrary topology with dynamic particles. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 82–87.

Terzopoulos, D. (1986). Regularization of inverse visual problems involving discontinuities. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(4):413–424.

Terzopoulos, D. (1988). The computation of visible-surface representations. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10(4):417–438.

Terzopoulos, D. and Metaxas, D. (1991). Dynamic 3d models with local and global deformations: Deformable superquadrics. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(7):703–714.

Terzopoulos, D., Witkin, A., and Kass, M. (1987). Symmetry-seeking models and 3d object reconstruction. *International Journal of Computer Vision*, 1(3):211–221.

Thirion, J. (1996). The extremal mesh and the understanding of 3d surfaces. *IJCV*, 19(2):115–128.

Tomasi, C. and Kanade, T. (1991). Shape and motion from image streams: a factorization method - part 3 detection and tracking of point features. Technical Report CMU-CS-91-132, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA.

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

Tomasi, C. and Shi, J. (1994). Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600.

Torr, P., Fitzgibbon, A., and Zisserman, A. (1998). Maintaining multiple motion model hypotheses over many views to recover matching and structure. In *International Conference on Computer Vision*, pages 485–491.

Torr, P. and Murray, D. (1993). Outlier detection and motion segmentation. *SPIE - The International Society for Optical Engineering*, 2059:432–443.

Triggs, B. (1995). Matching constraints and the joint image. In *International Conference on Computer Vision*, pages 338–343.

Triggs, B., McLauchlan, P., Hartley, R., and Fitzgibbon, A. (2000). Bundle adjustment – A modern synthesis. In Triggs, W., Zisserman, A., and Szeliski, R., editors, *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag.

van Ginneken, B., Koenderink, J., and Dana, K. (1999). Texture histograms as a function of irradiation and viewing direction. *IJCV*, 31(2/3):169–184.

Vasilescu, M. and Terzopoulos, D. (1992). Adaptive meshes and shells: Irregular triangulation, discontinuities, and hierarchical subdivisions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 829–832.

Vemuri, B. and Malladi, R. (1991). Deformable models: Canonical parameters for surface representation and multiple view integration. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 724–725.

Wang, Y. and Wang, J. (1990). Surface reconstruction using deformable models with interior and boundary constraints. In *International Conference on Computer Vision*, pages 300–303.

Wexler, Y., Fitzgibbon, A., and Zisserman, A. (2003). Learning epipolar geometry from image sequences. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Whaite, P. and Ferrie, F. (1991). From uncertainty to visual exploration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):1038–1049.

Williams, N., Luebke, D., Cohen, J. D., Kelley, M., and Schubert, B. (2003). Perceptually guided simplification of lit, textured meshes. In *Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 113–121. ACM Press.

Witkin, A., Terzopoulos, D., and Kass, M. (1987). Signal matching through scale space. *International Journal of Computer Vision*, 1(2):133–144.

Xia, J. C. and Varshney, A. (1996). Dynamic view-dependent simplification for polygonal models. In *Proceedings of the 7th conference on Visualization '96*, pages 327–334. IEEE Computer Society Press.

Zaidi, Q. and Li, A. (2002). Limitations on shape information provided by texture cues. *Vision Research*, 42(7):815–835.

Zhao, H. and Shibasaki, R. (2000). Reconstruction of textured urban 3d model by ground-based laser range and ccd images. In *IEICE Transaction on Information and Systems*, number 7 in E83-D, pages 1429–1440.

Zhao, H. and Shibasaki, R. (2001). A robust method for registering ground-based laser range images of urban outdoor object. *Photogrammetric Engineering and Remote Sensing*, 67(10):1143–1153.