Fall 2014

Shape from inconsistent silhouette: Reconstruction of objects in the presence of segmentation and camera calibration error

Amy Tabb
Purdue University

Graduate School ETD Form 9
(Revised 12/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Amy Tabb

Entitled

Shape from Inconsistent Silhouette: Reconstruction of Objects in the Presence of Segmentation and Camera Calibration Error

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

JOHNNY PARK
Chair

KARTHIK RAMANI

MIREILLE BOUTIN

RENFU LU

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): JOHNNY PARK

Approved by: M. R. Melloch 11-05-2014
Head of the Graduate Program Date

SHAPE FROM INCONSISTENT SILHOUETTE: RECONSTRUCTION OF OBJECTS IN THE PRESENCE OF SEGMENTATION AND CAMERA CALIBRATION ERROR

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Amy Tabb

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2014

Purdue University

West Lafayette, Indiana

ACKNOWLEDGMENTS

I would like to acknowledge the support of the Appalachian Fruit Research Station (ARS-USDA), as well as the partial support of a Pennsylvania Department of Agriculture grant "Improving the Fruit Quality of Pennsylvania Apples with Precision Pruning" (404-57, 815U 4510).

Dr. Johnny Park was key to moving the work to completion because of his comments, suggestions, and encouragement.

I would like to thank various advisors and colleagues at the Appalachian Fruit Research and the Agricultural Research Service for their help in many different aspects of the work: Mr. Larry Crim, Dr. D. Michael Glenn, Dr. Donald Peterson, Dr. Dariusz Swietlik, Dr. Tom Tworkoski, and Mr. Scott Wolford.

I wish to remember committee member Dr. Akio Kosaka, who passed away during the completion of this work. I have fond memories of his visits to Indiana; besides personal qualities which made him a great person to be around, he also had a great breadth of knowledge concerning the computer vision field.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

ABBREVIATIONS

ICP	Iterative Closest Point
HLMS-SfSPM	Hierarchical Local Minimum Search using Shape from Silhouette Probability Map heuristic
LMS	Local Minimum Search
LMS-SfSPM	Local Minimum Search using Shape from Silhouette Probability Map heuristic
SfS	Shape from Silhouette
SfSPM	Shape from Silhouette Probability Maps
SfIS	Shape from Inconsistent Silhouette
SDE	Shape Dissimilarity Error
SIE	Silhouette Inconsistency Error
SPOT	Sparse Pixel Occupancy Test
SPPT	Samped Pixel Projection Test
PBO	Psuedo-Boolean optmization
QPBO	Quadratic psuedo-Boolean optimization
RRVH	Real Robust Visual Hull
UH	Unbiased Hull
VH	Visual Hull

ABSTRACT

Tabb, Amy Ph.D., Purdue University, December 2014. Shape from Inconsistent Silhouette: Reconstruction of Objects in the Presence of Segmentation and Camera Calibration Error. Major Professor: Johnny Park.

Silhouettes are useful features to reconstruct the object shape when the object is textureless or the shape classes of objects are unknown. In this dissertation, we explore the problem of reconstructing the shape of challenging objects from silhouettes under real-world conditions such as the presence of silhouette and camera calibration error. This problem is called the Shape from Inconsistent Silhouettes problem. A psuedo-Boolean cost function is formalized for this problem, which penalizes differences between the reconstruction images and the silhouette images, and the Shape from Inconsistent Silhouette problem is cast as a psuedo-Boolean minimization problem. We propose a memory and time efficient method to find a local minimum solution to the optimization problem, including heuristics that take into account the geometric nature of the problem. Our methods are demonstrated on a variety of challenging objects including humans and large, thin objects. We also compare our methods to the state-of-the-art by generating reconstructions of synthetic objects with induced error.

We also propose a method for correcting camera calibration error given silhouettes with segmentation error. Unlike other existing methods, our method allows camera calibration error to be corrected without camera placement constraints and allows for silhouette segmentation error. This is accomplished by a modified Iterative Closest Point algorithm which minimizes the difference between an initial reconstruction and the input silhouettes. We characterize the degree of error that can be corrected with synthetic datasets with increasing error, and demonstrate the ability of the cam-

era calibration correction method in improving the reconstruction quality in several challenging real-world datasets.

1. INTRODUCTION

The reconstruction of the shape of objects without a reliable prior information is a problem encountered in diverse applications. Particularly when objects of interest are textureless, silhouette features may be used to reconstruct objects' shape when there are many cameras, accurate silhouettes, and no concavities in the object.

The work contained in this dissertation aims to use silhouette features to reconstruct various challenging classes of objects, including leafless trees. The reconstruction of leafless trees is necessary for some agricultural applications, such as robotic pruning and phenotype classification. These trees do not have concavities and since they lack distinctive texture, photo consistency approaches such as those in [1], [2], [3], [4], [5], return little useful information. See Figure 1.1 for an example of a challenging object, with a corresponding silhouette.

(a) Original image (b) Silhouette image

Fig. 1.1.: An example of a color image of a thin, textureless object, with a corresponding silhouette with segmentation error.

There has been a great amount of work on the general shape from silhouette problem, stemming from the theoretical background of the visual hull (VH) [6]. The visual hull is a type of SfS reconstruction generated by intersecting the backprojected silhouette viewing cones of N cameras [6]. The VH method can be implemented in fast algorithms for either voxel-based or polyhedral-based representations [7], [8], [9], [10], [11], [12], and allows the use of silhouettes when other features are not available or reliable.

In real-world applications, silhouette or camera calibration error is often present, and SfS approaches based on the visual hull theory are ill-equipped to deal with this error, particularly false negatives (false negatives are pixels that are marked background but represent the target object). Reconstructions of thin objects are particularly sensitive to the effects of silhouette errors, whether from silhouette extraction errors, noise, or camera calibration error, because the two-dimensional projections of thin objects may be only a few pixels wide. As a result, small errors in silhouette extraction can have large effects on the accuracy of an intersection-based reconstruction. In addition, voxel and pixel resolution settings, even when silhouettes are accurate, can produce intersection-based reconstructions that fail to reconstruct many portions of thin objects.

When a reconstruction resembles the original object, we call it a *representative reconstruction*. Given these preliminaries, a statement of our goals are: develop methods to reconstruct the shape of a variety of objects from silhouette images, when silhouette segmentation and camera calibration error are present, without camera placement restrictions. Furthermore, develop a method to correct camera calibration error in this context and improve reconstruction accuracy. We consider a solution to these goals to be the final step of a three-dimensional reconstruction system.

The set of input silhouettes may take one of two different forms. In the first, silhouette images are binary-valued. We call this problem the Shape from Inconsistent Silhouette (SfIS) problem.[1] The second is Shape from Silhouette Probability Maps

[1]What we call SfIS is the same as SfS-IS (Shape from Silhouette with Inconsistent Silhouettes) from Landabaso *et al.* in [13].

(SfSPM); instead of binary silhouettes, silhouette probability maps (SPMs) are given as input, where the probabilities that the pixels represent the object are continuously-valued. SPMs are used to avoid making early committments to one label or the other. In this dissertation, SfIS is considered a special case of the SfSPM problem, where the probabilites are restricted to binary values. We refer to a particular dataset with binary probability maps as an instance of the SfIS problem; conversely datasets with continuous probability maps are referred to as instances of the SfSPM problem.

Our approach to SfSPM is to penalize false positive and false negative SPM error equally. To that end, we give a pseudo-Boolean error function ($f : \mathbb{B}^n \mapsto \mathbb{R}$, where $\mathbb{B} = \{0, 1\}$ and $\mathbb{R}$ denotes the set of real numbers) that characterizes the match between the SPMs and the reconstruction as the pixel-by-pixel differences between the SPMs and the image of the reconstruction. This error function penalizes false positive and false negative error equally, unlike the SfS approach.

The error function is non-submodular, and to minimize a non-submodular pseudo-Boolean function is NP-Hard (unless P=NP).[2] Consequently, we focus on local minimum search methods to find representative reconstructions from SPMs, and describe a local minimum search algorithm that uses heuristics developed for SfSPM.

For camera calibration correction in SfIS, previous works either assume that the cameras are placed in a circular configuration, or that silhouettes are accurate and complete. These assumptions are not valid for the scenarios we consider with segmentation error, unrestricted camera placement, and large objects where one camera may view only a portion of the object.

We propose a camera calibration correction procedure that is not dependent on epipolar constraints. As mentioned previously, the use of epipolar constraints assumes that silhouettes are relatively accurate and reflect the complete object. Our approach is to minimize the projection error of the reconstruction and the silhouettes, using a three-step procedure. In the first stage, an initial reconstruction is estimated using a reconstruction method for SfIS, such as the local minimum search

[2]We discuss this in more depth in Section 2.2.

methods we propose. Then, the SfIS reconstruction is aligned to the input silhouettes using an Iterative Closest Point (ICP) approach. The resulting 3D-2D ICP optimization problem is non-linear, so we use the Levenberg-Marquadt method for finding an approximate solution. Then a new SfIS reconstruction is found using the updated camera calibration parameters.

Our contributions to the state-of-the-art on SfIS, SfSPM, and the reconstruction of challenging objects are as follows:

1. The formulation of the SfSPM (and by extension, SfIS) problem as a pseudo-Boolean optimization problem where false negative and false positive error is equally weighted.

2. Introduce local minimum search algorithms of pseudo-Boolean optimization to the SfSPM problem and show how heuristics developed for SfSPM allow for lower values of the error function to be found.

3. A method for correcting camera calibration error in a SfIS context with partial silhouettes and general camera placement. The method allows two different scenarios: correcting only external parameters or both internal and external parameters.

4. A description of a 2D-3D Iterative Closest Point algorithm.

5. Reconstruction methods that produce representative reconstructions of a variety of challenging objects in the presence of silhouette extraction and camera calibration error.

This dissertation is organized as follows. In Chapter 2, we discuss related work for the SfIS problem and pseudo-Boolean optimization and in Chapter 3, we discuss related work for correcting camera calibration in the SfIS context. We describe our formulation of the silhouette inconsistency function, SIE, in Section 4.1, and then describe how a local minima can be found in Section 4.2. The camera calibration correction method is explained in Chapter 5. We show and discuss results from the

reconstruction methods in Chapter 6. Results from the camera calibration correction method are shown in Chapter 7. We offer conclusions in Chapter 8.

The local minimum search method in this dissertation has been previously published in [14].

2. RELATED WORK IN SFIS, SFSPM, AND PSEUDO-BOOLEAN OPTIMIZATION

This chapter summarizes the recent work on reconstruction from silhouette. We discuss visual hull computation, when segmentation and calibration error is assumed to be zero in Section 2.1.1, as well as reconstruction methods when segmentation and calibration error is assumed to be present in Section 2.1.2. We also discuss the topic of pseudo-Boolean optimization in Section 2.2 as we use its principles in our method for SfSPM and SfIS reconstruction.

2.1 Reconstruction from Silhouetttes

2.1.1 Shape from Silhouette

As mentioned previously, the traditional SfS approach intersected backprojected silhouette cones in order to produce the visual hull. We will briefly summarize the two main approaches.

The first attempts to solve the SfS problem were volumetric methods. Volumetric methods divide up the region of the object into discrete units, or voxels. A voxel is marked as occupied if the voxel projects inside all silhouttes, and outside otherwise. Various methods were devised for dividing the space into voxels and testing a voxel for inclusion in the visual hull [10], [10], [7], [11], and [12].

The surface method approach estimated surface characteristics such as curvature using parameterizations of the epipolar geometry, as in [15], [16], [17], [18], [19], [20], and [21]. Lazebnik *et al.* [22], [23], [24] and [25], computed the visual hull in terms of its intrinsic characteristics: frontier points, intersection curves, and viewing edges, with incidence information between these entities. Exact Polyhedral Visual Hulls,

or EPVH [8], [9], computed a polyhedral representation of the exact visual hull. Numerical instability problems with frontier points were avoided by assuming that with finite resolution cameras, intersection curves will not cross at one point.

2.1.2 Shape from Inconsistent Silhouette and Shape from Silhouette Probability Maps

Recent works on SfIS and SfSPM have sought to compensate for the problems of VH approaches by delaying decisions about a voxel's label until more information about the voxel can be gained. All of these works use a voxel-based representation for the reconstruction. They can be divided into three main categories: sensor fusion, probabilistic, and minimization of silhouette inconsistency approaches.

In the sensor fusion approach, observations are represented using sensor models, and then the model information is fused to determine voxel occupancy probabilities. Franco and Boyer [26] use a forward sensor model for each pixel, which incoporated various elements of real camera systems, such as visibility, camera calibration noise, and sensor reliability. All observations are used to jointly infer the voxel occupancy probabilites. Díaz-Más *et al.* [27] considered camera pairs as sensors, and by incoporating geometric relationships between the camera pairs and voxels, produced a confidence model for each sensor. Sensor uncertainity models are fused using Dempster-Shafer theory to give voxel occupancy probabilities. Guan *et al.* in [28] dealt with the problem of detecting static occluding objects by using a Bayesian sensor formulation for voxels, and later, fusing the information to determine the location of occluders.

The next category considers probabilistic methods. Chueng *et al.* [29] proposed a projection test called SPOT, or Sparse Pixel Occupancy Test, which attempts to increase speed and reduce the effects of segmentation noise for SfS in the human detection application context. In SPOT, the entire voxel is projected to the image plane, and a pre-determined number n of pixels are selected from the projected voxel

region to be used for classifying the voxel. The minimum number n^* of foreground pixels that must project to a voxel in each view is determined by exhaustingly evaluating the error function representing the probability of false labeling, for each possible $n^* \in [1, n]$. Landabaso *et al.* [13] extended the ideas of [29] concerning projection tests with the Sampled Pixel Projection Test, which like Cheung's method uses an exhaustive search for the optimal n^*, though a different error function is used. Assuming that the voxel labels are independent, and that the probabilities of silhouette error are known, Landabaso *et al.* also construct and minimize an error function that represents the probability of voxel misclassification; they refer to this approach as the Unbiased Hull.

The following methods do not fit into the existing categories but should be mentioned in the context of SfIS. In Snow *et al.* [30], a background model and test images are used to generate the linear terms of an energy function composed of Boolean variables. A pair-wise smoothness prior is also incorporated into the pseudo-Boolean function. Since the function is submodular, a minimum can be found with max flow/min cut methods. The approach of Haro and Pardás [31] is to minimize an approximation of the silhouette inconsistency error, where the function used depends on whether the SfIS or SfSPM problem is being considered. They consider voxels as continuously-valued variables in the range of 0 and 1, and seek a minimum by gradient descent.

Our approach is different from the recent works in that we use a closed-form, exact silhouette inconsistency error function which is identical for SfIS and SfSPM and considers voxel labels as Boolean during the minimization process, and is not dependent on the setting of parameters or thresholds.

2.2 Pseudo-Boolean Optimization

As mentioned previously, we formalize the SfIS and SfSPM problems as a pseudo-Boolean optimization (PBO) problem. In this section we review the main theoretical results of PBO and other related works.

We mention now some potentially confusing aspects of discussing PBO. Given a function $f : \mathbb{B}^n \mapsto \mathbb{R}$, where $\mathbb{B} = \{0, 1\}$ and $\mathbb{R}$ denotes the set of real numbers, the subject of finding $f(\mathbf{x})$'s minimum,

$$\min_{\mathbf{x} \in \mathbb{B}^n} f(\mathbf{x}) \tag{2.1}$$

is called *pseudo-Boolean optimization* in the optimization and operations research community. If $f(\mathbf{x})$ is in closed form, it can be represented by additions and/or subtractions of terms containing Boolean variables; this is called the *multilinear form*. For instance, the function

$$f(\mathbf{x}) = x_0 - 2x_1 + x_2 - x_0x_1 + x_0x_2 \tag{2.2}$$

is a pseudo-Boolean function in multilinear form. A particular assignment of Boolean values to each variable in $\mathbf{x}$ is called a *configuration*.

In the computer vision community, the pseudo-Boolean function is generally given within the context of the minimization of energy functions, where the energy is usually (but not always) associated with Markov Random Field theory. Suppose that every variable in the function is assigned a label from a discrete set $\mathbb{L}$. A particular assignment of labels is called a *labeling* $\mathbf{l}$ (whereas in pseudo-Boolean optimization we would say configuration). The problem then is:

$$\min_{\mathbf{l} \in \mathbb{L}^n} \big(E_{data}(\mathbf{l}) + E_{smooth}(\mathbf{l}) \big) \tag{2.3}$$

The smoothness terms, $E_{smooth}(\mathbf{l})$, also called interaction terms, typically represent relationships between two or three variables. The data terms, $E_{data}(\mathbf{l})$, represents the

cost of assigning a particular label to a variable, independent of the labels of the other variables.[1]

When the number of labels in the set, $|\mathbb{L}| = 2$, the computer vision community's energy minimization problem is equivalent to the PBO interpretation, with equivalent theoretical guarantees. Depending on the characteristics of the function, one or the other representation is more convenient. While our problem is a computer vision one, the form of our function most naturally matches that of PBO. For this reason, we will discuss computer vision community related works using the terminology of PBO.

Hammer *et al.* in [33] provided an algorithm for solving the minimization problem $\min_{\mathbf{x}\in\mathbb{B}^n} f(\mathbf{x})$. If the function f is quadratic, meaning that the greatest number of variables multiplied together in a term is 2 such as in Equation 2.2, the authors present an algorithm that is commonly referred to as QPBO, or quadratic pseudo-Boolean optimization. In general, the *degree* of a function is the number of variables in a function's largest term. For instance, if $f(\mathbf{x}) = x_0x_1x_2x_3$, the degree of f is 4.

In QPBO, a capacitated network flow graph is constructed using the coefficients of the terms in f. A maximum flow algorithm is performed on the graph to produce a minimum cut of the graph; labels are given by what side of the cut variable nodes are from the source and sink nodes.

We will now discuss the property of submodularity, because this property relates to the solvability of PBO problems. The property of submodularity is similar to convexity in optimization problems dealing with real-valued variables. For a function $g(x_0, x_1)$ of two variables, g is submodular if and only if:

$$g(0,0) + g(1,1) \leq g(0,1) + g(1,0) \tag{2.4}$$

When a pseudo-Boolean function is in multilinear form, it is easy to tell if the function is submodular. Examine all of the terms whose degree is quadratic or greater. If all of the coefficients of these terms are negative, the function is submodular. If

[1]The notation presented here is very similar to that in [32].

the coefficients are all positive, the function is supermodular, and if the coefficients are a mix of negative and positive, the function is non-submodular.

If f is submodular, then the QPBO algorithm is guaranteed to label all variables, and if the value c is the value of the minimum cut of the network flow graph, then c is also the solution to the minimization problem, $c = \min_{\mathbf{x}\in\mathbb{B}^n} f(\mathbf{x})$. If f is non-submodular, the guarantees are fewer; the minimum cut in that case is a lower bound on $\min_{\mathbf{x}\in\mathbb{B}^n} f(\mathbf{x})$. In addition, some variables may be labeled as a result of performing QPBO on non-submodular functions; these labels found by QPBO are the same labels for those variables in the global minimum. Consequently, the submodular pseudo-Boolean minimization problem is in complexity class P, since graph cut algorithms are polynomial-time algorithms. Otherwise, pseudo-Boolean minimization problems with non-submodular functions are classified as NP-Hard problems.

Excellent surveys are contained in Boros *et al.* ([34] and [35]). A review of QPBO for the computer vision community is given in [36]. Kolmogorov and Zabih presented a graph construction method for quadratic and cubic submodular functions [37]. Their method is equivalent to QPBO in the submodular quadratic case, though the number of graph nodes and edges is smaller than the graph used in QPBO.

The maximum flow/minimum cut, or graph cut, technique for solving PBO problems has been popular for various computer vision problems. When the problems are not quadratic, it is necessary to convert them to quadratic problems in order to use a graph cut approach. Rosenberg provides a technique for transforming a pseudo-Boolean function to a quadratic one in [38], but it produces non-submodular terms with high coefficients, even when the original function is submodular.[2] Freeman and Drineas [40] offer a method to transform a cubic submodular function to a quadratic submodular function. Ishikawa in [39] developed a transformation method for submodular and non-submodular pseudo-Boolean functions. Using Ishikawa's transformation, submodular functions of any degree are transformed to submodular quadratic functions. To transform higher degree positive terms in non-submodular

[2]We discovered that high positive coefficient terms result in no labels being assigned to variables when using QPBO, an obsevation also noted by Ishikawa in [39].

functions, the number of terms introduced in Ishikawa's transformation is exponential in the degree of the term; as a result, Ishikawa's approach is feasible for cubic or perhaps quartic functions, but not for very high degree functions such those that are considered in our work. For instance, the degree of one of our datasets is 975.

3. RELATED WORK IN CORRECTING CAMERA CALIBRATION IN SFS

The recent work on correcting camera calibration parameters from silhouettes can be divided into three rough groups. The first concerns the calibration of cameras from silhouettes assuming circular motion, such as that of an object on a turntable ([41], [42], [43], [44], and [45]). The second group considers general camera motion ([46], [47], [48], [49]), while the third calibrates camera networks using sequences of silhouettes, usually of humans moving in the environment ([50] and [51]).

Within these groups are differing techniques for refining camera calibration given initial calibration parameters. Most utilize epipolar constraints, and differ in the manner that frontier points are used. We will give a brief technical overview of frontier points. A more in-depth description and analysis can be found in [9] and [52], among others.

This discussion concerning frontier points assumes perfect silhouettes and camera calibration, unlike the the situation we consider in this dissertation. A frontier point is a point on the object which projects to the silhouette boundary in two cameras. For instance, consider a sphere, in Figure 3.1. The frontier point projects to a point on the silhouette boundary in each camera, where the silhouette boundary is tangent to the epipolar line from the other camera. Given that the structure of the object(s) is unknown in the SfS problem, determining the location of frontier point projections in images is difficult and depends highly on accurate silhouettes. Åström *et al.* in [46] describes generalized epipolar constraints, and uses the curvature of silhouettes to create frontier point estimates. Boyer [47] uses a pairwise cone tangency constraint; consequently the object's complete silhouette must be present in every image. Sinha *et al.* in [51] get around the problem of few epipolar tangents by using video sequences of humans moving in the environment, and is somewhat tolerant

of segmentation error. Since many different video frames are avaliable for the same camera placement, Sinha *et al.* use the points on the silhouette which are also on the convex hull of the silhouette as frontier point estimates. In Huang and Lai [42], circular motion is estimated by exploiting the homography that relates silhouettes and epipoles. Mendonça *et al.* [43] estimates circular motion using a sequential approach where epipoles are estimated last. In Furukawa *et al.* [48], a RANSAC strategy is used to estimate camera calibration parameters and frontier points for orthographic cameras. Using a different approach, Zhang *et al.* [50] use the centroids of humans moving in the environment to create correspondences, and calibration is done using a structure from motion approach. Using a calibration generated by circular motion as an initializer, Wong and Cipolla [45] use a manually-aligned initialization for more general motion. Yamazoe *et al.* [49] minimize the distance between frontier points projected onto images and the silhouettes using bundle adjustment. Finally, Zhang and Wong [44] estimate the internal and external parameters using epipolar tangents in a circular turntable sequence.

The use of epipolar constraints requiring epipolar tangencies or frontier points makes several assumptions about the characteristics of the datasets, 1) that the silhouettes are generally accurate and 2) that the silhouettes capture the whole object, meaning that the image silhouettes are not truncated or partial silhouettes of the full object. Henández *et al.* [41] developed a circular motion calibration system for silhouettes under the assumption that silhouettes may be truncated or partial, using a silhouette consistency measure. Furukawa and Ponce [53] create a more accurate and efficient reconstruction pipeline by using a hierarchical process to generate camera and reconstruction parameters; scaled-down images are used first, and as the algorithm progresses larger and larger images are used to refine parameters using a structure from motion approach.

Our work is most similar to Henández *et al.* in [41] as they use a silhouette consistency measure and allow partial silhouettes. However, in [41] the silhouettes are assumed to be relatively accurate and the motion is assumed to be circular, while

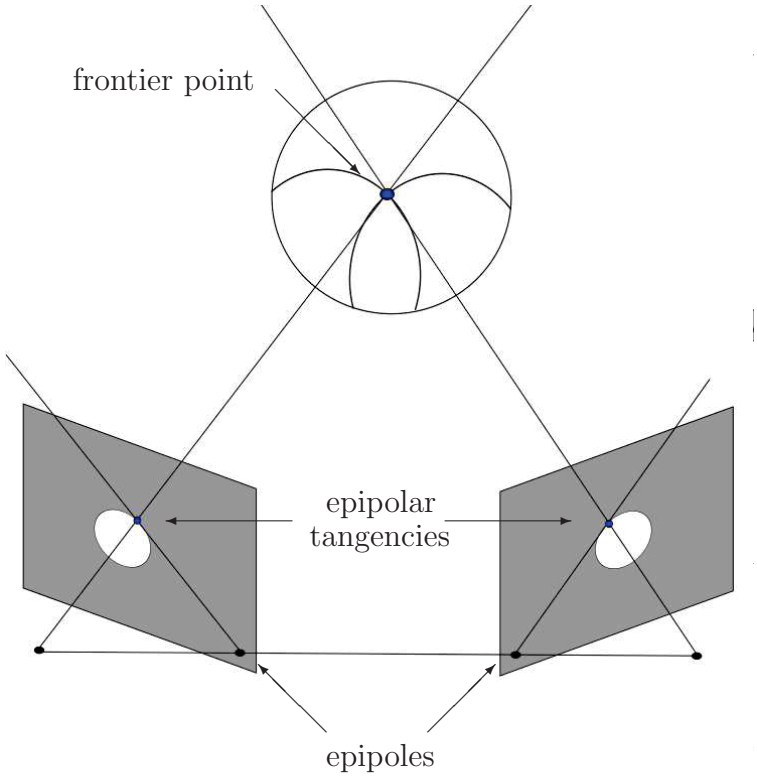


Fig. 3.1.: An illustration of the relationship between silhouettes, epipoles, and frontier points. The projection of epipolar tangencies results in a frontier point on the object. However, with an incorrect estimate of camera centers in the camera calibration problem and complicated, large objects that result in non-complete silhouettes and segmentation error, it is difficult to exploit the epipolar relationships to correct camera calibration error.

in our work we deal with general camera motion and silhouettes with segmentation error. Our method also has some similarities to Wong and Cipolla [45] in that the reconstruction image and silhouette image are aligned; however, they use a manual method to generate an initial calibration and their cost function is dependent on the presence of epipolar tangencies. Finally, the work of Furukawa and Ponce [53] is similar to ours in that the estimated reconstruction is projected to each image and matches are found, though their work estimates a new reconstruction after every round of matches, which we avoided because of the computational expense involved in the SfIS problem.

4. SHAPE FROM INCONSISTENT SILHOUETTE RECONSTRUCTIONS

In this chapter, we present our approach for generating reconstructions in the SfIS and SfSPM context. First, we explain our formulation of the SfSPM problem as a pseudo-Boolean minimization problem in Section 4.1. Next we describe our local minimum search approach in Section 4.2. We propose a time and memory efficient variation of our local minimum search method in Section 4.3.

4.1 Formulation of the Pseudo-Boolean Error Function

We represent the difference between silhouette probability maps (SPMs) and images of a reconstruction as a closed-form pseudo-Boolean function.

The pseudo-Boolean error function is formulated over a set of voxels $\mathbf{x} = \{x_0, x_1, ..., x_{n-1}\}$. A voxel is *empty* if its label is 1, and *occupied* if its label is 0. Note that this labeling is the reverse of most other literature on this subject, and is only done to make the function simpler. For instance, if we had labeled empty voxels 0 and occupied voxels 1, all of the variables (x_i) would have been negated ($\bar{x}_i$). By simply switching the labels, the formulation of the error function is more straightforward as it consists exclusively of non-negated variables.

Let an individual pixel in a SPM be p_i. r_i is the value of the reconstruction image pixel at the same location as p_i. Since the labels of occupied and empty are reversed for voxels, we also reverse the usual labeling for pixels. Consequently, the value of p_i represents the probability that p_i is viewing background. For example, if $p_i = 0$, then p_i is viewing the object with one hundrend percent probability $P(p_i = object) = 1$, and if $r_i = 0$, then r_i back-projects to the reconstruction where voxels are labeled

occupied. Pixel probabilities are binary for the SfIS problem, and continuously-valued for the SfSPM problem.

Given voxel labeling $\mathbf{x}$, the value of reconstruction pixels can be found. Let S_{p_i} be the set of voxels that are intersected by a viewing ray from pixel p_i. Then the label of reconstruction image pixel r_i is

$$r_i = \prod_{x_a \in S_{p_i}} x_a \tag{4.1}$$

In other words, $r_i = 1$, representing background, only if all voxels viewed by p_i are empty.

The Silhouette Inconsistency Error (SIE) function represents the differences between reconstruction images and silhouette probability maps. For SPMs and reconstruction image pair of pixels p_i and r_i,

$$SIE(p_i, r_i) = |p_i - r_i| \tag{4.2}$$

Before presenting the general formula for the SfSPM problem, we will first represent $|p_i - r_i|$ as a pseudo-Boolean function by considering the two cases of the SfIS problem: when p_i is 0 (a silhouette pixel) or when p_i is 1 (a non-silhouette pixel), and substituting for r_i as in Equation 4.1:

$$SIE(p_i, \mathbf{x}) = \begin{cases} \prod_{x_a \in S_{p_i}} x_a & p_i = 0, \\ 1 - \prod_{x_a \in S_{p_i}} x_a & p_i = 1 \end{cases} \tag{4.3}$$

We now represent $|p_i - r_i|$ as a pseudo-Boolean function for the general case that p_i is continuously-valued ($p_i \in [0, 1]$):

$$SIE(p_i, \mathbf{x}) = (1 - p_i) \prod_{x_a \in S_{p_i}} x_a + p_i (1 - \prod_{x_a \in S_{p_i}} x_a) \tag{4.4}$$

$$SIE(p_i, \mathbf{x}) = p_i - (1 - 2p_i) \prod_{x_a \in S_{p_i}} x_a \tag{4.5}$$

For the special case that $p_i \in \{0, 1\}$, Equation 4.5 is equivalent to Equation 4.3.

SIE for a set of input images $\mathbb{I}$ is the sum of the SIE error of the individual pixels as in Equation 4.6.

$$SIE(\mathbb{I}, \mathbf{x}) = \sum_{p_i \in \mathbb{I}} SIE(p_i, \mathbf{x}) \tag{4.6}$$

As mentioned in Chapter 1, the SIE portion of the cost function treats false positive and false negative errors equally.

4.1.1 Silhouette Inconsistency Error Treatment in Intersection-based Approaches

We will now show how traditional intersection-based approaches treat the two different types of error, with reference to our error function SIE, and applied to the SfIS problem.

First we split $SIE(\mathbb{I}, \mathbf{x})$ into two parts: the false positive error (FP) and the false negative error (FN). If a pixel p_i is part of the silhouette (0) and the reconstruction image pixel r_i is 1, then p_i is a false positive. The opposite case is a false negative. Then the false positive and false negative error is

$$FP(\mathbb{I}, \mathbf{x}) = \sum_{p_i \in \mathbb{I}, p_i = 0} SIE(p_i, \mathbf{x}) \tag{4.7}$$

$$FN(\mathbb{I}, \mathbf{x}) = \sum_{p_i \in \mathbb{I}, p_i = 1} SIE(p_i, \mathbf{x}) \tag{4.8}$$

$$SIE(\mathbb{I}, \mathbf{x}) = FP(\mathbb{I}, \mathbf{x}) + FN(\mathbb{I}, \mathbf{x}) \tag{4.9}$$

In the VH approach, the false negative error is zero: all non-silhouette pixels project to empty voxels. We can conclude that the VH approach minimizes false negative error, setting it to zero, while ignoring false positive error. We can represent the VH approach in terms of a pseudo-Boolean function as follows, where M is a very large constant, such as the number of pixels in all images.

$$SIEvh(\mathbb{I}, \mathbf{x}) = FP(\mathbb{I}, \mathbf{x}) + M \cdot FN(\mathbb{I}, \mathbf{x}) \tag{4.10}$$

The global minimum of $SIEvh$ is the VH reconstruction, where false positive and false negative errors are unequally weighted. As a result, false negative pixels have a disproportionally large impact on the VH reconstruction as compared to false positive pixels.

4.1.2 Complexity of Minimizing SIE

Provided that there is at least one pixel in the SPMs where $P(p_i = object) > P(p_i = background)$, SIE is non-submodular. Finding the global minimum of non-submodular pseudo-Boolean functions is a NP-Hard problem ([34]), as discussed in Section 2.2. Even after reduction to a quadratic pseudo-Boolean function, graph cut methods such as QPBO [33] are unable to label any voxel for SIE in Equation 4.6, even when there is a small number of voxels, such as 50, according to some of our preliminary work into this topic. For these reasons, in the next section we describe

an approximation solution to $\min_{\mathbf{x}\in\mathbb{B}^n} SIE(\mathbf{x})$ that finds a local minimum given an initial voxel labeling.

4.2 Local Minimum Search for SfIS and SfSPM

The search for a local minimum is a method borrowed from the optimization community. A more in-depth discussion can be found in [34], which is the source of our discussion on the topic.

First we begin with a definition of a local minimum for a pseudo-Boolean function f. For a labeling of voxels $\mathbf{x}$, there is a neighborhood $\mathcal{N}$ of other labelings $\mathbf{y}$, where $\mathbf{y}$ is equal to $\mathbf{x}$ except that the label of one voxel differs between the two labelings. A particular labeling $\mathbf{x}$ is called a local minimum if there are no other labelings $\mathbf{y}$ in the neighborhood of $\mathbf{x}$ that have a lower value of f than $\mathbf{x}$ does. In other words, $\mathbf{x}$ is a local minimum if and only if $f(\mathbf{x}) \leq f(\mathbf{y}) \quad \forall \mathbf{y} \in \mathcal{N}(\mathbf{x})$.

This property of local minima can be stated in terms of partial first derivatives as follows. Let $\frac{\partial f}{\partial x_i}$ be the partial first derivative of f with respect to x_i. Then, $\mathbf{x}$ is a local minimum if and only if for each voxel x_i of $\mathbf{x}$ the following is satisfied:

$$x_i = \begin{cases} 1 & \text{if } \frac{\partial f}{\partial x_i}(\mathbf{x}) \leq 0 \\ 0 & \text{if } \frac{\partial f}{\partial x_i}(\mathbf{x}) \geq 0 \end{cases} \tag{4.11}$$

In order to find a local minimum given an initial labeling $\mathbf{x}^{(0)}$, the labels of individual voxels are changed until Eq 4.11 is satisfied for all voxels. This process to find a local minimum $\mathbf{x}_{min}$ for f, when $f = SIE$, is presented in pseudocode by algorithm Local-Min-Search, Algorithm 1.

Local minimum search is similar to Iterated Conditioning Modes, or ICM, which has been used to minimize energy functions in computer vision ([54], [55], and is compared to other techniques for energy minimization in [56]). This approach of changing one label at a time is identical to what Boykov *et al.* [32] refer to as *standard moves*.

Algorithm 1 LOCAL-MIN-SEARCH($f(\cdot)$), $\mathbf{x}^{(0)}$, c)

Require: $c = f(\mathbf{x}^{(0)})$
1: $n =$ number of voxels
2: $j = 0$
3: **while** $\mathbf{x}^{(j)}$ is not a local minimum **do**
4: **for all** $i = 0$ to n **do**
5: $condition_0 = \frac{\partial f}{\partial x_i}(\mathbf{x}^{(j)}) < 0$ and $x_i^{(j)} = 0$
6: $condition_1 = \frac{\partial f}{\partial x_i}(\mathbf{x}^{(j)}) > 0$ and $x_i^{(j)} = 1$
7: **if** $condition_0$ or $condition_1$ **then**
8: $\mathbf{x}^{(j+1)} = \mathbf{x}^{(j)}$
9: $x_i^{(j+1)} = \neg x_i^{(j)}$
10: $c = c - |\frac{\partial f}{\partial x_i}(\mathbf{x}^{(j)})|$
11: $j = j + 1$
12: **end if**
13: **end for**
14: **end while**
15: **return** local minimum $\mathbf{x}^{(j)}$, c (the value of f at $\mathbf{x}^{(j)}$)

4.2.1 Local Minimum Searches for SfSPM

For any value c, where $SIE(\mathbf{x}) = c$, there are many different labelings $\mathbf{y}$ such that $SIE(\mathbf{y}) = c$. This observation is similar to that of the traditional VH theory as presented by Laurentini [6]. In the VH theory, there are many labelings of voxels that are silhouette consistent. However, the VH is chosen to be the labeling with the greatest volume. In our alteration of the local minimum search, we also specify that any local minimum $\mathbf{x}_{min}$ where $SIE(\mathbf{x}_{min}) = c$ have the greatest volume labeling out of all labelings $\mathbf{y}$ where $SIE(\mathbf{y}) = c$. We refer to this as the *greatest volume property.*

We require that the local minimum labeling $\mathbf{x}_{min}$ has the greatest volume property because of the following situation, as shown in Fig 4.1a. In that figure, Camera 0 has no silhouette error, as silhouette pixels from Camera 0 project to at least one of the three occupied voxels. Camera 1 has some false negative error from voxel v. If voxel v is removed as in Fig 4.1b, the false negative error for Camera 1 will decrease by e_1, but the false positive error for Camera 0 will increase by e_0, $e_0 > e_1$. Because of this type of deadlock, voxel v will never be removed during algorithm LOCAL-MIN-SEARCH.

Since the local minimum search is done iteratively, thin protrusions and isolated voxels in the local minimum, like those in Figure 4.1a are common, and the local minimum search would frequently stall on these types of labelings. However, by altering the local minimum search to require that local minima have the greatest volume property, it is possible to avoid getting trapped in minima with high values of SIE.

We will illustrate this process in Figure 4.1c. Here, the voxel labeling has the same value of SIE, c, as in Figure 4.1a, though Figure 4.1c represents the maximal labeling for $SIE(\mathbf{x}) = c$. Then when we test whether or not to change voxel v's label to empty, we can see in Figure 4.1d that the value of SIE decreases by e_1, since the false negative error is removed for Camera 1 and Camera 0 has neither false positive or false negative error.

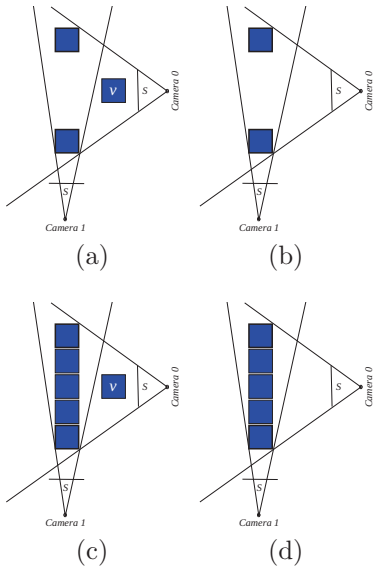


Fig. 4.1.: Example in 2D illustrating the need for an altered local minimum search for SfSPM. There are two 1D cameras, Camera 0 and Camera 1. The regions of both cameras that represent silhouette regions are denoted with an S, and the viewing rays on the boundary of S are black lines. Camera 0's entire image consists of silhouette pixels, while Camera 1 has two non-silhouette regions, one either side of a central silhouette region. Gray lines represent the boundary of Camera 1.

Algorithm 2 LOCAL-MIN-SEARCH-SFSPM($f(\cdot)$), $\mathbf{x}^{(0)}$, c)

Require: $c = f(\mathbf{x}^{(0)})$
1: $n =$ number of voxels
2: $j = 0$
3: **while** $\mathbf{x}^{(j)}$ is not a local minimum **do**
4: **for all** $i = 0$ to n **do**
5: $condition_0 = \frac{\partial f}{\partial x_i}(\mathbf{x}^{(j)}) < 0$ and $x_i^{(j)} = 0$
6: $condition_1 = \frac{\partial f}{\partial x_i}(\mathbf{x}^{(j)}) \geq 0$ and $x_i^{(j)} = 1$
7: **if** $condition_0$ or $condition_1$ **then**
8: $\mathbf{x}^{(j+1)} = \mathbf{x}^{(j)}$
9: $x_i^{(j+1)} = \neg x_i^{(j)}$
10: $c = c - |\frac{\partial f}{\partial x_i}(\mathbf{x}^{(j)})|$
11: $j = j + 1$
12: **end if**
13: **end for**
14: **end while**
15: **return** local minimum $\mathbf{x}^{(j)}$, value of f at $\mathbf{x}^{(j)}$, c

To alter LOCAL-MIN-SEARCH for SfSPM, we simply change the conditions on line 6 so that voxels with $\frac{\partial f}{\partial x_i}(\mathbf{x}^{(k)}) \geq 0$ and $x_i^{(k)} = 1$ will have their labels changed. We call this altered algorithm LOCAL-MIN-SEARCH-SFSPM and the pseudo-code is shown in Algorithm 2.

4.2.2 Implementation Details

In this section we discuss details of our implementation of LOCAL-MIN-SEARCH-SFSPM that result in reduced computational time than a naïve approach.

We introduce a way of storing the SIE function for a particular dataset that makes accessing partial first derivatives $\frac{\partial f}{\partial x_i}(\mathbf{x})$ and computing values of SIE for configurations efficient. Let the number of terms of $SIE(\mathbf{x})$ be n_t. Then the set of terms in $SIE(\mathbf{x})$ is representated as $\mathcal{T}$, and an individual term is t_j, where $j \in \{0, 1, 2, ..., n_t - 1\}$. Recall that terms refer to pixels in the SfSPM problem. Terms with positive coefficients reference pixels where the probability of being occupied is

greater than the probability of being empty, and *vice versa* for negative coefficient terms.

The naïve approach as we see it is to store the function as a set of terms, and for each term store the set of variables participating in that term. Using our example function,

$$f(\mathbf{x}) = x_0 - 2x_1 + x_2 - x_0x_1 + x_0x_2 \tag{4.12}$$

this representation would be:

$$t_0 : \{x_0\} \tag{4.13}$$

$$t_1 : \{x_1\} \tag{4.14}$$

$$t_2 : \{x_2\} \tag{4.15}$$

$$t_3 : \{x_0, x_1\} \tag{4.16}$$

$$t_4 : \{x_0, x_2\} \tag{4.17}$$

We call this the *term-variable representation*. Then, to compute the value of SIE for any configuration $\mathbf{x}$, we walk through the terms. Let $SIE = 0$. For each term, we inspect the set of variables for the term. If all of the variables of a term's set are 1 in $\mathbf{x}$, we add that term's coefficient to SIE.

The degree of our functions is large. As a result, the term-variable representation takes more memory and is less effecient than the *variable-terms representation*, which we introduce next.

In the variable-terms representation, a set of terms $\mathcal{T}_{x_i}$ is associated with each variable/voxel x_i. Every term in $\mathcal{T}_{x_i}$ contains the variable x_i. Again, using the function in Equation 4.12, and variable-terms representation is as follows:

$$x_0 : \mathcal{T}_{x_0} = \{t_0, t_3, t_4\} \tag{4.18}$$

$$x_1 : \mathcal{T}_{x_1} = \{t_1, t_3\} \tag{4.19}$$

$$x_2 : \mathcal{T}_{x_2} = \{t_2, t_4\} \tag{4.20}$$

For every term t_j, we store a variable representing number of zeros, and denoted by $zeros_j$. $zeros_j$ records the number of variables contained in term t_j that are 0 in the configuration $\mathbf{x}$. To update the number of zeros for a new configuration, we start with all $zeros_j = 0$, $j \in \{0, 1, 2, ..., n_t - 1\}$. For every variable x_i marked zero in $\mathbf{x}$, we update the $zeros$ quantity for every term in $\mathcal{T}_{x_i}$ by adding one to the previous quantity. Then, evaluation of SIE has complexity $O(t_n)$, as in Algorithm 3.

Algorithm 3 EVALUATE($f(\cdot), zeros, \mathbf{x}$)

Require: $zeros_j$ is up-to-date for $f(\mathbf{x})$ and all terms t_j.

```
1: n_t = number of terms
2: c = 0
3: for all j = 0 to n_t do
4:     if zeros_j == 0 then
5:         c = c+ coefficient(t_j)
6:     end if
7: end for
8: return c = f(x)
```

Using the variable-terms representation of $SIE(\mathbf{x})$, we can also easily compute partial first derivatives and evaluate these derivatives for a configuration. Again, we use the $zeros$ variables. Pseudocode is shown in Algorithm 4. In this algorithm, the partial first derivative of $SIE(\mathbf{x})$ with respect to x_i are computed and evaluated for configuration $\mathbf{x}$. The complexity of EVALUATE-PARTIAL-FIRST-DERIVATIVE is $O(|\mathcal{T}_{x_i}|)$.

The EVALUATE-PARTIAL-FIRST-DERIVATIVE algorithm then serves as an efficient way to calculate partial first derivatives, which is required for fast execution of the LOCAL-MIN-SEARCH-SFSPM algorithm. The $zeros$ variables can be updated

Algorithm 4 EVALUATE-PARTIAL-FIRST-DERIVATIVE($f(\cdot), zeros, \mathbf{x}, x_i$)

Require: $zeros_j$ is up-to-date for $SIE(\mathbf{x})$ and all terms t_j.

```
c = 0
for all t_j ∈ T_{x_i} do
    if zeros_j == 0 or (x_i = 0 and zeros_j == 1) then
        c = c+ coefficient(t_j)
    end if
end for
return c = ∂f/∂x_i (x)
```

every time a variable's label is changed by adding 1 (when a variable's label changes from 1 to 0) or subtracting 1 (when a variable's label changes from 0 to 1).

4.3 Hierarchical Version of Local Minimum Search

We found that with datasets with small resolution voxels, that the demand for memory to store the first derivative information and time became high. To deal with this problem, we introduce a hierarchical version of the local search algorithm.

The hierarchical version is essentially a divide-and-conquer approach for generating reconstructions, resulting in a reduced demand on computational resources. Use of the hierarchical version allows memory and processor time to be focused on regions where the occupied voxels border empty voxels. Pseudocode of the hierarchical version is shown in Algorithm 5.

Given an initial reconstruction generated with voxels of size s_0 by the LOCAL-MIN-SEARCH-SFSPM algorithm, voxels are broken into 8 new voxels under like an octtree, such that the length of one of the new voxel's sides is $\frac{s_0}{2}$. The labeling of the initial reconstruction is propogated to the labeling of the smaller voxels, where the child voxels have the same label as the parent voxels.

At the $\frac{s_0}{2}$ voxel size, we then compute projection information for a limited set of voxels. This limited set of voxels are within 8 voxels' distance from the border between occupied and empty voxels. Then the LOCAL-MIN-SEARCH-SFSPM algorithm is

Algorithm 5 HIERARCHICAL-LOCAL-MIN-SEARCH-SFSPM($f(\cdot)$), s_0, n_s)

1: Compute visual hull for voxel side size s_0; this labeling is $\mathbf{x}^{(0)}$
2: LOCAL-MIN-SEARCH-SFSPM($f(\cdot)$), $\mathbf{x}^{(0)}$, c)
3: **for all** $i = 1$ to n_s **do**
4: $s_i = s_{i-1}/2$
5: Propogate voxel labeling from size s_{i-1} to s_i; this is $\mathbf{x}^{(i)}$
6: Compute projection information for size s_i voxels marked occupied and empty voxels within 8 voxels' distance of an occupied voxel.
7: LOCAL-MIN-SEARCH($f(\cdot)$), $\mathbf{x}^{(i)}$, c) where voxels with no projection information remain empty
8: **end for**
9: **return** local minimum $\mathbf{x}^{(n_s)}$, value of f at $\mathbf{x}^{(n_s)}$, c

performed on those voxels with projection information, creating a new reconstruction for voxels of size $\frac{s_0}{2}$.

This dividing process continues until n_s divisions are performed. We will show results of HIERARCHICAL-LOCAL-MIN-SEARCH-SFSPM versus LOCAL-MIN-SEARCH-SFSPM and discuss other issues related to performance in Chapter 6.

5. CORRECTING CAMERA CALIBRATION ERROR IN SFIS

As discussed in Chapter 4, our reconstruction method searches an approximate solution to a minimization problem. While we discuss the results using local minimum searches in Chapter 6, we will say briefly here that when there is camera calibration error, the reconstruction image is usually offset from the silhouette image. This behavior is illustrated in Figure 5.1, which shows a color image of a pole and coil object, and then the reconstruction image overlaid on the silhouette image.[1]

The reconstruction image is similar to the silhouette image and the displacement between is small. Consequently, we pursued an alignment algorithm to correct the camera calibration error. In Section 5.1, we go over some preliminaries of notation. Section 5.3 gives an overview of the method using an Iterative Closest Point method for alignment, while Section 5.4 provides details about the Iterative Closest Point method for camera calibration correction.

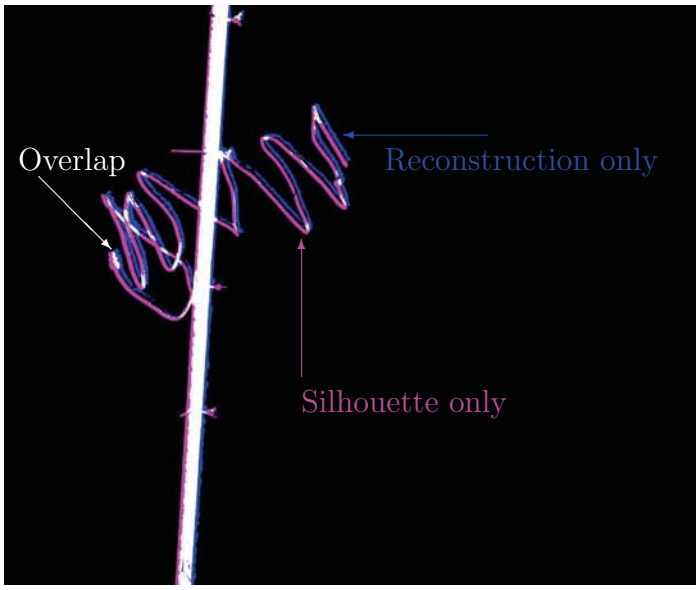
5.1 Preliminaries

Our notation for camera calibration parameters closely follows that of Hartley and Zisserman in [57].

We assume that the camera calibration parameters for n_c cameras are represented by the matrices $\mathbf{K} \in \mathbb{R}^{3\times3}, \mathbf{R} \in \mathbb{R}^{3\times3}, \mathbf{t} \in \mathbb{R}^{3\times1}$, and where the projection equations for the relationship between a three-dimensional point in homogenous coordinates $\mathbf{X} \in \mathbb{R}^{4\times1}$ and an two-dimensional image point in homogenous coordinates $\mathbf{x} \in \mathbb{R}^{3\times1}$ is:

[1]The original image is not corrected for radial distortion, which accounts for the differences in the shape of the object between the two images.

(a) Color image of a pole and coil



(b) Overlay of silhouette and reconstruction images

Fig. 5.1.: Illustration of an image of a pole and coil and an overlay of the silhouette and reconstruction images. Pixels marked magenta are from the silhouette only, pixels marked blue are from the reconstruction image only. White pixels indicate pixels where the silhouette and reconstruction image pixels overlap.

$$\mathbf{x} = \mathbf{K}[\mathbf{R}\,\mathbf{t}]\mathbf{X} \tag{5.1}$$

and $\mathbf{x} = \begin{bmatrix} x_0 \; x_1 \; x_2 \end{bmatrix}^T$, an image point in the x and y direction is the pair $(x_0/x_2, x_1/x_2)$.

$\mathbf{R}$ can be decomposed into three Euler angles, θ_x, θ_y, and θ_z, so we represent $\mathbf{R}$ as a function of three angles:

$$\mathbf{R} = R(\theta_x, \theta_y, \theta_z) \tag{5.2}$$

A parameterization of $\mathbf{R}$ is necessary to preserve the orthonormality of $\mathbf{R}$ during the Levenberg-Marquadt minimization in Section 5.4.4.

Furthermore, we assume that $\mathbf{K}$ is an upper triangular matrix of the form

$$\mathbf{K} = \begin{bmatrix} k_0 & 0 & k_2 \\ 0 & k_3 & k_4 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

However, other forms of $\mathbf{K}$ such as those described in [57] can be used as well.

Finally, we assume that the initial reconstruction can be represented by closed polyhedral meshes. Our implementation is a voxel-based technique where the voxels are cubes, so the polyhedral mesh is made up of square faces that represent the boundary between the reconstructed shape and empty space. However, other reconstruction methods whose output is or could be converted to a polyhedral mesh such as in EPVH [58] can be used with the camera calibration correction procedure we describe here.

5.2 Camera Configuration Scenarios

We present camera calibration correction procedures in two scenarios:

1. Adjust the $\mathbf{R}$ and $\mathbf{t}$ matrices for each camera, while keeping $\mathbf{K}$ constant for each camera.

2. Adjust the $\mathbf{R}$, $\mathbf{t}$, and $\mathbf{K}$ matrices for each camera.

Which scenario to use depends on the application and one's assumptions about the fidelity of $\mathbf{K}$. For instance, we have multi-camera datasets with poor camera calibration; in such a case the second scenario would be chosen to find an appropriate alignment of the reconstruction and the silhouettes. We also have a dataset where one camera is mounted on the end-effector of a robot and $\mathbf{K}$ is assumed to be accurate, in this case, we choose the first scenario. Various other scenarios are possible depending on the application, and can be derived from the these two basic scenarios and the framework we present in Sections 5.4.3.

When we describe the general method for camera calibration correction, we let the parameters be represented by a vector $\mathbf{p}$. The sizes of $\mathbf{p}$ for the first configuration is 6 (three angles for $\mathbf{R}$ and three elements of $\mathbf{t}$) and for the second is 10 (the 6 from the first scenario and 4 internal camera calibration parameters). Whatever the scenario used, we denote the matrix represention of $\mathbf{p}$ be $\mathbf{P}(\mathbf{p})$.

5.3 Method Overview

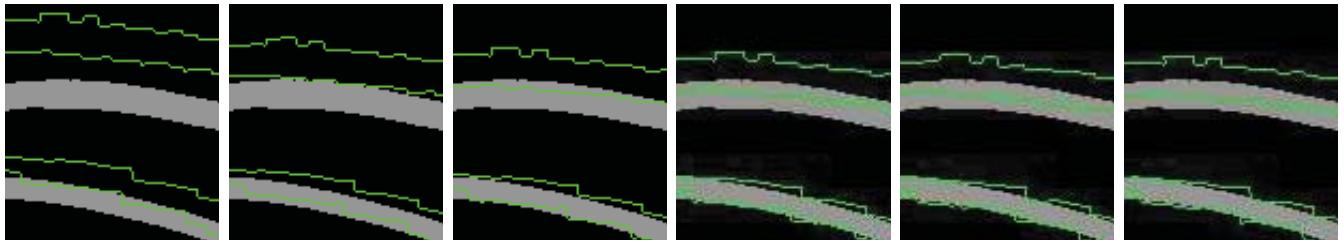
Here, we summarize the method for correcting camera calibration and relate it to the stopping criterion we use for our ICP algorithm. First, we denote a reconstruction's shape as $\mathbb{S}$. $\mathbb{S}$ consists of a set of faces of a closed polyhedral mesh as mentioned in Section 5.1.

As in Chapter 4, we use the SIE function. In this context, let the set of input silhouettes be $\mathbb{I}$, where an individual image is denoted by $\mathbf{I}$. For one image in the sequence of input silhouette images, the image of $\mathbb{S}$ is computed using the camera calibration parameters of that input image; this image is $\mathbf{I}_S$. Then the *silhouette inconsistency error* (SIE) of the reconstruction and the input image is $SIE = \sum_{\forall q} |\mathbf{I}(q) - \mathbf{I}_S(q)|$, where q is a pixel index.

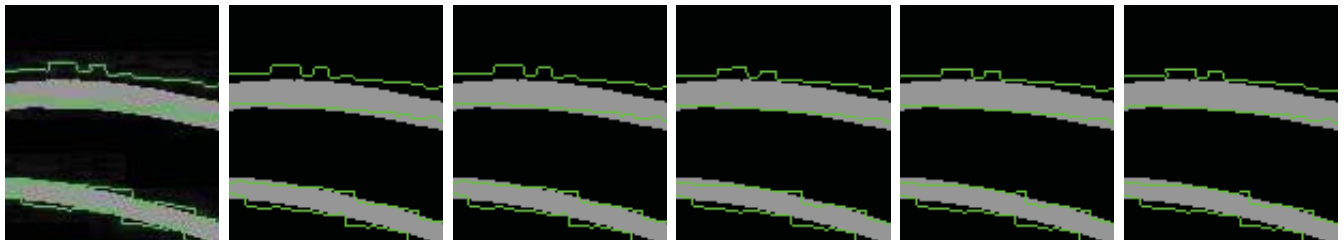
The general camera calibration correction algorithm is outlined in Algorithm 6. The SIE error is computed using the current camera calibration parameters; if some

better parameters can be found using the 2D-3D ICP algorithm then these parameters are accepted as the current parameters. We define "better" as parameters that result in a lower SIE value. This process is repeated for each image in the input silhouette sequence.

An illustration of the algorithm's progress on alignment is shown in Figure 5.2.



(a) The first six iterations of the ICP algorithm where $\mathbf{R}$ and $\mathbf{t}$ are allowed to change, $\mathbf{K}$ is kept fixed.



(b) The first six iterations of the ICP algorithm where $\mathbf{R}$, $\mathbf{t}$, and $\mathbf{K}$ are all allowed to change. This run of ICP is done after the completion of a round when only $\mathbf{R}$ and $\mathbf{t}$ are allowed to change.

Fig. 5.2.: Illustration of the progression of the camera calibration correction algorithm. Original silhouette image pixels $\mathbf{I}$ are medium gray; this silhouette boundary of the reconstruction image $\mathbf{I}_S$ is in green. The top row represents the alignment as a result of the first six iterations of the 2D-3D algorithm where $\mathbf{R}$ and $\mathbf{t}$ are adjusted. Once that process terminates as a result of the stopping criterion related to SIE, the 2D-3D algorithm is run again, the difference being that $\mathbf{R}$, $\mathbf{t}$, and $\mathbf{K}$ are adjusted. The second row represents the first six iterations of the second process, once the $\mathbf{R}$, $\mathbf{t}$ only adjustment has been completed. More details of particular experimental choices can be found in Chapter 7.

Algorithm 6 CALIBRATION-CORRECTION($\mathbf{I}, \mathbb{S}, \mathbf{p}^{(0)}$)

n is the maximum number of iterations
$\mathbf{I}_S^{(0)}$ is the image of $\mathbb{S}$ using $\mathbf{P}(\mathbf{p^{(0)}})$
$SIE^{(0)} = \sum_{\forall q} |\mathbf{I}(q) - \mathbf{I}_S^{(0)}(q)|$
$\mathbf{p}^* = \mathbf{p}^{(0)}$
for all $i = 0$ to n **do**
    Align $\mathbf{I}_S^{(i)}$ to $\mathbf{I}$ using the 2D-3D ICP; result is $\mathbf{p}^{(i+1)}$
    Compute $\mathbf{I}_S^{(i+1)}$ using $\mathbf{p^{(i+1)}}$
    $SIE^{(i+1)} = \sum_{\forall q} |\mathbf{I}(q) - \mathbf{I}_S^{(i+1)}(q)|$
    if $SIE^{(i+1)} \leq SIE^{(i)}$ **then**
        $\mathbf{p}^* = \mathbf{p}^{(i+1)}$
    else
        break;
    end if
end for
return $\mathbf{p}^*$

5.4 3D-2D ICP

This section details how we adapt the ICP algorithm, which is usually used for 2D-2D alignments or 3D-3D alignments, to the case of a 2D-3D alignment. While one option was to perform a standard 2D-2D alignment assuming a planar projective transform, and then to interpret those results as a camera calibration correction, this approach ignores the dimensionality of the original problem. There are various efficient variants of ICP available for aligning 3D meshes as discussed in [59]. We adapt the basic ICP form given in [59], which consists of a sequence of select, match, and minimize steps.

5.4.1 Selection of 3D Points

We now give more details about the projection of $\mathbb{S}$ to $\mathbf{I}_S$, and how $\mathbf{I}_S$ is used in the 2D-3D ICP algorithm.

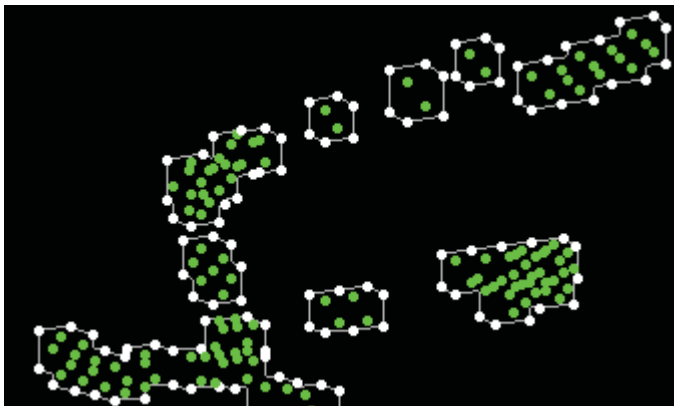


Fig. 5.3.: This figure is an illustration of how $\mathbf{I}_S$ is generated from $\mathbb{S}$. The points composing each face in $\mathbb{S}$ are projected using a current estimate of camera calibration parameters; the projected face is filled to generate a black and white image. The silhouette boundary is shown in this figure as medium gray lines, while projected points inside the boundary are shown as green circles. Points on the silhouette boundary are represented as white circles; the 3D points generating the white circles form the set $\mathbb{X}_{S_c}$ for a camera c.

Every face in $\mathbb{S}$ is made up of a sequence of three-dimensional points $\mathbf{X}$. We project each point to the camera specified by $\mathbf{P}(\mathbf{p})$, and repeat this process for all faces in $\mathbb{S}$; by filling in the convex polygon created with the sequence of projected points for each face, we can generate $\mathbf{I}_S$. From there, we determine which points $\mathbf{X}$, when projected, fall inside the silhouette of $\mathbf{I}_S$ and which fall outside. In Figure 5.3, we show the silhouette boundary of $\mathbf{I}_S$ for some large voxels as a medium gray lines, while those projected points that are on the silhouette boundary are represented by white circles, and those inside the silhouette boundary are represented as green circles.

The 3D points that we use for ICP are those points that are on the silhouette boundary of $\mathbf{I}_S$ – in other words, the points which projected produced white circles in Figure 5.3 – and we denote this set of points for camera c as $\mathbb{X}_{S_c}$. We use all of the points in $\mathbb{X}_{S_c}$ to generate matches.

5.4.2 Matching 3D points with 2D Image Coordinates

From the original image silhouettes ($\mathbf{I}$) and the silhouette of the reconstruction ($\mathbf{I}_S$) for camera c, we compute the surface normals for each pixel of the silhouette. Since we use square faces for $\mathbb{S}$, depending on the voxel size the projection of the reconstruction can have right angles and other severe changes in normal vector direction, particularly for large voxels as shown in Figure 5.3. To reduce this effect, we smooth the normals of the projected reconstruction silhouettes. Given the kth silhouette pixel in a contour, the smoothed normal is given by simple averaging, where $\mathbf{n}'_k$ is the smoothed normal at position k: $\mathbf{n}'_k = (\mathbf{n}_k + \mathbf{n}_{k-1} + \mathbf{n}_{k+1})/3$. This smoothing process is performed twice.

Given that the projection of a 3D point $\mathbf{X} \in \mathbb{X}_{S_c}$ is $\mathbf{x} = \mathbf{P}_i\mathbf{X}$, we search for the closest original image silhouette point to x where the angle between normals is less that $2\pi/3$. We represent this image silhouette point as $\phi(\mathbf{X})$.

Many ICP algorithms reject a percentage of the worst matches. Our approach to SfIS has been to assume that error exists, but not to specify the quantity of

source of that error. As a result, we are reluctant to use a pre-set threshold for rejecting matches. For instance, sometimes the matches are quite accurate, and discarding some of them according to a set percentage would result in discarding good information. On the other hand, some reconstructions $\mathbb{S}$ are quite noisy, so the reject percentage should be large. To avoid committing to a threshold for rejection ahead of time, we implemented the following scheme.

First, we perform the Levenberg-Marquadt minimization for the initial matches without rejecting any matches. If the resulting camera parameters $\mathbf{p}$ give a lower value of SIE, we accept those parameters as $\mathbf{p}^* = \mathbf{p}$. If not, we reject the worst 1% of matches and run the minimization again. This process continues until either parameters resulting in a smaller value of SIE are found, or the number of iterations is exceeded (typically set at 10 in our experiments).

5.4.3 Cost Function Formulation

Once the matches $\phi(\mathbf{X})$ are found, we seek camera calibration parameters that minimize the distance between the projections of $\mathbf{X}$ and the silhouette matched pixel $\phi(\mathbf{X})$.

$$\min_{\mathbf{p}} \sum_{i} ||\mathbf{P}(\mathbf{p})\mathbf{X}_i - \phi_i(\mathbf{X}_i)||^2 \tag{5.4}$$

We can represent Equation 5.4 as follows, where $\mathbf{P}(\mathbf{p})_1^T$ represents the first row of $\mathbf{P}(\mathbf{p})$, $\mathbf{P}(\mathbf{p})_2^T$ the second row and so on, as in Hartley and Zisserman [57], and where $\phi_i(\mathbf{X}_i)_1$ is the x component of the matching pixel to $\mathbf{X}_i$ and $\phi_i(\mathbf{X}_i)_2$ is the y component of the matching pixel to $\mathbf{X}_i$:

$$\hat{\mathbf{p}} = arg\min_{\mathbf{p}} \sum_i \left(\left(\frac{\mathbf{P}(\mathbf{p})_1^T \mathbf{X}_i}{\mathbf{P}(\mathbf{p})_3^T \mathbf{X}_i} - \phi_i(\mathbf{X}_i)_1 \right)^2 + \left(\frac{\mathbf{P}(\mathbf{p})_2^T \mathbf{X}_i}{\mathbf{P}(\mathbf{p})_3^T \mathbf{X}_i} - \phi_i(\mathbf{X}_i)_2 \right)^2 \right) \quad (5.5)$$

This is a nonlinear least squares problem; we can rearrange into the standard form with a residual vector as follows:

$$\hat{p} = arg\min_{\mathbf{p}} \sum_{j=0}^{2|\mathbb{X}_S|-1} r_j(\mathbf{P}(\mathbf{p}))^2 \quad (5.6)$$

where

$$r_{2i}(\mathbf{P}(\mathbf{p})) = \frac{\mathbf{P}(\mathbf{p})_1^T \mathbf{X}_i}{\mathbf{P}(\mathbf{p})_3^T \mathbf{X}_i} - \phi_i(\mathbf{X}_i)_1 \quad (5.7)$$

$$r_{2i+1}(\mathbf{P}(\mathbf{p})) = \frac{\mathbf{P}(\mathbf{p})_2^T \mathbf{X}_i}{\mathbf{P}(\mathbf{p})_3^T \mathbf{X}_i} - \phi_i(\mathbf{X}_i)_2 \quad (5.8)$$

for all $\mathbf{X}_i \in \mathbb{X}_S$.

While many other ICP algorithms use a weighting for each match, we instead use a constant weighting for each match.

5.4.4 Levenberg-Marquadt Modification for Newton's Method of Nonlinear Least Squares

To find an approximate solution to Equation 5.6, we use the Levenberg-Marquadt modification for nonlinear least squares. We quickly summarize the method here; more in-depth treatments can be found in optimization texts such as [60].

We compute the Jacobian $\mathbf{J}(\mathbf{P}(\mathbf{p}))$, which is a matrix of size $2|\mathbb{X}_S| \times |\mathbf{p}|$. Each element of the matrix is $J_{ij} = \frac{\partial r_i}{\partial p_j}$.

Given the Jacobian and the residual functions, the update formula for a new $\mathbf{p}^{(k+1)}$ is:

$$\mathbf{p}^{(k+1)} = \mathbf{p}^{(k)} - \Big(\mathbf{J}\big(\mathbf{P}(\mathbf{p})\big)^T \mathbf{J}\big(\mathbf{P}(\mathbf{p})\big) + \mu_k \mathbf{I}\Big)^{-1} \mathbf{J}\big(\mathbf{P}(\mathbf{p})\big)^T \mathbf{r}(\mathbf{P}(\mathbf{p})) \quad (5.9)$$

where $\mathbf{I}$ is an identity matrix of size $|\mathbf{p}| \times |\mathbf{p}|$ and $\mu_k \geq 0$ is the damping parameter and chosen according to the standard practice by starting with a small μ_k and then increasing it until the direction is a descent direction. We let $\mathbf{p}^{(0)}$ be the initial camera calibration parameters.

6. EXPEIRMENTAL RESULTS AND DISCUSSION FOR LOCAL MINIMUM SEARCH RECONSTRUCTIONS

We demonstrate the performance of our local minimum search algorithms on several datasets, synthetic and real. We compare the performance of three different local minimum search methods presented in Chapter 4: (1) Local-Min-Search, the local search algorithm without our greatest volume property heuristic; (2) Local-Min-Search-SfSPM, the local search algorithm with the heuristic, and Hierarchical-Local-Min-Search-SfSPM; and (3) a divide and conquer version of Local-Min-Search-SfSPM. We abbreviate these methods as LMS, LMS-SfSPM, and HLMS-SfSPM, respectively. Reconstruction results on synthetic and real datasets using the local minimum search methods are contained in Section 6.1 and 6.2, respectively.

In Section 6.3, we compare our local search methods with some state-of-the-art methods in SfIS. Throughout the remainder of this document, the results were generated on a workstation with one 8-core processor and 76 GB of RAM. All 3D results are visualized with Meshlab [61]. In addition, the silhouette probability maps for all of the datasets in this chapter are binary, meaning that the reconstructions in this chapter are SfIS reconstructions.

6.1 Synthetic Datasets with Known Ground Truth

We used the well-known Stanford bunny model from [62] and a model of a tree using the XFrog software[1] to generate synthetic datasets with a known ground truth. The ground truth models are converted to a voxel format so the reconstruction and the ground truth can be compared. The conversion process consists of piercing the

[1]Xfrog 3.5

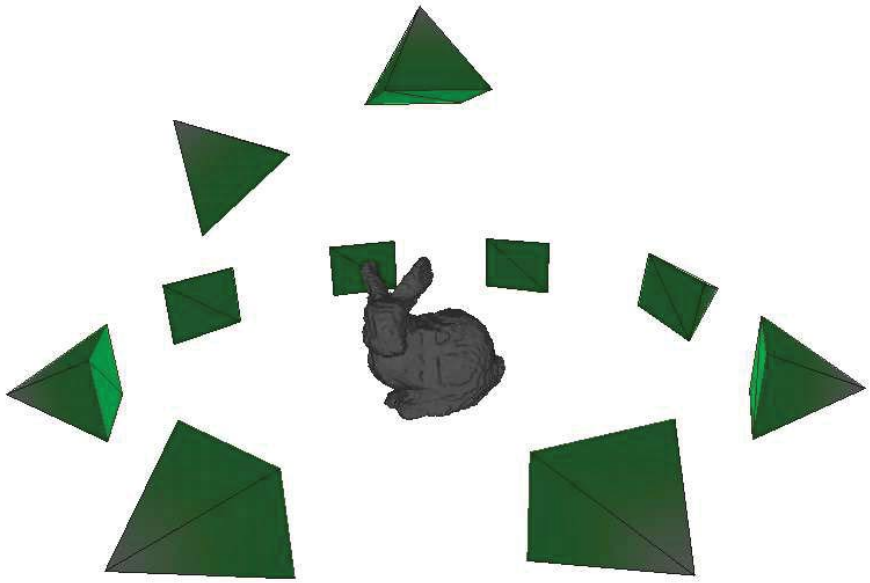


Fig. 6.1.: The camera locations for the synthetic Bunny dataset. There are ten cameras, eight cameras circling the bunny and two overhead.

ground truth model with rays and determining voxel labels based on the ray-model intersections.

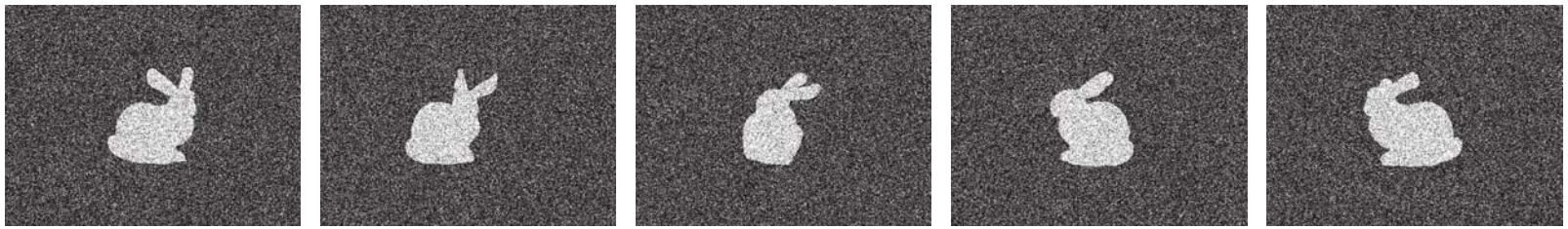
6.1.1 Bunny Datasets

The camera locations relative to the bunny figurine are shown in Figure 6.1. The voxel size was a 5 mm cube, with a search region of 1 million voxels. From the bunny model, we generated two datasets. The first dataset contains silhouettes with significant segmentation error. Most of the introduced error consisted of false negatives, though some false positives were introduced. We refer to this set as Bunny Segmentation Error and silhouette images are shown in Figure 6.2. The second dataset, Bunny Image Noise, consists of silhouettes without segmentation error but corrupted with 20 % salt and pepper noise. Silhouette images are shown in Figure 6.3.

(a) Silhouette image 0 (b) Silhouette image 1 (c) Silhouette image 2 (d) Silhouette image 3 (e) Silhouette image 4

(f) Silhouette image 5 (g) Silhouette image 6 (h) Silhouette image 7 (i) Silhouette image 8 (j) Silhouette image 9

Fig. 6.2.: Simulated segmentation error in the Bunny Segmentation Error dataset.



(a) Silhouette image 0 (b) Silhouette image 1 (c) Silhouette image 2 (d) Silhouette image 3 (e) Silhouette image 4

(f) Silhouette image 5 (g) Silhouette image 6 (h) Silhouette image 7 (i) Silhouette image 8 (j) Silhouette image 9

Fig. 6.3.: Simulated image noise (20 %) in the Bunny Image Noise dataset.

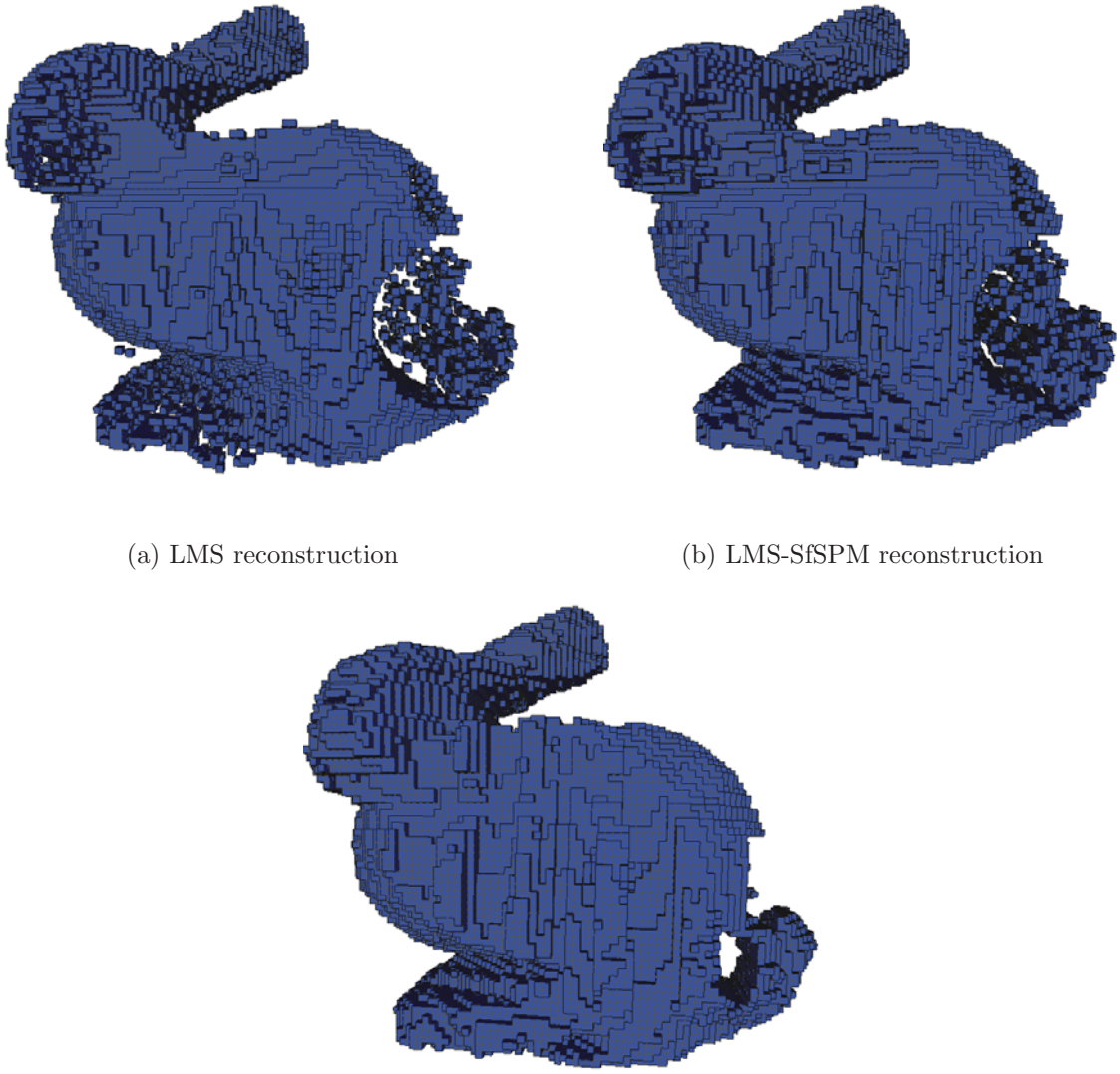
The reconstructions using our algorithm are shown in Figures 6.4 and 6.5. From the top rows down, the figures show the LMS (the local minimum search without our heuristic), LMS-SfSPM, and HLMS-SfSPM reconstructions. For the HLMS-SfSPM versions, the initial voxel size is 20 mm and the voxels are divided twice ($n_s = 2$).

Qualitatively, we can see that the reconstruction without the heuristic, LMS, contains more empty regions in the interior of the object than those reconstructions of LMS-SfSPM. Comparing HLMS-SfSPM to LMS-SfSPM, the HLMS-SfSPM reconstructions are less noisy and have fewer error as compared to the ground truth. Notice that in Figure 6.4b, the region near the rabbit's tail and rear leg is sparsely filled with voxels. In Figure 6.4c, the tail and rear leg region is recovered except for a hole. For the Bunny Image Noise dataset in Figure 6.5, the HLMS reconstruction is smaller and more similar to the original bunny's size than the LMS reconstruction.

The behavior of the LMS versus LMS-SfSPM reconstructions is explained by Figure 4.1; during the search for the local minimum, if empty voxels are not marked occupied when their first derivatives are zero, some voxels will never be maked empty on the outer edges of the object. In HLMS-SfSPM, the larger voxels result in an averaging effect over all pixels that view the voxel. Because the SfSPM heuristic is used, false positive regions are later removed or reduced in size. We show the progression of the HLMS-SfSPM method in Figure 6.6.

Table 6.1: SIE values for the local minimum search methods, Bunny Segmentation Error and Bunny Image Noise datasets.

Dataset	SIE LMS	SIE LMS-SfSPM	SIE HLMS-SfSPM
Bunny Segmentation Error	11,412	12,580	15,144
Bunny Image Noise	644,032	655,442	647,219

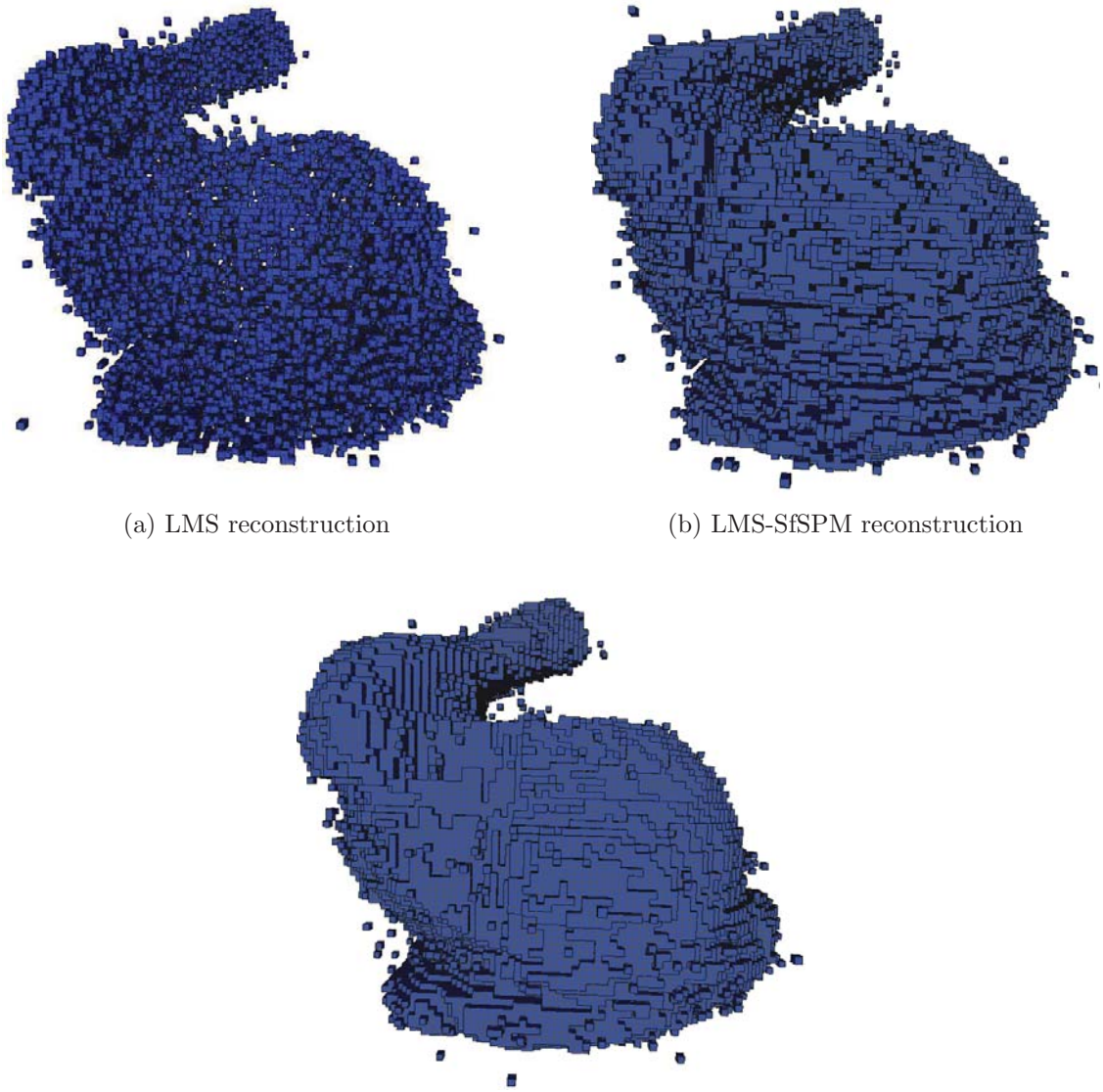


(a) LMS reconstruction

(b) LMS-SfSPM reconstruction

(c) HLMS-SfSPM reconstruction

Fig. 6.4.: Demonstration of the local minimum search methods on the Bunny Segmentation Error datasets.

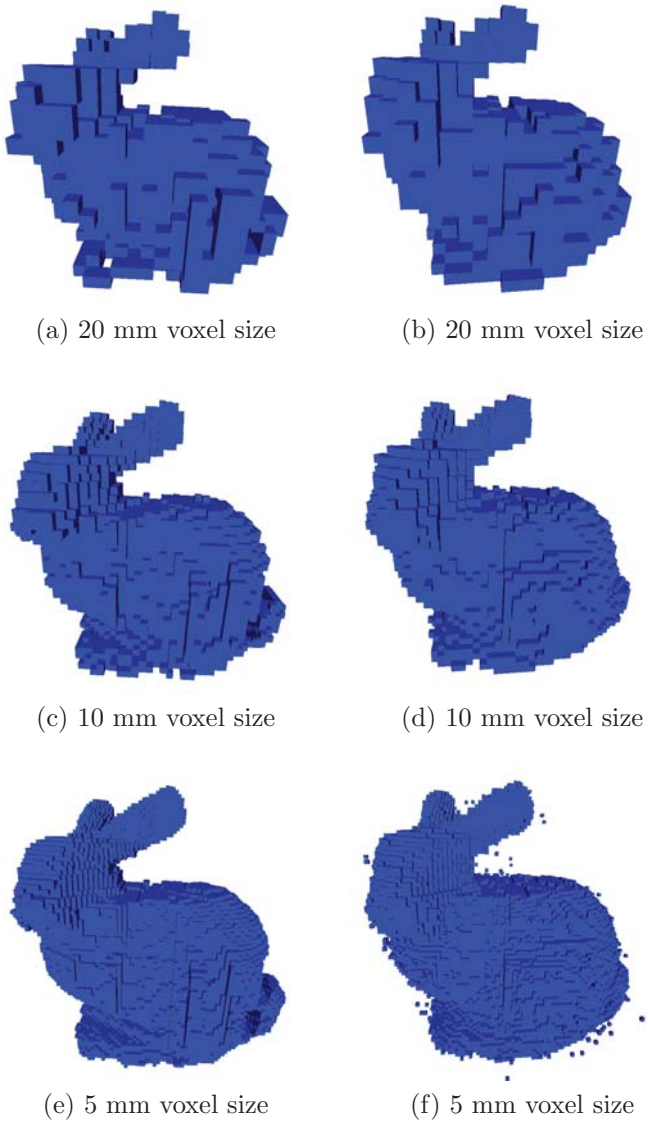


(a) LMS reconstruction

(b) LMS-SfSPM reconstruction

(c) HLMS-SfSPM reconstruction

Fig. 6.5.: Demonstration of the local minimum search methods on the Bunny Image Noise datasets.



(a) 20 mm voxel size (b) 20 mm voxel size

(c) 10 mm voxel size (d) 10 mm voxel size

(e) 5 mm voxel size (f) 5 mm voxel size

Fig. 6.6.: Progression of the HLMS-SfSPM method on the Bunny Segmentation Error and Bunny Image Noise datasets. The Bunny Segmentation Error progression is on the left while the Bunny Image Noise progression is on the right.

Table 6.2: Comparison of Local minimum search methods to the ground truth for the Bunny Segmentation Error dataset.

Method	Number Misclassifications	False Positive rate	False Negative rate	Run time
LMS	31,313	0.008972	0.2331	5.457 s
LMS-SfSPM	25,816	0.0113	0.1566	6.8 s
HLMS-SfSPM	19,415	0.0156	0.0538	7.63 s

Table 6.3: Comparison of Local minimum search methods to the ground truth for the Bunny Image Noise dataset.

Method	Number Misclassifications	False Positive rate	False Negative rate	Run time
LMS	90,177	0.005903	0.8515	9.97 s
LMS-SfSPM	42,626	0.0363	0.0994	5.25 s
HLMS-SfSPM	20,094	0.0219	0.0035	5.73 s

The values of the SIE function are shown for both of the bunny datasets in Table 6.1, and then comparisons to the ground truth and listing of running times are shown in Tables 6.2 and 6.3. From these tables, we can see that the SIE values are similar for the three local minimum search variants. However, Tables 6.2 and 6.3 show that the number of misclassified voxels of the LMS method is greater than that of the LMS-SfSPM method, sometimes significantly. For instance, for the Bunny Image Noise dataset, the LMS method's number of misclassified voxels is more than twice that of the LMS-SfSPM method. For both datasets, the HLMS-SfSPM method produces reconstructions with the lowest error as compared to the ground truth. Also, we can also see by examining the tables that the false negative rate is highest with the LMS method.

The running times for the methods do not follow a pattern; for datasets of this size and for this type of object, the random nature of the local search can influence the number of iterations of the algorithm, and consequently affect the running time.

6.1.2 Model Tree Datasets

We generated a simulated dataset composed of 32 cameras and using a tree model shown in Figure 6.7a. Camera calibration error was added to the dataset by altering one element of the translation component of each camera by $e \in \{-20, -19, ..., 19, 20\}$. The element altered ($x$, y, or z) and the value of e was determined randomly, with all possible values of e having a uniform probability of being selected, as were the three axes of $\mathbf{t}$. An example of a silhouette image from this dataset is shown in Figure 6.7, as well as the placement of cameras with respect to the model. An illustration of the differences between the ground truth calibration and camera calibration with error added is in Figure 6.8. All of the silhouette images for this set consist of partial silhouettes. The reconstruction voxel size is 5 mm. For the HLMS-SfSPM reconstruction, the intial voxel size was 20 mm and $n_s = 2$. The number of voxels is 3.168 million.

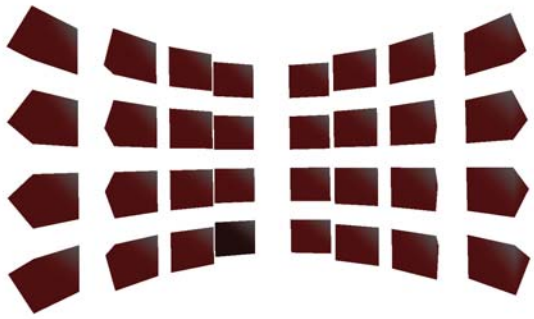
The reconstructions with the three local minimum search methods are shown with a voxel-based ground truth in Figure 6.10. The progression of the HLMS-SfSPM method is shown in Figure 6.11. Overlay of the silhouette with reconstruction images are shown in Figure 6.9.

In the Model Tree dataset, the structural differences between the methods are that the HLMS-SfSPM reconstruction has two fragmented branches, otherwise the appearance is similar. In Table 6.4, we can see that the number of misclassifications and SIE value differences between the methods is small, though the HLMS method has the lowest number of misclassiciations. From this table, we can also see that for this dataset the HLMS-SfSPM method's running time is less than the other methods, where the next fastest method takes 50 % longer to run. The reason for this is that computing and storing the voxel projection information consumes time and memory resources. While the LMS and LMS-SfSPM methods computed and stored projection information for all 3.168 million voxels in the search space, the HLMS-SfSPM variant

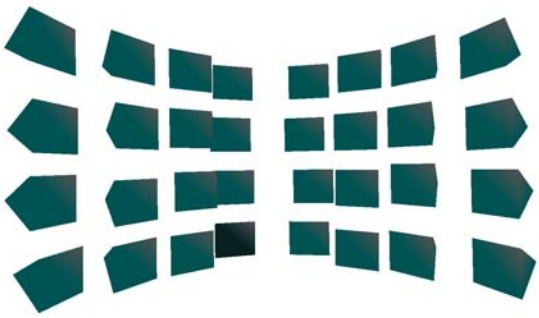
(a) (b)

(c)

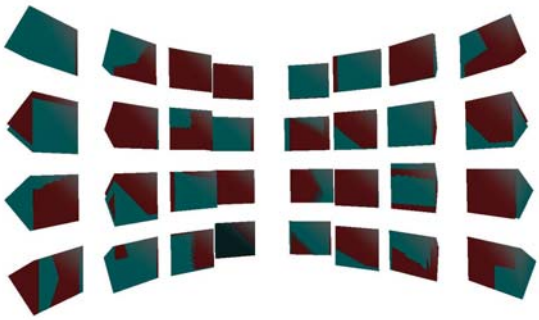
Fig. 6.7.: Illustration of the Model Tree synthetic dataset. In 6.7a, the synthetic tree used to generate images. In 6.7b, one of the 32 silhouette images of the synthetic tree. In 6.7c, the synthetic tree is shown with the 32 cameras, which are placed on two planes.



(a) Ground truth camera positions

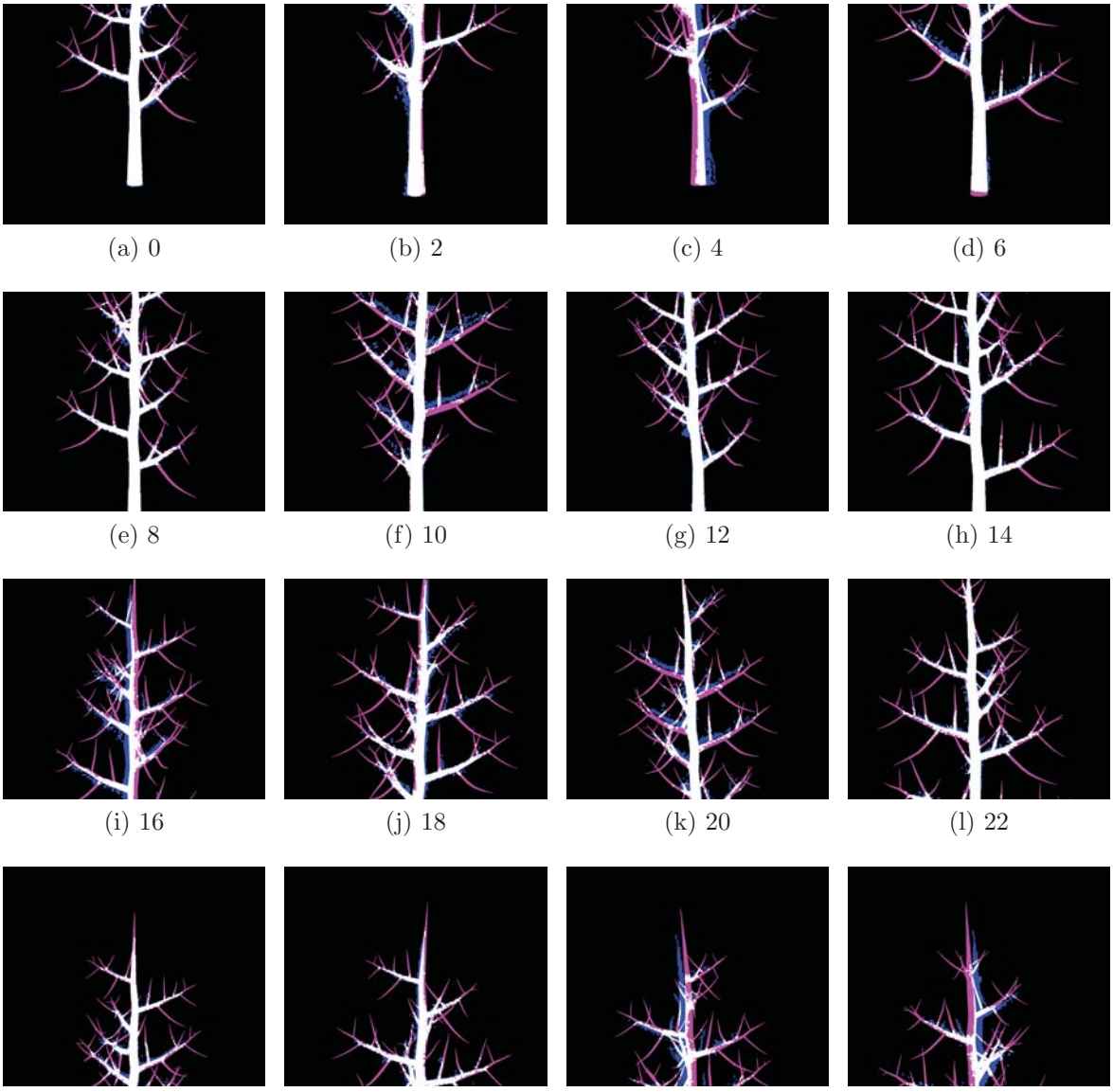


(b) Camera positions in the Model Tree dataset



(c) Juxtaposition of ground truth camera positions with Model Tree dataset camera positions

Fig. 6.8.: Illustration of the ground truth camera positions versus the camera positions from the Model Tree dataset.



(a) 0 (b) 2 (c) 4 (d) 6

(e) 8 (f) 10 (g) 12 (h) 14

(i) 16 (j) 18 (k) 20 (l) 22

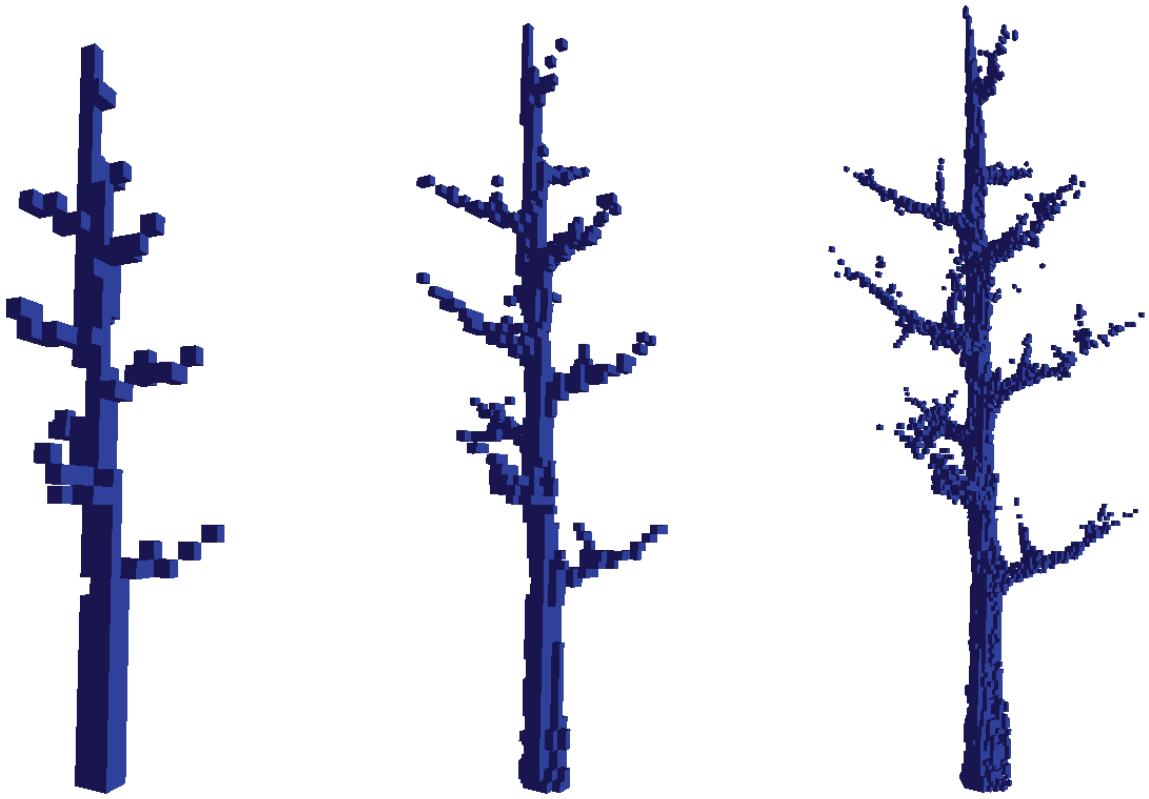
(m) 24 (n) 26 (o) 28 (p) 30

Fig. 6.9.: Silhouette and reconstruction images are overlaid for the Model Tree dataset reconstruction. Pixels marked magenta are from the silhouette only, pixels marked blue are from the reconstruction image only. White pixels indicate pixels where the silhouette and reconstruction image pixels overlap.

only required the computation and storage of projection information for 365,270 voxels at the 5 mm resolution (11.5 % of 3.168 million).

(a) Ground truth (b) LMS reconstruction (c) LMS-SfSPM reconstruction (d) HLMS-SfSPM reconstruction

Fig. 6.10.: Demonstration of the local minimum search methods on the Model Tree dataset.



(a) 20 mm resolution (b) 10 mm resolution (c) 5 mm resolution

Fig. 6.11.: Illustration of the progression of the HLMS-SfSPM method on the Model Tree dataset.

Table 6.4: Comparison of Local minimum search methods to the ground truth for the Model Tree dataset

Method	Number Misclassifications	False Positive rate	False Negative rate	SIE	Run time
LMS	5,853	0.00060636	0.482475	2,152,744	134.38 s
LMS-SfSPM	5,784	0.0006159	0.4703	2,150,544	159.289 s
HLMS-SfSPM	5,493	0.00060383	0.4393	2,152,423	90.5718 s

6.2 Real Datasets

In this section, we show reconstructions generated by the local minimum search methods on real objects. Because many works on SfIS are for the purposes of reconstructing people, our first dataset is of a human. The second and third datasets are of leafless trees.

6.2.1 Dancer Dataset

The source for the third dataset is the Dancer sequence from INRIA's 4D repository.[2] Frames corresponding to two moments in time were chosen to create the Dancer A and Dancer B datasets. Silhouette images were generated by using the provided background images and a threshold-based differencing method. Example foreground and silhouette images are shown in Figures 6.12 and 6.13. The scene consists of 8 cameras viewing the dancer. Dancer A and Dancer B have 917,700 and 1,357,800 voxels, respectively, when the voxel size is 15 mm. For the HLMS-SfSPM reconstruction, the initial voxel size is 60 mm and $n_s = 2$.

The reconstruction using the three local minimum search methods are shown in Figure 6.14, the progression of HLMS-SfSPM in Figure 6.15, and quantitive information in Table 6.5. The SIE values in Table 6.5 and visual inspection of Figure 6.14 shows that there are very few differences in between the reconstructions for these

[2] `dancer` dataset, 2007-07-01, `http://4drepository.inrialpes.fr/public/datasets`

datasets, aside from some empty voxels in the interior of the object for the LMS method. As seen from the previous dataset reconstructions displayed thus far, this is typical behavior for the LMS method. As before, the HLMS method is the fastest of the local minimum search methods.

Table 6.5: Characterisitcs of the Dancer A and Dancer B datasets for the three methods

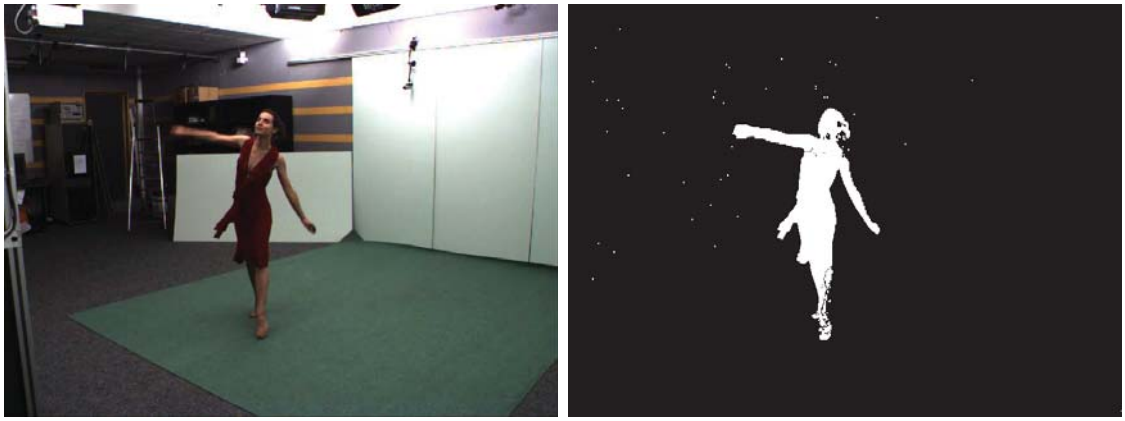
Dataset	Method	SIE	Run time
Dancer A	LMS	10,440	7.99 s
Dancer A	LMS-SfSPM	10,776	5.88 s
Dancer A	HLMS-SfSPM	11,600	5.44s
Dancer B	LMS	10,651	7.72 s
Dancer B	LMS-SfSPM	10,979	8.28 s
Dancer B	HLMS-SfSPM	11,627	5.9 s

6.2.2 Young Peach Trees

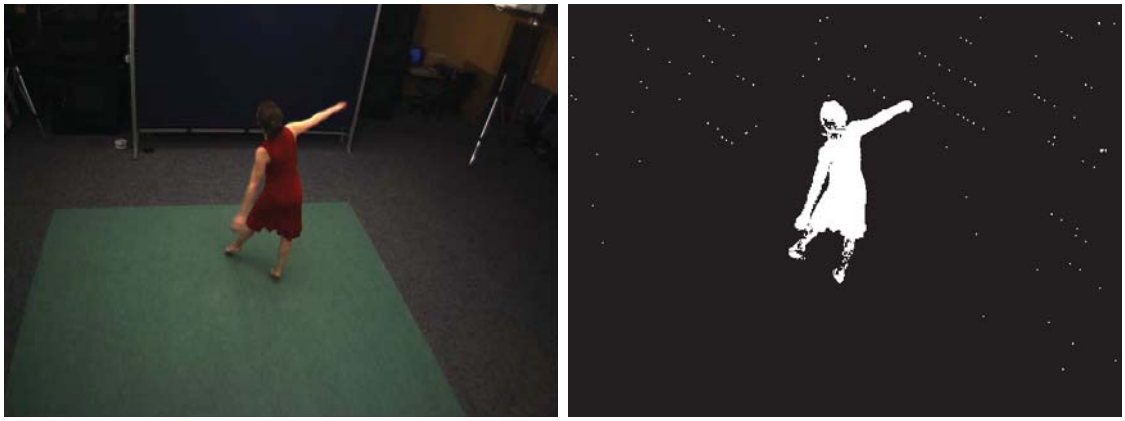
Datasets of young peach trees were acquired in the laboratory. Using a IEEE 1394 camera (640 x 480 images) mounted on the end of a robot arm, we acquired 88 images of the trees. The cameras were calibrated using a planar checkerboard pattern; the camera could view the pattern at all positions of the robot sequence so camera calibration parameters for the dataset were computed using the camera calibration method of Zhang [63].

Background images were obtained using the sequence of 88 robot positions. Silhouettes were generated using a threshold-based background differencing method. Original and silhouette images are shown for the three datasets in Figures 6.17, 6.18, and 6.19.

Three different cultivars of peach trees were used, each with a different growth habit. The cultivars are Crimson Rocket (Pillar growth habit), Bounty (Standard growth habit), and SweetNUp (Upright growth habit). The number of voxels in the peach tree datasets is large, so the HLMS-SfSPM method was the only local



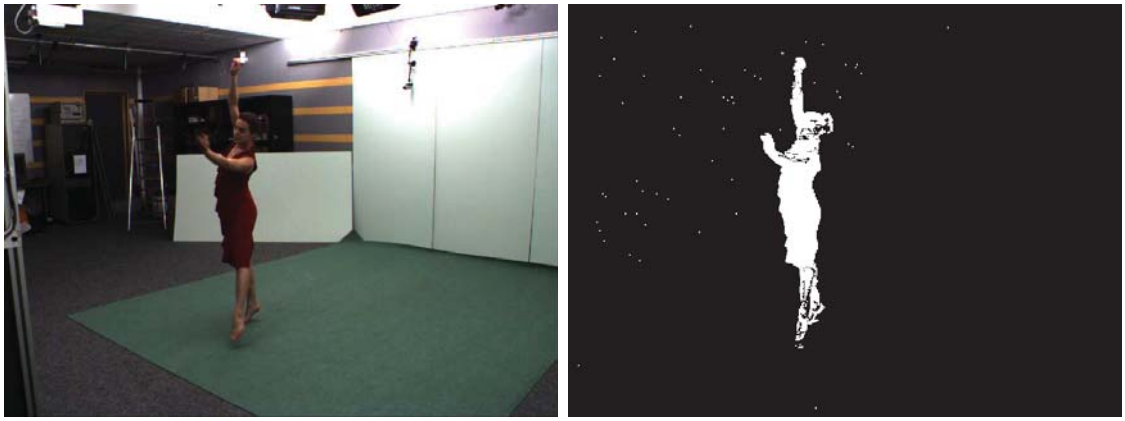
(a)



(b)

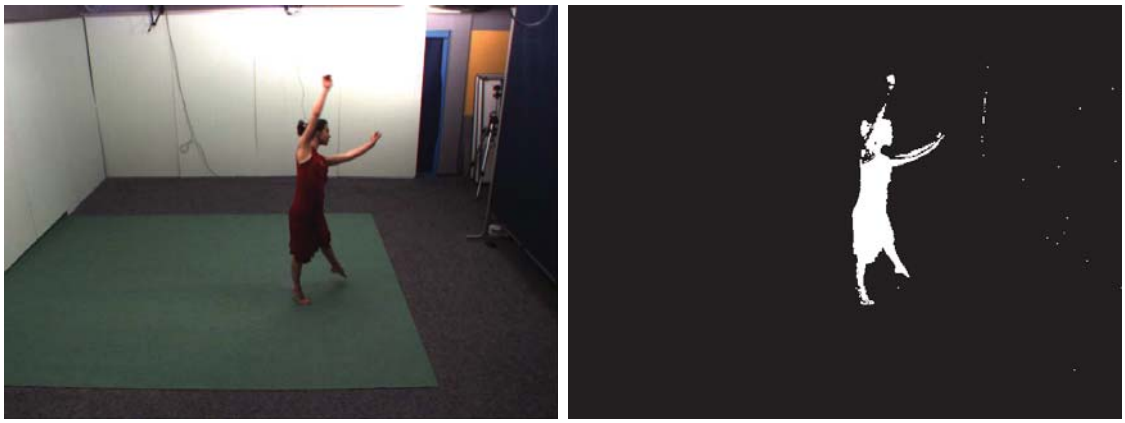
(c)

Fig. 6.12.: Input images and resulting silhouette images, with segmentation error, for Dancer A.



(a)

(b)



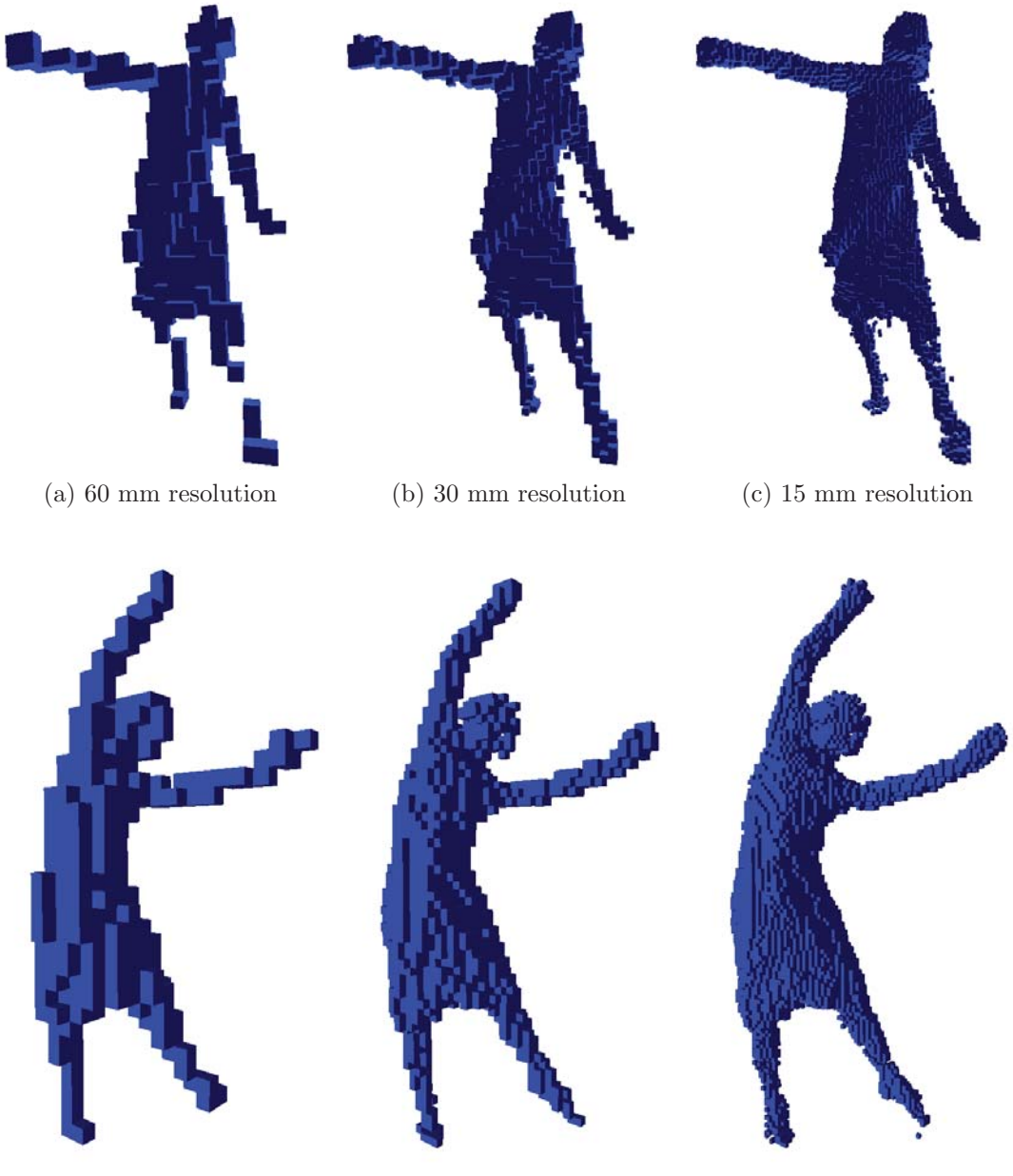
(c)

Fig. 6.13.: Input images and resulting silhouette images, with segmentation error, for Dancer B.

(a) LMS (b) LMS-SfSPM (c) HLMS-SfSPM

(d) LMS (e) LMS-SfSPM (f) HLMS-SfSPM

Fig. 6.14.: Demonstration of the local minimum search methods on the Dancer A and Dancer B datasets. The Dancer A dataset is shown on the top row, the Dancer B dataset on the bottom row.



(a) 60 mm resolution (b) 30 mm resolution (c) 15 mm resolution

(d) 60 mm resolution (e) 30 mm resolution (f) 15 mm resolution

Fig. 6.15.: Illustration of the progression of the HLMS-SfSPM method on the Dancer A and Dancer B datasets.

minimum search method that allowed us to run the reconstruction algorithms on our current hardware. For all datasets in this group, we chose the initial voxel size as 20 mm and $n_s = 3$, so the final voxel size is 2.5 mm. We list the number of voxels and running time for each tree's reconstruction using HLMS-SfSPM in Table 6.6.

Since the camera was attached to a robot arm, images were acquired from one side of the tree, as illustrated in Figure 6.16. Camera positions were created to get many different views of the objects, within the the constraints of the range of the robot arm and also to avoid colliding with the robot base.

Table 6.6: HLMS-SfSPM reconstruction characteristics

Dataset	Number voxels	Run time
Bounty	66,816,000	231.95 s
Crimson Rocket	59,392,000	179.36 s
SweetNUp	59,392,000	211.394 s

We show four views of the Bounty dataset's HLMS-SfSPM reconstruction in Figure 6.20. Some false positive regions are present near the base of the tree, which is an artefact of the camera positions used for the dataset. When viewed from the same side as the cameras such as in Figure 6.20d, the reconstruction is representative of the original tree. Other views (Figures 6.20b and 6.20c) are characterized by flat sections, which can be seen more clearly in a detail of the reconstruction, Figure 6.21. This characteristic is also a result of the positioning of the cameras all on one side and the use of silhouettes as features.

Figure 6.22 shows the progression of the HLMS-SfSPM method on the Bounty dataset. At the 20 mm resolution, large features are labeled; these are then extended and reduced in diameter in the 10 mm and 5 mm reconstructions. Differences between the 5 mm and 2.5 mm reconstructions consist of refining the branch diameter refinement and filling breaks in branches.

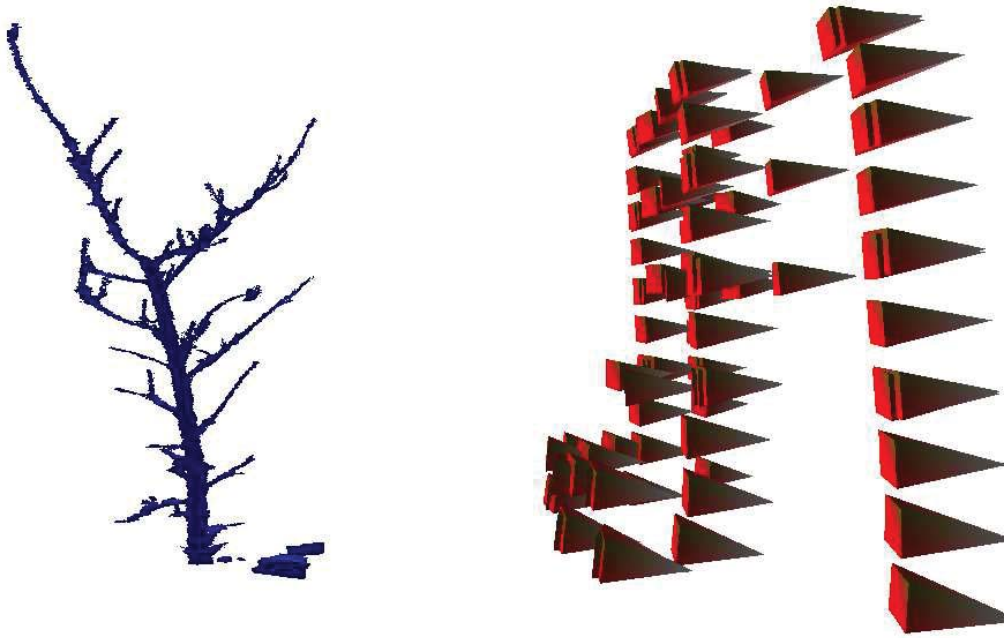


Fig. 6.16.: Placement of the peach tree relative to camera positions. The tree shown is a reconstruction the BOUNTY dataset.

The reconstruction of the Crimson Rocket dataset shown in Figure 6.23 and SweetNUp dataset shown in Figure 6.24 has many of the same characteristics of the the Bounty dataset: false positives near the base and the flat nature of the branches. The Crimson Rocket reconstruction also a false positive near the top, which is an artefact of camera placement as well.

This dataset was challenging for reconstruction because of the complicated nature of the trees, the camera placement constraints, and small voxel size, besides segmentation error. In spite of these complications, the HLMS-SfSPM method was able to generate acceptable reconstruction in reasonable time (under four minutes).

Fig. 6.17.: Input images and resulting silhouette images, with segmentation error, for BOUNTY.

Fig. 6.18.: Input images and resulting silhouette images, with segmentation error, for Crimson Rocket.

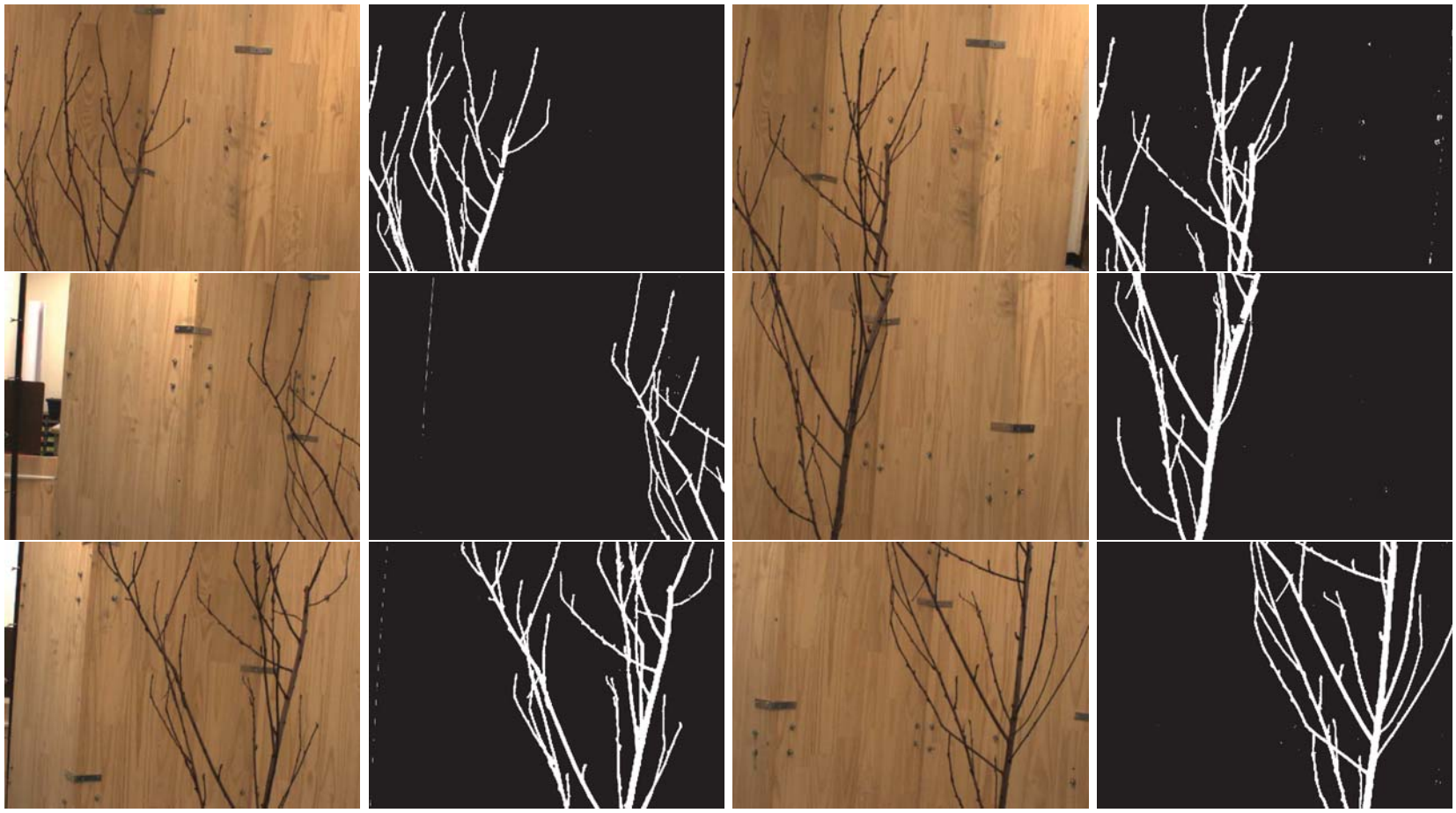
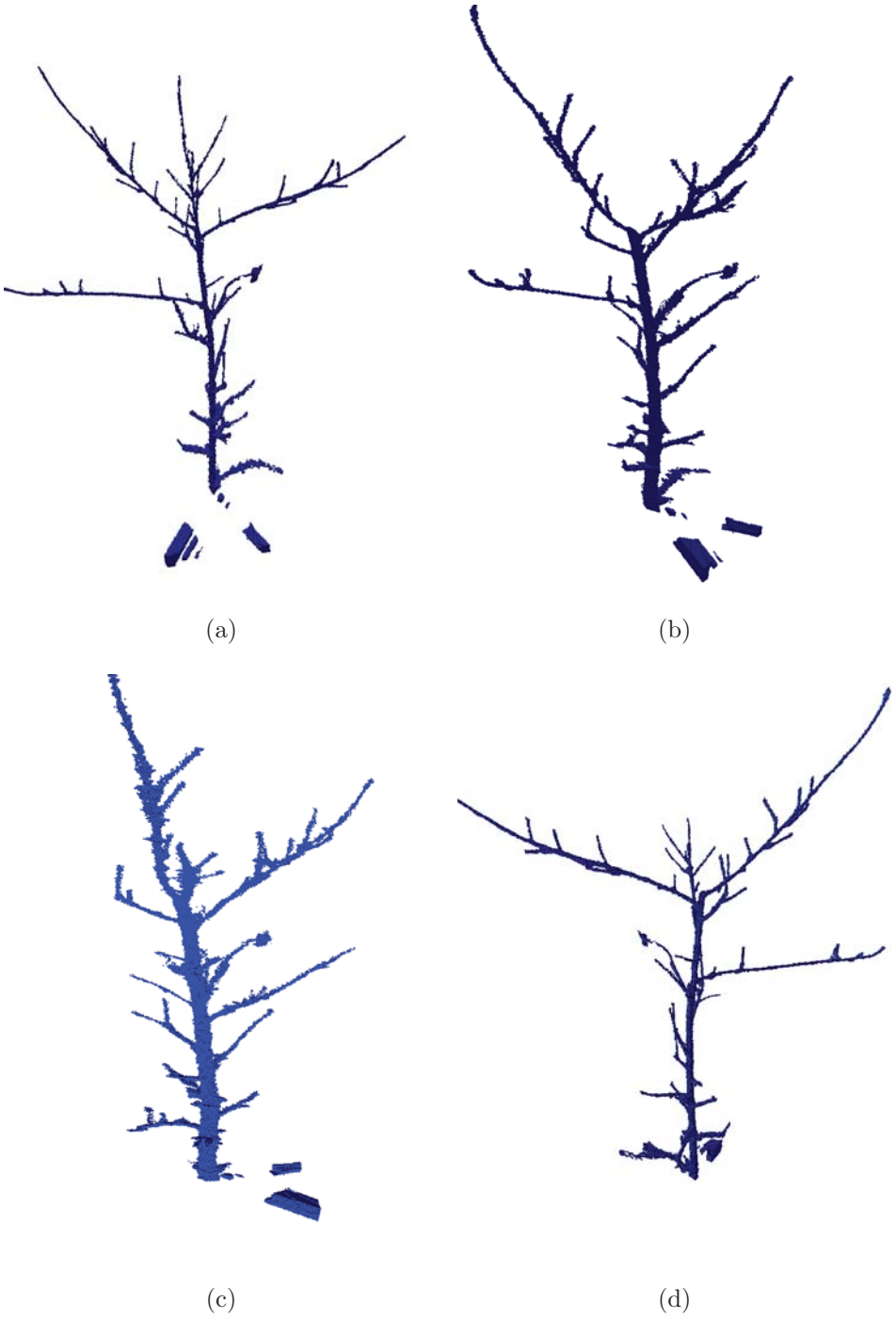


Fig. 6.19.: Input images and resulting silhouette images, with segmentation error, for SweetNUp.



(a) (b)

(c) (d)

Fig. 6.20.: HLMS-SfSPM reconstruction for the Bounty dataset; four views.

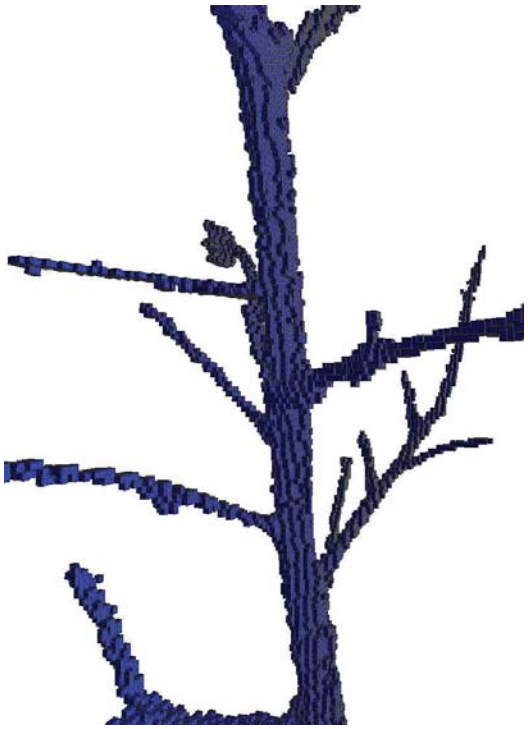
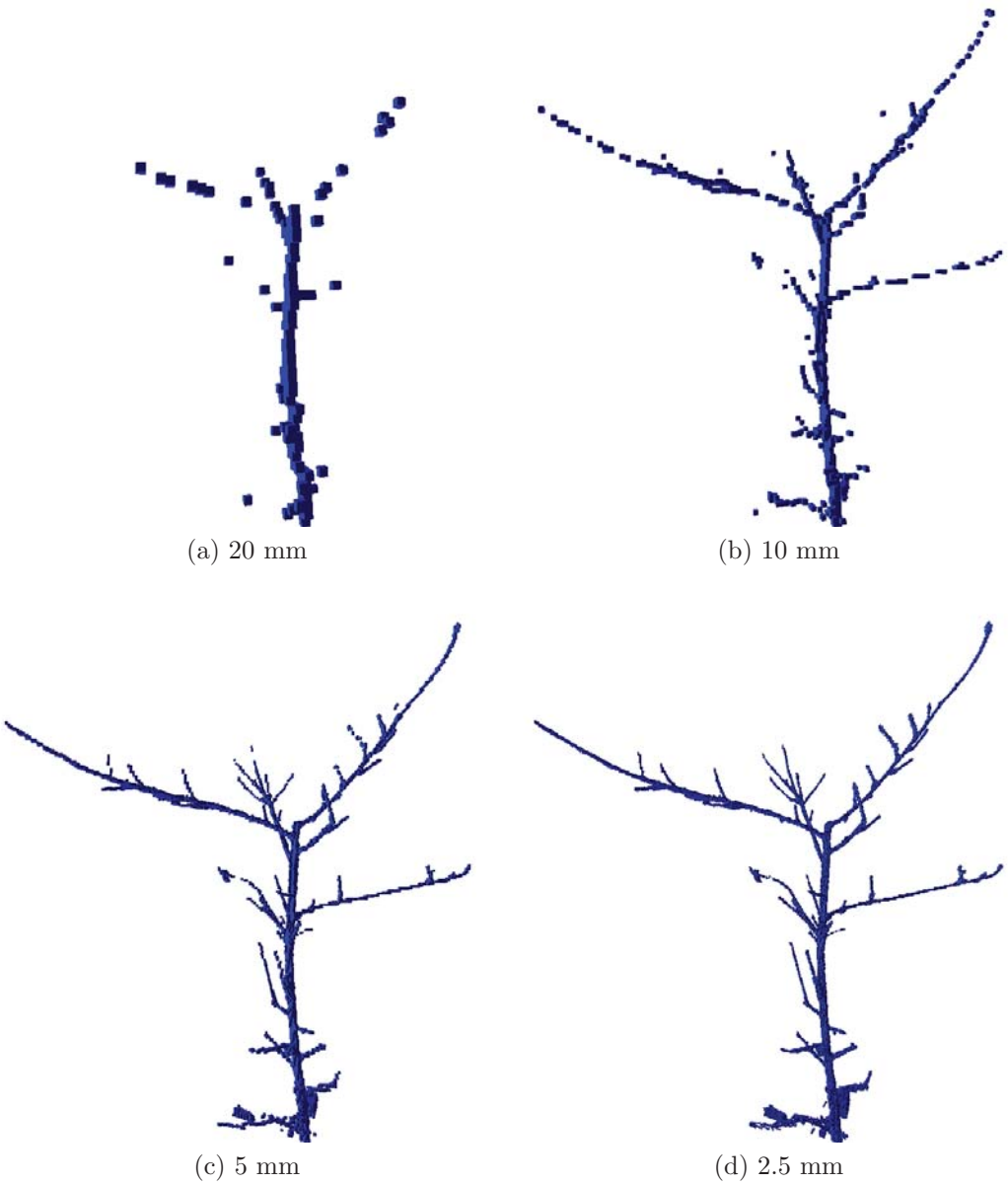


Fig. 6.21.: Detail of the BOUNTY recostruction, showing the flat branch shapes resulting from camera placement on only one side of the tree.



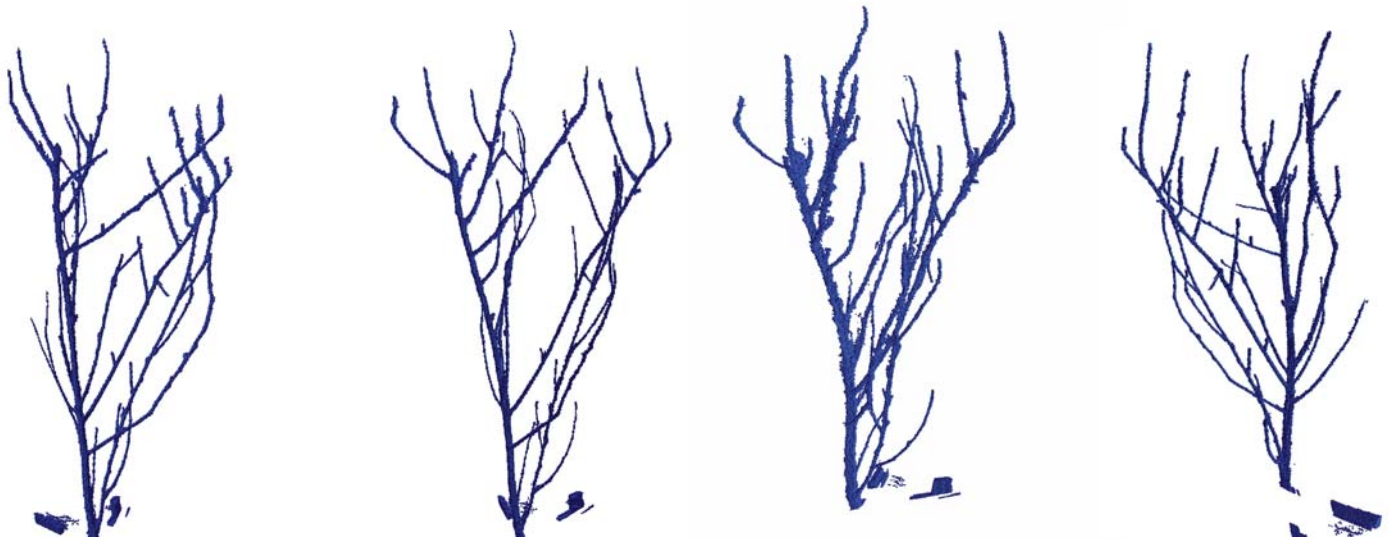
(a) 20 mm (b) 10 mm

(c) 5 mm (d) 2.5 mm

Fig. 6.22.: HLMS-SfSPM progression for the Bounty dataset

(a) (b) (c) (d)

Fig. 6.23.: HLMS-SfSPM reconstruction for the Crimson Rocket dataset; four views



(a) (b) (c) (d)

Fig. 6.24.: HLMS-SfSPM reconstruction for the SweetNUp dataset; four views

6.2.3 Large Objects Including Apple Trees

The large object datasets were acquired similarly as the young peach dataset: by a robot in a laboratory setting. A GigE camera (image size 2456 x 2058) pixels was mounted on the end effector and the robot is then moved to 38 different positions; the camera configuration is shown in Figure 6.25. The number of cameras is reduced in this set as compared to the young peach datasets because of the change of the camera. The GigE camera used in this set has a $\frac{1}{2}$ inch CCD as compared to the $\frac{1}{4}$ inch CCD camera for the peach set, and then same 8 mm lens was used. Because of this increased field of view, fewer camera positions were necessary to reconstruct the shape of the large objects. The external camera calibration parameters are estimated using a hand-eye, robot-world calibration technique.

As with the other datasets, segmentation was accomplished using a threshold-based background subtraction method. Examples of the color and silhouette images at the same position of the robot for each set are shown in Figure 6.26.

For this dataset, we only show the results for the HLMS-SfSPM version of the algorithm, for the same reasons as in the young peach dataset that the search region is too large for our machine capacity (over 124 million voxels). The parameters we use for the HLMS-SfSPM are that the initial voxel size is 12 mm, $n_s = 2$, so the final voxel size is 3 mm.

The first two large objects are apple trees without leaves. The first one we refer to as the Standard Apple. Horticulturists classify trees according to their growth habit; this tree most closely matches Lespinasse's type I classification found in [64]. The second tree we refer to as the Weeping Apple, which corresponds to a Lespinasse type IV. These trees are quite different in their growth habit, the first with long, upward-facing branches, while the second is characterized by short inter-node spacing and some downward facing tips. One may also notice that both of these trees are not conical, tending more to reside on a plane. Modern orchards are planted with a trellis and the trees are trained on a particular way such as to encourage a planar

shape. The last object is a metal pole with a copper coil attached, refered to as the Pole and Coil dataset.

We show the reconstruction generated by HLMS-SfSPM on the three datasets in Figures 6.27, 6.28, and 6.29. For the Standard Apple dataset, the 12 mm reconstruction in Figure 6.27a shows the overall shape of the tree but with many breaks in the branches. As the voxel size is decreased, the reconstruction more closely resembles the original tree. As with the young peach set, the branches are flat instead of cylindrical, as shown in Figure 6.27d. Also like the young peach set, the Standard Apple reconstruction has some false positives near the periphery of the search space resulting from the camera placement.

The Weeping Apple dataset represents a very different style of tree than Standard Apple, though the reconstruction in Figure 6.28 has similar features as the Standard Apple dataset in that progression of the LMS-SfSPM algorithm leads to more representatrive reconstructionss and branches are flat from a side view.

Since the Pole and Coil object is narrower than the trees, the HLMS-SfSPM reconstruction in Figure 6.29 is free of the large false positive regions that are in the tree reconstrudctions. We can also objserve that small details that were missed at the larger voxel resolution of 12 mm appear in the 6 mm and 3 mm resolutions, such as the end of the plastic tie that affixed the coil to the metal pole.

The running times of HLMS-SfSPM on the three datasets is shown in Table 6.7. The denser objects such as Weeping Apple take nearly four time longer to run (801 s) than the more sparse object, Pole and Coil (255 s). After the 12 mm voxel resolution, the HLMS-SfSPM expands the search region in the vicinity of the occupied/empty voxel border. Since the Weeping Apple dataset has more occupied voxels bordered by empty voxels, the search space is larger for the 6 mm and 3 mm resolutions in the Weeping Apple set versus the Pole and Coil and searching this space takes more computational time.

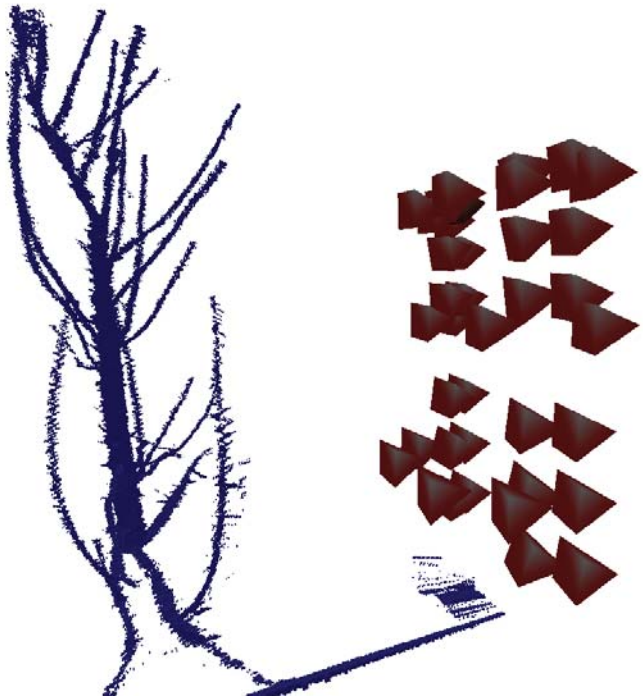
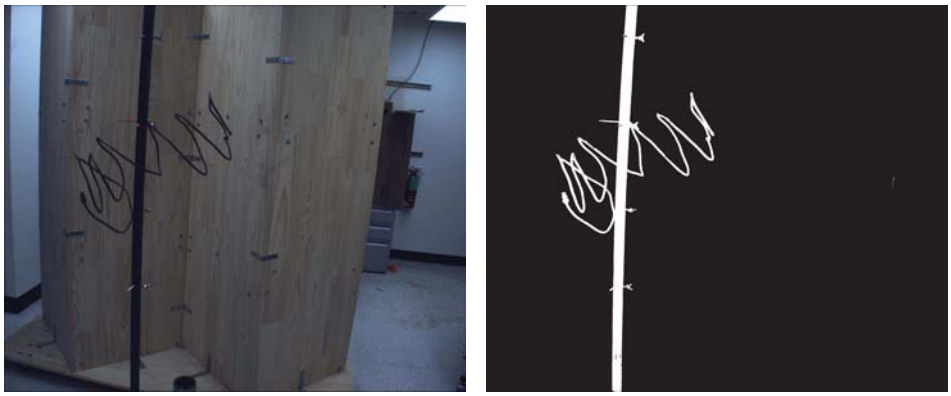


Fig. 6.25.: Placement of the large object relative to camera positions. The tree shown is a reconstruction the STANDARD APPLE dataset.

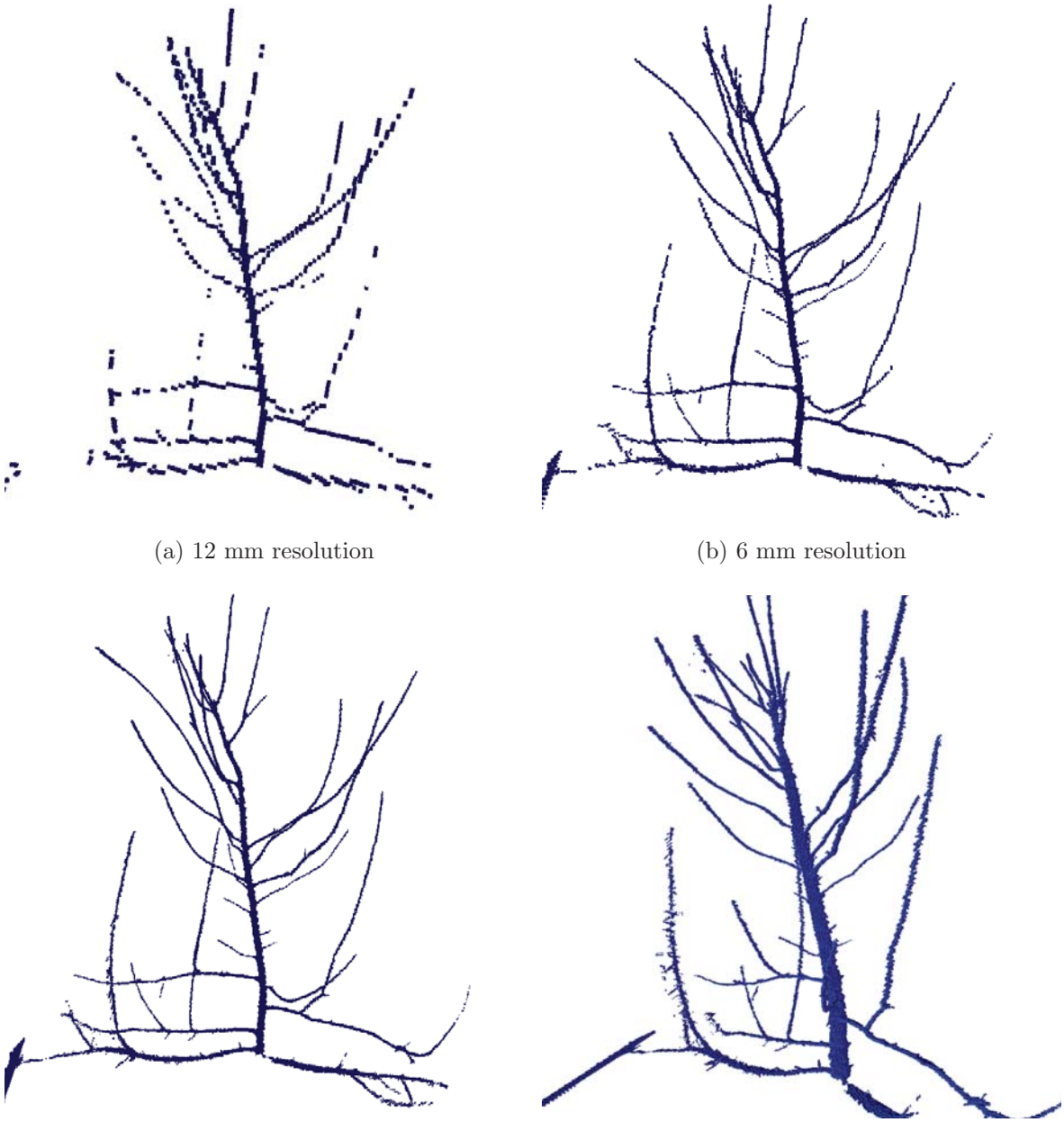
(a) (b)

(c) (d)



(e) (f)

Fig. 6.26.: Example color images and resulting silhouette images for the Standard Apple, Weeping Apple, and Pole and Coil datasets, respectively.



(a) 12 mm resolution

(b) 6 mm resolution

(c) 3 mm resolution, view 1

(d) 3 mm resolution, view 2

Fig. 6.27.: HLMS-SfSPM reconstruction for the Standard Apple dataset

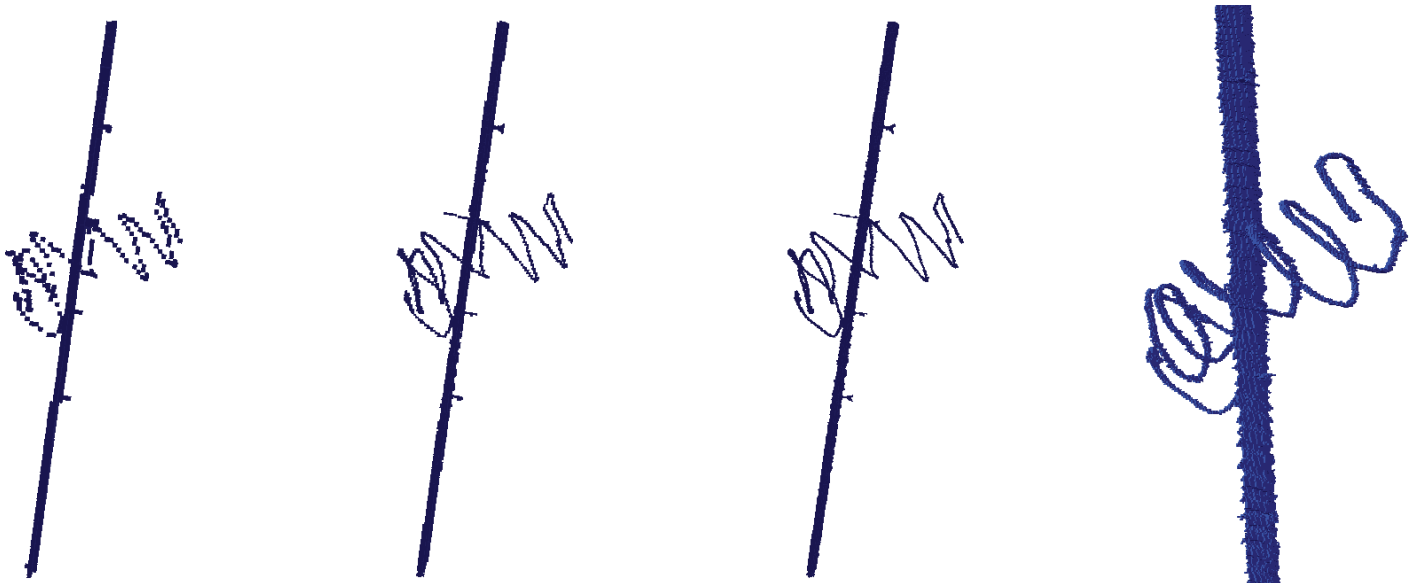
(a) 12 mm resolution

(b) 6 mm resolution

(c) 3 mm resolution, view 1

(d) 3 mm resolution, view 2

Fig. 6.28.: HLMS-SfSPM reconstruction for the Weeping Apple dataset



(a) 12 mm resolution (b) 6 mm resolution (c) 3 mm resolution, view 1 (d) 3 mm resolution, view 2

Fig. 6.29.: HLMS-SfSPM reconstruction for the Pole and Coil dataset

Table 6.7: HLMS-SfSPM reconstruction characteristics for the large objects datasets

Dataset	Number voxels	Run time
Standard Apple	124,293,600	492.015 s
Weeping Apple	124,293,600	801.048 s
Pole and Coil	124,293,600	255.776 s

6.3 Comparison with Other SfIS Methods

We compared our local minimum search methods to other methods for SfIS reconstruction. Appendix A describes the implementation of the various methods in detail. Here, we give brief descriptions of the comparison methods.

First is the Robust Visual Hull, the simplest comparison method (discussion of RRVH's origin is found in Appendix A). Whereas the VH reconstruction is the intersection of all N cameras' silhouettes, the Robust Visual Hull is the intersection of $N - k$ cameras' silhouettes, where $k < N$. Since in some situations the number of cameras that can view a voxel is not constant, we use a variation on the Robust Visual Hull that we call the Real Robust Visual Hull, or RRVH. In RRVH, we determine the ratio r of the number of cameras where a voxel is within the silhouettes versus the number of cameras that view a voxel. If r is greater than or equal to a threshold $m \in [0, 1]$, we mark the voxel as occupied and empty otherwise.

The second method is SPOT, or Sparse Pixels Occupancy Test developed by Cheung *et al.* in [29] and [65]. When whole voxels are projected to an image plane, the projected voxel covers multiple pixels. The SPOT method determines whether a voxel should be classified as inside or outside of a silhouette when the projected voxel covers both silhouette and non-silhouette pixels. SPOT chooses a threshold on the number of silhouette pixels necessary to classify a voxel as being inside a silhouette. It does this by minimizing a cost function including parameters η and ξ, representing the false negative and false positive pixel probability, respectively. If the number of

silhouette pixels in each camera for a particular voxel exceeds the threshold, then the voxel is labeled occupied and empty otherwise.

The Unbiased Hull, abbreviated as UH, of Landabaso *et al.* [13] is the third comparison method. In the UH method, the search space is partitioned into the visual hull (the set of voxels that project inside silhouettes for N cameras) and the inconsistent hull. Voxels in the inconsistent hull project inside the silhouettes in some cameras and outside in others. Some of the voxels in the inconsistnt hull are occluded by visual hull voxels. The UH algorithm then, for each possible number of occlusions by the visual hull, selects the parameter k for the robust visual hull, by minimizing a cost function that incorporates η and ξ from SPOT, as well as a probability of voxels being background, P_b. The resulting reconstruction then is a robust visual hull-type reconstruction, with k varying with the number of occlusions by visual hull voxels.

The Graph Cuts method is from Snow *et al.* [30]. They cast the SfIS problem as a energy minimization problem for Boolean variables. The energy minimization problem contains data term costs, which incorporate parameters A and B, and a smoothness cost penalty for neighbors with different labels, parameter λ. Since the cost function is submodular, the problem can be solved with graph cuts, meaning a network max flow/min cut problem.

We altered the SPOT and UH methods to create the SPOT-SIE and UH-SIE methods. SPOT-SIE and UH-SIE operate similarly as their namesakes, except that we substitute our SIE function for the cost functions requiring η, ξ, and P_b parameters. Consequently, no parameters need to be specified for the altered methods SPOT-SIE and UH-SIE other than voxel size.

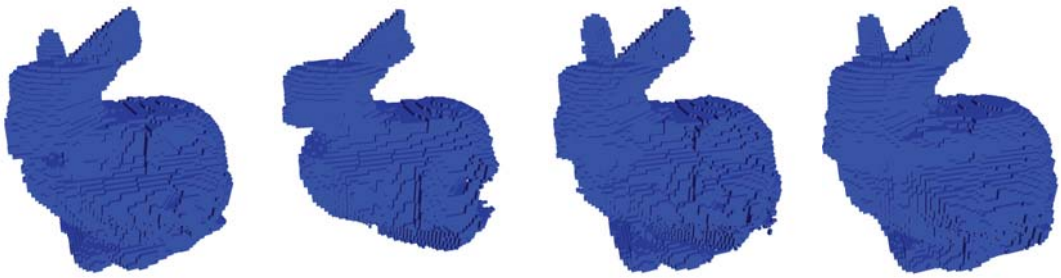
Because the SPOT, UH, and Graph Cuts methods required parameters, it was difficult to appropriately choose parameters in a way that would ensure that these methods were being represented correctly. To get around this problem, we iterated over a range of possible parameters and chose the parameters that resulted in the lowest number of misclassifications as compared to the ground truth. Since some of the comparison methods were intended for human reconstruction applications, we

also included the DANCER A dataset although a ground truth was not available. For the DANCER A dataset we show reconstructions with a range of parameters.

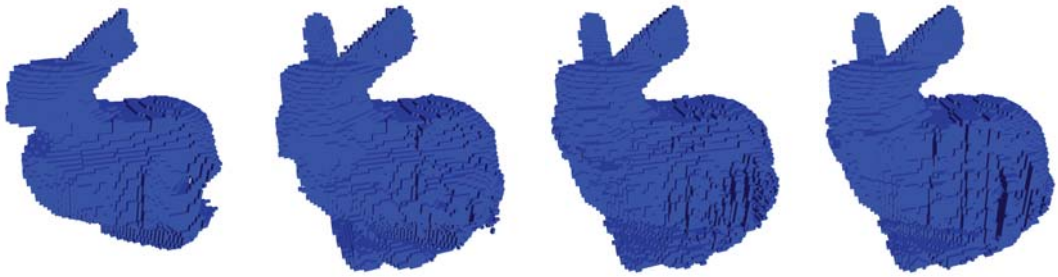
6.3.1 Bunny Segmentation Error Dataset Comparisons

The comparison methods using the BUNNY SEGMENTATION ERROR dataset are shown for two views, in Figures 6.30 and 6.31, and quantitative comparisons with the ground truth are shown in Table 6.8. The table also lists the parameters chosen for the SPOT, UH, and Graph Cuts methods. The voxels were 5 mm cubes.

Figures 6.30 and 6.31 show that the pixel occupany approach of SPOT and SPOT-SIE result in large regions of false negatives. The Graph Cuts approach reconstructs the shape of the bunny, though the reconstruction is larger than the original model. The remaining methods' performance on this dataset is similar, with the number of misclassified voxels ranging from $19,415$ to $26,185$. Differences between the remaining methods include the UH and UH-SIE reconstructions' absence of part of the front feet and part of the back, the missing lower jaw in the RRVH, and holes in the tail and back region in the local minimum search methods' reconstruction. In this dataset, there was little difference in between the original SPOT and UH reconstructions versus the SPOT-SIE, UH-SIE reconstructions.



(a) RRVH (b) SPOT (c) UH (d) Graph cuts

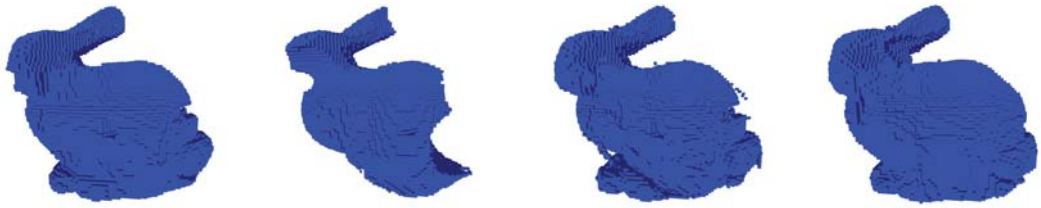


(e) SPOT-SIE (f) UH-SIE (g) LMS-SfSPM (h) HLMS-SfSPM

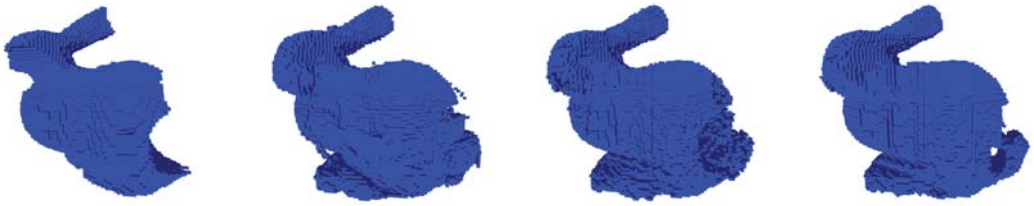
Fig. 6.30.: Illustration of the comparison reconstructions and the LMS-SfSPM and HLMS-SfSPM reconstructions for the Bunny Segmentation Error dataset, view 1.

Table 6.8: Results for comparison reconstructions as compared to the ground truth model for the Bunny Segmentation Error dataset

Method	Number Misclassifications	False Positive rate	False Negative rate
RRVH $m_t = 0.9$	20,661	0.03499	0.01908
SPOT $\eta = 0.05, \xi = 0.2$	32,510	0.007795	0.2558
UH $\eta = 0.1, \xi = 0.05, P_b = 0.2$	26,181	0.01246	0.1502
Graph cuts $A = 500, B = 400, \lambda = 0$	21,621	0.02032	0.03336
SPOT-SIE	32,786	0.007478	0.2614
UH-SIE	26,185	0.01246	0.1502
LMS-SfSPM	25,816	0.0113	0.1566
HLMS-SfSPM	19,415	0.0156	0.0538



(a) RRVH (b) SPOT (c) UH (d) Graph cuts



(e) SPOT-SIE (f) UH-SIE (g) LMS-SfSPM (h) HLMS-SfSPM

Fig. 6.31.: Illustration of the comparison reconstructions and the LMS-SfSPM and HLMS-SfSPM reconstructions for the Bunny Segmentation Error dataset, view 2

6.3.2 Bunny Image Noise Dataset Comparisons

The Bunny Image Noise dataset reconstruction had more variation in the methods' performance than the Bunny Segmentation Error dataset; reconstructions are shown in Figure 6.32 and quantitative results are in Table 6.9. Those methods with the highest number of misclassifcation were the SPOT and UH approahces, where the misclassifications consisted of almost exclusively false negatives. For this dataset, the SPOT-SIE and UH-SIE methods performed considerably better than the original SPOT and UH methods, but these reconstruction also had many false negatives in the interior regions of the bunny model.

The RRVH reconstruction had the next highest number of misclassifications, and a high percentage of false negatives as well. The LMS-SfSPM reconstruction also had false negatives in the interior regions of the bunny; the HLMS-SfSPM reconstruction was similar to LMS-SfSPM but had less than half the number of misclassifications as LMS-SfSPM did. Finally, the graph cuts approach performed very well for this dataset, reconstructing the shape of the bunny model very accurately in spite of 20% image noise.

Table 6.9: Error of the reconstruction methods as compared to the ground truth model for the Bunny Image Noise dataset

Method	Number Misclassifications	False Positive rate	False Negative rate
RRVH $m_t = 0.4$	85,463	0.02079	0.6697
SPOT $\eta = 0.05, \xi = 0.2$	99,654	1.11069e-06	0.99991
UH $\eta = 0.05, \xi = 0.05, P_b = 0.2$	99,662	0	1
Graph cuts $A = 300, B = 200, \lambda = 30$	14,685	0.01187	0.0401
SPOT-SIE	70,273	0.006704	0.6445
UH-SIE	70,288	0.006729	0.6445
LMS-SfSPM	42,626	0.0363	0.0994
HLMS-SfSPM	20,094	0.0219	0.0035

(a) RRVH (b) SPOT (c) UH (d) Graph cuts

(e) SPOT-SIE (f) UH-SIE (g) LMS-SfSPM (h) HLMS-SfSPM

Fig. 6.32.: Illustration of the comparison reconstructions and the LMS-SfSPM and HLMS-SfSPM reconstructions for the Bunny Image Noise dataset

6.3.3 Model Tree Dataset Comparisons

The reconstructions of the Model Tree dataset are shown in Figure 6.33 and quantitative results are shown in Table 6.10. In this dataset, there is a great deal of variation in how each method reconstructs the tree. The poorest reconstruction in terms of resembling the tree are the SPOT and SPOT-SIE methods, which consisted of many false negatives. The worst reconstruction in terms of misclassification error is the UH-SIE method, which was characterized by many false positives. Recall that the silhouettes in this set are truncated, not all voxels are seen by all cameras, and the visual hull consists of very few occupied pixels. With the visual hull empty (or close to empty), the UH-SIE degrades to a sort of robust visual hull method, using absolute camera numbers instead of percentages as we do in our RRVH implementation.

The RRVH and Graph Cuts methods produced very similar reconstructions, in that the trunk was reconstructed but the branches were not, leading to the lowest numbers of misclassification error of the methods compared and a very low false positive rate. Finally, the LMS-SfSPM and HLMS-SfSPM methods performed similarly, reconstructing the trunk and branches, though the branches contained false positives as compared to the ground truth.

From examining Table 6.10 and Figure 6.33, it may be difficult to see why some reconstructions score as they do. The ground truth model we use for comparison is stationary, so if the branches are at a different location than they are in the model, these count as false positives. Also, we illustrate in Figure 6.34 the methods with the lowest false negative rates. The Graph Cuts approach smooths the reconstruction, leading to cylindrically shaped regions that are slightly bigger than the central trunk, leading to a low false negative rate as the trunk contains more voxels than the branches. We can see that the RRVH method's trunk reconstruction is smaller than the ground truth trunk. Finally, the HLMS-SfSPM method reconstructs a central trunk with a smaller diameter than either the Graph cuts approach or RRVH. It also reconstructs some parts of branches, but they are offset from the ground truth

locations, leading to a higher false negative rate than the Graph cuts or RRVH reconstructions.

Table 6.10: Error of the reconstruction methods as compared to the ground truth model for the Model Tree dataset

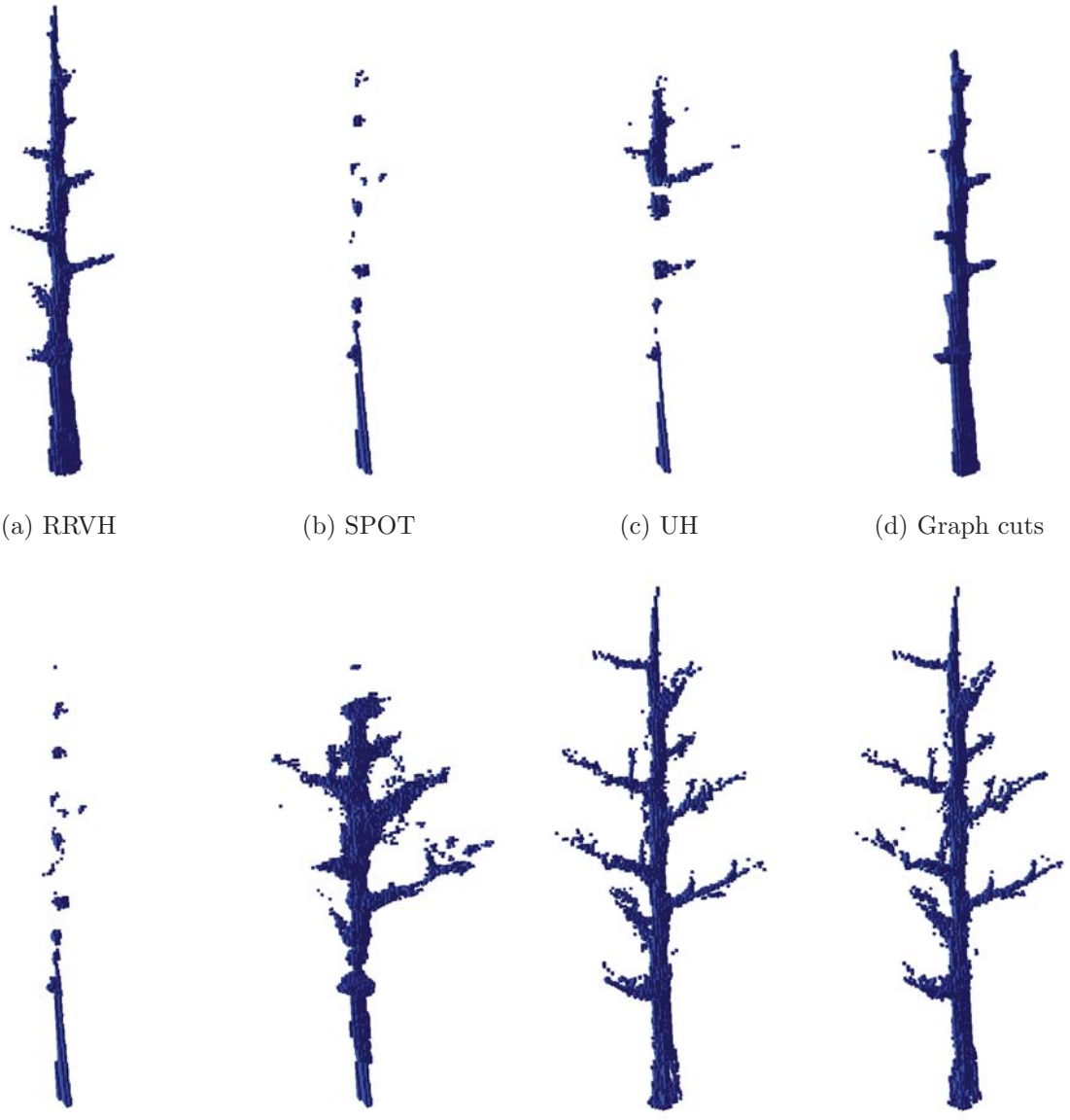
Method	Number Misclassifications	False Positive rate	False Negative rate
RRVH $m = 0.65$	4,548	0.000327865	0.430392
SPOT $\eta = 0.05, \xi = 0.15$	7,546	8.22826e-06	0.921569
UH $\eta = 0.1, \xi = 0.05, P_b = 0.2$	7,275	0.00010222	0.851961
Graph cuts $A = 400, B = 300, \lambda = 10$	4,129	0.000235771	0.414706
SPOT-SIE	7,483	1.42412e-05	0.91152
UH-SIE	10,028	0.00204029	0.438848
LMS-SfSPM	5,784	0.000615854	0.470343
HLMS-SfSPM	5,493	0.000603828	0.439338

6.3.4 Dancer A Dataset

Since the Dancer datasets lacked a ground truth, for the methods that require parameters, we chose the parameters that showed the most representative reconstructions. In some cases choosing one set of parameters was difficult, so for some methods we show two or more parameter choices in Figure 6.35.

The RRVH parameters are $m = 0.65$ and $m = 0.80$ in Figures 6.35a and 6.35b. We can see that the when $m = 0.65$ that the body is larger than the original person, while when $m = 0.80$, the left leg is not reconstructed. The SPOT method's reconstruction has false negatives in the arms and legs; however, the altered method SPOT-SIE was able to reconstruct the figure more accurately than SIE, except for a break in one leg.

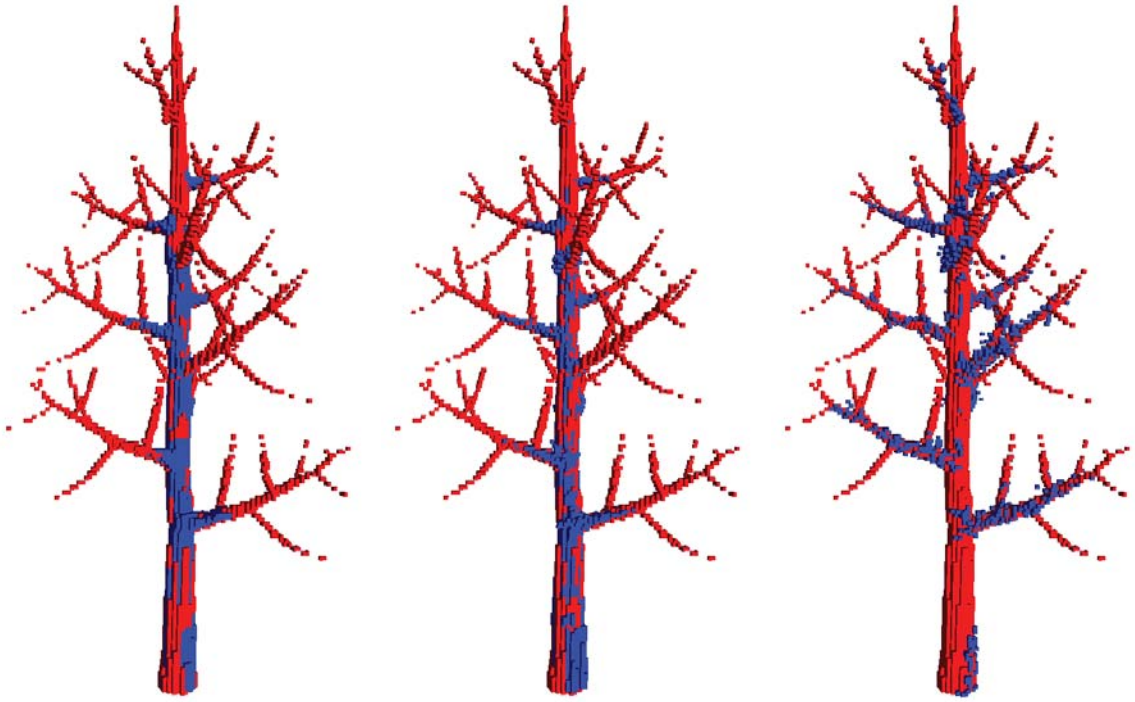
The UH method in Figure 6.35d has some noisy occupied voxels, but generally reconstructs the dancer. The success of the UH method on this dataset is highly dependent on the parameter choices, as shown by Figure 6.35e, where parts of the



(a) RRVH (b) SPOT (c) UH (d) Graph cuts

(e) SPOT-SIE (f) UH-SIE (g) LMS-SfSPM (h) HLMS-SfSPM

Fig. 6.33.: Illustration of the comparison reconstructions and the LMS-SfSPM and HLMS-SfSPM reconstructions for the Model Tree dataset.



(a) Graph cuts (b) RRVH (c) HLMS-SfSPM

Fig. 6.34.: Model Tree ground truth (in red) with the reconstructions of three of the comparion methods (in blue). These three methods have the lowest false negative rates, at 0.41 (Graph cuts), 0.43 (RRVH), and 0.44 (HLMS-SfSPM). Roughly, we can see that the more blue that can be seen means a higher false positive rate (i.e. the reconstruction is bigger than the model in the blue regions), whereas if the percentage of red is greater there is a higher false negative rate (the reconstruction is smaller than the model in the red regions).

legs are missing. In the Graph Cuts method, when the smoothing parameter $\lambda = 0$, only data costs remain. Figures 6.35f and 6.35g show the effect of using the same A and B parameters but varying λ. With $\lambda > 0$, thin regions such as the legs and arms are removed. The effect of the same A and B with different values of λ is also shown in Figures 6.35h and 6.35i, where the greater difference in between A and B results in less of a smoothing effect on the reconstruction.

The four parameter-free methods SPOT-SIE, UH-SIE, LMS-SfSPM, and HLMS-SfSPM perform similarly, reconstructing the dancer's arms and legs (with the exception of false negatives in the legs of SPOT-SIE). We judge the HLMS-SfSPM method to be the most accurate, as it reconstructs the outer shape of the figure and has few interior false negatives.

6.4 Analysis of Comparison Methods

The parameter methods (SPOT, UH, graph cuts) were sometimes very successful as compared to the ground truth, though in practice the selection of parameters must be determined empirically and for applications with little *a priori* information, cannot be gauranteed to perform as consistently as the local minimum search methods. We showed the comparisons with the altered methods SPOT-SIE and UH-SIE to demonstrate how the mechanisms of those methods can be effective when our SIE function is substituted for the original method's cost function, such as in the Bunny Image Noise results, Figure 6.32e.

However, except for the case of image noise, SPOT's pixel occupancy approach is not able to handle either determinstic segmentation error or calibration error, in the Bunny Segmentation Error and Model Tree datasets, respectively. While the UH-SIE approach worked well for the Bunny Segmentation Error dataset, in the case of partial silhouettes, UH-SIE's approach of searching for the best number of inconsistencies creates a resoncstruction with false positive around the voxels with the highest number of cameras viewing them such as in the Model Tree dataset.

(a) RRVH $m = 0.65$ (b) RRVH $m = 0.80$ (c) SPOT $\eta = 0.05, \xi = 0.15$ (d) UH $\eta = 0.05$, $\xi = 0.10, P_b = 0.2$ (e) UH $\eta = 0.15$, $\xi = 0.05, P_b = 0.5$

(f) Graph cuts $A = 450$ $B = 400, \lambda = 0$ (g) Graph cuts $A = 450$ $B = 400, \lambda = 10$ (h) Graph cuts $A = 500$ $B = 400, \lambda = 0$ (i) Graph cuts $A = 500$ $B = 400, \lambda = 10$

(j) SPOT-SIE (k) UH-SIE (l) LMS-SfSPM (m) HLMS-SfSPM

Fig. 6.35.: Illustration of the comparison reconstructions and the LMS-SfSPM and HLMS-SfSPM reconstructions Dancer A dataset.

The Graph Cuts method was able to generate very representative and low misclassification error reconstructions in some situations (the Bunny Image Noise and Model Tree datasets, respectively). Deciding the parameter settings and also their relationship to voxel size requires a lot of care, as in the Dancer reconstructions, Figures 6.35f to 6.35i.

Our LMS-SfSPM and HLMS-SfSPM methods performed reliably in a range of situations, though the number of misclassifications was not always the lowest among the competing methods. The HLMS-SfSPM method seems to produce the most reliable reconstructions, particularly with respect to false negatives.

7. EXPERIMENTAL RESULTS FOR CAMERA CALIBRATION CORRECTION

To validate our camera calibration correction method, we show results from simulated datasets and real datasets from the laboratory. Before describing the datasets, we first mention some key details in our implementation of the method.

The camera calibration method we describe in Chapter 5 can be used with any SfIS method. Note that our camera calibration correction method is only for SfIS, not for SfSPM. In the following experiments, we use our HLMS-SfSPM method for the SfIS reconstruction method. With n_s as the number of divisions performed in the HLMS-SfSPM method, our implementation of the camera calibration correction method is a follows. First, the HLMS-SfSPM algorithm progresses through $n_s - 1$ divisions. Given a reconstruction at the $n_s - 1$ level, isolated occupied voxels (with no 26-connected neighbors) are removed and the camera calibration correction procedure of Chapter 5 is performed. Then the HLMS-SfSPM division is performed for the n_sth division, using the updated camera calibration parameters.

Note that while this is one approach for applying the camera calibration procedure, there are many other variations possible; how and when to apply the camera calibration correction method in a reconstruction pipeline depends on the characteristics of the datasets and constraints of the application. We chose to use the camera calibration correction method as described above because it offered an increase in reconstruction accuracy for little computational time.

In order to compare the corrected versus uncorrected reconstructions, we also create a HLMS-SfSPM reconstruction at level n_s without the correction, using the same $n_s - 1$ level reconstruction the corrected reconstruction uses as an initial solution.

We deal with the two correction scenarios, when only external camera parameters are corrected, and when both internal and external camera parameters are corrected.

In our experiments, we correct the external camera parameters first. Then, if internal parameters are to be corrected as well, we correct external and internal camera parameters, using the solution generated by correcting only the external parameters as an initial solution.

Finally, for the implementation of the Levenberg-Marquadt method, we used the software package `levmar` [66]. On average, the camera calibration correction step takes 60 seconds or less.

7.1 Simulated Datasets

We use the simulated Model Tree dataset of Chapter 6. Recall that camera calibration error was added to the dataset by altering the translation component of each camera on one axis by $e \in \{-20, -19, ..., 19, 20\}$. The element altered (on the x, y, or z axis) and the value of e was determined randomly, with all possible values of e having a uniform probability of being selected, as did the three axes of $\mathbf{t}$.

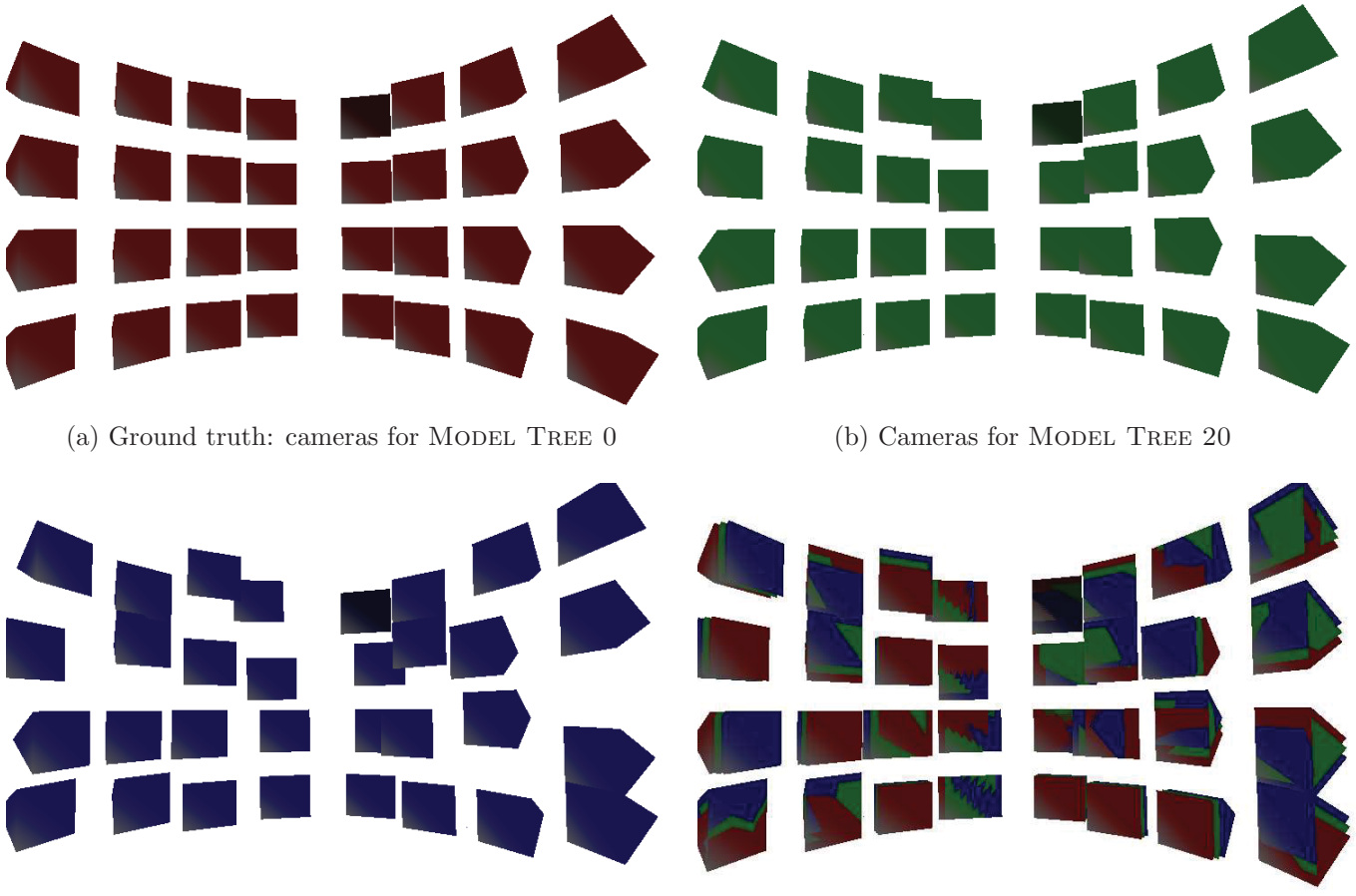
In order to determine what levels of error can be corrected by our method, we generated simulated datasets of the model tree with increasing error to assess the method's performance. First, a translation element (x, y, or z) is randomly chosen for each camera, as is a direction (-1 or $+1$). We let $e \in \{0, 2, 4, 6, ..., 48, 50\}$. For each possible e, we alter the ground truth camera calibration by adding the randomly selected direction times e to the randomly selected translation component for each camera.

We refer to the sets as follows: Model Tree 2 means the dataset where $e = 2$, Model Tree 4 refers to the dataset where $e = 4$, etc. When referring to the group of datasets for all values of e, we call this group the Model Tree e datasets.

Figure 7.1 shows the cameras for datasets where $e = 0$, $e = 20$, and $e = 40$. For each camera, the directions and axis to be altered are constant over all the sets in Model Tree e. For instance, consider the third camera on the top row in Figure

7.1a. When $e = 20$ in Figure 7.1b, that camera is moved up. When $e = 20$ in Figure 7.1c, the top row, third camera is moved up even more.

The silhouette images are acquired from the ground truth cameras, which are arranged on two planes, on regularly-spaced grids (Figure 7.1a). Consequently, the silhouette images used for all of the datasets in MODEL TREE e are the same. Note that this choice of ground truth versus the cameras in MODEL TREE e is the reverse of what one may experience in a real application; usually the cameras are assumed to have a regular pattern and errors occur; the silhouette images are then captured from the cameras with unknown error. Because we do not assume any prior knowledge about relationships in between cameras, this switch between a standard real scenario and our simulated scenario should result in to no difference between the two choices of experimental desgin.



(a) Ground truth: cameras for Model Tree 0

(b) Cameras for Model Tree 20

(c) Cameras for Model Tree 40

(d) Overlay of the Model Tree 0, Model Tree 20, and Model Tree 40 camera configurations

Fig. 7.1.: Illustration of the ground truth camera configuration and the datasets generated with $e = 20$ and $e = 40$.

As in Chapter 6, the ground truth model is converted to a voxel-based representation to facilitate comparison between reconstructions and the ground truth. For the Model Tree and Model Tree e datasets the settings for the HLMS-SfSPM method are: initial voxel size is 20 mm, $n_s = 3$, and final voxel size is a 2.5 mm cube.

We also show results using an alternate implementation on the Model Tree e datasets. Recall that the implementation in this chapter is that the camera calibration procedure is performed once and the $n_s - 1$ level, and then the HLMS-SfSPM method continues to the n_sth division with updated camera calibration parameters. The alternate implementation is to continue correcting calibration parameters and reconstructing at the $n_s - 1$ level, as long as the SIE value continues to decrease. Once the SIE value increases, the calibration correction process is halted and reconstruction is performed at the n_sth division with the best camera calibration parameters found so far, meaning those parameters that produce the lowest value of SIE so far.

7.2 Real Datasets

We generated real datasets in our laboratory using two different camera configurations. In the first configuration, there are 13 inexpensive webcameras (image size 1280×960 pixels) mounted such that they are to the side and above an object. The external camera calibration was estimated using the camera calibration procedure of Zhang [63], using a custom 2-plane calibration object. The datasets using this configuration are called Branch, Coil, and Coil and Cables. The HLMS-SfSPM method is used with $n_s = 2$ and final voxel size 1.5 mm.

The second configuration is like the Standard Apple, Weeping Apple, and Pole and Coil E from Chapter 6: one camera (image size 2456×2058 pixels) is mounted on the end effector of a robot; the robot moves to 38 different positions and acquires images of a large object. In fact, the datasets used in this chapter use the same images from Standard Apple, Weeping Apple, and Pole and Coil E but a different calibration. We refer to the datasets for this chapter as Standard

Apple E, Weeping Apple E, and Pole and Coil E. They were generated using a different position of the planar calibration pattern in the space and we use the hand-eye, robot-world calibration of Hirsh *et al.* [67]. Generating hand-eye, robot-world calibrations of higher accuracy in various conditions is one of our active areas of research, but will not be discussed in this thesis other than to mention that the large object datasets in this chapter (Standard Apple E, Weeping Apple E, and Pole and Coil E) have poorer calibration accuracy than those of the previous chapter (Standard Apple, Weeping Apple, and Pole and Coil E). The HLMS-SfSPM method is used with initial voxel size 12 mm, $n_s = 2$, so final voxel size is 3 mm.

For both of these dataset groups, silhouette images were generated using a threshold-based background subtraction method.

7.3 Results of the Camera Calibration Correction for the Model Tree e Datasets

The camera calibration correction method for external parameters was applied to the Model Tree e datasets, using the standard implementation (one camera calibration correction pass) versus the alternate implementation (multiple camera calibration correction passes). The silhouette inconsistency error (SIE) and number of misclassifications as compared to the ground truth for the uncorrected, single corrected, and multiple corrected reconstructions are in Table 7.1; graphs of the SIE and number of misclassifications of uncorrected, single corrected, and multiple corrected reconstructions is shown in Figures 7.2 and 7.3. A selection of reconstructions from the Model Tree e group of datasets are shown in Figures 7.4-7.6.

From examining Table 7.1 and Figures 7.2 and 7.3, the single pass camera calibration correction implementation always produces reconstructions with lower values of SIE and fewer misclassifications as compared to the ground truth model, when compared to the uncorrected reconstructions, for $e \leq 50$. Figure 7.2 show that as e increases, the SIE for the uncorrected and single pass corrected reconstructions

stabilizes near $e = 34$, and to a lesser extent so does the number of misclassifications in Figure 7.3. The tree model in this dataset has a large central branch. Even when error is severe at $e = 50$, that central branch is still reconstructed, while the secondary branches that emanate from that branch are not. The thickness of the branch as well as the characteristics of the datasets (all cameras point toward the center in the ground truth camera calibration configuration) also contribute towards this behavior.

The multiple pass camera calibration reconstructions display different trends than the single pass version. Use of the multiple pass version results in lower SIE values as compared to the uncorrected and single pass camera calibration implementation. In fact, the SIE values remain stable around $500,000$ from $e = 0$ to $e = 30$, at which point the SIE values do increase as e increases, though SIE values for the multiple pass correction remain lower than that of the single pass correction. The number of voxel misclassifications for the multiple pass version are lower than those of the uncorrected and single pass corrected versions from $e = 0$ to $e = 16$. For datasets where $e \geq 18$, the number of voxel misclassifications using the multiple pass method rises as e rises.

When $e = 0$, the single and multiple pass camera calibration implementations do change the camera calibration parameters because of discretization artefacts between projected voxel reconstructions and silhouettes. As a result, the corrected reconstructions for Model Tree 0 have a higher SIE and number of ground truth misclassifications. However, the visual difference between reconstructions in Figure 7.4a is small.

In Model Tree 10 reconstructions shown in Figure 7.4b, the uncorrected reconstruction contains some ambuiguity in the secondary branches. The single pass and multiple pass corrected reconstructions are visually similar to the Model Tree 0 reconstruction although the number of misclassifications is twice as high ($23,470$ and $21,369$ versus $10,826$). As explanation for this is that the HLMS-SfSPM method and the camera calibration correction procedure compensates for the calibration error in

Table 7.1: SIE values and number of misclassifications for Model Tree e.

e	SIE uncorrected	SIE corrected single	SIE corrected multiple	Number misclass. uncorrected	Number misclass. single	Number misclass. multiple
0	474,233	479,142	477,383	10,826	11,362	11,106
2	761,704	499,141	490,781	14,225	11,353	11,140
4	1,134,187	544,716	501,790	20,162	13,119	12,319
6	1,404,738	622,719	502,463	25,331	16,938	15,171
8	1,601,602	723,171	502,707	29,138	21,296	17,839
10	1,752,906	849,857	508,423	31,407	23,470	21,369
12	1,877,307	947,716	515,608	32,987	26,520	24,819
14	1,982,945	1,110,284	507,563	34,616	30,280	26,725
16	2,070,690	1,282,005	507,859	35,663	32,854	30,686
18	2,144,524	1,448,956	509,630	36,610	34,210	35,458
20	2,209,796	1,689,964	501,345	36,884	33,082	36,369
22	2,267,161	1,774,048	509,589	38,958	32,275	37,333
24	2,315,839	1,916,228	509,530	40,425	34,895	38,877
26	2,359,188	1,996,990	508,108	42,171	35,679	41,024
28	2,396,981	2,088,781	516,425	42,346	37,389	45,942
30	2,426,328	2,203,223	625,453	42,212	37,458	45,462
32	2,453,696	2,230,773	574,667	42,217	36,037	43,101
34	2,476,576	2,312,358	654,834	41,850	37,207	47,336
36	2,498,844	2,342,429	703,585	41,836	37,171	49,355
38	2,520,363	2,356,482	726,378	40,314	37,447	46,803
40	2,532,851	2,370,370	807,577	40,883	36,886	57,514
42	2,544,773	2,409,602	1,249,962	40,953	37,424	51,766
44	2,557,311	2,416,400	1,142,661	40,980	37,539	51,750
46	2,565,881	2,424,331	1,357,189	40,778	37,660	55,260
48	2,573,959	2,474,882	2,294,390	39,786	38,613	45,095
50	2,581,056	2,477,458	1,490,177	40,194	37,865	58,320

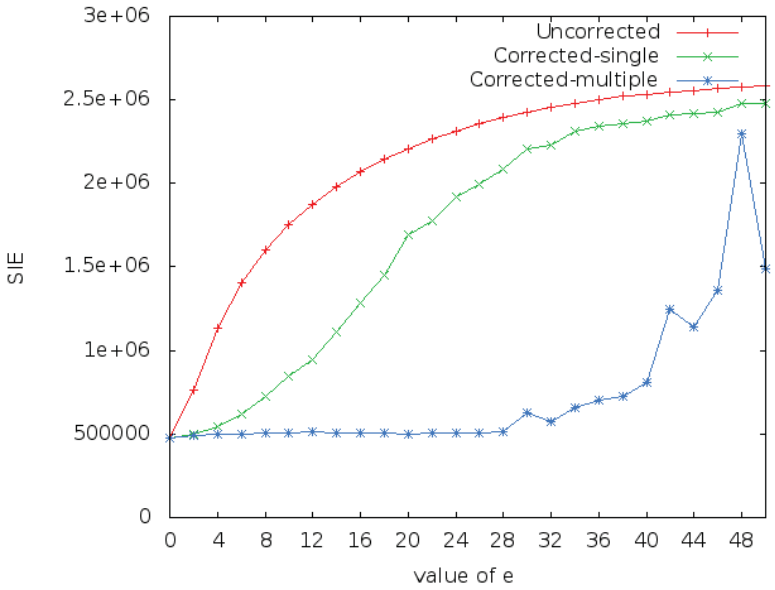


Fig. 7.2.: Graph of the SIE error for the Model Tree e datasets, for uncorrected, single pass corrected and multiple pass corrected reconstructions.

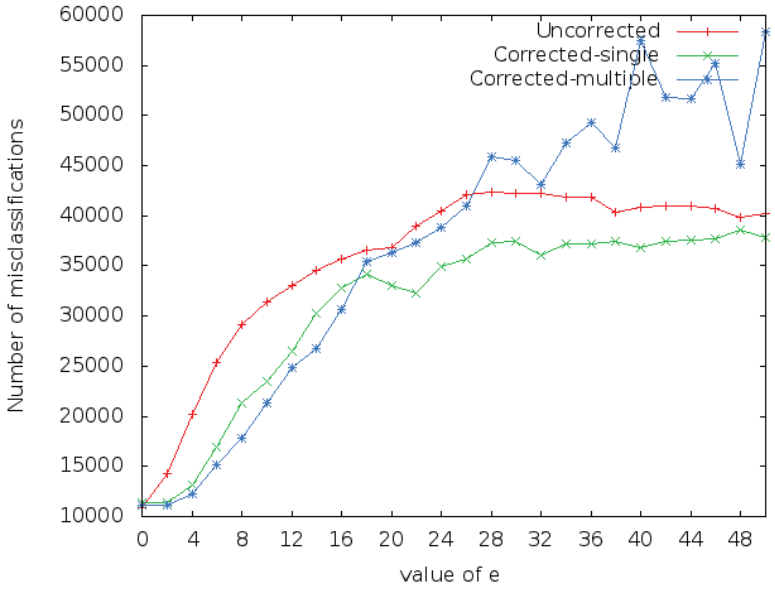


Fig. 7.3.: Graph of the number of misclassifications as compared to the ground truth for the Model Tree e datasets, for uncorrected, single pass corrected and multiple pass corrected reconstructions.

the dataset and is able to reconstruct the branching structure such that it resembles the ground truth images. However, the localization of the branches may not be exact; as for our camera calibration correction there is no global bundle adjustment procedure.

For the Model Tree 20, Model Tree 30, Model Tree 40, and Model Tree 50 uncorrected and corrected reconstructions in Figures 7.5 and 7.6, we observe more ambuiguity in the branches in the uncorrected version than in the Model Tree 0 and Model Tree 10 datasets, and this ambuiguity is reduced in the single pass corrected reconstruction. With increasing values of e, the reconstruction quality of the single pass corrected reconstructions degrades. When $e = 50$, we observe that the single pass corrected reconstruction does not resemble the original tree any more than the uncorrected reconstruction does, despite a difference in SIE and number of misclassifications between the two reconstructions.

However, using the multiple pass version, the reconstruction quality appears high for larger e, from visual inspection of reconstructions in Figures 7.5 and 7.6. Recall though that the number of misclassifications of multiple pass reconstructions versus the ground truth increase as e increases. In Figure 7.9, we show the ground truth reconstruction in red and the multiple pass corrected reconstructions in blue. From closely inspecting of the recconstructions versus the ground truth figure, the distance between branches increase in reconstructions. So, although these reconstructions appear complete, branch localization error exists.

We show the differences between uncorrected and single pass corrected reconstruction images in Figures 7.10-7.11, on a sampling of the Model Tree e datasets. We can see that the algorithm is able to align the images for the Model Tree 10 and Model Tree 20 datasets, but the numbers of mismatches between the silhouette and corrected reconstructions increases for the Model Tree 30 dataset and the Model Tree 50 datasets. A comparison of uncorrected and multiple pass corrected reconstruction images is shown Figures 7.12-7.13. For the first example image, shown in Figure 7.12, the difference between the uncorrected and multiple pass corrected re-

construction images is very small. On the other hand, we can see that for the second example image in Figure 7.13, the multiple pass reconstruction images are aligned at $e = 30$ but not for $e = 40$ and $e = 50$. Despite this, the reconstruction quality is visually good for the $e = 40$ and $e = 50$ multiple pass reconstructions. This is because our HLMS-SfSPM method is tolerant to some error, so a small number of erroneous camera calibration parameters does not greatly affect the reconstruction.

The camera calibration correction method operates under the assumption that corresponding pixels between the projected reconstruction and silhouettes are relatively small, so the maximum distance between corresponding points is set at the image width divided by 10 in our implementation, and this setting was constant over all datasets shown in this chapter. For some datasets in Model Tree e, this distance is too small to generate appropriate correspondences.

We have seen that use of the multiple-pass correction implementation results in reconstructions that resemble the original object more than the single-pass correction implementation. This increase in reconstruction quality does have a cost. In Figure 7.7, we show the number of times the calibration correction is performed for the datasets in Model Tree e, and in Figure 7.8, we show the difference in running time between the single-pass and multiple-pass correction implementations. Use of the single pass implementation results in run times of 3-5 minutes, while the running time for the multiple-pass correction implementation generally rises as e rises; the running time for the Model Tree 50 dataset is over 50 minutes.

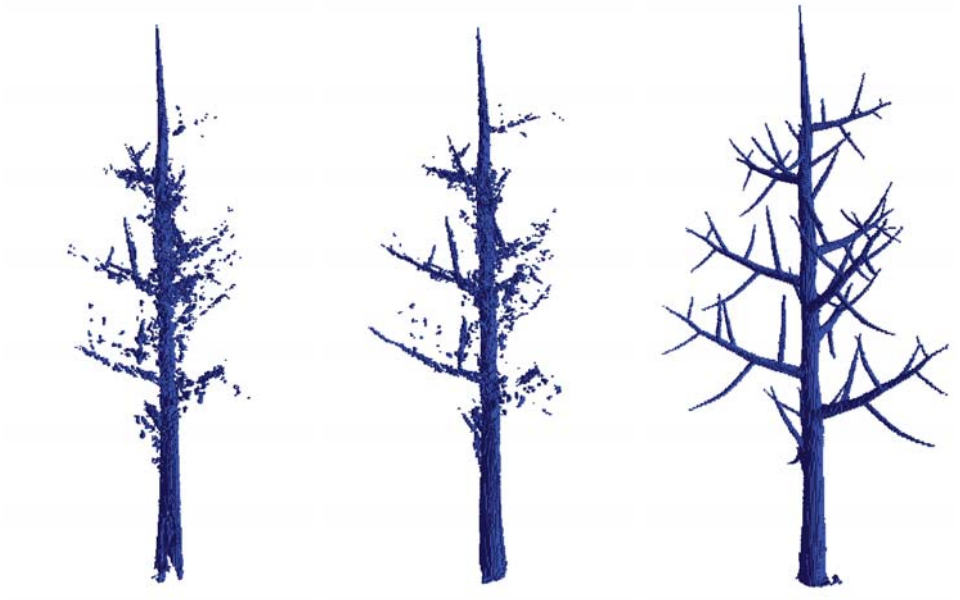
When using the multiple-pass correction, one possible way to reduce the migration of cameras and branch locations would be to use only the largest connected component of the reconstruction to align with the silhouettes. In this way, the influence of small noisy regions would be decreased, and alignment would then be based on the features of the object that are the most reliably reconstructed.

(a) $e = 0$

(b) $e = 10$

Fig. 7.4.: Each subfigure shows from left to right uncorrected, single pass corrected, and multiple pass corrected reconstructions for two of the MODEL TREE e datasets.

(a) $e = 20$



(b) $e = 30$

Fig. 7.5.: Each subfigure shows from left to right uncorrected, single pass corrected, and multiple pass corrected reconstructions for two of the MODEL TREE e datasets.

(a) $e = 40$

(b) $e = 50$

Fig. 7.6.: Each subfigure shows from left to right uncorrected, single pass corrected, and multiple pass corrected reconstructions for two of the MODEL TREE e datasets.

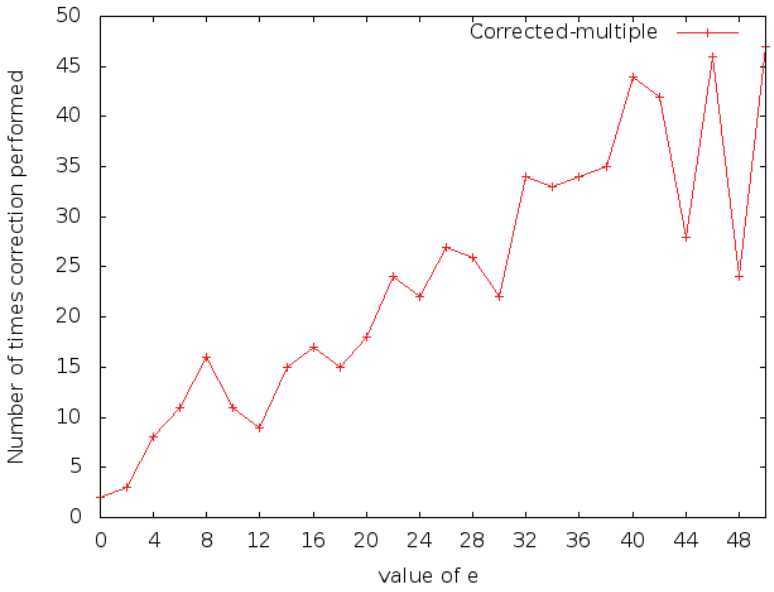


Fig. 7.7.: Number of iterations of the camera calibration correction for the multiple pass implementation on the Model Tree e datasets.

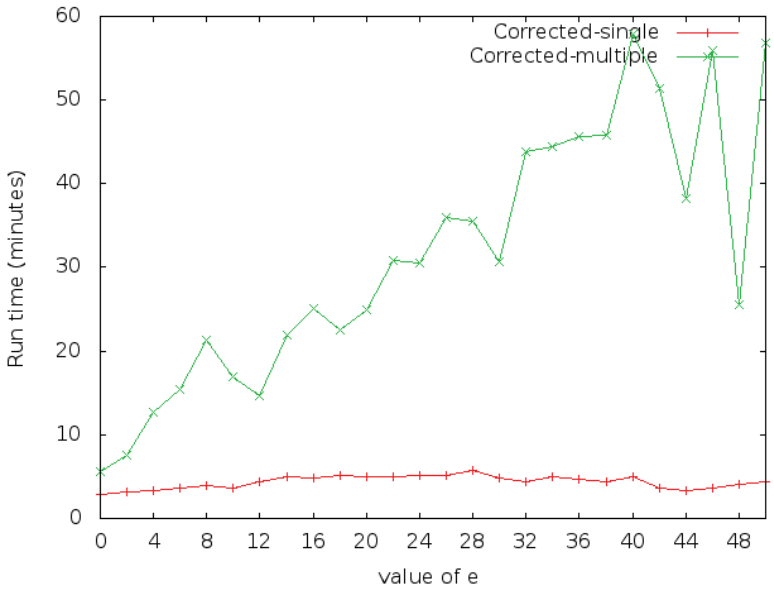
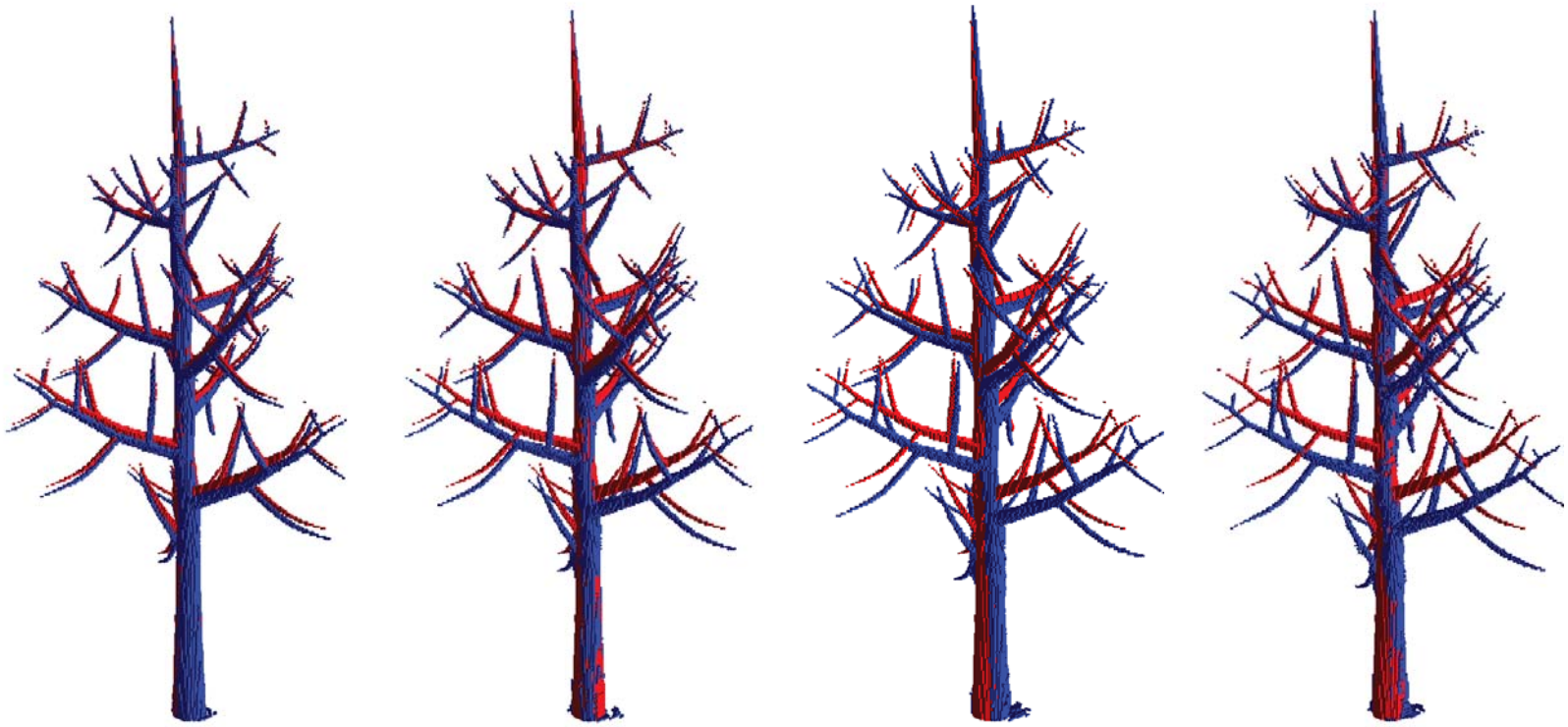
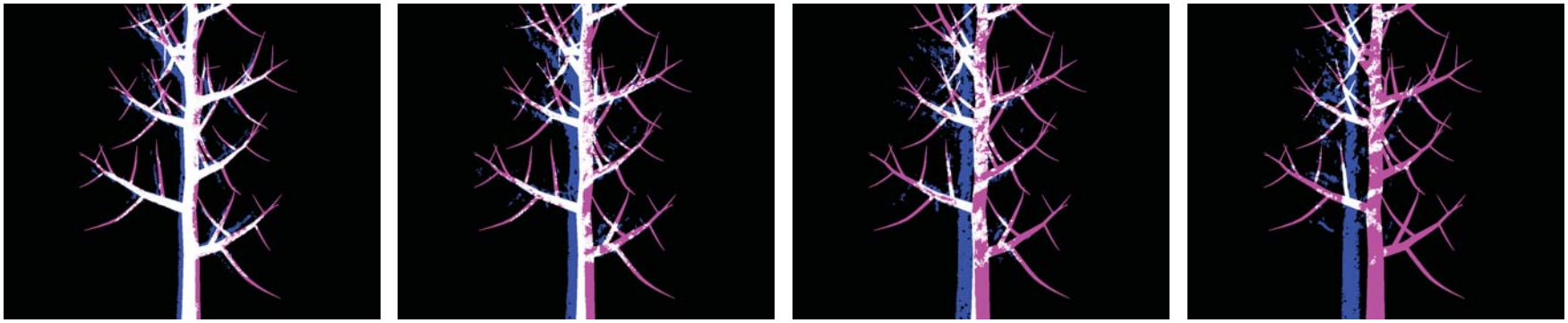


Fig. 7.8.: Running time comparison for the single pass versus multiple pass correction implementations for the Model Tree e datasets.

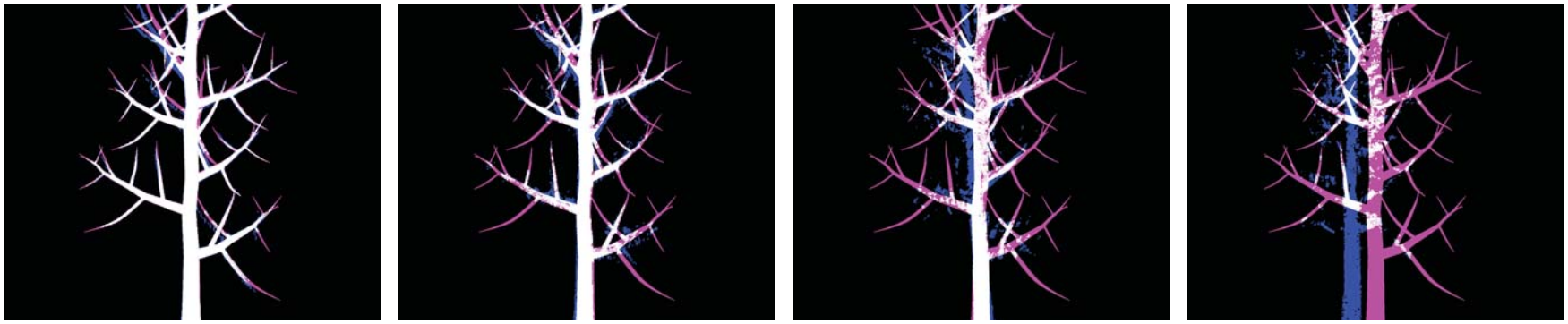


(a) $e = 20$ (b) $e = 30$ (c) $e = 40$ (d) $e = 50$

Fig. 7.9.: Illustration of the differences between multiple pass corrected reconstructions (in blue) versus the ground truth (in red).

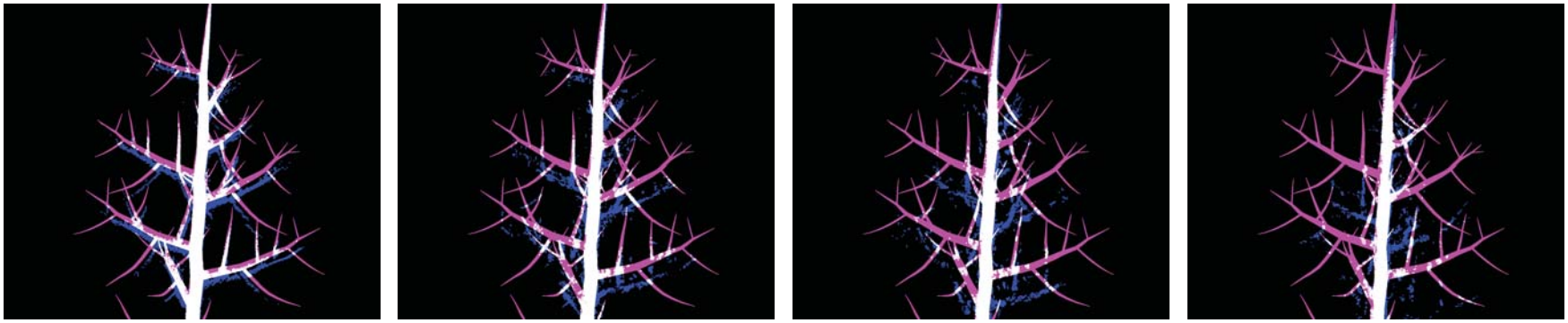


(a) $e = 10$ (b) $e = 20$ (c) $e = 30$ (d) $e = 50$

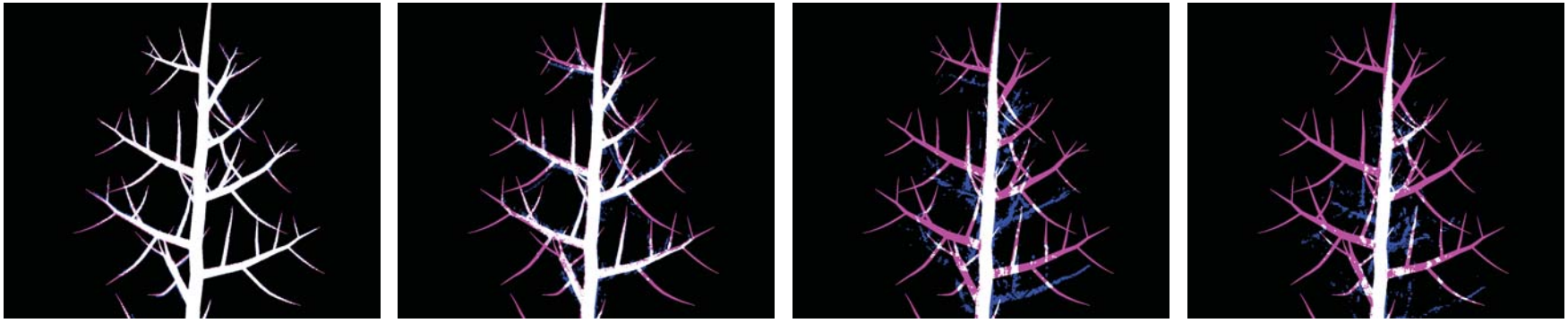


(e) $e = 10$ (f) $e = 20$ (g) $e = 30$ (h) $e = 50$

Fig. 7.10.: Silhouette and reconstruction images are overlaid for reconstructions using the uncorrected camera calibration parameters (top row) versus reconstructions generated with single corrected external camera calibration parameters (bottom row). Pixels marked magenta are from the silhouette only, pixels marked blue are from the reconstruction image only. White pixels indicate pixels where the silhouette and reconstruction image pixels overlap.

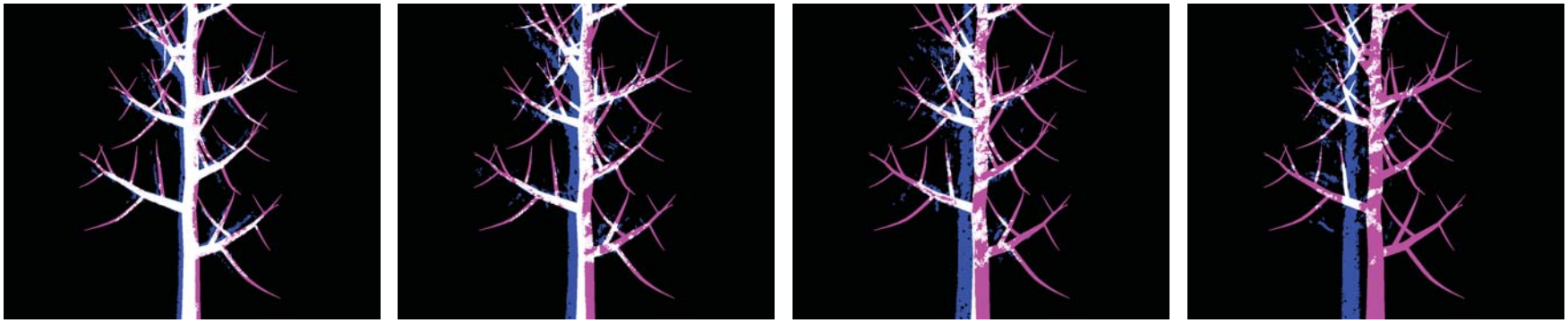


(a) $e = 10$ (b) $e = 20$ (c) $e = 30$ (d) $e = 50$

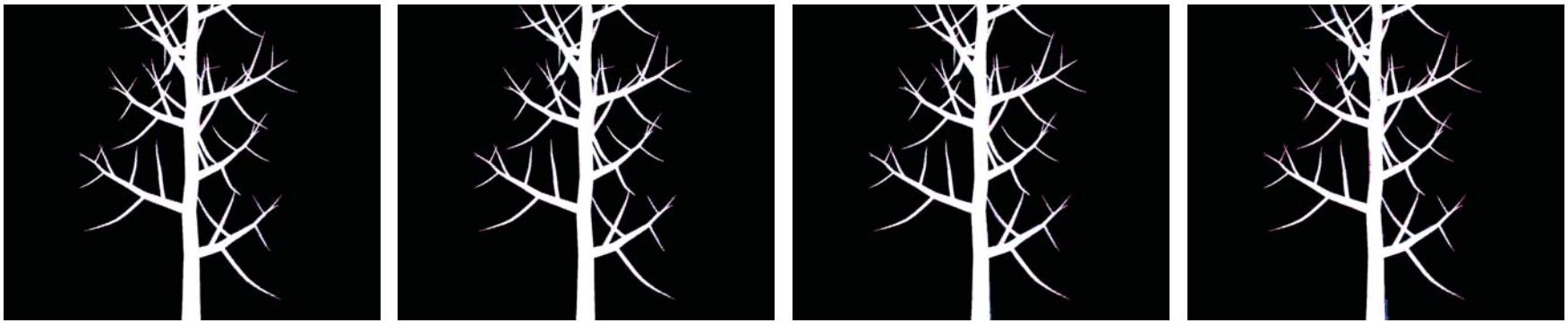


(e) $e = 10$ (f) $e = 20$ (g) $e = 30$ (h) $e = 50$

Fig. 7.11.: Silhouette and reconstruction images are overlaid for reconstructions using the uncorrected camera calibration parameters (top row) versus reconstructions generated with single pass corrected external camera calibration parameters (bottom row). Color coding as in Figure 7.10.

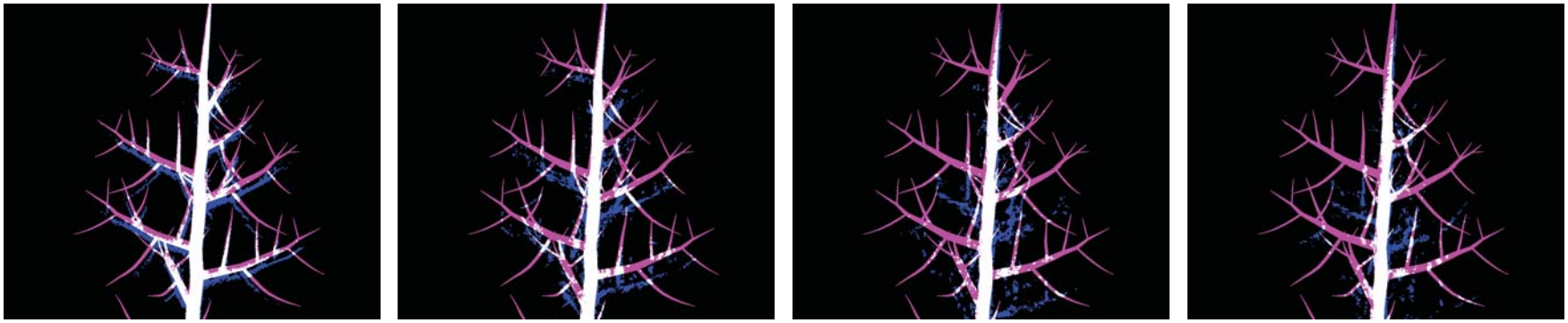


(a) $e = 10$ (b) $e = 20$ (c) $e = 30$ (d) $e = 50$



(e) $e = 10$ (f) $e = 20$ (g) $e = 30$ (h) $e = 50$

Fig. 7.12.: Silhouette and reconstruction images are overlaid for reconstructions using the uncorrected camera calibration parameters (top row) versus reconstructions generated with multiple pass corrected external camera calibration parameters (bottom row). Color coding as in Figure 7.10.



(a) $e = 10$ (b) $e = 30$ (c) $e = 40$ (d) $e = 50$

(e) $e = 10$ (f) $e = 30$ (g) $e = 40$ (h) $e = 50$

Fig. 7.13.: Silhouette and reconstruction images are overlaid for reconstructions using the uncorrected camera calibration parameters (top row) versus reconstructions generated with multiple pass corrected external camera calibration parameters (bottom row). Color coding as in Figure 7.10.

7.4 Results of the Camera Calibration Correction for the Model Tree and Real Datasets

Table 7.2: The Silhouette Inconsistency error (SIE) of the seven datasets

Dataset	SIE, no correction	SIE with $\mathbf{R}$, $\mathbf{t}$ correction	SIE with $\mathbf{R}$, $\mathbf{t}$, $\mathbf{K}$ correction
Model Tree	2,044,618	946,104	937,490
Branch	171,346	124,351	109,984
Coil	182,176	104,806	99,195
Coil and Cables	331,743	187,255	161,649
Weeping Apple E	3,001,539	2,036,888	1,739,336
Standard Apple E	1,969,238	1,149,083	1,029,106
Pole and Coil E	711,988	363,249	302,940

In this section, we show the differences in using the two camera calibration correction scenarios on synthetic and real datasets.

We first show the effect of camera calibration correction on SIE values, as shown in Table 7.2. Recall that the value of SIE roughly indicates the number of pixels that do not match between the input silhouette and the image of the reconstructed shape. Also, note that since we use a voxel-based method, the SIE is never zero except for the case of perfect camera calibration, perfect segmentation, and infinitely-small voxels.

Given all of these preliminaries, we can see from the Table 7.2 that correcting only the external parameters results in a great decrease in the value of SIE, when compared to the uncorrected results. For these datasets a reduction of 33% or more in SIE values can be gained by the camera calibration correction scenarios. We also notice that the SIE with internal and external parameters corrected is always lower than that only correcting external parameters.

The reconstructions of the seven datasets are displayed using no correction, external parameter correction, and external and internal parameter correction in Figures 7.14-7.28.

We discuss the synthetic dataset MODEL TREE first, in Figures 7.14 and an example reconstruction image in Figure 7.15. We can see in this figure that the uncorrected SfIS reconstruction in Figure 7.14b is quite noisy, and the detail of small branches is largely lost. However, in the reconstructions using corrected parameters (Figures 7.14c and 7.14d), the reconstruction more faithfully represents the ground truth model, though there are some noisy regions remaining for small branches. For the example image shown in Figure 7.15, use of the external and internal camera calibration correction produces closer alignment with the original image than use of the external parameter correction alone.

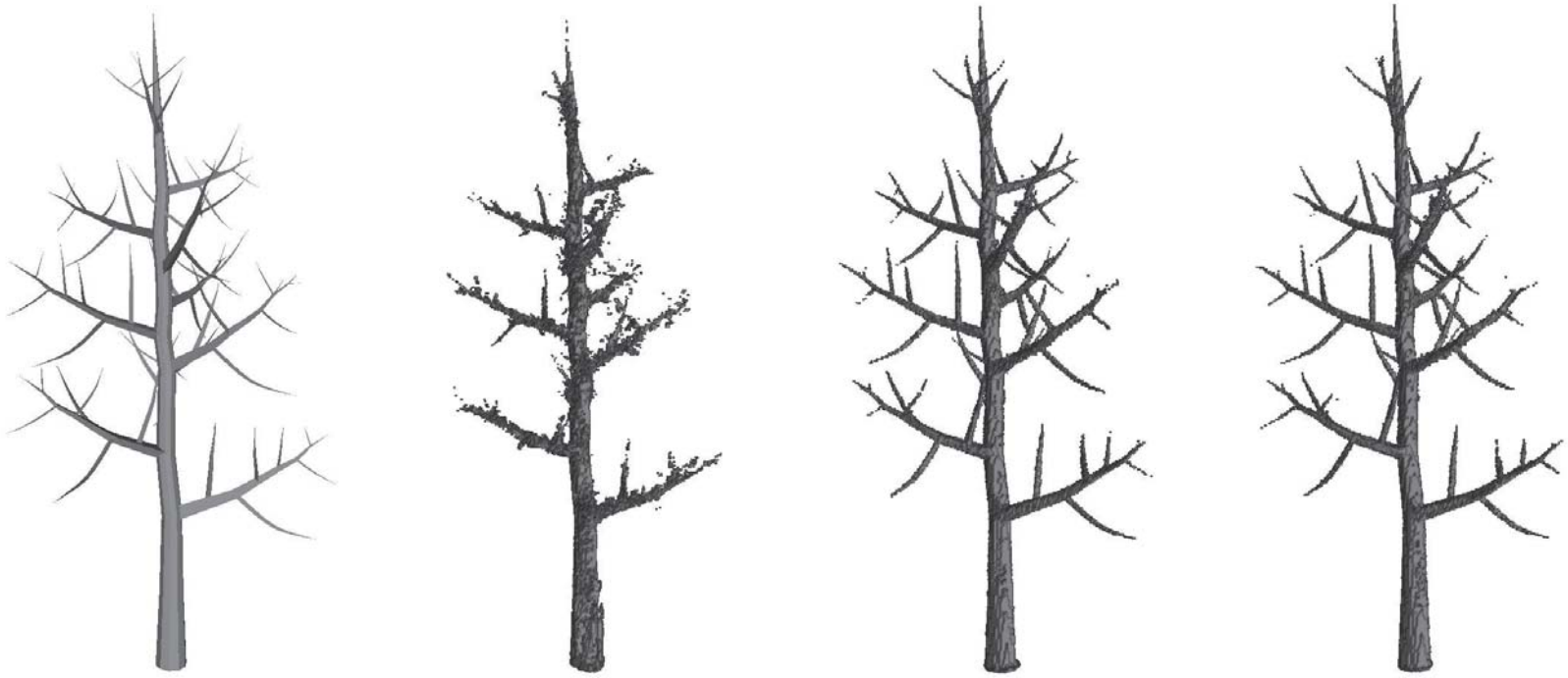
Table 7.3: Reconstruction accuracy as compared to a voxelated ground truth of the MODEL TREE dataset

Reconstruction	FP	FN
No correction	0.000504248	0.400538
$\mathbf{R}$, $\mathbf{t}$ correction	0.000412875	0.178787
$\mathbf{R}$, $\mathbf{t}$, $\mathbf{K}$ correction	0.000524409	0.175531

In Table 7.3, we show classification rates of the reconstructions as compared to a voxel-version of the ground truth model. In this table, 'FP' is false positive (in our context, a positive is an occupied voxel) and 'FN' is false negative. This table shows that the false negative rate decreases by 22 % when using either one of the corrections, while the false positive rate remains largely the same.

There is very little difference in appearance, number of misclassifications, and SIE between the reconstructions using the two types of corrections. For the MODEL TREE dataset, error is only induced to the translation component. There are slight differences between the reconstructions resulting from only the external camera calibration correction versus external and internal camera calibration correction. Matching projected voxels to image silhouettes introduces discretization artefacts, which results in some alteration of the internal parameters even when internal parameters have no

error; we can see for correcting both internal and external parameters results in a smaller SIE as compared to only correcting external parameters (Table 7.2: 937,490 to 946,104, respectively).



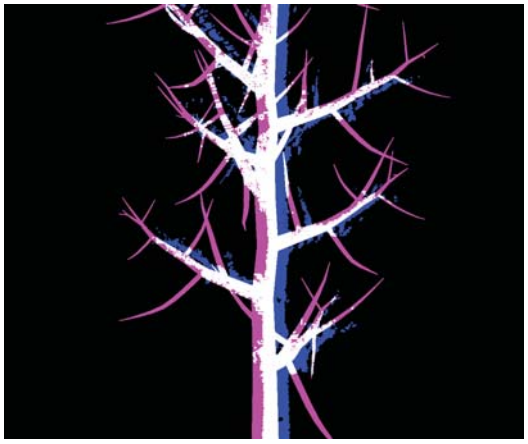
(a) Ground truth

(b) Without any camera calibration correction

(c) With correction of external parameters

(d) With correction of internal and external parameters

Fig. 7.14.: The reconstruction and camera calibration correction of the MODEL TREE dataset, under the two different scenarios.



(a) Silhouette and reconstruction images overlaid; uncorrected.

(b) External parameters corrected.

(c) External and internal parameters corrected.

Fig. 7.15.: Overlay of silhouette and reconstructions for the Model Tree dataset. Color coding as in Figure 7.10.

Considering the Branch, Coil, and Coil and Cables datasets, we can observe in reconstructions shown in Figures 7.16, 7.18, and 7.20 similar behavior as the Model Tree dataset: there are more erroneously-labeled voxels in the uncorrected reconstruction than in the corrected reconstruction. Since the objects in the dataset are very thin, in the uncorrected reconstructions there are breaks in the surface where it is continuous in the original object. The corrected reconstructions tend to repair these breaks and reduce noisy regions as well. This is especially true for the Coil and Cables dataset, where the small diameter wire, in the uncorrected reconstruction,

is broken into many pieces, but is connected in the corrected reconstruction (Figure 7.20). Figure 7.22 shows the looped thin cable (on the left side of the coil in Figure 7.20).

The use of external parameter correction produces reconstructions that are more representative than reconstructions generated without the correction. For this dataset, the best reconstructions were generated by using the correction of both internal and external parameters.

Original and reconstruction images are shown in Figures 7.17, 7.19, and 7.21. As with the Model Tree *e* datasets, even when the alignment quality is not accurate for some images as illustrated in Figure 7.17, left hand side, if enough other images are accurate the reconstruction method can compensate for those problems. In the other images, Figures 7.19, and 7.21, we can see the extent of the calibration error for the overhead images on the left hand sides of the figures.

The reconstructions of the second group of real datasets, Weeping Apple E, Standard Apple E, and Pole and Coil E are shown in Figures 7.23, 7.26, and 7.28. The first tree, in the Weeping Apple E dataset, is considered to have a 'weeping' form, and it has many small branches, whereas Standard Apple E has a more upright form. Figures 7.24 and 7.27 show overlays of the original silhouette and reconstruction images; we can see that the uncorrected reconstructions are sligtly offset from the silhouette images. The object in the Pole and Coil E dataset consists of a metal pole with a coil attached using a zip tie; there are four thumb screws along the pole's length. For all three of these datasets, there is a great deal of improvement in the reconstructions representing the original object when the camera calibration parameters are corrected. There is little difference between the two types of correction, but using a correction of the external and internal parameters seem to produce the best results, with more small details reconstructed. For instance, in Figure 7.28, the narrow plastic tie which holds the coil to the pole is reconstructed when all parameters are corrected and is not when only the external parameters are corrected.

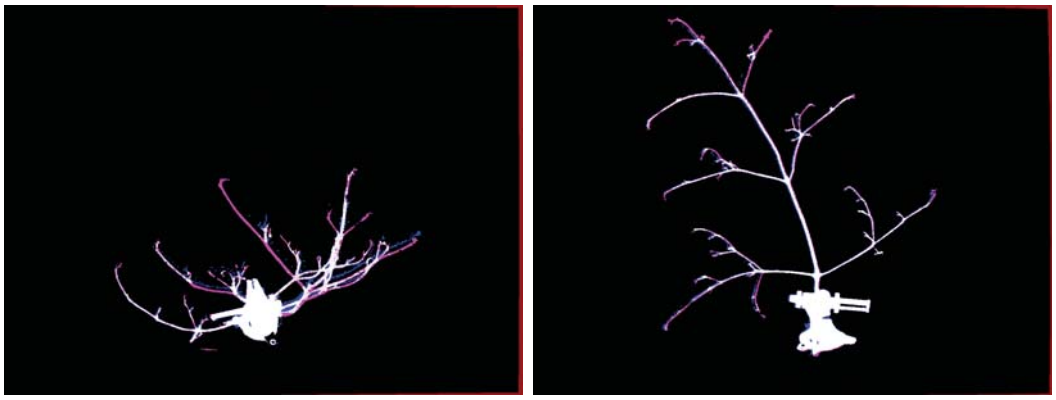
This behavior can also be seen by examining the silhouette and reconstruction image overlays in Figure 7.29.

(a) Without any camera calibration correction

(b) With correction of external parameters

(c) With correction of internal and external parameters

Fig. 7.16.: The reconstruction of the BRANCH dataset

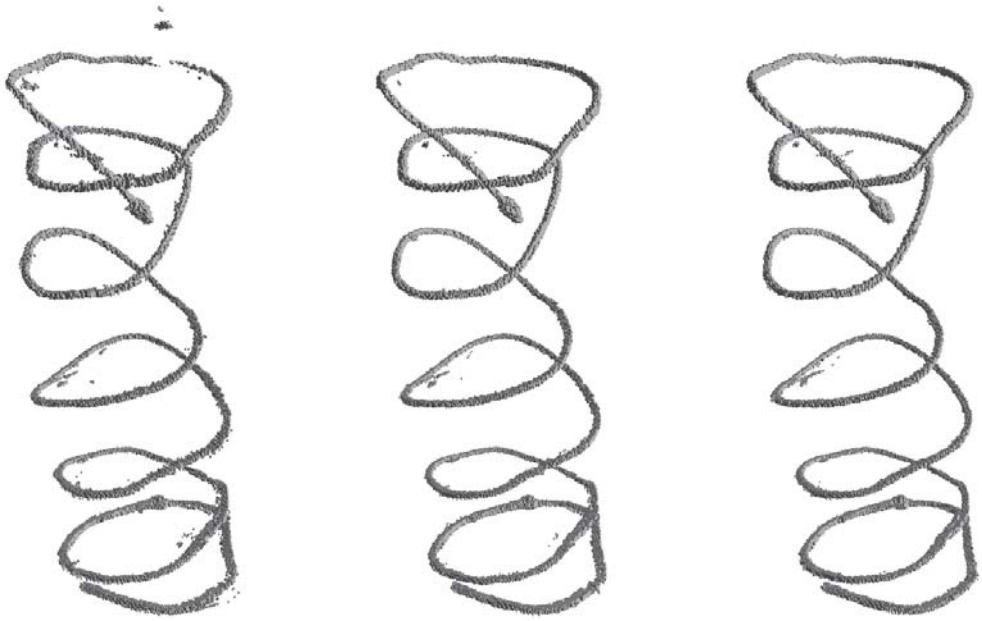


(a) Silhouette and reconstruction images overlaid; uncorrected.

(b) External parameters corrected.

(c) External and internal parameters corrected.

Fig. 7.17.: Overlay of silhouette and reconstructions for the BRANCH dataset, two images. Color coding as in Figure 7.10.

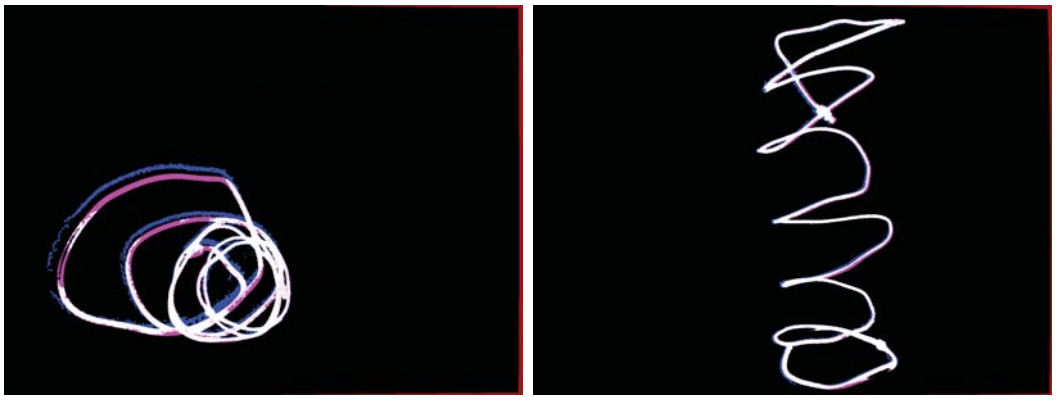


(a) Without any camera calibration correction

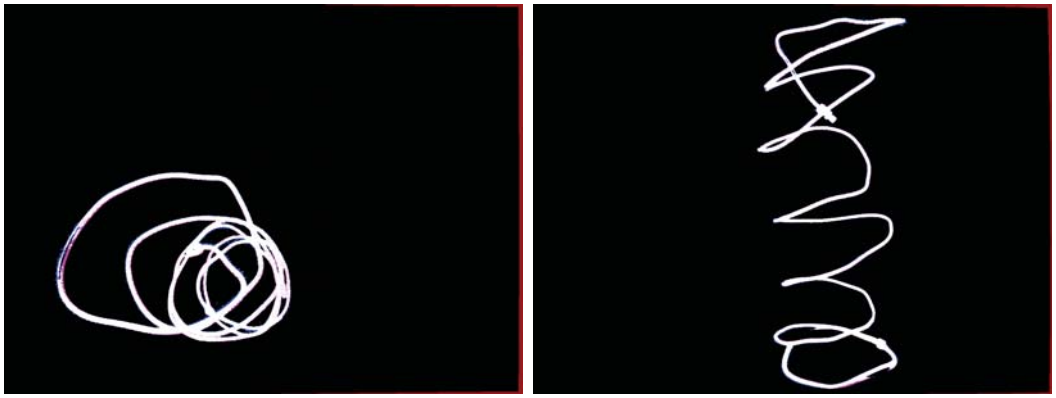
(b) With correction of external parameters

(c) With correction of external and internal parameters

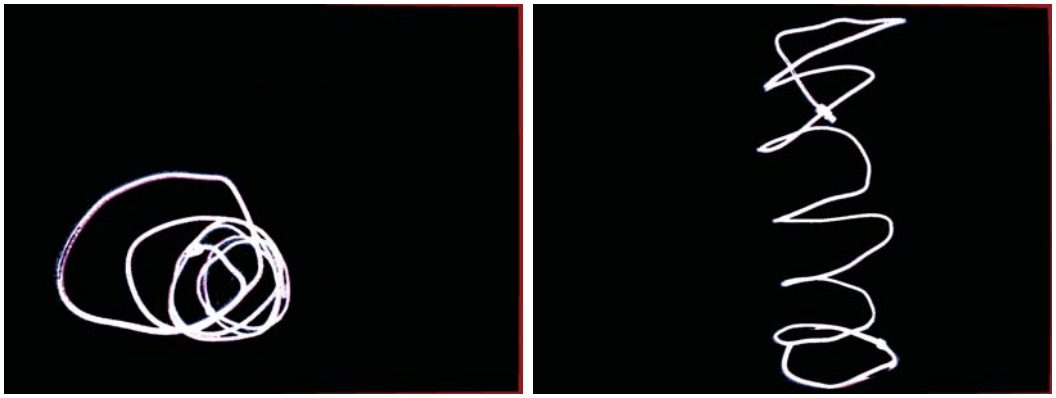
Fig. 7.18.: The reconstruction of the Coil dataset



(a) Silhouette and reconstruction images overlaid; uncorrected.

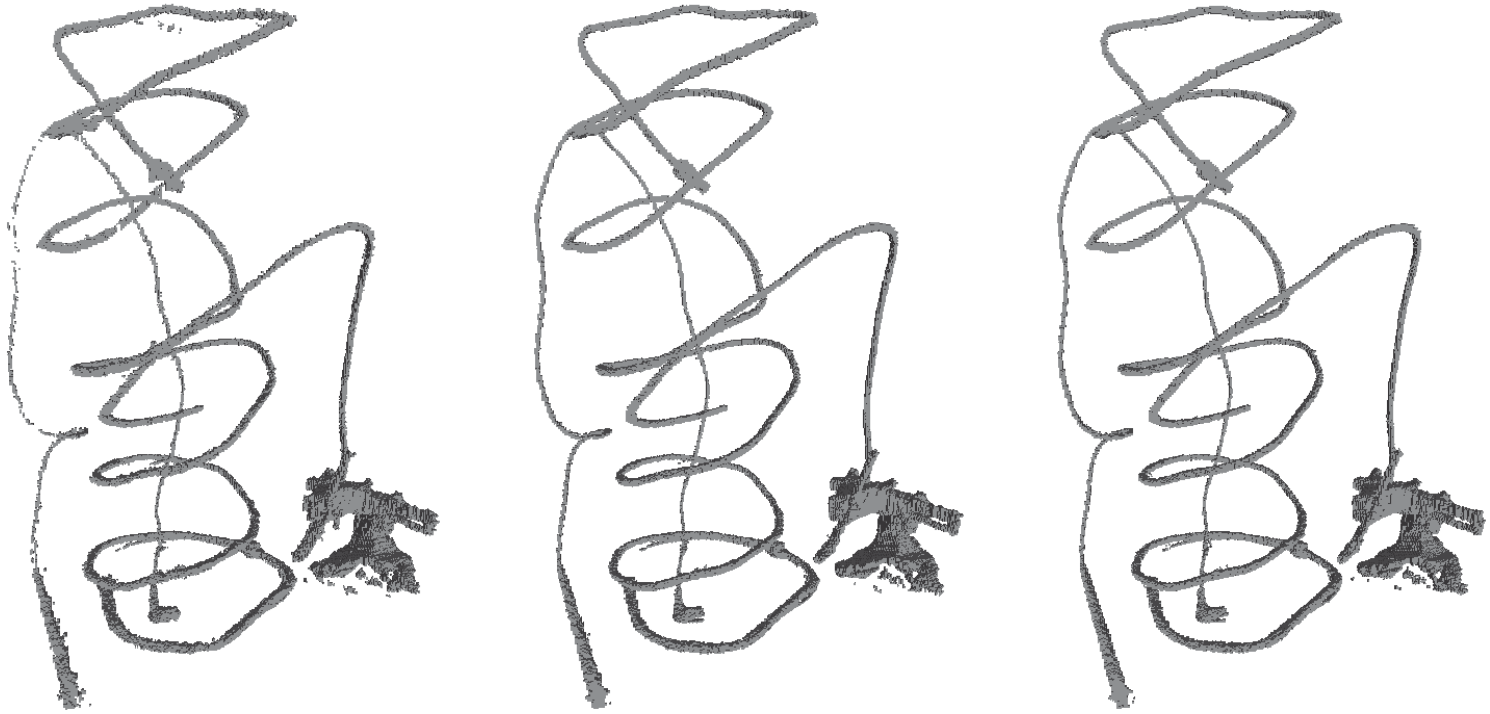


(b) External parameters corrected.



(c) External and internal parameters corrected.

Fig. 7.19.: Overlay of silhouette and reconstructions for the COIL dataset, two images. Color coding as in Figure 7.10.

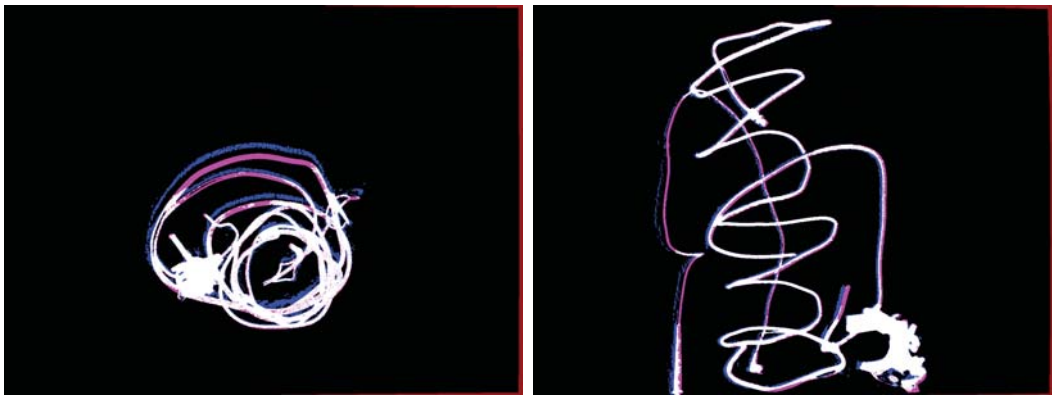


(a) Without any camera calibration correction

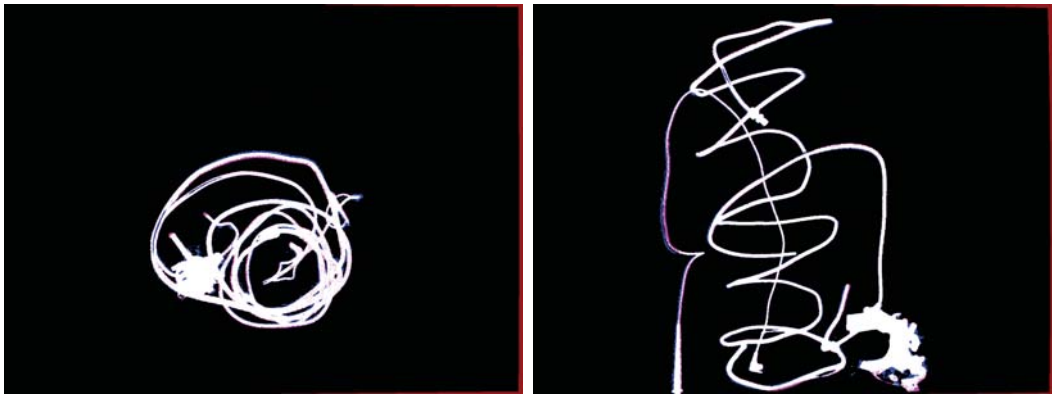
(b) With correction of external parameters

(c) With correction of external and internal parameters

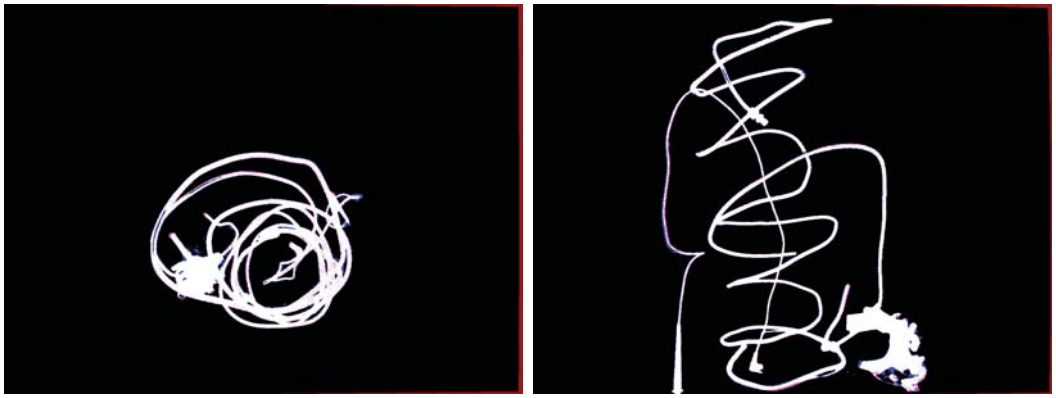
Fig. 7.20.: The reconstruction of the Coil and Cables dataset



(a) Silhouette and reconstruction images overlaid; uncorrected.



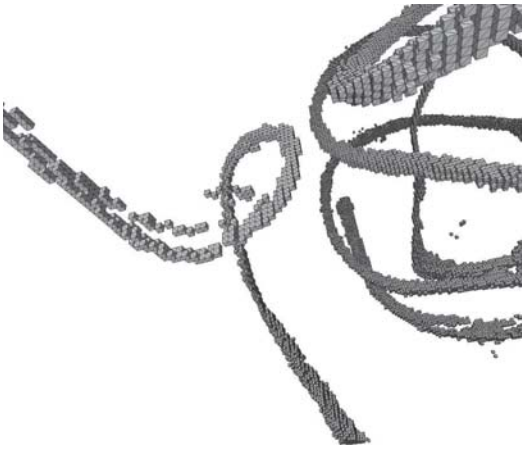
(b) External parameters corrected.



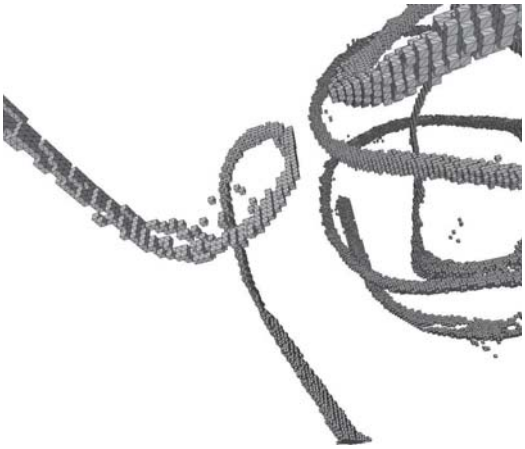
(c) External and internal parameters corrected.

Fig. 7.21.: Overlay of silhouette and reconstructions for the Coil and Cables dataset, two images. Color coding as in Figure 7.10.

(a) Without any camera calibration correction



(b) With correction of external parameters



(c) With correction of external and internal parameters

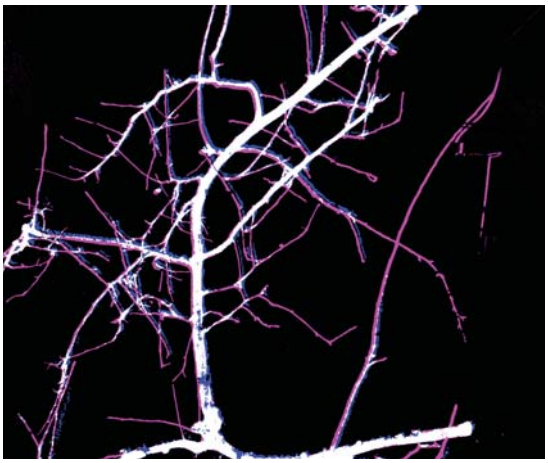
Fig. 7.22.: Detail of the Coil and Cables dataset's reconstruction

(a) Without any camera calibration correction

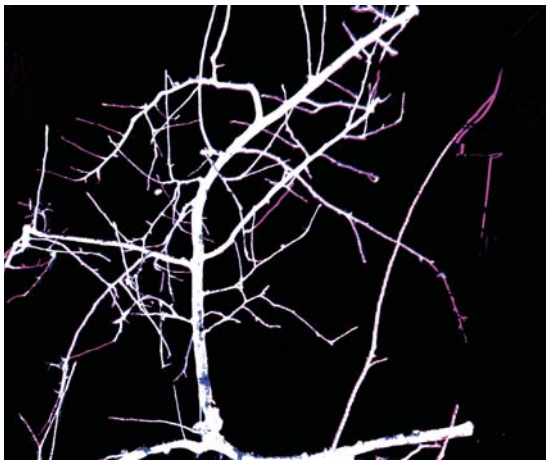
(b) With correction of external parameters

(c) With correction of external and internal parameters

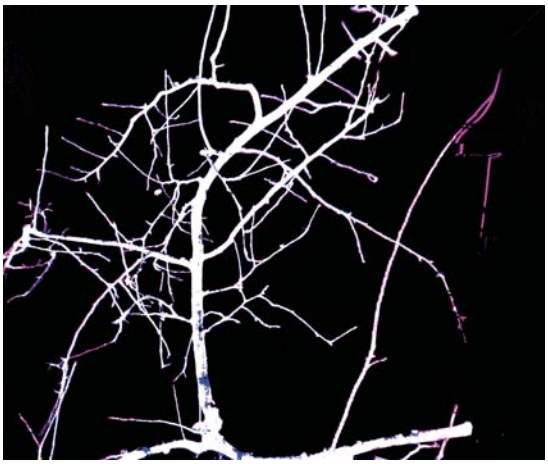
Fig. 7.23.: The reconstruction of the Weeping Apple E dataset



(a) Silhouette and reconstruction images overlaid; uncorrected.



(b) External parameters corrected.



(c) External and internal parameters corrected.

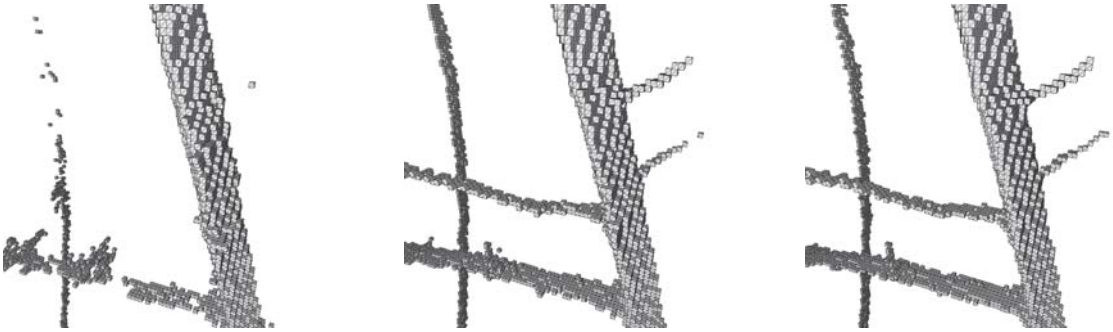
Fig. 7.24.: Overlay of silhouette and reconstructions for the Weeping Apple E dataset, two images. Color coding as in Figure 7.10.

(a) Without any camera calibration correction

(b) With correction of external parameters

(c) With correction of external and internal parameters

Fig. 7.25.: The reconstruction of the Standard Apple E dataset



(a) Without any camera calibration correction

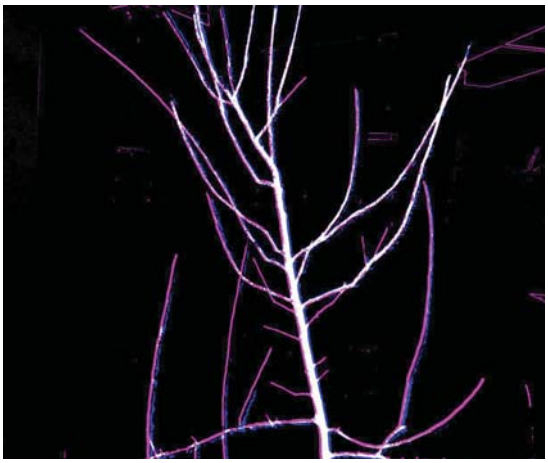
(b) With correction of external parameters

(c) With correction of external and internal parameters

Fig. 7.26.: Detail of the STANDARD APPLE E dataset's reconstruction

7.5 Conclusions about Camera Calibration Correction

After examining the behavior of the camera calibration correction method, we concluded that there is little to lose by performing the correction with both external and internal camera parameters. When internal camera calibration parameters are somewhat poor, performing the full correction results in more representative reconstructions (datasets BRANCH, COIL, and COIL AND CABLES). On the other hand, the similarity of the two types of corrections for the synthetic datasets show that performing the full correction will not degrade the reconstruction, even when it is known that the internal parameters were not perturbed by error. The reason for this is that our acceptance of updated camera calibration parameters is dependent on a lower SIE score than the score gained with the current parameters. This requirement prevents the calibration from deviating significantly from the true calibration.

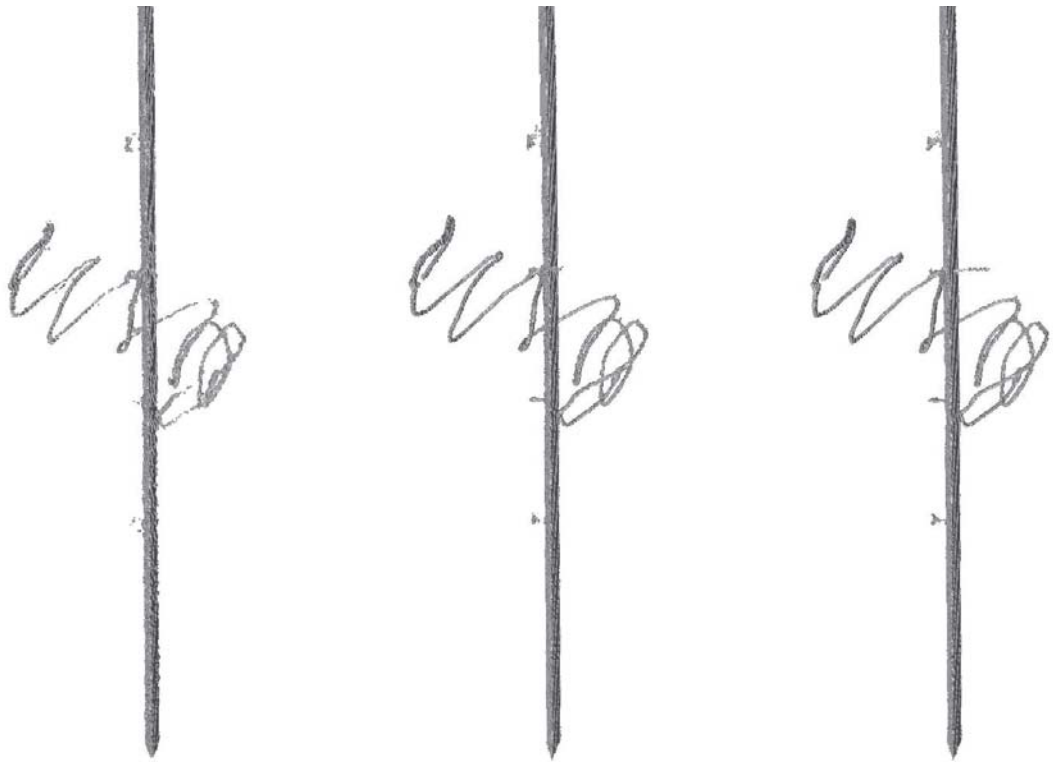


(a) Silhouette and reconstruction images overlaid; uncorrected.

(b) External parameters corrected.

(c) External and internal parameters corrected.

Fig. 7.27.: Overlay of silhouette and reconstructions for the Standard Apple E dataset. Color coding as in Figure 7.10.

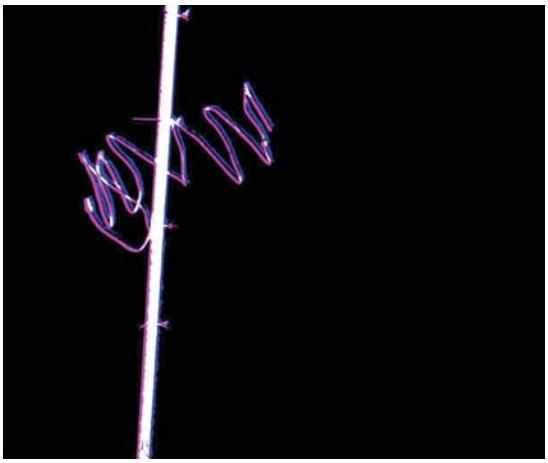


(a) Without any camera calibration correction

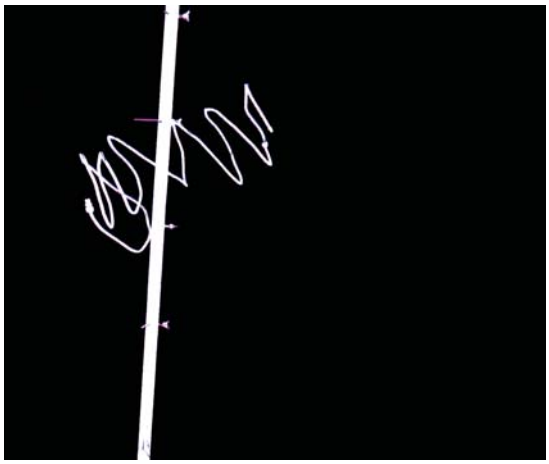
(b) With correction of external parameters

(c) With correction of external and internal parameters

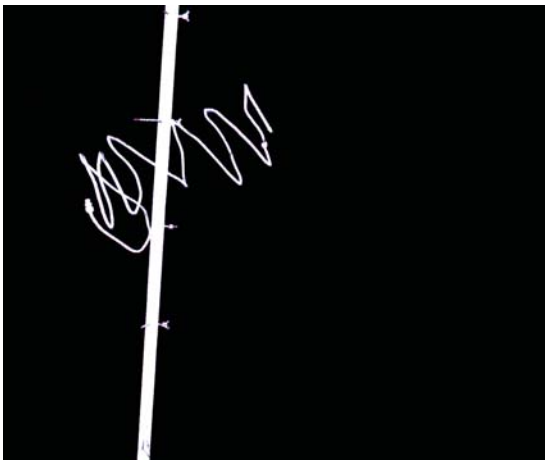
Fig. 7.28.: The reconstruction of the Pole and Coil E dataset



(a) Silhouette and reconstruction images overlaid; uncorrected.



(b) External parameters corrected.



(c) External and internal parameters corrected.

Fig. 7.29.: Overlay of silhouette and reconstructions for the Pole and Coil E dataset. Color coding as in Figure 7.10.

8. CONCLUSIONS AND FUTURE WORK

In this dissertation, we explored the problem of reconstructing objects from silhouettes in the presence of silhouette segmentation and camera calibration error. We formulated the reconstruction problem as a pseudo-Boolean optimization problem and gave some local minimum search methods for finding approximate solutions to the optimization problem. The local minimum search methods were demonstrated on a variety of challenging objects and compared to the state-of-the-art. We concluded that our method was able to generate representative reconstruction of challending objects and performed well in comparisons to other methods.

In this document, we did not perform any post-processing on the reconstructions. A post-processing step tuned to the application may result in better reconstructions.

We also developed a method for correcting camera calibration error in the context of SfIS. We presented an ICP-based approach that alters camera parameters such that the image of the reconstruction and the silhouette images are aligned. To our knowledge, ours is the only method available that is able to correct camera calibration error with partial silhouettes, general camera motion, and segmentation error without human assistance. We showed that our camera calibration method improves the reconstruction accuracy of SfIS reconstructions, in particular the reconstruction of small features.

Our future work consists of extending the methods in this document to operate on larger agricultural objects in an outdoor environment. As alluded to in our document, the management of computational resources becomes more critical as the number of voxels grow. In addition, some applications may require a different approach to balance accuracy with computational time. For instance, for a fruit pruning application, accuracy is most important near the central leader of the trunk and computational time must be low for real-time operation. For the phenotyping application, the re-

construction of tips of the branches may be more important. Consequently, different versions of the reconstruction methods may be required depending on the application.

LIST OF REFERENCES

LIST OF REFERENCES

[1] S. Seitz and C. Dyer, "Photorealistic scene reconstruction by voxel coloring," *International Journal of Computer Vision*, vol. 35, no. 2, pp. 151–173, 1999.

[2] K. N. Kutulakos and S. M. Seitz, "A theory of shape by space carving," *Int. J. Comput. Vision*, vol. 38, no. 3, pp. 199–218, 2000.

[3] G. Vogiatzis, C. Hernandez, P. H. S. Torr, and R. Cipolla, "Multiview stereo via volumetric graph-cuts and occlusion robust photo-consistency," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 12, pp. 2241–2246, 2007.

[4] S. Sinha and M. Pollefeys, "Multi-view reconstruction using photo-consistency and exact silhouette constraints: a maximum-flow formulation," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 1, pp. 349–356 Vol. 1, 2005.

[5] V. Kolmogorov and R. Zabih, "Multi-camera scene reconstruction via graph cuts," in *Computer Vision ECCV 2002* (A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, eds.), vol. 2352 of *Lecture Notes in Computer Science*, pp. 82–96, Springer Berlin Heidelberg, 2002.

[6] A. Laurentini, "The visual hull concept for silhouette-based image understanding," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 2, pp. 150–162, 1994.

[7] C. Chien and J. Aggarwal, "Volume/surface octrees for the representation of three-dimensional objects," *Computer Vision Graphics and Image Processing*, vol. 36, pp. 100–113, 1986.

[8] J.-S. Franco, *Three-dimensional modeling from silhouettes*. PhD thesis, Institut National Polytechnique de Grenoble, 2005.

[9] J.-S. Franco and E. Boyer, "Efficient polyhedral modeling from silhouettes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, pp. 414–427, March 2009.

[10] W. Martin and J. Aggarwal, "Volumetric description of objects from multiple views," *IEEE Transactions of Pattern Analysis and Machine Intelligence*, vol. 5, no. 3, pp. 150–158, 1983.

[11] M. Potmesil, "Generating octree models of 3d objects from their silhouettes in a sequence of images," *Comput. Vision Graph. Image Process.*, vol. 40, no. 1, pp. 1–29, 1987.

[12] R. Szeliski, "Rapid octree construction from image sequences," *CVGIP: Image Underst.*, vol. 58, no. 1, pp. 23–32, 1993.

[13] J.-L. Landabaso, M. Pardàs, and J. R. Casas, "Shape from inconsistent silhouette," *Computer Vision and Image Understanding*, vol. 112, no. 2, pp. 210 – 224, 2008.

[14] A. Tabb, "Shape from silhouette probability maps: reconstruction of thin objects in the presence of silhouette extraction and calibration error," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2013.

[15] P. Giblin and R. Weiss, "Reconstruction of surfaces from profiles," in *Proceedings of the First International Conference on Computer Vision*, (Washington, DC, USA), pp. 136–144, IEEE Computer Society, 1987.

[16] R. Cipolla and A. Blake, "Surface shape from the deformation of apparent contours," *Int. J. Comput. Vision*, vol. 9, no. 2, pp. 83–112, 1992.

[17] R. Cipolla, H. C. Longuet-Higgins, P. Giblin, P. H. S. Torr, O. Faugeras, A. Fitzgibbon, G. Hunter, A. Hopper, K. Stark, M. Isard, and T. Ihle, "The visual motion of curves and surfaces [and discussion]," *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, vol. 356, no. 1740, pp. 1103–1121, 1998.

[18] P. J. Giblin and R. S. Weiss, "Epipolar curves on surfaces," *Image and Vision Computing*, vol. 13, no. 1, pp. 33 – 44, 1995.

[19] R. Vaillant and O. Faugeras, "Using extremal boundaries for 3-d object modeling," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 14, pp. 157–173, Feb 1992.

[20] R. Szeliski and R. Weiss, "Robust shape recovery from occluding contours using a linear smoother," *Int. J. Comput. Vision*, vol. 28, no. 1, pp. 27–44, 1998.

[21] E. Boyer and M.-O. Berger, "3d surface reconstruction using occluding contours," *Int. J. Comput. Vision*, vol. 22, no. 3, pp. 219–233, 1997.

[22] S. Lazebnik, "Projective visual hulls," Master's thesis, University of Illinois at Urbana-Champaign, 2002.

[23] S. Lazebnik, E. Boyer, and J. Ponce, "On computing exact visual hulls of solids bounded by smooth surfaces," *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, pp. I–156–I–161 vol.1, 2001.

[24] S. Lazebnik and J. Ponce, "The local projective shape of smooth surfaces and their outlines," *Int. J. Comput. Vision*, vol. 63, no. 1, pp. 65–83, 2005.

[25] S. Lazebnik, Y. Furukawa, and J. Ponce, "Projective visual hulls," *Int. J. Comput. Vision*, vol. 74, no. 2, pp. 137–165, 2007.

[26] J.-S. Franco and E. Boyer, "Fusion of Multi-View Silhouette Cues Using a Space Occupancy Grid," in *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision*, (Beijing, China), pp. 1747–1753, IEEE Computer Society, 2005.

[27] L. Díaz-Más, R. Muoz-Salinas, F. Madrid-Cuevas, and R. Medina-Carnicer, "Shape from silhouette using dempstershafer theory," *Pattern Recognition*, vol. 43, no. 6, pp. 2119 – 2131, 2010.

[28] L. Guan, J.-S. Franco, and M. Pollefeys, "3D Occlusion Inference from Silhouette Cues," in *CVPR '07: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, (United States), pp. 1–8, June 2007.

[29] G. K. Cheung, T. Kanade, J.-Y. Bouguet, and M. Holler, "A real time system for robust 3d voxel reconstruction of human motions," *CVPR '00: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, vol. 2, p. 2714, 2000.

[30] D. Snow, P. Viola, and R. Zabih, "Exact voxel occupancy with graph cuts," in *CVPR '00: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition.*, vol. 1, pp. 345–352 vol.1, 2000.

[31] G. Haro and M. Pardàs, "Shape from incomplete silhouettes based on the reprojection error," *Image and Vision Computing*, vol. 28, no. 9, pp. 1354 – 1368, 2010.

[32] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222–1239, 2001.

[33] P. Hammer, P. Hansen, and B. Simeone, "Roof duality, complementation and persistency in quadratic 01 optimization," *Mathematical Programming*, vol. 28, pp. 121–155, 1984. 10.1007/BF02612354.

[34] E. Boros and P. L. Hammer, "Pseudo-boolean optimization," *Discrete Applied Mathematics*, vol. 123, no. 1-3, pp. 155 – 225, 2002.

[35] E. Boros, P. L. Hammer, and G. Tavares, "Preprocessing of unconstrained quadratic binary optimization," Tech. Rep. 10, RUTCOR, April 2006.

[36] V. Kolmogorov and C. Rother, "Minimizing nonsubmodular functions with graph cuts-a review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 1274–1279, July 2007.

[37] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts?," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, pp. 147–159, Feb. 2004.

[38] I. Rosenberg, "Reduction of bivalent maximization to the quadratic case," *Cahirs Centre Etudes Rech. Oper.*, vol. 17, pp. 71–74, 1975.

[39] H. Ishikawa, "Transformation of general binary mrf minimization to the first order case," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 1234–1249, 2010.

[40] D. Freedman and P. Drineas, "Energy minimization via graph cuts: settling what is possible," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, pp. 939 – 946 vol. 2, june 2005.

[41] C. Hernández, F. Schmitt, and R. Cipolla, "Silhouette coherence for camera calibration under circular motion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 2, pp. 343–349, 2007.

[42] P.-H. Huang and S.-H. Lai, "Silhouette-based camera calibration from sparse views under circular motion," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, June 2008.

[43] P. Mendonca, K. Y. K. Wong, and R. Cippolla, "Epipolar geometry from profiles under circular motion," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, pp. 604–616, Jun 2001.

[44] H. Zhang and K. Y. K. Wong, "Self-calibration of turntable sequences from silhouettes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, pp. 5–14, Jan 2009.

[45] K.-Y. Wong and R. Cipolla, "Reconstruction of sculpture from its profiles with unknown camera positions," *Image Processing, IEEE Transactions on*, vol. 13, pp. 381–389, March 2004.

[46] K. Åström, R. Cipolla, and P. Giblin, "Generalised epipolar constraints," *International Journal of Computer Vision*, vol. 33, no. 1, pp. 51–72, 1999.

[47] E. Boyer, "On Using Silhouettes for Camera Calibration," in *7th Asian Conference on Computer Vision (ACCV '06)* (P. J. Narayanan, S. K. Nayar, and H.-Y. Shum, eds.), vol. 3851/2006 of *Lecture Notes in Computer Science (LNCS)*, (Hyderabad, India), pp. 1–10, Springer-Verlag, 2006.

[48] Y. Furukawa, A. Sethi, J. Ponce, and D. Kriegman, "Robust structure and motion from outlines of smooth curved surfaces," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, pp. 302–315, Feb 2006.

[49] H. Yamazoe, A. Utsumi, and S. Abe, "Multiple camera calibration with bundled optimization using silhouette geometry constraints," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 3, pp. 960–963, 2006.

[50] X. Zhang, Y. Zhang, X. Zhang, T. Yang, X. Tong, and H. Zhang, "A convenient multi-camera self-calibration method based on human body motion analysis," in *Image and Graphics, 2009. ICIG '09. Fifth International Conference on*, pp. 3–8, Sept 2009.

[51] S. Sinha, M. Pollefeys, and L. McMillan, "Camera network calibration from dynamic silhouettes," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1, pp. I–195–I–202 Vol.1, June 2004.

[52] R. Cipolla, K. Åström, and P. Giblin, "Motion from the frontier of curved surfaces," in *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pp. 269–275, Jun 1995.

[53] Y. Furukawa and J. Ponce, "Accurate camera calibration from multi-view stereo and bundle adjustment," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, June 2008.

[54] J. Besag, "On the statistical analysis of dirty pictures," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 48, no. 3, pp. 259–302, 1986.

[55] D. M. Greig, B. T. Porteous, and A. H. Seheult, "Exact maximum a posteriori estimation for binary images," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 51, no. 2, pp. pp. 271–279, 1989.

[56] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparative study of energy minimization methods for markov random fields with smoothness-based priors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 6, pp. 1068–1080, 2008.

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

[58] J.-S. Franco and E. Boyer, "Exact polyhedral visual hulls," in *in British Machine Vision Conference (BMVC03*, pp. 329–338, 2003.

[59] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pp. 145–152, 2001.

[60] E. K. Chong and S. H. Zak, *An introduction to optimization*. John Wiley & Sons, 3rd ed., 2011.

[61] Meshlab, "Developed with the support of the 3d-coform project." http://meshlab.sourceforge.net/.

[62] G. Turk and M. Levoy, "Zippered polygon meshes from range images," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, SIGGRAPH '94, (New York, NY, USA), pp. 311–318, ACM, 1994.

[63] Z. Zhang, "A flexible new technique for camera calibration," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, pp. 1330–1334, Nov 2000.

[64] J. Lespinasse, *La conduite du pommier. IIL'axe vertical, la rnovation des vergers (2me partie)*. Paris: Ctifl, 1980.

[65] G. Cheung, *Visual hull construction, alignment, and refinement for human kinematic modeling, motion tracking and rendering*. PhD thesis, Carnegie Mellon University, 2003.

[66] M. Lourakis, "levmar: Levenberg-marquardt nonlinear least squares algorithms in C/C++." [web page] `http://www.ics.forth.gr/~lourakis/levmar/`, Jul. 2004. [Accessed on 31 Jan. 2005.].

[67] R. Hirsh, G. DeSouza, and A. Kak, "An iterative approach to the hand-eye and base-world calibration problem," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 3, pp. 2171–2176 vol.3, 2001.

APPENDIX

A. IMPLEMENTATION OF COMPARISON METHODS

In this appendix, we give more detail as to how the comparison methods are implemented. Throughout, we have sought that our implementation be as close to the original paper's implementation, so far as can be determined by reading the papers which describe the other methods. However, since all of the comparison methods depend on the setting of one or more parameters, using either trial and error or previous knowledge, it was challenging to implement the comparison methods exactly as their authors intended and in some of the methods implementation details were sparse. We note differences in between our and the original method in the text.

Throughout all of these methods, we have tried to standardize the notation with our own work. In the Graph Cuts method, our notation reflects more standard notation in the energy minimization community within computer vision in the 2000s.

A.1 Real Robust Visual Hull

To describe the Real Robust Visual Hull, of RRVH, we first start with the Robust Visual Hull. While the Robust Visual Hull is not based on a particular paper, it is a commonly-used method of comparison such as in [13], [30] using the heuristic that if the intersection of N silhouettes cones result in missing regions, then the intersection of $N-1$, $N-2$, $N-k$ etc. silhouettte cones may give acceptable results for a very low computational cost.

We considered that a voxel was inside the silhouette region of a camera if all pixels projected to the image plane were silhouette pixels. In the robust visual hull, a voxel is marked as occupied if the number of cameras where the voxel is classified as silhouette, s is greater than or equal to some number $N-k$.

In some of our datasets, not all cameras view all voxels. To deal with this dataset characteristic, we computed the percentage of cameras that classified a voxel as silhouette out of all cameras that could view the voxel. If n_c is the number of cameras that view a voxel, this percentage is s/n_c. A threshold $m \in [0, 1]$ is chosen and if $s/n_c \geq m$ then the voxel is marked occupied.

We determined the ideal threshold m for the comparisons with a ground truth by exhaustively searching the interval $[0, 1]$ in 0.05 increments. The threshold that resulted in the smallest number of misclassifications as compared to the ground truth was chosen as m.

A.2 SPOT

The SPOT (Sparse pixels occupancy test) method by Cheung et al in [29] and [65] was demonstrated on human subjects in a survelliance or tracking environment. The algorithm's goals are to speed up the standard visual hull algorithm as well as reduce the effects of segmentation noise on the reconstruction. It is assumed the two parameters η and ξ are known, where η represents the the pixel false negative probability, while ξ is the pixel false postive probability. In the original publications, η and ξ are determined experimentally from video sequences of images.

When a voxel is projected to the image plane it covers a particular number of pixels N. The SPOT algorithm consists of selecting only Q out of S pixels to test and storing the projection information for a specific choice of pixels in a lookup table for speed-related purposes. To deal with noise, once Q is set, a number $Q_\epsilon \leq Q$ is chosen. In SPOT, a voxel is considered within the silhouette of a particular camera if the number of silhouette pixels s is $s \geq Q_\epsilon$. Then, a voxel is considered occupied if all cameras have $s \geq Q_\epsilon$ and empty otherwise.

Q_ϵ is chosen such that the value of a cost function is minimized, by exhaustively iterating through all Q possibilities. The cost function is composed of two parts: the

probability of false positives (FP) and false negatives (FN) for voxels, shown in Eq.s A.1 and A.3.

$$P(FP)_{SPOT} = \big[\sum_{i=Q_\epsilon}^{Q} \binom{Q}{i} \xi^i (1-\xi)^{Q-i}\big]^N \tag{A.1}$$

$$\rho = \sum_{i=Q-Q_\epsilon+1}^{Q} \binom{Q}{i} \eta^i (1-\eta)^{Q-i} \tag{A.2}$$

$$P(FN)_{SPOT} = \rho \sum_{j=0}^{N-1} (1-\rho)^j \tag{A.3}$$

Consequently, Q_ϵ is chosen:

$$Q_\epsilon = min_{[0,\ldots,Q]} P(FP) + P(FN) \tag{A.4}$$

In our work, we use complete projection, meaning that the entire voxel is projected to the image plane and all pixels are used, not just a subset like in SPOT. Furthermore, we did not want to skew the results negatively with a randomly bad selection of Q. Finally, the number of pixels for each projected voxel varies according to where the voxel is with respect to the cameras viewing it.

To deal with all of these issues, we implemented SPOT in the following way. We project the whole voxel to the cameras that are able to view the voxel. We consider a voxel to be in the silhouette for a camera if at least Q_ϵ pixels are silhouette pixels; if all pixels are silhouette pixels, even if $n_s \leq Q_\epsilon$ we consider that the voxel is in the silhouette for that camera.

Since η and ξ are unknown for all except one dataset, we iterate over a set of η, ξ combinations, where $\eta \in [0.05, 0.2]$ and $\xi \in [0.05, 0.2]$. For datasets where the ground truth is known, we then choose the parameter combination that resulted in the smallest number of misclassifications as compared to the ground truth.

A.3 Unbiased Hull

This work by Landabaso et al in [13] consists of two parts. The first is called the 'Sampled pixels projection test', or SPPT as we abbreviate it here. The SPPT uses the same mechanism as SPOT to select Q_ϵ, except with a different cost function. Like SPOT, it is assumed that the pixel false negative probability η and pixel false positive probability ξ are known.

However, the form of $P(FP)$ and $P(FN)$ are different than that of SPOT:

$$P(FP)_{SPPT} = \sum_{i=Q_\epsilon}^{Q} \binom{Q}{i} \xi^i (1-\xi)^{Q-i} \tag{A.5}$$

$$P(FN)_{SPPT} = \sum_{i=Q-Q_\epsilon+1}^{Q} \binom{Q}{i} \eta^i (1-\eta)^{Q-i} \tag{A.6}$$

It is assumed that the probability that a voxel is background (P_B) and the probability that a voxel is foreground (P_F) are also known. Given that, their formulation of the probability of misclassification in 3D therefore is

$$P(Err_{3D}) = P_B (P(FP)_{SPPT})^N + P_S \big(1 - (1 - P(FN)_{SPPT})^N\big) \tag{A.7}$$

Under the assumption that the probability of false positives and false negatives is equal in all of the views.

Then for the SPPT, Q_ϵ is chosen such that $P(Err_{3D})$ is minimized:

$$Q_\epsilon^* = \underset{Q_\epsilon \in [0,\ldots,Q]}{\arg\min}\ P(Err_{3D}) \tag{A.8}$$

The second portion of the work of [13] concerns the unbiased hull. Given an initial reconstruction, the unbiased hull seeks to reconcile those regions that are inconsistent. The mechanism for doing so is similar to the robost visual hull in that an optimal number of intersections is chosen (in the text it is denoted as T^*). If a voxel projects to T^* or more cameras where the voxel is classified as in the silhouette but not occupied

by the visual hull (using SPOT, SPPT, or any other method), that voxel is marked as occupied.

The unbiased hull is different from the robust visual hull in that T^* is chosen such that it minimizes an error function for each possible number of occlusions by the visual hull, whereas in the Robust Visual Hull occlusions are not considered.

In the UH method, there are four variables: S, the number of cameras that classify the image region v projects to as silhouette, O, the number of cameras where the visual hull reconstruction (using one's choice of projection test) projects to the same portion of the image plane as v, and C, the number of cameras that view v. I is what the authors term 'inconsistencies'. $I = S - O$.

The unbiased hull then introduces the probability of misclassification in 3D as a function of the variables introduced above:

$$P(Err_{3D}) = P_B \sum_{i=max(T^*,1)}^{C-O-1} \binom{C}{i} P(FP)_{SPPT}^{i}(1 - P(FP)_{SPPT})^{C-i} \\ P_S \sum_{i=max(C-O-T^*+1,1)}^{C-O-1} \binom{C}{i} P(FN)_{SPPT}^{i}(1 - P(FN)_{SPPT})^{C-i} \quad \text{(A.9)}$$

The Unbiased Hull algorithm computes the optimal value of T^* for each count of occlusion instances o. Consequently, we search for $T^*(o)$ for each $o \in [0, C-1]$ such that Eq. A.9 is minimized. The relationship between $T^*(o)$ and the labeling of voxels is as follows. For a voxel v_i, given that $O_i = o$ then the voxel is marked as occupied is $T^*[o] \leq I_i$. The authors specify the following Alg. 7 for determining values of $T^*(o)$.

Algorithm 7 UNBIASED HULL

1: **for all** occlusion counts $o = [0, C-1]$ **do**
2: $\quad T^*(o) = \arg\min_{T^*} P(Err_{3D}(O = o))$
3: **end for**

As in our implementation of the SPOT method, we also iterate over a set of η, ξ combinations, where $\eta \in [0.05, 0.2]$ and $\xi \in [0.05, 0.2]$, at increments of 0.05. We also

iterate over a set of possible $P_B \in [0.2, 0.8]$ at increments of 0.1. For datasets where the ground truth is known, as with the other methods we then choose the parameter combination that had the smallest number of misclassifications as compared to the ground truth.

A.4 Graph Cuts

In the Graph Cuts approach of Snow et al [30] the labeling of voxels as occupied or empty is formulated an an energy minimization function:

$$E(\mathbf{x}) = \sum_{\forall x_i \in \mathbf{x}} D_i(x_i) + \sum_{x_i, x_j \in \mathcal{N}} V_{i,j}(x_i, x_j) \tag{A.10}$$

where $\mathbf{x}$ is a vector representing Boolean voxel labelings ($x_i = 1$ means that a voxel is occupied, empty otherwise), and $\mathcal{N}$ representing neighborhood relations. In this paper, $\mathcal{N}$ is defined as 6-connected neighborhoods.

The neighborhood cost function is

$$V_{i,j}(x_i, x_j) = \lambda \delta(x_i \neq x_j) \quad \lambda \geq 0 \tag{A.11}$$

In the original publication, images were assumed to be unsegmented from their backgrounds. Δ_i represents the difference in between the background images and the image with the object of interest for voxel v_i. The particular projection method is not given, so whether Δ_i results from the absolute difference between the projection of the center of the voxel, or some sort of averaging over the entire region that a voxel projects, in not known.

With these preliminaries the data cost term is defined by the authors, when there are 16 cameras as

$$D_i(x_i = 0) = \frac{\sum_c min(\Delta_{c,i}^2, 400)}{16} \tag{A.12}$$

$$D_i(x_i = 1) = 300 \tag{A.13}$$

We use silhouettes as opposed to background/foreground object pairs and in our datasets the number of cameras viewing each voxel is not constant. We construct the data cost functions as follows in our implemention:

$$D_i(x_i = 0) = \frac{\alpha \sum \Delta_{c,i}}{C_i} \tag{A.14}$$

$$D_i(x_i = 1) = \beta \tag{A.15}$$

$\Delta_{c,i} \in [0, 1]$ is the percentage, for a camera c and a voxel i, or silhouette pixels versus total pixels using complete projection. C_i is the number of cameras that can view a voxel v_i. $\alpha, \beta \geq 0$ are constants, $\alpha \geq \beta$. We use the neighborhood cost function from the original publication with no change.

The choice of α, β, λ naturally varies across datasets. For the comparisons using the ground truth, we iterate over many different options for the three variables. $\alpha \in [300 - 600]$, $\beta \in [200 - 500]$, with β's range constrained by $\alpha \geq \beta$, and for both variables incremented by 100. $\lambda \in [0, 50]$ and in increments of 10. To calculate the reconstruction accuracy, we select the α, β, λ combination that yields the smallest misclassification error.

If we were to use the choice of α, β, λ used by the authors in their experiments, $\alpha = 400$, $\beta = 300$ and $\lambda = 30$.

A.5 SPOT-SIE

We observed that the SPOT approach could be improved with a cost function that does not depend on assumptions about the probability of false positives and false negatives. We tested this idea with our own SIE function. In this context, we search for the optimal Q_ϵ that results in the lowest value of the SIE function.

A.6 Unbiased Hull-SIE

For the Unbiased Hull-SIE algorithm, we start with the SPOT-SIE reconstruction and then compute the optimal number of inconsistencies $T^*(o)$ by choosing the T^* that results in the smallest value of the SIE function, for each possible occlusion case.

VITA

VITA

Amy Tabb is from the state of West Virginia, and was raised on a dairy and crop farm. She double-majored in Mathematics/Computer Science as well as Music at Sweet Briar College and was graduated in 2001. In 2003, she started work with the USDA-ARS-AFRS (Appalachian Fruit Research Station) in the engineering group as a programmer and the USDA-ARS-NCCCWA (National Center for Cool and Coldwater Aquaculture) laboratory, also as a programmer. Eventually, her work there led to being a USDA-sponsored student at Purdue University, under the SCEP (Supervisored Career Experience Program). At Purdue University, she entered the Agricultural and Biological Engineering department as a graduate student in 2004 and later switched to the Direct PhD program in the Electrical and Computer Engineering department. Since late 2012, she has been working in a research scientist position at the USDA-ARS-AFRS laboratory in Kearneysville, West Virginia. Her research interests are computer vision, robotics, and automation of orchard tasks.