3D Surface Recovery via Deterministic Annealing based Piecewise Linear Surface Fitting Algorithm

Bing Han, Chris 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

The 3D surface fitting problem is to find a 3D surface that fits to a set of 3D points. Geometric fitting is commonly used in computer vision for 3D modeling and reconstruction. Finding a good fit to a given data set is a classical and challenging problem. Although there are many existing algorithms for specific cases, the geometric fitting problem is far from 'solved'. Geometric fitting is highly related to statistical regression, which is to approximate an unknown mathematical function that fits to input-output data pairs observed with random errors. In this paper, we present a new piecewise plane fitting method for 3D surface fitting. Different from traditional algorithms, we first segment the input space to several separate regions. With an assumption that each region is locally linear, we can use plane fitting to recover the 3D geometric surface. We propose a non-linear deterministic annealing algorithm for space partitioning. The non-linear deterministic annealing algorithm is able to avoid many shallow local optima. The algorithm also considers local structures to help data partitioning In the optimization process. The experimental results show that the new method can achieve better performance in both the average approximation error and correct identification rate on both synthetic data and real world data.

Index Terms

Deterministic Annealing, 3D surface recovery, surface fitting

I. Introduction

The geometric fitting problem is to find a geometrical surface that best fits to a set of 3D points. Geometric fitting is commonly used in 3D model fitting and 3D visual reconstruction in computer vision. The 3D surface fitting is highly related to statistical regression, which is an important tool in diverse areas.

Given a 3D point data set $\mathfrak{X} = \{\mathbf{x}_i\}, \mathbf{x}_i \in R^3, i = 1, 2, ..., n$, the geometrical fitting problem is usually stated as the optimization of a cost that measures how the geometrical surface function $\mathfrak{S} = \{\mathbf{x} : g_\theta(\mathbf{x}) = 0\}$ fits the data set $\mathfrak{X}$. The most commonly used objective function is the least squares cost,

$$D = \sum_{i=1,...N} d(\mathbf{x}_i, g_\theta)^2 \tag{1}$$

where

$$d(\mathbf{x}_i, g_\theta) = \min \|\mathbf{x}_i - \mathbf{x}_j\|^2, \quad \mathbf{x}_j \in \mathfrak{S} \tag{2}$$

The fitting function g_θ is learned by minimizing the design cost, D, measured over the input data set, $\mathfrak{X}$. It is well-known that for most choices of D, the cost measured during design monotonically decreases as the size of the learned fitting function g_θ is increased. With a large set of functions, it is easy to create a surface which passes through each input data point but is suspiciously complicated. 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 basis functions which yield a smoother, simpler surface which could well approximates the original data.

Generally, there are two approaches to solve the over fitting problem. One approach is to add penalty terms to the data set, like smoothness or regularization constraints. Another approach is to first build

a large model and then remove some parameters by retaining only the vital model structure. Although both approaches can generate parsimonious models, the descent based learning methods all suffer from a serious limitation. The non-global optima of the cost surface may easily result in poor local minima to the descent based learning methods. Techniques adding penalty terms to the cost function further increases the complexity of the cost surface and worsen the local minimum problem.

In this paper, we propose a different approach to solve the geometrical fitting problem. Instead of estimate a complicated function to fit all the data points, we partition the data set into several subset such that the data points in each subset could be approximated by a simpler model. The space partitioning helps to reduce the size of the surface model while keeping the design cost small enough.

One of the most popular clustering algorithm is Lloyd's algorithm, which starts by partitioning the input data into k initial sets. It calculates the centroid of each set via some metric. Usually, Lloyd's algorithm is used in a Euclidean space and centroid is calculated by averaging dimensions in Euclidean space. It iteratively associates each point with the closest centroid and recalculates the centroids of the new clusters. Alghouth widely used in real world applications, there are two serious limitations of Lloyd's algorithm. The first limitation is that the partitioning result depends on the initialization of the cluster centers, which may lead to poor local minima. The second limitation is that Lloyd's algorithm can only partition linear separable clusters.

In order to avoid initialization dependence, a simple but useful solution is to use multiple restarts with different initializations to achieve a better local minima. Global k-means [1] 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 [2] is another optimization technique to find a global minimum of a cost function. Deterministic annealing explore a larger cost surface by introducing a constraint of randomness. At each iteration, the randomness is constrained and a local optimization is performed. Finally, the imposed randomness is reduce to zero, and the algorithm optimizes over the original cost function.

Kernel method [3] is used to solve the second problem by mapping the data points from input space to a higher dimensional feature space through a non-linear transformation. Then the optimization is applied in the feature space. The linear separation in the feature space turns out to be a non-linear separation in the original input space.

In this paper, we propose a non-linear deterministic annealing approach for space partitioning in 3D Euclidean space. We use deterministic annealing to divide the input space into several regions with different sizes and shapes. With the partition, we can easily find a linear local surface to fit the data inside each region. Deterministic annealing method offers two great features: 1) the ability to avoid many poor local optima; 2) the ability to minimize the cost function even its gradients vanish almost everywhere. Due to the fact that the data is localized to a few relatively dense clusters, we design a kernel function to map the data point from the geometric space to surface feature space and apply deterministic annealing in the feature space instead of the geometric space. We compare the proposed non-linear deterministic annealing (NDA) algorithm with the widely used Lloyd's algorithm on both artificial data and real world data. The experimental results show that NDA algorithm outperforms Lloyd's algorithm in both mean squared approximation error and error probability.

In the following section we formally define the 3D geometric fitting problem and briefly describe deterministic annealing and kernel method for space partitioning. In Section III we present the proposed kernel deterministic annealing algorithm along with an analysis of its computational complexity. The experimental result is shown in Section IV. Finally Section V concludes this paper.

II. The 3D Geometric Fitting Problem

The following terms and notations are used throughout this report.

- Input samples $\mathbf{x}_1, ..., \mathbf{x}_N \in \mathcal{R}^3$ are 3D data points.
- A feature vector $\mathbf{f}_i = \Phi(\mathbf{x}_i) \in \mathcal{F}$ is computed by some mapping $\Phi : \mathcal{R}^3 \rightarrow \mathcal{F}$. It typically consists of a vector of d measurements: $\mathbf{f} = (f_1, ..., f_d)$.

- d is the dimensionality of the pattern or of the pattern space.
- A data set is denoted $\mathfrak{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$.

Given a set of data $\mathfrak{X}$ of scattered 3D points, we would like to find the geometric surface that best fits to the scattered data. The fitting problem is usually stated as the optimization of a cost that measures how well the fitting function $g(\mathbf{x}_i)$ fits the data. The most commonly used objective function is the least squares cost. Finding a good fit is a challenging problem and may be more of an art than a science. If we use a large set of functions as the basis, we may create a surface which passes through each data point but is suspiciously complicated. Using few basis functions may yield a smoother, simpler surface which only approximates the original data. Due to the over fitting problem, we propose an new approach to optimize the objective function via space partitioning. We first partition the data set into several subsets such that the data points $\mathbf{x}$ in each subset could be approximated by a linear surface model. In other words, we would like to use a set of plain models to approximate the date set. The objective of space partitioning is to minimize the geometric fitting error.

$$\min_{g_{\theta_k}} D = \sum_{k=1}^{K} \sum_{i \in C_k} d(\mathbf{x}_i, g_{\theta_k}) \tag{3}$$

where, $\mathbf{x}_i = [x_i, y_i, z_i]^T$ is the i-th point data, $\theta_k = [a_k, b_k, c_k]^T$ is the k-th linear surface model, and $d_{i,k}$ is is the fitting error between $\mathbf{x}_i$ and plane model $g_{\theta_k} = 0$ which is defined as

$$d_{i,k} = d(\mathbf{x}_i, g_{\theta_k}) = \frac{(\mathbf{x}_i^T g_{\theta_k} - 1)^2}{a_k^2 + b_k^2 + c_k^2} \tag{4}$$

A. Deterministic Annealing

The deterministic annealing (DA) approach [2] to clustering has demonstrated substantial performance improvement over traditional supervised and unsupervised learning algorithms. DA mimics the annealing process in static The advantage of deterministic annealing is its ability to avoid many poor local optima. The reason is that deterministic annealing minimizes the designed cost function subject to a constraint on the randomness of the solution. The constraint, Shannon entropy, is gradually lowered and eventually deterministic annealing optimize on the original cost function. Deterministic annealing mimics the simulated annealing [4] in statistical physics by the use of expectation. Deterministic annealing derives an effective energy function through expectation and is deterministically optimized at successively reduced temperatures. The deterministic annealing approach has been adopted in a variety of research fields, such as graph-theoretic optimization and computer vision. A. Rao et al. [5] extended the work for piecewise regression modeling. In this subsection, we will briefly review their work.

Given a data set $(\mathbf{x}, \mathbf{y})$, the regression problem is to optimize the cost that measures how well the regression function $f(\mathbf{x})$ approximates the output $\mathbf{y}$, where $\mathbf{x} \in \mathcal{R}^m$, $\mathbf{y} \in \mathcal{R}^n$, and $g : \mathcal{R}^m \rightarrow \mathcal{R}^n$. In the basic space partitioning approach, the input space is partitioned into K regions and the cost function becomes

$$\min_{\mathbf{\Lambda}_k} D = \sum_{k=1}^{K} \sum_{i \in C_k} d(\mathbf{y}_i, f(\mathbf{x}_i, \mathbf{\Lambda}_k)) \tag{5}$$

where $d(\cdot, \cdot)$ is the distortion measure function. Instead of seeking the optimal hard partition directly, randomness is introduced for randomized assignment for input samples.

$$D = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} P(\mathbf{x}_i \in C_j) d(\mathbf{y}_i, f(\mathbf{x}_i, \mathbf{\Lambda}_k)) \tag{6}$$

In A. Rao et al.'s work, they use the nearest prototype (NP) structure as constraint and given the set of prototypes$\{\mathbf{s}_j : j = 1, 2, 3, ..., K\}$ in the input space, a Voronoi criterion is defined for NP partition

$$C = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} P(\mathbf{x}_i \in C_j)||\mathbf{x}_i - \mathbf{s}_j||. \tag{7}$$

Although the ultimate goal is to find the hard partition, some "randomness" is desired during the assignment. Shannon entropy is introduced as a constraint of the randomness.

$$H = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} P(\mathbf{x}_i \in C_j)\log P(\mathbf{x}_i \in C_j). \tag{8}$$

Eventually, this constrained optimization problem could be rewritten as the minimization of the corresponding Lagrangian

$$\min_{\{\mathbf{\Lambda_j}\}\{\mathbf{s_j}\},\gamma} F = D - TH \tag{9}$$

where, γ is a nonnegative Lagrange multiplier which controls the randomness of the space partition.

B. Non-linear Partitioning

Kernel methods (KMs) are a class of algorithms for pattern analysis whose general task is to find and study types of relations of input data. KMs perform a nonlinear mapping of the input data to a higher dimensional feature space. Then a variety of methods can be applied for pattern analysis in the feature space. The advantage of KMs is that KMs do not need to compute the coordinates of the data in the feature space explicitly but only compute the innor products between all pairs of data in the feature space by using kernel functions.

Take the most popular k-means algorithm [6] as an example, kernel k-means maps data points from the input space to a higher dimensional feature space through a nonlinear transformation ϕ and then apply standard k-means in the feature space. The clustering result in linear separators in feature space corresponds to nonlinear separators in input space. Thus kernel k-means avoid the limitation of standard k-means that the clusters must be linearly separable.

III. Non-linear Deterministic Annealing

In this paper, we propose a new approach based on non-linear deterministic annealing to solve the 3D geometric fitting problem. We first use a non-linear function to map the input point data to a high dimensional feature space using the local geometric structure of the data. Then we apply deterministic annealing in the feature space to leverage the local geometric structure for clustering.

To solve the space partitioning problem, we do not use prototype to calculate the difference. The reason is that the prototype in space partitioning is generally not sufficient to represent a plane in 3D space. Instead, we estimate the linear plane model and calculate the fitting error as the Euclidean distance between the data and the plane. The traditional local optimization algorithm will likely stuck at a local optima. In order to avoid local optima, we use local geometric structure from neighboring data points and embedded the data vectors to a higher dimension as follows.

The input data is given as a 3D point, $\mathbf{x}_i = [x_i, y_i, z_i]^T$. With the assumption that nearest data points are on the same plane, we could estimate the local plane model, $\mathbf{L}_i = [a_i, b_i, c_i]^T$ of data point $\mathbf{x}_i$ and its K nearest neighbor points.

$$\mathbf{L} = \begin{bmatrix} a(\mathfrak{X}) \\ b(\mathfrak{X}) \\ c(\mathfrak{X}) \end{bmatrix} \tag{10}$$

$$\mathbf{f} = \begin{bmatrix} \mathbf{x} \\ \mathbf{L} \end{bmatrix} \tag{11}$$

Then we revise the distortion function as follows,

$$D(\mathbf{f}_i, g_{\theta_j}) = D_1(I_1\mathbf{f}_i, g_{\theta_j}) + D_2(I_2\mathbf{f}_i, g_{\theta_j}) \tag{12}$$

$$I_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{13}$$

$$I_2 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{14}$$

where $D_1 = d_{i,j}$ calculate the fitting error between the data point and the estimated plane, and D_2 calculate the difference between the local estimated plane model and the cluster scale estimated plane model. D_2 is defined as follows:

$$D_2(I_2\mathbf{f}_i, g_{\theta_j}) = \frac{I_2\mathbf{f}_i^T \times g_{\theta_j}}{|I_2\mathbf{f}_i| \times |g_{\theta_j}|} \tag{15}$$

After the mapping, we apply deterministic annealing algorithm to partition the data into several clusters as follows.

$$\min_{g_{\theta_j}} F = D - TH \tag{16}$$

where $g_{\theta_j} = [a_j, b_j, c_j]$ is the geometrical surface model parameter to be estimated, D is the sum of square of geometrical fitting error and H is the entropy constraint. We define D and H as follows:

$$D = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} p(\mathbf{x}_i, g_{\theta_j}) d(\mathbf{x}_i, g_{\theta_j}) = \sum_{i=1}^{N} p(\mathbf{x}_i) \sum_{j=1}^{K} p(g_{\theta_j}|\mathbf{x}_i) d(\mathbf{x}_i, g_{\theta_j}) \tag{17}$$

$$H(\mathbf{X}, g_\theta) = \sum_{i=1}^{N}\sum_{j=1}^{K} p(\mathbf{x}_i, g_{\theta_j}) \log p(\mathbf{x}_i, g_{\theta_j}) \tag{18}$$

To perform optimization we need to further analyze its terms. We can rewrite equation (18) by applying the chain rule of entropy as

$$H(\mathbf{X}, g_\theta) = H(\mathbf{X}) + H(g_\theta|\mathbf{X}) \tag{19}$$

Notice that the first term $H(\mathbf{X})$ is the entropy of the source and is therefore constant with respect to the cluster g_{θ_j} and association probabilities $p(g_{\theta_j}|\mathbf{x}_i)$. Thus we can just focus on the conditional entropy

$$H(g_\theta|\mathbf{X}) = \sum_{i=1}^{N} p(\mathbf{x}_i) \sum_{j=1}^{K} p(g_{\theta_j}|\mathbf{x}_i) \log p(g_{\theta_j}|\mathbf{x}_i) \tag{20}$$

The minimization of F with respect to association probabilities $p(g_{\theta_j}|\mathbf{x}_i)$ gives rise to the Gibbs distribution

$$p(g_{\theta_j}|\mathbf{x}_i) = \frac{\exp(-\frac{d(\mathbf{x}_i, g_{\theta_j})}{T})}{Z_x} \tag{21}$$

where the normalization is

$$Z_x = \sum_{j=1}^{K} \exp(-\frac{d(\mathbf{x}_i, g_{\theta_j})}{T}) \tag{22}$$

1) Algorithm 1 **NDA based geometrical segmentation algorithm**
2) **Set Limit**
3) K_{max}: maximum number of clusters
4) T_{init}: starting temperature
5) T_{min}: minimum temperature
6) δ: perturbation vector
7) α: cooling rate (must be < 1)
8) I_{max}: maximum iteration number
9) th: Iteration threshold
10) sth: Surface distance threshold
11) **Initialization**
12) $T = T_{init}, K = 2, \Lambda_1 = (X^T X)^{-1} X^T \vec{1}, \Lambda_2 = \Lambda_1, [p(\mathbf{\Lambda}_1|\mathbf{x}_i), p(\mathbf{\Lambda}_2|\mathbf{x}_i)] = [\frac{1}{2}, \frac{1}{2}], \forall i.$
13) **Perturb**
14) $\Lambda_j = \Lambda_j + \delta, \forall j.$
15) $L_{old} = D - TH.$
16) **Loop until convergence,** $i = 0\ \forall j$
17) For all $\mathbf{x}_i$ in the training data, compute the association probabilities

$$p(\mathbf{\Lambda}_j|\mathbf{x}_i) = \frac{\exp(-\frac{d(\mathbf{x}_i, \mathbf{\Lambda}_j)}{T})}{\sum_{j=1}^{K} \exp(-\frac{d(\mathbf{x}_i, \mathbf{\Lambda}_j)}{T})} \tag{25}$$

18) update the surface model

$$\mathbf{\Lambda}_j \longleftarrow \mathbf{\Lambda}_j + \alpha \nabla_{\mathbf{\Lambda}_j} F. \tag{26}$$

19) i = i+1;
20) if ($i > I_{max}$ or $\nabla_{\mathbf{\Lambda}_j} F < th$) End **Loop**
21) **Model Size Determination**
22) if($d(\mathbf{\Lambda}_j, \mathbf{\Lambda}_{j+1}) < sth$)
23) replace $\mathbf{\Lambda}_j, \mathbf{\Lambda}_{j+1}$ by a single plane
24) K =number of planes after merging
25) **Cooling Step**
26) $T = \alpha T.$
27) if ($T < T_{min}$)
28) perform last iteration for $T = 0$ and STOP
29) **Duplication**
30) Replace each plane by two planes at the same location, $K = 2K.$
31) **Goto Step 10**

Fig. 1. NDA based geometrical segmentation algorithm

The corresponding minimum of F is obtained by plugging equation (21) back into equation (16)

$$F^* = \min_{p(g_{\theta_j}|\mathbf{x}_i)} F = -T \sum_{i=1}^{N} p(\mathbf{x}_i) \log Z_x \tag{23}$$

To minimize the Lagrangian with respect to the cluster model g_{θ_j}, its gradients are set to zero yielding the condition

$$\nabla_{g_{\theta_j}} F = \frac{1}{N} \sum_{i=1}^{N} p(g_{\theta_j}|\mathbf{x}_i) \nabla_{g_{\theta_j}} d(\mathbf{x}_i, g_{\theta_j}) = 0 \tag{24}$$

Non-linear deterministic annealing method (NDA) introduces the entropy constraint to explore a large portion of the cost surface using randomness, while still performing optimization using local information, which is similar to fuzzy c-means algorithm. Eventually, the amount of imposed randomness is lowered so that upon termination NDA optimizes over the original cost function and yields a solution to the original problem.

However, there is no close form solution for NDA, therefore we use a gradient descent algorithm to solve this problem. In this paper, We compare NDA based geometrical segmentation algorithm to the projection based iterative algorithm (PI) and adaptive projection based iterative algorithm (API). I present our algorithm in Figure. 1. For comparison purpose, I also give PI algorithm in Figure. 2.

1) Algorithm 2 **Projection based iterative algorithm for geometrical segmentation**
2) **Set Limit**
3) K_{max}: maximum number of clusters
4) I_{max}: maximum iteration number
5) th: Iteration threshold
6) **Initialization**
7) Start with a random cluster assignment to all input vectors and estimate the linear plane model for each cluster by minimizing the total least squares.
8) **Loop until convergence,** $i = 0\ \forall j$
9) a. Assign each input vector $\mathbf{x}$ to the each cluster with the smallest geometrical fitting error.
10) b. Estimate the linear plane model for each cluster by minimizing the total least squares.

Fig. 2. Projection based iterative algorithm for geometrical segmentation

TABLE I

THE AVERAGE SQUARED APPROXIMATION ERROR.

K	PI	API	NDA
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}

IV. EXPERIMENTAL RESULTS

In this paper, I compared three geometric segmentation algorithms, PI algorithm, API algorithm, and NDA based geometric segmentation algorithm, based on both synthetic data and real world data.

A. NDA on Synthetic Data without Noise

The purpose of the first experiment is to compare NDA, PI, and API on synthetic data without noise. I generated the synthetic data using MATLAB 'randperm' function. The data is a set of 3D points on several linear planes without noise. In this experiment, I run each algorithm for 1000 times. Each time, a random data set is generated and used. We segment the same data set with different algorithms and calculate the average squared approximation error. Below is the experimental result in Table. I. K represents the number of planes in a test data set. For each plane, 100 random points are generated. The date set 1 contains 300 data in total from 3 non parallel planes. The data set 2 contains 400 data from 4 planes. The data set 3 contains 500 data from 5 planes and the data set 4 contains 600 data from 6 planes. The average squared approximation error of NDA is ignorable comparing to the errors of PI and NPI. From the experimental result, we can say that NDA algorithm outperforms both PI and API algorithms in the average squared approximation error. The reason NDA algorithm outperforms PI and API algorithms is that NDA is able to separate the space non-linearly and avoid many poor local optima.

We also measure the performance of the segmentation algorithms in percentage of correct identification of planes. We test the same data set as used in the previous experiment and compute the correct identification percentage averaging over all tests. Below is the experimental result in Table. II. We observed that correct identification rates of NDA and API are much higher than the correct identification rate of PI algorithm. The reason API algorithm outperforms PI algorithm is that API algorithm does not depends on random initialization while the segmentation results of PI algorithm heavily depends on initialization. Still NDA performs best among the three algorithms in correct identification rate.

B. NDA on Synthetic Data with Noise

The purpose of the second experiment is to compare NDA, PI, and API algorithms on synthetic data with noise. I generated the synthetic data with Gaussian noises in the same way as in the first experiment. In this experiment, I also run each algorithm for 1000 times. Each time, a random data set is generated

TABLE II

THE CORRECT IDENTIFICATION RATE.

K	PI	API	NDA
3	83%	96%	99%
4	79%	93%	99%
5	82%	94%	97%
6	78%	97%	98%

TABLE III

THE AVERAGE SQUARED APPROXIMATION ERROR.

K	PI	API	NDA
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}

and used. We segment the same data set with different algorithms and calculate the average squared approximation error. The experimental result is shown in Table. III. K represents the number of planes in a test data set. It shows that NDA algorithm outperforms both PI and API algorithm. The average squared approximation fitting error of NDA algorithm is less than 50% compare to the fitting error of PI algorithm. However, the performance gain is less compared to the first experiment. The reason is that the non-linear mapping in NDA depends on the estimation of the local geometric structures. While the estimation of the local geometric structures is very sensitive to the added noises. Even though the performance gain is less, we can still say that the NDA algorithm outperforms both PI and API algorithms in the average squared approximation error from the experimental result. We also show the experimental result in 3D view in Fig. 3 and Fig. 4. Fig. 3 shows the segmentation results of test data set 1 with three planes by the NDA algorithm. Fig. 4 shows the segmentation results of the same test data set by the PI algorithm.

C. NDA on Real World Data

In the second experiment, we test the geometric segmentation algorithm on some real world data. We use the 3D structure data set from the 'housing' image sequence. The data set includes 72 data points recovered by 3D reconstruction of 2D registered feature points. Most of the data points fall on the walls of the house in the image and we would like to estimate the surface model of the walls by geometric fitting. Fig. 5 shows the input 3D data points on the 1st frame of the 'housing' image sequence. The goal is to segment the data points into three groups and each group represent a wall in the image. Fig. 6 shows the geometric segmentation result by NDA algorithm and Fig. 7 shows the geometrical segmentation result by PI algorithm. It is pretty clear that NDA algorithm partitions the input data set into three clusters and each cluster represents a wall in the image. PI algorithm fails to find the geometric model of the walls and the data points are mixed. The experimental result on real world data shows that NDA algorithm can well segment the data sets based on their geometric relationship.

V. CONCLUSION

In this paper, we propose a kernel deterministic annealing approach for geometric fitting in 3D space. Due to the fact that the 3D data is localized to a few relatively dense clusters, we design a kernel function to map the data point from geometrical space to surface model space and apply deterministic annealing in the feature space. We then use deterministic annealing to partition the feature space into several regions with different sizes and shapes. For each region, we can easily find a linear plane model to fit the data. Deterministic annealing method offers two important features: 1) the ability to avoid many poor local

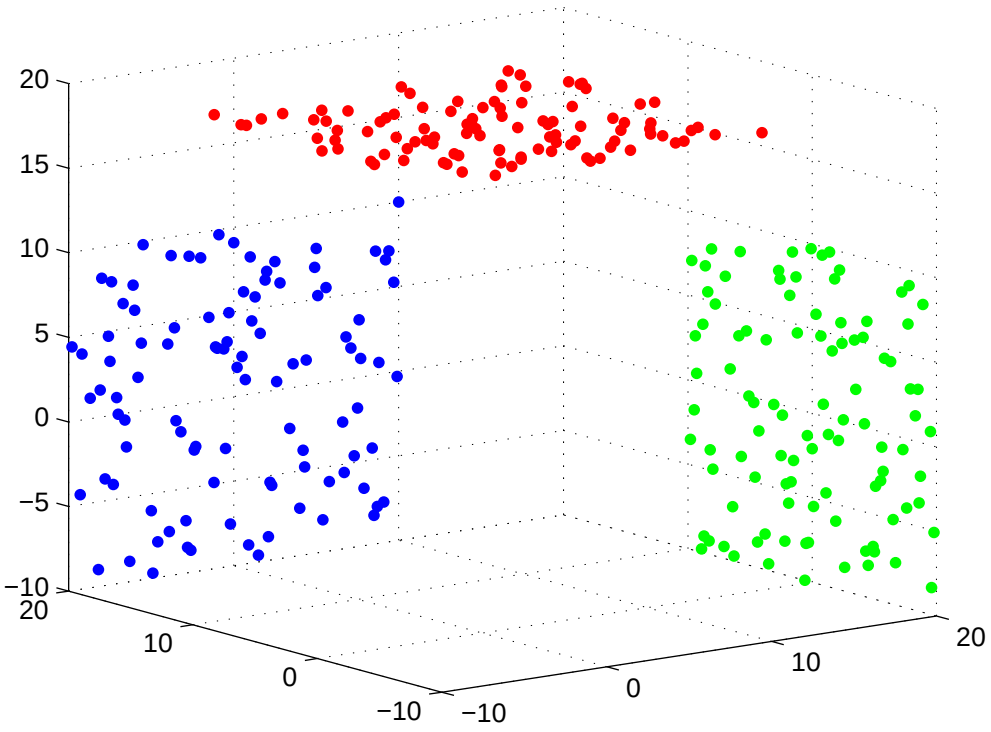

Fig. 3. The synthetic data set.

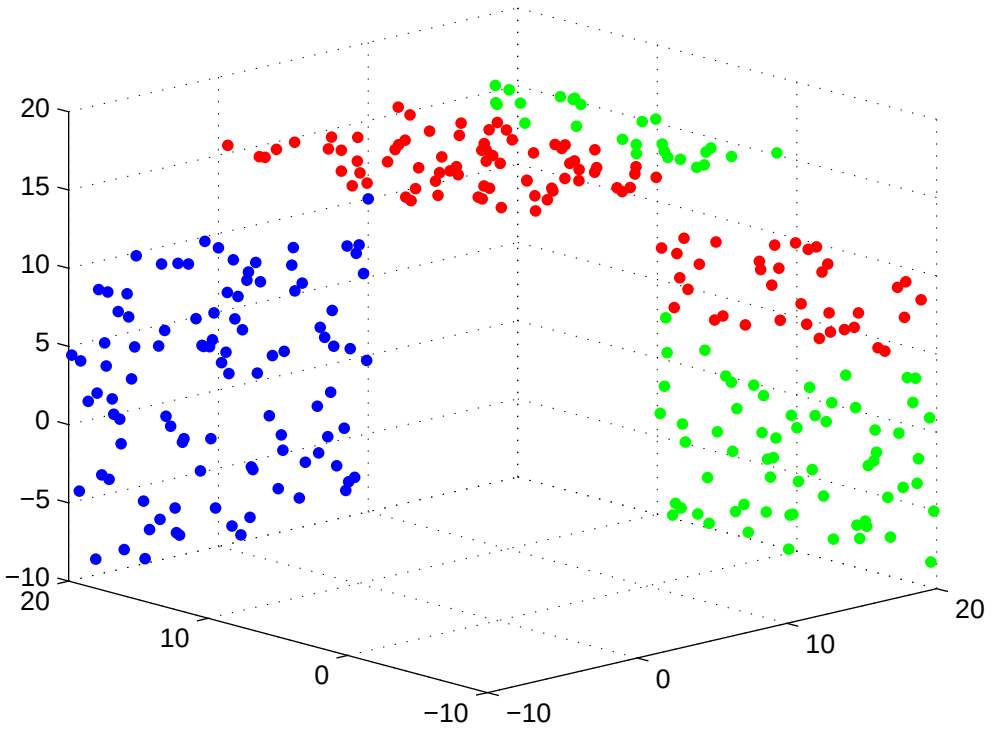

Fig. 4. The first group partitioned by K-means.

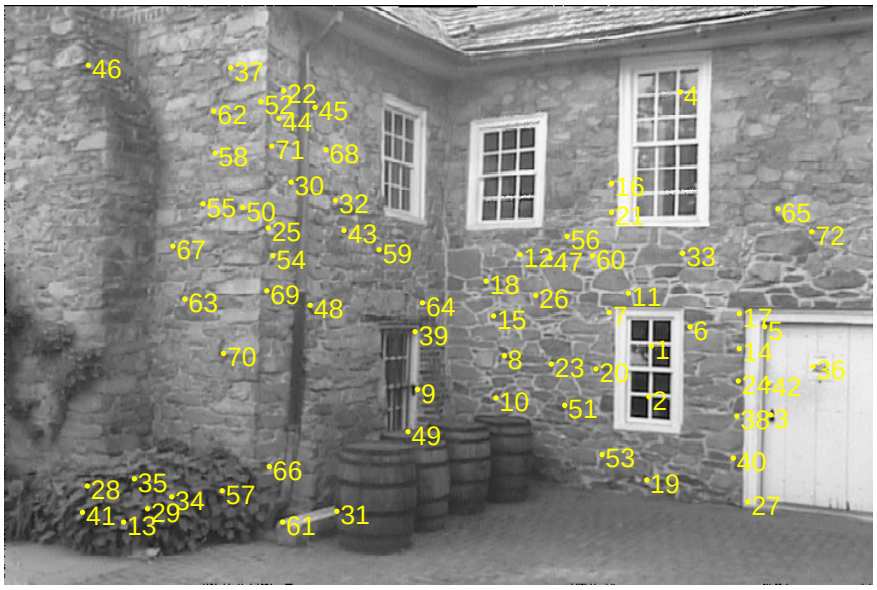

Fig. 5. The input data points on the 1st frame of 'housing' image sequence.

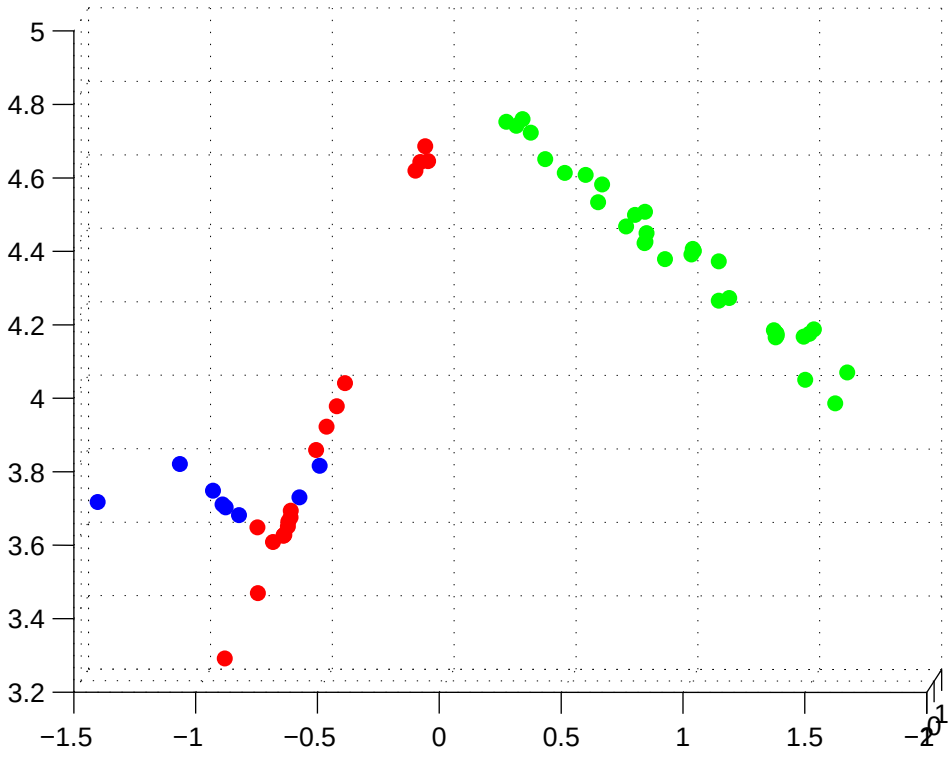

Fig. 6. The geometrical segmentation result by NDA of 'housing' data set.

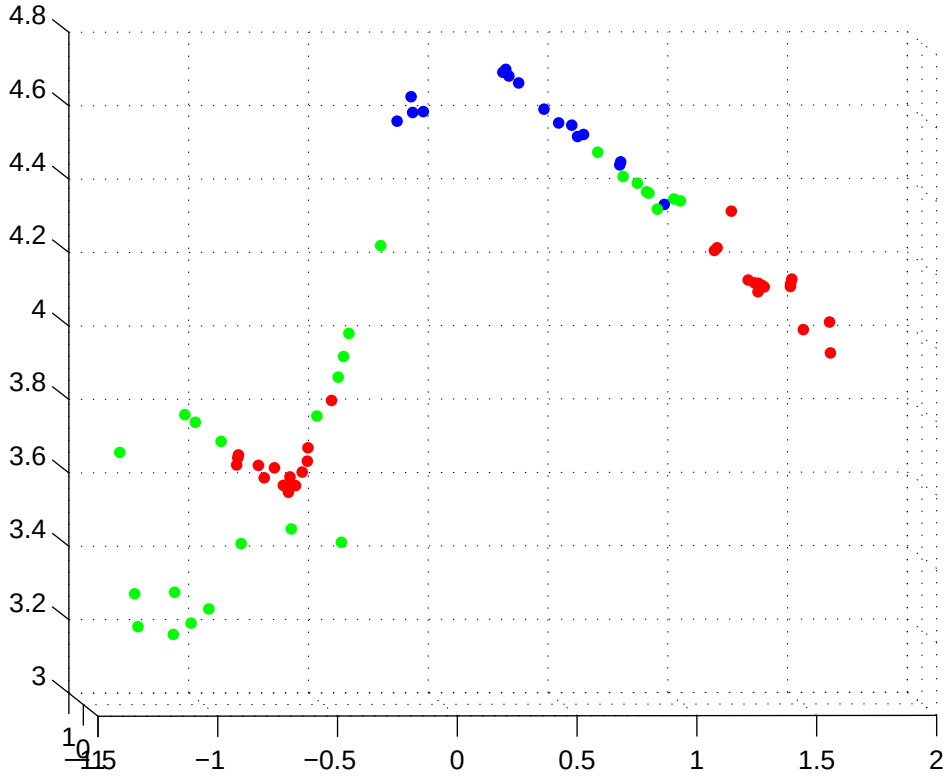

Fig. 7. The geometrical segmentation result by the PI algorithm of 'housing' data set.

optima; 2) the ability to minimize the cost function even its gradients vanish almost everywhere. We present experimental results that compare kernel deterministic annealing and classic Lloyd's algorithm and ISODATA on both artificial data and real data. From the experimental results, it shows that KDA algorithm outperforms both ISODATA and Lloyd's algorithm in both cases. In the future, we would like to study how to use KDA algorithm in image registration, computer vision, and 3D reconstruction.

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.

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] A. Likas, N. Vlassis, *et al.*, "The global k-means clustering algorithm," *Pattern Recognition*, vol. 36, no. 2, pp. 451–461, 2003.
[2] 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.
[3] 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.
[4] S. Kirkpatrick, "Optimization by simulated annealing: Quantitative studies," *Journal of Statistical Physics*, vol. 34, no. 5, pp. 975–986, 1984.

[5] 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.
[6] F. Camastra and A. Verri, "A Novel Kernel Method for Clustering," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, pp. 801–804, 2005.