3D Dense Reconstruction from 2D Video Sequence via 3D Geometric Segmentation

Bing Han, Christopher Paulson, and Dapeng Wu
Department of Electrical and Computer Engineering
University of Florida Gainesville, FL 32611
Correspondence author: Prof. Dapeng Wu, wu@ece.ufl.edu, http://www.wu.ece.ufl.edu

Abstract

3D reconstruction is a major problem in computer vision. This paper considers the problem of reconstructing 3D structures, given a 2D video sequence. This problem is challenging since it is difficult to identify the trajectory of each object point/pixel over time. Traditional stereo 3D reconstruction methods and volumetric 3D reconstruction methods suffer from the blank wall problem, and the estimated dense depth map is not smooth, resulting in loss of actual geometric structures such as planes. To retain geometric structures embedded in the 3D scene, this paper proposes a novel surface fitting approach for 3D dense reconstruction. Specifically, we develop an expanded deterministic annealing algorithm to decompose 3D point cloud to multiple geometric structures, and estimate the parameters of each geometric structure. In this paper, we only consider plane structure, but our methodology can be extended to other parametric geometric structures such as spheres, cylinders, and cones. The experimental results show that the new approach is able to segment 3D point cloud into appropriate geometric structures and generate accurate 3D dense depth map.

Index Terms

Geometric segmentation, surface fitting, dense matching, 3D reconstruction

I. Introduction

3D reconstruction is a major problem in computer vision. In the past, quite a few approaches have been developed for modeling and rendering 3D scene from 2D image sequences [1][2][3][4]. Currently, most of the systems and applications in 3D reconstruction are used for visual inspection and architecture modeling. However, there is an increasing demand for 3D entertainment, for example, 3D movies. The change of demand results in an attention for smooth visual quality of the reconstructed 3D scene. In this case, visual quality of reconstructed 3D scene becomes a dominant performance measure, while the foremost goal in previous approaches is accuracy of the estimated position of each point in 3D geometry.

In the last two decades, tremendous progress has been made on self-calibration and 3D surface modeling [5][6][7][8]. Most of the methods use 2D video sequences or 2D images as input and try to retrieve the depth information of the scene captured by the input images. The estimated depth helps to reconstruct the full 3D view of the scene. The existing techniques are able to accurately estimate the camera motion and compute a sparse depth map from the original image sequence [9][10][11][1][12][13][14]. Here, sparse depth map means a depth map that only contains depths of ‘a small portion’ of object points in a 3D scene. However, full reconstruction of a 3D scene requires knowledge of depth of every object point in a 3D scene. To obtain a dense depth map, one needs to solve the problem of dense matching/correspondence, i.e., alignment of every pixel between two adjacent input images [15][16][17].

A traditional solution to the dense matching problem is called epi-line searching. Epi-line search method uses geometric constraints to reduce the problem of 2D search to the problem of 1D search [18][19][20]. Although the search is constraint to 1D which seems easier to search, the blank wall problem, which is not solved in 2D feature correspondence, still exists in epi-line search. The blank wall problem is that given a textureless blank wall, it is very hard to find an accurate pixel-to-pixel correspondence between two input images.

Recently, more algorithms are proposed to solve the dense matching problem. Jangheon Kim and Thomas Sikora [21] proposed a dense disparity estimation algorithm which employs Gaussian scale-space with anisotropic disparity-field diffusion. Strecha and Van Gool [22] proposed a PDE based depth estimation algorithm based on the relative confidence that the system has in the data coming from the different views. Lhuillier and Quan proposed a quasi-dense approach to surface reconstruction in which they used a best first search based on combined 3D and 2D information [3][23]. Instead of using pixel-based searching and matching, volumetric reconstruction takes the scene as a tessellation of 3D cubes, called voxels. Each voxel may be either empty or occupied by the scene structure. Various methods have been proposed to build the volumetric model. Volumetric reconstruction could well recover the scene of the moving foreground, however, it is hard to reveal the static background structure using volumetric methods. In Arce and Marroquin paper [24], they proposed a stereo disparity reconstruction algorithm. By partitioning the reference image into a set of non-overlapping regions, the disparity reconstruction problem is formulated as a parametric segmentation problem.

In this paper, we propose a novel 3D dense reconstruction method based on geometric segmentation and surface fitting. We use the existing techniques for feature correspondence, projective reconstruction and self-calibration to achieve sparse 3D reconstruction. To obtain dense 3D reconstruction, we need to solve the dense matching problem. To address this, we use geometric segmentation to segment the 3D point cloud (obtained from sparse 3D reconstruction) into multiple geometric structures, and estimate the parameters of each geometric structure by surface fitting.

Different from previous 3D point segmentation algorithms [25][26][27][28][29], we develop an expanded deterministic annealing algorithm for geometric segmentation; under the assumption that each resulting geometric structure is a (linear) plane, we use surface fitting to estimate the depth of each object point in the geometric structures. Although we only consider plane structure in this paper, our methodology can be extended to other parametric geometric structures such as spheres, cylinders, and cones. Compared to the existing epi-line searching methods, our proposed approach is able to produce accurate dense depth map for textureless structures since we use geometric segmentation and surface fitting to bypass the blank wall problem. Compared to the existing volumetric reconstruction methods, our approach is able to produce accurate dense depth map for the static background structures due to geometric segmentation and surface fitting. In addition, our approach is able to generate smoother 3D dense reconstruction than the traditional methods. The experimental results show that our approach is able to segment 3D point cloud into appropriate geometric structures and generate accurate 3D dense depth map.

The rest of the paper is organized as follows. Section II describes the background and formulates the problem. Section III presents our system for 3D reconstruction. Section IV presents our methods for geometric segmentation and surface fitting. Experimental results are shown in Section V. Section VI concludes this paper.

II. Background and Problem Formation

In this section, we briefly review the 3D reconstruction techniques and formulate the geometric fitting problem mathematically.

A. 3D Reconstruction

3D reconstruction has been a major topic in computer vision for decades. Most 3D reconstruction approaches follow the same procedure [20] shown in Fig. 1.

As shown in Fig. 1, the first step in 3D reconstruction from a video sequence is to partition the whole video sequence into multiple scenes. For each scene, motion detection is needed to separate moving objects from the static background. Then, 3D reconstruction for moving foreground and static background are conducted separately; and the reconstructed 3D points from the foreground and background are combined to reconstruct the scene as a whole.

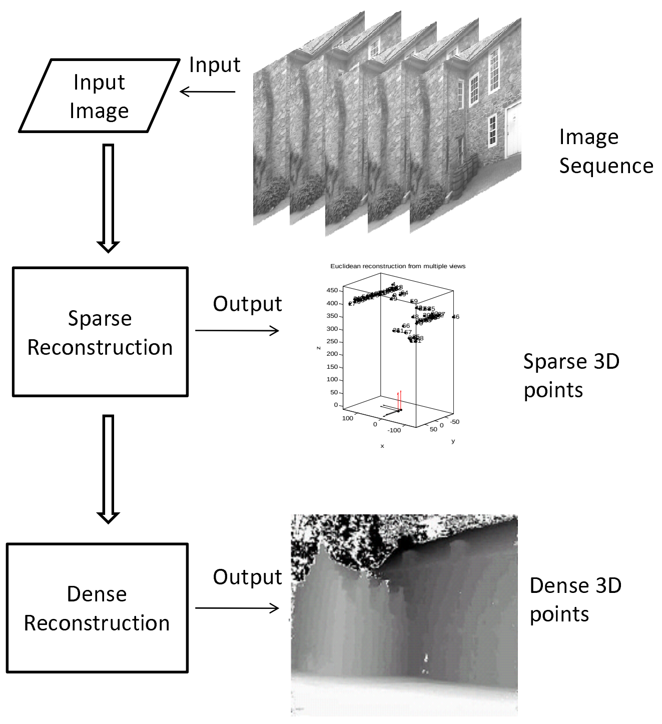


Fig. 1. The procedure for 3D video reconstruction.

The second step is sparse reconstruction. Sparse reconstruction consists of feature correspondence, projective reconstruction and Euclidean reconstruction. The camera motion is estimated and the Euclidean structure of the static background scene is recovered. For the moving regions, we introduce the virtual camera concept and apply the same reconstruction algorithm to recover the 3D structure. During the last two decades, tremendous progress has been made to camera self-calibration and structure computation. Sparse reconstruction starts from feature correspondence which is the most crucial part of the process. The goal of image correspondence, also called feature correspondence, is to identify two pairing points that are from two different image frames and correspond to the same 3D point [30][31]. It is known that not all points in a frame are suitable for matching or tracking; so only a few points are selected as feature points for matching [32]. Sparse reconstruction only relies on (sparse) feature point matching, which is different from dense reconstruction. Furthermore, feature points may be mismatched, known as outliers [33], which may degrade the accuracy of the sparse reconstruction result. After feature correspondence is done, projective reconstruction is conducted. Given correctly matched feature points from two input images, projective reconstruction is to find the relative pose between the two views. A projection is mathematically expressed by fundamental matrix. Given a sufficient number of matched feature point pairs, we are able to compute the fundamental matrix. The projective reconstruction is determined by a projective transformation. To identify this projective transformation, canonical decomposition is applied. However, the resulting projection is not suitable for visualization and an update to a full-fledged Euclidean reconstruction is required to recover the metric 3D geometric structure. The update to a metric structure, determined up to an unknown scalar factor, needs the information of intrinsic parameters of the camera. Since we have no prior knowledge of the camera, this approach is called self-calibration and has received a lot of attention in recent years. The approach we present here is called absolute conic constraint, or absolute quadric constraints.

The sparse reconstruction gives a sparse structure of the desired scene; however, it could not give a satisfied visual presentation. Thus, we still need to compute the depth of a lot more points, which is known as dense reconstruction or surface reconstruction. In this paper, we propose a novel approach to obtain the dense depth map. Unlike the previous approaches, we apply geometrical segmentation and surface fitting instead of dense searching and matching. Here we assume that the static background could be decomposed into multiple parametric surfaces, which allows us to partition the 3D point clouds into multiple clusters based on their geometric properties. For each cluster/region, we obtain a parametric model of the region via surface fitting. Under the assumption that each region has a sufficient number of feature point pairs, combined with the sparse depth map, we could then compute the depth of each pixel via the obtained parametric model of each region/sufface. Combining the depth maps of different regions, we finally obtain the depth map of the whole 3D scene. The merit of our approach is that it is able to generate smoother 3D dense reconstruction than the traditional methods. The geometric fitting problem is formulated in Section II-B and we give the solution to the geometric fitting problem in Section IV.

B. Geometric Fitting

The classic geometric fitting problem is to find a geometrical surface that best fits a set of 3D points. Geometric fitting is commonly used in 3D model fitting and 3D visual reconstruction in computer vision.

Given a 3D point data set $\mathcal{X} = \{\mathbf{y}_i\}, \mathbf{y}_i \in \mathbb{R}^3, i = 1, 2, ..., n$, a geometrical fitting problem can be formulated as the problem of minimizing a cost that measures how well the parametric model of a surface $g_\theta(\mathbf{y}) = 1$ fits the data set $\mathcal{X}$, where θ is the parameter vector of the surface model. A commonly used cost function is quadratic cost as below,

$$D = \frac{1}{n}\sum_{i=1}^{n}[d(\mathbf{y}_i, g_\theta)]^2 \tag{1}$$

$$d(\mathbf{y}_i, g_\theta) = \min_{\mathbf{y}\in\mathcal{S}} \|\mathbf{y}_i - \mathbf{y}\|_2, \tag{2}$$

where $\|\cdot\|_2$ is the Euclidean norm of a vector; and $\mathcal{S} = \{\mathbf{y} : g_\theta(\mathbf{y}) = 1\}$. The parameter θ of function $g_\theta(\mathbf{y})$ can be estimated by the method of least squares, i.e., minimizing D over θ.

There is another problem we need to consider here: how many surfaces are contained in the given 3D scene. Obviously, the more surfaces, the smaller D, resulting in over-fitting the data set $\mathcal{X}$. But too few surfaces will lead to under-fitting the data set $\mathcal{X}$. The principle of Occam's razor states that the simplest model that accurately represents the data is most desirable. So we prefer to use a few surface functions, which yield smoother surfaces that could well approximate the data set $\mathcal{X}$. Generally, there are three approaches to the over-fitting problem. The first approach is to add a penalty function to D; the penalty function increases with the number of surfaces. The second approach is to add smoothness or regularization constraints to the problem that minimizes (1). The third approach is to first assume a large number of surfaces and then identify a small number of (true) surfaces by compressive sensing techniques. Although the three approaches can generate parsimonious models, the three approaches typically use a gradient based computation procedure to solve the problem, which may produce local optimal solutions that are far away from the global optimal solution.

To improve the three approaches, one possible method is to use a clustering algorithm to partition the data set $\mathcal{X}$ into multiple clusters, each of which is used to estimate the parameter of a single surface. One of the most popular clustering algorithm is Lloyd algorithm, which starts with randomly selecting k initial centroids. Lloyd algorithm iteratively associates each point with the closest centroid and recalculates the centroids of the new clusters. Although widely used in real world applications, there are two serious limitations of Lloyd algorithm. The first limitation is that the partitioning result depends on the initialization of the cluster centroids, which may lead to poor local minima. The second limitation is that Lloyd algorithm can only partition points in linearly separable clusters rather than more complicated geometric structures such as spheres. In order to avoid initialization dependence, a simple but useful solution is to use multiple restarts with different initializations and select the best local minimum. Global k-means [34] is proposed to build the clusters deterministically, which use the original k-means algorithm as a local search step. At each step, global k-means add one more cluster based on previous partitioning result. Deterministic annealing [35] is another optimization technique to find a global minimum of a cost function. Deterministic

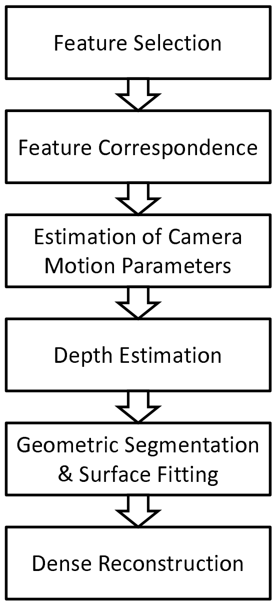

Feature Selection
Feature Correspondence
Estimation of Camera Motion Parameters
Depth Estimation
Geometric Segmentation & Surface Fitting
Dense Reconstruction

Fig. 2. Flowchart for our 3D video reconstruction system.

annealing explore a larger cost surface by adding a penalty on the product of temperature and the degree of randomness, i.e., entropy. At each iteration, the temperature is reduced and then a local optimization is performed. Finally, the temperature is reduced to zero and hence the imposed degree of randomness is reduce to zero, and the algorithm optimizes the original cost function D. Kernel method [36] can be used to mitigate the second limitation of Lloyd algorithm by nonlinearly mapping a data point in the input space to a higher dimensional feature space through an expanded transformation. Then the clustering problem is solved in the feature space. The linear separation in the feature space turns out to be a nonlinear separation in the original input space.

III. 3D Video Reconstruction

In this section, we describe our 3D reconstruction system. As shown in Fig. 2, our 3D reconstruction consists of the following steps: 1) feature points are extracted/selected from each image; 2) feature correspondence is established between any two consecutive images; 3) camera motion is estimated and the camera is calibrated; 4) the depth of each feature point is estimated; 5) our algorithm for geometric segmentation and surface fitting is applied to the 3D point cloud (corresponding to the feature points); 6) a dense depth map is obtained by using the parametric models of the surfaces. Next, we explain each component in Fig. 2.

A. Feature Selection

The first step in 3D reconstruction is to select candidate features in all images for tracking across different views. Point features can be identified by Harris' criterion [37], which is defined by

$$C(\mathbf{x}) = \det(G) + k \times \text{trace}^2(G) \tag{3}$$

where $\mathbf{x} = [x, y]^T$ is a candidate feature; $[\cdot]^T$ denotes the transpose of a vector/matrix; $C(\mathbf{x})$ is the quality of the feature; k is a user-specified parameter and G is a 2×2 matrix that depends on $\mathbf{x}$, given by

$$G = \begin{bmatrix} \sum_{W(\mathbf{x})} I_x^2 & \sum_{W(\mathbf{x})} I_x I_y \\ \sum_{W(\mathbf{x})} I_x I_y & \sum_{W(\mathbf{x})} I_y^2 \end{bmatrix} \tag{4}$$

where $W(\mathbf{x})$ is a rectangular window centered at $\mathbf{x}$ and I_x and I_y are the gradients along the x and y directions which can be obtained by convolving the image I with the derivatives of a pair of Gaussian filters. The size of the window can be decided by the user, for example 7×7. If $C(\mathbf{x})$ exceeds a user-specified threshold, then the point $\mathbf{x}$ is selected as a candidate point feature.

B. Feature Correspondence

Once the candidate point features are selected, the next step is to identify a pair of feature points (in two consecutive images), which correspond to the same object point in 3D space. The classic image alignment algorithm was the Lucas-Kanade algorithm [30]. In this subsection, we present the Lucas-Kanade algorithm for feature correspondence based on a translational model.

We use sum of squared differences (SSD) to quantify the dissimilarity of two point features. The objective of Lucas-Kanade algorithm is to minimize the SSD between two images. The feature correspondence problem is formulated by the following optimization problem:

$$\min_{\mathbf{d}} \sum_{\mathbf{x} \in W(\mathbf{x})} [I_2(\mathbf{x}+\mathbf{d}) - I_1(\mathbf{x})]^2 \tag{5}$$

where $\mathbf{d}$ is the displacement of a feature point at position $\mathbf{x}$ in Frame I_1, with respect to Frame I_2. The solution to (5) is given by

$$\mathbf{d} = -G^{-1}\mathbf{b}$$

where

$$\mathbf{b} \doteq \begin{bmatrix} \sum_{W(\mathbf{x})} I_x I_t \\ \sum_{W(\mathbf{x})} I_y I_t \end{bmatrix} \tag{6}$$

and G is given by (4), and $I_t \doteq I_2 - I_1$.

C. Estimation of Camera Motion Parameters

In this subsection, we present a method for estimating camera motion parameters from the established feature correspondence [38]. We will follow the notation used in Ref. [7].

The method is based on a perspective projection model with a pinhole camera. Suppose we have a generic point $p \in \mathbb{R}^3$ with coordinates $\mathbf{X}_p = [X, Y, Z, 1]^T$ relative to a world coordinate frame. Given two image frames of one scene which is related by a motion $g = (R, T)$, the two image projection point $\mathbf{x}_1$ and $\mathbf{x}_2$ are related as follows:

$$\lambda_1 \check{\mathbf{x}}_1 = \Pi_1 \mathbf{X}_p, \quad \lambda_2 \check{\mathbf{x}}_2 = \Pi_2 \mathbf{X}_p \tag{7}$$

where $\check{\mathbf{x}} = [x, y, 1]^T$ is measured in pixels; λ_1 and λ_2 are the depths of $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively; $\Pi_1 = [J, 0]$ and $\Pi_2 = [JR, JT]$ are the camera projection matrices; and J is the camera calibration matrix.

The camera calibration matrix J is also known as the intrinsic matrix. It contains 5 intrinsic parameters, including camera focal length, image format, and principal point. There are two classic approaches for camera calibration, one is Tsai's two stage algorithm [39] and the other is Zhang's algorithm [1] based on homography constrains. In our paper, we use Tsai's algorithm to estimate the camera calibration matrix J.

In order to estimate λ_1, λ_2, Π_1 and Π_2, we need to introduce the epipolar constraint. From Eq. (7), we have

$$\check{\mathbf{x}}_2^T J^{-T} \hat{T} R J^{-1} \check{\mathbf{x}}_1 = 0 \tag{8}$$

The fundamental matrix is defined as:

$$F_m \doteq J^{-T} \hat{T} R J^{-1} \tag{9}$$

With the above model, we could estimate the fundamental matrix F_m via the eight-point algorithm [7]. Then we could decompose the fundamental matrix to recover the projection matrices Π_1 and Π_2 and the 3D structure. We only give the solution here by canonical decomposition [38]:

$$\lambda_1 \check{\mathbf{x}}_1 = \mathbf{X}_p, \lambda_2 \check{\mathbf{x}}_2 = (\widehat{T'})^T F_m \mathbf{X}_p + T' \tag{10}$$

D. Depth Estimation

The Euclidean structure $\mathbf{X}_e$ is related to the projective reconstruction $\mathbf{X}_p$ by a linear transform $H \in \mathbb{R}^{4\times4}$,

$$\Pi_{ip} \sim \Pi_{ie} H^{-1}, \mathbf{X}_p \sim H\mathbf{X}_e, i = 1, 2, ..., m \tag{11}$$

where $\sim$ means equality up to a scale factor and

$$H = \begin{bmatrix} J & 0 \\ -\nu^T J & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \tag{12}$$

With the assumption that J is constant, we could estimate the unknowns J and ν with a gradient decent optimization algorithm [7]. In order to obtain a unique solution, we also assume that the scene is generic and the camera motion is rich enough.

E. Geometric Segmentation and Surface Fitting

One disadvantage of point feature correspondence is that only a small number of points are qualified as feature points. So, with the depth estimation, the previous 3D depth map recovery is sparse. The problem with sparse structure is that it is not suitable for human visualization. Therefore, in this paper, we propose a new dense matching method based on geometric segmentation.

We first segment the 3D point cloud (obtained from sparse 3D reconstruction) into multiple regions, each of which represents a geometric structure, i.e., surface. For simplicity, we model each surface by a parametric plane model. With the depth information of the feature points obtained from the sparse 3D reconstruction, we can compute the depth of each pixel on a plane, via the parametric model of the plane. Since the depth information we obtained is based on a plane model, the image rendered from the 3D model is much smoother than the traditional approaches. In order to simplify the problem of surface fitting (i.e., estimation of the parameters of the plane model), we first segment the input image based on its geometric structure. It is different from the traditional object-based image segmentation. The segmentation process is critical because proper segmentation could simplify the surface fitting. On the contrary, improper segmentation will result in too many surfaces/planes, leading to high complexity in surface fitting. Our algorithm for geometric segmentation and surface fitting will be presented in Section IV.

F. Dense 3D Reconstruction

Here, we only consider two images. Suppose for the first image, we have the 3D point set $\{\mathbf{X}_e^j\}_{j=1}^n$, which could be divided into K clusters, denoted by $\mathcal{X}_1$, $\mathcal{X}_2$, $\cdots$, $\mathcal{X}_K$. For each cluster, we assume there are at least three non-collinear points, which usually can be satisfied. Then we model each cluster by a plane. We use $\mathcal{X}_1$ as an example. We can use the following plane model to fit the points in $\mathcal{X}_1$,

$$\mathbf{X} \cdot \mu = 1 \tag{13}$$

where $\mu = [a, b, c, 0]^T$ is the plane parameter vector and can be estimated by the least squares method, given 3D points in $\mathcal{X}_1$.

For an arbitrary image point $\mathbf{x}^i = [x^i, y^i]^T$, which is an image of a 3D point in $\mathcal{X}_1$, we could estimate its depth λ^i by solving the following equation,

$$\lambda^i \tilde{\mathbf{x}}^i = H_1^{-1} \Pi_1 \mathbf{X}_e^i \tag{14}$$

where $\tilde{\mathbf{x}}^i = [x^i, y^i, 1]^T$, H_1^{-1} and Π_1 are estimated by Eq. (10) and Eq. (12), respectively. In Eq. (14), only λ^i is unknown and with the constraint on $\mathbf{X}_e^i$ via Eq. (13), we can obtain the value of λ^i.

Since we are able to obtain the depth of any image point via the above procedure, we can obtain a dense depth map of each frame.

IV. Geometric Segmentation and Surface Fitting

As we have discussed, not all points in an image are suitable for matching or tracking. The feature points that we have selected are only a small portion of a whole image. Therefore, the first reconstruction is a sparse 3D reconstruction. The sparse structure is not suitable for human visualization. For this reason, a dense matching is necessary to establish a 3D geometric view. It is known that the most popular solution for dense matching is based on the epi-polar constraint. This approach uses geometric constraints to restrict correspondence search from 2D to 1D range. The main disadvantage of this approach is that the dense depth map is not smooth because of outliers. Lhuillier and Quan [3] proposed a dense matching method called quasi-dense approach. However, the non-smoothness problem still exists.

In this section, we propose an expanded deterministic annealing approach for space partitioning and surface fitting in 3D Euclidean space. Under the assumption that the 3D scene under study consists of a few geometric structures, we design a non-linear function to map the data point from geometrical space to surface model space and apply deterministic annealing in the feature space to partition the feature space into multiple regions. For each region, we use a linear plane model to fit the 3D points in the region. We call our method as expanded deterministic annealing method. Our method has three merits: 1) the ability to avoid many poor local optima; 2) the ability to minimize the cost function even if its gradients vanish almost everywhere; 3) the ability to achieve non-linear separation of 3D points. However, there is no closed form solution to the expanded deterministic annealing problem; therefore we use a gradient descent algorithm to solve this problem. Next, we present the problem of geometric segmentation and surface fitting.

Given a set of 3D points $\{\mathbf{y}_i\}$, we would like to find multiple geometric surfaces that best fit the 3D point cloud $\{\mathbf{y}_i\}$. The problem can be formulated as below

$$\min_{\{\theta_k\}_{k=1}^K} \sum_{k=1}^{K} \sum_{\mathbf{y}_i \in \mathcal{X}_k} d(\mathbf{y}_i, g_{\theta_k}) \tag{15}$$

where K is the number of surfaces (i.e., planes here); $\{\theta_k\}_{k=1}^K$ denotes the set $\{\theta_1, \theta_2, \cdots, \theta_K\}$; $\mathcal{C}_k$ denotes the cluster of 3D points that belong to the k-th plane; $\mathbf{y}_i = [x_i, y_i, z_i]^T$ is the i-th point; $\theta_k = [a_k, b_k, c_k]^T$ is the parameter vector of the k-th plane model, where $\frac{1}{a_k}$, $\frac{1}{b_k}$, and $\frac{1}{c_k}$ are intercepts of the plane on x-axis, y-axis, and z-axis, respectively; and $d_{i,k}^2$ is the squared distance (fitting error) between $\mathbf{y}_i$ and plane model $g_{\theta_k}(\mathbf{y}) = \mathbf{y}^T\theta_k = 1$, which is defined as

$$d_{i,k}^2 = d^2(\mathbf{y}_i, g_{\theta_k}) = (\mathbf{y}_i^T\theta_k - 1)^2 \tag{16}$$

This is a joint problem of model selection and parameter estimation, i.e., we need to determine how many surfaces (or the number of clusters of 3D points) and estimate the parameters of the parametric model of each surface. This problem is particularly challenging because the more surfaces, the smaller fitting error but the higher probability of over-fitting; the fewer surfaces, the larger fitting error.

The problem in (15) can be solved by deterministic annealing (DA) [35]. The DA approach to clustering has demonstrated substantial performance improvement over traditional supervised and unsupervised learning algorithms. DA mimics the annealing process. DA works as below. First, it minimizes the cost function subject to a constraint on the degree of randomness of the solution. The constraint on Shannon entropy, is gradually shrunk as the temperature reduces, and the constraint eventually vanishes as the temperature goes to zero; hence the solution of DA converges to the solution of minimizing the original cost function. Similar to the simulated annealing [40], the cooling schedule allows DA to avoid many poor local optima. The DA approach has been adopted in a variety of research fields, such as graph-theoretic optimization and computer vision. Rao et al. [41] apply DA to solving a piecewise regression problem.

In this paper, we propose a new approach, called expanded deterministic annealing (EDA), to solve the geometric segmentation and surface fitting problem. Specifically, we first use a non-linear function to map the input 3D points to a high dimensional feature space using the local geometric structure of each

3D point. Then we apply deterministic annealing to the points in the feature space for clustering (i.e., geometric segmentation) and surface fitting. Different from Ref. [41], our EDA approach leverages local geometric structure for clustering. Next, we present our EDA approach.

The input data is a set of 3D points, $\mathbf{y}_i = [x_i, y_i, z_i]^T$ $(i = 1, \cdots, N)$. Under the assumption that L nearest 3D points of a given point $\mathbf{y}_i$ are on the same local plane, we use the least squares method to estimate this local plane model parameters, denoted by $\mathbf{L}_i = [a_i, b_i, c_i]^T$. Let

$$\mathbf{f}_i = \begin{bmatrix} \mathbf{y}_i \\ \mathbf{L}_i \end{bmatrix}$$

So we expand a 3D point $\mathbf{y}_i$ to 6D point $\mathbf{f}_i$; then we apply DA to this expanded 6D space to solve the geometric segmentation and surface fitting problem. Define

$$P_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix},$$

$$P_2 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

In our EDA algorithm, we use the following (new) distortion function

$$D(\mathbf{f}_i, g_{\theta_k}) = \beta \times D_1(P_1\mathbf{f}_i, g_{\theta_k}) + D_2(P_2\mathbf{f}_i, g_{\theta_k}), \tag{17}$$

where $D_1(\mathbf{y}_i, g_{\theta_k}) = d_{i,k}^2$, which is defined in (16); $D_2(P_2\mathbf{f}_i, g_{\theta_k})$ is defined by

$$D_2(P_2\mathbf{f}_i, g_{\theta_k}) = 1 - (P_2\mathbf{f}_i^T \cdot \theta_k)^2; \tag{18}$$

and β is a positive real number, which balances the two types of distortions D_1 and D_2. Note that D_1 quantifies the fitting error between a given 3D point $\mathbf{y}_i$ and the global plane, which $\mathbf{y}_i$ belongs to; D_2 quantifies the difference between the local plane model of $\mathbf{y}_i$ and the global plane model of $\mathbf{y}_i$; and $P_2\mathbf{f}_i^T \cdot \theta_k$ is a cosine similarity between the two plane models. One novelty of our EDA algorithm is the introduction of D_2, which can be regarded as locally averaged distortion and help mitigate the effect of outliers, i.e., an outlier 3D point only contributes distortion D_1 weighted by β. E.g., if we choose $\beta = 0.1$, then the contribution from D_1 is reduced by a factor of 10.

Denote the number of clusters by K. We apply DA to partition $\{\mathbf{y}_i\}$ into K clusters and estimate the parameters of planes, each of which corresponds to one cluster. Since K is not a given parameter, our EDA algorithm will search for the optimal value of K, as shown in Algorithm 1. For a given value K, our EDA algorithm solves the following problem.

$$\min_{\{\theta_k\}_{k=1}^K} F = D - TE \tag{19}$$

where $\theta_k = [a_k, b_k, c_k]^T$ $(k = 1, \cdots, K)$ is the surface model parameter to be estimated; D is defined in (17); and E is the entropy constraint. We define D and E as follows:

$$D = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} P(\mathbf{y}_i \in g_{\theta_k}) \times D(\mathbf{f}_i, g_{\theta_k}), \tag{20}$$

$$E = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} P(\mathbf{y}_i \in g_{\theta_k}) \times \log P(\mathbf{y}_i \in g_{\theta_k}), \tag{21}$$

where

$$P(\mathbf{y}_i \in g_{\theta_k}) = \frac{\exp(-\frac{D(\mathbf{f}_i, g_{\theta_k})}{T})}{\sum_{j=1}^{K} \exp(-\frac{D(\mathbf{f}_i, g_{\theta_j})}{T})} \tag{22}$$

We use a gradient descent algorithm to solve the problem (19) as shown in Algorithm 1. We explain Step 2 of Algorithm 1 as below. Since a plane model is specified by equation $g_\theta(\mathbf{y}) = \mathbf{y}^T\theta = 1$, given 3D points $\{\mathbf{y}_i\}_{i=1}^N$, for $K = 1$, we can estimate the parameter vector θ of the plane that fits all the 3D points, by solving the following problem

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{N} (\mathbf{y}_i^T\theta - 1)^2. \tag{23}$$

Hence, $\hat{\theta}$ can be obtained by the least squares solution as below

$$\hat{\theta} = (\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\vec{1}_N, \tag{24}$$

where $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_N]^T$ is a matrix of dimension $N \times 3$, and $\vec{1}_N = [1, 1, \cdots, 1]^T$ is a vector of dimension N. In Step 4b, $\Theta = [\theta_1^T, \theta_2^T, \cdots, \theta_K^T]^T$; $\nabla_\Theta F$ denotes the gradient of F with respect to Θ.

V. Experimental Results

In this section, we conduct experiments to demonstrate that our proposed approach is able to segment 3D point cloud into appropriate geometric structures and generate accurate 3D dense depth map. The rest of the section is organized as below. Section V-A shows the estimation accuracy of our proposed Algorithm 1 (EDA) for synthetic data with the knowledge of ground truth. In Section V-B, we present the results of the estimation accuracy of Algorithm 1 for noisy synthetic data with the knowledge of ground truth. Section V-C investigates the estimation accuracy of Algorithm 1 for real-world data.

In this section, we compared three geometric segmentation algorithms, Projection based iterative geometric segmentation algorithm (PI), Adaptive projection based iterative algorithm (API), and expanded DA based geometric segmentation algorithm(EDA), based on both synthetic data and real world data.

A. EDA on Synthetic Data without Noise

In this section, we show the estimation accuracy of Algorithm 1 (EDA) for synthetic data with the knowledge of ground truth; the synthetic data here does not contain noise. We also compare EDA to Algorithm 2 and Algorithm 3. Algorithm 2 is a projection based iterative geometric segmentation algorithm (PI for short); we design Algorithm 2 (PI) based on the same principle of the Lloyd algorithm (a.k.a., K-means). Algorithm 3 is an adaptive projection based iterative algorithm (API for short); we design Algorithm 3 (API) based on the same principle of ISODATA, which generalizes K-means by allowing K to be unspecified.

To generate synthetic data, we first determine the number of planes, denoted by K; then specify the analytic form of each of the K planes and the area of each plane; for each plane, we uniformly generate 100 3D points on the plane area. The same data set is applied to the three algorithms. In the experiment, we run each algorithm 1000 times; in different run, a different set of 3D points are (randomly) generated; then we compute the average performance of each algorithm in terms of average squared approximation error and correct identification rate. Table I shows average squared approximation error for PI, API, and EDA algorithms, where the squared approximation error is quantified by the Euclidean norm of the estimation error for the plane model parameters. We test four different number of planes, i.e., $K = 3, 4, 5, 6$. As observed from Table I, the approximation error of EDA is negligible comparing to that of PI and API, which demonstrates that the EDA algorithm significantly outperforms both PI and API algorithms in terms of estimation accuracy for the plane models. This is because EDA is able to separate the 3D point cloud in a non-linear manner, and can avoid many poor local optima.

Algorithm 1 EDA based joint geometrical segmentation and surface fitting

```
1) **Input**: $\mathbf{y}_1$, $\mathbf{y}_2$, ..., $\mathbf{y}_N$;
    $K_{max}$: maximum number of clusters
    $T_{init}$: starting temperature
    $T_{min}$: minimum temperature
    $\alpha$: cooling rate (must be $< 1$)
    $I_{max}$: maximum iteration number
    $\varepsilon$: threshold for $F$
    $\epsilon$: threshold for plane merging
    $\sigma^2$: variance of Gaussian perturbation
2) **Initialization**
    Compute $\theta_1$ via (24); $T = T_{init}; K = 2; \theta_2 = \theta_1; P(\mathbf{y}_i \in g_{\theta_1}) = P(\mathbf{y}_i \in g_{\theta_2}) = \frac{1}{2}, \forall i.$
3) **Perturb**
    Generate Gaussian vector $\delta_k$ of zero mean and covariance matrix $\sigma^2 I$; $\theta_k \leftarrow \theta_k + \delta_k$ $(k = 1, 2)$;
    $F_{old} = D - TH$;
    j=0;
4) **Loop until convergence**
4a) For each $i$ and each $k$, compute $P(\mathbf{y}_i \in g_{\theta_k})$ via (22);
4b) Update the surface models
        $\Theta \leftarrow \Theta - \gamma \nabla_\Theta F$ ($\gamma$ is obtained by Armijo rule);
    $F = D - TE$;
    $j = j + 1$;
    If ($j < I_{max}$ and $(F_{old} - F)/F_{old} > \varepsilon$)
        $F_{old} = F$; Goto Step 4a;
5) **Model size determination**
    { if ($||\theta_k - \theta_m||_2 < \epsilon$), then replace $\theta_k$ and $\theta_m$ by $(\theta_k + \theta_m)/2$ } $\forall k, m$;
    $K$ = number of planes after merging;
6) **Cooling**
    $T = \alpha T$;
    If ($T < T_{min}$)
        perform Step 4a, 4b and Step 5 for $T = 0$
        Goto Step 9
7) **Duplication**
    Replace each plane by two planes at the same location; $K = 2K$;
8) **Goto Step 3**
9) **Output**: $\{\theta_k\}_{k=1}^{K}$.
```

TABLE I

AVERAGE SQUARED APPROXIMATION ERROR.

K	PI	API	EDA
3	3.77×10^{-1}	3.00×10^{-9}	1.17×10^{-12}
4	4.01×10^{-1}	9.81×10^{-8}	2.21×10^{-12}
5	2.43×10^{-1}	2.86×10^{-9}	3.06×10^{-12}
6	2.94×10^{-1}	8.801×10^{-9}	3.00×10^{-12}
10	5.38×10^{-1}	3.32×10^{-8}	7.81×10^{-10}
20	1.01	6.73×10^{-7}	5.28×10^{-8}

Algorithm 2 PI based joint geometrical segmentation and surface fitting

```
1) **Input**: $\mathbf{y}_1$, $\mathbf{y}_2$, ..., $\mathbf{y}_N$;
       $K$: maximum number of clusters
       $I_{max}$: maximum iteration number
       $\varepsilon$: threshold for $\theta$
2) **Initialization**
       $\forall i$, use uniform distribution $\frac{1}{K}$ to generate index $k \in \{1, \cdots, K\}$ of the cluster that $\mathbf{y}_i$ belongs to;
       $\forall k$, estimate the plane parameter $\theta_k$ via (24), using only $\{\mathbf{y}_i \in \mathcal{C}_k\}$;
       $\forall k$, $\theta_k^{old} = \theta_k$;
       j = 0;
3) **Loop until convergence**
3a)  $\forall i$, compute $\hat{k} = \arg\min_k (\mathbf{y}_i^T \theta_k - 1)^2$, and assign $\mathbf{y}_i$ to cluster $\mathcal{C}_{\hat{k}}$;
3b)  Update the surface models
         $\forall k$, estimate the plane parameter $\theta_k$ via (24), using only $\{\mathbf{y}_i \in \mathcal{C}_k\}$;
3c)  j=j+1;
3d)  If ($j < I_{max}$ and $((\theta_k^{old} - \theta_k)/\theta_k^{old} > \varepsilon$, for some $k$))
         $\forall k$, $\theta_k^{old} = \theta_k$; Goto Step 3a;
4) **Output**: $\{\theta_k\}_{k=1}^K$.
```

TABLE II
CORRECT IDENTIFICATION RATE.

K	PI	API	EDA
3	83%	96%	99%
4	79%	93%	99%
5	82%	94%	97%
6	78%	97%	98%
10	73%	92%	95%
20	66%	88%	91%

Table II shows correct identification rate for PI, API, and EDA algorithms. The correct identification rate is quantified by the percentage of 3D points whose plane memberships are correctly identified. Note that there are K planes and we have the ground truth of which plane a 3D point belongs to. We observe that the correct identification rates of EDA and API are much higher than that of the PI algorithm. The reason why the API algorithm outperforms the PI algorithm is that the API algorithm is not sensitive to initialization while the PI algorithm is very sensitive to initialization. Again, EDA performs the best among the three algorithms in terms of correct identification rate. This is again because EDA is able to separate the 3D point cloud in a non-linear manner, and can avoid many poor local optima. We also observe that if the number of plane is too large, like $K > 10$, the performance get worse. The reason is that as the total number of planes increases, it becomes harder to distinguish two similar planes and it becomes possible that many of the points may be classified into wrong categories. Therefore, we consider it as one of the limitations of the EDA algorithm.

B. EDA on Synthetic Data with Noise

In this section, we show the estimation accuracy of Algorithm 1 (EDA) for synthetic data with the knowledge of ground truth; the synthetic data here contains additive white Gaussian noise. We also compare EDA to Algorithm 2 and Algorithm 3.

Algorithm 3 API based joint geometrical segmentation and surface fitting

```
1) **Input**: $\mathbf{y}_1$, $\mathbf{y}_2$, ..., $\mathbf{y}_N$;
     $K_{max}$: maximum number of clusters
     $K_{min}$: minimum number of clusters
     $I_{max}$: maximum iteration number
     $N_{min}$: minimum number of points in a cluster
     $\varepsilon$: threshold for $\theta$
     $\sigma^2$: variance of Gaussian perturbation
2) **Initialization**
     $\forall i$, use uniform distribution $\frac{1}{K}$ to generate index $k \in \{1, \cdots, K\}$ of the cluster that $\mathbf{y}_i$ belongs to;
     $\forall k$, estimate the plane parameter $\theta_k$ via (24), using only $\{\mathbf{y}_i \in \mathcal{C}_k\}$;
     $\forall k$, $\theta_k^{old} = \theta_k$;
     j = 0;
3) **Loop until convergence**
3a)  $\forall i$, compute $\hat{k} = \arg\min_k (\mathbf{y}_i^T \theta_k - 1)^2$, and assign $\mathbf{y}_i$ to cluster $\mathcal{C}_{\hat{k}}$;
3b)  Update the surface models
          $\forall k$, estimate the plane parameter $\theta_k$ via (24), using only $\{\mathbf{y}_i \in \mathcal{C}_k\}$;
     Discard those clusters whose sizes are less than $N_{min}$;
     $K$ = number of planes after discarding clusters;
     $\forall \mathbf{y}_i$ in the discarded clusters, compute $\hat{k} = \arg\min_k (\mathbf{y}_i^T \theta_k - 1)^2$, and assign $\mathbf{y}_i$ to cluster $\mathcal{C}_{\hat{k}}$;
     Perform Step 3b;
     If $(K < K_{min})$, CALL subroutine **Split**;
     If $(K > K_{max})$, CALL subroutine **Merge**;
     If ($j < I_{max}$ and $((\theta_k^{old} - \theta_k)/\theta_k^{old} > \varepsilon$, for some $k$))
          $\forall k$, $\theta_k^{old} = \theta_k$; Goto Step 3a;
     Else Goto Step 6;
4) **Split**
     Replace each plane by two planes at the same location; $K = 2K$;
     Generate Gaussian vector $\delta_k$ of zero mean and covariance matrix $\sigma^2 I$; $\theta_k \leftarrow \theta_k + \delta_k$ $(\forall k)$;
     Perform Step 3a and 3b; return.
5) **Merge**
     Find $k$ and $m$ that achieve $\min_{k \neq m} ||\theta_k - \theta_m||_2$; then replace $\theta_k$ and $\theta_m$ by $(\theta_k + \theta_m)/2$;
     $K = K - 1$; perform Step 3a and 3b; return.
6) **Output**: $\{\theta_k\}_{k=1}^K$.
```

In the experiment, for each 3D point, the variance of its additive white Gaussian noise is $1/100$ of the area of the plane where the 3D point is located. In the experiment, we run each algorithm 1000 times; in different run, a different set of 3D points and their additive white Gaussian noise are (randomly) generated; then we compute the average performance of each algorithm in terms of average squared approximation error and correct identification rate. Table III shows average squared approximation error for PI, API, and EDA algorithms, where the squared approximation error is quantified by the Euclidean norm of the estimation error for the plane model parameters. As observed from Table I, the EDA algorithm outperforms both PI and API algorithms in terms of estimation accuracy for the plane models. However, the performance gain of EDA is much less than that for the noiseless case. The reason is that the non-linear mapping in EDA depends on the accuracy of estimation of the local geometric structures and the estimation of the local geometric structures is very sensitive to the added noise.

Table IV shows correct identification rate for PI, API, and EDA algorithms. We observe that EDA

TABLE III

AVERAGE SQUARED APPROXIMATION ERROR UNDER NOISY DATA.

K	PI	API	EDA
3	6.61×10^{-1}	8.96×10^{-1}	2.41×10^{-1}
4	8.18×10^{-1}	5.98×10^{-1}	3.19×10^{-1}
5	6.98×10^{-1}	4.42×10^{-1}	3.96×10^{-1}
6	1.16	9.44×10^{-1}	6.71×10^{-1}

TABLE IV

CORRECT IDENTIFICATION RATE UNDER NOISY DATA.

K	PI	API	EDA
3	77%	92%	94%
4	80%	92%	93%
5	72%	91%	95%
6	74%	89%	93%

performs the best among the three algorithms. Since EDA is not suitable for large number of planes. In this experiment, we only perform comparison among all three algorithms with $K < 10$.

C. EDA on Real World Data

In this section, we show the geometric segmentation accuracy of Algorithm 1 (EDA) for real world data. We also compare EDA to Algorithm 2 and Algorithm 3.

The real world data we use is the ‘house’ video sequence. We first apply the schemes in Section III-A through Section III-D to the ‘house’ video sequence, and obtain 72 3D points. Specifically, Fig. 3 shows 72 feature points selected from the 1st frame of the ‘house’ video sequence, by the scheme in Section III-A; Fig. 4 shows two frames used for estimating the depths of the 72 feature points; Fig. 5 shows the estimated camera pose of the 1st and 88th frame, and the estimated 3D positions of the 72 feature points, by the scheme in Section III-C and Section III-D, respectively. From Fig. 3, it can be seen that most of the 72 feature points are on the walls of the house in the image.

We use the EDA algorithm to segment the 72 3D points shown in Fig. 5, and identify the geometric structures (i.e., the walls) where the 3D points are embedded. Fig. 6 shows the geometric segmentation result of the EDA algorithm, and Fig. 7 shows the resulting geometric structures (i.e., planes). Note that Fig. 3 is a front view of the house while Fig. 6 and Fig. 7 are top views of the house. From Fig. 6 and Fig. 7, it is observed that the EDA algorithm segments the 3D points into three groups, each of which correctly corresponds to a wall in the image; in addition, most of the 3D points are correctly classified into the true geometric structure (i.e., wall). Since the EDA algorithm is able to correctly identify the true geometric structures of the scene, the EDA algorithm produces accurate dense 3D reconstruction as shown in Fig. 7.

Fig. 8 and Fig. 9 show the geometric segmentation result of the PI and API algorithm, respectively. It is observed that both the PI and API algorithm produce poor geometric segmentation result; in other words, PI and API algorithms fail to identify the true geometric structures of the scene. Since the geometric segmentation results are very poor, we do not show the parametric models produced by the PI and API algorithm.

VI. CONCLUSION

In this paper, we propose a novel approach to the problem of dense 3D reconstruction. Our approach is based on geometric segmentation and surface fitting. We first use the existing techniques for feature

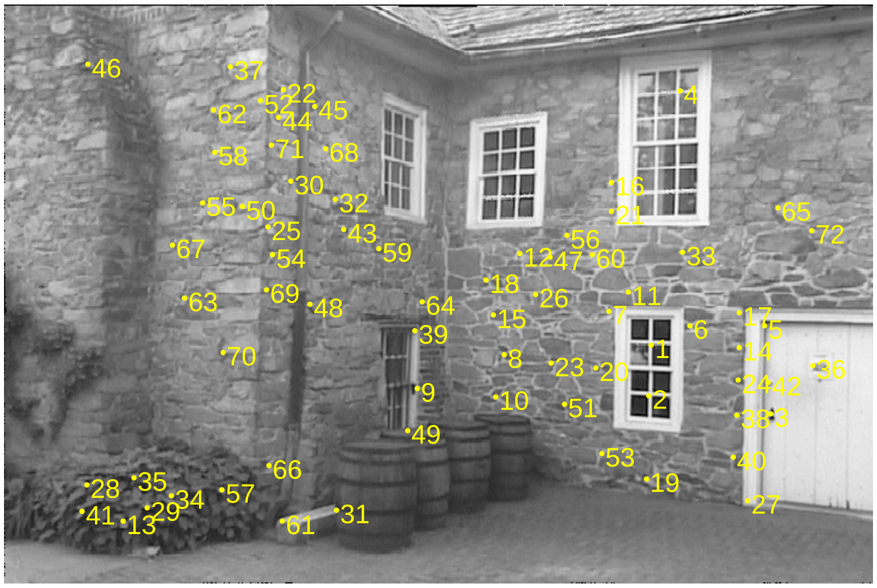


Fig. 3. Feature points on the 1st frame of the 'house' video sequence.

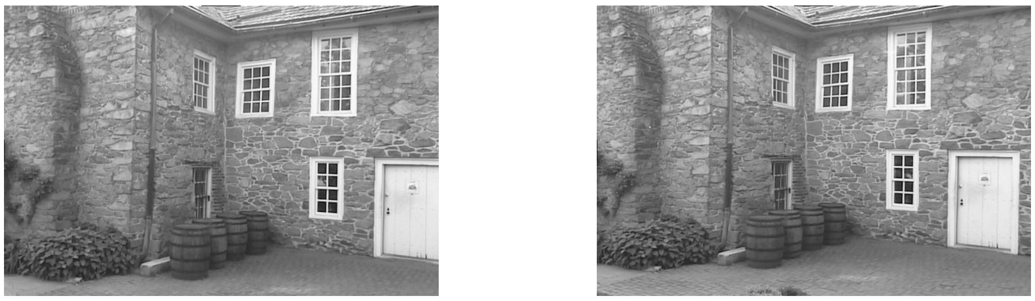
selection, feature correspondence, projective reconstruction and self-calibration to obtain sparse 3D reconstruction. Then, we use an expanded Deterministic Annealing (EDA) algorithm, which is proposed in this paper, to segment the 3D space into multiple geometric structures; at the same time, the EDA algorithm also produces estimates of the parameters of each geometric structure. The resulting geometric structures help achieve dense 3D reconstruction; i.e., for each pixel in an image, its depth can be estimated by computing the parametric model of the geometric structure that the pixel belongs to. Experimental results demonstrate that the EDA algorithm is able to segment the 3D space in a non-linear manner, and is more accurate in geometric segmentation and surface fitting, compared to the PI and API algorithms. Our proposed approach to dense 3D reconstruction can generate smoother dense map, compared to the traditional methods.

A limitation of our proposed method is that, just like every feature-point-based algorithm, our method also suffers from the well-known "blank wall" problem (i.e., the problem of having no texture); this is because our method relies on texture so that feature points can be extracted.

In our future work, we will develop surface fitting algorithms for non-linear surface models, e.g., sphere surface. The EDA algorithm used in this paper is based on a plane model $g_{\theta_k}(\mathbf{y})$. In order to apply EDA for non-linear surface models, we need to use a non-linear surface model $g_{\theta_k}(\mathbf{y})$ and related distortion function $d(\mathbf{y}_i, g_{\theta_k})$. The challenge is that if we simply substitute the plane model with the non-linear surface model, the computational complexity is much higher. Therefore, optimization is needed to apply the EDA for non-linear surface model.

Disclaimers

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFRL or the U.S. Government.



(a) The 1st frame in the ‘house’ video sequence (b) The 88th frame in the ‘house’ video sequence

Fig. 4. Two frames used for sparse depth estimation.

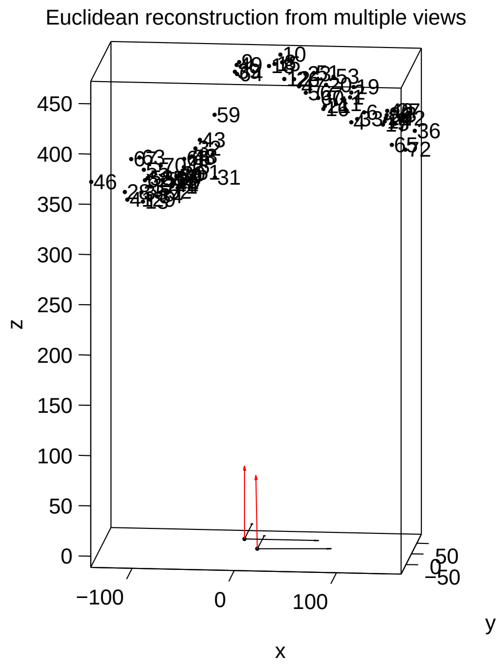


Fig. 5. Estimated camera pose of the 1st and 88th frame, and estimated 3D positions of feature points.

Acknowledgement

This material is based on research sponsored by AFRL under agreement number FA8650-06-1-1027. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

References

[1] Z. Zhang, “A flexible new technique for camera calibration,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 11, pp. 1330–1334, 2002.

[2] C. Strecha, T. Tuytelaars, and L. Van Gool, “Dense matching of multiple wide-baseline views,” in *International Conference on Computer Vision*, vol. 2, pp. 1194–1201, 2003.

[3] M. Lhuillier and L. Quan, “A Quasi-Dense Approach to Surface Reconstruction from Uncalibrated Images,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 418–433, 2005.

[4] H. Jin, S. Soatto, and A. Yezzi, “Multi-view Stereo Reconstruction of Dense Shape and Complex Appearance,” *International Journal of Computer Vision*, vol. 63, no. 3, pp. 175–189, 2005.

[5] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge University Press New York, NY, USA, 2003.

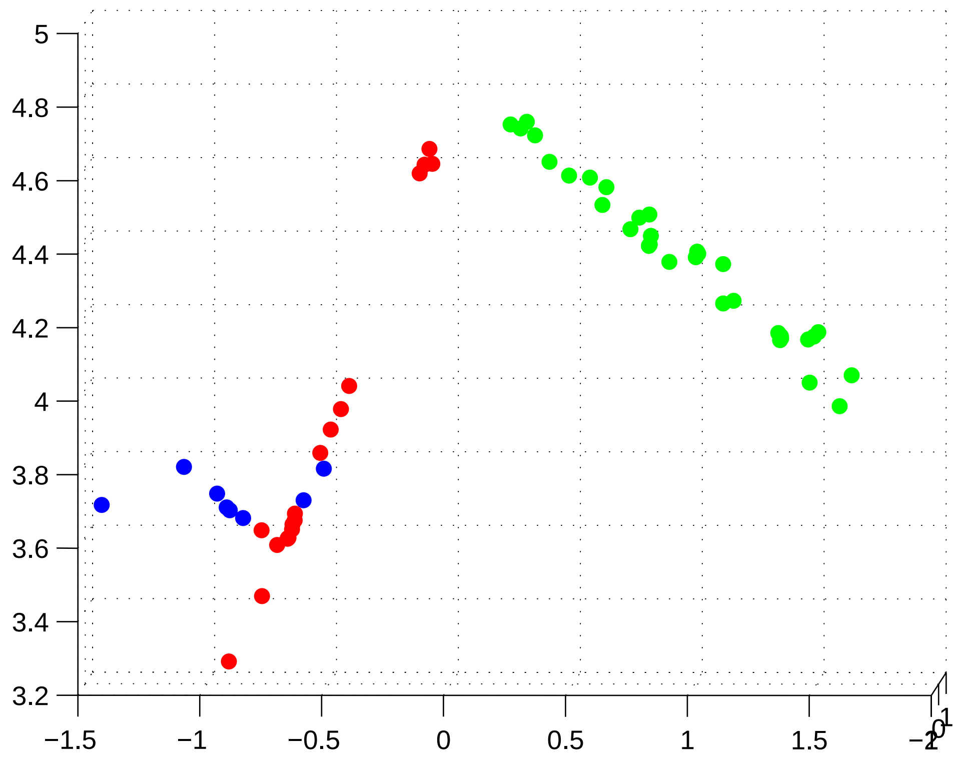


Fig. 6. Geometrical segmentation result of the EDA algorithm.

[6] E. Trucco and A. Verri, *Introductory techniques for 3-D computer vision*. Prentice Hall New Jersey, 1998.

[7] Y. Ma, S. Soatto, and J. Košecká, *An invitation to 3-d vision: from images to geometric models*. Springer Verlag, 2004.

[8] H. Li, B. Adams, L. Guibas, and M. Pauly, "Robust single-view geometry and motion reconstruction," in *ACM SIGGRAPH Asia 2009*, pp. 1–10, 2009.

[9] P. Beardsley, P. Torr, and A. Zisserman, "3D model acquisition from extended image sequences," in *Proceedings of the 4th European Conference on Computer Vision*, pp. 683–695, Springer-Verlag, 1996.

[10] A. Fitzgibbon and A. Zisserman, "Automatic camera recovery for closed or open image sequences," in *Proceedings of the 5th European Conference on Computer Vision-Volume I-Volume I*, pp. 311–326, Springer-Verlag, 1998.

[11] M. Pollefeys, R. Koch, and V. Gool, "Self-calibration and Metric Reconstruction in Spite of Varying and Unknown Internal Camera Parameters," *Journal of Computer Vision*, 1998.

[12] F. Devernay and O. Faugeras, "Automatic Calibration and Removal of Distortion from Scenes of Structured Environments," *Investigative and Trial Image Processing*, vol. 2567, pp. 62–72, 1995.

[13] N. Cornelis, B. Leibe, K. Cornelis, and L. Van Gool, "3d urban scene modeling integrating recognition and reconstruction," *International Journal of Computer Vision*, vol. 78, no. 2, pp. 121–141, 2008.

[14] M. Pollefeys, D. Nistér, J. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. Kim, P. Merrell, *et al.*, "Detailed real-time urban 3d reconstruction from video," *International Journal of Computer Vision*, vol. 78, no. 2, pp. 143–167, 2008.

[15] J. Yagnik and K. Ramakrishnan, "A model based factorization approach for dense 3D recovery from monocular video," in *Seventh IEEE International Symposium on Multimedia*, p. 4, 2005.

[16] V. Popescu, E. Sacks, and G. Bahmutov, "Interactive point-based modeling from dense color and sparse depth," in *Eurographics Symposium on Point-Based Graphics*, 2004.

[17] H. Chang, J. Moura, Y. Wu, K. Sato, and C. Ho, "Reconstruction of 3D dense cardiac motion from tagged MR sequences," in *IEEE International Symposium on Biomedical Imaging: Nano to Macro*, pp. 880–883, 2004.

[18] O. Faugeras and R. Keriven, "Complete dense stereovision using level set methods," in *Proceedings of the 5th European Conference on Computer Vision*, p. 393, 1998.

[19] R. Koch, M. Pollefeys, and L. Gool, "Multi viewpoint stereo from uncalibrated video sequences," in *Proceedings of the 5th European Conference on Computer Vision*, p. 71, 1998.

[20] M. Pollefeys, R. Koch, M. Vergauwen, and L. Van Gool, "Automated Reconstruction of 3D Scenes from Sequences of Images," *ISPRS Journal Of Photogrammetry And Remote Sensing*, vol. 55, no. 4, pp. 251–267, 2000.

[21] J. Kim and T. Sikora, "Gaussian scale-space dense disparity estimation with anisotropic disparity-field diffusion," in *In Proc. of IEEE 3DIM*, 2005.

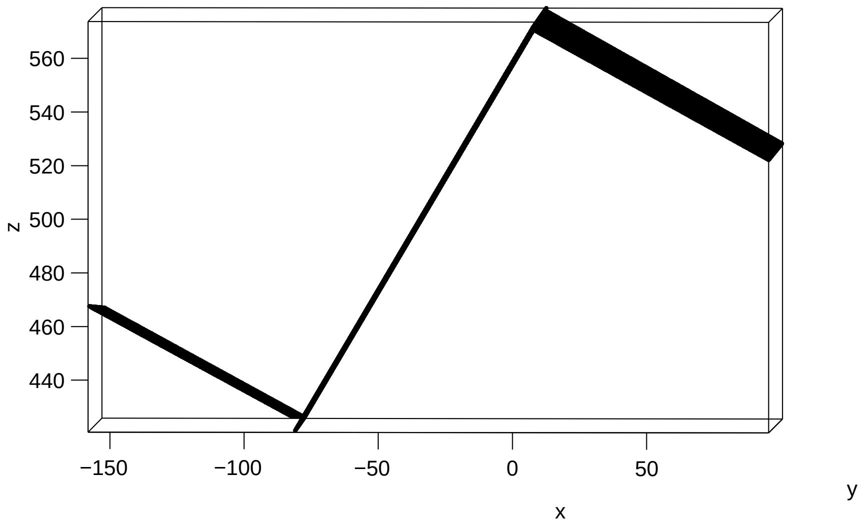


Fig. 7. Estimated geometric structures.

[22] C. Strecha and L. V. Gool, "Pde-based multi-view depth estimation," in *1 st International Symposium on 3D Data Processing Visualization and Transmission (3DPVT*, pp. 416–425, 2002.

[23] M. Lhuillier and L. Quan, "Surface reconstruction by integrating 3D and 2D data of multiple views," in *Proceedings of IEEE International Conference on Computer Vision 2003*, pp. 1313–1320, 2003.

[24] E. Arce and J. Marroquin, "High-precision stereo disparity estimation using HMMF models," *Image and Vision Computing*, vol. 25, no. 5, pp. 623–636, 2007.

[25] A. Fitzgibbon and A. Zisserman, "Multibody structure and motion: 3-D reconstruction of independently moving objects," *Computer Vision-ECCV 2000*, pp. 891–906, 2000.

[26] R. Vidal, Y. Ma, S. Soatto, and S. Sastry, "Two-view multibody structure from motion," *International Journal of Computer Vision*, vol. 68, no. 1, pp. 7–25, 2006.

[27] L. Wolf and A. Shashua, "Two-body segmentation from two perspective views," 2001.

[28] P. Torr, "Geometric motion segmentation and model selection," *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 356, no. 1740, p. 1321, 1998.

[29] G. Qian, R. Chellappa, and Q. Zheng, "Bayesian algorithms for simultaneous structure from motion estimation of multiple independently moving objects," *Image Processing, IEEE Transactions on*, vol. 14, no. 1, pp. 94–109, 2005.

[30] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *International Joint Conference on Artificial Intelligence*, vol. 3, p. 3, 1981.

[31] J. Barron, D. Fleet, and S. Beauchemin, "Performance of Optical Flow Techniques," *International Journal Of Computer Vision*, vol. 12, no. 1, pp. 43–77, 1994.

[32] J. Shi and C. Tomasi, "Good features to track," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593–600, 1994.

[33] M. Fischler and R. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[34] A. Likas, N. Vlassis, *et al.*, "The Global K-Means Clustering Algorithm," *Pattern Recognition*, vol. 36, no. 2, pp. 451–461, 2003.

[35] K. Rose, "Deterministic Annealing for Clustering, Compression, Classification, Regression, and Related Optimization Problems," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2210–2239, 1998.

[36] B. Kulis, S. Basu, I. Dhillon, and R. Mooney, "Semi-Supervised Graph Clustering: A Kernel Approach," *Machine Learning*, vol. 74, no. 1, pp. 1–22, 2009.

[37] C. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey vision conference*, vol. 15, p. 50, Manchester, UK, 1988.

[38] Z. Zhang, "Determining the epipolar geometry and its uncertainty: A review," *International Journal of Computer Vision*, vol. 27, no. 2, pp. 161–195, 1998.

[39] R. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses," *IEEE Journal of robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987.

[40] S. Kirkpatrick, "Optimization by Simulated Annealing: Quantitative Studies," *Journal of Statistical Physics*, vol. 34, no. 5, pp. 975–986, 1984.

[41] A. Rao, D. Miller, K. Rose, and A. Gersho, "A Deterministic Annealing Approach for Parsimonious Design of Piecewise Regression Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 2, pp. 159–173, 1999.

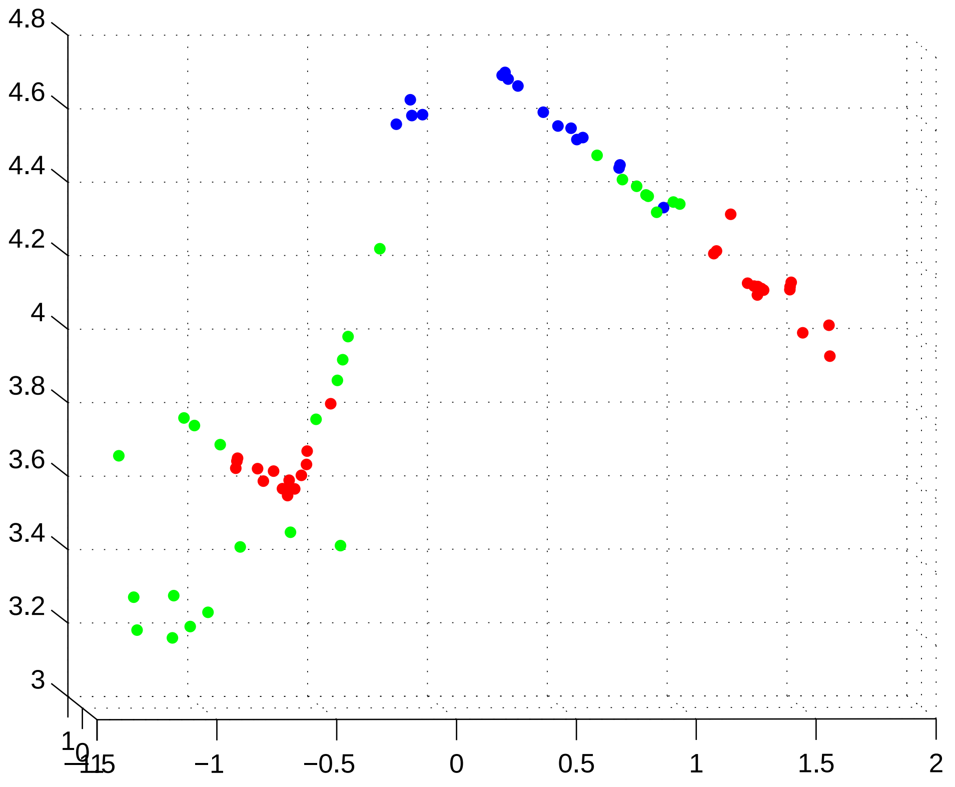


Fig. 8. Geometrical segmentation result of the PI algorithm.

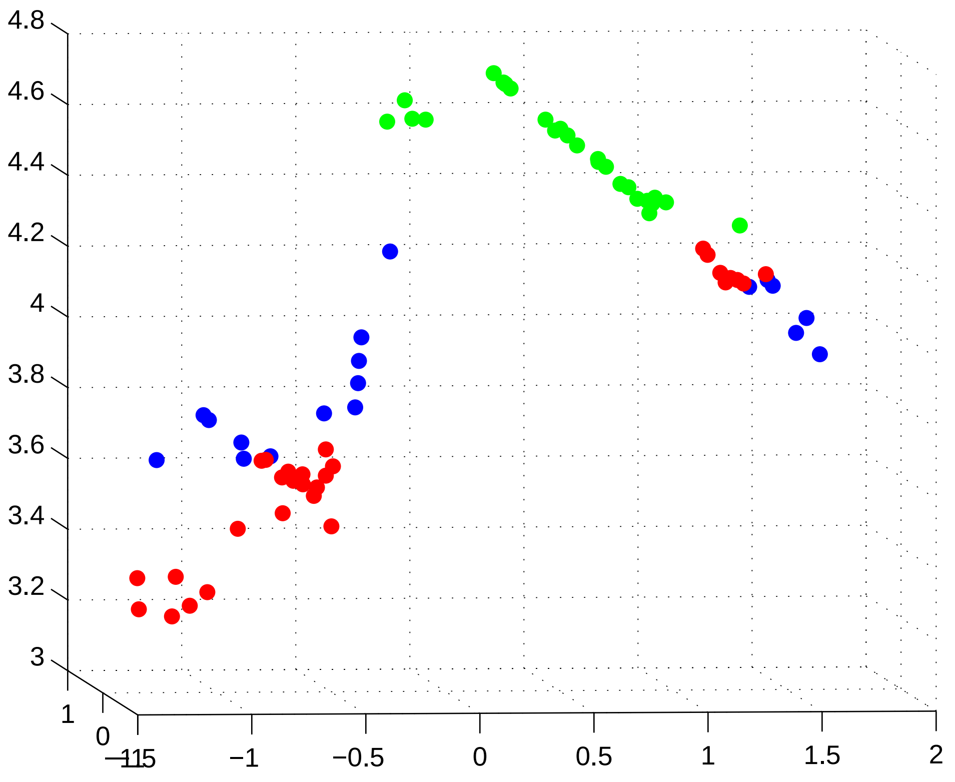


Fig. 9. Geometrical segmentation result of the API algorithm.