# Volumetric 3D Reconstruction and Parametric Shape Modeling from RGB-D Sequences

Yoichi Nakaguro[1], Waqar S. Qureshi[2(✉)], Matthew N. Dailey[2], Mongkol Ekpanyapong[2], Pished Bunnun[1], and Kanokvate Tungpimolrut[1]

[1] National Electronics and Computer Technology Center, Klong Luang, Pathum Thani 12120, Thailand
[2] Asian Institute of Technology, Klong Luang, Pathum Thani 12120, Thailand
waqar.shahid@gmail.com, waqar.shahid.qureshi@ait.asia

**Abstract.** The recent availability of low-cost RGB-D sensors and the maturity of machine vision algorithms makes shape-based parametric modeling of 3D objects in natural environments more practical than ever before. In this paper, we investigate the use of RGB-D based modeling of natural objects using RGB-D sensors and a combination of volumetric 3D reconstruction and parametric shape modeling. We apply the general method to the specific case of detecting and modeling quadric objects, with the ellipsoid shape of a pineapple as a special case, in cluttered agricultural environments, towards applications in fruit health monitoring and crop yield prediction. Our method estimates the camera trajectory then performs volumetric reconstruction of the scene. Next, we detect fruit and segment out point clouds that belong to fruit regions. We use two novel methods for robust estimation of a parametric shape model from the dense point cloud: (i) MSAC-based robust fitting of an ellipsoid to the 3D-point cloud, and (ii) nonlinear least squares minimization of dense SIFT (scale invariant feature transform) descriptor distances between fruit pixels in corresponding frames. We compare our shape modeling methods with a baseline direct ellipsoid estimation method. We find that model-based point clouds show a clear advantage in parametric shape modeling and that our parametric shape modeling methods are more robust and better able to estimate the size, shape, and volume of pineapple fruit than is the baseline direct method.

**Keywords:** Volumetric reconstruction · Parametric shape modeling · Fruit health monitoring · RGB-D sensors

## 1 Introduction

Vision-based simultaneous localization and mapping (SLAM) has come to the point of maturity in coping with large-scale environments, gradually imposing fewer assumptions on sensors. The PTAM algorithm [9] performs motion estimation and mapping in parallel based on efficient bundle adjustment (BA) of sparse point features. Along the same lines, SVO-SLAM [6] applies sparse point-based direct alignment [5,8] to localize micro aerial vehicles (MAVs) flying in

© Springer International Publishing Switzerland 2015
V. Murino and E. Puppo (Eds.): ICIAP 2015, Part I, LNCS 9279, pp. 500–516, 2015.
DOI: 10.1007/978-3-319-23231-7_45

outdoor environments. In contemporary work, LSD-SLAM [4] applies semi-dense direct alignment to monocular camera sequences and has been successfully used to map large-scale outdoor environments containing challenging scale changes. While the work mentioned above builds accurate maps represented as sparse 3D point clouds, for near-range scene mapping, we can obtain more dense representations of the environment using modern portable and inexpensive RGB-D sensors such as the Microsoft Kinect. The seminal KinectFusion algorithm [12] demonstrated real-time dense mapping of indoor scenes with weighted signed distance functions assigned to fixed voxel grids. However, its critical reliance on GPU hardware and heavy memory demands due to a non-adaptive voxel grid representation pushed the development of more optimized solutions. The TUM Computer Vision Group released a series of RGB-D methods that cover direct motion estimation [7,8], benchmarking [18], and large-scale surface reconstruction using a memory-efficient octree data structure [16]. More recently, an even more carefully optimized version of the octree-based surface reconstruction algorithm was introduced as FastFusion, which requires only a single CPU [17].

The availability of low-cost RGB-D sensors and the maturity of machine vision algorithms for motion estimation and large-scale surface reconstruction have provided opportunities to automate monitoring and inspection tasks in applications as diverse as surveillance, medical diagnostics, remote sensing, industrial quality control, and precision agriculture. Automation in agricultural monitoring and inspection can help farmers to increase their efficiency and productivity as well as optimize crop yield. Crops bearing fruit, such as pineapples, mangoes, apples, oranges, and guavas have attracted researchers' attention due to the high demand for and value of the crops. Fruit crops require monitoring at regular intervals across different stages of growth to acquire information regarding pest infestation, fruit health, and predicted yield. One aspect of fruit health monitoring for such crops is to estimate the size and volume of individual fruits in the pre-harvest stage.

An autonomous fruit crop inspection system incorporating one or more mobile camera sensors and a host processor able to analyze the video sequences in detail could help farmers to monitor fruit health and growth trajectories over time and predict crop yield. The first step is to retrieve images containing fruits through an RGB-D sensor. Then we must segment the fruit regions from the background and track the fruit regions over time. The segmented fruit regions can then be used to generate volumetric 3D models to estimate the size and volume of the fruit.

In this paper, we perform a case study on the application of 3D dense volumetric reconstruction and shape modeling to pineapple fruit. Pineapple is a high-value crop that is grown by many farmers and on a large scale in Thailand. Chaivivatrakul and colleagues [1,11] describe a method for 3D reconstruction of pineapple fruits based on sparse keypoint classification, fruit region tracking, and structure from motion techniques. The method finds sparse Harris keypoints, calculates SURF descriptors for the keypoints, and uses a SVM classifier trained offline on hand-labeled data to classify the local descriptors. Morphological clos-

ing is used to segment the fruit using the classified features. Fruit regions are tracked from frame to frame. Frame-to-frame keypoint matches within putative fruit regions are filtered using the nearest neighbor ratio, symmetry test, and epipolar geometry constraints, then the surviving matches are used to obtain a 3D point cloud for the fruit region. An ellipsoid model is fitted to the point cloud to estimate the size and orientation of each fruit. The main limitation of the method is the use of sparse features with SURF descriptors to segment fruit regions. Filling in the gaps between sparse features using morphological operations is efficient but leads to imprecise delineation of the fruit region boundaries. To some extent, robust 3D reconstruction methods can clean up these imprecise boundaries, but the entire processing stream would be better served by an efficient but accurate classification of *every pixel in the image*, and then a dense 3D reconstruction using the classified fruit pixels. Qureshi et al. [13] present a texture-based dense fruit segmentation method for pineapples that uses super-pixel over-segmentation, dense SIFT (scale invariant feature transform) descriptors that characterize the local gradient field of an image around a keypoint, and a bag-of-visual-word histogram classifier within each super-pixel. This enables classification of every pixel in the image as a member of a fruit or non-fruit region.

In this study, we present a new method for volumetric reconstruction and shape modeling of quadric objects, with the ellipsoidal shape of a pineapple as a special case in cluttered outdoor environments typical of agricultural fields using an RGB-D sensor. We first estimate the camera trajectory, then we perform volumetric reconstruction of the scene. We segment out the point cloud that belongs to the fruit regions. We then use two novel methods to estimate a parametric shape model for the dense point cloud. We compare our shape modeling methods with direct fitting of an ellipsoid to the segmented point cloud. We find that our methods are better able to estimate the size, shape, and volume of pineapple fruit than is the baseline direct method.

## 2 Methodology

In order to obtain volumetric 3D models of objects in a scene captured by an RGB-D camera, we execute four consecutive processes: (i) camera motion estimation, (ii) 3D reconstruction given the estimated motion sequence, (iii) 2D segmentation of the objects of interest, and (iv) parametric shape modeling. We use the DVO-SLAM algorithm [7] for motion estimation and the FastFusion algorithm [17] for volumetric model reconstruction. With DVO-SLAM, we obtain an estimated camera trajectory based on a sequence of RGB-D image data. We then apply FastFusion with the DVO-SLAM trajectory as input. FastFusion fuses the observed color and geometry data to acquire a volumetric model from which a high-quality mesh can be generated. Although the general approach to parametric shape modeling is applicable to any kind of object whose shape can be expressed parametrically, in the case study developed in this paper, we focus on modeling pineapple fruit as ellipsoidal volumes. For dense segmentation of

pineapple fruit from RGB-D point clouds, we use pixel classification based on super-pixel over-segmentation, clustering of dense SIFT (Scale Invariant Feature Transform) features into visual words, and bag-of-visual-word super-pixel classification using SVMs (Support Vector Machines). The implementations of the DVO-SLAM and FastFusion algorithms are open source and freely available [20,21].

### 2.1 Motion Estimation

Camera tracking is the task of estimating, at any point in a sequence of images, a frame-to-frame transformation

$$\mathtt{G}_{4\times 4} = \begin{bmatrix} \mathtt{R} & \boldsymbol{t} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{1}$$

consisting of a rotation matrix $\mathtt{R} \in SO(3)$ and a translation vector $\boldsymbol{t} \in \mathbb{R}^3$. Since $\mathtt{G}$ expresses an element of the group $SE(3)$, $\mathtt{G}$ can be parameterized by a 6-vector $\boldsymbol{\xi} = [\boldsymbol{\omega}_{3\times 1}^{\intercal}, \boldsymbol{v}_{3\times 1}^{\intercal}]^{\intercal} \in \mathbb{R}^6$, which is an element of the Lie algebra $\mathfrak{se}(3)$, where $\boldsymbol{\omega}$ and $\boldsymbol{v}$ are the angular and linear displacements. We write $\mathtt{G}(\boldsymbol{\xi})$ to indicate the transformation matrix corresponding to $\boldsymbol{\xi}$. Using the exponential map $\exp(\cdot)$ from $\mathfrak{se}(3)$ to $SE(3)$, we can calculate $\mathtt{G}(\boldsymbol{\xi})$ as

$$\mathtt{G}(\boldsymbol{\xi}) = \exp\left(\begin{bmatrix} \hat{\omega} & \boldsymbol{v} \\ \mathbf{0} & 1 \end{bmatrix}\right), \tag{2}$$

where $\hat{\omega}$ is the skew-symmetric matrix form of $\boldsymbol{\omega}$.

Let $\mathtt{T}$ and $\mathtt{T}'$ be the $4 \times 4$ transformation matrices relating points in the world coordinate system to camera frames $F$ and $F'$. We define the camera projection $\pi$ of a point $\boldsymbol{p} = [p_1, p_2, p_3, 1]^{\intercal}$ in the camera frame as

$$\pi(\boldsymbol{p}) = \left[\frac{f_u p_1}{p_3} - c_u, \frac{f_v p_2}{p_3} - c_v\right]^{\intercal}, \tag{3}$$

where $f_u$ and $f_v$ are the focal lengths and $[c_u, c_v]^{\intercal}$ is the principal point of the camera. Using $\pi$, the projected image points $\boldsymbol{x} = \pi(\mathtt{T}\boldsymbol{p}_w)$ and $\boldsymbol{x}' = \pi(\mathtt{T}'\boldsymbol{p}_w)$ of a world point $\boldsymbol{p}_w$, along with the relationship $\mathtt{T}' = \mathtt{GT}$, we can obtain a warping function $\tau$ explicitly written as a function of $\boldsymbol{\xi}$:

$$\begin{aligned} \tau(\boldsymbol{x}, \boldsymbol{\xi}) &= \boldsymbol{x}' \\ &= \pi(\mathtt{T}'\boldsymbol{p}_w) \\ &= \pi(\mathtt{G}(\boldsymbol{\xi})\mathtt{T}\boldsymbol{p}_w) \\ &= \pi(\mathtt{G}(\boldsymbol{\xi})\pi^{-1}(\boldsymbol{x}, Z(\boldsymbol{x}))). \end{aligned} \tag{4}$$

The inverse of $\pi$ is calculated from a pixel $\boldsymbol{x} = [u, v]^{\intercal}$ and $Z(\boldsymbol{x})$ (the observed depth of $\boldsymbol{x}$) as

$$\pi^{-1}(\boldsymbol{x}, Z(\boldsymbol{x})) = \left[\frac{u + c_u}{f_u} Z(\boldsymbol{x}), \frac{v + c_v}{f_v} Z(\boldsymbol{x}), Z(\boldsymbol{x}), 1\right]^{\intercal}. \tag{5}$$

We use DVO-SLAM to find the frame-to-frame transformations $\mathtt{G}(\boldsymbol{\xi})$ that optimally align the observed intensity and depth images. More specifically, DVO-SLAM attempts to minimize the combined error $\boldsymbol{r} = [r_I, r_Z]^\intercal$ consisting of the photometric residue $r_I$ and the depth residue $r_Z$, which are defined as

$$r_I = I'(\tau(\boldsymbol{x}, \boldsymbol{\xi})) - I(\boldsymbol{x}), \tag{6}$$

$$r_Z = Z'(\tau(\boldsymbol{x}, \boldsymbol{\xi})) - Z(\boldsymbol{x}), \tag{7}$$

where $I$, $I'$, $Z$, and $Z'$ are the intensity images and depth images captured from camera frames $F$ and $F'$, respectively. To obtain an optimal robust estimate of $\mathtt{G}(\boldsymbol{\xi})$, we seek the motion vector $\boldsymbol{\xi}$ that minimizes the sum of the weighted squares of $\boldsymbol{r}$ over all valid pixel indices $i$ in $I$:

$$\boldsymbol{\xi}^* = \underset{\boldsymbol{\xi}}{\operatorname{argmin}} \sum_i w_i \boldsymbol{r_i}^\intercal \Sigma_{\boldsymbol{r}}^{-1} \boldsymbol{r_i}, \tag{8}$$

where $w_i = (\nu + 1)/(\nu + \boldsymbol{r_i}^\intercal \Sigma_{\boldsymbol{r}}^{-1} \boldsymbol{r_i})$ is a pixel-wise weight and $\Sigma_{\boldsymbol{r}}$ is a scale matrix. DVO-SLAM assumes that the bivariate random variable $\boldsymbol{r}$ follows a t-distribution $p_t(\boldsymbol{\mu_r}, \Sigma_{\boldsymbol{r}}, \nu)$ with mean $\boldsymbol{\mu_r} = \mathbf{0}$ and $\nu = 5$ degrees of freedom. Nonlinear least squares estimation of $\boldsymbol{\xi}$ is performed iteratively using the Gauss-Newton algorithm with the following linearized normal equations:

$$\sum_i w_i \mathtt{J}_i^\intercal \Sigma_{\boldsymbol{r}}^{-1} \mathtt{J}_i \Delta\boldsymbol{\xi} = -\sum_i w_i \mathtt{J}_i^\intercal \Sigma_{\boldsymbol{r}}^{-1} \boldsymbol{r}_i, \tag{9}$$

where $\Delta\boldsymbol{\xi}$ is an unknown increment vector and $\mathtt{J}_i = \partial \boldsymbol{r}_i / \partial \boldsymbol{\xi}$ is the $2 \times 6$ Jacobian matrix of the residual vector $\boldsymbol{r}_i$ evaluated at $\boldsymbol{\xi} = \mathbf{0}$. On each Gauss-Newton iteration, $w_i$ and $\Sigma_{\boldsymbol{r}}$ are re-estimated using an expectation-maximization algorithm.

To reduce the accumulated drift across the estimated frame-to-frame transformations, DVO-SLAM uses a key-frame based pose SLAM method. A new key-frame is selected as the uncertainty of motion estimation relative to the last key-frame grows. To obtain an optimized camera trajectory, we construct and optimize a pose graph of the key-frames where the edges between adjacent key-frames are weighted by the uncertainty of the corresponding motion estimation. For further details, we refer the reader to [7].

### 2.2 Model Reconstruction

As previously mentioned, we use the FastFusion algorithm for volumetric 3D reconstruction. FastFusion is based on three main concepts: implicit surface representation, an efficient octree data structure, and mesh generation.

**Surface Representation via Signed Distance Function.** After motion estimation, we have the estimated trajectory of the camera $\mathtt{T}_1, \cdots \mathtt{T}_t$. In FastFusion [17], following Curless and Levoy [2], a 3D surface is implicitly expressed as a collection of signed distances assigned to the centers of voxels in the space.

Let $\boldsymbol{p}$ be the center of a voxel in the world coordinate frame. For a given camera frame with estimated world-to-camera transformation $\mathtt{T}_t$ at time $t$, the center point $\boldsymbol{p}_c$ in the camera coordinate frame is

$$\boldsymbol{p}_c = \mathtt{T}_t \boldsymbol{p}. \tag{10}$$

On the other hand, using the image projection function $\pi(\boldsymbol{p}_c)$ and the depth map $Z_t$, an observed point $\boldsymbol{p}_{obs}$ along the ray from the camera center through $\boldsymbol{p}_c$ can be calculated as

$$\boldsymbol{p}_{obs} = \pi^{-1}(\pi(\boldsymbol{p}_c), Z_t(\pi(\boldsymbol{p}_c))). \tag{11}$$

$Z_t(\pi(\boldsymbol{p}_c))$ can be interpolated from neighboring pixel depth measurements. The signed distance function $d_t$ is defined as

$$d_t(\boldsymbol{p_c}, Z_t) = \max(\min(|\boldsymbol{p}_c - \boldsymbol{p}_{obs}|, \Phi), -\Phi), \tag{12}$$

where $|\cdot|$ is the Euclidean norm with a sign indicating which side of the surface containing $\boldsymbol{p}_{obs}$ the voxel center $\boldsymbol{p}_c$ lies, and $\Phi(>0)$ is a cut-off threshold set to twice the voxel scale. Along with $d_t$, FastFusion also defines a weight function

$$w_t = \begin{cases} 1 & \text{if } d_t < \delta \\ \frac{\Phi - d_t}{\Phi - \delta} & \text{if } \delta < d_t < \Phi \\ 0 & \text{if } d_t > \Phi, \end{cases} \tag{13}$$

where $\delta$ is set to one tenth of the voxel resolution. This weight gives linearly decreasing confidence when the voxel center is behind the surface (the sign of $|\boldsymbol{p}_c - \boldsymbol{p}_{obs}|$ is positive).

When a new observation for a previously observed voxel is obtained at time $t$, again following Curless and Levoy, the previously assigned weight $W_{t-1}$ and signed distance $D_{t-1}$ are updated according to the following rules:

$$W_t = w_t + W_{t-1}, \tag{14}$$

$$D_t = \frac{D_{t-1} W_{t-1} + d_t w_t}{w_t + W_{t-1}}. \tag{15}$$

Similarly, the previously stored RGB color vector $\boldsymbol{C}_{t-1} = [C_{\mathrm{R},t-1}, C_{\mathrm{G},t-1}, C_{\mathrm{B},t-1}]^\intercal$ is updated as

$$\boldsymbol{C}_t = \frac{\boldsymbol{C}_{t-1} W_{t-1} + \boldsymbol{I}_t^C(\pi(\boldsymbol{p}_c)) w_t}{w_t + W_{t-1}}, \tag{16}$$

where $\boldsymbol{I}_t^C = [I_{\mathrm{R},t}, I_{\mathrm{G},t}, I_{\mathrm{B},t}]^\intercal$ is the observed RGB pixel at time $t$.

**Multi-resolution Octree Representation of Surfaces.** To store observed surfaces in a memory efficient manner, FastFusion uses a novel octree-based surface representation with the previously explained signed distance function.

Associated with every branch and leaf in the octree is a set of $8 \times 8 \times 8$ voxels that equally partition its volume. Each tree level represents a 3D model at a particular resolution, and we only allocate voxels within the vicinity of observed 3D points, making the data structure sparse and memory efficient.

When a new observation is obtained, we first determine the resolution of the observed surface points based on their depth values, then we conduct a depth-first-search to find a bounding volume in the level corresponding to that resolution. Finally, we allocate a new set of voxels if the corresponding volume is empty, or otherwise, update the signed distances and color information based on Equations 14, 15, and 16.

**Mesh Generation with Marching Cubes.** Now, given the signed distance associated with each voxel in the the multi-resolution grid, we apply the well-known marching cubes algorithm to extract an explicit surface representation from the grid. Since voxels near to and in front of an observed surface will have negative signed distances, and voxels near to and behind an observed surface will have positive signed distances, an excellent estimate of the surface would be a 3D triangle mesh corresponding to the 0 level set of the grid. The well-known marching cubes algorithm is ideal for obtaining such a mesh.

The mesh extraction is thus straightforward when the voxels being considered are all at the same level of resolution in the octree. With multi-resolution voxels, however, we have to solve for border cases where voxels at a higher and lower resolution are adjacent to each other. FastFusion proposes a recursive algorithm capable of solving this problem as follows.

Suppose a branch $B^s$ at scale $s$ is divided into eight subbranches $B_i^{s+1}, i \in \{1, \cdots, 8\}$ at scale $s+1$. We can categorize voxels $V_{i,j}^{s+1}, j \in \{1, \cdots, 8^3\}$ belonging to each of the eight subbranches into four types: interior, face, edge, and corner. An interior voxel is a voxel, when considered as the origin of a group of eight neighboring voxels considered for mesh generation, whose seven higher voxels are all within the same subbranch. A face voxel is a voxel having neighbor voxels belonging to one other subbranch. An edge voxel is a voxel having neighbors belonging to three other subbranches. Finally, a corner voxel is a voxel having neighbors belonging to seven other subbranches. If any of the subbranches affecting meshing of $V_{i,j}^{s+1}$ are subdivided into a higher scale $s+2$, the voxels in the higher resolution neighboring subbranches could themselves be either interior, face, edge, or corner voxels depending on the situation. Therefore, we can construct a recursive algorithm to perform meshing of the entire tree. When a marching cube contains voxels with lower resolution than other voxels, we perform interpolation to break the lower resolution voxels into corresponding higher resolution voxels.

### 2.3 Fruit Segmentation

For dense segmentation of fruit regions in images, we need to classify each pixel into fruit and non-fruit regions. Color-based dense classification fails when the

objects of interest have coloration similar to that of the background, such as pineapple fruits in the field. Chaivivatrakul et al. [1] note the limitations of color and shape cues for recognition of green fruit on trees and plants and propose texture-based classification instead. Qureshi et al. [13] describe a texture-based classification method that selects fruit regions using a dense pixel segmentation method. The method performs over-segmentation into super-pixels, extracts Dense-SIFT descriptors for the pixels in a super-pixel, maps SIFT descriptors to clusters, constructs a bag-of-visual-words histogram for the clusters appearing in the super-pixel, and then classifies the histogram for a super-pixel as fruit or non-fruit using a support vector machine (SVM). Dense-SIFT is a type of gradient orientation histogram descriptor that captures the distribution of local gradients in a pixel's neighborhood. Clustering local gradient descriptor improves sensitivity to noise, then the bag-of-visual-words histogram characterizes the differing distribution of gradient descriptors within a superpixel. Finally, since local histograms based on a small number of pixels in a region with uniform coloration can be quite sparse, augmenting each histogram by including the histograms of neighboring super-pixels. Put together, these techniques enable us to create a unique signature of a region for classification. The classifier requires off-line training prior to runtime utilization of the classifier model. Training requires a set of training images along with ground truth data. Ground truth labeling (assigning a label of "fruit" or "non-fruit" to each pixel) is performed manually.

To segment an input point cloud into fruit and non-fruit regions, we first obtain a 2D image mask indicating likely fruit regions, then we segment the point cloud by filtering out 3D points in correspondence with non-fruit regions in the 2D mask. To obtain the 2D mask, we use the method described by Qureshi et al. [13].

For the experiments reported upon in this paper, we used the same parameters (quick-shift variables, dense-SIFT bin size, size of the dictionary for the $k$-means clustering of SIFT descriptors, scale at which to compute SIFT features, number of neighbors used to construct super-pixel histograms, and conditional random field (CRF) post-processing) as reported by Qureshi et al. [13].

At runtime, segmenting a new image using the trained model requires super-pixel over-segmentation, dense SIFT descriptor computation, visual word histogram calculation, SVM classification, and CRF post-processing. The classifier outputs a confidence for each super-pixel; a super-pixel is classified as a fruit region if the confidence is higher than a threshold. The threshold, determined by the SVM training algorithm, is that which best separates the fruit super-pixels from the non-fruit super-pixels in its training set.

As already mentioned, once a 2D mask for likely fruit regions is obtained, we use the mask to filter the 3D point cloud (raw or generated through volumetric reconstruction) according to the label assigned to each 3D point's 2D correspondence.

### 2.4 Parametric Fruit Modeling

A pineapple fruit shape can be best described by an ellipsoid [1]. An ellipsoid is a special case of the general quadric, which is the set of homogeneous 3D points $\boldsymbol{p}$ such that

$$\boldsymbol{p}^T \mathtt{Q} \boldsymbol{p} = 0, \tag{17}$$

with $\mathtt{Q}$ a $4 \times 4$ symmetric homogeneous matrix.

When the quadric is a centered, axis-aligned ellipsoid, the diagonal terms of $\mathtt{Q}$ must have the same sign to be characterized as ellipsoids. Enforcing this constraint constrains the parameters of the general quadric, reducing sensitivity to errors. One method to estimate an ellipsoid from a 3D point cloud is Li and Griffiths' [10] direct least squares method. However, when we use an RGB-D sensor in the field, it is not in general possible to move slowly around every fruit, so we typically obtain only a partial view. Also, the raw or FastFusion-based point clouds we obtain as previously described contain noise from the RGB-D sensor as well as small non-fruit regions arising due to false positives in the 2D image segmentation method. We find that the direct least squares method for estimating ellipsoids from sparse point clouds does not work well for dense point clouds containing false fruit regions and other noise. Therefore, robust estimation of an ellipsoid requires eliminating outliers (false positive fruit points) that would otherwise affect the ellipsoid model estimate. Moonrinta et al. [11] present a robust ellipsoid estimator using RANSAC and the direct least squares method with sparse samples of 3D points from the fruit surface. Here, we present an extension of their parametric shape modeling method to dense point clouds that is more robust to noise and false fruit regions. Then we present a new nonlinear optimization method for finding the parameters of an ellipsoid that minimizes the dense SIFT descriptor distance between pixels in correspondence according to the hypothesized ellipsoid.

**Robust Parametric Shape Fitting.** Prior to shape fitting, to eliminate noisy and non-fruit points far from the target point cloud, we perform clustering of the points in the 3D point cloud using $k$-means. In the experiments reported upon in this paper, we manually set $k$ to be one more than the number of actual fruits observed in the point cloud. In future work, we plan to automate the selection of $k$ (using, for example, the Bayesian Information Criterion (BIC)). After performing $k$-means, we remove the points belonging to the smallest cluster on the assumption that it contains only noise.

After $k$-means and noise removal, we perform robust estimation of the ellipsoid model with Li and Griffiths' direct least squares method as the basic estimator. Following the general approach of random sample consensus (RANSAC), we alternate between estimation of a model from a randomly selected minimum sample from the data set and checking the size of the "consensus" or inlier set for the estimated model. The sample-and-test process is terminated after a fixed number of iterations.

When testing an estimated model in this procedure, we use Torr et al.'s [19] ranking of the consensus set. The original RANSAC method simply maximizes the count of the number of points in the consensus set $CS$:

$$r(CS) = \sum_i |CS|. \tag{18}$$

Following Torr et al., this can also be written as minimization of a cost function

$$r(CS) = \sum_i \rho(e_i^2), \tag{19}$$

where $e_i^2$ is the orthogonal distance between point $i$ and the estimated model, and

$$\rho_{RANSAC}(e_i^2) = \begin{cases} 0 & e_i^2 < T \\ 1 & \text{otherwise,} \end{cases} \tag{20}$$

where $T$ in our case is a threshold on the allowable distance of each 3D point to the ellipsoid model's surface.

Torr et al. propose, rather than a hard threshold and inlier count, an alternative cost function inspired by M-estimation:

$$\rho_{MSAC}(e_i^2) = \begin{cases} e_i^2 & e_i^2 < T \\ T & \text{otherwise.} \end{cases} \tag{21}$$

In MSAC, outliers are given a fixed penalty as in RANSAC, but inliers are graded by how fit they are for the model. This sample ranking method results in better estimates than the original RANSAC ranking criterion. We use Zuliani's implementation of MSAC [24]. The complete estimation procedure can be summarized as follows:

1. Randomly select 10 points from the point cloud.
2. Estimate the ellipsoid `Q` best fitting the selected points [10].
3. Translate and rotate the 3D point set into the ellipsoid's coordinate system.
4. Find the orthogonal distance of each point to the surface of the ellipsoid [3].
5. Find the inlier consensus set, i.e., the set of points lying within the distance threshold from the ellipsoid surface.
6. Find the MSAC cost of the sample [19]. If it is the lowest-cost sample seen so far, save the model.
7. Repeat from step 1 until a maximum number of iterations is reached.

**Nonlinear Optimization.** Here we present a new iterative nonlinear optimization method for estimating an ellipsoid parametric shape model from RGB-D point cloud data. We aim to find the parameters of the ellipsoid that minimizes the dense SIFT descriptor distance between pixels in correspondence according to the hypothesized ellipsoid. We use two key-frames to find the optimized ellipsoid. We first eliminate noisy and non-fruit 3D points far from the target point

cloud by $k$-means as explained in Section 2.4. We use the Levenberg-Marquardt (LM) algorithm to find the best $\mathtt{Q}$. We initialize LM with the ellipsoid estimated from the 3D point cloud using Li and Griffiths' [10] direct least squares method. Mathematically we can state the problem as follows:

> Given a set of 2D points $\boldsymbol{x_i} \in \mathbb{P}^2$ in image $I$ with $i \in 1 \cdots n$ and $n \geq 10$ and a second image $I'$, find a $4 \times 4$ symmetric homogeneous matrix $\mathtt{Q}$ that minimizes the cost function
>
> $$\chi = \sum_{i=1}^{n} d(D_{\boldsymbol{x_i}}, D_{\boldsymbol{x_i'}})^2, \tag{22}$$
>
> subject to
>
> $$\boldsymbol{p}_i^{\mathsf{T}} \mathtt{Q} \boldsymbol{p}_i = 0.$$
>
> In the equation, $d(\cdot)$ is Euclidean distance, $D_{\boldsymbol{x}_i}$ and $D_{\boldsymbol{x}_i'}$ are SIFT descriptors of $\boldsymbol{x_i}$ and $\boldsymbol{x_i'}$ in image $I$ and $I'$, respectively, $\boldsymbol{p}_i$ is the back projection of $\boldsymbol{x}_i$ onto $\mathtt{Q}$, and $\boldsymbol{x_i'}$ is the reprojection of $\boldsymbol{p}_i$ into image $I'$.

For lack of space, we omit the derivation of how to find $\boldsymbol{x}'$ from $\boldsymbol{p}$ and $\mathtt{Q}$.

To compute SIFT descriptors for the reprojected 2D points $\boldsymbol{x}'$ with subpixel accuracy, we first compute a dense SIFT descriptor for each pixel in image $I'$, then we use spline interpolation to interpolate the SIFT descriptors of the reprojected 2D points.

Whenever the ray back-projected from point $\boldsymbol{x}_i$ in image $I$ does not intersect with the hypothesized quadric, there is no corresponding 3D point $\boldsymbol{p}_i$, in which case we assign a cost $\varrho$ in place of $d(D_{\boldsymbol{x}_i}, D_{\boldsymbol{x}_i'})$ in Equation 22.

Pineapple fruit can be modeled by ellipsoids (nearly spheroids) that have a major-axis to minor-axis ratio in the range of 1.0 to 2.0. To encourage LM to traverse only the family of ellipsoids that have a major-axis to minor-axis ratio of $r$ such that $1 < r < 2$, we add another penalty $\sigma$ in the cost function given in Equation 22 penalizing extreme ratios. As a final modification, since the point cloud from the RGB-D sensor is dense, and since the depths of the points in the point cloud give reasonable estimates of the location of the ellipsoid in the z-direction, we further constrain LM by adding a penalty $\zeta$ discouraging changes in the average depth of the hypothesized quadric by more than $\pm 1.0\%$.

The new cost function, after adding penalties $\varrho$, $\sigma$, and $\zeta$, becomes

$$\chi = \sum_{i \in 1 \cdots n | \boldsymbol{x}_i' \text{exists}} d(D_{\boldsymbol{x_i}}, D_{\boldsymbol{x_i'}})^2 + \sum_{i \in 1 \cdots n | \boldsymbol{x}_i' \neg \text{exists}} \varrho + \sigma + \zeta. \tag{23}$$

A summary of the steps of the LM optimization of $\mathtt{Q}$ is as follows:

1. Find the 3D points lying on $\mathtt{Q}$ back-projected from the fruit points in image $I$.
2. Find the re-projections of the back-projected 3D points into image $I'$.
3. Compute SIFT descriptors of fruit pixels in image $I$ and $I'$.
4. Compute SIFT descriptors of re-projected pixels in $I'$ using spline interpolation.

5. Compute the L2 distances between SIFT descriptors of corresponding fruit pixels in image $I$ and $I'$.
6. For each back-projected 2D point that does not intersect $\mathtt{Q}$, add a penalty $\varrho$ instead of the SIFT descriptor distance.
7. Find the ratio of major-axis to minor axis of the hypothesized ellipsoid.
8. If the ratio $r$ is in the range $1 < r < 2$, then $\sigma = 0$, else $\sigma = 5$.
9. If the mean depth of the back-projected 2D points is within 1% of the point cloud's average depth, then then $\zeta = 0$ else $\zeta = 5$.

## 3 Experimental Evaluation

We performed a real-world experiment involving a case study on 3D reconstruction of pineapple fruits as an empirical evaluation of the feasibility of our approach. We captured video data from an outdoor scene containing two fruits then applied the modeling method to the resulting RGB-D image sequence. In this section we detail the experimental design then present the experiment's results.

### 3.1 Experimental Methods

To simulate conditions in a pineapple field, we placed two pineapple fruits (Fruit A and Fruit B) on top of other plants with long leaves that resemble real pineapple leaves. The horizontal distance between the two pineapples was approximately 2m. The entire volume needed for this small-scale mock pineapple field was approximately 4m × 2m × 0.5m. To record an RGB-D sequence of the mock field, we used the Apple Primesense Carmine 1.09 short-range sensor, which has an operational range of 0.35m–1.4m. Since this device requires only a standard USB connection to operate, we can use it with a laptop computer in both indoor and outdoor environments. In the case of an outdoor environment, however, the scene illumination should not substantially exceed typical indoor illumination, or the depth sensor fails. Therefore, in this experiment, we enforced this weak lighting constraint by recording in the late afternoon. We recorded an RGB-D sequence of 400 frames while manually moving the sensor over the pineapples with the camera facing toward the fruits. The frame rate for both RGB and depth image acquisition was set to 30 fps. After acquiring the RGB-D sequence, we applied DVO-SLAM to obtain an estimated trajectory for the sequence of camera frames. With the trajectory and RGB-D data as input, we then applied FastFusion to incrementally build a volumetric model of the scene. After completing the fusion of all 400 RGB-D frames, we exported a textured triangle mesh of the final model into our OpenGL-based custom software. Fig. 1 shows the entire view of the final model along with the estimated camera trajectory.

Based on the FastFusion mesh, we used OpenGL to render a new sequence of 100 RGB and 100 depth images corresponding to every four camera frames in the original trajectory. At this point, we had 100 synthetic RGB and depth images rendered in OpenGL relative to a sequence of estimated camera frames. Fig. 2 shows a pair of synthetic RGB and depth images of Fruit A. For each

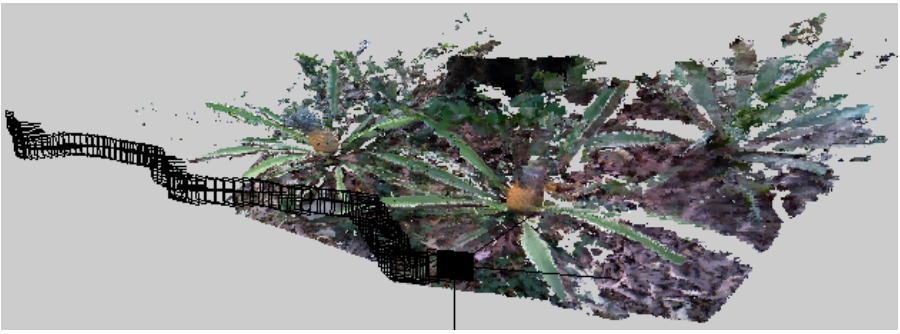


**Fig. 1.** Reconstruction of a mock pineapple field with the estimated camera trajectory.

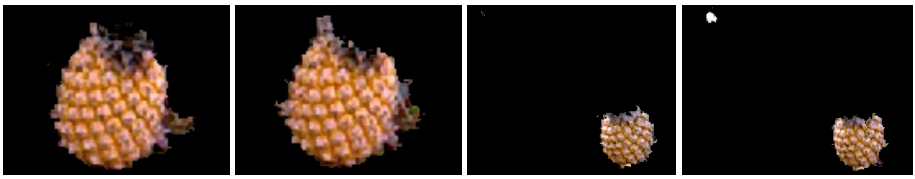
resulting synthetic RGB image, we applied the 2D fruit region mask obtained from dense segmentation of the corresponding original image. Finally, we generated a 3D point cloud from the masked synthetic RGB-D image. For purposes of experimental comparison, in addition to these 100 synthetic point clouds, we also generated 100 corresponding 3D point clouds based on the raw RGB-D sensor data using the same fruit mask.

(a) (b) (c)

**Fig. 2.** Sample data for Fruit A. (a) Original RGB image. (b) Synthetic RGB image rendered from volumetric model at the same camera position as (a). (c) Depth buffer corresponding to (b).

For parametric shape modeling, we compare three methods of estimating an ellipsoid from the segmented 3D-point cloud data, where the point cloud is either the raw RGB-D sensor based point cloud or the point cloud synthesized from the volumetric model. The first estimation method is direct least squares [10], the second is MSAC, and the third is SIFT differences, which were discussed in section 2.4. For the direct estimation we perform $k$-mean clustering to remove potential false fruit regions similar to the steps. For method two and three we follow the steps mentioned in and respectively.

We tuned the penalty parameters experimentally. For each of the two fruits in the experimental sequence, we extracted point clouds based on two manually selected key-frames. Fig. 3 shows the dense segmentation of the four key-frames



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

**Fig. 3.** Segmented Fruit A and Fruit B. (a) Segmented Fruit A in frame 1. (b) Segmented Fruit A in frame 2. (c) Segmented Fruit B in frame 3. (d) Segmented Fruit B in frame 4.

used to segment the point clouds for fruits A and B. The segmentations of the fruit A images contain several non-fruit false positive regions connected to the true positive region, while the segmentation of fruit B is more accurate, with one false positive region not connected to the true positive region in Fig. 3(d).

### 3.2 Results

Results for the application of three parametric shape modeling methods (Direct, MSAC, and SIFT differences) to two types of point clouds (raw and synthetic model-based) are given in Table 1. We obtained ground truth geometries of Fruit A and Fruit B by manual measurements. Parametric shape modeling was performed on segmented point clouds corresponding to synthetic models and raw sensor observations, respectively. Fig. 4 shows shape models of fruit B projected onto an original RGB image.

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

**Fig. 4.** Shape models of fruit B. (a) Original RGB image. (b) Direct method (failure). (c) MSAC method. (d) SIFT differences method.

There is a clear advantage to model-based point clouds. Except for the divergence of the direct method applied to model-based point clouds of fruit B, the error rates for all shape modeling methods are smaller due to the higher accuracy of model-based point clouds than raw point clouds from the sensor.

The baseline direct method shows instability when it is applied to model Fruit B due to false fruit regions. We used non-connected fruit region for direct method, MSAC and LM. The $k$-mean was not able to remove the non connected path. We assume the same steps for each. Parametric shape modeling using other

two methods however successfully converged to the approximate shape and size of the fruits.

The performance comparison between MSAC and SIFT differences is less clear. Although SIFT differences uses the richer information contained in the SIFT descriptors as constraints, it does not show any clear performance gain over the MSAC method.

**Table 1.** Point cloud types vs. shape modeling methods.

| Point Cloud | Parameters | Ground Truth | | Observations | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Direct | | MSAC | | SIFT differences | |
| | | Fruit A | Fruit B | Fruit A | Fruit B | Fruit A | Fruit B | Fruit A | Fruit B |
| Model | Major | 8.9 | 7.8 | 10.5 | 30.7 | 9.2 | 7.7 | 9.2 | 7.9 |
| | Minor 1 | 5.1 | 5.2 | 4.8 | 24.8 | 5.9 | 5.2 | 5.9 | 5.3 |
| | Minor 2 | 5.1 | 5.2 | 5.1 | 20.9 | 5.6 | 5.8 | 5.6 | 5.9 |
| | Volume | 969.2 | 883.0 | 1079.0 | 66431.7 | 1257.7 | 974.9 | 1268.8 | 1047.6 |
| | Major/Minor | 1.75 | 1.50 | 2.12 | 1.35 | 1.61 | 1.41 | 1.61 | 1.41 |
| | Major error | – | – | 18.0% | 294.0% | **3.3%** | **-0.7%** | 3.6% | 1.7% |
| | Minor 1 error | – | – | **-5.5%** | 376.3% | 14.8% | **0.3%** | 15.2% | 2.8% |
| | Minor 2 error | – | – | **-0.2%** | 301.0% | 9.4% | **10.9%** | 9.7% | 13.5% |
| | Volume error | – | – | **11.3%** | 7423.3% | 29.8% | **10.4%** | 30.9% | 18.6% |
| | Major/Minor error | – | – | 21.5% | -10.2% | **-7.8%** | -6.0% | -7.9% | **-5.9%** |
| | | | | | | | | | |
| Raw | Major | 8.9 | 7.8 | 11.1 | 23.1 | 9.8 | 8.5 | 9.8 | 8.5 |
| | Minor 1 | 5.1 | 5.2 | 6.6 | 14.9 | 6.1 | 5.6 | 6.1 | 5.6 |
| | Minor 2 | 5.1 | 5.2 | 7.2 | 13.6 | 7.3 | 4.8 | 7.3 | 4.9 |
| | Volume | 969.2 | 883.0 | 2183.1 | 19510.5 | 1815.4 | 965.8 | 1814.1 | 971.3 |
| | Major/Minor | 1.75 | 1.50 | 1.61 | 1.62 | 1.47 | 1.62 | 1.47 | 1.62 |
| | Major error | – | – | 24.6% | 195.6% | **10.1%** | **8.7%** | **10.1%** | 8.9% |
| | Minor 1 error | – | – | 28.8% | 186.3% | **19.3%** | **8.0%** | **19.3%** | 8.2% |
| | Minor 2 error | – | – | **40.4%** | 161.1% | 42.6% | -6.8% | 42.5% | **-6.6%** |
| | Volume error | – | – | 125.3% | 2109.5% | 87.3% | **9.4%** | **87.2%** | 10.0% |
| | Major/Minor error | – | – | **-7.5%** | **8.0%** | -16.0% | 8.1% | -15.9% | 8.1% |

## 4 Conclusion

In this paper, we have presented a new method for volumetric reconstruction and shape modeling of quadric objects using an RGB-D sensor. We first estimate the camera's trajectory, then we perform volumetric reconstruction of the scene. We segment out point clouds belonging to object regions. We then use two novel methods for robust estimation of a parametric shape model for the extracted dense point cloud. We compare our shape modeling methods with direct fitting of an ellipsoid to the segmented point cloud.

The main limitations of the experimental setup are (i) the small scale of the mock pineapple field (4 m × 2 m × 0.5 m, with only two fruit), (ii) the limited operational range of the sensor (0.35 m–1.4 m), (iii) the limited resolution of the RGBD camera (640 × 480), (iv) the requirement for diffuse lighting, and (v) the requirement for sufficient fruit surface visibility given the camera angles. In future work, we will attempt to mitigate these limitations.

We find that model-based point clouds show a clear advantage over raw depth sensor point clouds for parametric shape modeling. Also, our methods are more robust and better able to estimate the size, shape, and volume of pineapple fruit than is the baseline direct method. Although we hypothesized

that LM optimization of dense SIFT descriptor distances would perform better than MSAC, we did not observe a clear difference in the performance of the two algorithms. One reason for this may be the noise due to false positive fruit pixels around the segmented fruit boundaries (see Fig. 3).

In the future, we plan to investigate possible improvements, for example constraining the parametric model to spheroids rather than general ellipsoids. We also plan to improve the SIFT difference method's sensitivity at object boundaries. We also plan to investigate the possibility of obtaining similarly high accuracy without the RGB-D sensor and volumetric modeling, instead estimating the camera trajectory using structure from motion (SfM) based techniques similar to those of Wu et al. [23]. After obtaining monocular camera positions, we can perform SIFT difference optimization to estimate the parametric shape model by initializing it with a quadric estimated from the sparse set of 3D points estimated through SfM. We also plan to test our methods on a large scale real fruit crop in Thailand. Efficient fitting of quadric shapes to unstructured point clouds or triangle meshes is an important component of many reverse engineering systems [15,22]. Up till now, the use of ellipsoid shapes has been limited, but ellipsoids have proven useful for body-part modeling in the past [14]. Our methods (MSAC and SIFT differences) could both be used in a general framework for quadric surface modeling to point-cloud data. Although most objects are not purely quadratic, an extension to piecewise-quadratic surface estimation would enable efficient and accurate modeling of a large class of real-world objects more compactly than the current polygon mesh based approaches.

## References

1. Chaivivatrakul, S., Moonrinta, J., Dailey, M.N.: Towards automated crop yield estimation: detection and 3D reconstruction of pineapples in video sequences. In: International Conference on Computer Vision Theory and Applications (2010)
2. Curless, B., Levoy, M.: A volumetric method for building complex models from range images. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996, pp. 303–312. ACM, New York (1996)
3. David Eberly: Distance from a Point to an Ellipsoid (2008). http://www.geometrictools.com/
4. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: large-scale direct monocular SLAM. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014, Part II. LNCS, vol. 8690, pp. 834–849. Springer, Heidelberg (2014)
5. Engel, J., Sturm, J., Cremers, D.: Semi-dense visual odometry for a monocular camera. In: IEEE International Conference on Computer Vision (ICCV). Sydney, Australia, December 2013
6. Forster, C., Pizzoli, M., Scaramuzza, D.: Svo: fast semi-direct monocular visual odometry. In: Int. Conf. on Robotics and Automation, pp. 15–22 (2014)
7. Kerl, C., Sturm, J., Cremers, D.: Dense visual SLAM for RGB-D cameras. In: Proc. of the Int. Conf. on Intelligent Robot Systems (IROS), pp. 2100–2106 (2013)
8. Kerl, C., Sturm, J., Cremers, D.: Robust odometry estimation for RGB-D cameras. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), May 2013

9. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2007), Nara, Japan, November 2007
10. Li, Q., Griffiths, J.G.: Least squares ellipsoid specific fitting. In: Proceedings of the Geometric Modeling and Processing, 2004, pp. 335–340. IEEE (2004)
11. Moonrinta, J., Chaivivatrakul, S., Dailey, M.N., Ekpanyapong, M.: Fruit detection, tracking, and 3d reconstruction for crop mapping and yield estimation. In: IEEE International Conference on Control, Automation, Robotics and Vision (2010)
12. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohli, P., Shotton, J., Hodges, S., Fitzgibbon, A.: Kinectfusion: real-time dense surface mapping and tracking. In: Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality. ISMAR 2011, pp. 127–136. IEEE Computer Society, Washington, DC (2011)
13. Qureshi, W.S., Satoh, S., Dailey, M.N., Ekpanyapong, M.: Dense segmentation of textured fruits in video sequences. In: 9th International Conference on Computer Vision Theory and Applications (VISAPP 2014), pp. 441–447 (2014)
14. Sarris, N., Strintzis, M.G.: 3D modeling and animation: Synthesis and analysis techniques for the human body. IGI Global (2005)
15. Shamir, A.: A survey on mesh segmentation techniques. In: Computer Graphics Forum, vol. 27, pp. 1539–1556. Wiley Online Library (2008)
16. Steinbruecker, F., Kerl, C., Sturm, J., Cremers, D.: Large-scale multi-resolution surface reconstruction from RGB-D sequences. In: IEEE International Conference on Computer Vision (ICCV), Sydney, Australia, pp. 3264–3271 (2013)
17. Steinbruecker, F., Sturm, J., Cremers, D.: Volumetric 3D mapping in real-time on a CPU. In: Int. Conf. on Robotics and Automation, Hongkong, China, pp. 2021–2028 (2014)
18. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: Proc. of the International Conference on Intelligent Robot Systems (IROS), October 2012
19. Torr, P.H., Zisserman, A.: MLESAC: A new robust estimator with application to estimating image geometry. Computer Vision and Image Understanding **78**(1), 138–156 (2000)
20. TUM Computer Vision Group: Dense Visual Odometry and SLAM (2013). https://github.com/tum-vision/dvo_slam
21. TUM Computer Vision Group: Volumetric 3D Mapping in Real-Time on a CPU (2014). https://github.com/tum-vision/fastfusion
22. Varady, T., Martin, R.R., Cox, J.: Reverse engineering of geometric modelsan introduction. Computer-Aided Design **29**(4), 255–268 (1997)
23. Wu, C.: Towards linear-time incremental structure from motion. In: 2013 International Conference on 3D Vision-3DV 2013, pp. 127–134. IEEE (2013)
24. Zuliani, M.: Ransac for dummies. Tech. rep., November 2008