3D MODELING OF POINT CLOUDS

By

THOMAS PATTEN

Bachelor of Science in Mechanical Engineering
Oklahoma State University
Stillwater, OK, United States of America
2002

Master of Science in Mechanical Engineering
Oklahoma State University
Stillwater, OK, United States of America
2004

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2009

3D MODELING OF POINT CLOUDS

Thesis Approved:

Dr. Guoliang Fan

Thesis Advisor

Dr. Martin Hagan

Dr. Weihua Sheng

Dr. Gordon Emslie

Dean of the Graduate College

ACKNOWLEDGMENTS

I would like to thank....Dr. Fan and everyone the Visual Computing and Image Processing Lab (VCIPL), Oklahoma State University, Oklahoma Alliance for Minority Participation (OKAMP), David Monismith, my father Noel Patten and my wife Penny Patten.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Recent advances in computer speed and laser measuring equipment have made it possible to create devices that can accurately sample the surface of real world objects. Graphics processing power has greatly increased and is able to handle very large amounts of data. This has made way for the advancement of systems like LiDAR (Light Detection And Ranging) which use lasers to discretely sample the surface of objects ranging in size from small cups to skyscrapers and complete cities. While these systems are mostly used for reverse engineering, Figure 1.1, the potential has been created to model large existing man-made structures like buildings (both inside and outside) roadways and bridges and even land used for water runoff and waste collection [4, 1].

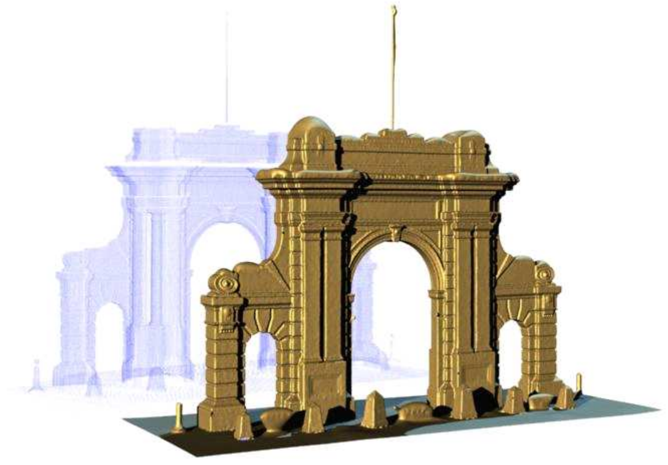


Figure 1.1: This is a surface reconstruction of a gate using LiDAR data. The point cloud is behind the reconstructed surface of the gate. This image is from www.farfieldtechnology.com.

An accurate three dimensional model of a man-made object could be used to monitor the condition of the object or to engineering additional objects to interact with the existing object. For instance, buildings could be modeled to aid in precisely engineering construction of new buildings that are attached to the original. A bridge could be modeled to allow engineers to know how to accurately add more lanes or ensure a new road underneath the bridge has the clearance it needs. Refineries could be modeled to know the exact destination of pipes that exit or enter different building. Building interiors could be modeled to give fire fighters a map when smoke makes it hard to find people or exits.

Although it is possible to very accurately sample objects with systems like LiDAR, these systems only discretely digitize objects. They create many points, called a point cloud, that represent the surface of an object. Nothing is known about how the points should be connected or how smooth the surface is around each point. Most man-made objects consist of a large portion of basic geometry that we will show can be easily and accurately segmented from more complex areas of point clouds. This is advantageous since the reduction in data is potentially immense.

Consider the interior sections of many buildings. Most contain simple planes that should be easily identified and segmented. A densely sampled planar surface could contain hundreds, thousands, or even millions of points. If the planar surface is a simple rectangular wall, it could be accurately described using only four points. A planar surface with a complex boundary could also be represented in a similar manner more efficiently than a point cloud. Cylinders and cones can also be represented more efficiently using reference points and equations instead of a point cloud. Most point clouds contain more complex surfaces than planes, cylinders, and cones but may also contain these simple shapes. Complex surface modeling would benefit from the reduction of points where possible.

We will show that three basic shapes, planes, cylinders and cones can easily and quickly be modeled in a point cloud using Expectation Maximization (EM) clustering combined with Minimum Description Length (MDL) to estimate model order. We will also show the three shapes can be modeled using Markov Chain Monte Carlo (MCMC) using Reversible Jump (RJ-MCMC) to estimate model order. We will show that the MCMC algorithm is much more robust than EM and does not have the initialization issues that the EM algorithm does.

1.1 Organization of Thesis

This thesis is organized in the order that the research was completed. First we will talk about the related work that has been done in this field and the research goals we want to accomplish. We will then talk about extracting the features from the point cloud for the clustering algorithms. Our method is novel since we cluster using only the normal vectors of the point cloud. We assume an unorganized point cloud. An unorganized point cloud is simply a list of three dimensional points with no connectivity or normal vector information. Since we use the normal vectors for clustering, they need to be estimated. This estimation process is discussed in detail in Chapter 2.

Our first stages of research focused on clustering planes only using the EM algorithm which is discussed in Chapter 3. After planar clustering with EM, we decided to extend the research to cylinders and cones. Cylinders and cones lead to an atypical cluster grouping that is unique when compared to planes. Chapter 4 discusses the new likelihood function we derived for the unique cluster grouping of cylinders and cones. Chapter 5 explains the EM clustering modified with our new likelihood function for clustering all three shapes simultaneously. Chapter 6 shows how the likelihood function can also be used with a robust MCMC algorithm and the conclusions are discussed in Chapter 7.

1.2 Related Work

Processing data from point clouds is a very diverse area of research since there are many applications ranging in size from small to very large. In the past, research mostly focused on noisy, poorly sampled point clouds. The goals were to extract smooth surfaces by minimizing the error or energy of the surface. This usually resulted in a tradeoff between more complex, more accurate surfaces and less complex, less accurate surfaces. Advances in computer power and memory sizes have allowed for changes in the point cloud research. Laser range finders are faster and can create very large point clouds that accurately describe surfaces with little noise. Much research is now done to reduce the large amount of data wherever possible. Our goal in this thesis is to reduce the volume of data where possible while preserving the true geometry of the object. There are many areas of research that are very active: Feature extraction, feature enhancing and sharpening, multiple sensor registration, point cloud meshing, point cloud segmentation, and 3D model inference.

1.2.1 Feature Extraction

Feature extraction [5, 6] became a popular area of research as the laser scanning technology advanced. The goal of such papers was to locate areas of the point cloud that are not smooth surfaces. An example would be where two surfaces come together to form and edge. These identifying features can be used as boundaries of smooth areas of the point cloud which would be helpful to surface fitting algorithms. These papers are very important even though we are not clustering sharp features. Feature Extraction [1] was able to locate sharp features in point clouds using a neighbor graph as input to the algorithm. The neighbor graph is similar to a mesh but locates the neighboring points without inferring the connectivity. This is less computationally expensive than using a mesh. Features to be inferred include crease loops, border loops and crease lines that appear on junctions between two different surfaces and can be seen in Figure 1.2. Although this process is more computationally expensive than the algorithm we are proposing, the idea allowed us to create an efficient algorithm to estimate normal vectors of point cloud points.

1.2.2 Feature Enhancing and Sharpening

Another area of research is in sharpening features [7, 8, 9] of the point cloud. A point cloud cannot sample the edge of an object very efficiently. An edge can be recovered and enhanced using assumptions about the surfaces that make up the edges. A mesh [2] is also

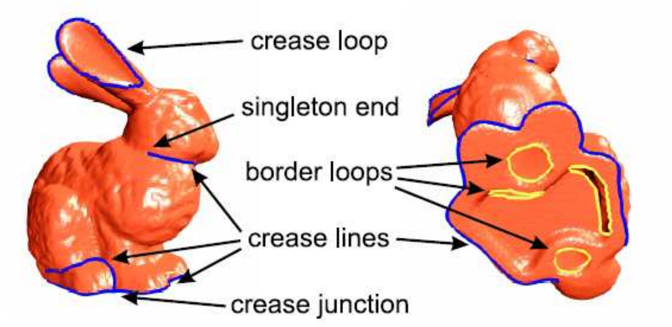


Figure 1.2: Example of a smooth surface of an object and the types of features to be located using the Feature Extraction algorithm. This image taken from “Feature Extraction” [1].

a very help input to this problem. If the point cloud of the object is densely sampled and the point cloud also has an accurate mesh, edge sharpening can occur to sharpen the mesh of the object and calculate the location of the edge. This is done by identifying the points that lie near the edge of an object in the mesh and processing the nearby surface information to find an intersection line. Figure 1.3 shows an example of a meshed point cloud of a cube that has the edges smoothed over.

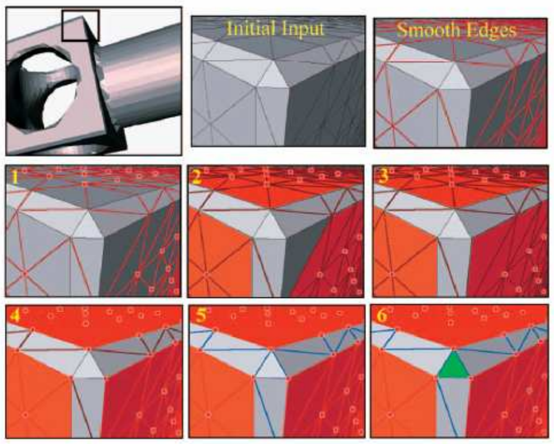


Figure 1.3: Example mesh of a simple cube object. Note that the edges of the cube are cut off in the meshing process. The edges can be recovered through edge sharpening of the cube. Image from “Sharpen & Bend” [2]

The information can be recovered to produce a point cloud of the cube that does have accurate edge lines, but the input to the algorithm has to be a mesh. One problem with the use of meshes is the tendency of a mesh to cut across the edges of an object or smooth out the corners of an object. This is a problem for point clouds captured by LiDAR because the object is not usually sampled on the edge. A point could fall on the edge of an object, but in general this is not the case. As mentioned above, meshes are also computationally expensive and do not always produce accurate results. Another downfall of this approach is the sensitivity of the algorithm to noise in the point cloud.

Edge sharpening of the mesh is not the focus of our research, but is useful because of the edge location in the point cloud. In order to create a good 3D model of a object, the edges and surface intersections of that object should be accurate. Since the point cloud does not sample on the edges of objects, estimation of the edges is important to modeling.

1.2.3 Multiple Sensor Registration

Multiple sensor data and registration [10, 11] are used to find a 3D model of a scene while simultaneously texturing the model. The point clouds are created using stereo images instead of laser range equipment. This approach is advantageous since texture data and surface color are included, but the accuracy of the point cloud is a function of distance to the stereo cameras. Laser scanners, on the other hand, produce point clouds with very little noise.

1.2.4 Point Cloud Meshing

Point cloud meshes [12, 13, 14, 15] are a very popular area of research. Meshing a point cloud [16] is a very wide area of research. Although we avoid the mesh application, we feel it is an important subject since meshes are used so much in computer graphics display and as input to many of the other papers we are mentioning. Generally, creating a mesh of a point cloud is a very complex and computationally expensive algorithm. In our algorithm, we use points that lie in a neighborhood to estimate the normal vectors. This is similar to a mesh, but we do not care about how the points are connected along the surface. We only use the neighborhood to infer the smoothness of the surface about a point.

1.2.5 Point Cloud Segmentation

Point cloud segmentation [17, 18, 19, 20] is the area closest to our research. The goal is to segment point clouds into meaningful section or structures. Segmentation can be done with both proximity and local smoothness comparisons as in [17] to grow the regions of the point cloud. This example starts with the low level processing to determine the smoothness of each point and then compares with neighbors similar to the meshing examples. [19] starts with the local smoothness to determine if points are on planar or smooth surfaces and then fits a mathematical surface accordingly. [18] is also very similar since they are calculating local smoothness and comparing on the global scale to find similar smoothness. The idea is very close to our idea, but we approach from the normal vector space instead of the point cloud space.

1.2.6 3D Model Inference

Model Inference of point clouds, [4, 3, 21, 22, 1], is usually accomplished by using an algorithm like those listed above to locate the edges or shapes of point clouds. Point simplification is usually used to find sections of the point cloud that are locally smooth and use fewer points to represent the locally flat area. This reduces the number of points in the point cloud, but still contains most of the data by adding knowledge about the areas of the point cloud that are flat.

The goals of 3D Model Inference change depending on the object that is to me modeled. In most situations, a smooth model is inferred in order to minimize the energy of the surface. Situations like this are usually for objects where no prior knowledge is available. The different algorithms attempt to locate features in the point cloud like edges and then connect them.

Model inference also assumes a sparse noisy point cloud. In order to find a smooth

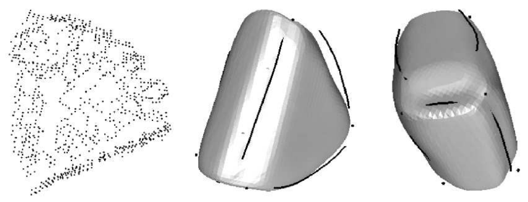


Figure 1.4: This is an example of the type of point cloud used in "Inference of Integrated Surface" [3]. The object being modeled is smooth and hard to see from just the point cloud. The point cloud is also noisy.

model of an object, they use every point in the point cloud to help with normal vector estimation. The point clouds used for this algorithm are more dense and our goal is to find the original geometry of the object. The man-made objects are also not smooth. The objects have many hard edges that need to be recovered in order to make a good model. Because of this, slightly different techniques than [4, 3] will be used in order to recover the original geometry.

1.3 Research Goal and Objectives

The original goal of the research was to be able to locate and segment planar surfaces in point cloud using the EM algorithm. We were able to extend this idea to segmenting cylinder and cones using a new likelihood function we derived. This new likelihood function is very versatile and can also be used with the MCMC algorithm. Using this method we were able to segment many different shapes from a complex point cloud. The three major sections of the research are explained by our objectives.

1.3.1 Objective 1. Location and Segmentation of Planar Surfaces in Point Clouds using the EM algorithm

Our original goal of the research was to be able to locate and segment planar surfaces from an unorganized point cloud. This was going to be done by clustering the normal vectors of each point using the Expectation Maximization (EM) algorithm. The normal vectors of planar point cloud sections appear as tight clusters which can be easily clustered using EM. Once the normal vector clusters are found, the orientations of planes in the point cloud are known. The points can be segmented and an equation for each plane can easily be found.

1.3.2 Objective 2. Location and Segmentation of More Complex Surfaces in Point Clouds using the EM algorithm

After planar surfaces could be segmented using the EM algorithm, the algorithm was extended to include two more simple shapes, Cylinders and Cones. In order to use the EM algorithm for more complex shapes, a new likelihood function had to be derived for each new shape. However, we were able to derive a new likelihood function that can be used to cluster all three shapes in the point cloud. Clustering does not have to be done for each shape independently, but can be done for all three types of shapes in the point cloud. We include a detailed derivation of the new likelihood function along with results for different point clouds.

1.3.3 Objective 3. Location and Segmentation of Surfaces using the MCMC algorithm

The likelihood function that was created in Objective 2 can be used in the MCMC algorithm. The MCMC algorithm is much more robust than the EM algorithm without the initialization needed by EM. MCMC also has the ability to handle much more complex point clouds than EM. We will show some clustering results for MCMC compared with EM. We will also show some point clouds that are too complex for EM to handle, but can be handled by MCMC.

CHAPTER 2

Point Cloud Feature Extraction

The goal of this thesis is to identify three basic shapes in point cloud: Planes, Cylinders and Cones. When the identification is complete, the quantity of each shape in the point cloud will be known along with the orientation in the point cloud. This information can be used to simplify point clouds by reducing the amount of data to simple equations where possible. The input to this algorithm could be a point cloud as seen in Fig. 2.1.

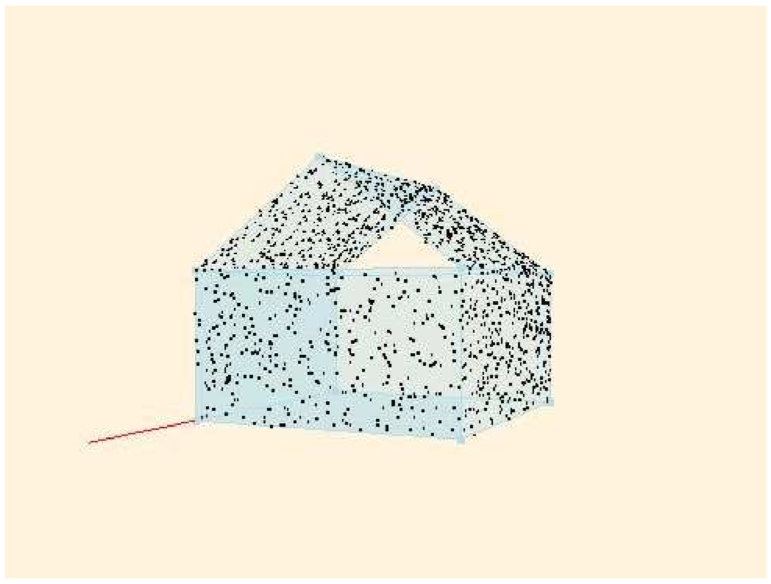


Figure 2.1: Example input point cloud of a simple house. The planes are shaded in with blue to help visualization. The red line is an axis and is used to help oriented the user only.

In order to find the orientation and number of planes in the house point cloud, clustering is done on normal vectors only in the normal vector space. Clustering the normal vectors allows a unique opportunity of being able to identify the type of shape being clustered (Plane, Cylinder or Cone) but the ambiguity of not knowing specific shape information.

Clustering is done on the normal vectors of the point cloud, but most point clouds are unorganized. An unorganized point cloud is simply a collection of three dimensional points with no normal vector or mesh information. The surface at a point cloud point is not known and can be oriented in any direction. The normal vectors can be inferred by studying the neighboring points of the point cloud.

Different shapes in the point cloud can have similar orientations and be mapped to the same cluster in normal vector space. For example, multiple planes with the same orientation all share the same normal vector. This ambiguity can be resolved by a second clustering algorithm in the point cloud space. This second clustering algorithm is similar to the first but will only be discussed briefly at the end of this thesis.

2.1 Feature Extraction

The features used for clustering are the normal vectors of the point cloud points. Point clouds are assumed to be densely sampled as to provide a good estimate of the surface normals. These normal vectors are calculated using the covariance of the points in a neighborhood around each point. [4, 1] In our case, each input point, $\mathbf{x}$, is a three dimensional spacial point. The neighborhood consists of the nearest m points,

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m]^T. \tag{2.1}$$

The covariance matrix of the neighborhood is

$$\mathbf{\Sigma} = \mathrm{E}[(\mathbf{X} - \mathrm{E}[\mathbf{X}])(\mathbf{X} - \mathrm{E}[\mathbf{X}])^T], \tag{2.2}$$

where $\mathrm{E}[\mathbf{X}]$ is the expected value of the points in $\mathbf{X}$. The Singular Value Decomposition, SVD, of Eq. (2.2) is found in order to obtain the eigenvectors and eigenvalues of the covariance matrix.

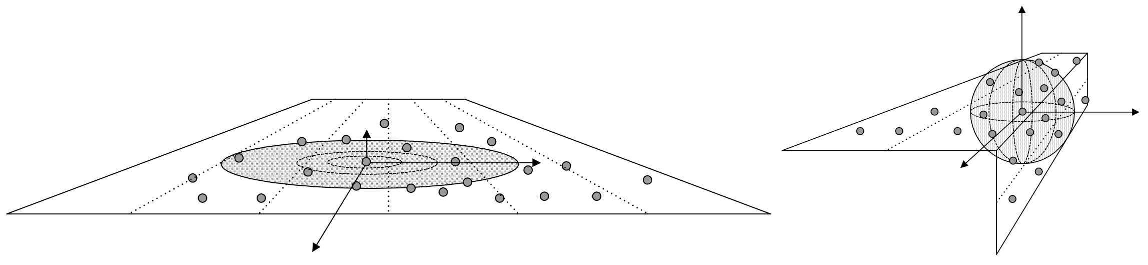


Figure 2.2: Two examples of normal vector calculation: (a) shows an area of the point cloud with a smooth planar surface. The data spreads mostly in two of the three dimensions. The covariance matrix of this spread will have a high condition number and the normal vector is very trustworthy; and, (b) shows the opposite case. This point is near a feature and the covariance matrix of this data spread will be well conditioned (close to 1) leading to a normal vector that is not trustworthy and should not be used for clustering.

2.2 Two Normal Vector Estimation Scenarios

Assuming a smooth surface surrounding each point, the eigenvector corresponding to the smallest eigenvalue is taken as the normal vector for said point. See Fig. 2.2a. The two larger eigenvectors, with eigenvalues λ_0 and λ_1, lie along the surface while the smallest eigenvector, with eigenvalue λ_2, is perpendicular to the surface. The relationship of the eigenvalues, λ, for a smooth surface is seen in Eq. (2.3),

$$\lambda_0 > \lambda_1 > \lambda_2, \qquad \lambda_0, \lambda_1 \gg \lambda_2. \tag{2.3}$$

Clearly the assumption of a smooth surface surrounding each point is not always the case. See Fig. 2.2b. When a point cloud point is close to a sharp feature on the object, the covariance matrix should be well conditioned. The eigenvalues will be approximately the same length meaning a well conditioned covariance matrix, Eq. (2.4),

$$\lambda_0 \approx \lambda_1 \approx \lambda_2. \tag{2.4}$$

Therefore, only point cloud points with an ill-conditioned covariance matrix should be used for clustering. The higher the condition number (meaning worse conditioning), the smoother the surface near the point. A higher condition number threshold will exclude a larger number of points but can lead to better modeling in the normal vector space. The condition number also depends on the size of the neighborhood used to calculate the normal vector and is application specific.

2.3 Potential Normal Estimation Problems

When calculating the normal vectors from a point cloud, there is an ambiguity in the direction. The normal vectors can be oriented one of two directions. A simple example would be the normal vector of a horizontal plane. The direction can either be up or down as both are mathematically correct. All normal vectors of a point cloud shape can be oriented outward from center, inward towards the center or some combination of both. To avoid ambiguity in the clustering process, we use both the positive and negative normal vectors as input to our algorithms. When the normal vectors are estimated for the point cloud in Fig. 2.3(a) and the well conditioned normal vectors are filtered out, the result is Fig. 2.3(b).

The image in Fig. 2.3(b) is of normal vector space. Each point is the head of a normal vector calculated from the point cloud in Fig. 2.3(a). All the normal vectors lie on the unit sphere as they are all normalized.

There are seven planes in the point cloud in Fig. 2.3(a) but there are only five cluster pairs in Fig. 2.3(b). The reason is the left and right walls of the house have the same normal vector. The front and back walls of the house also share a normal vector. The floor plane and the two roof planes have a unique normal vector when compared to the other planes in this point cloud. Also remember that we use both the positive and negative normal vectors for each input. This causes a mirror cluster on the opposite side of the unit sphere. You may realize there are only nine clusters of point in Fig. 2.3(b). This is due to the center cluster hiding the mirror cluster directly behind.

Another potential problem with normal vector estimation is smooth surfaces of objects with different shapes. A smooth sphere would be a good example. If the surface of the

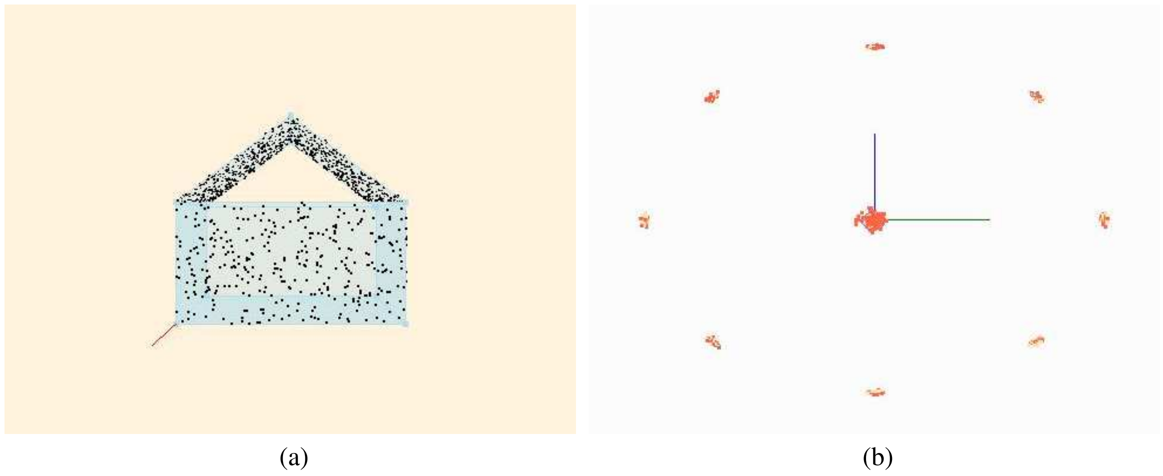


Figure 2.3: Point cloud of a simple house consisting of planes: (a) The house is made up of four walls, a floor and a roof; (b) shows the results of normal vector estimation for this point cloud. The clusters have noise from the noise in the point cloud. There are five orientations of planes which leads to ten total clusters. The solution consists of several small cluster pairs that fit tightly on the points.

sphere were sampled densely enough, the normal vectors could be included in the clustering process. The normal vectors of a sphere would not form a tight cluster in the normal vector space. We can not predict what would happen during the clustering process but acknowledge there are many problems that could occur related to the above scenario. It would probably be necessary to perform a more complex filtering of normal vector points than the simple condition number filtering we use in this thesis. In order to circumvent this problem, we assume the point clouds contain only combinations of three shapes since our research is on clustering and not normal vector estimation.

CHAPTER 3

EM Planar Clustering

We can now cluster the normal vectors calculated in Chapter 2. In this chapter, we are only clustering planar shapes using the EM algorithm. The clusters in this chapter can be thought of as three dimensional clusters in normal three dimensional space. The definition of a cluster will change in subsequent chapters and will be discussed more at that time. The EM algorithm is not modified for this chapter.

3.1 Clustering of Normal Vectors

Since the point cloud is assumed to have noise and roundoff error, and the normal calculation is only an estimation of the normal vector at each point, the normal vectors in the point cloud are clustered assuming a Gaussian Mixture Model in order to find the number and direction of the optimal normal vector clusters. A Gaussian Mixture Model consists of several Gaussian models that are combined to create a single density function. A two dimensional example can be seen in Figure 3.1.

The optimal number of normal vector clusters for the point cloud is estimated based on the Rissanen order identification criteria known as Minimum Description Length (MDL). The process is equivalent to Maximum Likelihood when the number of clusters is fixed. The results of the clustering algorithm will estimate the number of dominant normal vectors in the point cloud and the direction of the normal vector for each of the planes. The actual output of the clustering algorithm produces clusters that have different parameters as follows.

- K - the number of clusters in the data;

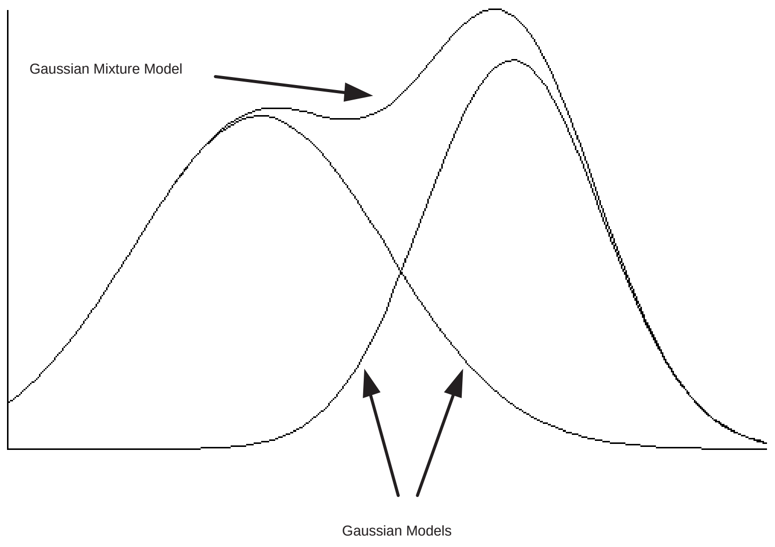


Figure 3.1: Example of a two dimensional Gaussian Mixture Model. There are two different Gaussian densities that are combined into one mixture model.

- π_k - the probability that a pixel belongs to a cluster $\boldsymbol{\theta}_k$. π_k is also known as the cluster mixing weight;
- $\boldsymbol{\mu}_k$ - the M dimensional mean of cluster $\boldsymbol{\theta}_k$;
- $\boldsymbol{\Sigma}_k$ - The $M * M$ covariance matrix for cluster $\boldsymbol{\theta}_k$.

In normal vector clustering, the mean, $\boldsymbol{\mu}_k$, of each cluster, $\boldsymbol{\theta}_k$, is the direction of the normal vector, $\vec{\mathbf{N}}_k = \langle A, B, C \rangle$. The covariance matrix, $\boldsymbol{\Sigma}_k$, for each cluster has information about the spread of the data that produced the cluster mean. The covariance matrix is used to check the confidence of each cluster. A covariance matrix with very small eigenvalues represents a tight cluster and a very accurate estimation of the normal vector. The cluster mixing weight, $\pi_k \in (0,1)(k = 1, 2, \ldots, K)$, contains information about the number of normal vectors that were used to calculate the every cluster. If the value is large, then many normal vectors belong to that cluster, meaning a high probability that cluster is a normal vector for a plane in the point cloud.

The EM clustering algorithm with MDL [23] comes from Charles Bouman in the School of Electrical Engineering from Purdue University. We present an overview of the clustering algorithm. The goal of the algorithm is to calculate the number and parameters of clusters from a given set of data.

The algorithm begins by initializing the number of clusters, K, to some large number. We use 20, but this should be a number larger than the final number of clusters. The parameters for each cluster are also initialized. For each cluster, the mean, $\boldsymbol{\mu}_k$, is set to the value of one of the data points and the covariance matrix, $\boldsymbol{\Sigma}_k$ is set to the covariance of the entire data set. These parameters are the *soft* parameters for each cluster. They are called *soft* because they will soon be changed by the Expectation Maximization (EM) algorithm.

3.1.1 Expectation Maximization

Intuitively, the EM algorithm is a two step iterative algorithm. First the likelihood values of points belonging to the current cluster parameter set are calculated. These likelihood values are used along with the point values to update the cluster parameter set. The algorithm is repeated until clusters converge.

Expectation

The likelihood of a point belonging to a cluster is calculated using the likelihood function,

$$p(\mathbf{y}_n|\boldsymbol{\theta}_k) = \frac{1}{(2\pi)^{\frac{M}{2}}|\boldsymbol{\Sigma}_k|^{\frac{1}{2}}} \exp\left\{ - \frac{(\mathbf{y}_n - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\mathbf{y}_n - \boldsymbol{\mu}_k)}{2} \right\}, \tag{3.1}$$

where $\mathbf{y}_n$ is the n^{th} point and $\boldsymbol{\theta}_k$ is the k^{th} cluster with parameters $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k)$. The likelihood value is calculated for all points and all clusters. Since the parent cluster is not known, the total likelihood for each point is calculated by summing over k,

$$p(\mathbf{y}_n|\boldsymbol{\Theta}) = \sum_{k=1}^{K} \pi_k p(\mathbf{y}_n|\boldsymbol{\theta}_k), \tag{3.2}$$

where $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_K\}$ represents the entire cluster parameter set. The log likelihood for the data set $\mathbf{Y}$ is calculated by taking the natural log of Eq. (3.2) and summing over N,

$$\ln p(\mathbf{Y}|\boldsymbol{\Theta}) = \sum_{n=1}^{N} \ln\left(\sum_{k=1}^{K} \pi_k p(\mathbf{y}_n|\boldsymbol{\theta}_k) \right). \tag{3.3}$$

EM Algorithm with MDL

```
1: Initialize EM Algorithm with a large number of clusters
2: repeat
3:    E Step:
4:    Calculate the probability of each point belonging to each cluster.
      $p(\mathbf{y}_n|\boldsymbol{\theta}_k)$ in Eq. (3.1)
5:    Calculate likelihood of each point given current parameter set,
      $p(\mathbf{y}_n|\boldsymbol{\Theta})$ in Eq. (3.2)
6:    Calculate Log-Likelihood of the entire data set, $\ln p(\mathbf{Y}|\boldsymbol{\Theta})$ in
      Eq. (3.3)
7:    Calculate the probability of a point belonging to a cluster,
      $p(\boldsymbol{\theta}_k|\mathbf{y}_n)$ in Eq. (3.4), for all clusters and points.
8:    M Step:
9:    Update $N_k$ and $\pi_k$ for all clusters for the next iteration, Eq. (3.5)
      and (3.6).
10:   Update $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ for each cluster using Eq. (3.7) and (3.8).
11: until Convergence
12: Calculate the MDL number for the given number of clusters, Eq. (3.9).
13: $K = K - 1$
14: if $K > 0$ then
15:    GOTO Step 2
16: end if
17: Determine the number of clusters corresponding to the Minimum MDL
    number
```

In order to update the parameters for the k^{th} cluster, the cluster probability is calculated using Eq. (3.1) and the current cluster weights,

$$p(\boldsymbol{\theta}_k|\mathbf{y}_n) = \frac{p(\mathbf{y}_n|\boldsymbol{\theta}_k)\pi_k}{\sum_{l=1}^{K} p(\mathbf{y}_n|\boldsymbol{\theta}_l)\pi_l}. \tag{3.4}$$

Maximization

Once the likelihood of all data points have been classified for the current cluster set, the cluster parameters, N_k, π_k, $\boldsymbol{\mu}_k$, and $\boldsymbol{\Sigma}_k$, are updated. After new parameters are calculated, the algorithm is repeated from the beginning until the new parameters for all clusters do not change from the previous iteration.

The number of points, N_k, for each cluster is calculated by summing the cluster probability over all points, N,

$$N_k = \sum_{n=1}^{N} p(\boldsymbol{\theta}_k|\mathbf{y}_n). \tag{3.5}$$

N_k is used to find the mixing weight, π_k, of each cluster for the next iteration,

$$\pi_k = \frac{N_k}{N}, \tag{3.6}$$

and to update the mean, $\boldsymbol{\mu}_k$,

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{n=1}^{N} \mathbf{y}_n p(\boldsymbol{\theta}_k|\mathbf{y}_n), \tag{3.7}$$

and covariance matrix, $\boldsymbol{\Sigma}_k$,

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k}\sum_{n=1}^{N} (\mathbf{y}_n - \boldsymbol{\mu}_k)(\mathbf{y}_n - \boldsymbol{\mu}_k)^T p(\boldsymbol{\theta}_k|\mathbf{y}_n). \tag{3.8}$$

The algorithm starts back at the Expectation step until the cluster parameters do not improve. Once the clusters parameters do not improve, the final cluster parameters, $\boldsymbol{\Theta}$, are recorded along with the number of clusters, K, and the final likelihood value for K clusters. The number of clusters is reduced by one and the EM process starts again until the number of clusters is 0. The problem with the process is that number of cluster that fit the data the best is always the most number of clusters. This is where the Minimum Description Length (MDL) part of the clustering algorithm helps.

3.1.2 Minimum Description Length

The MDL puts a penalty function on the number of clusters used to describe the data set and can be seen in Eq. (3.9).

$$MDL(K, \mathbf{\Theta}) = -\ln p(\mathbf{Y}|\mathbf{\Theta}) + \frac{L\ln(NM)}{2}, \tag{3.9}$$

where $L\ln(NM)$ is the penalty function, N is the number of data points and L is a constant defined by Equation (3.10).

$$L = K\left(1 + M + \frac{(M+1)M}{2}\right) - 1, \tag{3.10}$$

where K is the number of clusters and M is the dimension of the data. In our case, $M = 3$ for the three dimensional vectors being clustered.

The MDL number, also known as the Rissanen criterion, is calculated for the number of clusters in the clustering algorithm. The goal of the MDL is to find the number of clusters that best fits the data without over-fitting the data by using too many clusters. Once the MDL value is calculated for every number of clusters, the number of clusters that corresponds to the smallest MDL value is set as the number of clusters that best fits the data. The parameters for that number of clusters is used as the output from the clustering algorithm. An example output of the MDL is in Figure 3.2. The minimum number occurs at 5 clusters meaning this is the number of clusters that best fit the data.

3.2 Simulation Results

Simulation results for Chapter 3 were obtained by using point clouds created using a computer program that will calculate points between user created corner points. In order to test the plane location and position optimization for this algorithm, every shape that was tested is made up of planar surfaces. The current shapes are a Cube, Pyramid, Three Planes Crossing, Two Steps and Two Planes that come together like a roof. The point cloud program was created using Visual Studio 2005 and other processing was done by Matlab.

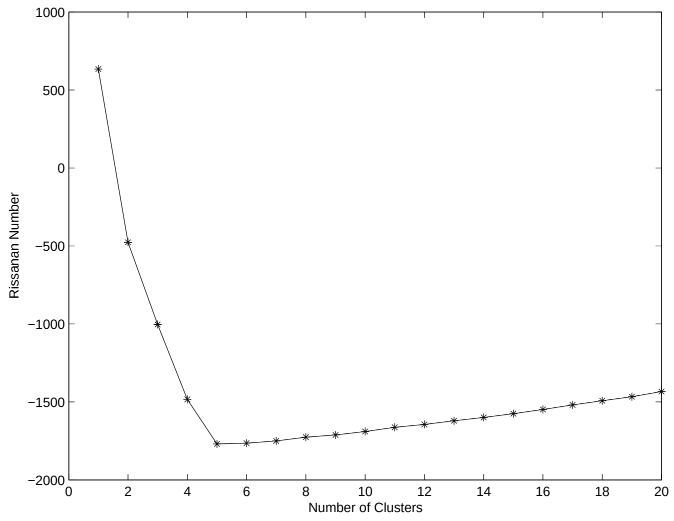


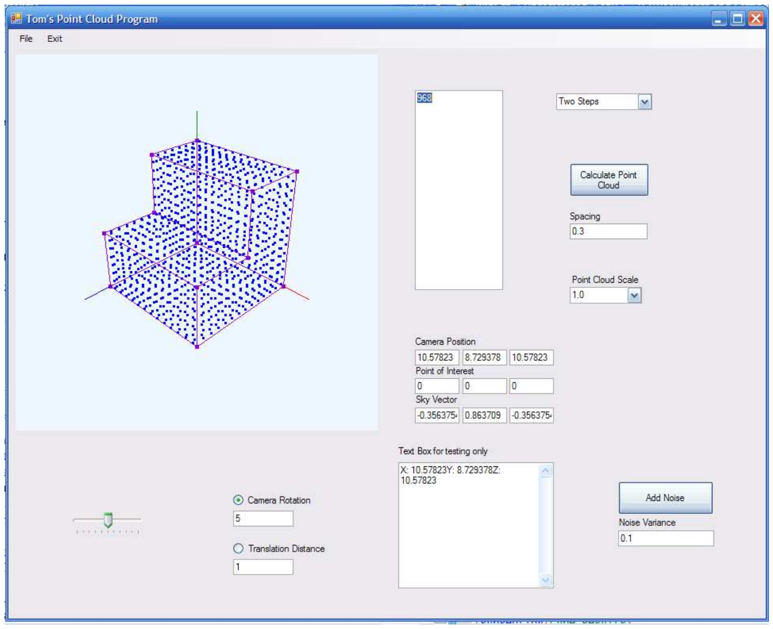
Figure 3.2: Example output of the clustering process. The X axis is the number of clusters and the Y axis is the Rissanen number for each number of clusters. In this case, the minimum Rissanen number occurs at 5 clusters.

The point clouds are created using a spacing between each of the points which is usually 0.5. The spacing can be adjusted along with the overall scale of the object in the point cloud. By default, each point cloud can fit in a cube that is 4 units on each side, but can be scaled to be any size. Many different densities of points in the point clouds can be tested easily. In order to test the ability of the algorithm, noise was added to each point cloud before testing. For each point cloud, white noise was added with variances of 0.0001 and 0.0025, respectively.

Once the different point clouds are created, Matlab is used to calculate the normal vector for each point and test the condition of the covariance matrix. It was found through the process of trial and error the best results were obtained using only those normal vectors which have a condition number greater than 25. The more complicated the point cloud is, the larger the condition number should be in order to get reliable results.

3.2.1 Point Cloud Creation Program

The program used to create the point could was created using Microsoft's Visual Studio 2005. The graphics were handled by Microsoft's graphics engine, DirectX 9.0c. Figure 3.3 shows a screen shot of the program. The program is very simple to operate. The user selects the shape of the point cloud from the drop down list on the right. The user sets the scale of the object and the spacing of the points to be drawn and the program computes the location of the points and draws them to the screen. Future revisions could make it possible to change the location of corner points and create different and more complex shapes.



968

Two Steps

Spacing
0.3

Point Cloud Scale
1.0

Camera Position
10.57823 | 8.729378 | 10.57823

Point of Interest
0 | 0 | 0

Sky Vector
-0.356375 | 0.863709 | -0.356375

Text Box for testing only
X: 10.57823Y: 8.729378Z: 10.57823

Camera Rotation
5

Translation Distance
1

Noise Variance
0.1

Figure 3.3: Screen shot of the program used to create the point cloud for testing. Program was created using Microsoft Visual Studio 2005.

3.2.2 Cube Results

The point cloud of a cube was first used because of the simplicity and ease of testing the program. There would be known results for the normal vectors and the location of the real planes in the image are also easily calculated. The cube can be seen in Figure 3.4.

There are six sides to the cube which means there are six planes for this shape. The

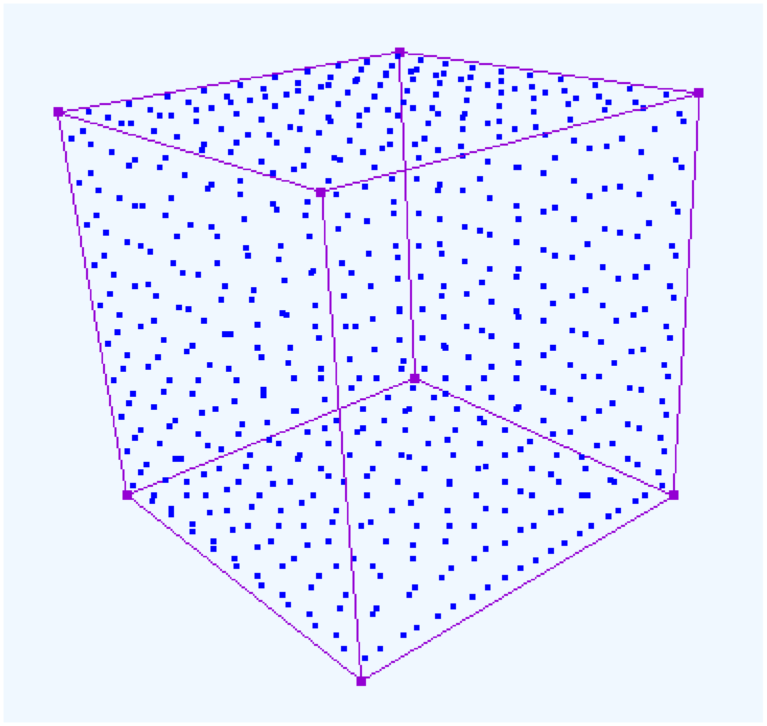


Figure 3.4: Point cloud of a cube. This point cloud contains six hundred points and each side of the cube is five units long.

point cloud is made up of 600 points. Each side of the cube is five units long with one corner at the point $(0, 0, 0)$ and the opposite corner at the corner $(5, 5, 5)$. The ground truth constant values for the planes are listed in the tables. Instead of writing the entire equation, we just write the constants in the form of $(A_{ideal}, B_{ideal}, C_{ideal})$. The estimated values for the constants also appear in the tables. They have the same format as the ground truth constants, $(A_{est}, B_{est}, C_{est})$. An example of the normal vectors calculated that are used for clustering can be seen in Figure 3.5.

Table 3.1 shows the results for the cube point cloud. There are three separate planes calculated for the three dominant normal vectors in the point cloud. The results are very close to the ground truth values which is expected.

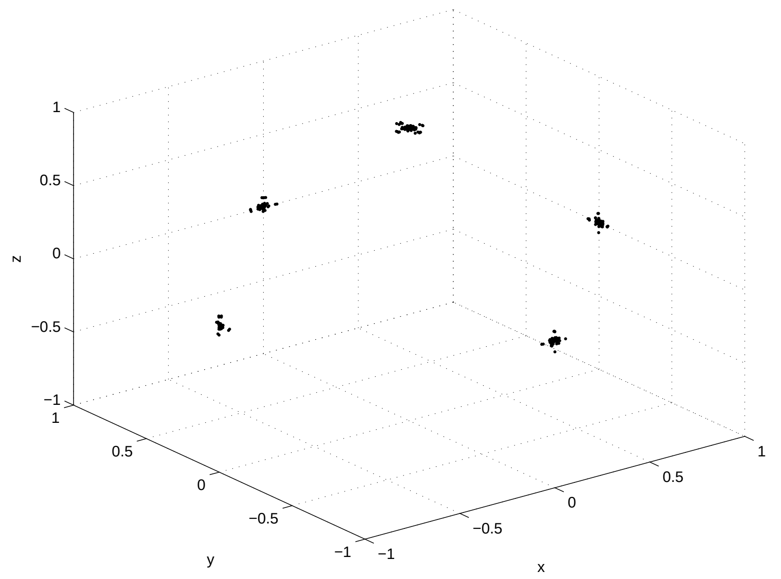


Figure 3.5: Each dot in this figure is a normal value that has been calculated for the point cloud. They are all normalized so they appear on a sphere with a radius of 1 around the origin point. Because of the ambiguity of the point cloud, normal vectors are sometimes calculated that point in opposite directions. There are five different clusters in this image that represent the three dominant normal vectors of the cube point cloud.

Table 3.1: Estimated and Ideal Values for the Cube Point Cloud.

	A	B	C	σ^2
EM	1.000	-0.003	0.000	
	0.009	1.000	-0.006	
	-0.007	0.005	1.000	0.0001
EM	1.000	-0.008	0.000	
	0.012	1.000	-0.007	
	-0.009	0.009	1.000	0.0025
Real	1.000	0.000	0.000	
	0.000	1.000	0.000	
	0.000	0.000	1.000	

3.2.3 Planes Results

Another point cloud that was used to test the algorithm was a point cloud of three different planes that meet at a point in the middle of each plane. This is a case that is used by the different researchers [4], [3], [24] etc. An example of the point cloud can be seen in Figure 3.6. The point cloud contains 768 points at a spacing of 0.3. This is a good point cloud to test because the point where the planes comes together makes normal vector calculation difficult. The program seems to be robust enough to handle this situation. The results for the Planes are in Table 3.2.

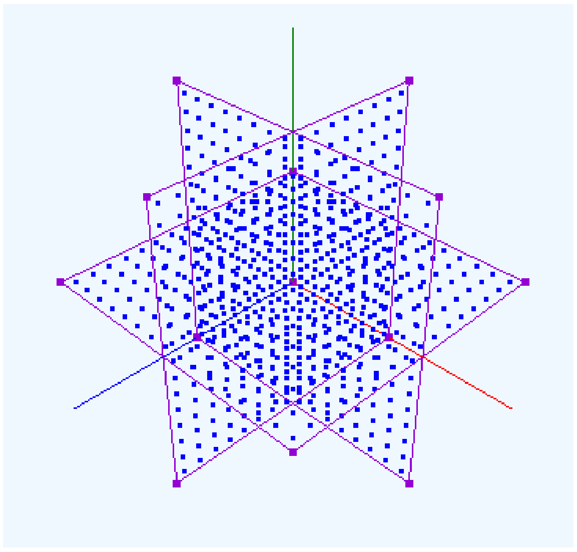


Figure 3.6: This is a point cloud of three different planes meeting at their centers. This point cloud contains 768 points and each side is 4.8 units long. Points are at a spacing of 0.3.

Table 3.2: Estimated and Ideal Values for the Three Planes Point Cloud.

	A	B	C	σ^2
EM	1.000	-0.004	0.003	
	-0.001	1.000	0.000	
	-0.001	0.000	1.000	0.0001
EM	0.997	0.006	0.075	
	-0.011	1.000	-0.011	
	0.005	0.003	1.000	0.0025
Real	1.000	0.000	0.000	
	0.000	1.000	0.000	
	0.000	0.000	1.000	

3.2.4 Pyramid Results

Figure 3.7 shows an image of the Pyramid point cloud. This point cloud consists of 836 points with a spacing of 0.3 . The pyramid is made up of 5 different planes that all have different normal vectors. The normal vector calculation of the pyramid estimate many outliers and appears to have the more clusters then it does. There results estimate the correct number of clusters as can be seen in Table 3.3. One reason for the good results in the pyramid is the density of the points in the original point cloud. If the object is not sampled densely enough, clustering problems may present themselves.

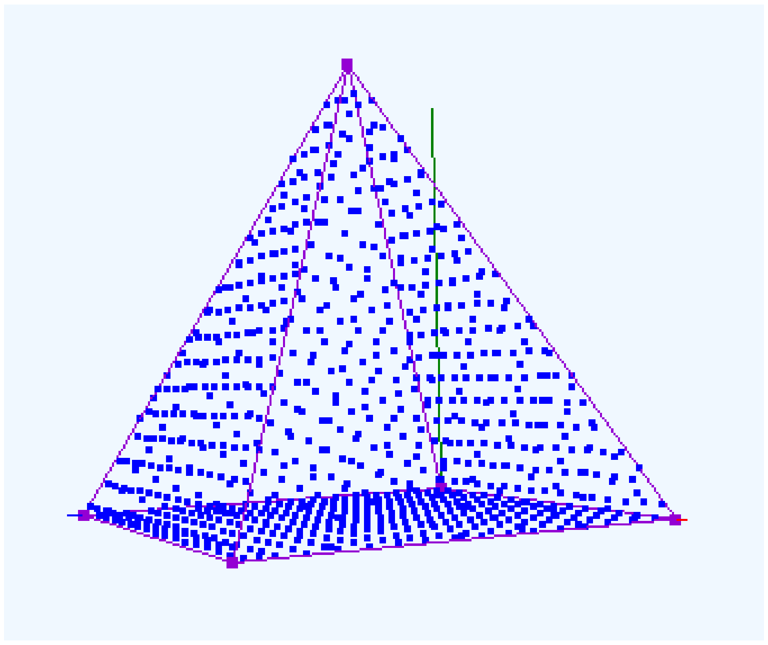


Figure 3.7: This is a point cloud of a pyramid. This point cloud contains 836 points and each side of the base is 4.8 units long. Points are at a spacing of 0.3.

Table 3.3: Estimated and Ideal Values for the Pyramid Point Cloud.

	A	B	C	σ^2
EM	-0.001	-0.448	0.894	
	-0.002	0.451	0.892	
	0.895	-0.447	0.002	
	0.894	0.448	0.002	
	-0.001	-1.000	0.002	0.0001
EM	0.002	-0.450	0.893	
	-0.003	0.449	0.894	
	0.893	-0.450	0.003	
	0.892	0.453	0.005	
	0.005	1.000	-0.001	0.0025
Real	0.000	-0.447	0.894	
	0.000	0.447	0.894	
	0.894	-0.447	0.000	
	0.894	0.447	0.000	
	0.000	1.000	0.000	

3.2.5 Steps Results

The steps in Figure 3.9 is probably the most complicated point cloud we tested with this algorithm. There are only three different dominant normal vectors for this shape, but there are a total of eight planes. Two of the normal vectors have three planes associated with each one. This point cloud has a spacing of 0.25 between each point and 1408 points for the entire point cloud. Very good results are estimated for this and every other point cloud that was tested. The constant values are within 1% of the ideal values. The results for the Planes are in Table 3.4.

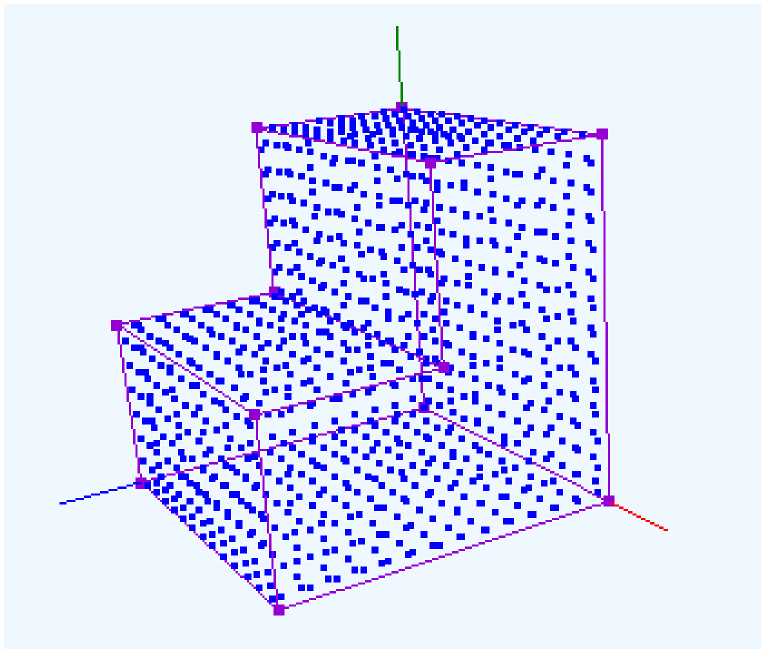


Figure 3.8: This is a point cloud of two steps. This point cloud contains 968 points and each side of the base is 4 units long. The points are spaced at 0.25.

Table 3.4: Estimated and Ideal Values for the Steps Point Cloud.

	A	B	C	σ^2
EM	1.000	0.000	-0.006	
	0.000	1.000	-0.007	
	0.000	0.004	1.000	0.0001
EM	1.000	0.000	0.003	
	0.000	1.000	0.007	
	0.000	0.004	1.000	0.0025
Real	1.000	0.000	0.000	
	0.000	1.000	0.000	
	0.000	0.000	1.000	

3.2.6 Roof Results

This is a point cloud that was created in order to test the normal vector calculation between two different planes that are not perpendicular to each other. The height of the roof can change but the overall scale of the point cloud does not change. This allows different angles to be tested. The results appear in Table 3.5.

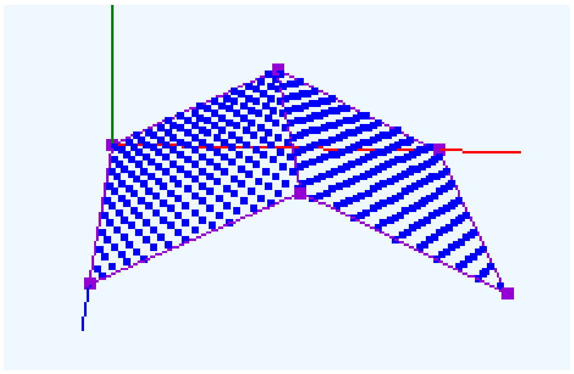


Figure 3.9: This is a point cloud of a roof. This point cloud contains 450 points and each side of the base is 4 units long. The points are spaced at 0.2.

Table 3.5: Estimated and Ideal Values for the Roof Point Cloud.

	A	B	C	σ^2
EM	0.437	0.899	−0.001	
	0.444	−0.896	0.002	0.0001
EM	0.438	0.899	−0.003	
	0.445	−0.898	0.001	0.0025
Real	0.447	0.894	0.000	
	0.447	−0.894	0.000	

CHAPTER 4

New Likelihood Function for EM and MCMC

In order to model cylinders and cones, we needed a new likelihood function. As in Chapter 3, we are clustering in the normal vector space using only the normal vector values calculated in Section 2.1. All of the normal vectors, Eq. (4.1), are normalized (length of 1). As before, these can be visualized as points that lie on the unit sphere,

$$\vec{N} = \langle A, B, C \rangle, \tag{4.1}$$

where $\vec{N}$ is the normal vector and A, B, C are the x, y, z components of the normal vector.

The Gaussian likelihood function works very well on planes in the point cloud since the normal vectors bunch together in a tight 3D cluster. The mapping process for planes can be seen in Fig. 4.1. This is not the case for cylinders, Fig. 4.2, or cones, Fig 4.3. The normal vectors for both cylinders and cones are circles in the normal vector space. Normal vectors for cylinders map as a unit circle. Normal vectors for cones map as a circle with radius between 0 and 1 depending on the interior angle of a cone.

Our new likelihood function models a shape in the normal vector space as the intersection of a plane, Eq. (4.2), and the unit sphere. To avoid confusion, we refer to this as the cluster plane instead of a point cloud plane in the real world space which will just be called a plane. The normal vector of the cluster plane will also be known as the cluster normal vector.

$$Ax + By + Cz + D = 0, \tag{4.2}$$

where the cluster normal vector, Eq. (4.1), is perpendicular to the surface of the cluster plane and D is the offset from the origin of the cluster plane. Our cluster mean vector $\boldsymbol{\mu}$

will change from three dimensional to four dimensional,

$$\boldsymbol{\mu} = \langle A, B, C, D \rangle. \tag{4.3}$$

Since the likelihood function is modified as the intersection of the cluster plane and the unit sphere, the parameters in Eq. (4.3) are restricted to the following values,

$$\begin{gathered} -1 \leq A \leq 1, \\ -1 \leq B \leq 1, \\ -1 \leq C \leq 1, \\ ||\langle A, B, C \rangle|| = 1, \\ |D| \leq 1. \end{gathered} \tag{4.4}$$

There is a very helpful reason to model shapes in this manner; All three shapes can be modeled using the same likelihood function. Consider a plane in the real world space with a normal vector of $\langle 0, 0, 1 \rangle$, Fig. 4.1. All of the points on the plane have the same normal vector, $\vec{N} = \langle 0, 0, 1 \rangle$, so when they appear in normal vector space, they appear as a very tight cluster. This cluster of points can be thought of as a cluster plane tangent to the unit sphere. In other words, the cluster plane has an offset value, $D = 1$, which makes the cluster mean value $\boldsymbol{\mu} = \langle 0, 0, 1, 1 \rangle$.

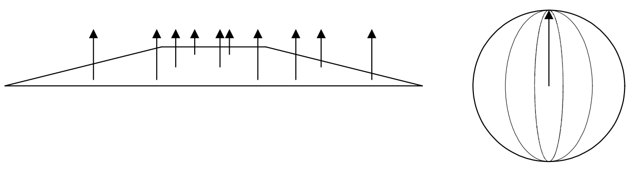


Figure 4.1: Example of plane in the real world space and the resulting normal vector mapping. The plane consists of several different normal vectors for each point in the point cloud. When these normal vectors are mapped to the normal vector space, they all point the same direction and overlap leading to a very tight clustering.

This idea can be extended to both the cylinder and cone cases. First the cylinder case in Fig. 4.2. All of the normal vectors calculated for the points on the cylinder point outward from the center. When these normal vectors are mapped to the normal vector space, they create a ring with radius, $r = 1$. The center line of the cylinder determines the orientation of the ring in normal vector space because the direction is the same as the cluster normal vector. In Fig. 4.2, the normal vector is the same as Fig. 4.1, $\vec{N} = \langle 0, 0, 1 \rangle$, but the offset value is different, $D = 0$. The mean vector for the cylinder is $\boldsymbol{\mu} = \langle 0, 0, 1, 0 \rangle$.

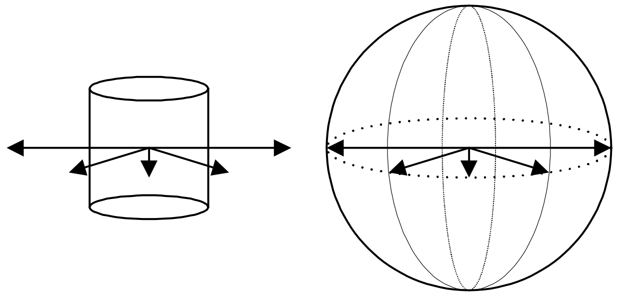


Figure 4.2: Example of the likelihood function in the case of a cylinder. The normal vectors of the cylinder all point outward from the center of the cylinder. When the normal vectors are mapped to the normal vector space, they create a ring of points. For cylinders, the ring is always a unit circle that is centered at the origin. The center line of the cylinder is the same as the cluster normal vector used to model the cylinder.

The cone case, Fig. 4.3, is the general case for all of the shapes. When using the new likelihood function, the D value determines the shape that is found in the clustering process. This can be seen in Table 4.1. The D value is also related to the interior angle of the cone, Eq. (4.5),

$$D = \sin^{-1} \frac{\phi}{2}, \tag{4.5}$$

where ϕ is the interior angle of the cone. If ϕ is increased to π radians, the cone is a plane. When ϕ is decreased to 0 radians, the cone tip lies at infinity and the cone is a cylinder. The radius, r, of the cluster ring is also related to the interior angle of the cone, Eq. (4.6),

$$r = \cos^{-1} \frac{\phi}{2}. \tag{4.6}$$

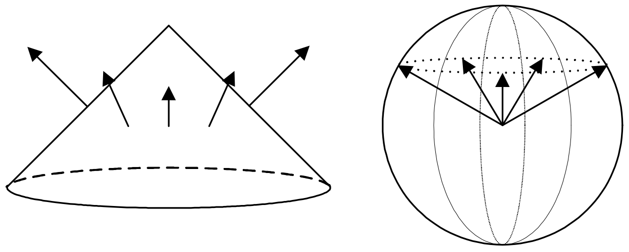


Figure 4.3: An example of a cone case show that the normal vectors also create a ring in the normal vector space. This is the general case that can be extended to the plane and cylinder cases. The plane case is a cone cluster with a ring of radius 0 and the cylinder case is a cone cluster with radius 1.

Table 4.1: Table of the D (cluster plane offset) values and the corresponding shapes being modeled. All three shapes can be thought of as cones. Planes and cylinders are extreme cases of the cone have D values of 1 and 0 respectively.

D Value	Shape
$D \cong 0$	Cylinder
$0 < \lvert D \rvert < 1$	Cone
$\lvert D \rvert \cong 1$	Plane

4.1 Arc Length Calculation

The clusters we are using for modeling are rings that lie on a unit sphere. The mean value is not a point, but a ring that can be as large as a unit circle. The minimum distance between a normal vector point and the cluster mean along the unit sphere is used in the likelihood function for both the Modified EM algorithm (Chapter 5) and the RJ-MCMC algorithm (Chapter 6). Since the normal vectors and the cluster means exist only on the unit sphere, we use the distance traveled on the unit sphere as the distance in the likelihood calculation. The arc length, $D_{arc}(\mathbf{y}_n, \boldsymbol{\mu}_k)$, is calculated using normal vector point $\mathbf{y}_n = (x_n, y_n, z_n)$ and cluster mean $\boldsymbol{\mu}_k = \langle A, B, C, D \rangle$ for the k_{th} cluster. The arc length is calculated for all normal vectors, $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\}$, to every $K \geq 1$ clusters.

The arc length, Fig. 4.4, for all points is calculated by first calculating the projected distance, d to a reference cluster plane ($N_{ref} = \langle A_k, B_k, C_k \rangle$, $D_{ref} = 0$) passing through the origin,

$$d = |A_k x + B_k y + C_k z|. \tag{4.7}$$

The arc length, l, is derived from d and radius, $r = 1$, as follows,

$$l = \sin^{-1} \frac{d}{r} = \sin^{-1} d. \tag{4.8}$$

The arc length, L, of the current cluster to the reference cluster is derived from the current cluster D value and the radius, $r = 1$, in a similar manner,

$$L = \sin^{-1} \frac{|D|}{r} = \sin^{-1} |D|. \tag{4.9}$$

The two arc lengths, l and L, are calculated in this manner in order to calculate the true arc length from a point to the cluster plane,

$$D_{arc}(\mathbf{y}_n, \boldsymbol{\mu}_k) = |L - l|. \tag{4.10}$$

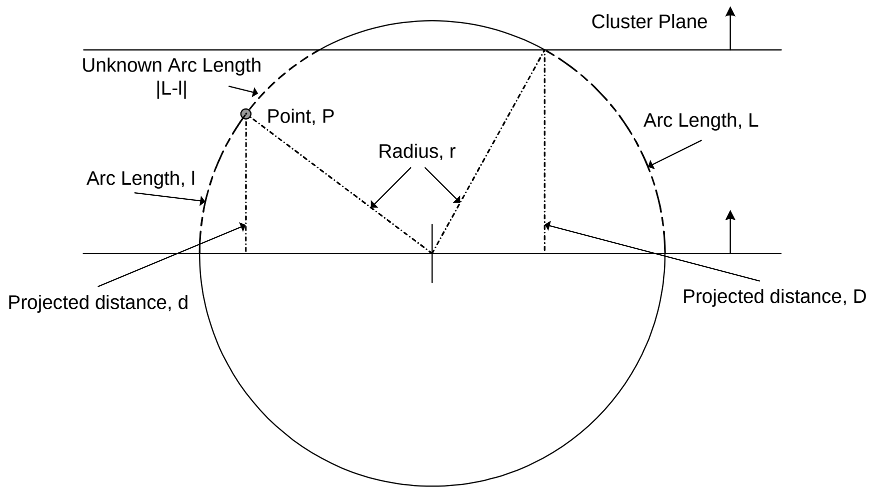


Figure 4.4: Example of the arc length calculation. The final arc length is the difference between the two reference arc lengths, Eq. (4.10).

4.2 Cluster Variance

Since we are using a one dimensional distance from each normal vector to the cluster mean, we no longer have a covariance matrix for each cluster. We instead have a single variance value. When the EM algorithm was attempted using the new likelihood function, we found the algorithm found the incorrect solution. A good example of this problem can be seen in Fig. 4.5.

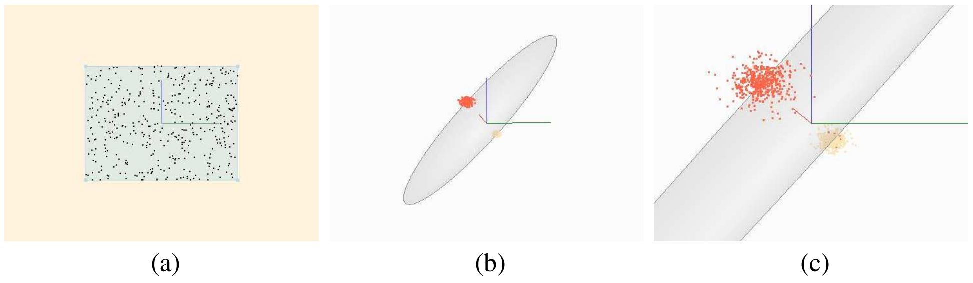
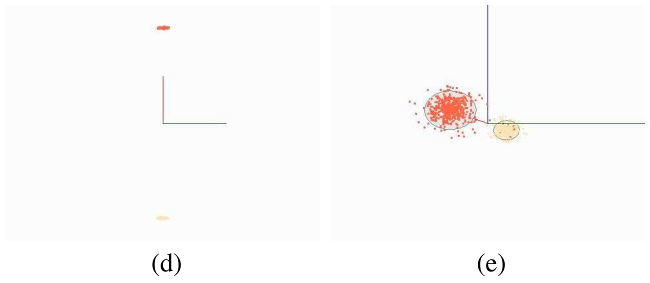



Figure 4.5: Point Cloud and clustering results of a simple plane: (a) is a simple plane with normal vector pointing out of the page; (b)–(c) show the bad results obtained from the clustering algorithm. The solution, visualized by the transparent grey circle, cuts through the cluster; and, (d)–(e) show the correct clustering solution. There are two cluster representing the positive and negative normal vector of the plane. There is a small grey circle about each cluster which is expected.

The point cloud in Fig. 4.5(a) is of a single planar surface. Fig. 4.5(b)–(c) show the incorrect clustering solution. For comparison, the correct solution can be seen in Fig. 4.5(d)–(e). The correct solution should be a tight circle close to the points. The incorrect solution in this case found a cylinder in the point cloud instead of a planar surface. The incorrect solution in Fig. 4.5 is a local likelihood maxima that can be a global likelihood maxima at times.

Our solution for this problem was to study the variance of different shapes in the normal vector space. We empirically found the variance of each cluster varied exponentially with the offset value D. We also found this exponential function changed very little with changes of noise in the point cloud. With this knowledge, we update the variance of each cluster using the exponential function based on the D value of each cluster. This update process further explored in Chapter 5. This relationship of variance versus offset value D can be seen in Fig. 4.6.

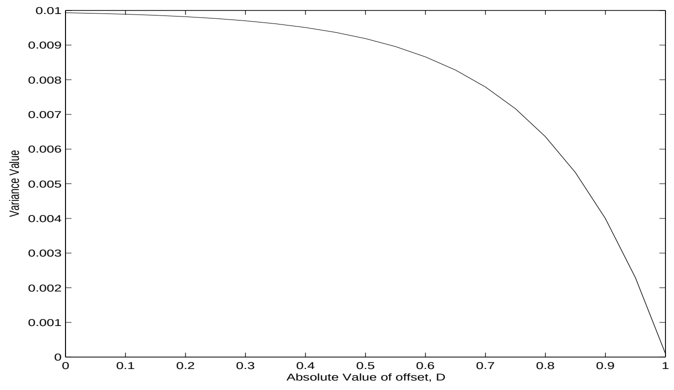


Figure 4.6: Graph of the exponentially decreasing variance based on the D value. This variance is plotted against the absolute value of D since the value of the variance only depends on the distance of the plane from the origin.

CHAPTER 5

EM Clustering with New Likelihood Function

Now that we have derived a new likelihood function, we will revisit and modify the EM algorithm. We still assume a Gaussian distribution of the data, but we have modified the likelihood calculation [25] and the cluster update steps to work with the ring clusters. These steps will be discussed further in the following Sections.

5.1 Modified EM Algorithm

The mixing weights for each cluster are $\pi_k \in (0,1)(k = 1, 2, \ldots, K)$ which is subject to summation $\sum_{k=1}^{K} \pi_k = 1$. The entire parameter set is denoted by $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_K\}$ where each cluster has the parameters $\boldsymbol{\theta}_k = (\boldsymbol{\mu}_k, \Sigma_k, \pi_k)$. $\boldsymbol{\mu}_k = \langle A_k, B_k, C_k, D_k \rangle$ is the cluster mean and Σ_k is the cluster variance. The variance value is now one dimensional as discussed in Chapter 4.2.

5.1.1 Expectation

The probability density function for normal point $\mathbf{y}_n$ given cluster $\boldsymbol{\theta}_k$ is given by

$$p(\mathbf{y}_n|\boldsymbol{\theta}_k) = \frac{\exp\left\{\frac{D_{arc}(\mathbf{y}_n,\boldsymbol{\mu}_k)\Sigma_k^{-1}D_{arc}(\mathbf{y}_n,\boldsymbol{\mu}_k)}{-2}\right\}}{(2\pi)^{\frac{3}{2}}|\Sigma_k|^{\frac{1}{2}}}, \tag{5.1}$$

where $D_{arc}(\mathbf{y}_n, \boldsymbol{\mu}_k)$ is the arc length from Chapter 4.1. The cluster parent is not known so sum over k to find the likelihood of each point given the current parameter set, $\boldsymbol{\Theta}$,

$$p(\mathbf{y}_n|\boldsymbol{\Theta}) = \sum_{k=1}^{K} \pi_k p(\mathbf{y}_n|\boldsymbol{\theta}_k), \tag{5.2}$$

```
 EM Algorithm with MDL
1: Initialize EM Algorithm with a large number of clusters
2: repeat
3:    E Step:
4:    Calculate the probability of each point belonging to each cluster.
      $p(\mathbf{y}_n|\boldsymbol{\theta}_k)$ in Eq. (5.1)
5:    Calculate the likelihood of each point given current parameter
      set, $p(\mathbf{y}_n|\boldsymbol{\Theta})$ in Eq. (5.2)
6:    Calculate Log-Likelihood of the entire data set, $\ln p(\mathbf{Y}|\boldsymbol{\Theta})$ in
      Eq. (5.3)
7:    Calculate the probability of a point belonging to a cluster,
      $p(\boldsymbol{\theta}_k|\mathbf{y}_n)$ in Eq. (5.4), for all clusters and points.
8:    M Step:
9:    Update $N_k$ and $\pi_k$ for all clusters for the next iteration, Eq. (5.6)
      and (5.7).
10:   Use $N_k$ and $\pi_k$ to update the mean normal vector, $\vec{N_k} = \langle A_k, B_k, C_k \rangle$,
      using Eq. (5.8)
11:   Update the offset of the cluster mean, $D_k$, using Eq. (5.9) and
      (5.10).
12: until Convergence
13: Calculate the MDL number for the given number of clusters, Eq. (5.11).
14: $K = K - 1$
15: if $K > 0$ then
16:    GOTO Step 2
17: end if
18: Determine the number of clusters corresponding to the Minimum MDL
    number
```

then the log-likelihood for the entire data set given the current parameter set is

$$\ln p(\mathbf{Y}|\boldsymbol{\Theta}) = \sum_{n=1}^{N} \ln\left(\sum_{k=1}^{K} \pi_k p(\mathbf{y}_n|\boldsymbol{\theta}_k)\right). \tag{5.3}$$

In order to update the parameters for the Maximization step, the probability of cluster $\boldsymbol{\theta}_k$ given data point $\mathbf{y}_n$ is calculated for all data points and all clusters,

$$p(\boldsymbol{\theta}_k|\mathbf{y}_n) = \frac{p(\mathbf{y}_n|\boldsymbol{\theta}_k)\pi_k}{\sum_{l=1}^{K} p(\mathbf{y}_n|\boldsymbol{\theta}_l)\pi_l}. \tag{5.4}$$

which logically sums to 1 over k,

$$\sum_{k=1}^{K} p(\boldsymbol{\theta}_k|\mathbf{y}_n) = 1. \tag{5.5}$$

5.1.2 Maximization

The number of points, N_k, belonging to each cluster for the next iteration is calculated by summing the probability of all points belonging to cluster $\boldsymbol{\theta}_k$,

$$N_k = \sum_{n=1}^{N} p(\boldsymbol{\theta}_k|\mathbf{y}_n). \tag{5.6}$$

N_k is used to find the mixing weight, π_k, of each cluster for the next iteration,

$$\pi_k = \frac{N_k}{N}, \tag{5.7}$$

and to update the cluster mean, $\boldsymbol{\mu}_k$. $\boldsymbol{\mu}_k$ is calculated in a similar manner to the normal vector estimate of Chapter 2.1. A weighted covariance matrix, $\boldsymbol{\Sigma}_{\boldsymbol{\mu}_k}$, is calculated using the data set, $\mathbf{Y}$ and the current cluster center, $\boldsymbol{\nu}_k = |D_k| \cdot \langle A_k, B_k, C_k \rangle$,

$$\boldsymbol{\Sigma}_{\boldsymbol{\mu}_k} = \frac{1}{N_k} \sum_{n=1}^{N} (\mathbf{y}_n - \boldsymbol{\nu}_k)(\mathbf{y}_n - \boldsymbol{\nu}_k)^T p(\boldsymbol{\theta}_k|\mathbf{y}_n). \tag{5.8}$$

$\boldsymbol{\mu}_k$ is the eigenvector corresponding to the smallest eigenvalue of the cluster covariance matrix, $\boldsymbol{\Sigma}_{\boldsymbol{\mu}_k}$. This process is discussed in greater detail in Section 5.1.4.

N_k is also used to calculate the average arc length, ϕ_k, for a cluster.

$$\phi_k = \frac{1}{N_k} \sum_{n=1}^{N} D_{arc}(\mathbf{y}_n, \boldsymbol{\mu}_k) p(\boldsymbol{\theta}_k|\mathbf{y}_n), \tag{5.9}$$

which is used to update the D_k in the parameter set for the next iteration,

$$D_k = \sin(\phi_k). \tag{5.10}$$

D_k is also used to set the variance value for each cluster for the next iteration, Chapter 4.2.

5.1.3 MDL

The MDL [26] section of the algorithm has not changed from Chapter 3 but will be stated again for convenience. After the EM algorithm has converged for a parameter set, the Minimum Description Length number, $MDL(K, \mathbf{\Theta})$, is calculated for the parameter set by adding the MDL Penalty value (the second term of Eq. (5.11)) to the negative log-likelihood,

$$MDL(K, \mathbf{\Theta}) = -\ln p(\mathbf{Y}|\mathbf{\Theta}) + \frac{1}{2} L \ln(NM). \tag{5.11}$$

where N is the number of points in the data set, M is the dimension of the feature vector and L is defined using the current number of clusters, K,

$$L = K\left(1 + M + \frac{(M+1)M}{2}\right) - 1. \tag{5.12}$$

This process is performed for a large number of clusters and repeated with one less cluster until the number of clusters is 1. The lowest MDL number corresponds to the correct number of clusters in the model.

5.1.4 EM Algorithm Mean Update

The update section of the EM algorithm calculates a new mean by first calculating a new cluster normal vector, $\vec{N}_k$. In order to get the new cluster normal vector, the covariance matrix, Eq. 5.14, for each cluster has to be calculated using

$$\mathbf{c}_{w_{nk}} = p(\boldsymbol{\theta}_k|\mathbf{y}_n)^{1/2}(\mathbf{y}_n - \boldsymbol{\nu}_k), \tag{5.13}$$

where $\mathbf{c}_{w_{nk}}$ is the vector from $\mathbf{y}_n$ to $\boldsymbol{\nu}_k = |D_k| \cdot \langle A_k, B_k, C_k \rangle$ which is the center point of $\boldsymbol{\theta}_k$. This vector is weighted by the root of the probability, $p(\boldsymbol{\theta}_k|\mathbf{y}_n)$ of point $\mathbf{y}_n$ belonging to cluster $\boldsymbol{\theta}_k$. The covariance matrix for cluster $\boldsymbol{\theta}_k$, is defined as

$$\mathbf{\Sigma}_{\boldsymbol{\mu}_k} = \mathrm{E}[\mathbf{C}_k \mathbf{C}_k^T], \tag{5.14}$$

where $\mathbf{\Sigma}_{\boldsymbol{\mu}_k}$ is the covariance matrix and $\mathbf{C}_k$ is the vector containing all weighted distance vectors, $\mathbf{c}_{w_{nk}}$ for cluster $\boldsymbol{\theta}_k$.

The eigenvector corresponding to the smallest eigenvalue of the covariance matrix is the new normal vector for the EM update step. Since the clusters are rings, they are ideally two dimensional. There is some spread in the third dimension, but it is much smaller than the spread in the other two dimensions. The vectors of points lying near the ring have a much higher weight than points far from the ring.

$$\lambda_0 > \lambda_1 > \lambda_2, \qquad \lambda_0, \lambda_1 \gg \lambda_2, \tag{5.15}$$

where λ is an eigenvalue of $\mathbf{\Sigma}_{\boldsymbol{\mu}_k}$. Once the normal vector is calculated for each cluster, the offset value, D_k, can also be calculated by averaging the weighted projection of the points to the origin plane.

5.2 EM Clustering Limitations

There are three main issues when using EM for clustering in the normal vector space. The first is the ambiguity of calculating a normal vector from a point cloud. The normal vectors are calculated, [1, 4, 3], in a manner similar to the mean update for the clusters, Section 5.1.4. When calculated in this manner, the normal vector can be oriented one of two directions. A simple example would be the normal vector of a horizontal plane. The direction can either be up or down as both are mathematically correct. All normal vectors of a point cloud shape can be oriented outward from center, inward towards the center or some combination of both. To avoid ambiguity in the clustering process, we use both the positive and negative normal vectors as input to the EM algorithm. This can clearly be seen in Fig. 6.2(c)–(d). The top and bottom ring are from the cone object in the point cloud. The top ring is the outward normal vectors and the bottom ring the inward normal vectors. If viewing this document in color, the yellow points are positive and the red points are negative values of the input. The clusters that appear in Table 6.2– 6.6 are just one of the cluster pair. Each cluster in the cluster pair is differentiated by the sign of the D value.

The second issue is the quality of the input points. The normal vector calculation works best when the point is on a flat surface as in Fig. 5.1(a). The normal vector calculated in this example has a high probability of being the correct normal vector for the surface at that point.

Fig. 5.1(b) shows an example of a less trustworthy condition number calculation. The point lies near an sharp feature in the point cloud (in this case and edge). The quality of the normal vectors can easily be determined by the condition number of the covariance matrix used to calculate each normal vector. The covariance matrix in each figure is visualized by the circles around the point. In Fig. 5.1(a) the covariance matrix has very little spread in the vertical direction, meaning a small eigenvalue which leads to a large condition number. Fig. 5.1(b) show spread approximately equal in all directions meaning the covariance matrix is well conditioned.

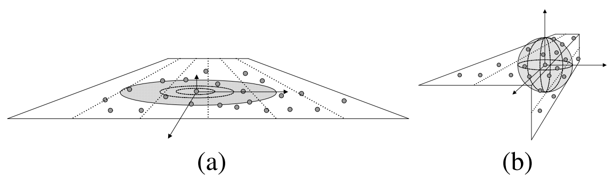


Figure 5.1: Two examples of normal vector calculation: (a) shows an area of the point cloud with a smooth planar surface. The data spreads mostly in two of the three dimensions. The covariance matrix of this spread will have a high condition number and the normal vector is very trustworthy; and, (b) shows the opposite case. This point is near a feature and the covariance matrix of this data spread will be well conditioned (close to 1) leading to a normal vector that is not trustworthy and should not be used for clustering.

The third issue is the EM initialization. This is potentially always a problem with EM due to local minima in the data set. The data in our case has a limited area to exist. This can lead to great overlap with increasing complexity of the point cloud. We use a simple initialization algorithm that tests the likelihood of several random parameter sets to start with a parameter set in the correct neighborhood. Some limitations can be seen in Fig. 6.3 and 6.5.

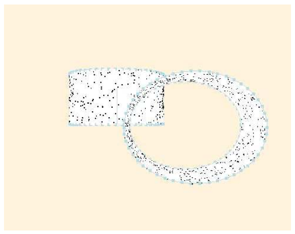
5.3 Simulation Results

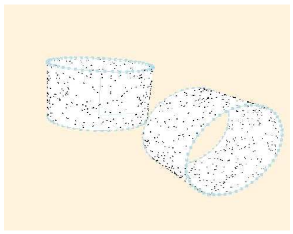
We have included some simulation results to aid in the visualization of this algorithm. The point clouds were created with a newer program than was used in Chapter 3. The program has a similar appearance but has the ability to create much more complex point clouds. There is no spacing of points in this chapter as there was before. The points are created randomly allowing the user to test many different point clouds of the same objects. Noise can also be added to the point clouds with any variance.

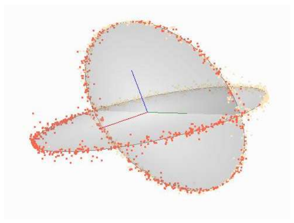
The EM and MCMC (Chapter 6) algorithms are contained in the new program so Matlab does not have to be utilized. The program was created using Visual C# and Microsoft's DirectX graphics engine. The speed of C# allowed us to create and test point clouds up to ten times the size of Chapter 3. The visualizations of this chapter are also going to be a little different. We show two views of the original point cloud with a transparent shape outline to help the reader see the shapes tested. We also show the results in the normal vector space. The clustering results are visualized as transparent grey discs that are outlined by the normal vector points of the point cloud.

5.3.1 Two Cylinders

The first is two cylinders approximately the same size as can be seen in Fig. 5.2(a)–(b). The centerline in the two cylinders is perpendicular and the normal vector space can be seen in Fig. 5.2(c)–(d). The grey transparent circles are the visual representation of the clustering result. The numerical results can be seen in Table 5.1. All error values are calculated by averaging the arc length of every point to the closest cluster.


(a)


(b)


(c)

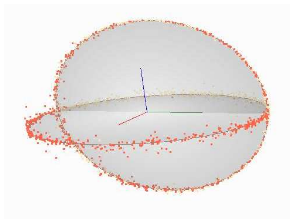

(d)

Figure 5.2: Point cloud of two cylinders with perpendicular center lines, Table 5.1: (a)–(b) show the two cylinders the make up the point cloud. The images are taken from a slightly different angle to help the viewer see the depth; and, (c)–(d) show the normal vector space that was calculated from the original point cloud. The clustering results are seen as the slightly transparent grey circles that fill the rings made by the normal vectors. The X, Y and Z axes are shown with a Red, Green and Blue line, respectively. There are two different angles to help the viewer see the depth in the images.

Table 5.1: Results for the Two Cylinders point cloud, Fig. 5.2.

	A	B	C	D	σ^2	Error
EM	-1.000	0.001	0.000	0.000		
	0.002	0.002	1.000	0.000	0.0001	0.020
EM	-1.000	0.003	-0.007	0.000		
	0.001	0.000	1.000	0.000	0.001	0.034
EM	-1.000	0.000	0.009	0.054		
	0.003	0.005	1.000	0.056	0.005	0.043
Real	-1.000	0.000	0.000	0.000		
	0.000	0.000	1.000	0.000		

5.3.2 Cone and Cylinder

The second example shows a cone and a cylinder with the same center line, meaning the normal vector in the cluster will be the same. The shapes are distinguished by the difference in the D values as can be seen in Table 5.2. This is a good example of the duplex data points used in the clustering process. There is only one cone, but the normal vector space shows two rings for the cone (the top and bottom). There are also two rings for the cylinder, but they overlap leaving the appearance of one center ring.

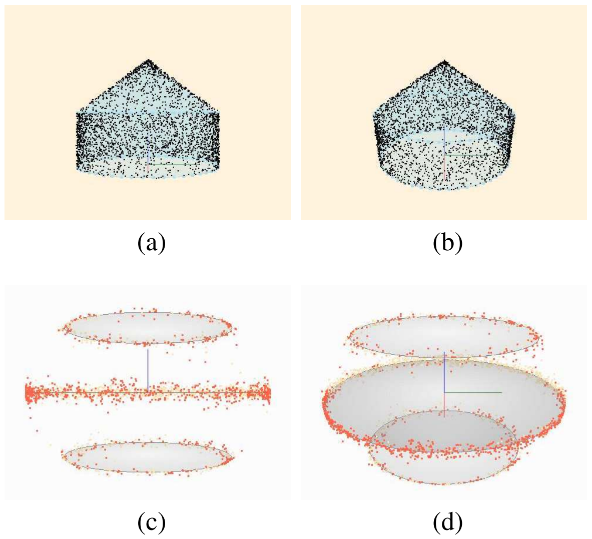


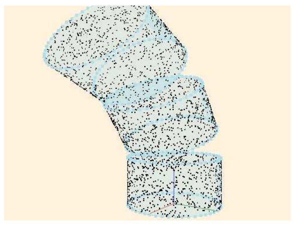
Figure 5.3: Point cloud containing a single cone and cylinder, Table 5.2: (a)–(b) show the cylinder with a cone attached at the top. The radius of the large end of the cone is the same as the radius of the cylinder; and, (c)–(d) show the results of the clustering algorithms. There are three rings visible in the image even though there are only two shapes in the point cloud. This is due to the cone and the positive and negative direction of the normal vector points. The top and bottom ring are mirror images of each other.

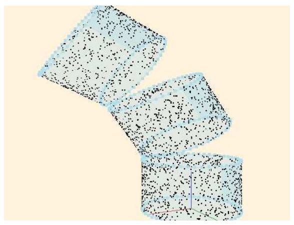
Table 5.2: Results for the Cone and Cylinder point cloud, Fig. 5.3.

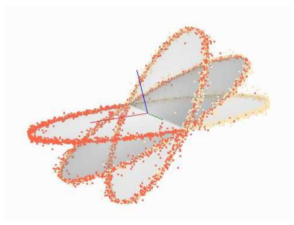
	A	B	C	D	σ^2	Error
EM	0.004	0.004	-1.000	-0.706		
	0.003	0.001	-1.000	0.000	0.0001	0.034
EM	-0.008	0.004	-1.000	-0.708		
	0.000	-0.003	-1.000	0.000	0.001	0.038
EM	0.021	-0.015	-1.000	-0.688		
	0.002	0.019	-1.000	0.083	0.005	0.065
Real	0.000	0.000	-1.000	-0.707		
	0.000	0.000	-1.000	0.000		

5.3.3 Three Cylinders

The next example, Fig. 5.4, shows some of the limitations of the algorithm. Very low noise levels produce good results, but higher noise levels produce worse results. The results at high noise can be very bad based on the initialization of the EM algorithm.


(a)


(b)


(c)

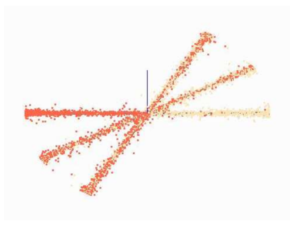

(d)

Figure 5.4: Point cloud of three cylinders rotated about an axis, Table 5.3: (a)–(b) show the point cloud images; and, (c)–(d) show the correct solution. EM has some trouble with this solution, getting it correct $\approx 80\%$ of the time with moderate noise and less with greater noise.

Table 5.3: Results for the Three Cylinders point cloud, Fig. 5.4.

	A	B	C	D	σ^2	Error
EM	0.501	0.002	0.866	0.000		
	-0.002	0.000	-1.000	0.000		
	0.865	0.001	0.502	0.000	0.0001	0.011
EM	0.328	0.014	0.945	-0.147		
	0.018	-0.027	-1.000	-0.001		
	0.863	0.001	0.505	0.000	0.001	0.044
EM	0.350	-0.020	0.937	0.145		
	0.062	-0.016	-0.998	0.059		
	0.872	0.016	0.489	0.084	0.005	0.050
Real	0.500	0.000	0.866	0.000		
	0.000	0.000	-1.000	0.000		
	0.866	0.000	0.500	0.000		

5.3.4 Cone, Cylinder and Plane

The final example is similar to the first, but a plane has been added. When the point cloud has little noise, the correct solution can be found with EM, Table 5.4. With a moderate amount of noise, the clusters have more spread and MDL has a difficult time estimating the correct number of clusters. The correct solution is obtained, but an extra cluster appears along with the correct solution. This is due to the moderate noise in the point cloud and the close proximity of the features. The high levels of noise produce incorrect results. Two of the clusters are correct, but there are four incorrect clusters in the solution. The high levels of noise spread the clusters evenly over the sphere. More clusters are needed in the solution to obtain a good fit for the data.

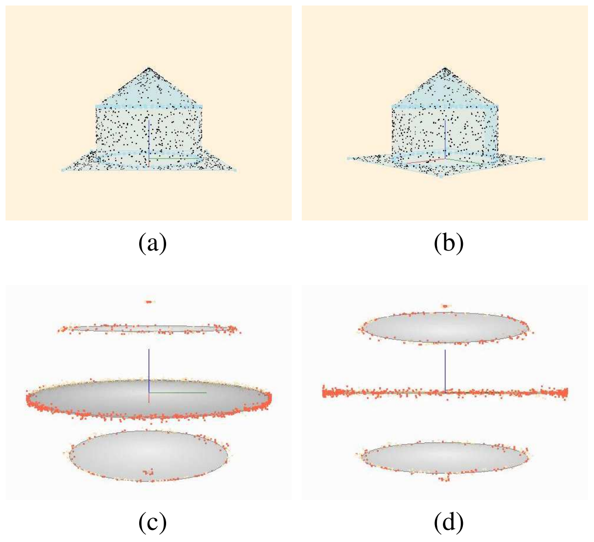


Figure 5.5: Point cloud the three shapes, Cone, Cylinder and Plane, Table 5.4: (a)–(b) show the point cloud images; and, (c)–(d) show the correct solution. EM gets the correct solution at low noise levels but has a lot of trouble with this point cloud when the noise level is high. The clusters are no longer tight leading to incoherent results.

Table 5.4: Results for the Cone, Cylinder and Plane point cloud, Fig. 5.5.

	A	B	C	D	σ^2	Error
EM	0.001	0.016	1.000	1.000		
	0.000	0.001	1.000	0.005		
	−0.003	−0.003	−1.000	−0.713	0.0001	0.008
EM	0.006	−0.003	1.000	0.998		
	0.003	−0.002	1.000	−0.005		
	−0.014	−0.009	−1.000	−0.708		
	−0.449	−0.893	−0.029	0.0261	0.001	0.027
EM	0.005	0.023	1.000	0.015		
	0.011	0.020	1.000	−0.022		
	0.013	0.016	−1.000	−0.712		
	−0.754	−0.657	−0.007	0.005		
	0.865	−0.502	0.017	−0.016		
	−0.884	−0.312	−0.349	−0.371	0.005	0.033
Real	0.000	0.000	1.000	1.000		
	0.000	0.000	1.000	0.000		
	0.000	0.000	−1.000	−0.707		

CHAPTER 6

MCMC Clustering of Point Clouds

In this chapter, we use Reversible Jump Markov Chain Monte Carlo (RJ-MCMC) with the likelihood function from Chapter 4 to determine quantity and orientation of unique shapes in the normal vector space. MCMC works by utilizing a random walk from an initial state [27, 28] to locate the global maxima (or minima depending on the situation). Each sweep of the algorithm calculates the likelihood ratio of the new state versus the old state. The likelihood ratio is compared to a uniform random variable and accepted if the random variable is less than the ratio. This is a very important distinction when compared to the EM algorithm [23] because a new state could be accepted even if the likelihood is lower than the current state. This gives MCMC the ability to escape local maxima that EM does not have, meaning MCMC initialization is not as important as EM initialization.

Reversible Jump MCMC (RJ-MCMC) [29] is very similar to MCMC but has the ability to switch between discrete dimensions. This is used to determine the system order by increasing or decreasing the number of clusters. The likelihood ratio is still used with two provisions. A Jacobian is calculated to account for the change in dimensions and priors for model order are used protect against over fitting the data.

6.1 Normal Space Inference Problem using the MCMC Algorithm

The MCMC algorithm for our clustering is similar to the EM algorithm since the log-likelihood, Eq. (6.1), is calculated the same way for both,

$$\ln p(\mathbf{Y}|\mathbf{\Theta}) = \sum_{n=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k p(\mathbf{y}_n|\boldsymbol{\theta}_k) \right), \tag{6.1}$$

where $p(\mathbf{Y}|\boldsymbol{\Theta})$ is the probability of data set $\mathbf{Y}$ given parameter set $\boldsymbol{\Theta}$. The parameter set, $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_K\}$, consists of K clusters with each cluster having the parameters $\boldsymbol{\theta}_k = (\boldsymbol{\mu}_k, \Sigma_k, \pi_k)$. $\boldsymbol{\mu}_k$ is the cluster mean, Σ_k is the cluster variance, and π_k is the weight of each cluster. $\pi_k \in (0,1)(k = 1, 2, \ldots, K)$ which is subject to summation $\sum_{k=1}^{K} \pi_k = 1$. The probability of point $\mathbf{y}_n$ belonging to cluster $\boldsymbol{\theta}_k$ is

$$p(\mathbf{y}_n|\boldsymbol{\theta}_k) = \frac{\exp\left\{\frac{D_{arc}(\mathbf{y}_n,\boldsymbol{\mu}_k)\Sigma_k^{-1}D_{arc}(\mathbf{y}_n,\boldsymbol{\mu}_k)}{-2}\right\}}{(2\pi)^{\frac{3}{2}}|\Sigma_k|^{\frac{1}{2}}}. \tag{6.2}$$

where $D_{arc}(\mathbf{y}_n, \boldsymbol{\mu}_k)$ is the one dimensional arc length and Σ_k is the one dimensional covariance matrix (variance value). The probability of point $\mathbf{y}_n$ given the data set $\boldsymbol{\Theta}$ is found by summing over variable k,

$$p(\mathbf{y}_n|\boldsymbol{\Theta}) = \sum_{k=1}^{K} \pi_k p(\mathbf{y}_n|\boldsymbol{\theta}_k). \tag{6.3}$$

$\boldsymbol{\Theta}$ denotes the current parameters and $\boldsymbol{\Theta}'$ denotes the proposal parameters. The apostrophe, $'$, signifies the proposal state for the entire MCMC algorithm. $\pi_{pdf}(\boldsymbol{\Theta})$ is the pdf of the current set of parameters and $\pi_{pdf}(\boldsymbol{\Theta}')$ is the pdf of the proposed set of parameters. The proposed parameters are chosen with density $q(\boldsymbol{\Theta}'|\boldsymbol{\Theta})$, but they need not depend on the current parameters, $\boldsymbol{\Theta}$. It does not matter in our case since the densities are equal, $q(\boldsymbol{\Theta}'|\boldsymbol{\Theta}) = q(\boldsymbol{\Theta}|\boldsymbol{\Theta}')$, and will cancel each other. The posterior distribution of parameters given the observed data, $\mathbf{Y}$, is used for the target distribution, $\pi_{pdf}(\boldsymbol{\Theta})$, in Eq. (6.4),

$$\pi_{pdf}(\boldsymbol{\Theta}) = p(\boldsymbol{\Theta}|\mathbf{Y}) = \frac{p(\mathbf{Y}|\boldsymbol{\Theta})p(\boldsymbol{\Theta})}{p(\mathbf{Y})}, \tag{6.4}$$

with $\pi_{pdf}(\boldsymbol{\Theta}')$ calculated in the same manner. The MCMC acceptance ratio, α, is then given by Eq. (6.5),

$$\alpha(\boldsymbol{\Theta}, \boldsymbol{\Theta}') = min\left\{1, \frac{\pi_{pdf}(\boldsymbol{\Theta}')q(\boldsymbol{\Theta}|\boldsymbol{\Theta}')}{\pi_{pdf}(\boldsymbol{\Theta})q(\boldsymbol{\Theta}'|\boldsymbol{\Theta})}\left|\frac{\partial(\boldsymbol{\Theta}')}{\partial(\boldsymbol{\Theta}, \mathbf{u})}\right|\right\}. \tag{6.5}$$

When the posterior probabilities are substituted in Eq. (6.5), α is reduced to the ratio of likelihood values, Eq. (6.6),

$$\alpha(\boldsymbol{\Theta}, \boldsymbol{\Theta}') = min\left\{1, \frac{p(\boldsymbol{\Theta}')L(\boldsymbol{\Theta}'|\mathbf{Y})}{p(\boldsymbol{\Theta})L(\boldsymbol{\Theta}|\mathbf{Y})}\left|\frac{\partial(\boldsymbol{\Theta}')}{\partial(\boldsymbol{\Theta}, \mathbf{u})}\right|\right\}. \tag{6.6}$$

```
MCMC Algorithm
1: Initialize MCMC with 1 cluster
2: repeat
3:    Choose a cluster randomly from the set, $\boldsymbol{\theta}_k$, and available move
   type:  Update, Birth/Death, or Split/Combine.
4:    Calculate the new parameter set, $\boldsymbol{\Theta}'$.
5:    Calculate Acceptance Ratio, $\alpha(\boldsymbol{\Theta}, \boldsymbol{\Theta}')$.
6:    if $\alpha(\boldsymbol{\Theta}, \boldsymbol{\Theta}') > \mathcal{U}(0,1)$ then
7:       Accept new parameter set
8:       $\boldsymbol{\Theta} = \boldsymbol{\Theta}'$
9:    else
10:      Keep the same parameter set
11:      $\boldsymbol{\Theta} = \boldsymbol{\Theta}$
12:   end if
13: until Maximum sweep number met
14: Calculate the frequency of acceptances per cluster.
15: Calculate average parameters for most popular cluster quantity.
```

Since we are using RJ-MCMC [30, 29] to estimate the model order, [26], there are several different types of update steps that are utilized: Simple Update, Birth/Death, or Split/Combine. One cluster is selected during each cycle (or sweep) of the RJ-MCMC algorithm along with a move type to adjust the cluster parameters. The parameters are adjusted according to the move type and the acceptance ratio is calculated. The move is accepted if α is greater than a uniform random number, $\mathcal{U}(0, 1)$.

6.1.1 Simple Update

The Update step will change the parameters of a cluster in one of two distinct ways. The first will create a random vector, $\mathbf{u}$,

$$\mathbf{u} = \langle A_{\mathbf{u}}, B_{\mathbf{u}}, C_{\mathbf{u}}, D_{\mathbf{u}} \rangle, \tag{6.7}$$

where $\mathbf{u}$ is created using 4 zero mean Gaussian random numbers. $\mathbf{u}$ is added to the current cluster, Eq. (6.8),

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + \mathbf{u} = \langle A', B', C', D' \rangle. \tag{6.8}$$

The new cluster is constrained by Eq. (6.9),

$$\begin{gathered} ||\langle A', B', C' \rangle|| = 1, \\ ||D'|| < 1. \end{gathered} \tag{6.9}$$

The second way to update a cluster is to just accept the random vector, $\mathbf{u}$ as the new cluster, Eq. (6.10).

$$\boldsymbol{\theta}' = \mathbf{u}. \tag{6.10}$$

This can be considered a new global cluster that can appear anywhere. This cluster does not depend on the previous cluster making it one of the most important parts of MCMC. The global parameter update is very useful for escaping local minima that can easily trap the EM algorithm. All new clusters are constrained by Eq. (6.9).

6.1.2 Birth/Death

The Birth and Death moves are slightly more complex than the Update move because the number of clusters changes and the Jacobian, Eq. (6.11),

$$\mathcal{J} = \frac{\partial(\boldsymbol{\Theta}')}{\partial(\boldsymbol{\Theta}, \mathbf{u})}, \tag{6.11}$$

has to be calculated. There are two types of Birth moves that are very similar to the Update moves. The first type will copy the parameters of an existing cluster and add them to a random vector, Eq. (6.8). The second type of Birth move will create a globally random cluster, Eq. (6.10). The Death move is slightly easier. A cluster from the list is chosen at random and removed. The Jacobian for the Death move is the inverse of the Jacobian for the Birth move.

Table 6.1: MCMC move types. Basic move types only update a single cluster while more complex types have the ability to update two clusters simultaneously.

Move Type	Number of Clusters Affected	Number of Clusters Changed
Update	1	No Change
Birth	1	$+1$
Death	1	-1
Split	2	$+1$
Combine	2	-1

6.1.3 Split/Combine

Both the Split and Combine moves are slightly more complicated than the Birth/Death moves. For Split, a cluster is chosen at random, $\boldsymbol{\theta}$, and split into two different clusters using random vector $\mathbf{u}$. The two new clusters are updates from the current cluster, Eq. (6.12),

$$\begin{aligned}\boldsymbol{\theta}'_1 &= \boldsymbol{\theta} + \tfrac{1}{2}\mathbf{u},\\ \boldsymbol{\theta}'_2 &= \boldsymbol{\theta} - \tfrac{1}{2}\mathbf{u},\end{aligned} \tag{6.12}$$

where $\boldsymbol{\theta}'_1$ and $\boldsymbol{\theta}'_2$ are the two clusters created from $\boldsymbol{\theta}$ and $\mathbf{u}$.

Combine is performed by randomly picking a cluster, $\boldsymbol{\theta}_1$, and averaging it with the nearest cluster, $\boldsymbol{\theta}_2$, to get the new cluster, $\boldsymbol{\theta}'$, Eq. (6.13),

$$\boldsymbol{\theta}' = \frac{\boldsymbol{\theta}_1 + \boldsymbol{\theta}_2}{2}. \tag{6.13}$$

The Jacobian of the Combine move is the inverse of the Split move.

Both the Combine and Death moves can only be performed when the number of clusters is greater than the minimum number. The Birth and Split moves can only be performed when the number of clusters is less than the maximum. The different move types are illustrated in Table 6.1.

6.2 Simulation Results

We have included a number of examples to help in the visualization of this algorithm. We are also including results of each data set found using the Expectation Maximization (EM) algorithm [31] combined with Minimum Description Length (MDL), Chapter 5, to determine system order for comparison. We intend to show some simple examples to aid with the new cluster idea we are using. The simple point clouds will show very similar results using both the RJ-MCMC and EM algorithm. We will show some slightly complicated examples where EM will occasionally find a local maxima instead of the correct solution. We will also show some more complicated examples that will show how robust the RJ-MCMC algorithm can be. The correct solution of some complicated examples can rarely be found using EM.

6.2.1 Algorithm Initialization

Initialization for the RJ-MCMC algorithm is very easy since it is not sensitive to initialization. Our initialization is done by creating a single cluster and picking $\langle A, B, C, D\rangle$ values at random. The number of clusters will quickly increase and start locating the correct solution.

We found EM to be much more sensitive to initialization than RJ-MCMC. This led to a simple initialization done by creating a large number of clusters with random values. The likelihood value is calculated and the clusters are randomly initialized again. This process is repeated a few hundred times and the configuration with the highest likelihood is used as the initialization for the algorithm. This solution is not elegant but works well for this application. Simple point clouds do not need the initialization as much as more complicated point clouds. However, even with initialization, the correct solution is not guaranteed using the EM algorithm.

6.2.2 Simple Example: Two Cylinders Point Cloud

We have included some simulation results to aid in the visualization of this algorithm. The first is two cylinders approximately the same size as can be seen in Fig. 6.1(a)–(b). The centerline in the two cylinders is perpendicular and the normal vector space can be seen in Fig. 6.1(c)–(d). The grey transparent circles are the visual representation of the clustering result. The numerical results can be seen in Table 6.2. All error values are calculated by averaging the arc length of every point to the closest cluster.

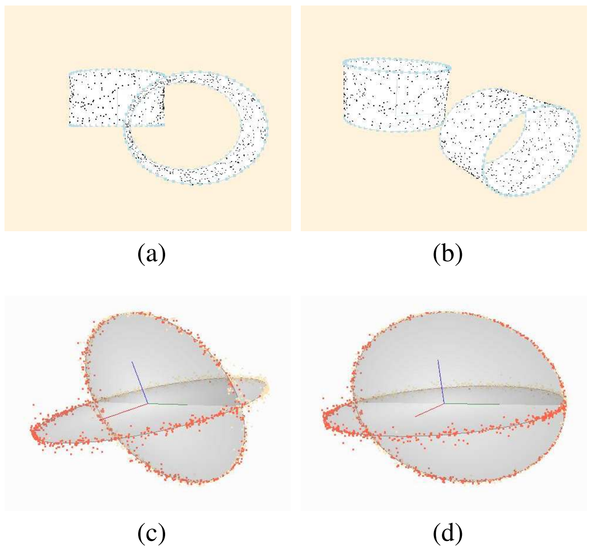


Figure 6.1: Point cloud of two cylinders with perpendicular center lines, Table 6.2: (a)–(b) show the two cylinders the make up the point cloud. The images are taken from a slightly different angle to help the viewer see the depth; and, (c)–(d) show the normal vector space that was calculated from the original point cloud. The clustering results are seen as the slightly transparent grey circles that fill the rings made by the normal vectors. The X, Y and Z axes are shown with a Red, Green and Blue line, respectively. There are two different angles to help the viewer see the depth in the images.

Table 6.2: Results for the Two Cylinders point cloud, Fig. 6.1. This example was also seen in Chapter 5 and is simple enough that both EM and RJ-MCMC can easily obtain a good estimate of the correct solution.

	A	B	C	D	Error
EM	-1.000	0.001	0.000	0.000	
	0.002	0.002	1.000	0.000	0.0205
RJ-MCMC	-1.000	-0.004	0.005	0.000	
	0.003	-0.001	1.000	-0.003	0.0170
Correct	-1.000	0.000	0.000	0.000	
	0.000	0.000	1.000	0.000	

6.2.3 Simple Example: Cone and Cylinder Point Cloud

The second simple point cloud example shows a cone and a cylinder with the same center line, meaning the normal vector in the cluster will be the same. The shapes are distinguished by the difference in the D values as can be seen in Table 6.3. This is a good example of the duplex data points used in the clustering process. There is only one cone, but the normal vector space shows two rings for the cone (the top and bottom).

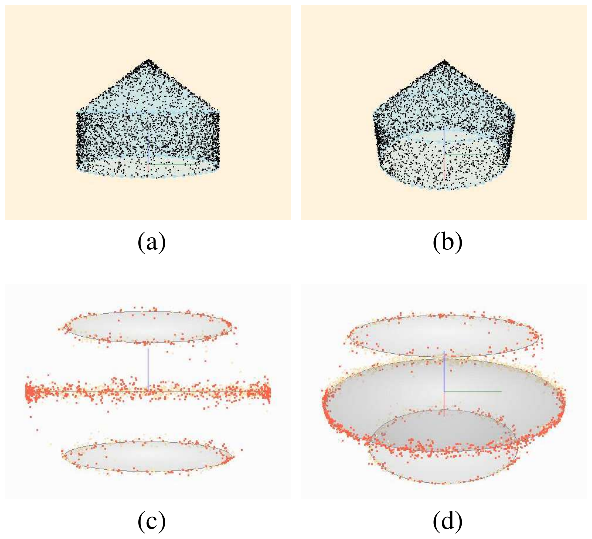


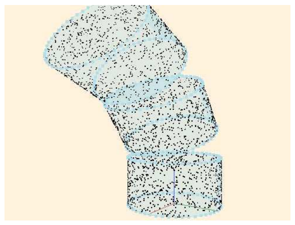
Figure 6.2: Point cloud containing a single cone and cylinder, Table 6.3: (a)–(b) show the cylinder with a cone attached at the top. The radius of the large end of the cone is the same as the radius of the cylinder; and, (c)–(d) show the results of the clustering algorithms. There are three rings visible in the image even though there are only two shapes in the point cloud. This is due to the cone and the positive and negative direction of the normal vector points. The top and bottom ring are mirror images of each other.

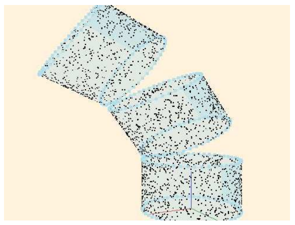
Table 6.3: Results for the Cone and Cylinder point cloud, Fig. 6.2.

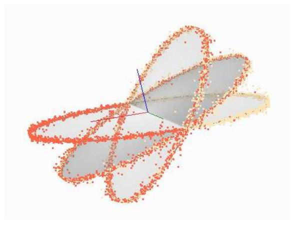
	A	B	C	D	Error
EM	0.004	0.004	-1.000	-0.706	
	0.003	0.001	-1.000	0.000	0.0339
RJ-MCMC	0.009	0.008	-1.000	-0.704	
	0.001	-0.004	-1.000	0.000	0.0317
Correct	0.000	0.000	-1.000	-0.707	
	0.000	0.000	-1.000	0.000	

6.2.4 Simple Example: Three Cylinders Point Cloud

This point cloud of cylinders is a good example of odd overlapping that can occur in the normal vector space. Visually, the clusters appear to be very well separated, but this point cloud has some local maxima that can fool EM. This is the first point cloud where the differences between EM and RJ-MCMC start to appear. RJ-MCMC will find the correct solution most of the time even with different levels of noise, while EM can struggle to find the solution.


(a)


(b)


(c)

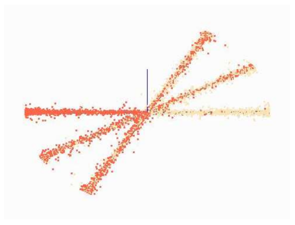

(d)

Figure 6.3: Point cloud of three cylinders rotated about an axis, Table 6.4: (a)–(b) show the point cloud images; and, (c)–(d) show the correct solution. MCMC will find the correct solution most of the time with different levels of noise.

Table 6.4: Results for the Three Cylinders point cloud, Fig. 6.3.

	A	B	C	D	Error
EM	0.501	0.002	0.866	0.000	
	-0.002	0.000	-1.000	0.000	
	0.865	0.001	0.502	0.000	0.011
RJ-MCMC	0.501	-0.001	0.865	0.001	
	0.002	-0.007	-1.000	-0.003	
	0.866	-0.001	0.500	-0.003	0.009
Real	0.500	0.000	0.866	0.000	
	0.000	0.000	-1.000	0.000	
	0.866	0.000	0.500	0.000	

6.2.5 Complex Example: House Point Cloud

The first more complex case is a house that is made up of five planes, Fig. 6.4, meaning there are five cluster pairs. Four of the cluster pairs can be though of as a plane rotated about an axis and oriented four different directions. Although there is plenty of separation, the solution can appear to be a single cylinder instead of four planes, Fig. 6.4(c)–(d). EM will find this solution most of the time (without long initialization) and the error is three times larger than the correct solution, Table 6.5. The error in the cluster data is completely incorrect due to the number of clusters and the D values.

Table 6.5: Results for a house point cloud, Fig. 6.4.

	A	B	C	D	Error
EM	-1.000	-0.000	0.001	0.000	
Bad Results	0.001	0.001	-1.000	0.000	0.0109
RJ-MCMC	-1.000	-0.000	0.009	1.000	
	-0.002	1.000	0.004	1.000	
	-0.005	-0.008	-1.000	1.000	
	-0.005	-0.705	-0.709	-1.000	
	0.001	0.699	-0.718	1.000	0.0035
Correct	-1.000	0.000	0.000	1.000	
	0.000	1.000	0.000	1.000	
	0.000	0.000	-1.000	1.000	
	0.000	-0.707	-0.707	-1.000	
	0.000	0.707	-0.707	1.000	

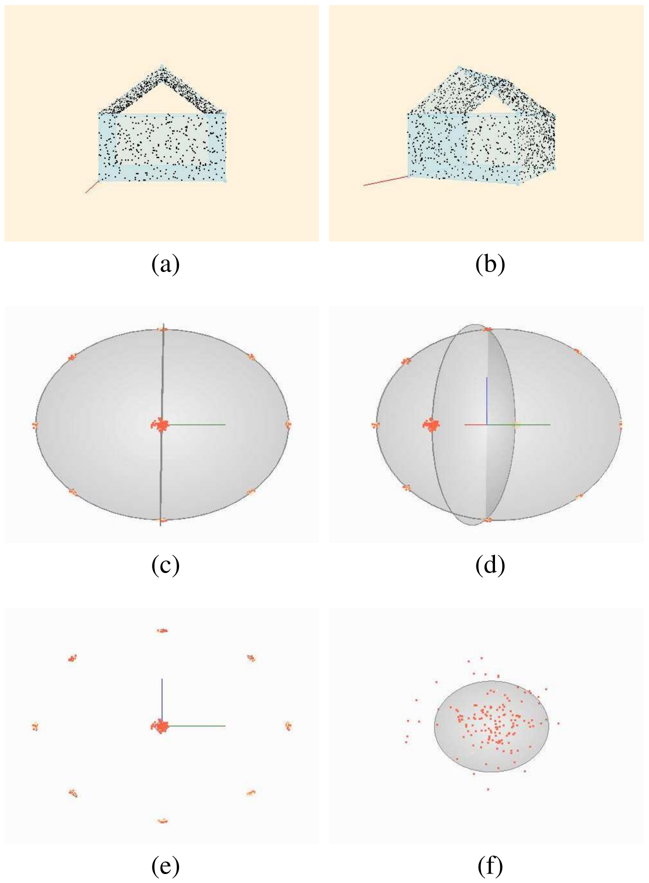


Figure 6.4: Point cloud of a house consisting of planes, Table 6.5: (a)–(b) are two images of the house. It is made up of four walls, a floor and a roof. There are five cluster pairs for a total of ten clusters in this point cloud; (c)–(d) are and example of the incorrect solution to this problem. The solution has two large circles which indicates two cylinders in the point cloud. This is a good fit mathematically, but not what is wanted. EM has a difficult time finding the correct solution for this point cloud; (e) is the correct solution for this point cloud. The solution consists of several small cluster pairs that fit tightly on the points; and, (f) shows a closeup of one of the clusters. The other clusters in the solution are similar in appearance.

6.2.6 Complex Example: Cone, Cylinder and Plane Point Cloud

The final complex point cloud example has all three shapes at once. RJ-MCMC does a good job on this point cloud, but EM can obtain an incorrect solution. We have included two different solutions for EM along with the RJ-MCMC solution for comparison. The incorrect EM solution in this case has an extra cluster. Notice that the extra cluster helps to reduce the error measurement. It should also be noted that the incorrect EM solution was obtained even though the initialization was used. This is a good example of the robustness of RJ-MCMC.

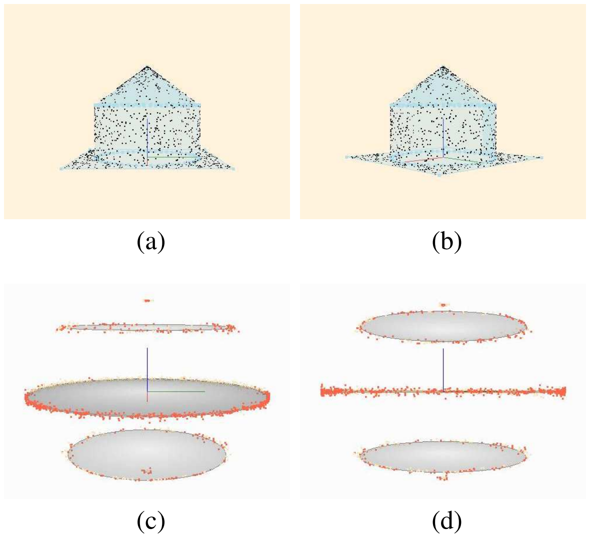


Figure 6.5: Point cloud the three shapes, Cone, Cylinder and Plane, Table 6.6: (a)–(b) show the point cloud images; and, (c)–(d) show the correct solution. EM gets the correct solution at low noise levels but has a lot of trouble with this point cloud when the noise level is high. The clusters are no longer tight leading to incoherent results.

Table 6.6: Results for the Cone, Cylinder and Plane point cloud, Fig. 6.5. A good and bad EM result is included along with the RJ-MCMC result for this point cloud. This point cloud demonstrates the robustness of RJ-MCMC over EM.

	A	B	C	D	Error
EM	0.001	0.016	1.000	1.000	
	0.000	0.001	1.000	0.005	
Good Results	−0.003	−0.003	−1.000	−0.713	0.014
EM	0.006	−0.003	1.000	0.998	
	0.003	−0.002	1.000	−0.005	
	−0.014	−0.009	−1.000	−0.708	
Bad Results	−0.449	−0.893	−0.029	0.0261	0.011
RJ-MCMC	−0.020	0.007	1.000	0.999	
	0.008	0.006	1.000	−0.001	
Good Results	−0.013	−0.005	−1.000	−0.711	0.012
Real	0.000	0.000	1.000	1.000	
	0.000	0.000	1.000	0.000	
	0.000	0.000	−1.000	−0.707	

6.2.7 Robust Example: Coupling with Cylinders Point Cloud

A good example of the robust cases that can be handled by MCMC and not EM is in Fig. 6.6. There is a simple pipe coupling that consist of a couple of cylinders and a cone along with some other cylinders in the point cloud. Each of the cylinders is oriented differently which creates a lot of overlap in the normal vector space. In order to get the correct solution with EM, the initialization process has to be long leaving little work for EM. MCMC can find the correct solution, but the number of sweeps will need to be higher, ≈ 200000, meaning much computation time.

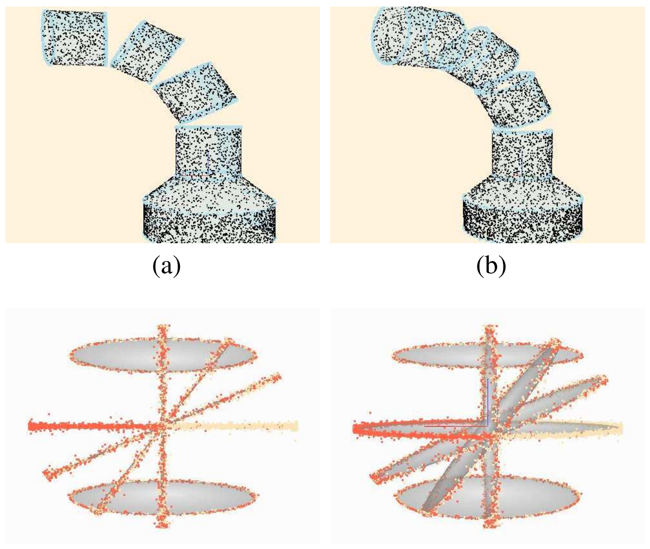


Figure 6.6: Point cloud of a coupling with three cylinders, Table 6.7: (a)–(b) are images of the point cloud; and, (c)–(d) show the results of clustering. These results are very hard to get with EM but much easier with MCMC.

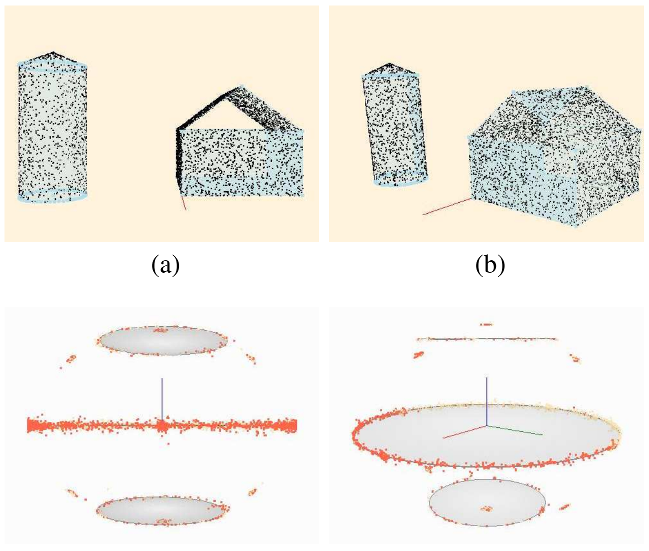
Table 6.7: Results for the coupling and cylinders, Fig. 6.6. The EM algorithm produces very bad results while the RJ-MCMC algorithm produces good results.

	A	B	C	D	Error
EM	0.922	0.006	0.388	0.321	
	-0.001	-0.001	-1.000	0.000	
	0.269	0.014	0.963	0.216	
Very Bad	-0.424	0.019	0.906	-0.407	
Results	0.869	0.001	0.496	0.001	0.0206
RJ-MCMC	1.000	-0.001	0.003	0.004	
	-0.003	-0.002	-1.000	0.000	
	0.004	0.000	1.000	-0.706	
Correct	-0.502	0.001	-0.865	-0.003	
Results	-0.855	-0.002	-0.519	-0.001	0.0054
Correct	1.000	0.000	0.000	0.000	
	0.000	0.000	-1.000	0.000	
	0.000	0.000	1.000	-0.707	
	-0.500	0.000	-0.866	0.000	
	-0.866	0.000	-0.500	0.000	

6.2.8 Robust Example: House and Silo Point Cloud

The example of a house with a silo structure, Fig. 6.7, is another example of the robustness of MCMC. There is a lot of overlap in the normal vector space. Two of the plane clusters lie in the middle of the cylinder cluster and are difficult to see. Another reason this is difficult is the small size of the conical roof for the silo. This means a much small cluster since the area is small comparatively (point density in the point cloud is the same for all shapes).

Watching MCMC work on this point cloud is interesting since it usually finds the incorrect solution first. This incorrect solution resembles the incorrect solution in Fig. 6.4. The number of clusters will eventually increase and the incorrect clusters will disappear when a better solution emerges.



(d)

Figure 6.7: House and Silo point cloud, Table 6.8: (a)–(b) show the house from Fig. 6.4 with a silo structure nearby; and, (c)–(d) show the results of clustering with RJ-MCMC.

Table 6.8: Results for clustering the house and silo, Fig. 6.7. Only RJ-MCMC results are included since EM will never obtain good clustering for this complex point cloud.

	A	B	C	D	Error
MCMC	0.001	-0.001	-1.000	0.004	
	0.009	0.006	1.000	0.871	
	-0.001	-0.002	-1.000	1.000	
	0.002	0.711	-0.703	1.000	
	-1.000	0.004	0.004	-1.000	
	0.000	-0.716	-0.698	-1.000	
Good Results	0.008	1.000	-0.010	-1.000	0.0023
Correct	0.000	0.000	-1.000	0.000	
	0.000	0.000	1.000	0.866	
	0.000	0.000	-1.000	1.000	
	0.000	0.707	-0.707	1.000	
	-1.000	0.000	0.000	-1.000	
	0.000	-0.707	-0.707	-1.000	
	0.000	1.000	0.000	-1.000	

6.2.9 Limitations of clustering

The EM algorithm can perform clustering very fast and accurately as long as the point cloud is not too complicated. As seen in this Chapter, the point clouds can get complicated very quickly. There are also several different situations that do not look complicated but can be.

When the EM algorithm does find the correct solution, the results are usually closer to the ground truth than the MCMC results. This may seem counterintuitive since the error values are usually larger than the MCMC error values. This is due to the cluster pair that is used in the clustering process. The EM algorithm will find the correct D value for cylinders, $D = 0$, but the MCMC algorithm will find a different D value, $|D| > 0$. This will lead to a slightly smaller error for the MCMC even though the EM results are technically better. The robustness of MCMC more than makes up for this small difference in error values.

CHAPTER 7

Conclusions and Future Research

In this thesis, we presented two algorithms for clustering simple shapes in point clouds. In Chapter 3 we used the EM algorithm to cluster planes in a point cloud. The EM algorithm works very well for this situation since planes in the point cloud form small clusters in the normal vector space. The EM algorithm used in Chapter 3 did not require any modification and was able to estimate the system order well with MDL.

In order to expand the clustering ability of EM, a new likelihood function was derived in Chapter 4. When the EM algorithm was modified for use with the new likelihood function, both cones and cylinders could be clustered along with planes. We are very proud of the new likelihood function since it can be used with all three shapes simultaneously. The EM algorithm was modified in Chapter 5 to be used with the new likelihood function.

We found the algorithm to work well with simple point clouds but had problems with the complexity of the point cloud increased. There were two problem areas that needed work. The first problem was the variance of the clusters being modeled. The EM algorithm was able to model a simple plane as two different shapes: a plane with a small variance or a cylinder with a large variance. This problem was corrected empirically by studying the variance of different shapes in the point cloud. It was found that the variance of a cluster changed with the shape of the cluster. As a cone changed from tubular to planar, the variance of the cluster had a smooth exponential decrease. This was found to be the case regardless of the noise in the original point cloud. This exponential function is used instead of calculating the variance for each cluster. The second problem with the EM algorithm was the initialization. In complex point clouds, there are several local maxima in which the

EM algorithm can be trapped. The solution to this problem was to create an initialization algorithm that would attempt to find a good starting point for EM. This solution is not mathematically pretty, but helped get the correct model in complex situations.

Once we had the new likelihood function, we used it to modify the MCMC algorithm in Chapter 6. Using the reversible jump form of the MCMC algorithm, we were able to very accurately model the shapes in point clouds. The MCMC algorithm worked much better than the EM algorithm. The MCMC algorithm does not need the complex initialization that EM uses. Using the MCMC algorithm, we were able to model complex point clouds that EM would never model correctly. The downside of MCMC is the amount of computation required. Since MCMC updates by randomly picking a new solution, it is possible that the correct solution may take a while to find. There are several solutions to this dilemma though. A complex point cloud can be segmented into several smaller point clouds for clustering. Newer multicore processors can also be used to increase the speed of algorithms. Although MCMC can be slower than EM, MCMC seems to be the algorithm capable of producing the best results.

7.1 Limitations

The algorithms we have discussed have several limitations. Normal estimation of unorganized point clouds is one of the major limitations. This process depends on the smoothness of the object, the sampling density, and the noise of the input point cloud. When a point cloud is created using a laser ranging system, the noise level is usually very small. There are situations where this is untrue, such as a laser ranging system attached to an airplane or a point cloud of a very busy surface such as a tree. However, most point clouds created with laser ranging create very accurately sampled surfaces with small amounts of noise. The technology has also advanced enough to create very densely sampled surfaces with millions of points.

Another limitation is the small surface area of the normal vector space. As we have shown, the normal vector space can lead to overlap of shape clusters. This can create many problems for EM. While MCMC handles this situation well, the processing time can be immense. We were unable to find a good solution to this problem. One solution could be to limit the size of the point clouds used as input to the clustering algorithm by segmenting into several smaller point clouds. This could work, but several different segments of the point cloud would probably contain the same shapes, meaning clustering the same shapes multiple times. Another possible solution would be to down sample the point cloud. Down sampling could potentially lead to excluding smaller shapes in the point cloud as noise.

The initialization of the EM algorithm is also a limitation. While it worked well in the beginning for the simple plane clustering, the algorithm needed much adjustment to get good results for more complex shapes. In hindsight, we worked on EM too long trying to get good results. It might have been better to attempt different clustering algorithms once we had derived our new likelihood function. The initialization we used in this paper is very similar to running MCMC. Complex point clouds required long initialization which could not guarantee good results for EM. We are very pleased with the results of MCMC though.

7.2 Future Research

Future research on this topic will involve taking the clustering results of the normal vector space and finding the locations and boundaries of shapes in the real world space. The first step is to determine how many real world shapes belong to each normal vector cluster. This is necessary since multiple shapes can be mapped to the same location. A simple example of three planes sharing the same normal vector can be seen in Fig. 7.1. There are three different planes, but will be seen as as one unique plane in the normal vector space. This is not a problem since the points belonging to these planes can easily be segmented after the first clustering. Once segmented, a one dimensional clustering algorithm can be performed to find the number of planes sharing the same normal vector and the real world position of each plane. A similar process can be done for cylinders [32] and cones and will be expedited by using the results from the first clustering.

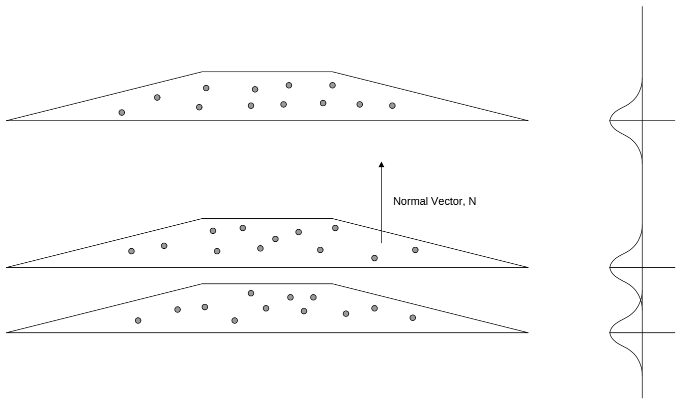


Figure 7.1: Three planes in the real world that share the same normal vector. $1D$ clustering in the real world space will determine both the quantity and location of the planes in relation to an arbitrary point in space. The line on the right shows the Gaussian spread from noise in the point cloud.

Once the number of shapes and location is known, the boundary of each shape needs to be determined. It would also be possible at this point to include points that were excluded due to the condition number of the normal vector estimation. It seems this would be a very difficult problem. A balance would need to be found between a smooth boundary and a more accurate boundary. We assume at this time a line fitting algorithm might be used to estimate the boundary. Once the boundary of each shape is known, a simple model of the shapes would be known and modeling would be complete.

BIBLIOGRAPHY

[1] S. Gumhold, X. Wang, and R. Macleod, “Feature Extraction from Point Clouds,” in *Proceedings of the Tenth International Meshing Roundtable*, pp. 293–305, Sandia National Laboratories, October 2001.

[2] Marco Attene and Bianca Falcidieno and Jarek Rossignac and Michela Spagnuolo, “Sharpen&Bend: Recovering Curved Sharp Edges in Triangle Meshes Produced by Feature-Insensitive Sampling,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, pp. 181–192, March 2005.

[3] C. Tang and G. Medioni, “Inference of Integrated Surface, Curve and Junction Descriptions from Sparse 3D Data,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 1205–1223, November 1998.

[4] G. Guy and G. Medioni, “Inference of Surfaces, 3D Curves, and Junctions from Sparse, Noisy, 3D Data,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 1265–1277, November 1997.

[5] P. Csakany and A. Wallace, “Representation and classification of 3-d objects,” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 33, pp. 638–647, Aug. 2003.

[6] M. Pauly, L. P. Kobbelt, and M. Gross, “Point-based multiscale surface representation,” *ACM Trans. Graph.*, vol. 25, no. 2, pp. 177–193, 2006.

[7] F. Pedersini, A. Sarti, and S. Tubaro, “Visible surface reconstruction with accurate localization of object boundaries,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 10, pp. 278–292, Mar 2000.

[8] A. Jalba and J. Roerdink, “Efficient surface reconstruction using generalized coulomb potentials,” *Visualization and Computer Graphics, IEEE Transactions on*, vol. 13, pp. 1512–1519, Nov.-Dec. 2007.

[9] M. Attene, B. Falcidieno, J. Rossignac, and M. Spagnuolo, “Edge-sharpener: recovering sharp features in triangulations of non-adaptively re-meshed surfaces,” in *SGP ’03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, (Aire-la-Ville, Switzerland, Switzerland), pp. 62–69, Eurographics Association, 2003.

[10] W. Zhao, D. Nister, and S. Hsu, “Alignment of continuous video onto 3d point clouds,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1305–1318, August 2005.

[11] Z. Lu, S. Baek, and S. Lee, “Robust 3d line extraction from stereo point clouds,” in *RAM*, pp. 1–5, 2008.

[12] J. Huang and C.-H. Menq, “Automatic data segmentation for geometric feature extraction from unorganized 3-d coordinate points,” *Robotics and Automation, IEEE Transactions on*, vol. 17, pp. 268–279, June 2001.

[13] J. Barhak and A. Fischer, “Parameterization and reconstruction from 3d scattered points based on neural network and pde techniques,” *Visualization and Computer Graphics, IEEE Transactions on*, vol. 7, pp. 1–16, Jan-Mar 2001.

[14] C. Dietrich, C. Scheidegger, J. Schreiner, J. Comba, L. Nedel, and C. Silva, “Edge transformations for improving mesh quality of marching cubes,” *Visualization and Computer Graphics, IEEE Transactions on*, vol. 15, pp. 150–159, Jan.-Feb. 2009.

[15] X. Renbo, L. Weijun, and W. Yuechao, “A robust and topological correct marching cube algorithm without look-up table,” in *Computer and Information Technology, 2005. CIT 2005. The Fifth International Conference on*, pp. 565–569, Sept. 2005.

[16] H. Woo, E. Kang, S. Wang, and K. H. Lee, “A new segmentation method for point cloud data,” *International Journal of Machine Tools Manufacture*, vol. 42, pp. 167–178, January 2002.

[17] W. Zou and X. Ye, “Multi-resolution hierarchical point cloud segmenting,” *Computer and Computational Sciences, 2007. IMSCCS 2007. Second International Multi-Symposiums on*, pp. 137–143, Aug. 2007.

[18] R. Unnikrishnan and M. Hebert, “Robust extraction of multiple structures from non-uniformly sampled data,” *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 2, pp. 1322–1329 vol.2, Oct. 2003.

[19] P. Gotardo, K. Boyer, O. Bellon, and L. Silva, “Robust extraction of planar and quadric surfaces from range images,” *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 2, pp. 216–219 Vol.2, Aug. 2004.

[20] S. R. Lach and J. P. Kerekes, “Robust extraction of exterior building boundaries from topographic lidar data,” *Geoscience and Remote Sensing Symposium, 2008. IGARSS 2008. IEEE International*, vol. 2, pp. II–85–II–88, July 2008.

[21] G. Vosselman and B. G. H. Gorte and G. Sithole and T. Rabbani, “Recognising Structure in Laser Scanner Point Clouds,” in *International Archives of Photogrammetry*, pp. 33–38, Remote Sensing and Spatial Information Sciences, 2004.

[22] H. Schuster, “Segmentation of LiDAR Data Using the Tensor Voting Framework,” in *International Archives of Photogrammetry*, Remote Sensing and Spatial Information, 2004.

[23] C. A. Bouman, *CLUSTER: An Unsupervised Algorithm for Modeling Gaussian Mixtures*. Purdue University, July 2005, http://www.ece.purdue.edu/~bouman.

[24] M. Alexa and T. Klug and C. Stoll, "Direction Fields Over Point-Sampled Geometry," *Journal of WSCG*, vol. 11, February 2003.

[25] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 34, pp. 1–38, 1977.

[26] S. J. Roberts, D. Husmeier, I. Rezek, and W. Penny, "Bayesian approaches to gaussian mixture modeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 1133–1142, 1998.

[27] Z. Zhang, K. L. Chan, Y. Wu, and C. Chen, "Learning a multivariate gaussian mixture model with the reversible jump mcmc algorithm," *Statistics and Computing*, vol. 14, pp. 343–355, October 2004.

[28] M. Figueiredo and A. Jain, "Unsupervised learning of finite mixture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 381–396, 2002.

[29] S. Richardson and P. Green, "On bayesian analysis of mixtures with unknown number of components," 1997.

[30] P. J. Green, "Reversible jump markov chain monte carlo computation and bayesian model determination," *Biometrika*, vol. 82, pp. 711–732, 1995.

[31] J. G. Dias and M. Wedel, *An empirical comparison of EM, SEM and MCMC performance for problematic Gaussian mixture likelihoods Journal Statistics and Computing*, vol. 14. Springer Netherlands, October 2004.

[32] A. Linka, J. Picek, P. Volf, and G. Ososkov, "New solution to circle fitting problem in analysis of rich detector data," *Czechoslovak Journal of Physics*, vol. 49, pp. 161–168, February 1999.

[33] C. Tang and G. Medioni, “Curvature-Augmented Tensor Voting for Shape Inference from Noisy 3D Data,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 858–864, June 2002.

[34] N. Mitra and A. Nguyen, “Estimating Surface Normals in Noisy Point Cloud Data,” in *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, pp. 322–328, ACM Symposium on Computational Geometry, ACM Press, 2003.

[35] M. Peternell, “Recognition and Reconstruction of Developable Surfaces from Point Clouds,” in *Proceedings of the Geometric Modeling and Processing*, pp. 301–310, 2004.

[36] H. Hoppe and T. DeRose and T. Duchamp and M. Halstead and H. Jin and J. McDonald and J. Schweitzer and W. Stuetzle, “Piecewise Smooth Surface Reconstruction,” pp. 295–302, 1994.

[37] U. Clarenz and M. Rumpf and A. Telea, “Finite Elements of Point Based Surfaces,” in *Eurographics Symposium on Point-Based Graphics* (M. Alexa, S. Rusinkiewicz, ed.), 2004.

[38] J. Solem and A. Heyden, “Reconstructing Open Surfaces from Unorganized Data Points,” *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 653–660, June-July 2004.

[39] N. Amenta and Y. Joo Kil, “Defining Point-Set Surfaces,” *ACM Transactions on Graphics*, vol. 23, pp. 264–270, August 2004.

[40] C. Andrieu and J. de Freitas and A. Doucet, “Sequential MCMC for Bayesian Model Selection,” in *Proceedings of the IEEE Signal Processing Workshop on Higher-Order Statistics*, pp. 130–134, IEEE Higher Order Statistics Workshop, 1999.

[41] C. Andrieu and N. de Freitas and A. Doucet and M. Jordan, “An Introduction to MCMC for Machine Learning,” *Machine Learning*, vol. 50, pp. 5–43, January 2003.

[42] Q. Shi and N. Xi, “Automated data processing for a rapid 3d surface inspection system,” *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 3939–3944, May 2008.

[43] J. Liu, J. Zhang, and J. Xu, “Cultural relic 3d reconstruction from digital images and laser point clouds,” *Image and Signal Processing, 2008. CISP ’08. Congress on*, vol. 2, pp. 349–353, May 2008.

[44] P. Hebert, D. Laurendeau, and D. Poussart, “Scene reconstruction and description: geometric primitive extraction from multiple viewed scattered data,” *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR ’93., 1993 IEEE Computer Society Conference on*, pp. 286–292, Jun 1993.

VITA

Thomas Patten

Candidate for the Degree of

Master of Science

Thesis: 3D MODELING OF POINT CLOUDS

Major Field: Electrical Engineering

Biographical:

Personal Data: Elk City, OK, United States on May 4, 1980.

Education:

Received a B.S. degree from Oklahoma State University, Stillwater, OK, United States, 2002, in Mechanical Engineering

Received a M.S. degree from Oklahoma State University, Stillwater, OK, United States, 2004, in Mechanical Engineering

Completed the requirements for the degree of Master of Science with a major in Electrical Engineering Oklahoma State University in July, 2009.

Name: Thomas Patten Date of Degree: July, 2009

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: 3D MODELING OF POINT CLOUDS

Pages in Study: 83 Candidate for the Degree of Master of Science

Major Field: Electrical Engineering

We present our research on modeling simple shapes in a point cloud using both Expectation Maximization (EM) and Markov Chain Monte Carlo (MCMC). The shapes to be modeled are Planes, Cylinders and Cones. Although most point clouds consist of more complex shapes, they may also contain these three simple shapes. In the case of man-made structures, a large percentage of a point cloud could be made up of simple shapes. Creating a model of shapes can lead to a huge reduction in data.

ADVISOR'S APPROVAL: Dr. Gouliang Fan