Toward an Alternative Approach to Multi-Camera Scene Reconstruction

Jianfeng Yin

Doctor of Philosophy

Department of Electrical and Computer Engineering

McGill University

Montreal,Quebec

October 2008

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Doctor of Philosophy

DEDICATION

This thesis is dedicated to my wife and my parents.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Professor Jeremy R. Cooperstock, who provided me the opportunity to work on this interesting project and gave me valuable guidance and help to complete the Ph.D. I would also like to thank Professor James Clark, Professor Frank Ferrie, and Professor Michael Langer for their advice on my thesis research. I would like to thank Stephane Pelletier for translating the abstract into French for me. I would like to acknowledge Stephen Spackman, Wei Sun, Shawn Arseneau, Zhi Qi, Stephane Pelletier, Yuwen Li for offering various help to my study.

ABSTRACT

This dissertation addresses several issues related to 3D object reconstruction in a video-projected immersive environment, based on the views obtained from multiple cameras. One such issue is color correction to account for the differences between cameras and projectors. Various methods are investigated, and a neural network approach is proposed as an effective solution. The problem of textureless or occluded regions on the construction of depth maps is also considered. As an improvement, depth information is propagated by nonlinear diffusion processing based on image gradient constraints.

Unlike traditional methods such as space carving and shape from silhouettes, this dissertation treats 3D reconstruction as a classification problem. The challenge is to find a suitable feature to distinguish surface points from non-surface ones. Two such features are proposed, one based on the color histogram of the projections of each voxel onto every camera, and the other, the Frobenius norm of the camera agreement matrix. Tensor voting is used to refine the reconstruction and the results are evaluated experimentally on synthetic and physical data.

ABRÉGÉ

Cette dissertation traite de différents aspects reliés à la reconstruction d'objets en 3D à partir d'images provenant de plusieurs caméras dans un environnement immersif de projection vidéo. Un des aspects est la correction des couleurs servant à compenser les différences entre caméras et projecteurs. Plusieurs méthodes sont analysées et une approche basée sur les réseaux neuronaux est proposée comme solution. Le problème des régions cachées ou uniformes sur la construction des cartes de profondeur est aussi considéré. Comme amélioration, l'information de profondeur est propagée à l'aide de traitement non linéaire de diffusion basé sur des contraintes de gradient d'image.

Contrairement aux méthodes traditionnelles telles le *space carving* et le *shape-from-silhouettes*, cette dissertation considère la reconstruction 3D comme un problème de classification. Le défi consiste à trouver un attribut approprié afin de distinguer les points de surface de ceux qui n'en sont pas. Deux attributs sont proposés, l'un basé sur l'histogramme de couleurs des projections de chaque voxel sur toutes les caméras, l'autre sur la norme de Frobenius de la matrice d'entente des caméras. Le vote de tenseurs est employé pour raffiner la reconstruction et les résultats sont évalués expérimentalement sur des données réelles et synthétiques.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
Introduction

1.1 The Problem

The problem considered here is to segment an object from its background, and build a 3D model of it, as Figure 1–1 suggests. Reconstructing a 3D model from a set of images is a major topic in computer vision. It is important for applications including virtual reality, tele-presence, and visual navigation. It is challenging because of complex geometry, a large number of degrees of freedom, and the lack of obvious representations. One proposed application for this problem is Shared Reality, where people at different sites are extracted from their local environments, modelled, and then inserted into a shared virtual environment, as if they were working in the same place. A functioning prototype of such a shared reality environment is shown in Figure 1–2.

Many algorithms have been proposed to solve the reconstruction problem, including stereo vision, shape from silhouettes, and volumetric reconstruction (for example, voxel coloring and space carving). Stereo vision usually involves two steps as shown in Figure 1–3: stereo matching and triangulation. Stereo matching [137, 26, 43, 25, 10] tries to find features (e.g., pixels, edges, or corners) in different images that belong to the same surface features. This is the most difficult task. Once the matching between images is established, 3D features can be reconstructed using

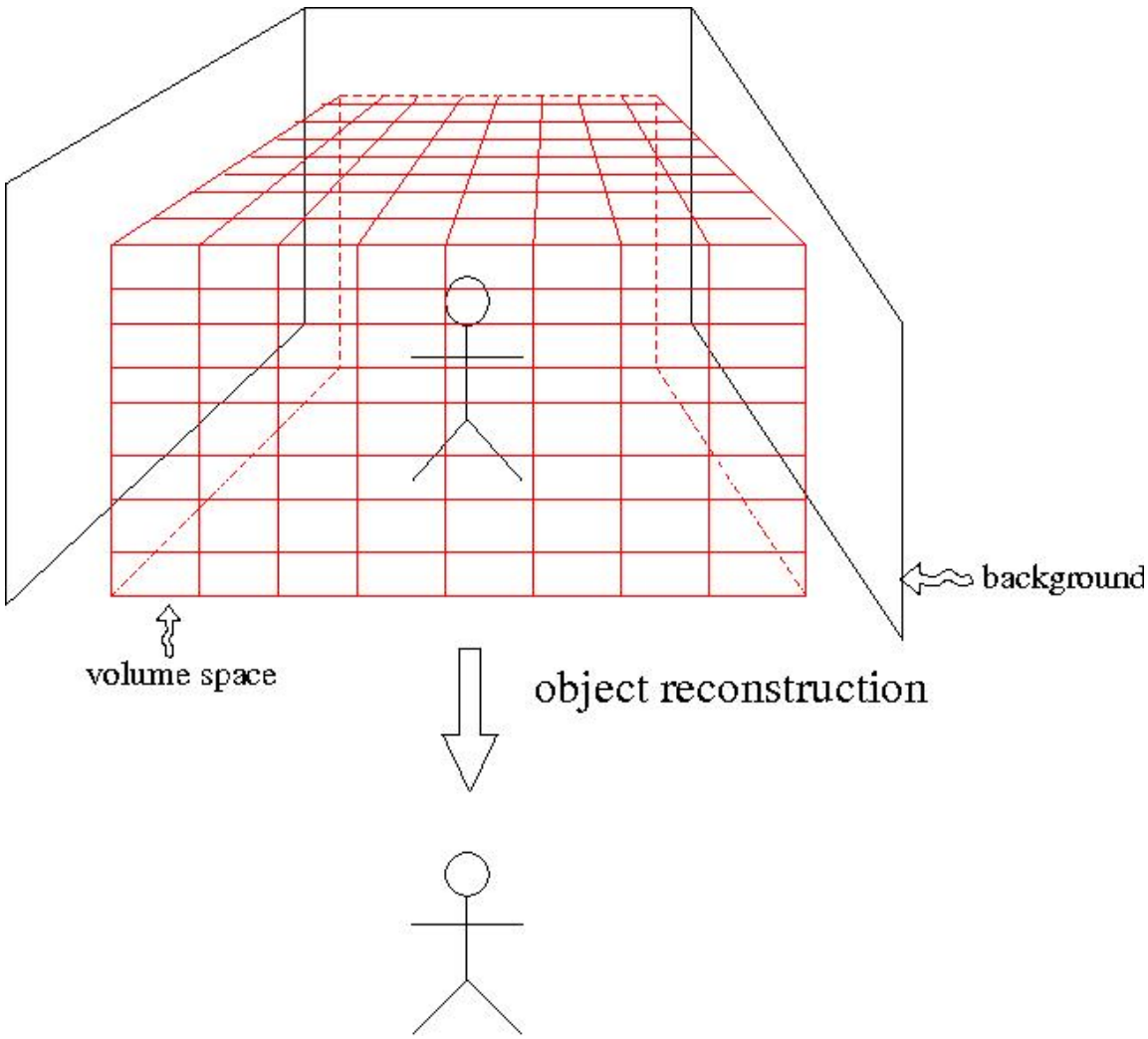


Figure 1–1: Object modeling with background removal. This is done by limiting the volume so that it excludes the background (consisting of the three planar screens).

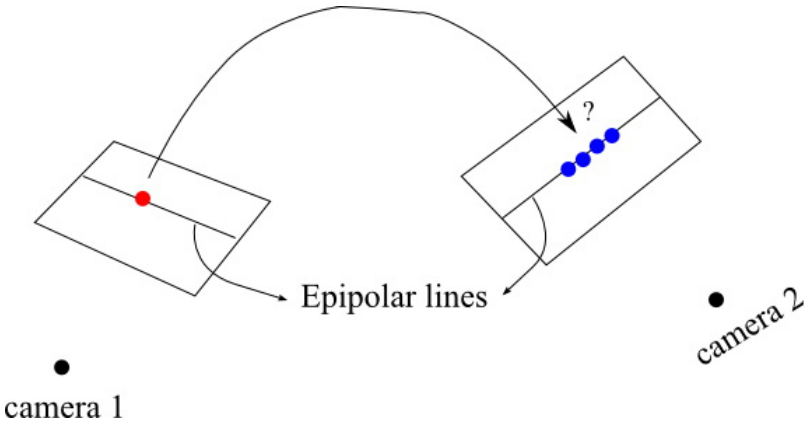
Figure 1–2: A functioning prototype of the Shared Reality Environment(SRE).

triangulation techniques [66]. Stereo methods generate more accurate correspondences in the case of a small line segment between the two camera centers (known as the *baseline*), which imposes a serious restriction on camera setup and requires a large number of cameras in order to recover the scene effectively. For complex scenes with a large range of disparities and occlusions, stereo algorithms have difficulty obtaining good results. Shape from silhouettes [92, 104] works by recasting silhouettes from images to space, and finds the intersection of all visual cones, as Figure 1–4 indicates. It must first extract the silhouettes, which is a complex task in itself for many applications. Volumetric reconstruction starts by discretizing space into voxel arrays (Figure 1–5), and tries to determine if they are occupied by testing whether they are photo-consistent among cameras that can see them. Voxel coloring and space carving (e.g., [141, 90, 89, 40, 13, 23, 24, 164, 1]) are typical algorithms. Two important problems of volumetric reconstruction are the computation of visibility

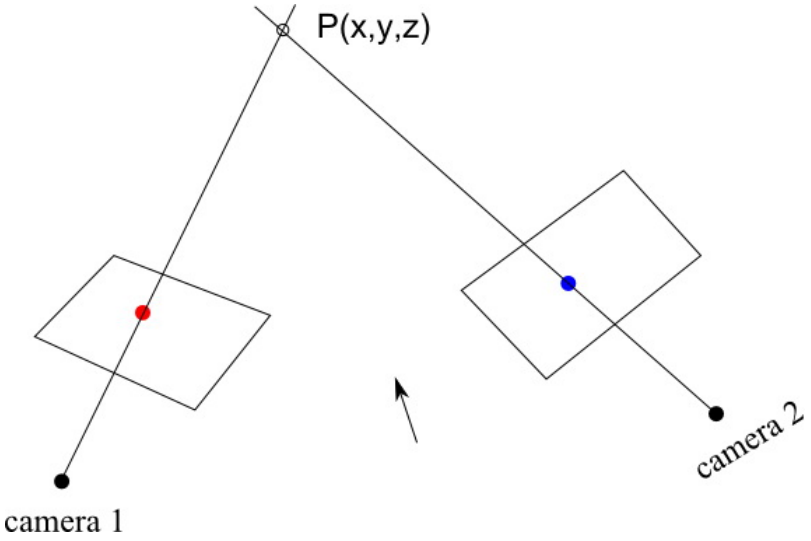
and photo consistency. The former concerns the problem of which cameras can see a voxel, while the latter is used to determine whether a given voxel is occupied. In general, the two tasks are coupled, as a voxel is considered to be photo-consistent when its color appears to be similar to all the cameras that can see it. The photo consistency test is applied to the corresponding pixels from cameras that can see the voxel, which in turn, depends on the visibility result. Similarly, in order to solve visibility for a particular voxel, we need to know whether any intervening voxels between it and the camera are occupied, as determined by the photo consistency tests. Existing techniques either restrict the camera configuration to solve the visibility problem a priori [141], or use multi-pass algorithms to solve the coupled problem [90, 89].

For the background removal problem, we can exploit the known environment geometry (for example, planar screens with known coordinates in our environment). Supposing the cameras are calibrated, we can relate pixels that are projections of the same point on the background in different cameras. Under the Lambertian assumption, these pixels should have the same color. Based on this observation, we infer that if pixels have the same color that are projections of a point on the background, they can be accepted as background pixels and thereby removed, as illustrated by the examples in Figure 1–6.

The geometry-based approach is simple to implement and fast. In reality, it is hard to measure the background geometry accurately and calibrate cameras precisely, so pixel correspondences computed from the background geometry and camera parameters are usually inaccurate. Consequently, a neighbourhood spanning a few



(a)



(b)

Figure 1–3: Stereo Vision.(a) stereo matching tries to find a pixel along the epipolar line in the right image corresponding to the pixel in the left image; (b) Once a pair of matching pixels is found, the corresponding 3D point $P(x, y, z)$ can be calculated by triangulation.

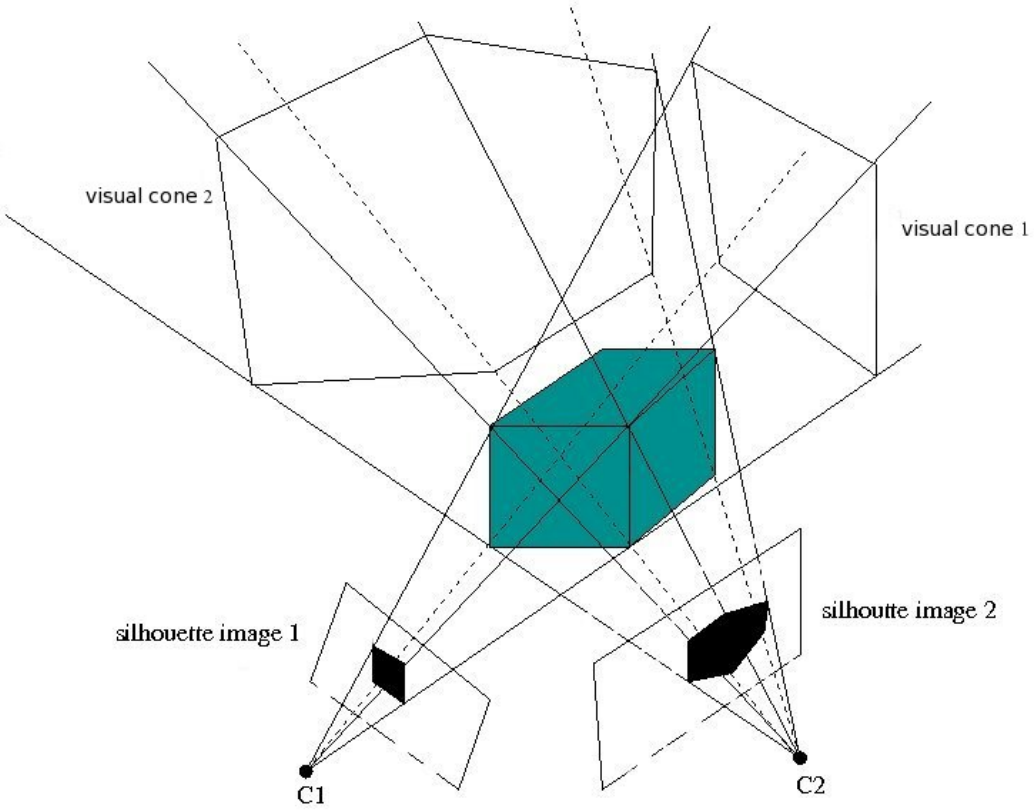


Figure 1–4: Shape from Silhouettes. The shape of an object can be estimated by intersecting the visual cones.

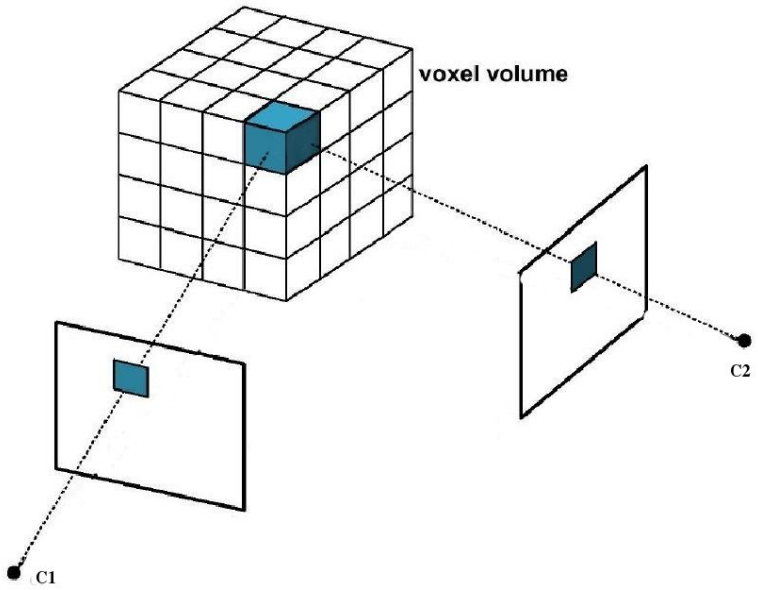


Figure 1–5: Volumetric Reconstruction.

pixels is often employed. Furthermore, it is unlikely for pixels to have exactly the same color, but they can be considered the same if their color variation is under a certain threshold. Due to illumination changes and non-Lambertian parts in the scene, it is difficult to choose a proper threshold and segment the background correctly. If the threshold is set too low, an excessive amount of background may be retained. If it is set too high, some foreground objects may be eliminated. Occlusions can also be problematic. When a background point can only be seen by some cameras and occluded in others, it is impossible to be removed by naive geometry-based methods, as Figure 1–6 suggests. Despite these disadvantages, it may still be helpful to take the geometry-based approach as a front end to other methods when using a conservative threshold to preserve foreground objects.

original images results with background removed

Figure 1–6: Background removal based on the known geometry. The planar screen (background) induces a homography transform between two cameras. If two pixels in two cameras related by the homography have the same color, they are regarded as projections of the same point (a background point) on the screen, and are removed. In the results, background pixels are painted as black.

The alternative to background removal relies on the fact that objects and background have different depths. If depth maps of the scene can be built by some stereo matching methods, it is possible to perform segmentation based on depth differences. A stereo matching method intended for this purpose is proposed in Chapter 4.

Both the aforementioned approaches work in image space. If we start from object space instead, since we know the environment geometry, we can limit the working volume to include foreground objects while excluding the background. In this manner background removal is naturally combined with object reconstruction, as illustrated in Figure 1–1.

In this thesis, we focus on volumetric reconstruction. Some of the questions we consider, related to this problem, include:

1. How much can we say about a voxel based only on reference images and camera parameters, i.e., the only information at hand? More formally, for a voxel V, suppose its projection to a camera c is $P_V^c = \{p_c^j | j = 1, 2, ..., m_c\}$ (where, p_c^j is a pixel belonging to the projection of V in camera c), and its occupancy is o_V. Given only the projection pixel set to all reference cameras $P_V = \{P_V^c, c = 1, ..., N\}$, can we determine o_V, and if so, how?
2. How should neighbouring voxels interact with each other? Intuitively, if some neighbouring points of a voxel are already on a surface, it is very likely the current voxel is on the same surface. How can this knowledge be incorporated into an algorithm?

In answering these questions, it is worth noting that voxels can be classified into categories, each with its particular color distribution. Based on this observation,

some judgements about voxel occupancy can be made. Furthermore, we can view voxel occupancy as a classification problem, for which the challenge is to find a suitable feature to distinguish different categories. Based on the premise that voxel occupancy is determined largely by agreement between reference cameras, we propose to build a camera agreement matrix for each voxel, from which various features can be extracted for classification. We also apply a tensor voting technique to exploit neighbouring voxel information. Figure 1–7 shows a result of our proposed method. Note that this thesis is concerned primarily with determining volumetric properties through voxel occupancy; the problem of model coloring is not addressed here.

Before we can carry out a space carving or similar voxel classification process, we must first solve the problem of color consistency between the different cameras. As illustrated in Figure 1–8, the source images from individual cameras can vary significantly in color. The correction of such color differences is another topic considered in Chapter 3.

1.2 Contributions

The contributions of this thesis are as follows:

- The color correction problem is analyzed in a digital projection environment, various methods are investigated, and a neural network approach is proposed as an effective solution.
- A nonlinear diffusion algorithm incorporating image gradient constraints is developed to improve depth maps in textureless and occluded regions.

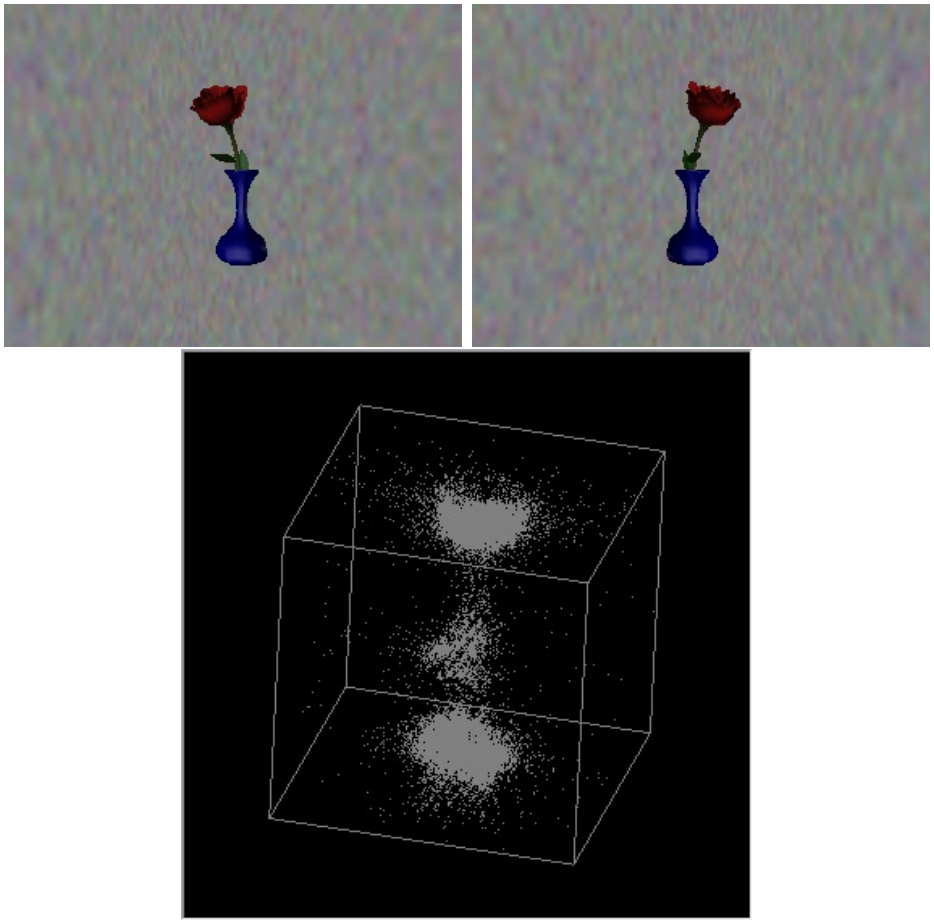
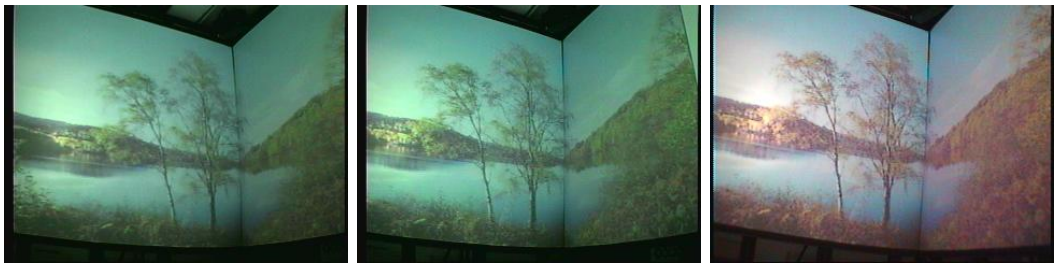


Figure 1–7: Object modeling with background removal. Note that we are focusing on the voxel occupancy, not its coloring.



a b c

Figure 1–8: Images captured of the same scene by three different cameras.

- Voxel characteristics of different categories are analyzed and a new approach to the volumetric reconstruction problem is proposed, treating this as a classification problem.
- Various features based on color histograms of voxels and their camera agreement matrices are proposed to solve the voxel classification problem.
- Voxel neighbourhood information is exploited to refine reconstruction results by tensor voting, which makes it possible to correct some errors from the classification step. This differs from space carving, where errors may be propagated and cannot subsequently be reversed.
- Our reconstruction method uses local information only, permitting a parallel implementation which offers a computational improvement.

1.3 Outline

The thesis is structured as follows:

Chapter 2 reviews 3D reconstruction-related works, including stereo-based methods, space carving, level set methods, shape from silhouettes, and active vision methods.

Chapter 3 discusses color correction methods in digital projection environments.

Chapter 4 investigates the use of image gradient constraints to improve depth maps in textureless and occluded regions with a nonlinear diffusion technique.

Chapter 5 analyzes characteristics of voxel categories and reconstruction of object models by using the highest frequency of the color histogram of voxel projections on all images.

Chapter 6 views voxel occupancy problem as a classification problem and extracts features from camera agreement matrices. It also exploits voxel neighbourhood information by tensor voting.

Chapter 7 discusses possible future work and summarizes the results obtained in the thesis.

CHAPTER 2
Literature Review

Many algorithms have been developed to solve the 3D reconstruction problem. These typically fall into two categories: image space and object space approaches. The former begin by finding 2D relationships between images (i.e., pixel correspondences), then infer 3D geometry from these. The later start from the 3D space where objects are defined, projecting 3D samples (for example, points, voxels, or surfels) onto images, then measuring similarities between those projections to induce object structure. These methods are discussed in further detail in the following sections.

2.1 Image Space Approaches

The first step, finding pixel correspondences, constitutes the primary task of stereo matching. The second step, building 3D geometry, can be accomplished by triangulation. Normally one stereo pair generates a single depth map, which can only provide a partial model of a scene. In order to obtain a full model, techniques to merge depth maps from multiple stereo pairs are needed.

2.1.1 Dense Stereo Matching

Stereo matching tries to find pixel correspondences in different images. Some stereo matching techniques can generate sparse pixel correspondences by using features such as edges and corners ([4, 5, 61, 144, 64]). It is hard to generate a full 3D model based on such sparse features. Here we will concentrate only on methods that can produce a dense matching between two images. Because of epipolar constraints,

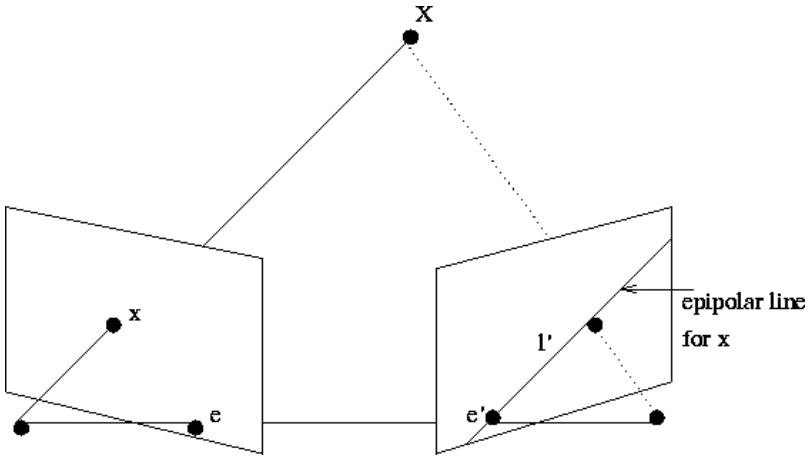


Figure 2–1: The epipolar geometry, the corresponding pixel of x lies on the epipolar line on the right image.

we can search the corresponding pixel from the left image along a 1D epipolar line in the right image (Figure 2–1). We assume that the images are rectified, so that there are only horizontal disparities, i.e., the epipolar line is along the x-axis, and two corresponding pixels have the same y values.[1]

The goal of stereo matching is to construct a disparity map $d = d(x, y)$ such that pixel (x, y) in the left image and pixel $(x + d, y)$ in the right image are projections of the same 3D point. In other words, for each pixel (x, y), the correct disparity value needs to be found among all possible matches. In order to do so, a measurement needs to be defined for how well two pixels match given a disparity value. Many options are proposed in the literature ([18, 134, 114, 169, 14, 34, 131, 100, 101, 82, 74, 77]), of

[1] For information about stereo rectification, readers can consult references such as [56, 66, 171, 97].

which the most commonly used are the *normalized cross correlation* (NCC, Equation 2.1) and the *sum of squared differences* (SSD, Equation 2.2). Occasionally, the *sum of absolute differences* (SAD, Equation 2.3) is used in place of SSD for computational efficiency. Such computations are usually applied to a neighbourhood, for example, a 5x5 window centered at the pixel under consideration. Given two reference images, I_1 and I_2, and letting $N(x, y)$ represent the neighborhood of the computation, the various matching metrics can be defined as in Equations 2.1 - 2.3.

$$NCC(x, y, d) = \frac{\sum_{u,v \in N(x,y)} (I_1(x+u, y+v) - \bar{I}_1)(I_2(x+u+d, y+v) - \bar{I}_2)}{\sqrt{\sum_{u,v \in N(x,y)} (I_1(x+u, y+v) - \bar{I}_1)^2 (I_2(u+x+d, v+y) - \bar{I}_2)^2}} \tag{2.1}$$

$$SSD(x, y, d) = \sum_{u,v \in N(x,y)} (I_1(x+u, y+v) - I_2(x+u+d, y+v))^2 \tag{2.2}$$

$$SAD(x, y, d) = \sum_{u,v \in N(x,y)} |I_1(x+u, y+v) - I_2(x+u+d, y+v)| \tag{2.3}$$

For any of these measures, the disparity of a pixel is chosen as the value with the smallest cost. However, this simple strategy often yields poor results, motivating the incorporation of further constraints as an improvement. Two popular constraints are uniqueness and smoothness ([100]), which state that each pixel should have at most one disparity value and that disparity varies smoothly except at a few places such as object boundaries. Based on these constraints, support and inhibition areas (as shown in Figure 2–2) can be defined for every pixel. Other examples include the disparity gradient constraint [27, 120] and the ordering constraint [115, 33, 70, 143]. The initial local cost measurements can be refined by various algorithms applying

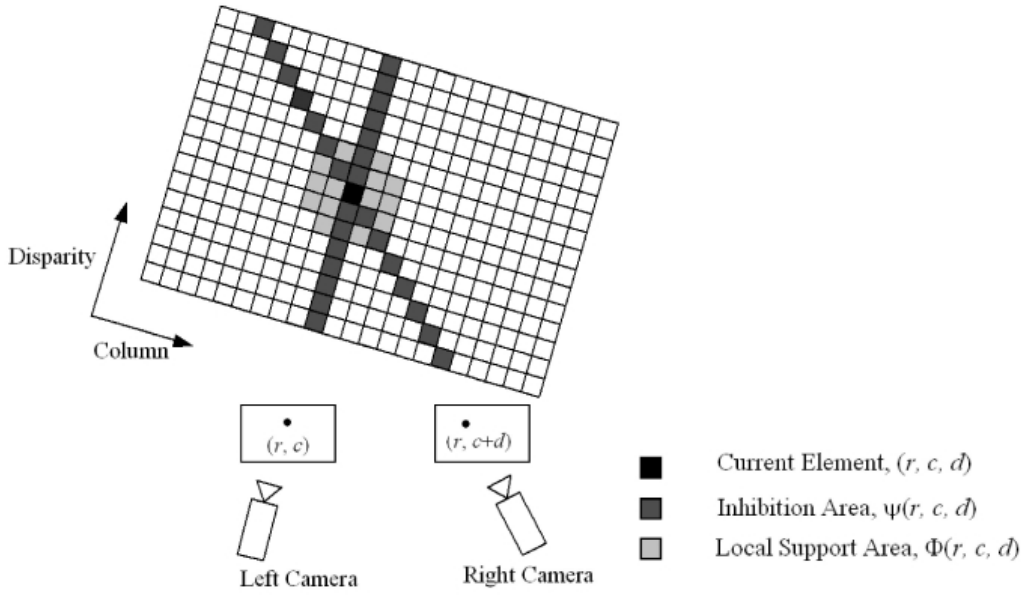


Figure 2–2: 2D slice of the 3D disparity space. The current element is shown in black. The inhibitory regions are designed to constrain the element's disparity to a single value. The support region enhances smoothness, so that neighboring elements have similar disparity values. Reprinted from Zitnick and Kanade [172] with permission, ©2000 IEEE.

these constraints, for example, cooperative algorithms [100, 172] and diffusion algorithms [136, 143]. After iterative updating, a sub-optimal disparity function can be found.

Global methods consider disparity computation as a whole. This problem is often formulated as an energy minimization problem, as Equation 2.4 suggests.

$$E(d) = E_{data}(d) + E_{smooth}(d) \tag{2.4}$$

For one disparity configuration, $E_{data}(d)$ measures how well the disparity function $d = d(x, y)$ agrees with the image pair. i.e.,

$$E_{data}(d) = \sum_{x,y} C(x, y, d(x, y)) \tag{2.5}$$

where, $C(x, y, d(x, y))$ is a cost measurement between pixel (x, y) in the left image, and $(x + d, y)$ in the right image. It can be just the intensity difference of the two pixels, or some other cost measurements such as the aforementioned SSD and NCC.

$E_{smooth}(d)$ is a regularization term. It constrains the disparity function to be mostly smooth except at a few discontinuities. An example is shown in Equation 2.6:

$$E_{smooth}(d) = \sum_{x,y} \rho(d(x, y) - d(x + 1, y)) + \rho(d(x, y) - d(x, y + 1)) \tag{2.6}$$

where, ρ is some monotonically increasing cost function of the difference between two neighbouring disparity values.

An occlusion term $E_{occlusion}(d)$ is sometimes added to the energy function (Equation 2.4) to address occlusions.

Global optimization methods seek a disparity function that minimizes total energy. Various methods, including gradient descent [123, 152] , simulated annealing ([84, 60, 102, 9]), mean-field annealing ([58]), maximum flow and graph cut ([130, 71, 87, 21]) can be used to minimize the energy function defined in Equation 2.4.

More detailed description and comparison of stereo matching methods can be found in survey articles ([137, 26, 43, 25, 10]), and their references.

2.1.2 Merging Depth Maps

Stereo matching methods typically generate one or several depth or disparity maps. If camera parameters are known, 3D geometry can be reconstructed. However, one depth map only provides a partial model of the scene. To obtain a full model, we need to find ways to merge these partial models [62]. Many techniques based on marching cubes [98, 127, 38, 68, 16] can be applied to such a task. The basic idea is to discretize the space into a series of cubes, and subsequently replace the cubes by a series of polygons that approximate the surface of any objects they intersect. Different cube classification combinations are shown in Figure 2–3. Marching cube algorithms must calculate the signed distance from 3D points to the estimated surface. One such method used by Rander [126] is described below.

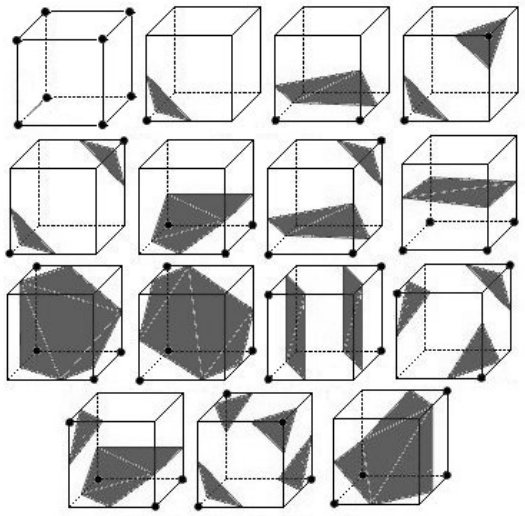


Figure 2–3: The 15 cube combinations for marching cubes.

Computation of signed distance The signed distance function $f_i(\bar{v})$ for camera i at a voxel $\bar{v} = (X, Y, Z)$ is defined in Equation 2.7. $D_{\bar{v}}$ is the depth of the voxel from the viewpoint of camera i while $D_i(\bar{q})$ is the value from the depth map of camera i at image coordinates $\bar{q}$, corresponding to the projection of v.

$$f_i(\bar{v}) = D_{\bar{v}} - D_i(\bar{q}) \tag{2.7}$$

If the 3x4 parameter matrix of camera i is P_i, the projective image coordinates q of voxel $\bar{v}$ in camera i are:

$$q = \begin{bmatrix} xw \\ yw \\ w \end{bmatrix} = [P_i] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.8}$$

From the projective coordinate q, we know $D_v = w$, and $\bar{q} = (x, y)$ the image coordinate of the projection of $\bar{v}$ in camera i. The depth $D_i(\bar{q})$ is computed by interpolating depths of $\bar{q}$'s three neighbours $\bar{q}_0, \bar{q}_1, \bar{q}_2$ (see Figure 2–4).

2.2 Object Space Approaches

Other alternatives start from object space directly, as exemplified by voxel coloring or space carving [141, 90, 89, 145, 37, 147, 40, 13, 24, 1, 124], level set methods [48, 121, 148, 75, 44, 76], and shape from silhouettes [92, 104, 122, 103, 151]. Good reviews of various methods can be found in [46, 140]. Before describing these in detail, we first discuss their common problems, namely, representation, testing for photo consistency, and visibility.

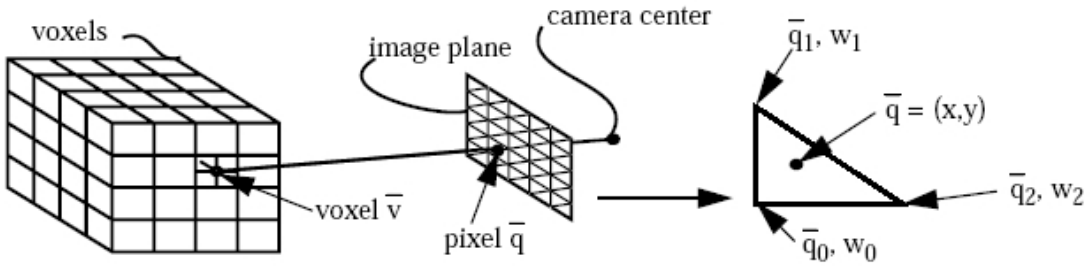


Figure 2–4: Basic operation in computing signed distance. Each voxel is projected into the image plane of each camera using the camera models already computed for stereo. The depth of the voxel can be extracted from this process. The range image is interpolated to compute the depth from the camera to the surface. The signed distance from the voxel to the surface is computed by subtracting these depths. Reprinted from Rander [126] with permission.

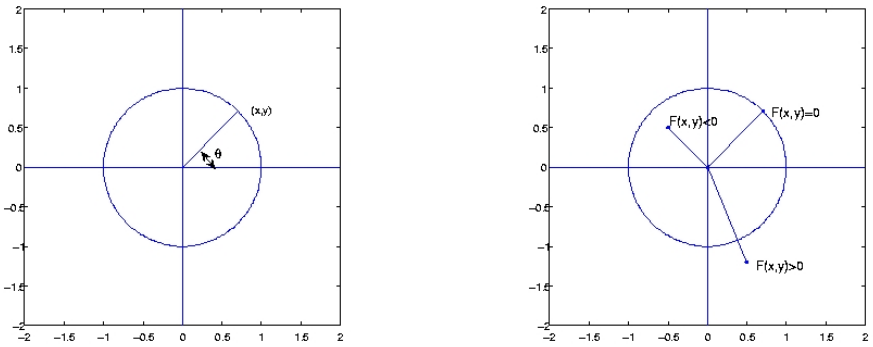
2.2.1 Representation

The geometry of an object can be formulated explicitly or implicitly. An explicit representation defines object points by a parametrization. For example, the unit circle in the plane can be parameterized as $(x, y) = (cos\theta, sin\theta), \quad \theta \in [0, 2\pi]$. In an implicit representation, an object is determined indirectly through a classification function that defines the relationship between the object and its corresponding points. The implicit equation of the unit circle in a plane is $F(x, y) = 0$, where

$$F(x, y) = x^2 + y^2 - 1, \qquad x, y \in \mathbf{R} \tag{2.9}$$

A graphical view of these two representations is shown in Figure 2–5.

Both explicit and implicit forms are used in various reconstruction algorithms. Common explicit representations include polygon meshes and voxels (as seen in Figures 2–6 and 2–7, respectively), while level sets (e.g. as seen in Figure 2–8) are



a b

Figure 2–5: The unit circle, a) an explicit representation, b) an implicit representation.

the main implicit type seen in reconstruction algorithms. Other surface representations, for example, surfels which represent objects with a set of small surface patches [119, 55, 28], can also be employed. The following discussion, however, concentrates on polygonal, voxel, and level set representations.

A polygonal representation, widely used in computer graphics, uses a set of polygons (usually triangles) to approximate an arbitrary shape. When a scene contains large planar surfaces, a polygonal representation is efficient to store and render. A polygonal model can be built automatically by matching image features (e.g., Beardsley, Torr and Zisserman [11]), or with user assistance (Devbevec et al. [41], and Cipolla and Robertson [32]), or by combining the results of many stereo pairs (e.g., Koch, Pollefeys and van Gool [85, 86]). There are typically many approaches that can be applied to partition a scene into polygons; it is often not obvious how to choose the best among these.

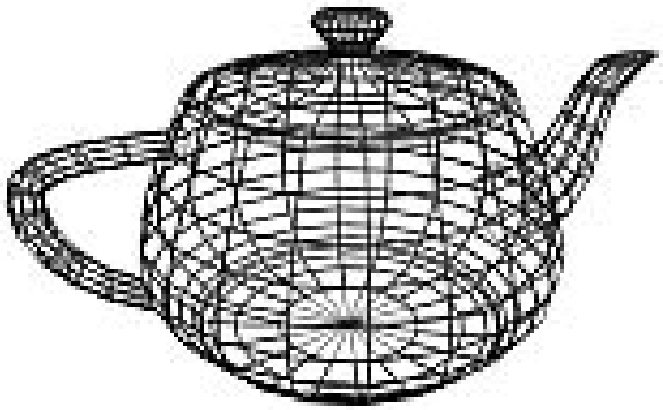


Figure 2–6: The polygonal representation of a teapot.

Voxel representations discretize space into a regular 3D grid and require a method to determine which voxels are occupied. This representation is simple, uniform, and powerful to approximate arbitrary shapes.

In the level set representation, a regular grid space is used, similar to voxel representations. Instead of finding an occupancy function, here each point in space is assigned a value, which is the distance to the curve (2D case) or surface (3D case). The location of the curve or surface is obtained by finding the zero contour of the distance function.

2.2.2 Photo Consistency Test

An obvious way to evaluate a reconstruction is to reproject it into input images, and observe the similarity to the originals. If the differences are negligible, the reconstruction is considered photo-consistent. In the following, we use π^V to denote the set of pixels from which a voxel V is visible. Kutulakos and Seitz [90] suggest

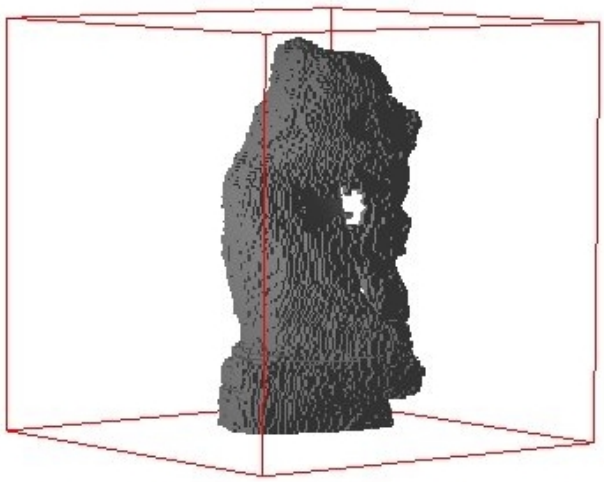


Figure 2–7: The voxel representation of a gargoyle.

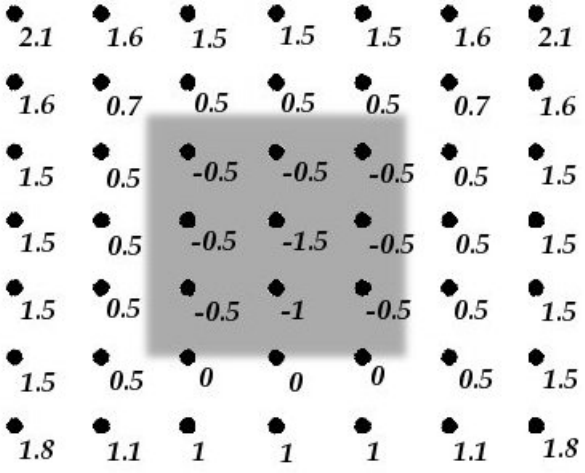


Figure 2–8: A 2-D level set representation.

photo consistency to be *monotonic*. That is, if a test is photo-consistent on π_V, it should be also consistent on all subsets of π_V. While this is a desirable property, experiments show that non-monotonic tests can sometimes produce better results [145]. In the following text, we describe several of the more popular photo-consistency tests.

Monotonic Consistency Tests

Seitz and Dyer [141] determine the photo consistency of a voxel, V, by the likelihood ratio test (LRT):

$$(n-1)s^2_{\pi^V} < \tau \tag{2.10}$$

where s_{π^V} is the standard deviation of the intensities of the pixels in π^V, n is the number of intensities in π^V, and τ is a threshold that is set experimentally. If colors are used instead of intensities, LRT can be applied to each channel respectively. LRT can produce a reasonable reconstruction, but it is sensitive to the number of pixels in π^V. The likelihood of a voxel being considered as empty, i.e., *carved*, increases with the number of pixels from which it is visible.

Other monotonic tests use the length of the great diagonal of the bounding box of colors in π_V, or the greatest distance between colors in π_V, as in the following inequality:

$$\max\{\text{dist}(\text{color}(p_1), \text{color}(p_2)) | p_1, p_2 \in \pi^V\} < \tau \tag{2.11}$$

where *dist* is a L_1 or L_2 norm in color space [145]. The advantage of these color-distance-based tests is that they are not sensitive to the number of pixels, although they are more sensitive to pixel noise than LRT.

Non-monotonic Consistency Tests

A simple non-monotonic consistency test is to threshold the square of standard deviation of the intensities of pixels in π_V:

$$s^2_{\pi^V} < \tau \tag{2.12}$$

Models based on such non-monotonic tests depend on the order in which voxels are processed, so voxels may be carved, even though they would be consistent in the final model. However, Slabaugh *et al.* [145] show that non-monotonic tests can reconstruct models with a good resemblance to the scene. Many photo consistency tests use a single threshold, which is hard to choose. In scene areas containing significant texture, a higher threshold is required, whereas a lower threshold is necessary to limit bulging effects [141, 90] in areas with less color variance. This suggests the use of an *adaptive* consistency test called the *adaptive standard deviation test* (ASDT) [145] that adjusts the threshold in proportion to the color variance in each image. Let π_i^V be the set of pixels in image i from which voxel V is visible, and $\bar{s}$ be the average of $s_{\pi_i^V}$ for all images i from which V is visible. The ASDT is defined as follows:

$$s_{\pi^V} < \tau_1 + \tau_2 \bar{s} \tag{2.13}$$

where τ_1 and τ_2 are thresholds whose values are determined experimentally.

Measurements such as standard deviation can only account for second order statistics of the color distribution of a voxel, and usually assume a Gaussian distribution. A voxel usually represents a small 3D volume as a single value (e.g., its

intensity). Since its extent may span color boundaries or regions of non-trivial texture, the constant color assumption is generally invalid. Thus, tests based on simple Gaussian statistics may well prove inadequate. In this situation, the histogram, a reasonable approximation to any distribution, is a more appropriate choice. A consistency test based on histograms is proposed by Stevens *et al.* [125, 145] as follows:

$$\forall_{i,j} \text{Hist}(\pi_i^V) \cap \text{Hist}(\pi_j^V) \neq \emptyset \qquad i \neq j \tag{2.14}$$

This declares a voxel to be consistent if every pair of cameras that can see the voxel observe it to have the same color. In other words, the histograms of the voxel's projections to the image planes of the two cameras intersect. If there is a single pair of views that do not have overlapping colors, the voxel is considered inconsistent. However, such a histogram intersection test is unreliable when a voxel is only visible by a small number of pixels in some images. Hence, if this number is below a certain threshold, we may exclude the corresponding image from the consistency test [145, 125].

2.2.3 Visibility

Visibility determines whether a scene point can be seen by a camera, and in turn, which cameras to be used in evaluating photo consistency. This relies on object geometry, i.e., the information we wish to reconstruct, and camera setup. The visibility and photo consistency problems are often coupled, as illustrated in Figure 2–9. Fortunately, the visibility problem can be simplified. When the camera configuration satisfies the *ordinal visibility constraint* of Seitz and Dyer [141], there exist efficient algorithms ([141, 40, 24, 124]) to process scene points.

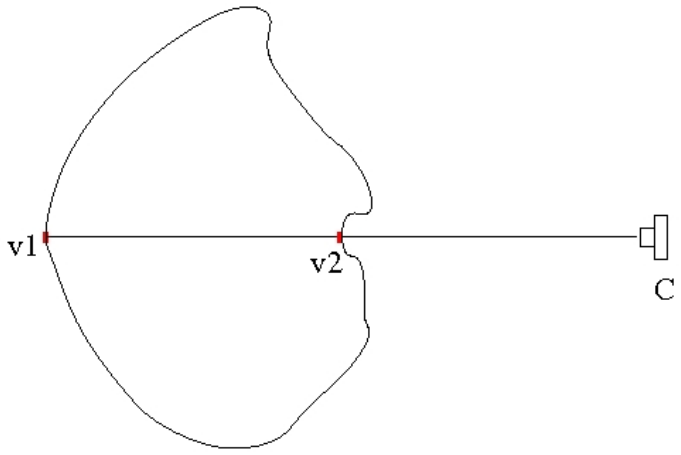


Figure 2–9: The coupled problem of visibility and the photo consistency test. Whether $v1$ is visible to camera C depends on the occupancy of $v2$, which in turn depends on the photo consistency test on $v2$. In order to test the photo consistency of $v2$, one must first determine its visibility. Note that this is a global problem, as $v1$ and $v2$ can be far apart.

> **Ordinal visibility constraint:** *There exists a norm* $||.||$ *such that for all scene points* P *and* Q*, and input images* I*,* P *occludes* Q *in* I *only if* $||P|| < ||Q||$.

This constraint simply states the fact that a distant point may be occluded by a closer one from the viewpoint of a camera. It implies that the visibility problem can be solved if the cameras are arranged in such a way that they each observe the same occlusion ordering of scene points. In this case, the points can be processed in the same order, from near to far, for all cameras. Two example camera configurations are shown in Figure 2–10, while two point-scan strategies are shown in Figure 2–11. All scenes that lie outside the convex hull of camera centers satisfy the ordinal visibility constraint.

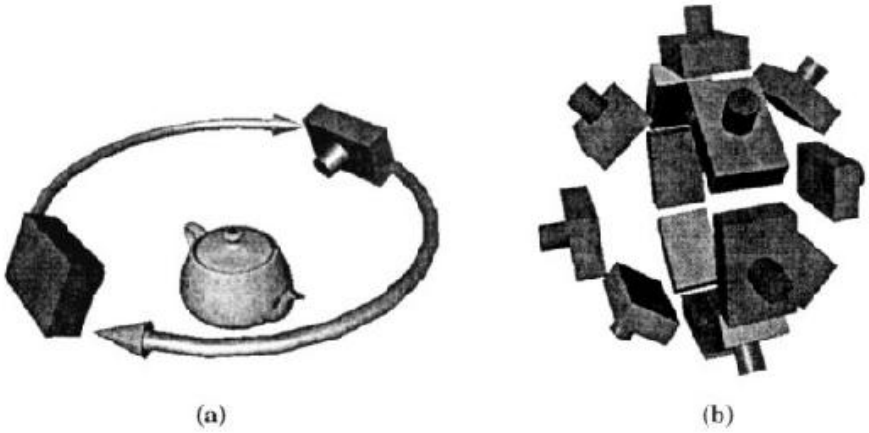


Figure 2–10: Two camera configurations satisfying the ordinal visibility constraint. Reprinted from Seitz and Dyer [141] with permission.

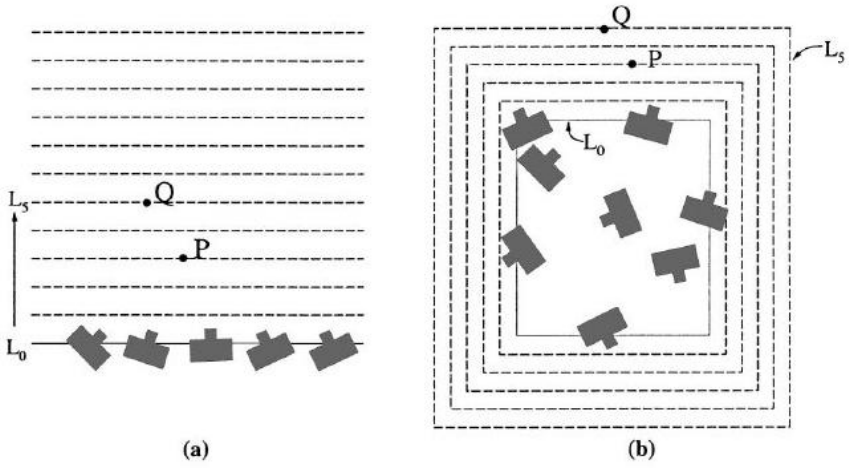


Figure 2–11: Two one-pass point scan examples. Reprinted from Seitz and Dyer [141] with permission.

Space carving and level set methods start from an initial model that encloses scene objects, for example, a volume including all voxels [90, 89, 13, 145]), or a surface enclosing scene objects ([48, 121, 148, 75, 76]). These two methods work iteratively. At each step, visibility can be computed based on the known geometry, and then the model can evolve either by carving the volume or updating the level sets.

2.2.4 Space Carving Methods

Space carving methods make use of the voxel representation. The problem is to determine voxel occupancy and assign each voxel a color consistent with all images of the scene. A voxel color is *invariant* if and only if its color is the same in all the cameras that can see it [141]. A set of voxels whose colors are invariant is said to be a *color consistent* set. The volumetric model is the union of all possible sets consistent with the reference images, in other words, the *photo hull* from these images [90].

Space carving starts from an initial volume that contains all objects, then proceeds to carve away photo-inconsistent voxels. The remaining voxels constitute the model. Voxel coloring [141] is a special case of space carving, where cameras satisfy the ordinal visibility constraint, so it can start from an empty set and construct the model by adding photo-consistent voxels. The pseudocode below shows the generic procedure of space carving.

```
set all voxels occupied
loop {
    for every occupied voxel V {
        find π^V
        if ( π^V is inconsistent )
            carve V
    }
} until (no voxel carved on this iteration)
```

Space carving methods [141, 90, 89, 19, 145, 132] favour *maximal surfaces* [140]. They remove voxels only when these are photo-inconsistent. The result is the largest photo-consistent reconstruction, the union of all possible photo-consistent sets or the *photo hull* [141, 90]. Because no assumption is made of smoothness, these techniques can reconstruct arbitrary shapes, including high curvature or thin structure. However, in regions of low surface texture, they tend to produce extra voxels [141, 90].

2.2.5 Level Set Methods

Level set methods [48, 121, 148, 75, 44, 76, 142, 117] try to find a minimal surface that is photo-consistent with the reference images. This approach is similar to space carving in that both start from a large bounding volume that includes scene objects, then shrink it inward to the objects. They differ in that the former can locally extend when needed, and favor minimal surfaces [140] that tend to smooth high curvature regions. As shown in Equation 2.15, an energy functional $E(\cdot)$ of a surface S is defined, where P is a surface point, N is its surface normal (refer to Figure 2–12), and $\Phi(P, N)$ measures how well point P agrees with the reference images, increasing

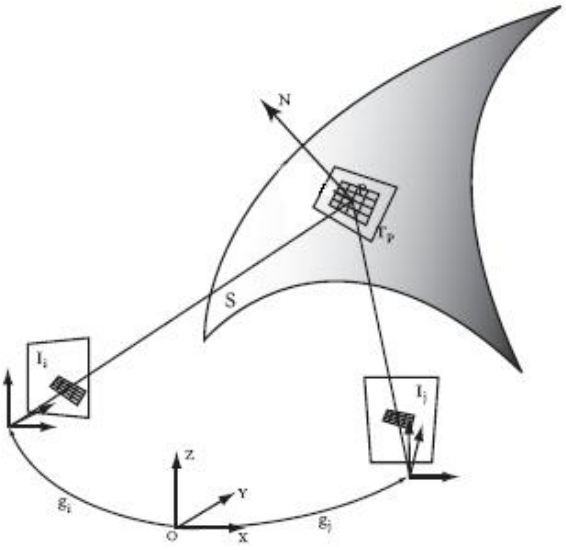


Figure 2–12: The tangent patch Γ_P of a surface point P and its projections in images.

as a function of distance of P from the object surface. The objective is to find a surface minimizing this energy functional. This leads to a set of Euler-Lagrange equations that guide how the initial shape evolves to the object model [142, 117].

$$E(S) = \int_S \Phi(P, N) dA \tag{2.15}$$

2.2.6 Shape from Silhouettes

A silhouette is a binary image where every pixel indicates whether it is a projection of objects or background. Silhouettes provide strong constraints on the shape of an object. From the view center of a silhouette image, a cone is defined within which the 3D object must lie. This is true for any number of silhouette images, so

the 3D shape must lie within the intersection of all these cones (refer to Figure 1–4). In the limit where the number of such silhouettes from outside the convex hull of the object approaches infinity, the result is known as the *visual hull* [92]. Volumetric models can be reconstructed from a set of silhouette images by many algorithms [2, 30, 92, 93, 94, 103, 104, 122, 147, 150, 151]. Similar to space carving and level set methods, these can start from a volume enclosing the entire scene, then project each voxel into every silhouette image to see whether it is inside the silhouette. The octree is often used to make scene traversal more efficient [122, 151, 2, 150].

Shape from silhouettes can be applied to surfaces, as well as volumes. Some algorithms [31, 158] use apparent (or occluding) contours to build a surface-based representation. Occluding contours or silhouettes can be helpful in the reconstruction. Cross and Zisserman [35] demonstrated how silhouettes are used to initialize a voxel volume for space carving.

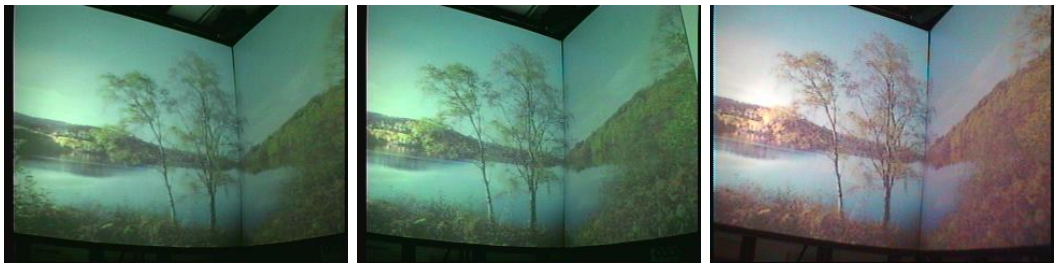
2.3 Active Vision Methods

The methods discussed so far work only with passively obtained camera images. Active vision methods belong to another category of techniques as they can interact with their environments. Such systems usually include a projector (or laser) and a camera [149]. The optical geometry of such an active vision system is the same as a binocular stereo system, with the understanding that a projector acts as the inverse of a camera, emitting light rays instead of receiving them. The system first projects a well-chosen pattern into a scene, then detects the pattern in the camera image. This is a similar process to stereo matching. Once the match is established, the following

3D reconstruction can be performed by triangulation, in a similar manner to stereo reconstruction.

Various patterns can be designed to facilitate the matching process according to the demands of particular application environments. One can simply sweep a light plane [78, 149] across an object with a contrasting color. A flying spot [129] or a color dot pattern [39] works as well. One can also project a hierarchical set of gray stripe patterns [133] or a series of alternating color stripes [20, 170] to encode a scene. While active vision methods can simplify reconstruction, they require synchronization between cameras and projectors. In general, they are more applicable to scenes with limited texture. For objects with complicated colors and textures, the design of a suitable pattern remains a challenge.

CHAPTER 3
Color Correction Methods with Applications to Digital Projection Environments



a b c

Figure 3–1: images captured by three different cameras.

Having surveyed the set of techniques relevant to 3D object modeling, we now turn to problems of observing a scene through multiple cameras with heterogenous properties, in particular, when the scene is generated, at least in part, by multiple projectors with non-uniform characteristics. Due to a number of factors including illumination, optics, sensor characteristics, and hardware processing, different cameras typically produce different color values for the same objects or scenes, as illustrated in Figure 3–1. These differences complicate the task of computer vision applications involving the use of more than one camera. An approach is thus required to correct

---

This chapter is based on the author's paper [165]. Permission to include these contents here is provided by the publisher.

the images so that colors of the same object appear to be similar in the output from each camera.

This correction typically takes one of two forms. In the first, the mapping is found between the true color values (which may be unavailable in many cases) and the observed colors for each camera, while in the second, the transform between each camera and one reference camera is found. The latter approach is generally simpler, as it is determined solely by the camera parameters and the mapping can be characterized by a relatively small number of data samples. In this thesis, we focus more on the second case; that is, we are concerned with improving the color consistency among cameras. We assume, in either case, that the cameras focus on approximately the same portion of the scene, thus receiving similar visual information. However, we do not wish to impose additional constraints, such as an assumption of uniform illumination or matte objects.

A closely related but different problem is that of color constancy [6, 8], in which a relationship is sought between surface colors and illumination, in order to map the observed color to the correct one under some canonical illumination [54]. Common solutions include the *gray world* approach, which assumes that the average color in an image is gray; the *white patch* approach, derived from retinex theory [91], which assumes that the maximum value of each channel is white; and neural network methods [29], which estimate the illuminant chromaticity of an image using a neural network, which usually needs a large database of illuminants and reflectances(surfaces) for training. They also assume each image is taken under one uniform illuminant, which

is not valid in our environment. Finlayson *et al.* [52] consider varying illumination, but assume a difference in illumination can be identified. Other approaches involving gamut mapping methods [54, 49] and Bayesian methods [22] require either large datasets of reflectance spectra from a wide variety of common objects or knowledge of the camera sensor responses, both of which are generally difficult to obtain. Some other related works on color calibration and color reproduction can be found in [72, 81, 161, 154]. In this thesis, we focus on finding the relationship between cameras; that is, we try to make color appear consistent between them. Thus, we do not need to estimate the chromaticity of the actual illuminant, nor do we require a large training set, as the problem can be addressed sufficiently with relatively sparse data and a simple method.

For our specific application, we are interested in color correction for an immersive environment employing digital rear projection (see Figure 3–4), in which the output of several, possibly heterogeneous cameras, must be correlated. In such an environment, the color of any pixel as registered by each camera is affected by many factors, including non-uniform background lighting conditions, projector color gamut, uneven intensity distribution over the screen, and differing camera poses and sensitivities. As a result, the appearance of colors obtained by the cameras can vary widely, as pictured in Figure 3–1. While an inter-projector calibration that produces a uniform color response across projectors would help reduce these effects, the problem of different camera responses to these pixels remains. It is this problem on which we focus here, leaving for future work the question of projector calibration.

Following an overview of other color constancy methods, we investigate several options for dealing with the color correction problem. We first examine linear methods in Section 3.1 and then compare these with our proposed approach of a neural network in Section 3.2, concluding the chapter with a summary of experimental results in Section 3.3.

3.1 Linear Color Correction Methods

In this section, several methods based on linear models are discussed. The RGB color space is used in the thesis because it is the most popular space used in sensor and display devices. Most methods in the thesis should be applicable to other color spaces, but a comparison of different color spaces is beyond the scope of the current thesis. The color transfer method in Section 3.1.4 converts the RGB space to an $l\alpha\beta$ space first, then works on that space, and converts back to RGB at the final step. The least squares approximation method (Section 3.1.3) requires the estimation of a transform matrix, which is similar in approach to the training of a neural network, as described in Section 3.2. However, the remaining methods are based only on single-camera models, and as such, do not undergo any training or estimation step.

3.1.1 Gray world (GW)

The *gray world* approach assumes the average color of an image is some predefined value of "gray," for example, half the value of the maximum intensity for each color component, (128,128,128). Based on this assumption, image colors are corrected through the following normalization:

$$R_n = R_o * 128/\bar{R}$$

$$G_n = G_o * 128/\bar{G}$$
$$B_n = B_o * 128/\bar{B} \tag{3.1}$$

where (R_o, G_o, B_o) is the original color, $(\bar{R}, \bar{G}, \bar{B})$ is the average color, and (R_n, G_n, B_n) is the corrected color. One might also consider the use of the average color components $(\bar{R}, \bar{G}, \bar{B})$ from an arbitrary reference camera, and use these, rather than the fixed value (e.g., 128) as the normalizing term. However, this may suffer problems if the reference camera's color distribution is not well-balanced.

3.1.2 White patch (WP)

The *white patch* approach is similar to the *gray world* method but assumes that the maximum value of each channel should correspond to full white (255, 255, 255). Image colors are corrected through the following normalization:

$$R_n = R_o * 255/R_m$$
$$G_n = G_o * 255/G_m$$
$$B_n = B_o * 255/B_m \tag{3.2}$$

where, (R_o, G_o, B_o) is the original color, (R_n, G_n, B_n) is the corrected color, and R_m, G_m, and B_m are the maximum observed color components in the three channels, respectively.

Again, we may consider using one camera as a reference, with the same caveats as earlier.

3.1.3 Least squares (LS) approximation

The gray world and white patch approaches use diagonal matrix transforms, assuming that the different channels are independent. While various research suggests that diagonal transforms should suffice [50], or suffice with sensor sharpening [51], this is not the case in general with complex scenes. Worse still, sensor sharpening techniques may be unstable [7].

Instead, we consider the use of a full matrix transform, i.e.,

$$(\mathbf{C'_1}, \mathbf{C'_2}, \ldots, \mathbf{C'_n}) = \mathbf{T} \cdot (\mathbf{C_1}, \mathbf{C_2}, \ldots, \mathbf{C_n}) \tag{3.3}$$

in which $\mathbf{C'_i}$ and $\mathbf{C_i}$ $(i = 1, \ldots, n)$ are colors from two different cameras and $\mathbf{T}$ is the transformation matrix between them. From a set of corresponding colors from two cameras, $\mathbf{T}$ can be estimated by least squares approximation methods. Here, we use the color from the first row of Figure 3–2 to estimate $\mathbf{T}$. The image from Figure 3–2c) is taken as the standard color or reference, from which we estimate transforms between it and the images produced by the other two cameras. These transforms are then used to correct the colors.

3.1.4 Color transfer between images

Reinhard *et al.* [128] proposed a color transfer method that can be applied to color correction. It first decorrelates the RGB values to an $l\alpha\beta$ color space and then computes the statistics (mean and standard deviation) of source and target images. The source colors are corrected by scaling and offsetting according to the mean and standard deviation of the target image, as follows:

$$
\begin{aligned}
l'_s &= (l_s - \overline{l_s}) * \frac{\sigma_t^l}{\sigma_s^l} + \overline{l_t} \\
\alpha'_s &= (\alpha_s - \overline{\alpha_s}) * \frac{\sigma_t^\alpha}{\sigma_s^\alpha} + \overline{\alpha_t} \\
\beta'_s &= (\beta_s - \overline{\beta_s}) * \frac{\sigma_t^\beta}{\sigma_s^\beta} + \overline{\beta_t}
\end{aligned} \tag{3.4}
$$

Where, $\bar{c}_i, \sigma_i^c, c = l, \alpha, \beta$ is the mean and standard deviation of an image for each channel, respectively, and $i = s, t$ refers to the source and target, respectively. Following this transform, the image is converted back to RGB space. The conversions between $l\alpha\beta$ and RGB spaces can be found in [128].

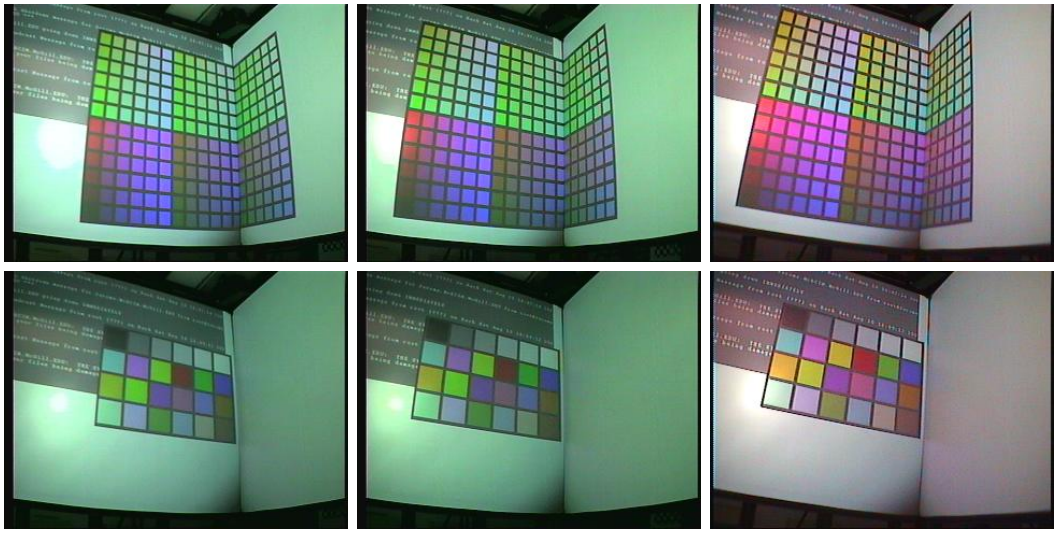
We also consider the use of color transfer, followed by the gray world method, in order to normalize the results of the transfer. The results of this combination, as illustrated later in Fig 3–5 and Fig 3–6 appear to be superior to the color transfer itself.

3.2 Neural Network Color Correction

Suppose there is a surface patch in space. A camera S captures its color as $\mathbf{C}_S$, and another camera T captures it with a different color $\mathbf{C}_T$. We want to find a mapping $f(\cdot)$ between camera S and T, as Equation 3.5 suggests.

$$
\mathbf{C}_T = f(\mathbf{C}_S) \tag{3.5}
$$

The previous methods assume the mapping is linear. For complex scenes, this sometimes proves inadequate to correct colors from different cameras. In our environment, the color of any pixel as registered by each camera is affected by many factors,



a b c

Figure 3–2: 1st row: training data; 2nd row: validation data.

including non-uniform background lighting conditions, projector color gamut, uneven intensity distribution over the screen, and differing camera poses and sensitivities. All these factors may contribute to a non-linear mapping $f(\cdot)$. While it is often difficult to find a suitable, explicit, nonlinear representation, neural network methods [29] have been shown capable of performing similar tasks, such as estimating the illumination of an image, given a large database of known illuminations and surface colors.

3.2.1 Neural Network Architecture

The network architecture used here was a simple two-layer backpropagation network (BPNN) with 10 hidden-layer neurons, as shown in Figure 3.2. The inputs are the source RGB values, and the outputs are the corrected RGB values. Suppose neuron j is one of the output nodes, corresponding to the red, green, and blue channels.

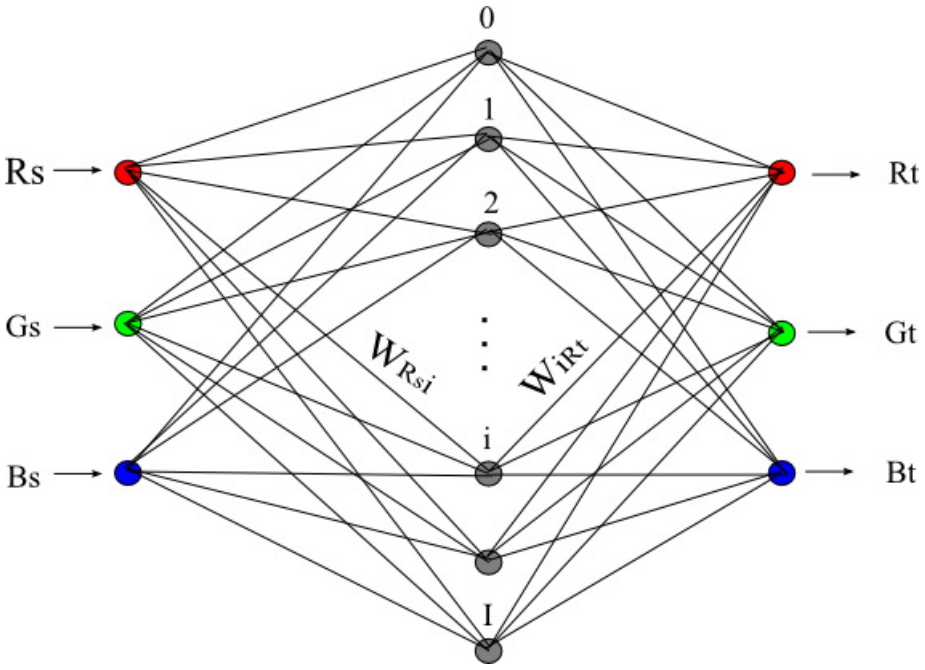


Figure 3–3: the BPNN architecture

The error $e_j(n)$ at neuron j for the n'th training data is the difference between the desired value $d_j(n)$ and its estimated value $y_j(n)$, provided by the network output. The total error E, which we wish to minimize, is calculated as the sum of squared errors over all output neurons and all training data, as defined in Equation 3.6.

$$e_j(n) = d_j(n) - y_j(n)$$
$$E = \sum_n \sum_j e_j^2(n) \tag{3.6}$$

In our case, $d_{0,1,2}(n)$ are the RGB values from the reference camera. The estimated value at neuron j is a function of inputs $x_i(n)$ from all neurons in the hidden layer, i.e., $y_j(n) = \varphi(\sum_i w_{ij} x_i(n))$, where $\varphi()$ is the activation function, chosen here

as a logistic sigmoid function, $\varphi(t) = 1/(1+exp(-t))$, and w_{ij} is the weight connecting hidden neuron i and output j. Similarly, the value $x_i(n)$ at hidden neuron i is a function of the input data $s_k(n)$, where $s_{0,1,2}(n)$ are the RGB values of the input color, $x_i(n) = \varphi(\sum_k w_{ki} s_k(n))$. The network adapts its weights so as to minimize the total error between the estimated colors and the desired colors, E in Equation 3.6, by a standard BPNN algorithm. Further background and general theory about BPNN training can be found in many neural network reference texts, e.g., Haykin [67].

Since we need only find the relation between colors from different cameras, assuming the same lighting is applied to the views of each camera, a simple training set proves to be sufficient. The training data consists of 216 color checkers, uniformly distributed in RGB space. Validation data is provided by the image of the Macbeth 24-color checkers pattern, projected onto the screens and captured by the cameras, as shown in Figure 3–2. The samples are collected manually by clipping a rectangle in each color square and then computing its mean value. The values obtained from one of the cameras (Figure 3–2c) are taken as the reference and the color values from the remaining cameras are corrected accordingly. For example, using the 216 color samples in Figure 3–2a and c, we can train a BPNN that maps camera a's colors to camera c's. The same can be done to Figure 3–2b and c.

3.2.2 Empirical Results

To evaluate the performance of these methods empirically, we measure the error both as absolute differences of the individual color components, δ_r, δ_g, and δ_b, as well as the Euclidean distance between the components of the true object color, $\mathbf{x_t}$, and the estimated color, $\mathbf{x_e}$, as follows:

$$Err = \frac{\sqrt{\delta_r^2 + \delta_g^2 + \delta_b^2}}{255} \tag{3.7}$$

where 255 is the maximal value for each color component.

The error statistics for a corrected camera corresponding to the image of Figure 3–1a are provided in Table 3–1. Since both corrected camera images exhibited similar results, we only list one of these here. For comparison purposes, we include the results obtained by the other methods described in section 3.1. Since the least squares approximation method involves the estimation of a transform matrix based on observed data, this is similar in approach to the training session of the neural network, so it is also meaningful to compare performance on independent test data. However, the other methods do not include such training steps, so no comparison with test data is relevant.

The results obtained demonstrate, both quantitatively and qualitatively, the superiority of the backpropagation neural network. Complex images, such as those shown in Figure 3–5, exhibit significantly better correction by the BPNN method than with the other approaches.*

3.2.3 Digital Projection Results

We applied the various color correction methods described previously to a set of sample images taken by video cameras in our environment. The results are shown in Figure 3–5 and Figure 3–6. Since some of the simple methods are based on

* The differences are only apparent in a color printout or screen display of this document.

method	training set		validation set	
	μ	σ	μ	σ
Gray World	6.53 (4.43,2.05,3.31)	3.26 (3.17,1.62,2.39)	-	-
White Patch	8.89 (6.25,3.16,3.86)	4.65 (4.61,2.30,3.17)	-	-
Color Transfer	7.33 (4.84,2.62,3.67)	3.39 (3.65,2.28,2.67)	-	-
Color Transfer + Gray World	6.14 (3.39,2.43,3.03)	3.30 (3.82,1.72,2.92)	-	-
Least Squares	7.22 (5.02,1.87,3.76)	3.58 (3.35,1.65,2.87)	15.75 (12.13,5.37,6.43)	5.53 (5.46,3.81,4.28)
BPNN	3.69 (1.89,1.38,2.25)	2.15 (1.63,1.11,1.95)	14.03 (9.94,6.50,5.89)	7.02 (5.30,4.62,4.68)

Table 3–1: Error statistics for color correction applied to one of the cameras, expressed as percentages. μ is the average error and σ the standard deviation of these measures. The top row of each cell corresponds to the Euclidean error metric, whereas the bottom row corresponds to the individual component differences, $\delta_r, \delta_g, \delta_b$.

strong assumptions, such as constant illumination, which are not satisfied in our environment, these often fail to perform adequately.

Again, the neural network method outperforms other strategies. This is particularly evident in Figure 3–6, in which the presence of a person standing in the scene changes the illumination level from that used during training. In this example, all of the methods apart from the neural network approach exhibit noticeable degradation. While the transform matrix, T, used for the least squares and the neural network methods were estimated or trained using the data of Figure 3–2, i.e., independent of both test scenarios, only the neural network method proved to be robust to the change in illumination.

3.3 Conclusions

We have considered the problem of color correction for a set of heterogeneous cameras in a general environment, in which constant illumination cannot be assumed. Various methods, based on solutions to the color constancy problem, were applied to

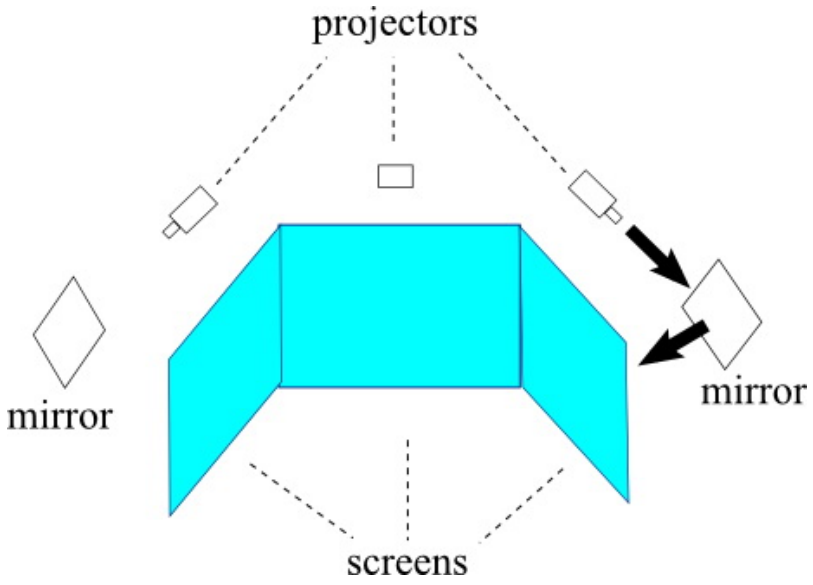


Figure 3–4: Illustration of the rear projection environment used for our application. Images are sent from projectors to mirrors and reflected to screens.

this task and their results compared. We found that under non-idealized conditions, our proposed use of a simple backpropagation neural network achieves results that are superior to other methods for correcting images from different cameras to produce results that appear similar to each other in color. The neural network method allows for simple training and proves to be robust to significant scene variations.

Although we have only evaluated these approaches within our rear projection environment described previously, we see no reason why the neural network approach would not succeed equally well in other environments or on natural scenes, provided suitable training data, such as the Munsell color checkers, can be used.

An interesting avenue for ongoing research is how to extend these results to the far more challenging problem of color correction for heterogeneous projection equipment and cameras that are no longer viewing the same portion of the scene.

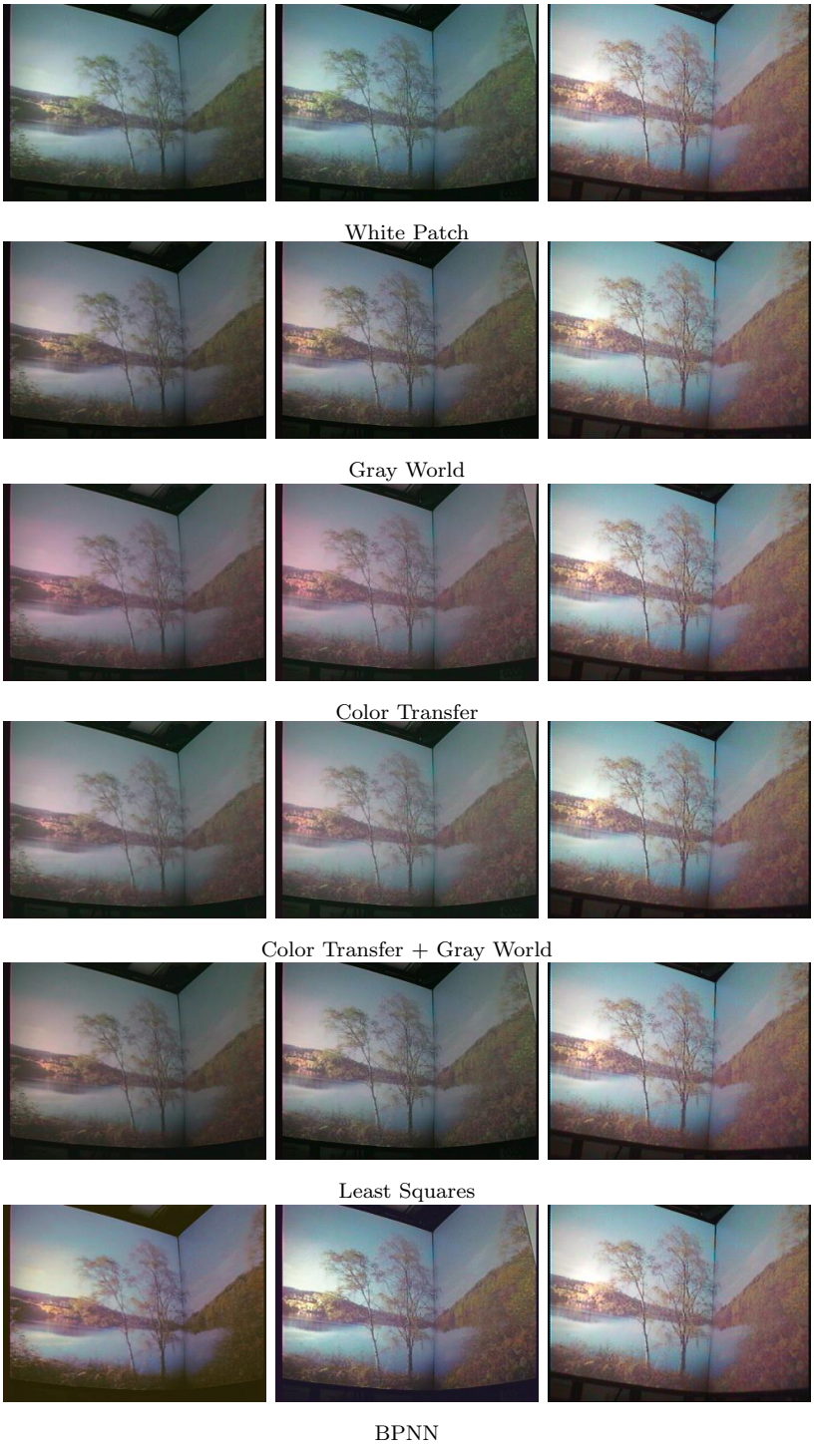
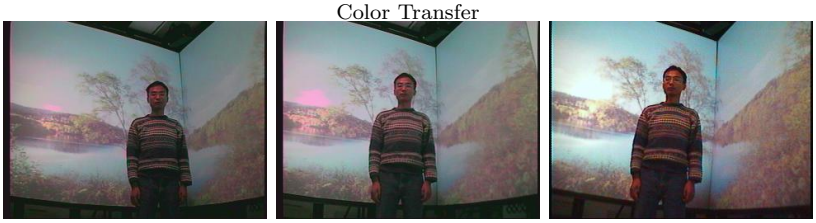


Figure 3–5: Comparison of color correction results for the original scene.

Original images

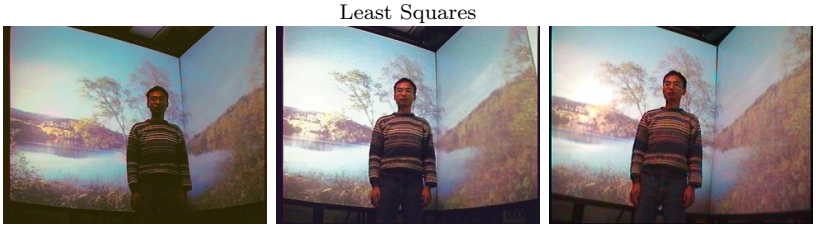
Gray World

Color Transfer



Color Transfer + Gray World

Least Squares



BPNN

Figure 3–6: Color correction results including a person standing in the scene.

CHAPTER 4
Improving Depth Maps by Nonlinear Diffusion

Dense depth maps produced by stereo matching algorithms often have problems in regions of occlusion or limited texture. Many stereo algorithms treat such regions as having the same or similar depth as neighbouring areas by using smoothness constraints, thus causing objects to appear larger or wider, an obviously undesirable effect. Various algorithms have been developed to address this problem. For example, Kanade and Okutomi [79, 116] use adaptive windows or multiple cameras, and Scharstein and Szeliski [135] aggregate support in stereo matching by nonlinear diffusion instead of using an explicit window. Belhumeur and Mumford[12], Intille and Bobick [70], and Geiger *et al.*[59] incorporate occlusion information directly into their stereo matching algorithms by Bayesian methods and use dynamic programming for optimization. Gamble and Poggio [57] integrate discontinuity and depth information by a coupled Markov random field model. More elegant methods adopt a global strategy and use graph cuts to minimize the energy, which can consider such situations explicitly [21, 87, 130]. Although these approaches solve the problem to a certain extent, they are insufficient to address the complexities of general scenes, such as that illustrated in Figure 4–2, where there are challenges of large disparity

---

This chapter is based on the author's paper [166]. Permission to include these contents here is provided by the publisher.

ranges, significant occlusion, large areas of constant color, repeated patterns, and camera distortions. Further, while many algorithms consider the binocular case, they cannot be extended to $N(\geq 3)$ camera problems, employing a generic configuration. For such cases, rectification may be difficult, if not impossible, and the range of disparities can be very large.

Improvements to the depth map can be obtained through filtering or interpolation. For example, median filters or morphological filters can fill small gaps and correct depth errors (e.g., [138]), but their ability to do so is rather limited. Linear interpolation techniques (e.g., [83]) can fill gaps along epipolar lines or scanlines when images are rectified. The drawback is that these methods use only the information along one line, which may be difficult to estimate correctly when the epipolar geometry is complicated. Furthermore, such interpolation methods do not consider the information provided by other neighbouring areas. Notwithstanding these efforts, we suggest that further improvements to the depth map may be obtained through considering image intensity constraints.

The technique we propose here, inspired by Perona and Malik's work on edge detection [118], can be applied to a depth map produced by any stereo matching algorithm. It utilizes neighbouring information in a natural manner through nonlinear diffusion, filling large gaps while maintaining sharp object boundaries. The main difference between Perona and Malik's work and ours is that we use the gradient of the original image rather than that of the depth image. Hence, discontinuities in depth are assured to be consistent with intensity discontinuities, often a desirable property

[57]. Assuming that object shapes rarely vary dramatically apart from boundaries, this technique can eliminate many errors caused by textureless regions or occlusions.

In the thesis, anisotropic diffusion is applied to improve depth maps by considering image intensity information. While similar techniques have been applied to other tasks such as image enhancement [118, 156, 168, 15] and image segmentation [163], this represents, to the best of our knowledge, the first time diffusion has been used to incorporate intensity information for the stereo matching problem. Scharstein and Szeliski [135] used diffusion to aggregate support in stereo matching, but did not apply intensity gradient information during the diffusion process. Anisotropic diffusion used here can be considered as applying an oriented smoothness constraint to depth fields. This is similar to the use of smoothness constraints in a regularization framework to improve optical flow estimation [69, 112, 113, 42, 3, 162]. In that case, image gradient information is combined in a regularizer as a weighting matrix to constrain the smoothness operation in the direction perpendicular to the image gradient. The difference is that we use an anisotropic diffusion technique, while the previous works cited use regularization. Our motivation for using diffusion is that this technique can be applied to the results from any stereo matching algorithm. Image gradient constraints may be applied to depth maps during the diffusion process. Details about how such a scheme works will be discussed in Section 4.2. Unlike the case of regularization, a parameter controlling the balance between the data and smoothness constraints has to be chosen experimentally. Different choices can give very different results. For diffusion, a parameter is used to control the numerical

stability and convergence of the diffusion process. Within a certain range, a different choice of the parameter does not have a significant effect on the result.

The remainder of this chapter is organized as follows. In Section 4.1, a generalized multi-baseline stereo is presented, which is used to produce all of our sample depth maps. Section 4.2 describes our method for improving depth maps by nonlinear diffusion, Section 4.3 discusses several applications that can benefit from such an improvement, and finally, some questions and directions for future research are discussed in Section 4.4.

4.1 Generalized Multiple-Baseline Stereo

Okutomi and Kanade [116] proposed a multiple-baseline stereo algorithm that applied to parallel cameras (i.e., there are only horizontal disparities). For a general camera setup, images must first be rectified, which usually requires a re-sampling of the images, during which some information may be lost. Here, we generalize the Okutomi and Kanade algorithm to deal with an arbitrary camera configuration.

First, we assume that all cameras are calibrated, for example, using Tsai's method [157]. The parameters of camera i are represented by $\mathbf{M_i}$. Knowing the center of projection for the camera, we may compute the ray $\mathbf{r}$ passing through the center and a given pixel $\mathbf{p} = (x, y)$ in the image. If one dimension is known of the real world point $\mathbf{P} = (X, Y, Z)$ corresponding to the pixel $\mathbf{p}$, for example, if we know $Z = c$, then we can compute the 3D position of $\mathbf{P}$ by intersection of ray $\mathbf{r}$ with the plane at $Z = c$, and from this, we may also compute its projection in the image planes of the other cameras, as shown in Figure 4–1, where C_i, C_j are centers of projection for camera i, j, respectively.

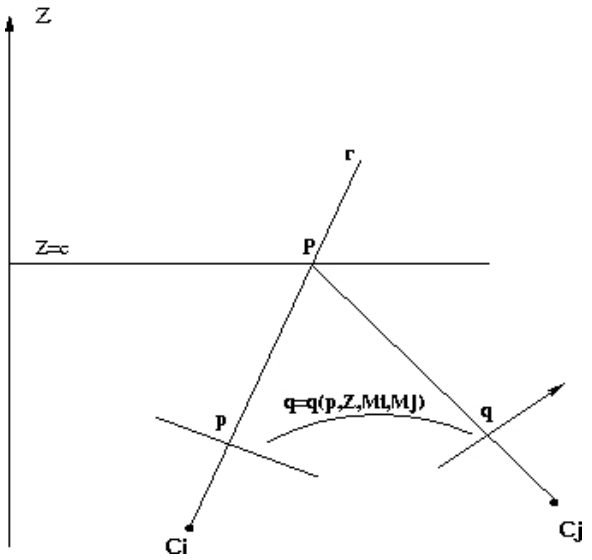


Figure 4–1: Scene point computing and reprojection

Suppose $\mathbf{q} = \mathbf{q}(\mathbf{p}, Z, \mathbf{M_i}, \mathbf{M_j})$ is a function relating pixel $\mathbf{p}$ in camera i to pixel $\mathbf{q}$ in camera j. Given that the sum of squared differences (SSD) between two corresponding regions in an image pair can be used to measure similarity between these images, the sum of SSD (SSSD) over all image pairs may be used to determine the depth.

$$SSSD(\mathbf{p}, Z) = \sum_{i \neq j} \sum_{\mathbf{p}' \in \mathbf{N_p}} (I_i(\mathbf{p}') - I_j(\mathbf{q}(\mathbf{p}', Z, \mathbf{M_i}, \mathbf{M_j})))^2 \tag{4.1}$$

where $\mathbf{N_p}$ is the neighbourhood of pixel $\mathbf{p}$, $I_i(\mathbf{p})$ is the intensity of pixel $\mathbf{p}$ in camera i.

We take camera i as a reference and compute the sum of SSDs between the images obtained by camera i and all other cameras. The best depth estimate for

each pixel $\mathbf{p}$ is the value of Z that minimizes the SSSD:

$$Z(\mathbf{p}) \quad = \quad argmin_z \text{SSSD}(\mathbf{p}, Z) \tag{4.2}$$

For best results, Z should be discretized as finely as possible subject to computational constraints.

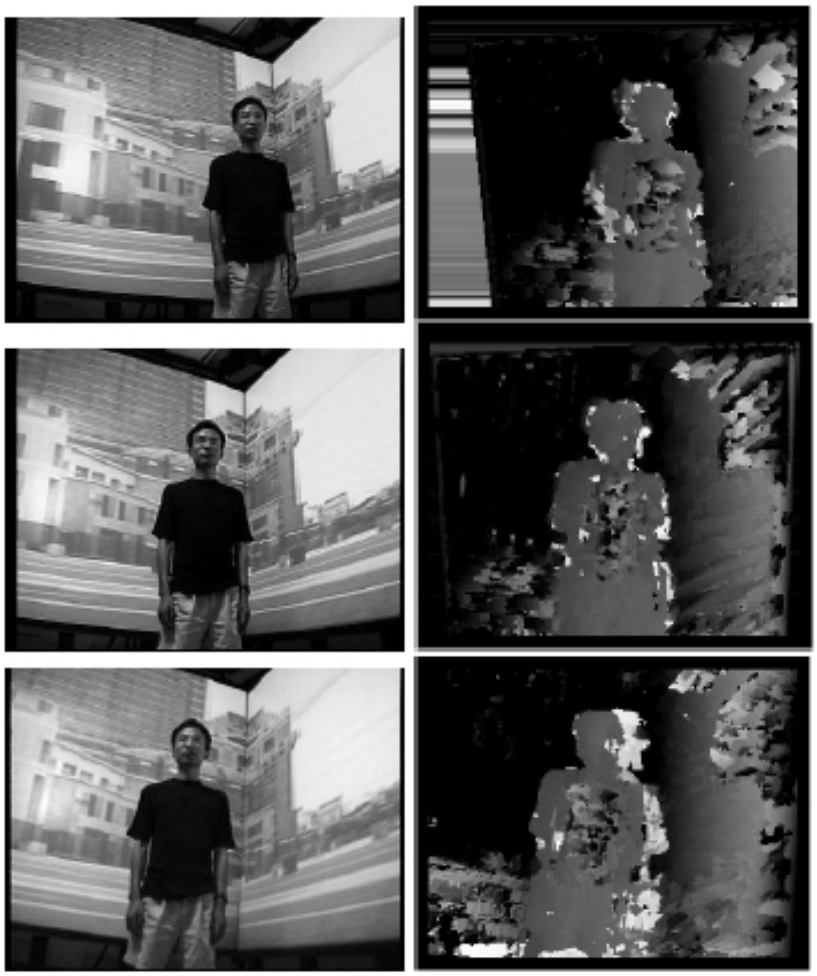


Figure 4–2: Original camera images (left column) and their corresponding depth maps (right column).

Unfortunately, this approach yields unsatisfactory results, as illustrated in Figure 4–2 with three cameras. The original images are pictured in the first column and their corresponding depth maps in the second. While the results are generally reasonable, there remain many errors, typically visible as bright points and black holes on the body, caused by occlusions, textureless regions, repeated patterns, and depth discontinuities.

4.2 Improving Depth Maps by Nonlinear Diffusion

If two nearby pixels are in the same region or belong to the same object, their respective depths should be similar. One way to achieve this smoothness constraint is to apply a weighted averaging, such as Gaussian smoothing, to the depth map. Unfortunately, such techniques also tend to blur boundaries, a problem we would like to avoid. Borrowing from Perona and Malik [118], who suggested an anisotropic diffusion method to improve edge detection, we apply the same technique to depth maps. Consider an updating function:

$$\begin{aligned} Z(x,y)_t = Z(x,y)_{t-1} + \lambda[&c_N(x,y,t)\cdot\delta Z_N + c_S(x,y,t)\cdot\delta Z_S \\ &+c_E(x,y,t)\cdot\delta Z_E + c_W(x,y,t)\cdot\delta Z_W] \end{aligned} \tag{4.3}$$

where λ is a constant for numerical stability and convergence, it should be in the range of [0, 0.25] [118].* Nearest-neighbor differences (not gradients) are indicated by $\delta Z_N, \delta Z_S, \delta Z_E, \delta Z_W$. These are defined as follows:

* For the results illustrated in this chapter, we use a value of $\lambda = 0.25$.

$$\delta_N Z = Z(x, y-1) - Z(x, y)$$

$$\delta_S Z = Z(x, y+1) - Z(x, y)$$

$$\delta_E Z = Z(x-1, y) - Z(x, y)$$

$$\delta_W Z = Z(x+1, y) - Z(x, y)$$

where N, S, E and W stand for north, south, east and west neighboring pixels. This is a discrete implementation of Laplace operator:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

(Refer also to Equations 3, 7 and 8 in [118]). The coefficients c_N, c_S, c_E, c_W in Equation 4.3 are used to weight the Laplace operator such that diffusion is applied anisotropically. These are functions of image gradient, of which some examples are shown in Equation 4.4 and 4.5.

To achieve the desired effect, the coefficient $c(x, y, t)$ should be high in the interior of each region, low at boundaries, and should have a steep threshold between the two cases. We note that the gradient G of the intensity image tends to large values along edges and small values in interior regions. Thus, an excellent choice for c is a function that responds maximally to low values of gradient, i.e., $c(x, y, t) = f(G)$, where $f(\cdot)$ takes on some shape approximating that shown in Figure 4–3, for example,

$$f(G) = e^{-(\|G\|)/K)^2)} \tag{4.4}$$

or

$$f(G) = (1 + (\|G\|/K)^2)^{-1} \tag{4.5}$$

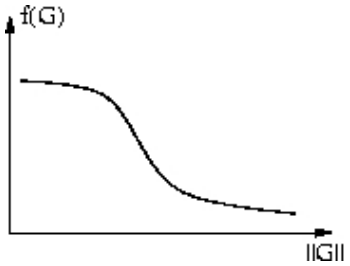


Figure 4–3: The qualitative shape of $f(\cdot)$.

Equation 4.5 is used here, which favours wide regions over smaller ones [118]. Unlike Perona and Malik's approach [118], we smooth the depth map based on the gradient of the original intensity image rather than that of the depth map itself. In this manner, we incorporate edge information into the depth map so as to recover the regions of occlusion. Through an iterative update as described by Equation 4.3, the depth map can be smoothed as desired, with significant improvements to the resulting depth map, as illustrated in Figure 4–4. The depth errors, seen as holes in the subject's body and bright points near boundaries, are gradually smoothed out, while the boundaries are kept sharp. The main disadvantage of such an iterative method is computational cost. For this example of a 320x240 resolution image, our implementation required a total running time of approximately 3.6 seconds under Matlab on a Pentium III 1.1 Ghz machine. The iteration process stops either after a

certain number of iterations or when the sum of depth changes over all pixels from one iteration to the next is below some predefined threshold.†

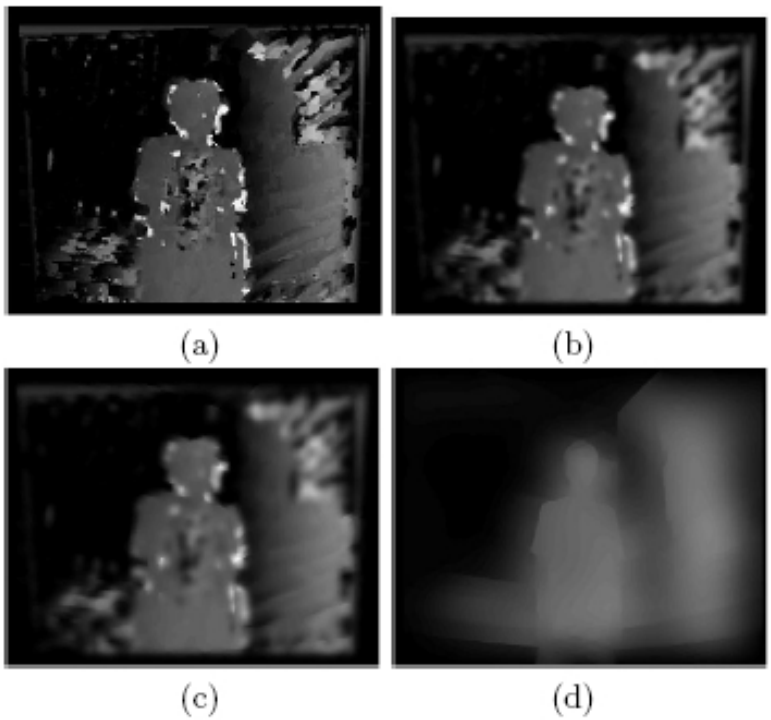


Figure 4–4: Results of nonlinear iteration diffusion: a)original depth map, b) after 10 iterations c) after 20 iterations, c) after 200 iterations.

4.3 Applications

In this section, we summarize two important applications of our nonlinear diffusion technique for improved depth maps, namely, 3D scene reconstruction and background removal.

† In the example shown here, the threshold was equivalent to a mean depth difference of 0.01.

4.3.1 3D scene reconstruction

Starting from an intensity image and a corresponding depth map, it is possible to synthesize a novel view from a nearby viewing position, as illustrated in Figure 4–5. Given the parameters of a virtual camera, the 3D position corresponding to a pixel can be computed from its position and depth. A surface can then be polygonized as described by Kanade *et al.* [80]. A mesh (two triangles) is constructed, using the 3D coordinates of four neighbouring pixels as the vertices of the triangles and the texture is obtained from the corresponding intensity map.

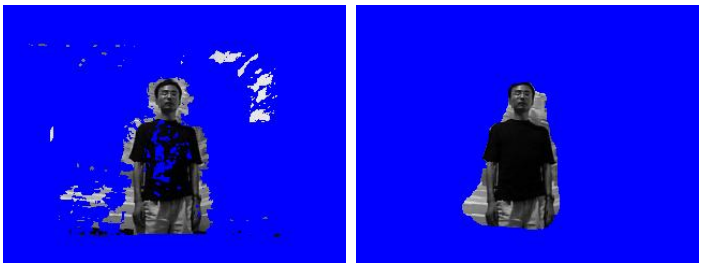
Due to estimation errors, discontinuities in the depth map, and the displacement between the virtual camera and the reference camera, some artificial surfaces typically appear in the synthesized view, as shown in Figure 4–5a. These can be eliminated by adding smoothness constraints; that is, the mesh will not be rendered unless the depths of the three vertices of a triangle are similar. Unfortunately, this results in the appearance of *holes* in the image, as seen in Figure 4–5b, 4–7d,g. However, using our improved depth map, as described above, the result appears to be improved greatly, as pictured in Figure 4–5c,4–7e.

(a) (b) (c)

Figure 4–5: A novel synthesized view a)based on the original depth map b) with the addition of smoothness constraints c) using the improved depth map of Figure 4–4d, generated by nonlinear diffusion.

4.3.2 Background removal

In virtual reality and immersive telepresence applications, it is of critical importance to extract foreground objects (typically people) from the background. Since the background may change dynamically, it is often infeasible to perform such segmentation based on a 2D reference image, such as that employed by bluescreen techniques [146]. Instead, we wish to perform this task based on 3D information from a CAVE-like environment.[‡] Captured images typically contain two perpendicular screens as background, which we can represent by the planes $X = 0$ and $Y = 0$. Since the environmental geometry is relatively simple, 3D scene information can be estimated from the depth map easily and we can then separate objects from the background by thresholding based on depth estimates. Sample results are illustrated in Figure 4–6, 4–7f,g.



(a) (b)

Figure 4–6: Segmented images based on a) the original depth map, b) the improved depth map from Figure 4–4d.

[‡] CAVE stands for Cave Automatic Virtual Environment [36].

Due to the estimation errors of the original depth map, the segmented images in Figure 4–6a and 4–7f include some portions of the background and some holes in the foreground. Using the improved depth map, instead, the results, pictured in Figure 4–6b and 4–7g, are significantly improved, although still imperfect. The remaining problems are due to the fact that the diffusion effect is determined by image gradient information, which may not be consistent with the scene geometry. In Figure 4–6 and 4–7, the background consists of planar screens, but the corresponding gradients are not flat because of the complex projected images appearing on them. We note that this situation is likely to pose problems for many stereo matching techniques, making use only of the visible light spectrum. As a result, occlusions still produce some artifacts near object boundaries, which cannot be removed entirely by diffusion. Due to the difficulty of tuning the diffusion process, smoothing over boundaries may still occur, thereby resulting in the occasional depth error.

4.4 Discussion

We have demonstrated that depth maps can be improved by nonlinear diffusion techniques, reducing the problems caused by textureless regions and occlusions. Since the diffusion process is based simply on image gradient information, it may be applied as a post-processing step to the benefit of a wide range of applications.

However, it is clear that nonlinear diffusion is not a panacea. The amount of improvement possible to the depth map is limited by the initial quality of the stereo matching algorithm; errors in the initial depth estimates tend to be propagated during the diffusion steps. Thus, it would be useful to have some method of evaluating the quality of the initial depth map. While no reliable measurements

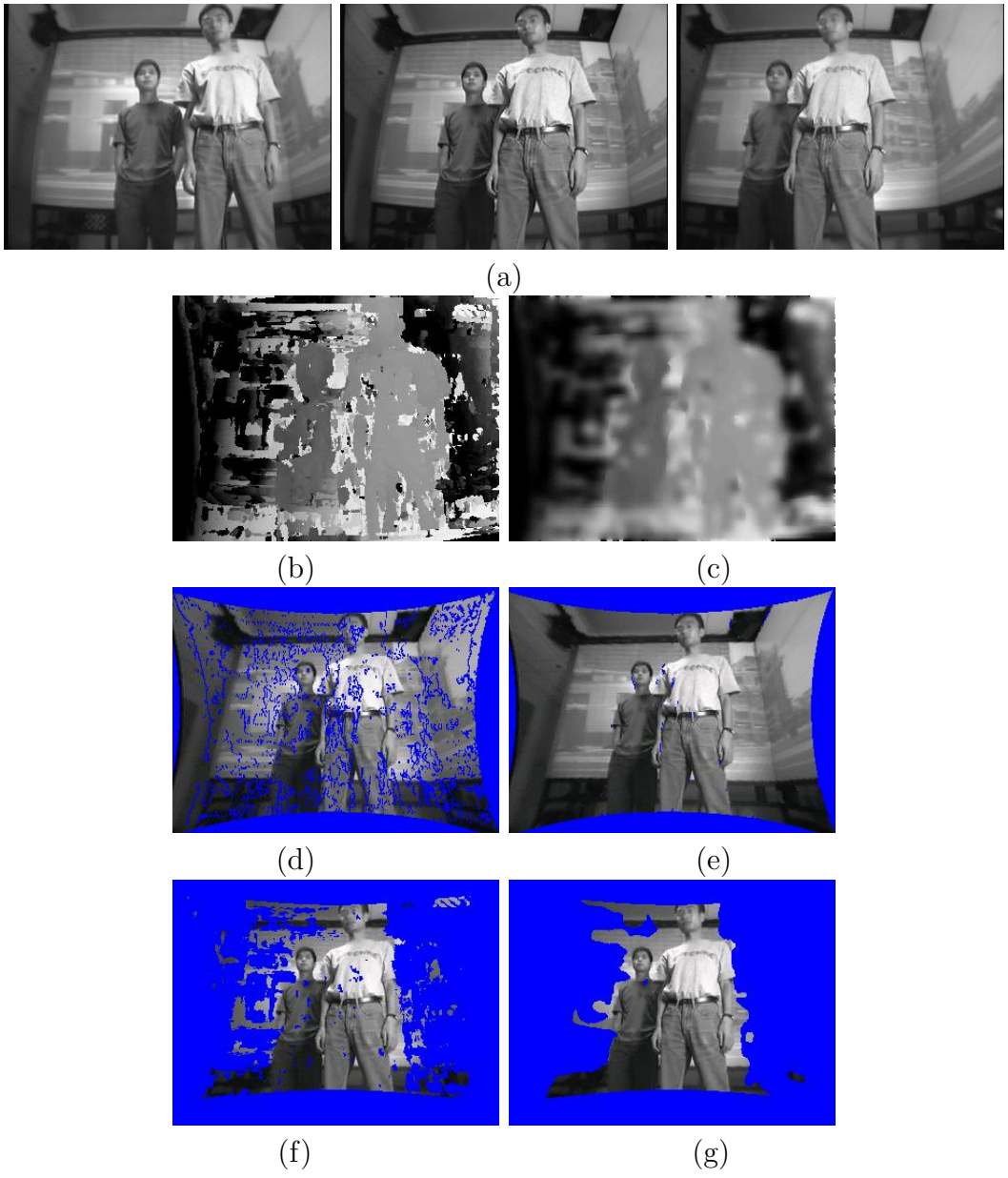


Figure 4–7: (a) original images, (b) depth map, (c) depth map after diffusion, (d) reconstruction based on (b), (e) reconstruction based on (c), (f) segmentation based on (b), (g) segmentation based on (c)

exist to evaluate the quality of stereo matching results, certain cues may be useful. For example, Scharstein and Szeliski [135] proposed a disparity certainty measure for each location based on *winner margin* or *entropy*, which may be used to estimate an overall certainty. Such certainty measures can be used to determine automatically the need for a post-processing step and might be able to suggest earlier stopping criteria for the nonlinear diffusion iteration. A similarity score (e.g., NCC) of a pixel can indicate the confidence of a match, i.e., the correct match should have a high score, although the converse is not necessarily the case. Egnal *et al.* [47] suggested a stereo confidence metric based on such cues. An improvement to our method may be obtained by weighting areas based on the degree of confidence in the corresponding area of the original depth map, thus reducing the influence of errors in the initial depths. Another issue affecting performance is the choice of a better *edge-stopping* coefficient function $f(G)$ than the one currently used (Equation 4.5). The function should have a shape as indicated in Figure 4–3. The location and steepness of the transition from high to low values determines how diffusion is constrained when the gradient is high, i.e., when there is an edge between pixels. Black *et al.* [15] analyzed anisotropic diffusion in a statistical framework and related this technique to robust estimators and regularization with a line process. They studied desirable properties of an *edge-stopping* coefficient function that can be useful in their design. It might also be worth changing the diffusion coefficient function by using the depth gradient instead of the image intensity gradient after a sufficient number of diffusion iterations have been applied to ensure reasonable propagation of depth information. According to Perona and Malik [118], the depth gradient should be helpful to smooth a depth

map. The difficulty of such an approach rests in determining when to make such a switch. Ideas concerning stopping rules discussed by Mrazek and Navara[111] might be useful for this purpose. In our continuing research, we hope to develop such ideas further.

CHAPTER 5
A Probabilistic Observation Regarding Voxel Occupancy

Stereo matching can only generate depth maps and partial models of a scene. For a full reconstruction, some techniques are needed to merge the partial models. In the remainder of this thesis, we focus on object space methods, using a voxel representation. As discussed in chapter 2, visibility information is important. If it were to be modeled, it appears unavoidable to follow the space carving approach, and process voxels in an iterative manner. This would be computationally expensive and as such, undesirable from our perspective. On the other hand, visibility can be, to a certain degree, considered implicitly. When two cameras observe the same color projection from a voxel, this is a strong indication that the voxel is visible to both cameras, and in turn, influences the value calculated in measurements utilizing color similarities. We believe that it should be possible to infer voxel occupancy from local information while not modeling visibility information, and investigate this hypothesis in the thesis. This chapter analyzes the characteristics of different voxel categories. Based on this analysis, a new photo consistency test is proposed.

This chapter is based on the author's paper [167]. Permission to include these contents here is provided by the publisher.

5.1 Voxel Categories

We assume Lambertian surfaces, so pixels that are the projections of the same voxel should have the same color. We also assume that a voxel is small enough so that it has only one color within it and large enough so that its projection to a camera covers at least one pixel.

Given a set of reference images $I_1, I_2, \ldots, I_N$, we want to determine if a voxel v is occupied and what is its color. Suppose p_i^j is the jth pixel that is v's projection in image i ($0 \leq j \leq N_i$, N_i is the total number of pixels of v's projection in image i), and $C_i^j = I_i(p_i^j)$ is its color. $C = \{C_i^j | i = 1, \ldots, N, j = 0, \ldots, N_i\}$ is v's color samples in all images. Let S_v represents voxel v's state, i.e., its occupancy and color, $S_v = (v\text{'s occupancy}, v\text{'s color})$, so the problem becomes how to estimate $p(S_v|C)$, which is usually difficult because many of these colors are just outliers. We approach this problem based on the following observation. As shown in Figure 5–1, voxels can be classified into three categories:

1. Empty voxels (such as the gray one): For these voxels, what cameras see are the corresponding points on objects or the background that intersect the rays passing through them and the camera centers. The colors in different cameras correspond to different surface parts; thus the color distribution is usually quite flat.
2. Surface voxels (such as the black one): For those cameras that can see them, the colors are the true surface colors, while for other cameras, the colors are those of other (occluding) objects. Since several cameras agree on the correct color, the color distribution usually has a peak.

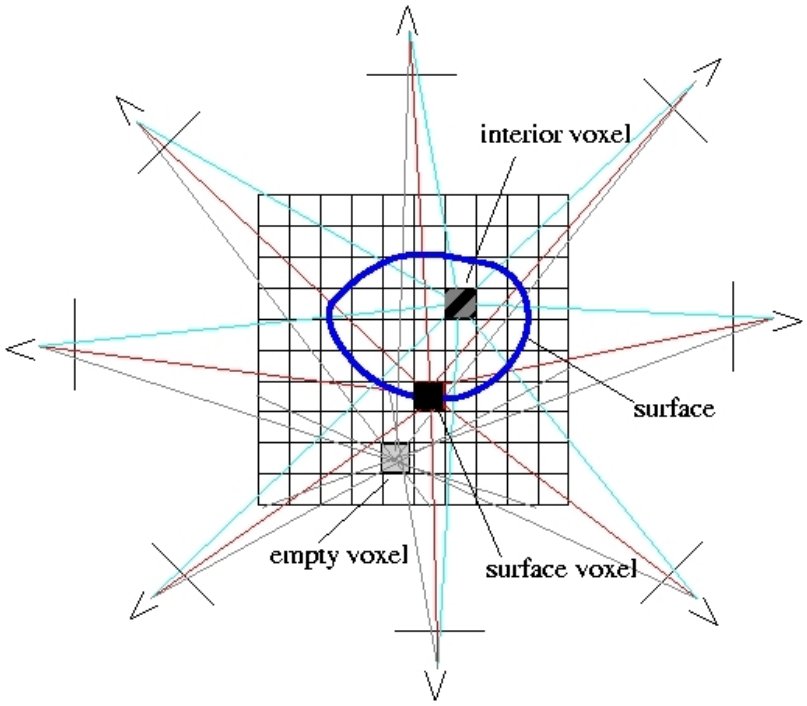


Figure 5–1: Voxel categories.

3. Interior voxels (such as the black-gray one): No camera can see them, so the colors can be arbitrary surface colors. Its color distribution is similar to the empty voxel case.

5.2 A New Photo Consistency Test

From Bayes' rule, we know

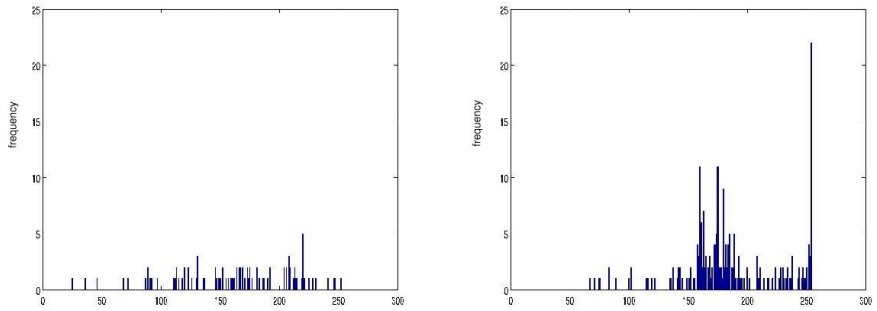
$$p(S_v|C) \propto p(C|S_v)p(S_v) \tag{5.1}$$

where, C is the set of color samples of the voxel v, and S_v represents v's state, as defined in the last section. Since we do not know the prior information $p(S_v)$, we can only obtain a maximum likelihood estimate from $p(C|S_v)$, i.e.,

$$S_v = \text{argmax}_{S_v} p(C|S_v) \tag{5.2}$$

This equation can be used to determine the color that exhibits the highest level of photo-consistency among the color samples. While it is possible to use some expensive optimal estimation techniques to find a solution (e.g., Fitzgibbon *et al.* [53]), we try a simpler idea. Based on above observations from our voxel classification, we can simply test each candidate voxel by determining the count of the most frequently appearing color among all potentially associated pixels. If this is beyond a certain threshold, we may accept the candidate as a surface voxel (see Figure 5–2). For an empty voxel, it is unlikely that the threshold will be reached, and thus, it will not be accepted as a surface voxel. However, in the case that a large percentage of the foreground and background surface points occluding the candidate exhibit a similar color, we have no means of distinguishing this from a true surface voxel, and so, the candidate will be accepted as a false positive. The case of an interior voxel is similar, but we may post-process it based on its visibility after reconstruction. Keeping in mind that no camera should be able to see an interior voxel, there must be a surface voxel along the ray connecting an interior voxel to each camera center.

In practice, two colors are regarded as the same if they are close enough (by some distance metric, e.g., the Euclidean distance in color space). This is most reliably



(a) An interior voxel. (b) A surface voxel.

Figure 5–2: Color histogram of different voxels. A surface voxel usually has a color whose frequency is much higher than others.

handled through the use of some clustering algorithm (e.g., ISODATA) [73] and choosing the center of the largest cluster as the estimated color. If the population of that cluster is beyond a certain threshold, the corresponding voxel is accepted as a surface voxel. Unfortunately, clustering algorithms are usually expensive; if the scene is decomposed into millions of voxels, the process may be unacceptably slow. A simple alternative is to calculate the color histogram and see which color appears most frequently. Although, based on camera geometry, there are usually tens to hundreds of pixels corresponding to each voxel, these are sparsely scattered in the 256x256x256 color space, and therefore we use a hash table to calculate the histogram. Suppose we use an N-element array $T[N]$ to implement the hash table (N is a constant, depending on the voxel size and number of cameras. In our experiments, a value of several hundred is usually sufficient). Each cell of the array stores the accumulated count of a particular color. The hash index is determined by the RGB color value of the corresponding pixel, that is,

$$\text{Index} = h(r, g, b) = (r + g + b)\%P \tag{5.3}$$

where P is a prime. In order to avoid collisions, we set P dynamically for each voxel to be the largest prime less than the number of corresponding pixels.* Collisions may still occur, in which case, there are numerous collision resolution techniques that may be used. At present, we simply reserve a number of additional cells in the array to deal with collisions. Each voxel is projected to all cameras and the corresponding pixel color values are collected. Each such value is then mapped to one cell of the hash table by Eq. 5.3 and the corresponding cell contents are incremented by one. Since data can be noisy, indices of the same color can be different in practice. In order to resolve this problem, neighbouring colors (in color space) are considered to be equivalent. In our experiments, this includes colors that differ by no more than a value of 3 along each axis of the RGB color space. This is accomplished by a 3D convolution applied to the elements of the hash table to accumulate the counts of neighbouring colors. The most frequently occurring color is then chosen. If its frequency is beyond the threshold, the corresponding candidate is accepted as a surface voxel.

A direct benefit of this algorithm is the speed improvement it provides. Since we no longer need to worry about the visibility problem between voxels, the voxel test operation can be performed completely in parallel.

---

* N must be larger than P to ensure that the array bounds are not exceeded.

5.3 Experimental Results

We conducted both simulation and real experiments using the method discussed above. For simulation experiments, objects are synthesized using OpenGL and viewed by twenty four surrounding cameras. Some results are shown in Figures 5–3 to 5–6. While reasonable models are reconstructed in general, errors may arise under certain circumstances. For example, in Figure 5–5, there are some large textureless areas that result in the body being fattened, the skin surface (head and hands) not fully reconstructed, and some parts (collar and tie) wrongly colored. This problem arises because the large number of similarly colored pixels from surrounding areas may generate a count for an incorrect surface color exceeding that of the true color.† However, it should be noted that similar problems arise with many other algorithms in textureless regions. The results from real scene data are worse than simulation experiments as Figure 5–7, 5–8 shows. One reason is probably that real scenes do not fit Lambertian assumption as well as synthetic images do.

In practice, we find that the quality of the model, and hence, a sample reconstruction, depends on the voxel size, object color distribution, and threshold selection. Some errors can be corrected by a post-processing step, for example, by median filtering to remove isolated errors (e.g., those noise points in Figure 6–13.), and by reprojecting the reconstructed model to the cameras and comparing these

† An example of this effect can be seen in the incorrectly colored gray pixels of the white coat collar in Figure 5–5. Since the cameras viewing the character from behind all see the person's gray back, this color overwhelms the count of white pixels.

with the original images. Specularity can also be a source of trouble, as illustrated in Figure 5–3. In this case, techniques such as those of Yang *et al.* [164] and Bonfort and Sturm [19] can be used to improve the results.

5.4 Conclusion

We have described a novel approach to testing photo consistency for voxel coloring. It avoids the visibility and photo consistency coupling problem and enables parallel computation, thereby providing reasonable results quite efficiently. Since it is a purely local operator, no doubt it is hard to perfect results. For applications requiring more accurate geometrical models, the results provided by the proposed method can be used as an initial model. As one example, this could be followed by some surface fitting technique, which may employ constraints such as surface smoothness, thereby producing a better surface model.

We currently use RGB colors and assume Lambertian surfaces. It might be preferable to weaken the Lambertian assumption by considering only chromaticities for photo consistency. In order to improve the current method, the problem can be analyzed on a more sound theoretical basis. The color distribution of each voxel arises from a combination of the true surface color distribution, other surface color distribution, and a background color distribution. Thus, we might consider a statistical model, for example, a mixture of Gaussians, to formulate the problem. Voxel occupancy could then be decided by testing the hypothesis that the color distribution is a mixture of the true surface distribution (which usually has a peak), other surface color distribution and background surface color distribution (which are typically much flatter). Our current method uses a maximum likelihood estimation. We did

not consider the prior term $p(S_v)$ (see Equation 5.1) because this information was not available. However, if some prior knowledge about the foreground voxel color distribution were available, it would likely be helpful to incorporate it.

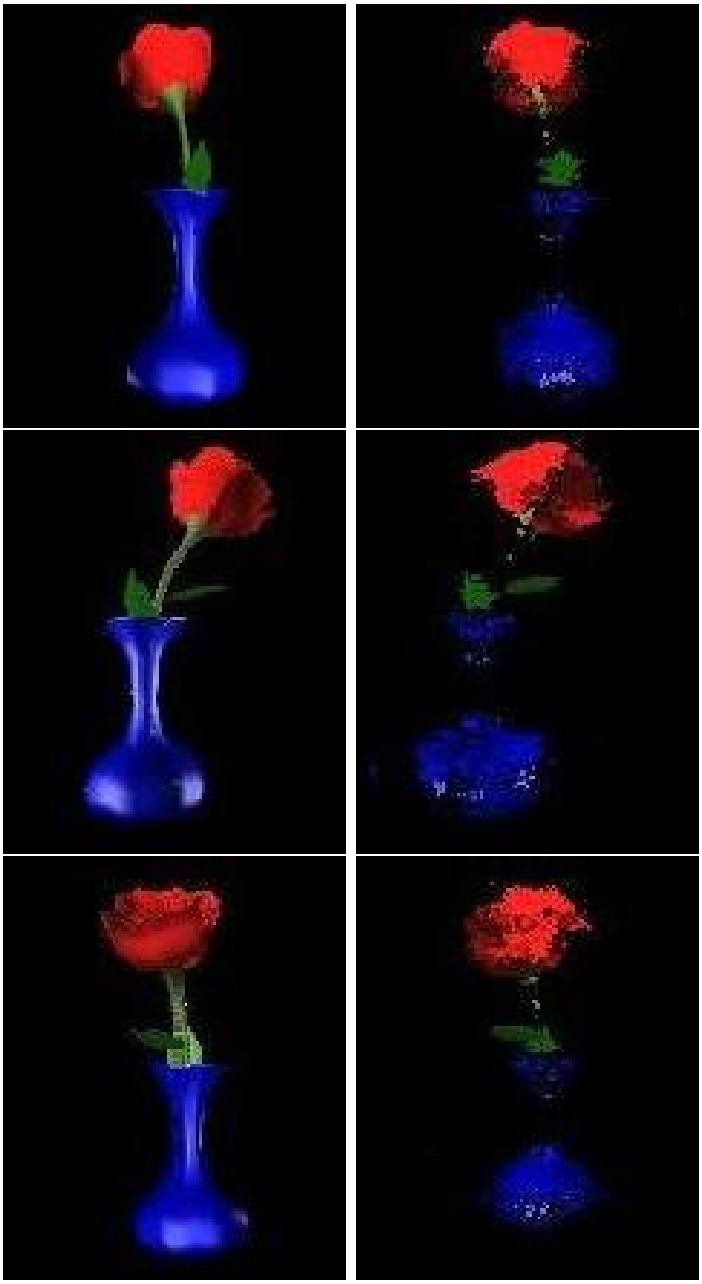


Figure 5–3: Rose reconstruction: volumetric reconstruction based on most frequent color estimation. The first column shows original images, the second column is reconstructed. Twenty four cameras are used, of which three are shown here. The reconstructed models are rendered to the same camera positions for comparison.

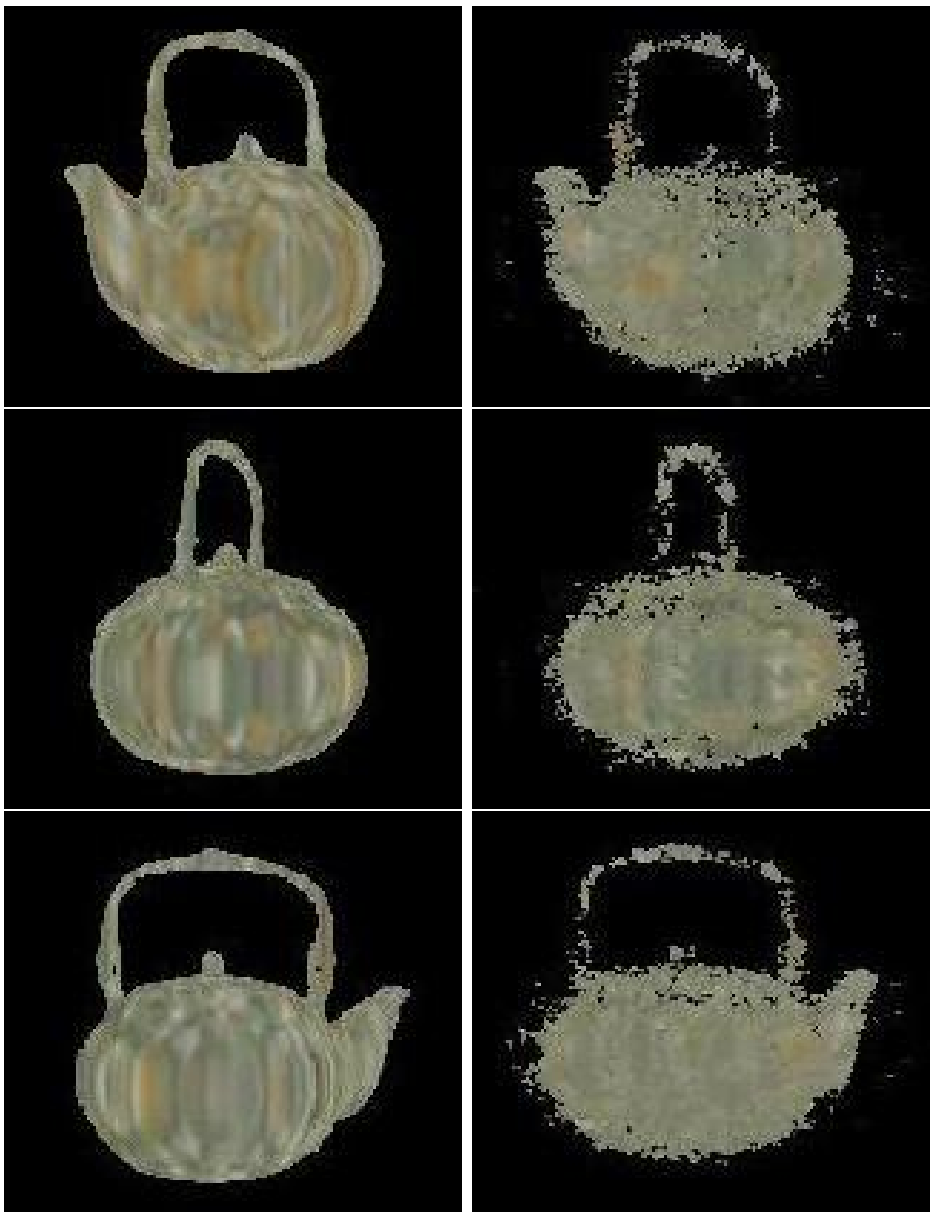


Figure 5–4: Teapot reconstruction, the same configuration as in Figure 5–3 is used.

Figure 5–5: Person reconstruction, the same configuration as in Figure 5–3 is used.

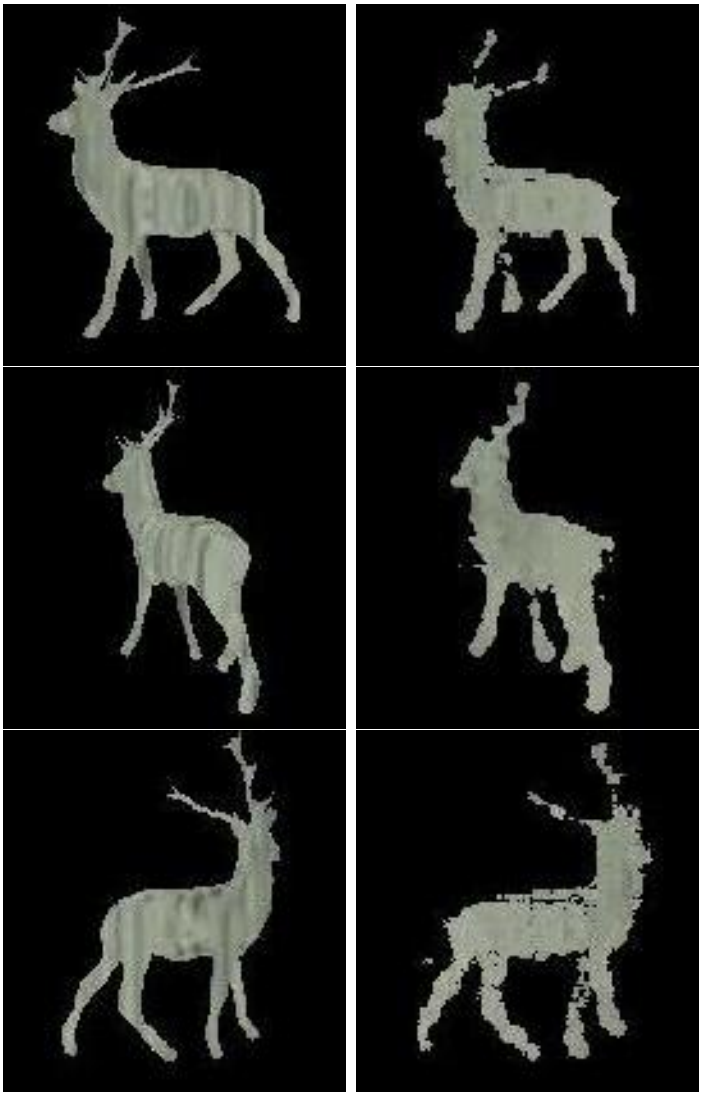


Figure 5–6: Deer reconstruction, the same configuration as in Figure 5–3 is used.

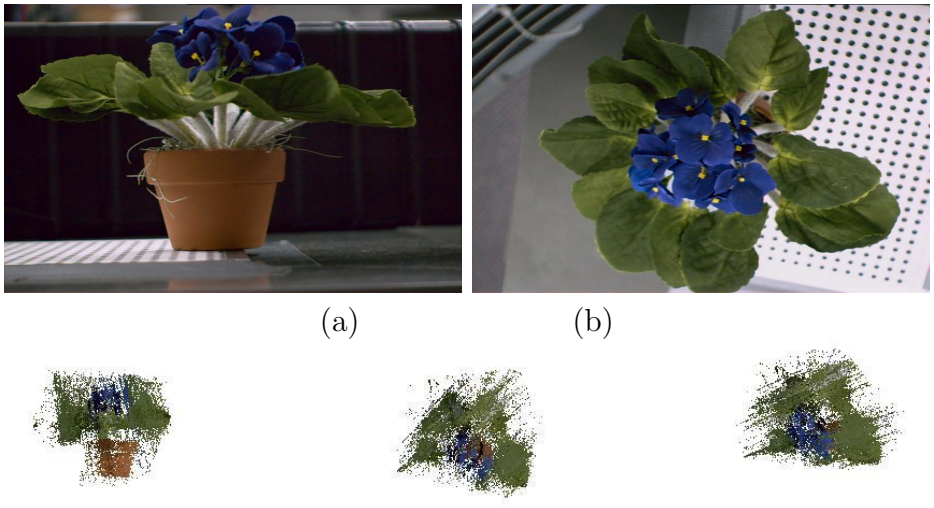


Figure 5–7: Violet: (a)(b) two of 40 original images, (b)-(d) reconstruction. Data set courtesy of Kyros Kutulakos [88].

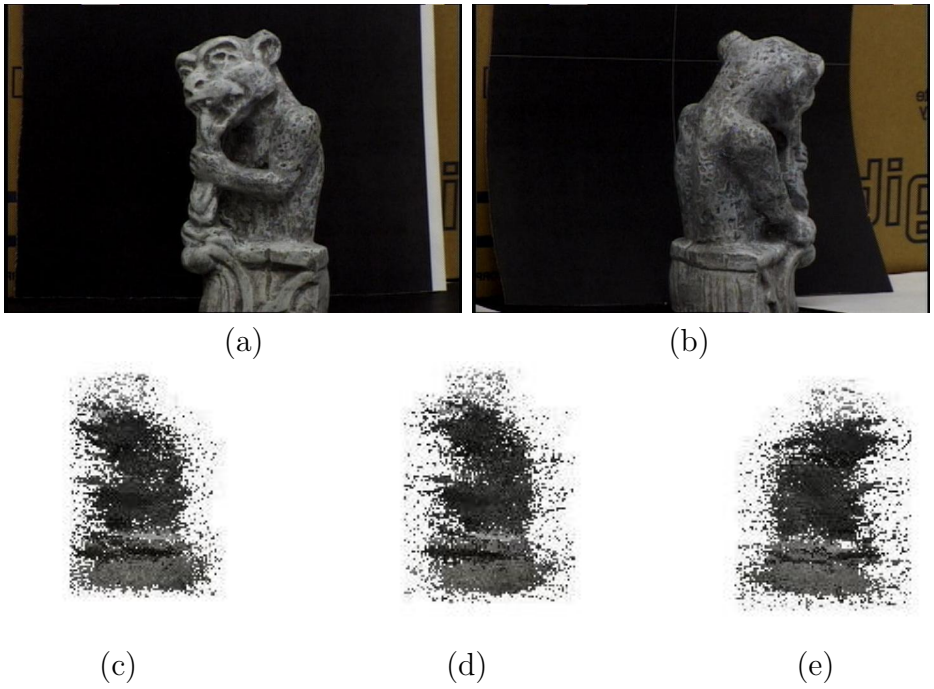


Figure 5–8: Gargoyle: (a)(b) two of 16 original images, (c)-(e) reconstruction. Data set courtesy of Kyros Kutulakos [88].

CHAPTER 6
Occupancy as Classification

In the previous chapter, voxel occupancy and coloring are solved together in one step. However, as we are more interested in what determines the occupancy of a voxel, we will not address the coloring problem here. Once the model is constructed, coloring can be done by reprojecting voxels into reference images and extracting colors from them. In some applications, it might be even desirable to use colors other than from reference images. Based on the analysis in the previous chapter, we can view voxel occupancy determination as a classification problem. That is, the problem is how to classify voxels into two categories: occupied (surface voxel) or not (empty and interior voxel). Previously, we used the maximum frequency of the color histogram of a voxel's projections as the classification feature, but the experimental results were somewhat disappointing. In this chapter, we investigate another feature and a tensor voting technique to improve reconstruction by exploiting neighbouring information.

6.1 Camera Agreement Matrix and its Frobenius Norm

For a space point P, supposing its projection to camera C_i is p_i, and N_{p_i} is the neighbourhood of p_i in the reference image I_i, we need a representation that can capture the information from all reference images and imply the possibility whether P is on the object surface. Here we propose a camera agreement matrix to serve such a purpose. It is defined as follows:

$$\mathbf{M} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \end{pmatrix}$$

where a_{ij} measures the agreement between camera i and j about a voxel. It is a function of the voxel projections in camera i and j, i.e., $a_{ij} = f(I_i(N_{p_i}), I_j(N_{p_j}))$. For example, it can be the NCC (Normalized Cross Correlation) of $I_i(N_{p_i})$ and $I_j(N_{p_j})$, or some other metric.

In this thesis, we simply use a $0-1$ representation. If two cameras agree with each other (i.e., if the projected color difference is very small), we set a_{ij} to one, otherwise, zero. Note, the matrix is symmetric and a_{ii} is always one. The camera agreement matrix includes voxel information from all cameras. Features used for deciding voxel occupancy should be extracted from this matrix. In the following, we use a simple feature, the Frobenius norm, to represent the overall agreement on a voxel. That is,

$$F_M = \sqrt{\sum_{i,j} |a_{ij}|^2} \tag{6.1}$$

With a $0-1$ representation, this simplifies to the sum of all ones, i.e., the number of camera pairs that agree on the voxel's color.

6.2 Occupancy Classification Algorithm

Each voxel has such a F-norm, and voxel classification, based on these F-norms, is performed as follows:

1. For each voxel:
   project it into all cameras,
   compute the camera agreement matrix,
   calculate the F-norm.
2. Build a histogram of the F-norms of all voxels.
3. Find a threshold based on the histogram.
4. For any voxel whose F-norm is greater than the threshold, mark it as occupied.

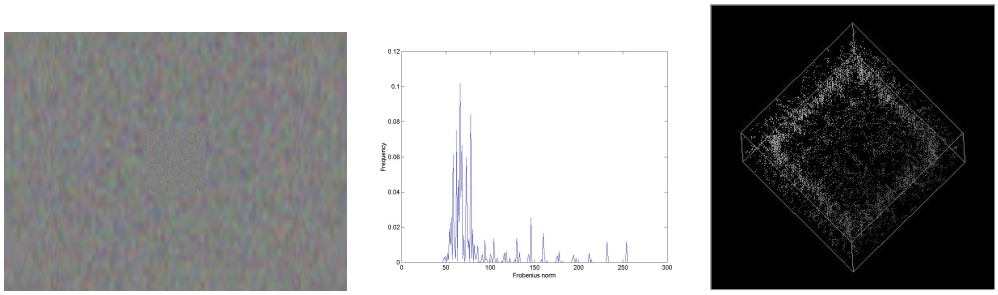
Various classification algorithms [73, 45] can be used to perform the binary classification task (i.e., classify voxels as either occupied or unoccupied). In this thesis, classification is accomplished by thresholding. Here, we choose a threshold automatically based on the histogram of F-norms by a method similar to that of Gonzalez and Woods [63] for image segmentation. First, we choose an initial threshold T, for example, as the average value of all F-norms. Next, we classify the F-norms into two sets, based on this threshold T, and compute the averages F_1 and F_2 of these two sets, respectively. A new threshold is then set to $T = (F_1 + F_2)/2$. This procedure is repeated until the change in T between successive iterations is sufficiently small.

Some synthetic experimental results are shown in Figure 6–1. The scenes are rendered using OpenGL with 24 cameras surrounding the objects. Each synthetic image has size of 320x240. The backgrounds are texture-mapped with an image in which the color of each pixel is chosen randomly. Here, a 256x256 texture image is used. For each pixel of the image, its rgb values are assigned by three independently generated random numbers in the range $[0, 255]$. For the OpenGL texture

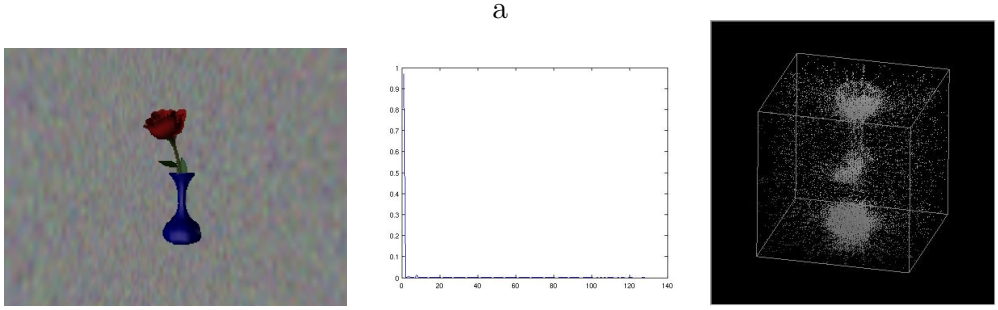
environment, we applied mipmapping, the nearest mipmap and bilinear interpolation is used for the texture min filter (i.e., set GL_LINEAR_MIPMAP_NEAREST for GL_TEXTURE_MIN_FILTER), and bilinear interpolation for the texture mag filters. That is, the closest level of mipmaps to the surface is chosen for the mapping, and a bilinear filter is used when applying the texture. Further details of the OpenGL texture mapping process can be found in the OpenGL programming guide [17]. In Figure 6–1a, a synthesized box (without top and bottom face) is used as the foreground object; the background is a much larger box. Cameras are placed inside the background box facing the foreground box. The distances from the camera to the foreground and the background are approximately 34 and 105 units respectively. The vertical view angle of the camera is 60 degrees. Each face of the background and foreground boxes is texture-mapped by the same random color texture as mentioned above. The foreground box face size is 32x32 units, and the background box face size is 110x200 units. The reference image size is 320x240. Thus, in Figure 6–1a, a background pixel spans approximately 1.5x1.5 to 2x2 pixels in the original texture image, and a foreground pixel approximately 4x4 pixels. Pixel spans are calculated based on the respective areas of foreground or background objects in the image, in relation to the texture size. Since these use the same random color texture, color alone does not provide sufficient information to distinguish foreground from background. In Figures 6–1(b-d), the scene consists of various scanned objects rendered by OpenGL. Among them, Figures 6–1b-c are not texture mapped. Figures 6–1d is mapped with a 1024x2506 texture, the same texture environment setting as Figure 6–1a is used, so a foreground pixel spans approximately 8x14 pixels in the original texture. Here,

since cameras and objects are well-controlled in the synthetic scenes, we only project the center of a voxel into cameras i and j. Supposing the projected pixels are p_i and p_j respectively, we set a_{ij} to zero if colors of p_i and p_j are different. Otherwise, we set it to one.

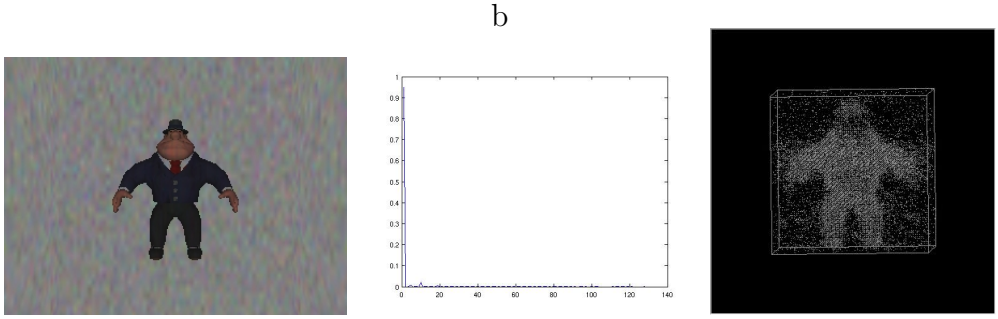
Although the initial results based on the camera agreement matrices provide a reasonable outline of basic object shapes, these are still unacceptably noisy, suggesting that our choice of feature is not sufficiently accurate to perform voxel classification. A possible reason for this problem might be that visibility information is not addressed explicitly by the camera agreement matrix. As Figure 6–2 shows, cameras C1 and C2 cannot see voxel $V1$, but the two voxels they observe in that position coincidentally have the same colors, leading to an undesirable contribution to the camera agreement matrix, and ultimately, an incorrect result. This kind of error might be solved once a model is built because we can update visibility based on the model. This is an issue we hope to address in our future research. Another possible error is that for an empty voxel, cameras seeing through it may accidentally obtain the same colors from other voxels. As Figure 6–2 shows, cameras C3 and C4 see the same color for voxel $V2$, and thus, their contribution to the camera agreement matrix incorrectly suggest that $V2$ is a surface voxel. This case is similar to the previous one, but cannot be corrected by visibility information. Provided these errors do not appear continuously in a limited area, we can consider them as noise, and correct using neighbourhood information, as discussed in the next section.



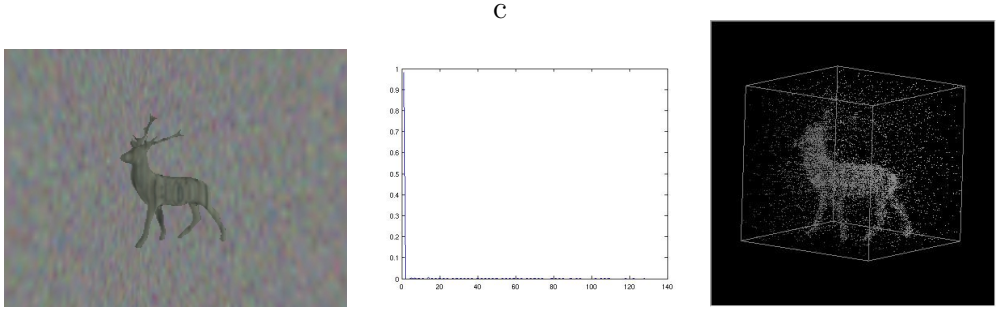
a



b



c



d

Figure 6–1: Reconstruction based on F-norms. 1st column: one of 24 reference images. 2nd column: histogram of F-norms. 3rd column: initial occupancy. Here, the background are planar screens texture-mapped with a random-color image. In a, the foreground is a synthesized box texture-mapped with the same image as the background, while in b-d, the foregrounds are scanned objects rendered by OpenGL.

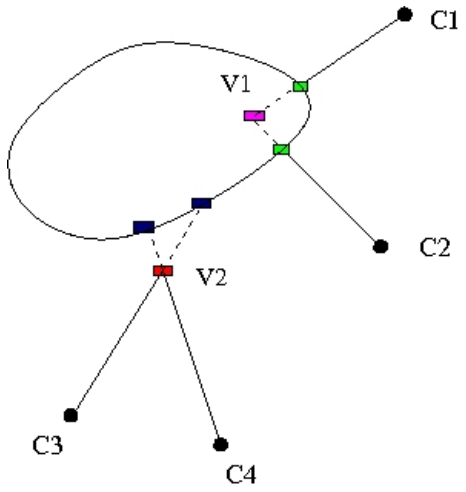


Figure 6–2: Two cases in which cameras contribute undesirably to the camera agreement matrices.

6.3 Refine Reconstruction by Tensor Voting

In stereo matching problems, as the initial estimate of disparities often contains many errors, neighbouring information can be considered to improve the results [100, 172, 137]. Similarly, for our volumetric reconstruction method, the initial result from the camera agreement matrices can be improved based on neighbouring information. The rationale is that a voxel is very likely on a surface if its neighbouring voxels are on the same surface. Unlike cooperative algorithms [100, 172], which iteratively accumulate neighbourhood information, tensor voting [110, 105, 95, 96, 153] provides a framework to propagate information into neighbours from various features such as points, curves, and surface patches in a unified way. Its use here is hypothesized to improve the initial results from the method as obtained in the previous section. We now provide a brief description of tensor voting, then explain how it is applied in our

experiments. For more details about tensor voting, readers can consult [110, 105, 108, 95, 96, 153, 107, 109, 155].

6.4 Tensor Voting Framework

6.4.1 Tensor Voting in 2D

Second order representation A 2D, symmetric, non-negative definite, second order tensor can be viewed as a 2x2 matrix, or equivalently an ellipse. The tensor can be decomposed as in Equation 6.2 ([110, 105]), where, e_1 and e_2 are the two eigenvectors of the 2x2 matrix, and λ_1 and λ_2 are their eigenvalues. Figure 6–3 shows how a generic tensor is decomposed into stick and ball components. The former represents a curve token with e_1, the eigenvector corresponding to the largest eigenvalue, as its normal direction. It is oriented, and corresponds to a degenerate elongated ellipsoid. The ball component represents a circular disk, as indicated by the second term in Equation 6.2. It corresponds to a structure without orientation preference, i.e., is unoriented. Figure 6–4 shows how to encode oriented and unoriented 2D inputs. $\lambda_1 - \lambda_2$ indicates the saliency of the oriented curvel, which is a small line segment with a normal (Figure 6–4).

$$T = \lambda_1 e_1 e_1^T + \lambda_2 e_2 e_s^T = (\lambda_1 - \lambda_2) e_1 e_1^T + \lambda_2 (e_1 e_1^T + e_2 e_2^T) \tag{6.2}$$

Second order voting Given the tensor at a point O, we want to know how its information is propagated to its neighbours. Consider a pure stick tensor at the origin, with its normal pointing in the $+y$ direction. We need to compute what information it propagates at P if it belongs to the same smooth perceptual structure, as shown in Figure 6–5. Assuming an *osculating circle* is defined by O, its normal,

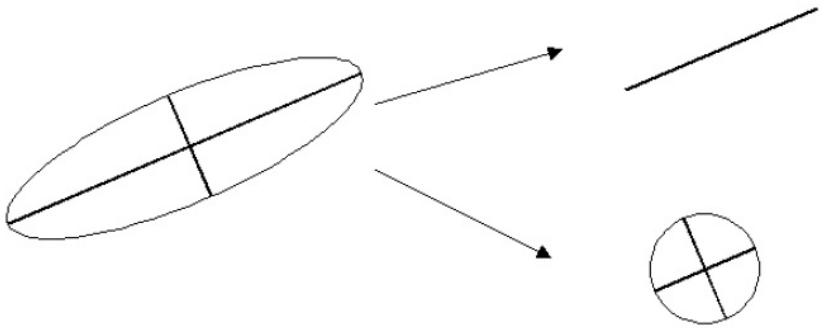


Figure 6–3: A second order generic tensor and its decomposition in 2D. Reprinted from Mordonhai [110] with permission.

Input	Second order tensor	Eigenvalues	Quadratic form
 oriented		$\lambda_1 = 1, \lambda_2 = 0$	$\begin{bmatrix} n_1^2 & n_1 n_2 \\ n_1 n_2 & n_2^2 \end{bmatrix}$
 un-oriented		$\lambda_1 = \lambda_2 = 1$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Figure 6–4: Encoding oriented and unoriented 2D inputs as 2D second order symmetric tensors. Reprinted from Mordonhai [110] with permission.

and P, the propagated vote at P should also be a stick tensor with its normal along the radius of the osculating circle. Its magnitude should be a function of the arc length (s) of OP, the curvature κ, and the scale σ of voting, which determines how far O can influence, i.e., the effective neighbourhood size. A *saliency decay function* [95, 153, 96, 110, 105] is shown in Equation 6.3, where c is a constant to control the degree of decay. A good choice for c is given by $c = \frac{-16log(0.1)\times(\sigma-1)}{\pi^2}$ [65, 110, 105]. A vote field of a unit stick voter at O with its normal along the y-axis can be defined as a function of the distance l between the voter O and the receiver P, and the angle θ (see Figure 6–5), as described in Equation 6.4 [110, 105].

$$DF(s,\kappa,\sigma) = e^{-(\frac{s^2+c\kappa^2}{\sigma^2})} \tag{6.3}$$

$$S_{SO}(l,\theta) = DF(s,\kappa,\sigma) \begin{bmatrix} -sin(2\theta) \\ cos(2\theta) \end{bmatrix} [-sin(2\theta) \qquad cos(2\theta)] \tag{6.4}$$

We constrain voting only to receivers with θ less than 45°. Such a vote field is shown in the upper part of Figure 6–6a. If the voter's normal is not along the y-axis, we can generate the vote fields by a rotation, as indicated in Figure 6–6b.

The ball vote field can be generated from the stick vote fields by integrating over all possible directions. At receiver P, the ball vote tensor is defined as follows [110, 105].

$$B(P) = \int_0^{2\pi} R_\theta^{-1} S(R_\theta P) R_\theta^{-T} d\theta \tag{6.5}$$

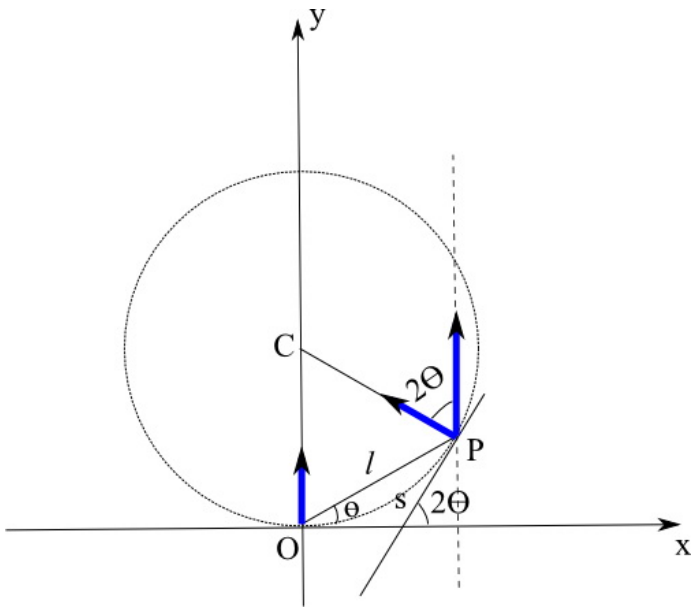


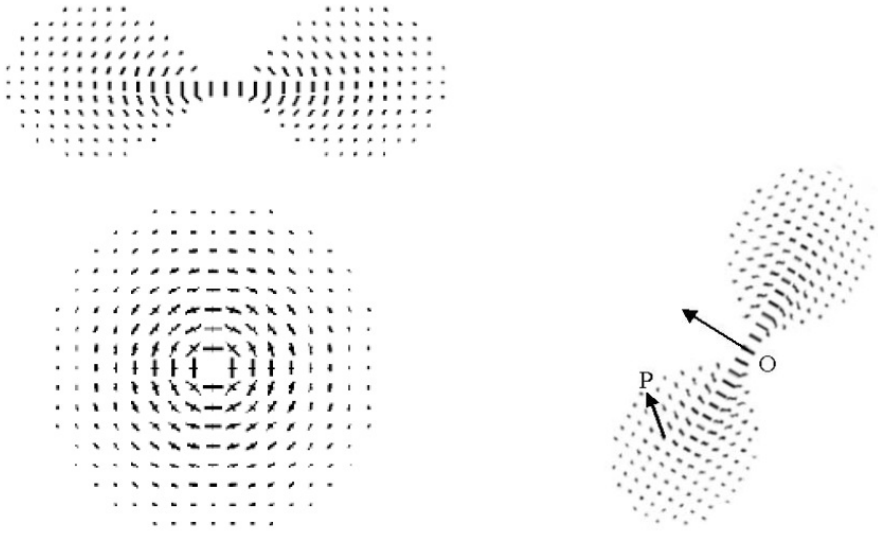
Figure 6–5: Second order vote cast by a stick tensor located at the origin. Note the orientation at the receiver P is a rotation of 2θ degrees of the voter O's orientation.

where R_θ is a rotation matrix. A ball vote field is shown in the lower part of Figure 6–6a.

Once we know how to propagate tensor information, the cast votes at every location are accumulated. Decomposing these tensors leads to the creation of saliency maps, from which structures can be extracted.

6.4.2 Tensor Voting in 3D

Second order representation Similar to the 2D cases, a 3D, second order, symmetric, non-negative definite tensor is equivalent to a 3x3 matrix and a 3D ellipsoid. The eigenvectors of the tensor are the axes of the ellipsoid. A 3D tensor can be decomposed as in Equation 6.6 [110, 105]. Figure 6–7 shows how a generic



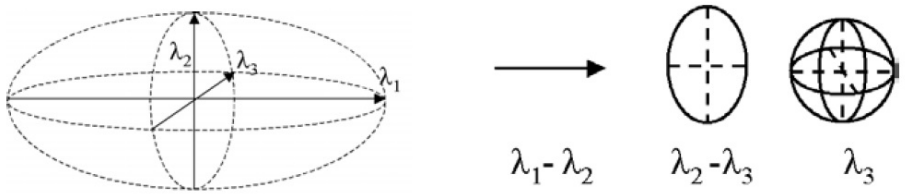
(a) The 2-D stick and ball fields (b) Stick vote cast from O to P

Figure 6–6: Voting fields in 2D. Reprinted from Mordonhai [110] with permission.

tensor is decomposed into stick and ball components. Figure 6–8 shows how to encode oriented and unoriented 3D inputs.

$$\begin{aligned} T &= \lambda_1 e_1 e_1^T + \lambda_2 e_2 e_2^T + \lambda_3 e_3 e_3^T \\ &= (\lambda_1 - \lambda_2) e_1 e_1^T + (\lambda_2 - \lambda_3)(e_1 e_1^T + e_2 e_2^T) + \lambda_3 (e_1 e_1^T + e_2 e_2^T + e_3 e_3^T) \end{aligned} \tag{6.6}$$

Second order voting For a pure stick tensor, since the voter, the receiver, and the stick tensor at the voter define a plane, its voting mechanism can be defined exactly the same as in 2D. Similar to the 2D case, a ball vote field is an integration of stick vote fields at all orientations. A plate tensor evaluates orientational uncertainty



(a) A 3-D generic tensor (λ_i are its eigenvalues in descending order)

(b) Decomposition into the *stick, plate and ball* components

Figure 6–7: A second order generic tensor and its decomposition in 3D. Reprinted from Mordonhai [110] with permission.

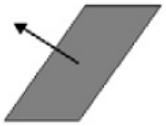
around one axis, so it is estimated by integrating stick tensors spanning a unit circle around that axis. Plate tensors are not used in our experiments.

6.5 Applying Tensor Voting

Here, we use $3D$ tensors with a ball component and a stick component. The ball component is determined by a surface voxel position, and the stick component is determined by the surface estimated based on the distribution of the voxel's neighbours. A surface voxel is discarded if there is an insufficient number of neighbouring surface voxels. Otherwise, we fit a surface to these voxels, and calculate its normal to construct the stick tensor component.

6.6 Experiments

Experiments with simulated and real data were conducted. These used a 5x5x5 neighbourhood for tensor voting, with a planar surface to fit the voxels. Results of the former, making use of tensor voting software from Medioni et al. [106], are shown in Figures 6–9 to 6–12. In Figure 6–9, both the foreground object (a box) and the background are texture-mapped with the same random-color image, so it is almost impossible to distinguish them by color information alone. Instead, we must rely on

Input	Tensor	Eigenvalues	Quadratic form
 surfel		$\lambda_1 = 1,$ $\lambda_2 = \lambda_3 = 0$	$\begin{bmatrix} n_1^2 & n_1 n_2 & n_1 n_3 \\ n_1 n_2 & n_2^2 & n_2 n_3 \\ n_1 n_3 & n_2 n_3 & n_3^2 \end{bmatrix}$
 curvel		$\lambda_1 = \lambda_2 = 1,$ $\lambda_3 = 0$	**P** (see below)
 un-oriented		$\lambda_1 = \lambda_2 = \lambda_3 = 1$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

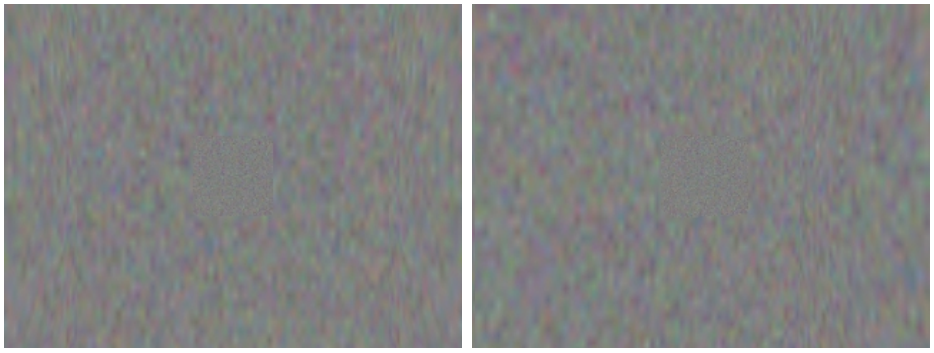
$$\mathbf{P} = \begin{bmatrix} n_{11}^2 + n_{21}^2 & n_{11}n_{12} + n_{21}n_{22} & n_{11}n_{13} + n_{21}n_{23} \\ n_{11}n_{12} + n_{21}n_{22} & n_{12}^2 + n_{22}^2 & n_{12}n_{13} + n_{22}n_{23} \\ n_{11}n_{13} + n_{21}n_{23} & n_{12}n_{13} + n_{22}n_{23} & n_{13}^2 + n_{23}^2 \end{bmatrix}$$

Figure 6–8: Encoding oriented and unoriented 3D inputs as 3D second order symmetric tensors. Reprinted from Mordonhai [110] with permission.

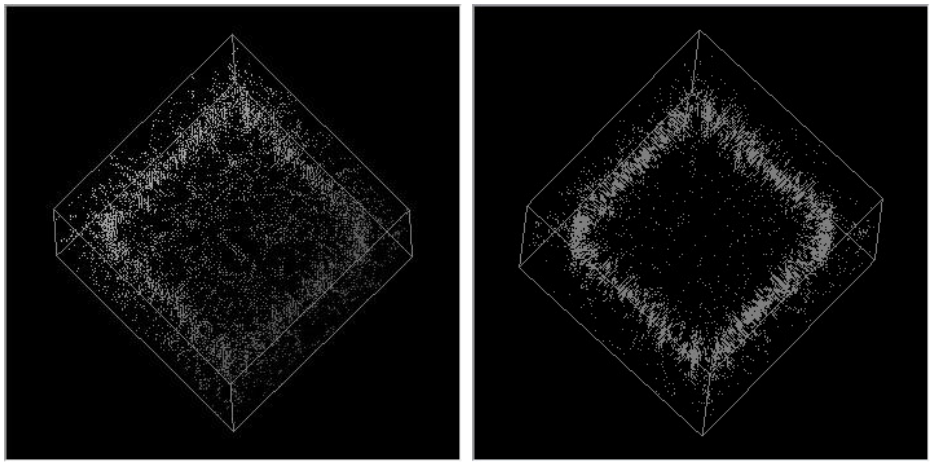
properties of geometry. The result indicates that our method can separate foreground from background and reconstruct its geometrical structure correctly. Figures 6–10 to 6–12 also demonstrate that this approach can successfully reconstruct objects against a cluttered background. Experimental results using real data are shown in Figures 6–13 to 6–16. Here, NCC with a 5x5 neighbourhood is used to compute a_{ij}. If the NCC exceeds the chosen threshold, $a_{ij} = 1$, otherwise, it is zero. The results obtained from real data are much less noisy than those from simulated data. This is likely due to the differences in color between foreground and background in real scenes, whereas we deliberately applied the same random color texture to both in the simulated case. However, we can see that some concave areas are poorly reconstructed, for example, the area between the arm and base in the gargoyle reconstruction (Figure 6–13). This may suggest that the $0-1$ representation in camera agreement matrices is over-simplified, leading to some loss of information. Future research should investigate the selection of improved features.

6.7 Discussion

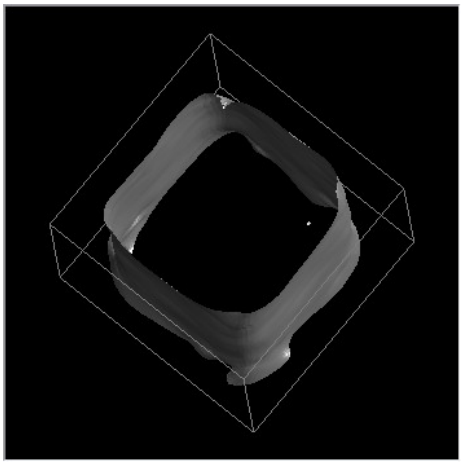
The 3D modeling approach described in this thesis overcomes three problems associated with space carving. First, if space carving removes a voxel incorrectly, it cannot be recovered in subsequent processing, and this error will propagate. Thus, space carving tends to carve voxels conservatively, i.e., uses a high threshold. In contrast, our method allows for the correction of a misclassified voxel through the tensor voting process.



2 of 24 original images

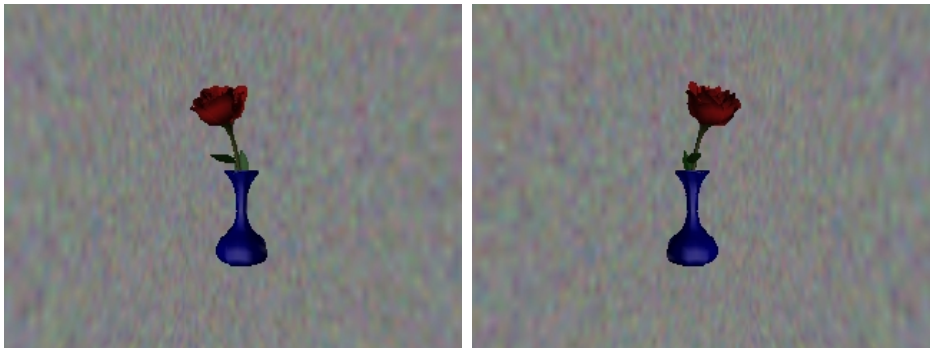


initial result after tensor voting

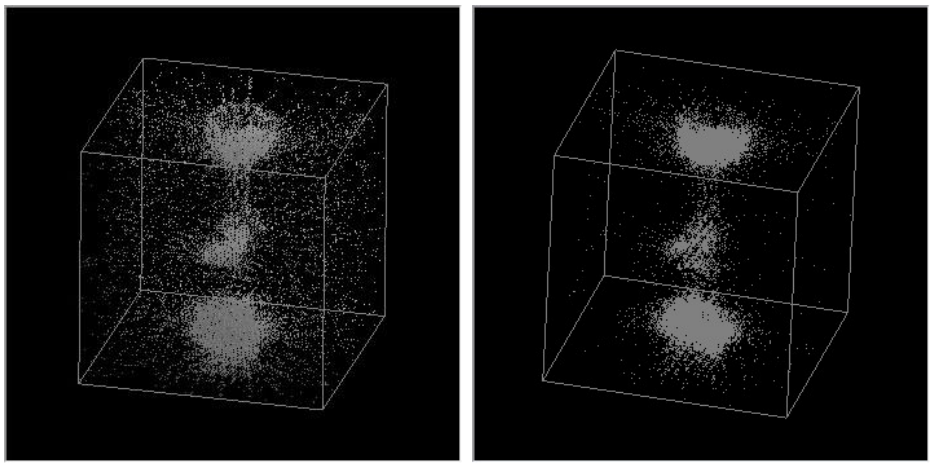


a surface extracted based on tensor saliency

Figure 6–9: Reconstruction results on simulated 'box' data using our F-norm method. 80x80x80 voxels are used.



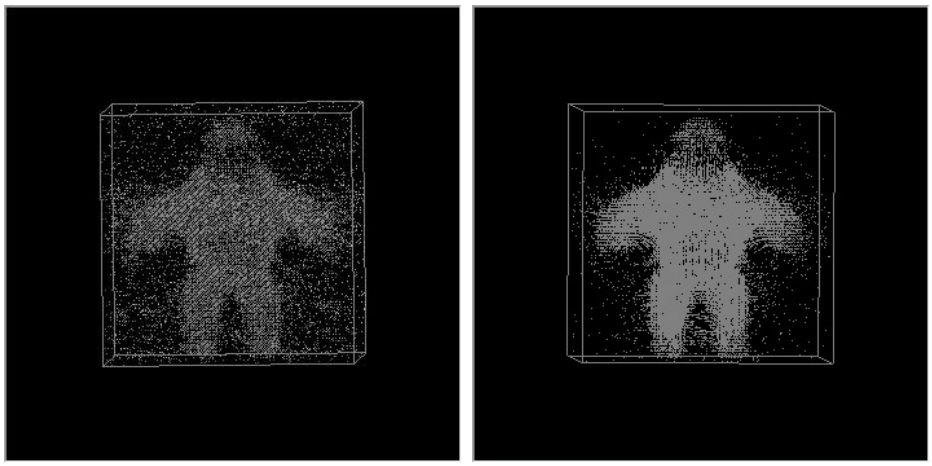
2 of 24 original images



initial result after tensor voting

Figure 6–10: Reconstruction results on simulated 'rose' data using our F-norm method. 80x80x80 voxels are used.

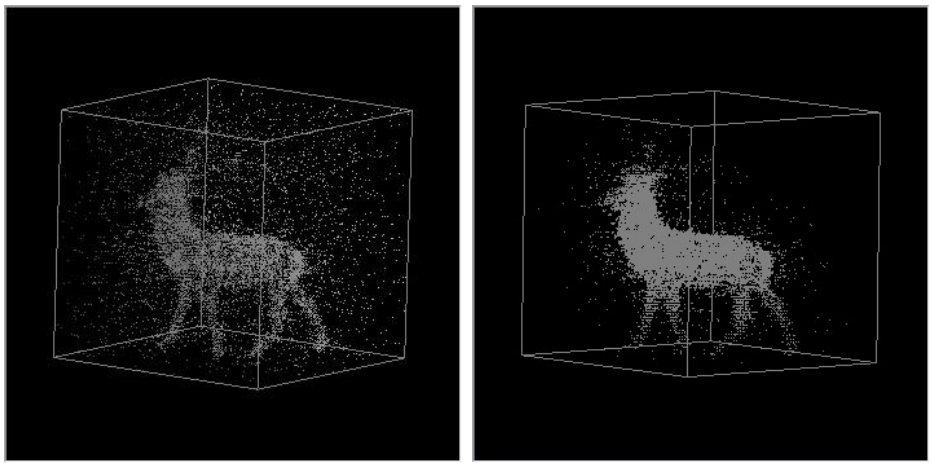
2 of 24 original images



initial result after tensor voting

Figure 6–11: Reconstruction results on simulated 'al' data using our F-norm method. 80x80x80 voxels are used.

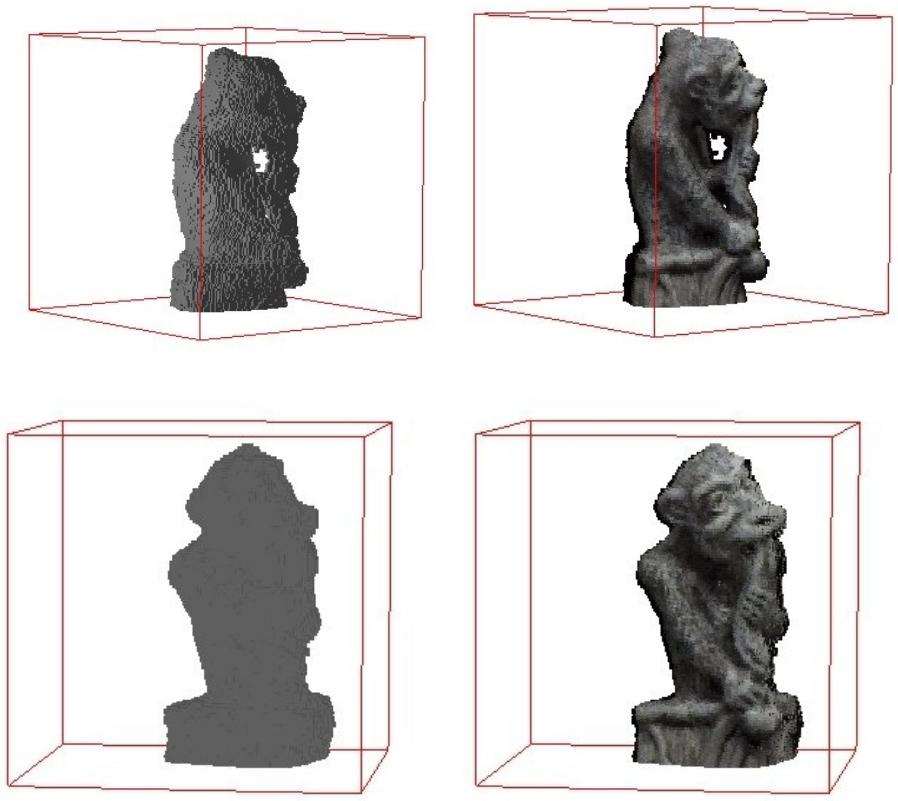
2 of 24 original images



initial result after tensor voting

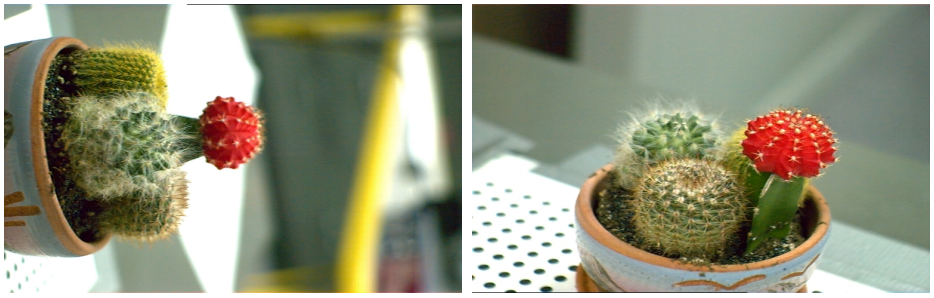
Figure 6–12: Reconstruction results on simulated 'deer' data using our F-norm method. 80x80x80 voxels are used.

2 of 16 original images

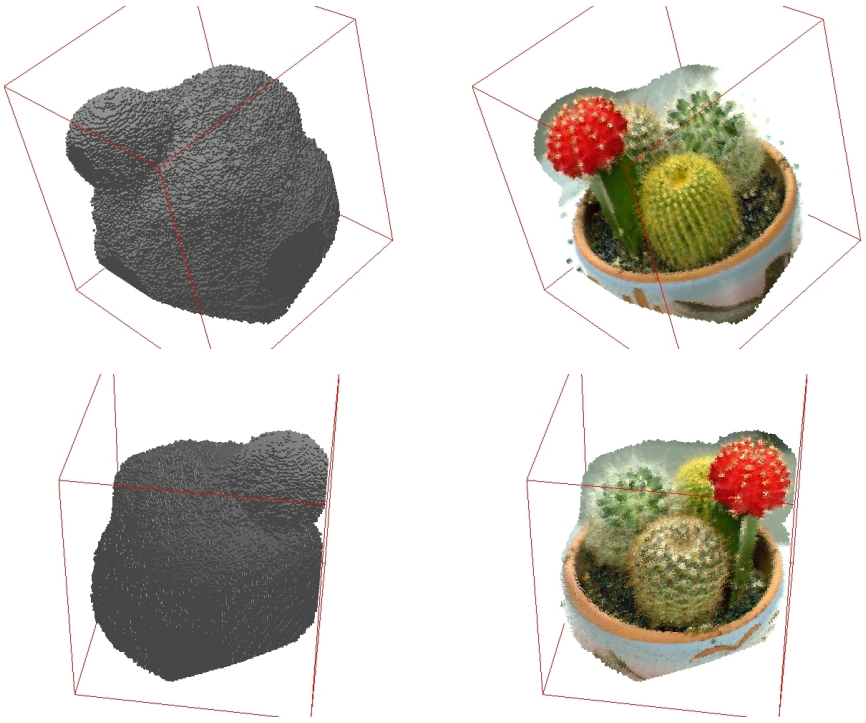


shaded model texture-mapped model

Figure 6–13: Reconstruction results on 'gargoyle' data using our F-norm method. 128x128x128 voxels are used. Note that while the shaded models appear to contain very little detail, the results are much more convincing after texture mapping. This observation reflects the difficulty of making objective comparisons of view reconstruction quality between different forms of output. Data set courtesy of Kyros Kutulakos [88].



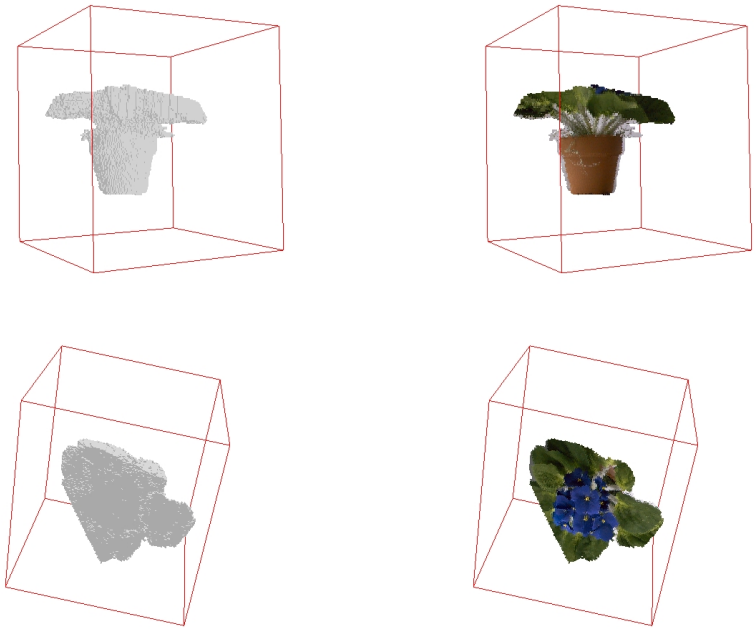
2 of 30 original images



shaded model texture-mapped model

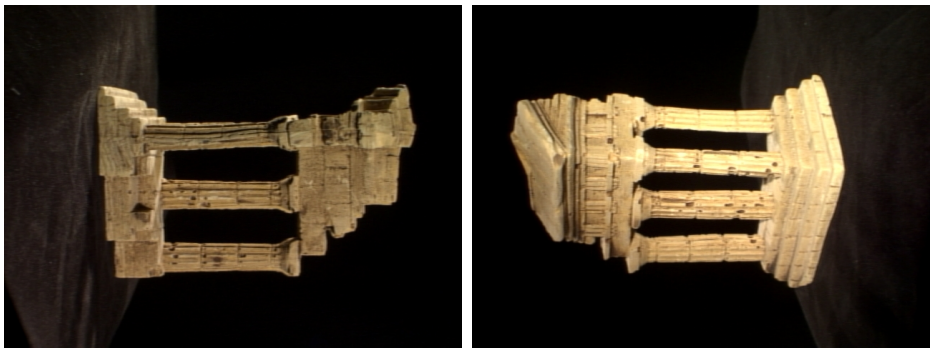
Figure 6–14: Reconstruction results on 'cactus' data using our F-norm method. 128x128x128 voxels are used. Data set courtesy of Kyros Kutulakos [88].

2 of 40 original images

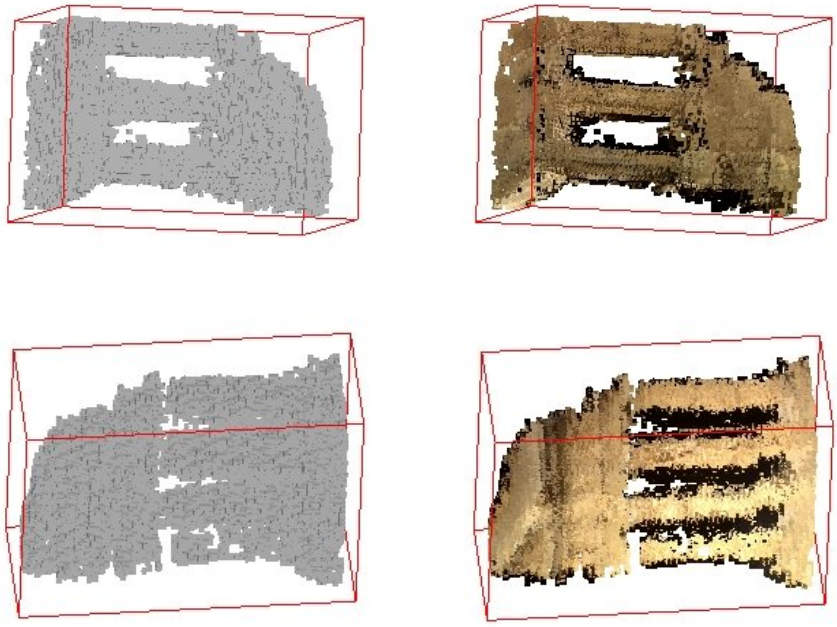


shaded model texture-mapped model

Figure 6–15: Reconstruction results on 'violet' data using our F-norm method. 128x128x128 voxels are used. Data set courtesy of Kyros Kutulakos [88].



2 of 16 original images



shaded model texture-mapped model

Figure 6–16: Reconstruction results on 'temple' data using our F-norm method. 41x65x31 voxels are used. Data set courtesy of Steven Seitz, *et al.* [139].

Second, the threshold for the photo-consistency test must be determined experimentally by space carving, whereas in our method, thresholding is postponed until all F-norms are built, and can be chosen automatically based on their histogram.

Third, space carving is an iterative process, in which a change in the occupancy of every voxel can affect the occupancy decision for other voxels. In our method, both the construction of the camera agreement matrix and the tensor voting step use only local information. This enables a parallel implementation, leading to potential computational advantages.

There are, however, some limitations with our method. As mentioned previously in the thesis, visibility information is not used explicitly in the camera agreement matrix, despite it providing a valuable source of additional information. This can introduce inaccuracies in the reconstruction. For example, a camera that does not see a voxel might nevertheless contribute additional weight to the camera agreement matrix for that voxel in the case where the camera happens to agree with others on the relevant measurement, e.g., color. Furthermore, the $0-1$ representation may be over-simplified. Prior to tensor voting, an initial estimate must be made of the surface voxels, where some information is likely to be lost. Other alternatives may be preferable, for example, saving the photo-consistent measurements (e.g., NCC) directly in the camera agreement matrix, and using eigenvectors as features instead of F-norms. Voxel occupancy could then be determined by classification techniques [73, 45] based on such features.

If visibility information were to be modeled, it appears unavoidable to follow the space carving approach, and process voxels in an iterative manner. This would

be computationally expensive and as such, undesirable from our perspective. On the other hand, visibility is, to a certain degree, considered implicitly. When two cameras observe the same color projection from a voxel, this is a strong indication that the voxel is visible to both cameras, and in turn, influences the value calculated in our camera agreement matrix. We believe that it should be possible to infer voxel occupancy from local information while not modeling visibility information, and investigate this hypothesis in the thesis. Our experiments using the F-norm method suggest that this might indeed be feasible.

6.7.1 Comparison with Generalized Voxel Coloring (GVC)

Space carving algorithms all work in a similar manner. Starting from a volume containing objects, they carve away voxels iteratively from the outermost regions until hitting object surfaces that are supposed to be photo-consistent. The process stops when no more voxels can be carved. The remaining voxels are taken as reconstructed objects. In some camera configurations, for example, when objects are in front of all cameras, the visibility order is known. This allows space carving to be performed using a more efficient plane sweep method, as in Seitz and Dyer's voxel coloring algorithm [141], only need to pass every voxel once. However, this method imposes certain constraints on camera configuration, in order to ensure that a visibility order can be established prior to processing.

It is possible to extend plane sweep to standard space carving. For example, Kutulakos and Seitz [90] suggest performing plane sweep from six directions and using cameras behind the sweep planes only. The problem with this is that only a subset of cameras (i.e., partial visibility information) are used to determine a voxel's

photo-consistency. It is therefore possible that a voxel may be carved incorrectly as a consequence of high color variation within the reduced set of cameras, even though it would be kept if more cameras are considered [90]. Generalized voxel coloring (GVC) [37, 145] is an efficient and representative space carving algorithm, which uses full visibility information. We use Loper's implementation [99] for which the source code is publicly available, for our experimental comparison.

When using simulation images, we know the polygonal geometry of the object, so true voxel occupancy is easy to determine. If a voxel intersects a polygon of the model, it is set as occupied, otherwise, it is empty. The ground truth is obtained like this. Once we reconstruct voxel occupancy, we can compare it to the ground truth. A match means that the occupancies (empty or occupied) of a voxel are the same in both the reconstructed model and the ground truth model. The match rate is the ratio between the number of matched voxels and the total number of voxels, that is,

$$\text{match rate} = \frac{\text{number of matched voxels}}{\text{total number of voxels}} \tag{6.7}$$

We use the match rate as a measure of the quality of reconstruction. While this is indicative of the overall reconstruction correctness for both surface and non-surface voxels, it is insufficient as a means of indicating the quality of surface estimation. In particular, if the majority of the volume space is empty, any method that recovers most of this empty space, such as an overly aggressive space carving method, would achieve a high match rate. In fact, even a blind method that labels every voxel as empty would often achieve a high score. We therefore define two additional metrics that focus exclusively on surface voxels. First, we define the $(2n-1) \times (2n-1) \times$

$(2n-1)$ neighborhood as the n-voxel tolerance region about a particular voxel. For a reconstructed surface voxel, if there is a true surface voxel (from the ground truth) within its n-voxel tolerance region, we consider it as a *valid* surface voxel with n-voxel tolerance. Let the number of estimated valid surface voxels with n-voxel tolerance be $NESV_R(n)$, and the total number of true surface voxels from the ground truth be NSV_{GT}. Surface voxel accuracy with n-voxel tolerance $SVA(n)$ is defined by the ratio between them, as shown in Equation 6.8.

$$SVA(n) = \frac{NESV_R(n)}{NSV_{GT}} \tag{6.8}$$

As we increase the tolerance, more false positives may be considered as valid surface voxels, so it is possible that this yields an accuracy measure greater than unity. We considered changing the denominator in Equation 6.8 to the total number of reconstructed surface voxels, NSV_R, i,e.,

$$SVA(n)(\text{not used}) = \frac{NESV_R(n)}{NSV_R} \tag{6.9}$$

so that ratios never exceed unity. However, values close to one, are not necessarily indicative of a perfect reconstruction. As an extreme case, suppose only one surface voxel of a larger object is reconstructed, but this is a true surface voxel (i.e., on the ground truth surface). According to Equation 6.9, this would yield an accuracy measure of unity, even though the reconstruction clearly failed. As such, we rejected this change.

In addition, we also consider a surface voxel measurement we refer to as surface voxel completeness (SVC). For a ground truth surface voxel, if there is a reconstructed

surface voxel in its n-voxel tolerance region, we say that it is *covered* with n-voxel tolerance. The SVC measure is indicative of how well the actual surface, from ground truth, is covered by a particular reconstruction. Assume that the number of covered surface voxels with n-voxel tolerance in the ground truth is $NCSV_{GT}(n)$. The surface voxel completeness with n-voxel tolerance ($SVC(n)$) is defined as:

$$SVC(n) = \frac{NCSV_{GT}(n)}{NSV_{GT}} \tag{6.10}$$

While SVA and SVC are fairly specific measures, we can draw some general conclusions regarding reconstruction quality when these metrics are combined with the overall match rate.

The F-norm method only provides voxel occupancy information, and thus, we are limited to reproducing a shaded model rather than one with actual color. From our experiments, we can see that although the overall model resulting from this method is correct, there are obvious errors, in particular with respect to concave regions. Unfortunately, it is very difficult to make a meaningful evaluation of reconstruction quality given only a shaded model. This is particular evident when assessing the cactus reconstruction, shown in Figure 6–17. Although the shaded model appears to be missing the intricate detail of the cactus thorns, we find that this is the case with other reconstruction methods as well. Such details are likely provided almost entirely by the texture map, rather than from actual depth estimates. In order to provide a more useful comparison of our algorithm with the GVC approach, which estimates color along with occupancy, we assign a color to each

surface voxel. This is calculated using the average pixel color from all cameras that can, according to the model, observe the voxel.

Table 6–1 summarizes the match rates comparisons for several simulated data sets. We compared these scenes both with and without the additional of a textured background (the addition of "nbg" to the dataset name, e.g., al-nbg refers to the case without background). Surface voxel accuracy and completeness results are compared using a 3-voxel tolerance in Tables 6–2 and 6–3. Figures 6–18-6–20 illustrate sample results of the different methods. From a qualitative perspective, our methods provide results that are as good as GVC for source data without background, and better than GVC for data with background. Apart from the rose data set, for which GVC has some difficulty, all three methods achieve good surface completeness. While GVC obtains better surface accuracy on the other data sets, its overall match rates (Table 6–1) are generally worse than our methods, in particular for those including backgrounds, as many empty voxels are not removed, as illustrated in Figures 6–18-6–20. With respect to our goal of object segmentation and reconstruction, this is clearly unacceptable. While our F-norm method also retains many empty voxels, these are generally isolated, as noise, and can be removed easily by tensor voting or similar techniques, as illustrated in Figure 6–29. This approach is discussed below in further detail.

For the results shown here, the histogram method appears to offer better quality reconstructions, in particular for the data sets including a background. This may be a result of using only the mode of color frequency (histogram). Provided that there is a prominent color in the projections of a voxel, it is relatively easy to obtain a good

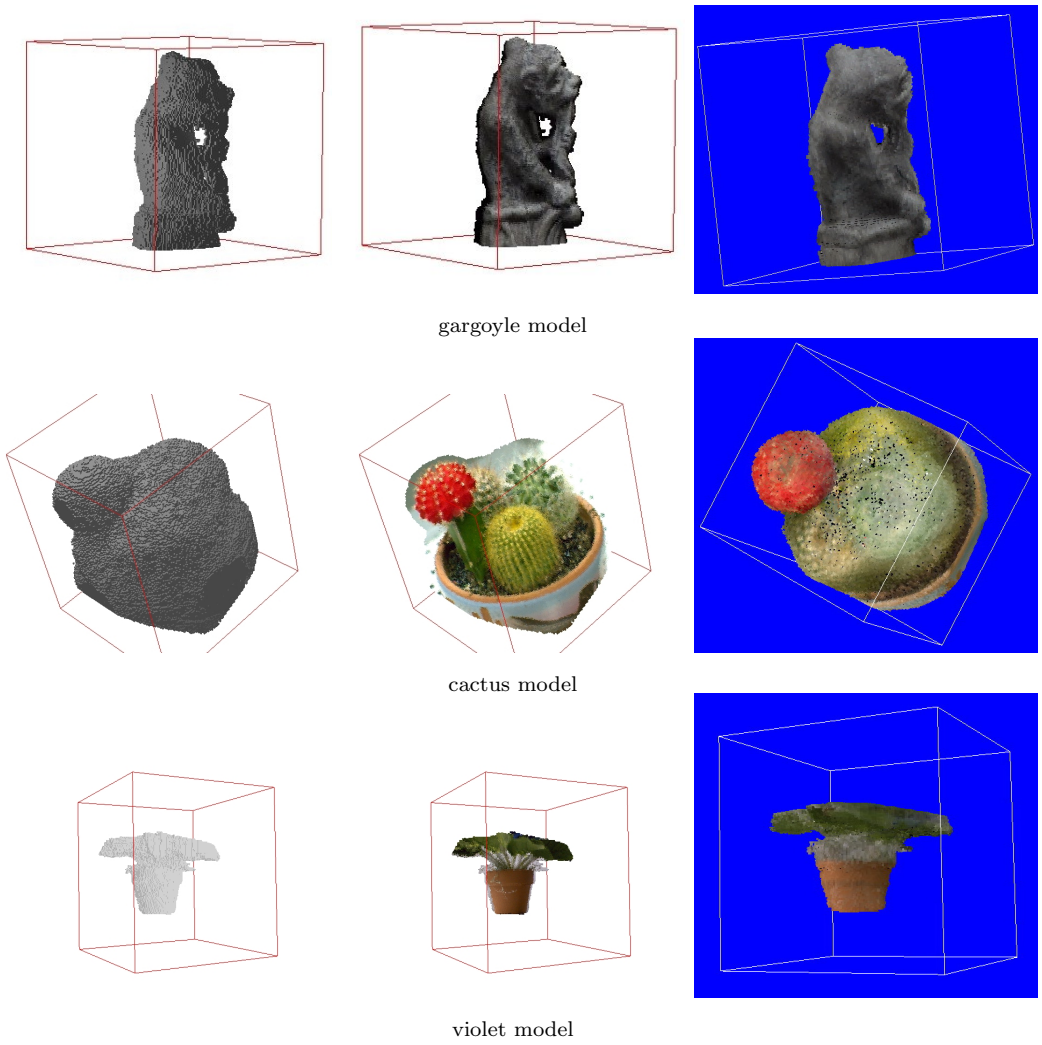


Figure 6–17: Visualization of reconstructed models based on F-norms in shaded models (1st column), texture mapped models (2nd column), and colored models (3rd column).

reconstruction. In the simulation data sets we have chosen, this seems to be the case, as the foreground objects have a relatively small number of colors that are generally different from the background. However, as demonstrated by our experiments in Chapter 5, the histogram method performs poorly with real data (see Figure 5-7 and 5-8, page 78 and 79), suggesting that it does not extend well to real scenes. For these, the F-norm is likely a better choice, as our comparisons indicate that the F-norm is more robust for data sets including a cluttered background. This is confirmed by our experiments with real scene data, as shown in Figures 6–21-6–23.

Both GVC and the F-norm method provide reasonable reconstructions for the gargoyle data set, although the former does a better job in concave regions. For example, GVC reproduces the holes formed by the head and arm in the second row of Figure 6–21, in better agreement with the reference images, while the F-norm seems better at removing empty voxels. Both methods have difficulty reconstructing fine surface details, for example, the sculpture in the base. Again, GVC is slightly superior as seen in the details of the hands and arms in the third row. We speculate that the inferior results of the F-norm method may be due to it not using visibility information and to the overly simplistic $0-1$ representation of its camera agreement matrix.

The results on the cactus data set of Figure 6–22 differ more significantly. In its current form, the F-norm method can only reconstruct regions that are visible to all cameras, as required for construction of a valid camera agreement matrix. As a result, we fail to reconstruct the base of the cactus pot, because the associated voxels are not visible to all cameras. This is a clear shortcoming of our present

	al	al-nbg	rose	rose-nbg	deer	deer-nbg
GVC	66.52	90.41	62.42	97.80	32.5	97.58
Hist	92.67	92.67	98.15	98.13	98.67	98.69
F-norm	93.09	93.06	96.05	96.38	97.07	97.17

Table 6–1: Match rates of GVC, Hist, and F-norm for simulation data sets, expressed as percentages.

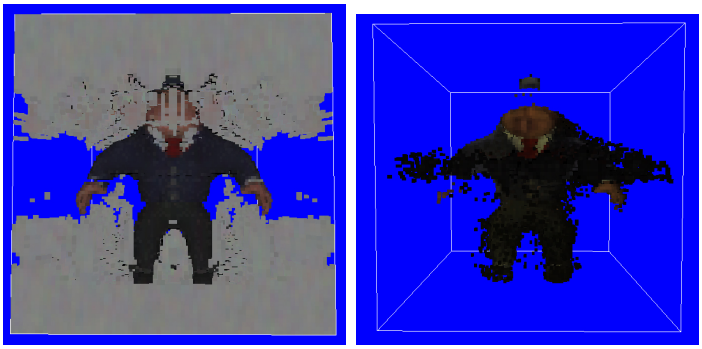
implementation, which the GVC method does not suffer, as the latter uses only the cameras that can see any particular voxel. Similarly, this explains why the flower and leaves of the violet data set in Figure 6–23 appear noticeably flat in the F-norm results. However, while GVC reconstructs a greater portion of the cactus base, its shape is clearly incorrect, as seen in the second and third row of Figure 6–22. An additional difference is that the F-norm method sometimes exhibits bulges in its reconstruction, as seen in the exaggerated size of the red cactus. For the violet data set of Figure 6–23, GVC retains more empty voxels, but provides a perceptually superior surface model.

It should be noted, however, that GVC implementation [99], which uses a weighted color variance photo-consistency test, is not the best reconstruction method for the cactus and violet data sets. Kutulakos [89] reported similar results for these with standard space carving techniques and suggested that improved results could be achieved using an r-shuffle photo-consistency test [89]. Some of his results are shown here in Figure 6–24 and 6–25 for comparison purposes.

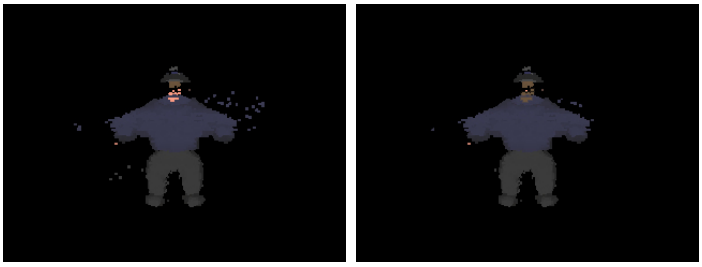
6.7.2 Comparison of computational requirements

Both our methods and space carving have $O(N)$ complexity, where N is the number of voxels used. However, while our methods must scan voxels once, space

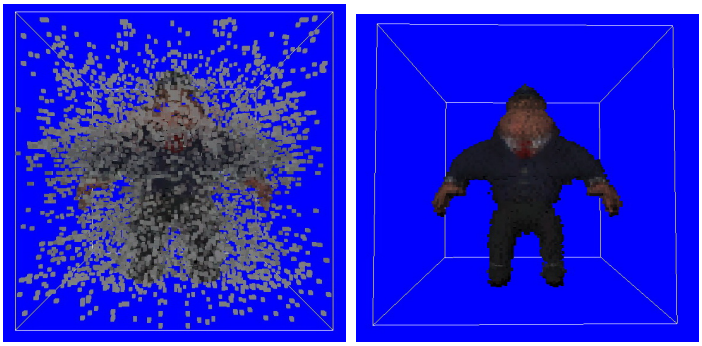
original images, with and without background (both use 24 images in total)



Generalized Voxel Coloring



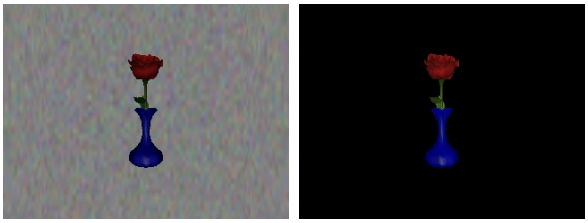
Histogram



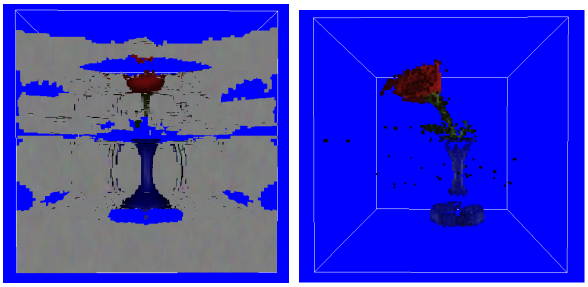
F-norm (our method)

with background without background

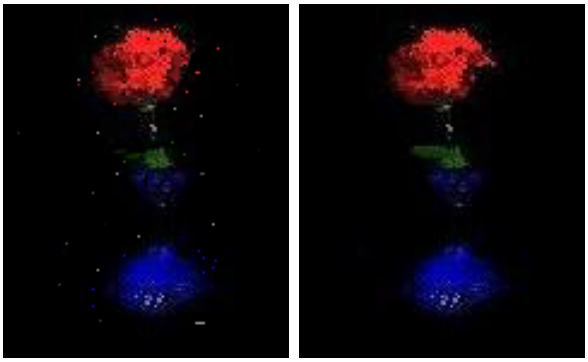
Figure 6–18: Reconstruction results on simulated 'al' data. 24 cameras and 80x80x80 voxels are used.



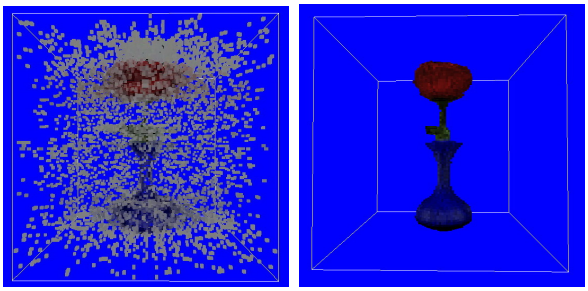
original images, with and without background (both use 24 images in total)



Generalized Voxel Coloring



Histogram

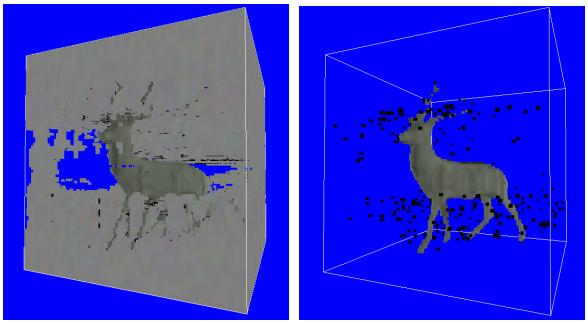


F-norm (our method)

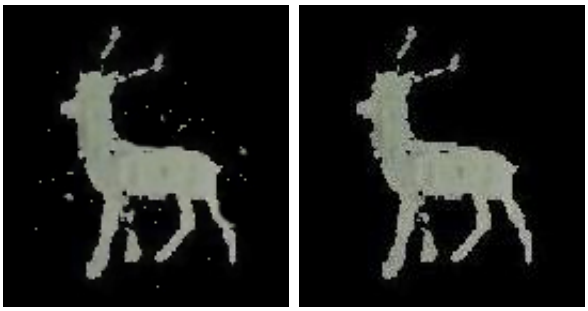
with background without background

Figure 6–19: Reconstruction results on simulated 'rose' data. 24 cameras and 80x80x80 voxels are used.

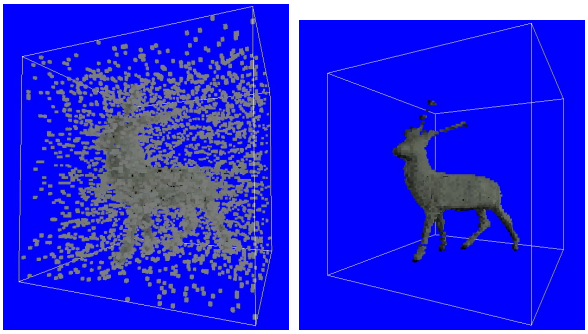
original images, with and without background (both use 24 images in total)



Generalized Voxel Coloring



Histogram

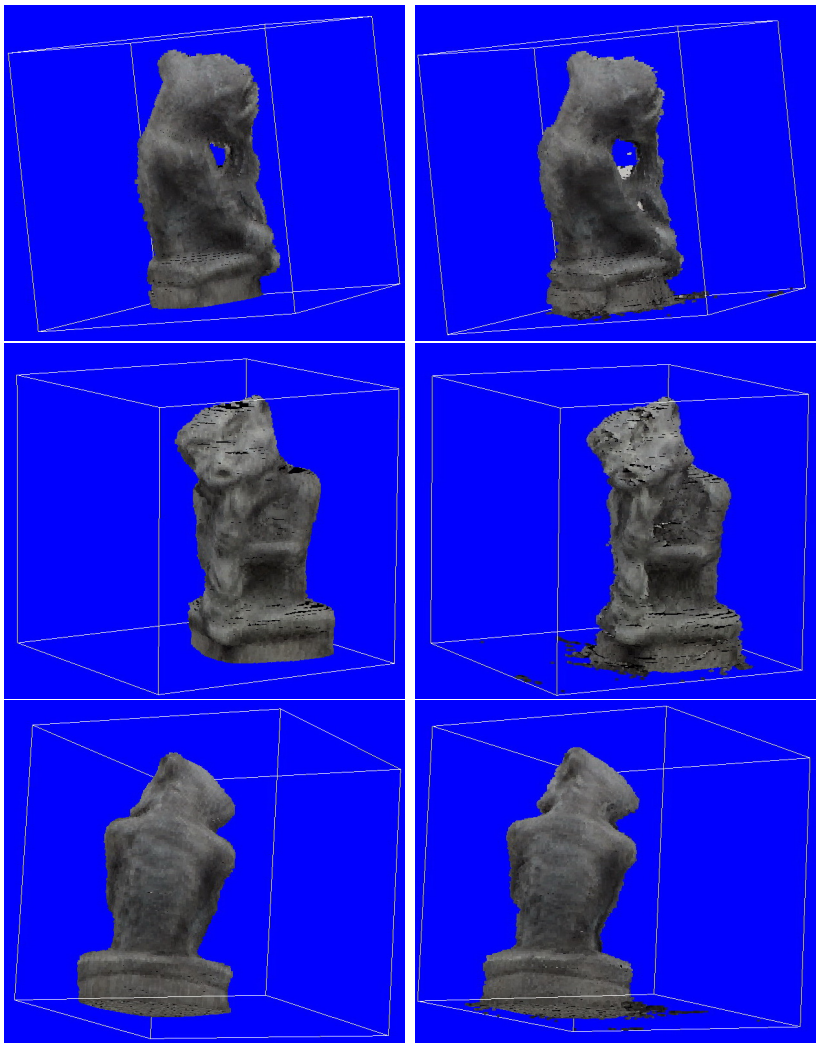


F-norm (our method)

with background without background

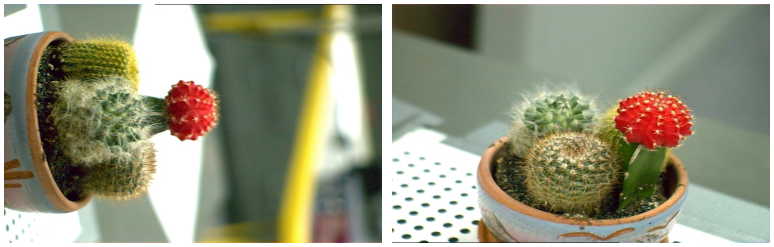
Figure 6–20: Reconstruction results on simulated 'deer' data. 24 cameras and 80x80x80 voxels are used.

original images (out of 16 images)

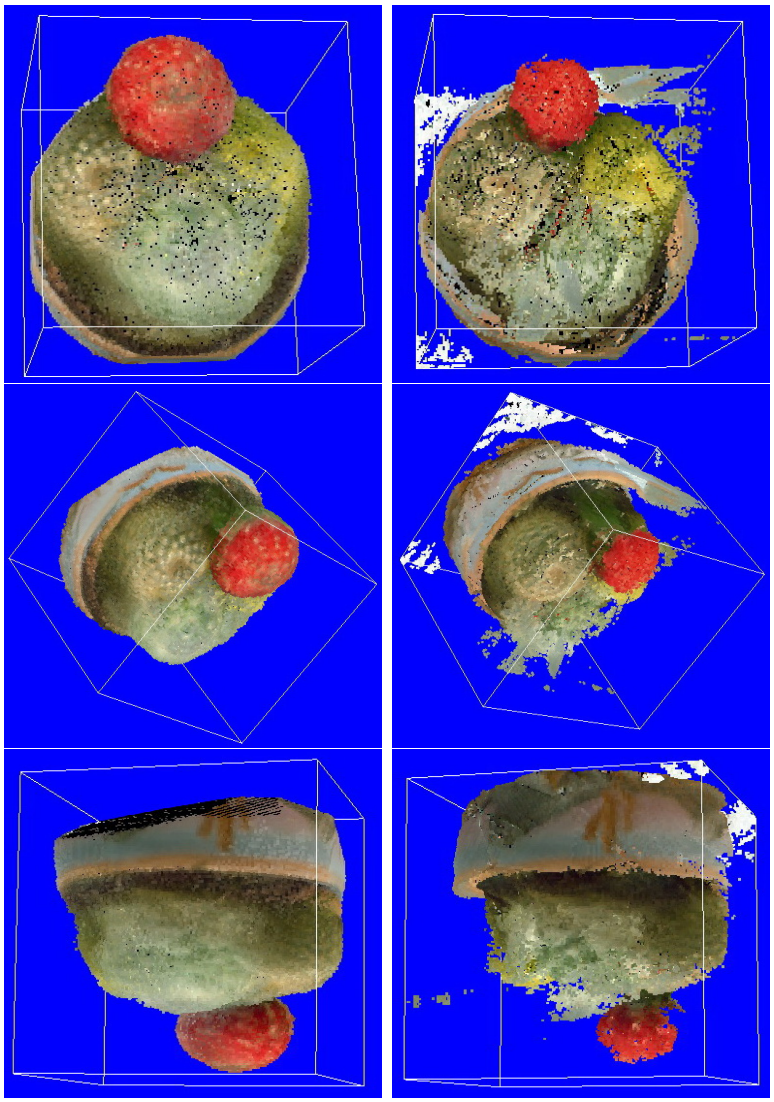


F-norm (our method) Generalized Voxel Coloring

Figure 6–21: Comparison of reconstruction results on 'gargoyle' data. 128x128x128 voxels are used.



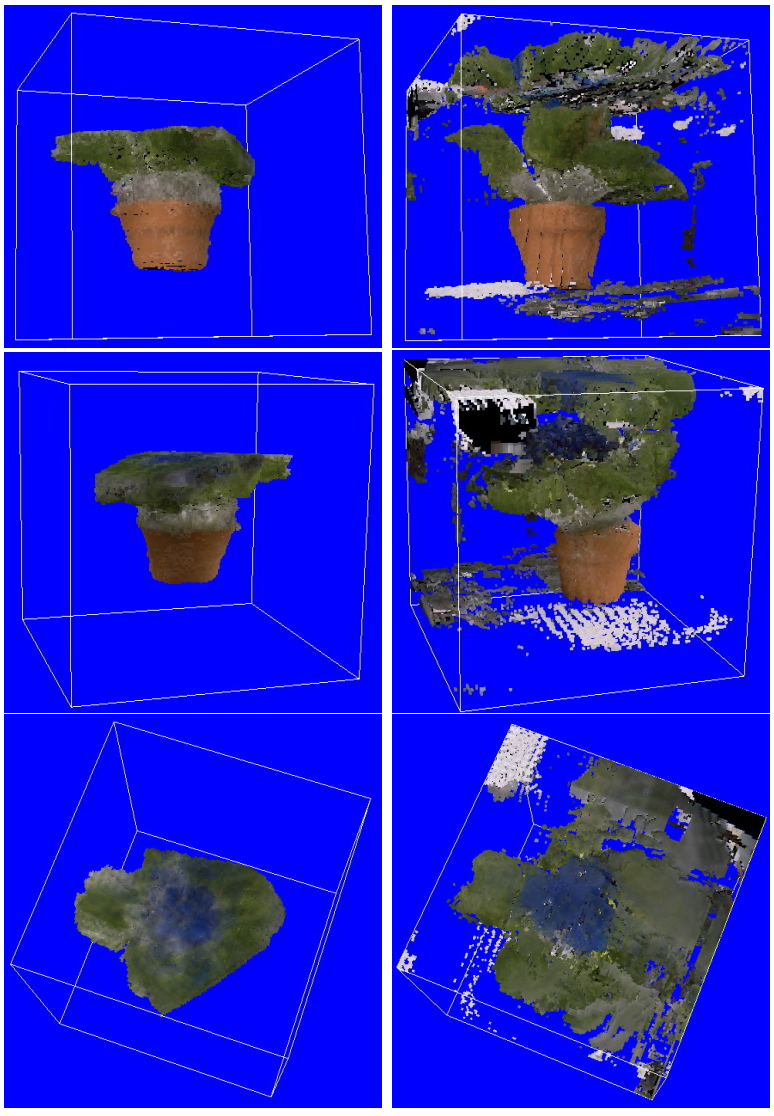
original images (out of 30 images)



F-norm (our method) Generalized Voxel Coloring

Figure 6–22: Comparison of reconstruction results on 'cactus' data. 128x128x128 voxels are used.

original images (out of 40 images)



F-norm (our method) Generalized Voxel Coloring

Figure 6–23: Comparison of reconstruction results on 'violet' data. 128x128x128 voxels are used.

	al	al-nbg	rose	rose-nbg	deer	deer-nbg
GVC	88.17	70.35	52.96	54.80	98.63	70.88
Hist	62.14	64.02	68.53	68.94	68.26	68.85
F-norm	62.52	63.74	66.05	66.83	66.21	67.45

Table 6–2: Surface voxel accuracy (within 3-voxel tolerance) of GVC, Hist, and F-norm for simulation data sets, expressed as percentages.

	al	al-nbg	rose	rose-nbg	deer	deer-nbg
GVC	96.52	90.67	72.33	86.99	95.11	94.67
Hist	88.34	88.21	90.12	90.61	92.76	92.58
F-norm	95.73	96.77	90.23	93.38	93.23	94.91

Table 6–3: Surface voxel completeness (within 3-voxel tolerance) of GVC, Hist, and F-norm for simulation data sets, expressed as percentages.

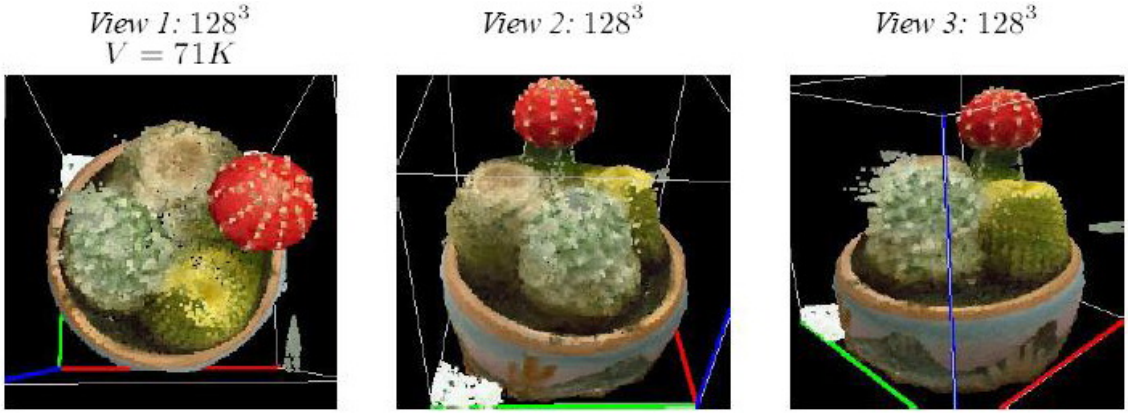


Figure 6–24: Cactus reconstruction by approximate space carving. Reprinted from Kutulakos [89] with permission, ©2000 Springer-Verlag.

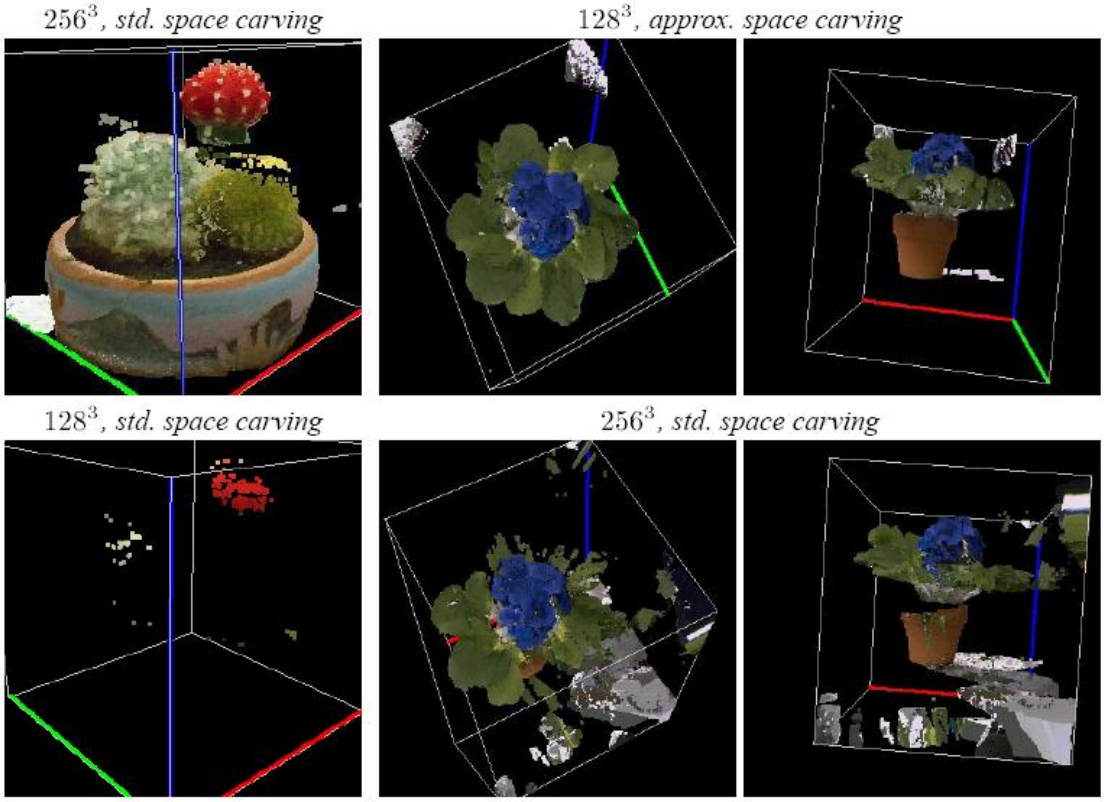


Figure 6–25: Comparison of reconstruction results computed with the standard and approximate space carving algorithm. Reprinted from Kutulakos [89] with permission, ©2000 Springer-Verlag.

carving methods may carry out multiple passes, requiring more processing time. Our experiments were performed using an Intel dual core 2.8GHz processor with 2GB of RAM. All of the simulation data sets consisted of 24 images, while the real data sets ranged between 16 images (gargoyle), 30 images (cactus), and 40 images (violet). For all the methods compared, we used a model of 80x80x80 voxels for simulated data sets and 128x128x128 voxels for real data sets.

In order to resolve visibility and collect projection colors for every voxel, GVC must reproject the current estimated model to all reference images on every iteration. The number of iterations depends both on scene complexity and number of images, which in turn, affect how many voxels can be carved in one iteration. In contrast, the F-norm does not reconstruct objects iteratively. Instead, each voxel is projected into each image only once, thereby offering significant time savings. Since only local information is used, a parallel implementation of our F-norm method is also possible, providing further computational advantages.

As shown in Table 6–4, computational performance of the F-norm method was dramatically superior to GVC, often by an order of magnitude. Execution time was largely dependant on number of voxels, number of source images, and scene complexity. As a consequence, both F-norm and GVC methods require considerably more time to process real data than simulation data. The difference is even more obvious in the case of the F-norm method. This is because we use the more expensive NCC to compute the camera agreement for real data, as opposed to simple pixel color difference for simulated data.

	gargoyle	cactus	violet	al	rose	deer
GVC	21	102	139	20	19	42
F-norm	12	14	15	1	1	1

Table 6–4: Comparison of running time (in minutes) between GVC and F-norm implementations.

6.7.3 Effect of number of cameras

To validate our method, we evaluated its performance with a varying number of source cameras, with representative results shown in Figure 6–26. Surface voxel accuracy and completeness comparisons are provided in Figures 6–27 and 6–28 and match rates are listed in Table 6–5. As expected, the quality of results increases as a function of the number of cameras used, with steadily diminishing returns. For example, the reconstruction result using 84 cameras is not significantly better than that using 42 cameras. Normally, a greater surface voxel accuracy is indicative of improved surface reconstruction, but as mentioned earlier, using too great a tolerance results in higher numbers of false positives. This is reflected by accuracy measurements that exceed unity, as is the case for 24 cameras and more with a 7-voxel tolerance; lower tolerances are desired to achieve a more accurate reconstruction. In this regard, the results of Figure 6–28 are encouraging, as we see that as the number of cameras increases, the surface voxel completeness measurements approach 100% with a smaller voxel tolerance. While the quantitative measures indicate improved reconstruction quality as the number of cameras increases, the 24 camera case appears noisier than 8 and 12 camera cases, as shown in Figure 6–26. This suggests the possibility that more non-surface voxels may be taken accidentally to be surface

camera number	8	12	24	42	84
match rate	95.66	96.25	97.05	98.35	98.37

Table 6–5: Match rate with different number of cameras for the deer data set, expressed as percentages.

	al	rose	deer
before tv	93.09	96.05	97.07
after tv	93.52	96.53	97.55

Table 6–6: Match rate changes before and after tensor voting, expressed as percentages.

voxels, given a certain number of cameras, and that these errors may be reduced by using a larger number of cameras (as in the results from 42 and 84 cameras shown).

6.7.4 Effect of tensor voting

We also investigate the improvements offered by tensor voting applied to our method, as shown in Figure 6–29. Noisy voxels are removed and more surface voxels are recovered, demonstrating that tensor voting can recover errors from the early classification stage. Quantitative comparisons of match rate and surface voxel accuracy are listed in Table 6–6, 6–7, and 6–8, which also demonstrate this improvement, in particular with respect to surface voxel accuracy.

	al	rose	deer
before tv	62.52	66.05	66.21
after tv	73.27	74.83	78.84

Table 6–7: Surface voxel accuracy (within 3-voxel tolerance) changes before and after tensor voting, expressed as percentages.

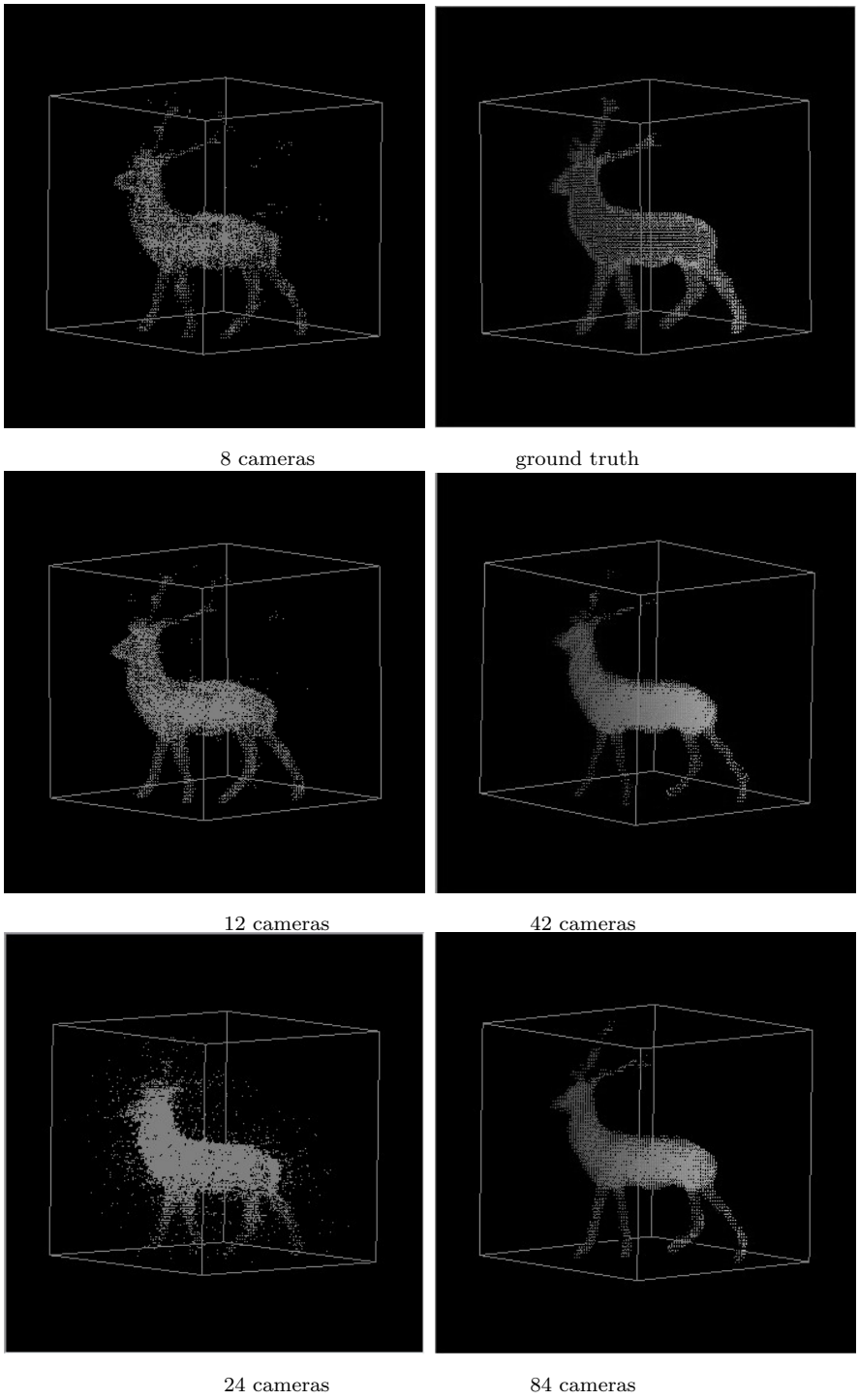


Figure 6–26: Performance of F-norm algorithm with different number of cameras for the deer data set.

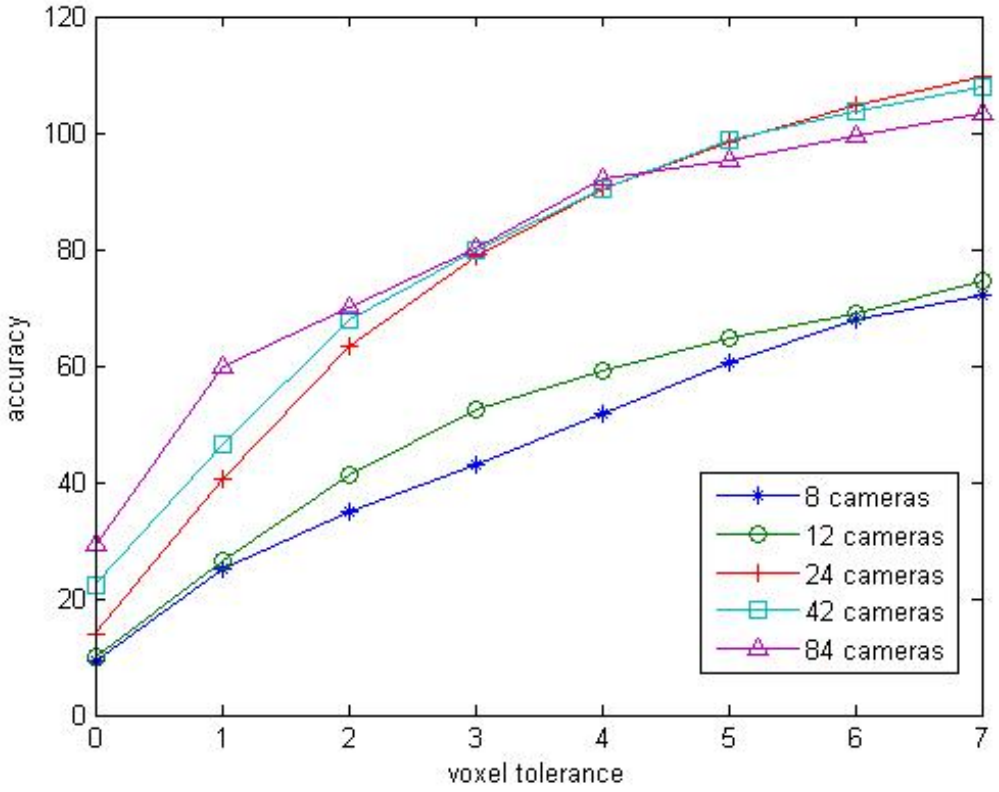


Figure 6–27: Reconstruction accuracy within n-voxel tolerance for the deer data set.

	al	rose	deer
before tv	95.73	90.23	93.23
after tv	98.18	93.16	95.67

Table 6–8: Surface voxel completeness (within 3-voxel tolerance) changes before and after tensor voting, expressed as percentages.

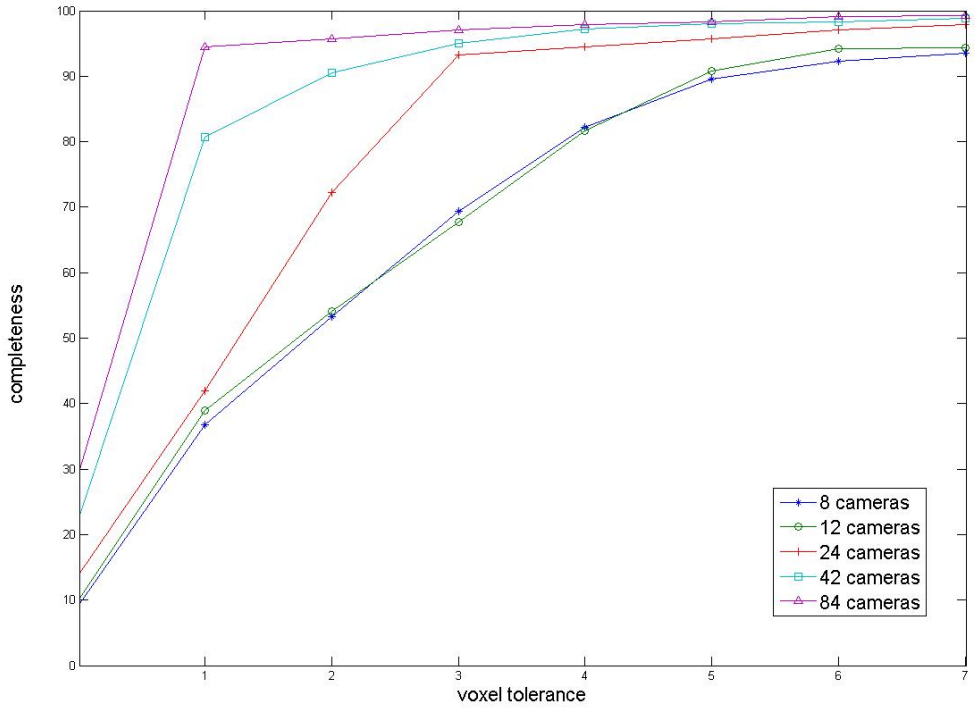
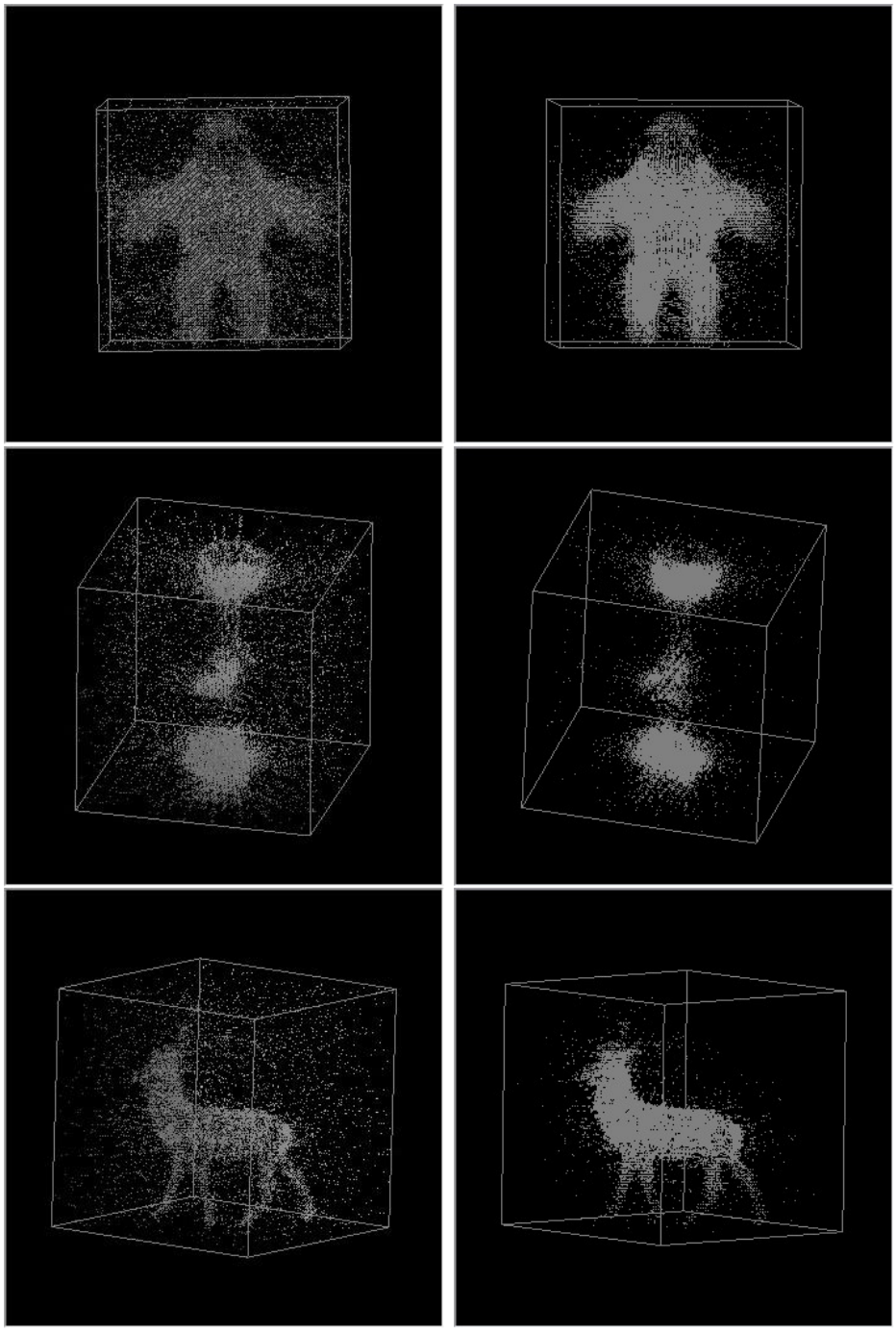


Figure 6–28: Surface voxel completeness within n-voxel tolerance for the deer data set.



before tensor voting after tensor voting

Figure 6–29: Effect of tensor voting. The results from Figure 6–10 through 6–12 are reproduced here for comparison.

CHAPTER 7
Future Work and Conclusions

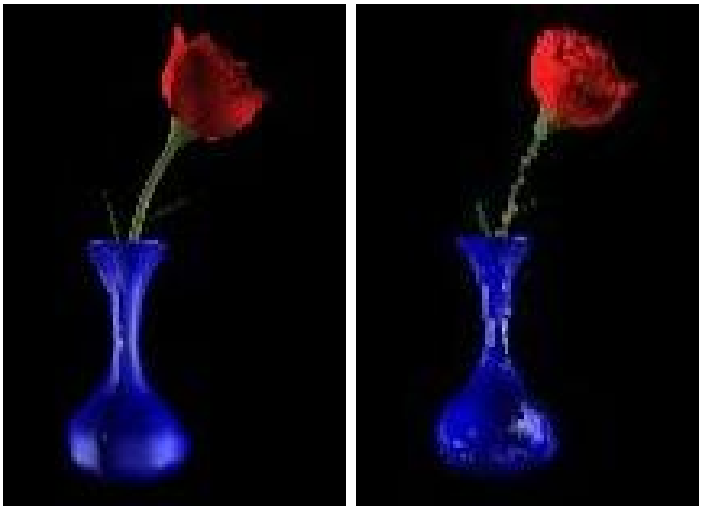
A volumetric reconstruction method has been developed. It provides a new point of view to analyze the reconstruction problem by considering it as classification. It overcomes some drawbacks related to space carving, including: being able to recover early carving errors, choosing a threshold automatically, and improving computational efficiency. However, it still has some limitations. Visibility information is not addressed explicitly. The simple representation and feature is not powerful enough. Some possible future improvements are discussed in the following sections.

7.1 Deal with Specularity

Throughout this thesis, we have assumed Lambertian properties, which is not always valid. Figure 7–1 shows such an example, where the reconstruction based on our method exhibits obvious artifacts as a result. Some techniques as suggested by Yang *et al.* [164] and Bonfort and Sturm [19] may be considered to address this problem.

7.2 Consider Other Features than F-norm

As mentioned in the previous chapter, a $0-1$ representation for the camera agreement matrices may be over-simplified. Possible improvements include the use of measurements like SSD or NCC in the camera agreement matrix and eigenvectors as features. Numerous pixel similarity measurements used in stereo matching could be applied in the camera agreement matrix.



a b

Figure 7–1: Problematic reconstruction in specular regions. (a) a reference image. (b) a reconstructed image.

Depending on the information available for different applications, various additional features may be considered, such as deviation from a known background or the shape of the color distribution of a voxel's projections. As indicated in Chapter 5, a peak in the distribution may suggest a surface voxel.

7.3 Exploit Neighbouring Information by Methods other than Tensor Voting

At present, our method makes an initial surface estimate based on a voxel score (F-norm), and tensor voting works on the structure tokens (surface points) to refine the result. It may be advantageous to adjust voxel scores directly when considering neighbouring information from cooperative algorithms or diffusion algorithms. This way, judgements about surface voxels are postponed to the final step.

7.4 Extension to Dynamic Scenes and Video Sequences

We also want to extend our methods to dynamic scenes with video sequences as input. As new images are captured, the model built in the previous time step needs to be updated. Since images are captured at video rates, objects generally cannot move far between consecutive frames, and we thus only need to consider a restricted neighbourhood of existing voxels to check whether they now represent surfaces. Some results using the photo consistency measurement described in Chapter 5 are shown in Figure 7–2, in which the object performs a rotation while translating.

Vedula *et al.* [159, 160] combined volumetric representation and motion estimation methods, obtaining good results for interpolating a volume model from 3D scene flow. While our task is quite similar, they use an offline process that can achieve arbitrary spatial and temporal interpolation. As a result, optical flow estimation is required to compute 3D scene flow, which is needed for the interpolation. We might do something similar by considering flow information as well. For example, we can limit the search space to a range determined by the model from the previous frame and 3D scene flows. When determining whether or not a voxel is occupied, besides considering color similarity in the camera agreement matrix, we may also consider constraints from optical flow.

7.5 Conclusions

In this thesis, various color correction methods, useful for multi-camera applications, have been investigated. A nonlinear diffusion technique was proposed to improve depth maps by considering intensity gradient constraints. This may also be used as a post-processing step to any stereo algorithm. Finally, characteristics of

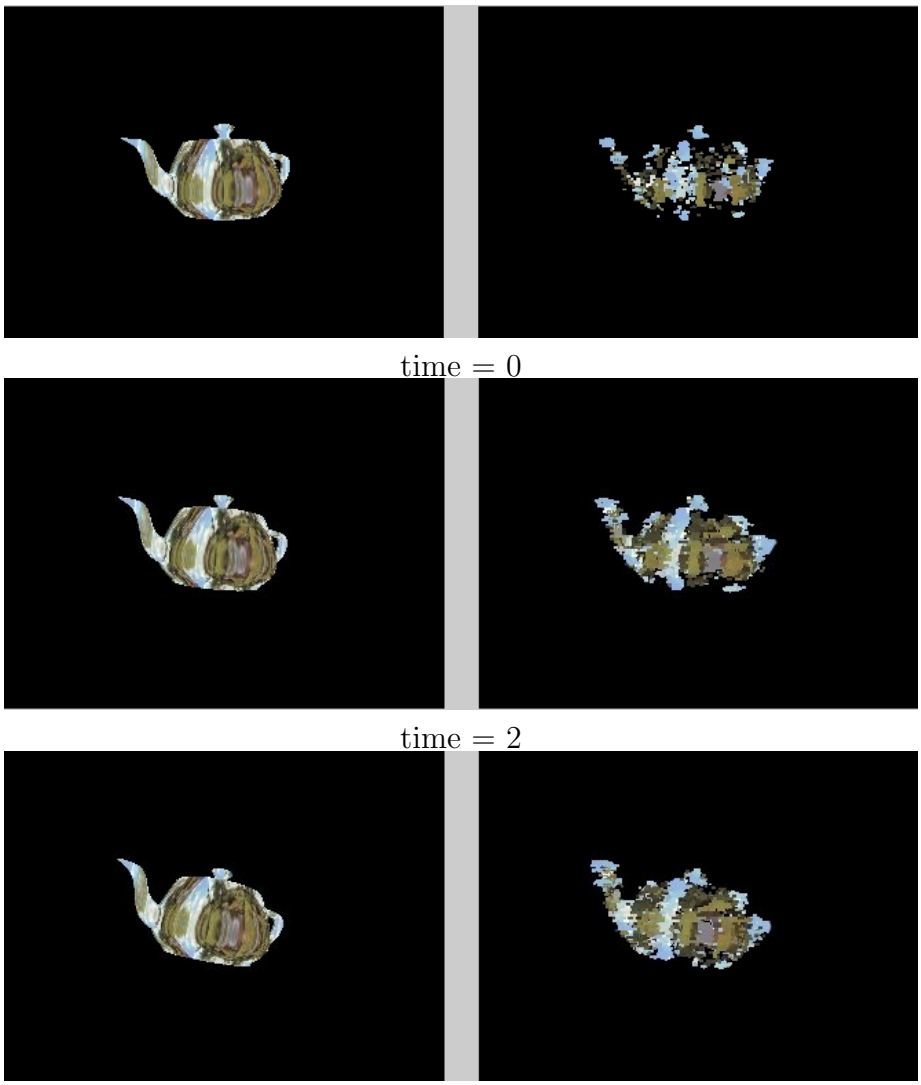


Figure 7–2: Model updating. The images from one camera (of a total of 24) at different times are shown here. The object undergoes both translation and rotation. The reconstructed models are projected to the same camera position.

different voxel categories were analyzed, and voxel occupancy was taken as a classification problem. This suggests a new way of approaching the reconstruction problem. The challenge with this new approach is to find suitable features. One idea proposed was to accumulate information into a camera-agreement matrix, using features such as the Frobenius norm to classify voxels. Tensor voting was applied to improve the result. Experimental results demonstrated the feasibility of the proposed method. Compared with existing methods such as space carving, this offers several advantages, such as the ability to recover incorrectly carved voxels, automatic calculation of a suitable threshold, and potential computational improvements. However, it is limited by the simple representation and feature currently used; we thus need to investigate more advanced representations and features in the future.

Our experiments lead to several observations of relevance to 3D reconstruction applications. First, color differences between cameras or projectors may be resolved by simple training data sets and correction methods as discussed in chapter 3. Second, A simple strategy, such as our histogram or F-norm method, followed by a tensor voting post-processing stage can provide reasonable reconstruction results. Most importantly, these results can be obtained without using explicit visibility information. This is of significance, as using only local information leads to substantial reduction of computational cost. Finally, as expected, increasing the number of cameras leads to improved reconstruction, although the effect is non-linear, with diminishing returns of quality. Since additional cameras entail increased processing demands, the tradeoff of quality with computation must be considered.

References

[1] M. Agrawal and L. S. Davis. A probabilistic framework for surface reconstruction from multiple images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 470–476, 2001.

[2] N. Ahuja and J. Veenstra. Generating octrees from object silhouettes in orthographic views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(2):137–149, 1989.

[3] L. Alvarez, J. Esclarin, M. Lefbure, and J. Snchez. A pde model for computing the optical flow. In *Proceedings of CEDYA XVI, Universidad de Las Palmas de Gran Canaria*, pages 1349–1356, 1999.

[4] H. H. Baker. Edge-based stereo correlation. In *In Proc. DARPA Image Understanding Workshop*, pages 168–175, 1980.

[5] H. H. Baker and T. O. Binford. Depth from edge and intensity based stereo. In *International Joint Conferences on Artificial Intelligence*, pages 631–636, 1981.

[6] K. Barnard, V. Cardei, and B. Funt. A comparison of computational color constancy algorithms—part 1: Methodology and experiments with synthesized data. *IEEE Trans. Image Processing*, 11(9):972–983, 2002.

[7] K. Barnard, F. Ciurea, and B. Funt. Sensor sharpening for computational color constancy. *Journal of the Optical Society of America A*, 18:2728–2743, 2001.

[8] K. Barnard, L. Martin, A. Coath, and B. Funt. A comparison of computational color constancy algorithms—part 2: Experiments with image data. *IEEE Trans. Image Processing*, 11(9):985–996, 2002.

[9] S. T. Barnard. Stochastic stereo matching over scale. *International Journal of Computer Vision*, 3(1):17–32, May 1989.

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

[11] P. A. Beardsley, P. H. S. Torr, and A. Zisserman. 3D model acquisition from extended image sequences. In *European Conference on Computer Vision*, pages 683–695, 1996.

[12] P. Belhumeur and D. Mumford. A Bayesian treatment of the stereo correspondence problem using half-occluded regions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 506–512, 1992.

[13] R. Bhotika, D. J. Fleet, and K. N. Kutulakos. A probabilistic theory of occupancy and emptiness. In *European Conference on Computer Vision*, pages 112–130, 2002.

[14] S. Birchfield and C. Tomasi. Depth discontinuities by pixel-to-pixel stereo. In *IEEE International Conference on Computer Vision*, pages 1073–1080, 1998.

[15] M. J. Black, G. Sapiro, D. H. Marimont, and D. Heeger. Robust anisotropic diffusion. *IEEE Trans. Image Processing*, 7(3):421–432, 1998.

[16] J. Bloomenthal. An implicit surface polygonizer. In *Graphics gems IV*, pages 324–349. Academic Press Professional, Inc., 1994.

[17] OpenGL Architecture Review Board, D. Shreiner, M. Woo, J. Neider, and T. Davis. *OpenGL(R) Programming Guide : The Official Guide to Learning OpenGL(R), Version 2 (5th Edition)*. Addison-Wesley Professional, 2005.

[18] R. C. Bolles, H. H. Baker, and M. J. Hannah. The jisct stereo evaluation. In *In Proc. DARPA Image Understanding Workshop*, pages 263–274, 1993.

[19] T. Bonfort and P. Sturm. Voxel carving for specular surfaces. In *IEEE International Conference on Computer Vision*, pages 591–596, 2003.

[20] K. L. Boyer and A. C. Kak. Color-encoded structured light for rapid active ranging. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 9(1):14–28, 1987.

[21] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[22] D. H. Brainard and W. T. Freeman. Bayesian color constancy. *Journal of the Optical Society of America A*, 14:1393–1411, 1997.

[23] A. Broadhurst and R. Cipolla. A statistical consistency check for the space carving algorithm. In *British Machine Vision Conference*, pages 282–291, 2000.

[24] A. Broadhurst, T. W. Drummond, and R. Cipolla. A probabilistic framework for space carving. In *IEEE International Conference on Computer Vision*, pages 388–393, 2001.

[25] L. G. Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–376, 1992.

[26] M. Z. Brown, D. Burschka, and G. D. Hager. Advances in computational stereo. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(8):993–1008, August 2003.

[27] P. Burt and B. Julesz. A disparity gradient limit for binocular vision. *Science*, 208:615–617, 1980.

[28] R. L. Carceroni and K. N. Kutulakos. Multi-view scene capture by surfel sampling: From video streams to non-rigid 3D motion, shape and reflectance. *International Journal of Computer Vision*, 49(2-3):175–214, 2002.

[29] V. Cardei. *A neural network approach to color constancy*. PhD thesis, Simon Fraser Univ., Burnaby, BC, Canada, 2000.

[30] G. K. M. Cheung, T. Kanade, J. Bouguet, and M. Holler. A real time system for robust 3D voxel reconstruction of human motions. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 02, pages 714–720, 2000.

[31] R. Cipolla and A. Blake. Surface shape from the deformation of apparent contours. *International Journal Computer Vision*, 9(2):83–112, 1992.

[32] R. Cipolla and D. Robertson. 3D models of architectural scenes from uncalibrated images and vanishing points. In *ICIAP '99: Proceedings of the 10th International Conference on Image Analysis and Processing*, pages 824–829, 1999.

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

[34] I. J. Cox, S. Roy, and S. L. Hingorani. Dynamic histogram warping of image pairs for constant image brightness. In *IEEE International Conference on Image Processing*, pages 2366–2369, 1995.

[35] G. Cross and A. Zisserman. Surface reconstruction from multiple views using apparent contours and surface texture. In *Confluence of computer vision and computer graphics*, pages 25–47. Kluwer Academic Publishers, 2000.

[36] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart. The cave: audio visual experience automatic virtual environment. *Communications of the ACM*, 35(6):64–72, 1992.

[37] W. B. Culbertson, T. Malzbender, and G. G. Slabaugh. Generalized voxel coloring. In *Workshop on Vision Algorithms*, pages 100–115, 1999.

[38] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996.

[39] C. J. Davies and M. S. Nixon. A hough transform for detecting the location and orientation of 3-dimensional surfaces via color encoded spots. *IEEE Trans. Systems, Man, and Cybernetics*, 28(1B):90–95, February 1998.

[40] J. S. de Bonet and P. Viola. Roxels: Responsibility weighted 3D volume reconstruction. In *IEEE International Conference on Computer Vision*, pages 418–425, 1999.

[41] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *Computer Graphics*, 30(Annual Conference Series):11–20, 1996.

[42] R. Deriche, P. Kornprobst, and G. Aubert. Optical-flow estimation while preserving its discontinuities: A variational approach. In *Proc. Second Asian Conference on Computer Vision*, pages 290–295, 1995.

[43] U. R. Dhond and J. K. Aggarwal. Structure from stereo: A review. *IEEE Trans. Systems, Man, and Cybernetics*, 19(6):1489–1510, November 1989.

[44] Y. Duan, L. Yang, H. Qin, and D. Samaras. Shape reconstruction from 3D and 2D data using pde-based deformable surfaces. In *European Conference on Computer Vision*, pages 238–251, 2004.

[45] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition).* Wiley-Interscience, 2000.

[46] C. R. Dyer. Volumetric scene reconstruction from multiple views. In *Foundations of Image Understanding*, pages 469–489. Kluwer, 2001.

[47] G. Egnal, M. Mintz, and R. Wildes. A stereo confidence metric using single view imagery. In *Proc. Vision Interface*, pages 162–170, 2002.

[48] O. Faugeras and R. Keriven. Variational principles, surface evolution, pde's, level set methods and the stereo problem. *IEEE Trans. Image Processing*, 7(3):336–344, 1998.

[49] G. D. Finlayson. Color in perspective. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18:1034–1038, 1996.

[50] G. D. Finlayson, M. S. Drew, and B. V. Funt. Diagonal transforms suffice for color constancy. In *International Conference on Computer Vision*, pages 164–171, 1993.

[51] G. D. Finlayson, M. S. Drew, and B. V. Funt. Spectral sharpening: Sensor transformations for improved color constancy. *Journal of the Optical Society of America A*, 11:1553–1563, 1994.

[52] G. D. Finlayson, B. V. Funt, and K. Barnard. Color constancy under varying illumination. In *International Conference on Computer Vision*, pages 720–725, 1995.

[53] A. Fitzgibbon, Y. Wexler, and A. Zisserman. Image-based rendering using image-based priors. In *IEEE International Conference on Computer Vision*, pages 1176–1183, 2003.

[54] D. Forsyth. A novel algorithm for color constancy. *International Journal of Computer Vision*, 5:5–36, 1990.

[55] P. Fua and Y. G. Leclerc. Object-centered surface reconstruction: combining multi-image stereo and shading. *International Journal of Computer Vision*, 16(1):35–55, 1995.

[56] A. Fusiello, E. Trucco, and A. Verri. A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, 12(1):16–22, 2000.

[57] G. Gamble and T. Poggio. Visual integration and detection of discontinuities: The key role of intensity edges. AI Lab Memo 970, MIT, 1987.

[58] D. Geiger and F. Girosi. Parallel and deterministic algorithms from mrf's: Surface reconstruction. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13:401–412, 1991.

[59] D. Geiger, B. Ladendorf, and A. Yuille. Occlusion and binocular stereo. In *European Conference on Computer Vision*, pages 425–433, 1992.

[60] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[61] B. Georgescu and P. Meer. Point matching under large image deformations and illumination changes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(6):674–688, 2004.

[62] M. Goesele, B. Curless, and S. M. Seitz. Multi-view stereo revisited. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2402–2409, 2006.

[63] R. C. Gonzalez and R. E. Woods. *Digital Image Processing (2nd Edition).* Prentice Hall, 2002.

[64] W. E. L. Grimson. Computational experiments with a feature based stereo algorithm. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 7:17–34, 1985.

[65] G. Guy and G. Medioni. Inferring global perceptual contours from local features. *International Journal of Computer Vision*, 20(1-2):113–133, 1996.

[66] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2000.

[67] S. Haykin. *Neural Networks – a Comprehensive Foundation.* Prentice Hall, New Jersey, 2nd edition, 1999.

[68] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 71–78, 1992.

[69] B. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–204, 1981.

[70] S. S. Intille and A. F. Bobick. Incorporating intensity edges in the recovery of occlusion regions. In *International Conference on Pattern Recognition*, pages A:674–677, 1994.

[71] H. Ishikawa and D. Geiger. Occlusions, discontinuities, and epipolar lines in stereo. In *European Conference on Computer Vision*, pages 232–248, 1998.

[72] M. Jackowski, A. Goshtasby, S. Bines, and D. Roseman. Correcting the geometry and color of digital images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(10):1152–1158, 1997.

[73] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[74] M. R. M. Jenkin, A. D. Jepson, and J. K. Tsotsos. Techniques for disparity measurement. *Computer Vision Graphics and Image Processing(CVGIP): Image Understanding*, 53(1):14–30, 1991.

[75] H. Jin, S. Soatto, and A. J. Yezzi. Multi-view stereo beyond lambert. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 171–178, 2003.

[76] H. Jin, S. Soatto, and A. J. Yezzi. Multi-view stereo reconstruction of dense shape and complex appearance. *International Journal of Computer Vision*, 63(3):175–189, 2005.

[77] D. G. Jones and J. Malik. A computational framework for determining stereo correspondence from a set of linear spatial filters. In *European Conference on Computer Vision*, pages 395–410, 1992.

[78] T. Kanade, A. Gruss, and L. R. Carley. A very fast vlsi rangefinder. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation (ICRA '91)*, volume 2, pages 1322–1329, April 1991.

[79] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. *IEEE Trans. Pattern Recognition and Machine Intelligence*, 16(9):920–932, 1994.

[80] T. Kanade, P. Rander, and P. J. Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE Multimedia, Immersive Telepresence*, 4(1):34–47, 1997.

[81] H. R. Kang and P. G. Anderson. Neural network applications to the color scanner and printer calibrations. *Journal of Electronic Imaging*, 1(2):125–135, 1992.

[82] M. Kass. Linear image features in stereopsis. *International Journal of Computer Vision*, 1(4):357–368, January 1988.

[83] P. Kauff, N. Brandenburg, M. Karl, and O. Schreer. Fast hybrid block- and pixel- recursive disparity analysis for real-time applications in immersive teleconference scenarios. In *Proc. of 9th International Conference in Central Europe on Computer Graphics, Visualization, and Computer Vision*, pages 198–205, 2001.

[84] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220, 4598:671–680, 1983.

[85] R. Koch. 3D surface reconstruction from stereoscopic image sequences. In *IEEE International Conference on Computer Vision*, pages 109–114, 1995.

[86] R. Koch, M. Pollefeys, and L. J. Van Gool. Multi viewpoint stereo from uncalibrated video sequences. In *European Conference on Computer Vision*, pages 55–71, 1998.

[87] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions via graph cuts. In *IEEE International Conference on Computer Vision*, pages 508–515, 2001.

[88] K. N. Kutulakos. http://www.cs.toronto.edu/~kyros/softdata/static/index.html. Accessed in March 2005.

[89] K. N. Kutulakos. Approximate n-view stereo. In *European Conference on Computer Vision*, pages 67–83, 2000.

[90] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218, July 2000.

[91] E. H. Land. The retinex theory of color vision. *Scientific American*, 237:108–129, 1977.

[92] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 16:150–162, 1994.

[93] A. Laurentini. How far 3D shapes can be understood from 2D silhouettes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(2):188–195, 1995.

[94] A. Laurentini. How many 2D silhouettes does it take to reconstruct a 3D object? *Computer Vision and Image Understanding*, 67(1):81–87, 1997.

[95] M. Lee and G. Medioni. Inferred descriptions in terms of curves, regions, and junctions from sparse, noisy binary data. In *Proc. Int. Workshop on Visual Form, Capri, Italy*, pages 350–367, 1997.

[96] M. Lee, G. Medioni, and P. Mordohai. Inference of segmented overlapping surfaces from binocular stereo. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(6):824–837, 2002.

[97] C. Loop and Z. Zhang. Computing rectifying homographies for stereo vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages I: 125–131, 1999.

[98] W. E. Lorensen and H. Cline. Marching cubes: a high resolution 3D surface construction algorithm. In *Siggraph'87*, pages 163–169, 1987.

[99] Loper M. Archimedes: Shape reconstruction from pictures. http://matt.loper.org/Archimedes/Archimedes_docs/html/index.html. Accessed in December 2007.

[100] D. Marr and T. Poggio. Cooperative computation of stereo disparity. *Science*, 194(4262):283–287, 1976.

[101] D. Marr and T. Poggio. A computational theory of human stereo vision. *Proceedings of the Royal Society of London. Series B*, 204:301–328, 1979.

[102] J. L. Marroquin, S. Mitter, and T. Poggio. Probabilistic solution of ill-posed problems in computational vision. *Journal of the American Statistical Association*, 82(397):76–89, March 1987.

[103] W. N. Martin and J. K. Aggarwal. Volumetric descriptions of objects from multiple views. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 5(2):150–158, 1983.

[104] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. In *Siggraph 2000*, pages 369–374, 2000.

[105] G. Medioni and P. Mordohai. The tensor voting framework. In G. Medioni and S. B. Kang, editors, *Emerging Topics in Computer Vision*, chapter 5. Prentice Hall, 2004.

[106] G. G. Medioni, M. Lee, and C. Tang. http://iris.usc.edu/~tensorvt. Accessed in September 2006.

[107] P. Mordohai and G. Medioni. Perceptual grouping for multiple view stereo using tensor voting. In *ICPR '02: Proceedings of the 16 th International Conference on Pattern Recognition, Volume 3*, pages 639–644, 2002.

[108] P. Mordohai and G. Medioni. Dense multiple view stereo with general camera placement using tensor voting. In *3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium*, pages 725–732, 2004.

[109] P. Mordohai and G. Medioni. Stereo using monocular cues within the tensor voting framework. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(6):968–982, 2006.

[110] P. Mordonhai. *A Perceptual Organization Approach for Figure Completion, Binocular and Multiple-View Stereo and Machine Learning using Tensor Voting*. PhD thesis, Electrical Engineering, University of Southern California, August 2005.

[111] P. Mrazek and M. Navara. Selection of optimal stopping time for nonlinear diffusion filtering. *International Journal of Computer Vision*, 52(2-3):189–203, May 2003.

[112] H. H. Nagel. Constraints for the estimation of displacement vector fields from image sequences. In *International Joint Conferences on Artificial Intelligence*, pages 945–951, 1983.

[113] H. H. Nagel and W. Enkelmann. An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(5):565–593, 1986.

[114] H. K. Nishihara. Practical real-time imaging stereo matcher. *Optical Engineering*, 23(5):63–72, 1984.

[115] Y. Ohta and T. Kanade. Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 7:139–154, 1985.

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

[117] S. Osher and R. Fedkiw. *Level set methods and dynamic implicit surfaces*. Springer, 2003.

[118] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Recognition and Machine Intelligence*, 12(7):629–639, 1990.

[119] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: surface elements as rendering primitives. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 335–342, 2000.

[120] S. B. Pollard, J. E. W. Mayhew, and J. P. Frisby. Pmf: A stereo correspondence algorithm using a disparity gradient limit. *Perception*, 14:449–470, 1985.

[121] J.-P. Pons, R. Keriven, O. Faugeras, and G. Hermosillo. Variational stereovision and 3D scene flow estimation with statistical similarity measures. In *IEEE International Conference on Computer Vision*, pages 597–602, 2003.

[122] M. Potmesil. Generating octree models of 3D objects from their silhouettes in a sequence of images. *Computer Vision, Graphics, and Image Processing*, 40(1):1–29, 1987.

[123] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 2 edition, 1992.

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

[125] Stevens M. R., Culbertson B., and Malzbender T. A histogram-based color consistency test for voxel coloring. In *ICPR '02: Proceedings of the 16 th International Conference on Pattern Recognition, Volume 4*, pages 118–121, 2002.

[126] P. Rander. *A Multi-Camera Method for 3D Digitization of Dynamic, Real-World Events*. PhD thesis, Robotics Institute, Carnegie Mellon University, May 1998.

[127] P. Rander, T. Kanade, and P. J. Narayanan. Virtualized reality: Constructing time-varying virtual worlds from real events. In *Proceedings of IEEE Visualization'97*, pages 277–283, 1997.

[128] E. Reinhard, M. Ashikhmin, B. Gooch, and R. Shirley. Color transfer between images. *IEEE Computer Graphics and Applications*, 21(5):34–41, 2001.

[129] M. Rioux, G. Bechthold, D. Taylor, and M. Duggan. Design of a large depth of view three-dimensional camera for robot vision. *Optical Engineering*, 26(12):1245–1250, 1987.

[130] S. Roy and I. J. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In *IEEE International Conference on Computer Vision*, pages 492–502, 1998.

[131] T. W. Ryan, R. T. Gray, and B. R. Hunt. Prediction of correlation errors in stereo-pair images. *Optical Engineering*, 19(3):312–322, 1980.

[132] H. Saito and T. Kanade. Shape reconstruction in projective grid space from large number of images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–54, June 1999.

[133] K. Sato and S. Inokuchi. Three-dimensional surface measurement by space encoding range imaging. *Journal of Robotic Systems*, 2:27–39, 1985.

[134] D. Scharstein. Matching images by comparing their gradient fields. In *International Conference on Pattern Recognition*, pages A:572–575, 1994.

[135] D. Scharstein and R. Szeliski. Stereo matching with non-linear diffusion. In *IEEE Conference on Comptuer Vision and Pattern Recognition*, pages 343–350, 1996.

[136] D. Scharstein and R. Szeliski. Stereo matching with nonlinear diffusion. *International Journal of Computer Vision*, 28(2):155–174, 1998.

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

[138] J. Schmidt, H. Niemann, and S. Vogt. Dense disparity maps in real-time with an application to augmented reality. In *the Sixth IEEE Workshop on Applications of Computer Vision*, pages 225–230, 2002.

[139] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. http://vision.middlebury.edu/mview/. Accessed in March 2007.

[140] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 519–528, 2006.

[141] S. M. Seitz and C. R. Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):151–173, November 1999.

[142] J. A. Sethian. *Level set methods and fast marching methods.* Cambridge University Press, 1999.

[143] J. Shah. A nonlinear diffusion model for discontinuous disparity and half-occlusions in stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 34–40, 1993.

[144] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, Seattle, June 1994.

[145] G. G. Slabaugh, W. B. Culbertson, T. Malzbender, M. R. Stevens, and R. W. Schafer. Methods for volumetric reconstruction of visual scenes. *International Journal of Computer Vision*, 57(3):179–199, May 2004.

[146] A. R. Smith and J. F. Blinn. Blue screen matting. In *SIGGRAPH'96 Conference Proceedings, Annual Conference Series*, pages 259–268, 1996.

[147] D. Snow, P. Viola, and R. Zabih. Exact voxel occupancy with graph cuts. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 345–352, 2000.

[148] S. Soatto, A. J. Yezzi, and H. Jin. Tales of shape and radiance in multi-view stereo. In *IEEE International Conference on Computer Vision*, pages 974–981, 2003.

[149] RSI 3D Systems & Software. http://www.rsi.gmbh.de.

[150] S. K. Srivastava and N. Ahuja. Octree generation from object silhouettes in perspective views. *Computer Vision, Graphics, and Image Process*, 49(1):68–84, 1990.

[151] R. Szeliski. Rapid octree construction from image sequences. *Computer Vision Graphics and Image Processing(CVGIP): Image Understanding*, 58(1):23–32, 1993.

[152] R. Szeliski and P. Golland. Stereo matching with transparency and matting. *International Journal of Computer Vision*, 32(1):45–61, 1999.

[153] C. Tang, G. Medioni, and M. Lee. N-dimensional tensor voting and application to epipolar geometry estimation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(8):829–844, 2001.

[154] S. Tominaga. Color coordinate conversion via neural networks. In Lindsay W. MacDonald and M. Ronnier Luo, editors, *Colour Imaging: Vision and Technology*, pages 166–178. John Wiley & Sons Ltd., 1999.

[155] W. S. Tong, C. K. Tang, P. Mordohai, and G. Medioni. First order augmentation to tensor voting for boundary inference and multiscale analysis in 3D. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(5):594–611, 2004.

[156] F. Torkamaniazar and K. E. Tait. Image recovery using the anisotropic diffusion equation. *Image Processing*, 5(11):1573–1578, November 1996.

[157] R. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE J. Robotics and Automation*, RA-3(4):323–344, 1987.

[158] R. Vaillant and O. Faugeras. Using extremal boundaries for 3-d object modeling. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(2):157–173, 1992.

[159] S. Vedula, S. Baker, and T. Kanade. Spatio-temporal view interpolation. In *Proceedings of the 13th ACM Eurographics Workshop on Rendering*, pages 65–76, 2002.

[160] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three-dimensional scene flow. In *IEEE International Conference on Computer Vision*, pages 722–729, 1999.

[161] M. J. Vrhel and H. J. Trussell. Color scanner calibration via a neural network. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 6, pages 3465–3468, 1999.

[162] J. Weickert and C. Schnorr. A theoretical framework for convex regularizers in pde-based computation of image motion. *International Journal of Computer Vision*, 45(3):245–264, December 2001.

[163] R. T. Whitaker and S. M. Pizer. Geometry-based image segmentation using anisotropic diffusion. In *MDSG94: Shape in Picture: Mathematical Description of Shape in Grey-level Images*, pages 641–650, 1994.

[164] R. Yang, M. Pollefeys, and G. Welch. Dealing with textureless regions and specular highlights: A progressive space carving scheme using a novel photo-consistency measure. In *IEEE International Conference on Computer Vision*, pages 576–584, 2003.

[165] J. Yin and J. R. Cooperstock. Color correction methods with applications to digital projection environments. *Journal of the Winter School of Computer Graphics*, 12(3):499–506, 2004.

[166] J. Yin and J. R. Cooperstock. Improving depth maps by nonlinear diffusion. In *12th International Conference on Computer Graphics, Visualization and Computer Vision, Plzen, Czech*, pages 305–311, 2004.

[167] J. Yin and J. R. Cooperstock. A new photo consistency test for voxel coloring. In *CRV '05: Proceedings of the 2nd Canadian conference on Computer and Robot Vision*, pages 566–570, 2005.

[168] Y. L. You, M. Kaveh, W.Y. Xu, and A. Tannenbaum. Analysis and design of anisotropic diffusion for image processing. In *International Conference on Image Processing, volume 2*, pages 497–501, 1994.

[169] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *European Conference on Computer Vision*, pages 151–158, 1994.

[170] L. Zhang, B. Curless, and S. M. Seitz. Rapid shape acquisition using color structured light and multi-pass dynamic programming. In *The 1st IEEE International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 24–36, June 2002.

[171] Z. Zhang. Determining the epipolar geometry and its uncertainty: A review. *International Journal of Computer Vision*, 27(2):161–195, 1998.

[172] C. Zitnick and T. Kanade. A cooperative algorithm for stereo matching and occlusion detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(7):675 – 684, July 2000.