Optimizing Clustering Algorithms for Computer Vision

by

John A. Viloria

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of and Master of Engineering and Electrical Engineering and Computer Science at the MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2001

© John A. Viloria, MMI. All rights reserved.

The author hereby grants to MIT permission to reproduce and distribute publicly paper and electronic copies of this thesis document in whole or in part.

Author

Department of Electrical Engineering and Computer Science

Certified by......

Trevor Darrell
Assistant Professor
Thesis Supervisor

Accepted by...........

Arthur C. Smith
Chairman, Department Committee on Graduate Students
Optimizing Clustering Algorithms for Computer Vision

by

John A. Viloria

Submitted to the Department of Electrical Engineering and Computer Science on May 23, 2001, in partial fulfillment of the requirements for the degree of
and Master of Engineering and Electrical Engineering and Computer Science

Abstract

Clustering is the problem of taking a set of data points and partitioning them into groups such that points in the same group are similar to each other. For this thesis, I describe the clustering problem in depth, and present the K-means and Expectation Maximization (EM) techniques as two methods to solving the clustering problem. These algorithms are implemented, and their strengths and weaknesses are examined. To demonstrate the usefulness of clustering, I offer results of the clustering algorithms when run on images of people, and show how these algorithms can fit inside of a stereo system to perform person tracking.

Thesis Supervisor: Trevor Darrell
Title: Assistant Professor
Contents

1 Introduction ........................................... 9

2 Description of Clustering Algorithms ................. 13
   2.1 The K-means Algorithm ............................ 13
       2.1.1 Description .................................. 13
       2.1.2 Example ...................................... 14
       2.1.3 Advantages and disadvantages of using k-means .. 15
       2.1.4 Recent work with the k-means algorithm .. 16
   2.2 The EM algorithm .................................. 17
       2.2.1 Description .................................. 18
       2.2.2 Example ...................................... 23
       2.2.3 Advantages and disadvantages of using EM .. 23
       2.2.4 Recent work with the EM algorithm .. 25

3 An In-depth Example: K-means and EM within a Stereo System .... 27
   3.1 Description of Person Tracking System ............ 27
       3.1.1 Scene ......................................... 29
       3.1.2 Stereo Cameras ................................ 29
       3.1.3 Dense Background Model ....................... 29
       3.1.4 Background Subtraction ......................... 29
       3.1.5 Conversion to Plan-View ....................... 30
       3.1.6 Merge ......................................... 30
       3.1.7 Clustering ..................................... 30
### 3.2 Optimizing the Clustering Algorithms for Tracker Performance

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.1 K-means</td>
<td>31</td>
</tr>
<tr>
<td>3.2.2 EM</td>
<td>39</td>
</tr>
</tbody>
</table>

### 4 Conclusion

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
</tr>
</tbody>
</table>
List of Figures

2-1  K-Means Example ........................................... 15
2-2  EM Example .................................................. 24
3-1  Person Tracking System ..................................... 28
3-2  K-Means Person Tracking Results - Sample 1 ............ 33
3-3  K-Means Person Tracking Results - Sample 2 ............ 33
3-4  K-Means Person Tracking Results - Sample 3 ............ 34
3-5  K-Means Person Tracking Results - Sample 4 ............ 34
3-6  K-Means Trajectories - View 1 ............................ 35
3-7  K-means Trajectories - View 2 ............................. 36
3-8  K-means Trajectories - View 3 ............................. 37
3-9  EM Person Tracking Results - Sample 1 .................. 42
3-10 EM Person Tracking Results - Sample 2 .................. 42
3-11 EM Person Tracking Results - Sample 3 .................. 43
3-12 EM Person Tracking Results - Sample 4 .................. 43
3-13 EM Trajectories ............................................ 44
3-14 EM Trajectories ............................................ 45
3-15 EM Trajectories ............................................ 46
Chapter 1

Introduction

Clustering is a procedure which takes a set of $N$ patterns and partitions the points into $K$ disjoint classes or clusters so that points within the same class are similar, and points taken from different classes are dissimilar. How similarity is measured depends on the type of the clustering algorithm that is used. For *supervised* clustering algorithms, such as those used for object recognition systems, the classes under which the data could fall are already known. The task of the supervised clustering algorithm is to refine parameters in order to recognize new data as falling into one of the previously known classes. On the other hand, for *unsupervised* clustering algorithms, the classes are unknown. In the case where the patterns are points, similarity is usually defined as the Euclidian distance between the points; though other metrics for measuring similarity are possible.

There are several ways to measure the quality of a clustering[6]. When *external* measures are used, quality is measured by matching the clustering to some external knowledge about the data. For example, a clustering within a stereo vision system with a high quality value using external measurements would cluster points from the same object to the same class. When *internal* measures are used, quality is measured by taking statistics from the results of the clustering algorithm. One such statistic is the compactness of the clusters, which is the ratio of the mean of the variances of the clusters to the overall variance of the points:
where \( x_i \) are the points, \( m_{k(i)} \) is the mean of the cluster with point \( x_i \) in it, and \( m_{all} \) is the mean of all N points. Small values for \( V_{\text{compactness}} \) signify that the points within the clusters are closer to each other; hence the points within the clusters are similar to each other.

Another internal measurement statistic is the isolation of a cluster. If the mean of a cluster is defined as the arithmetic mean of all the points within that cluster and the average radius of a cluster as the average distance from points within a cluster to its mean, then the isolation of a cluster can be defined as the ratio of the distance from that cluster’s mean to the nearest cluster’s mean over the average radius of that cluster:

\[
I_h = \frac{\min_j (m_j - m_h)^2}{\sum_{i \in \text{cluster} k} (x_i - m_h)^2 / n_h}
\]  

Here \( I_h \) is the isolation measurement, \( m_j \) and \( m_h \) are means of clusters, and \( n_h \) is the number of points in cluster \( h \). Large values for \( I_h \) mean that the cluster is far from other clusters, and that the members of that class are very dissimilar to members of other classes.

For unsupervised classification problems, internal methods are more commonly used to measure quality because of the difficulties inherent in obtaining external knowledge about the data. Extra sensors or human observers are often required to retrieve this data; for example, for the stereo vision to check whether the data points representing a chair were clustered properly, the system would first need some external observer to identify which points corresponded to the chair. Likewise, quality for supervised algorithms is often measured using external measures.

In the context of computer vision, however, clustering within is generally an unsupervised problem. It is impractical to assume that a vision system can build a database containing a sample of every possible object or person that a camera can see. Hence, computer vision systems often rely on unsupervised clustering methods.
The rest of this paper is organized as follows. Chapter 2 describes the K-means algorithm and the expectation maximization (EM) algorithm, both of which perform unsupervised clustering. The algorithms are derived, and simple examples of their execution are provided. Chapter 3 examines a more complex example of clustering - from within a computer vision system which performs person tracking. The steps to prepare images for clustering are described, and the K-means and EM algorithms are customized to count the number of people in the image and to pinpoint each person’s location.
Chapter 2

Description of Clustering Algorithms

What follows in this chapter are descriptions of the K-Means and EM clustering algorithms.

2.1 The K-means Algorithm

The K-means algorithm was invented in 1965, and is attributed to MacQueen, who first coined the term k-means; and to Forgy, who developed the first iterative algorithm [5, 6]. K-means is often used for clustering because of its simplicity.

2.1.1 Description

In the basic k-means algorithm, the N points and the parameter k are given as inputs to the algorithm. The outputs are the k partitioned clusters centered around the means $c_1$, $c_2$, ... $c_k$. K-means attempts to minimize the expression:

$$
\min_{m_1, m_2, ..., m_{k-1}, m_k} \sum_{i=1}^{N} \min_{h=1,2,...,k} (x_i - m_h)^2
$$

(2.1)

by finding cluster centers $m_1$, $m_2$, ... $m_k$ such that the distance between each point and its cluster’s mean is minimized. This process is done iteratively. First, each
mean $m_j$ is fixed, and the points closest to each mean are identified. Then, each $m_j$ is minimized by setting $m_j$ to be the mean of the points attributed to it. More formally, the k-means algorithm can be described as follows:

**K-means algorithm**

1. Form an initial partitioning of the points into $k$ classes. For simplicity, usually a random partitioning of the points is implemented. The mean of each cluster is calculated.

2. Loop until the termination condition is met:
   
   (a) Assign each point to the nearest cluster. The distance from a point to a cluster is calculated by measuring the distance from the point to the mean of that cluster.
   
   (b) Re-assign the mean of each cluster by taking the arithmetic mean of the points within it.

Some possible termination conditions are until some percentage of points no longer switch classes or until some number of loop iterations are run. Within the loop, each of the $N$ points is compared with the means of each of the $K$ clusters, so the running time for the k-means algorithm is $O(NKD)$, where $D$ is the dimensionality of the points.

**2.1.2 Example**

Figure 2-1 shows a typical execution of the k-means algorithm partitioning a set of points into two clusters. Step 0 in the figure shows the set of points and the value $k$. In this example, $k$ equals two. Then at step 1, the points are randomly assigned to the two clusters, and the mean of each cluster is set to be the mean of the points assigned to it.

In the figure above, each class is represented with a different color, and the means are denoted by X's. At step 2a, each point is assigned to cluster with the closest mean, so the points at the top right part of the space are assigned to the red cluster.
and the points at the bottom left part of the space are assigned to the blue cluster. At step 2b, each mean is recalculated, so the red and blue means move to the center of their clusters. At this stage, the loop executes 2a and 2b again until the termination condition is reached. It so happens that in this example, the clusters are already optimally separated.

2.1.3 Advantages and disadvantages of using k-means

The k-means algorithm has several characteristics which make it an attractive clustering algorithm for use within person-tracking systems. Because it is so simple, it can be used in a flexible manner and can be embedded within other algorithms easily[4]. K-means is a greedy algorithm in that on each iteration it moves towards
a local minimum. For well-isolated data, a decent clustering can often be obtained after the first few iterations. Thus a person-tracking system using k-means with the right termination conditions can calculate clusters quickly.

However, the k-means algorithm has several key weaknesses which make it undesirable for use within person-tracking systems. Despite its simplicity, the algorithm can become slow when the number of points or the dimensionality of the points involved is large. For person-tracking systems, although the dimensionality of the points is not a problem, the clustering step can actually become the bottleneck within a person-tracking system for a large number of points. Another drawback is that K has to be provided as a parameter to the system; generally for an image, the number of people being tracked is not known beforehand. A third weakness is that the clusterings are not always the same; because of the random initialization element and because the algorithm is greedy, the clustering often converges at a local minimum rather than a global minimum[4].

2.1.4 Recent work with the k-means algorithm

Much of the recent research on the k-means algorithm has been done to combat these weaknesses. Alsbati, Ranka, and Singh[1] and Kanungo et al[4] propose similar algorithms which use k-d trees to organize the points in space. These k-d trees are helpful in computing the nearest neighboring mean to each point, as they filter away the more distant means during the assign points to nearest means stage. The result of using the k-d tree is a speedup, though at the cost of additional complexity. Looney[5] describes a fuzzy k-means algorithm which assigns probabilities that points belong in each cluster instead of a hard member/non-member assignment. The algorithm guesses the value of K by initially assuming a large number of fuzzy clusters and then merging them. Looney’s fuzzy algorithm is more robust to reordering of the points, but it uses the original k-means algorithm as a sub-procedure and is slow as a result of all of its extra calculations. To counter the local minima problem, Bradley, Bennett, and Demiriz[2] added the constraint to the k-means problem that each cluster had to have some minimum number of points; they report fewer local solutions but
neglect to report the speed cost. Eldershaw and Hegland[3] propose using the graph theory method of triangulation instead of a sum-squared-difference to measure distances, since the sum-squared-distance method is biased towards generating spherical clusters. They found that the graph theory methods generated better clusters, but at a significant speed cost.

2.2 The EM algorithm

The expectation-maximization (EM) algorithm is another algorithm that performs unsupervised clustering. Given N data points $x_1, x_2, ..., x_N$ derived from some probability density function $p(X|\Theta)$ and associated probability function $P(X|\Theta)$ such that the data points are independent of each other, EM finds the parameters $\Theta$ that characterize the distribution. That is, EM assumes that there are some underlying parameters which could be useful for clustering the data, but these parameters are unknown to the user; EM finds these parameters. It does this my maximizing the likelihood that a particular setting of parameters generated the data. More formally, if $L(\Theta|X)$ is the likelihood function of the parameters given the data, EM finds

$$
\max_\Theta L(\Theta|X) = \max_\Theta p(X|\Theta) = \max_\Theta \prod_{i=1}^{N} p(x_i|\Theta),
$$

(2.2)

where the individual $p(x_i|\Theta)$ can be factored because of the independence of the data points $x_i$.

The probability density function $p(x|\Theta)$ can take many forms. This paper focuses on the case that the pdf $p(x|\Theta)$ is a mixture of multi-variate Gaussian or normal distribution functions. Gaussians are most commonly chosen because they arise naturally in statistics and because they can be represented with very few parameters - just one mean and covariance matrix per Gaussian. However, other choices for the density function can be used; for example, when studying data from positron emission tomography, the density function is modelled better with Poisson’s distribution function than Gaussians[7, 9].

By assuming that $p(X|\Theta)$ is derived by a mixture of Gaussians, we assume that
the data is the aggregate of data obtained from many individual Gaussians. That is, the missing parameters \( \Theta \) are assumed to be the parameters \((\mu_1, \mu_2, \ldots, \mu_k, \Sigma_1, \Sigma_2, \ldots, \Sigma_k)\), where \( \mu_j \) and \( \Sigma_j \) are the mean and covariance matrix of Gaussian \( j \), and \( j \) ranges from 1 to \( k \). One can think of deriving a data point \( x_i \) from the aggregate density function by first picking one of the clusters to generate the data point. The probability that cluster \( j \) is responsible for generating the point is \( P(c_j) \), where the \( P(c_j) \) are subject to the constraint that \( \sum_{j=1}^{k} P(c_j) = 1 \). \( P(c_j) \) are sometimes called mixing parameters, because these parameters represent how much each cluster’s density function is weighted in the aggregate density function \( p(X|\Theta) \). Once a cluster is picked, the density function for that cluster, \( p(x_i|c_j) \), can be used to generate the data point.

The density function \( p(X) \) for the entire data set \( X = \{ x_1, x_2, \ldots, x_N \} \) is

\[
p(X) = \sum_{j=1}^{k} \sum_{i=1}^{N} p(x_i|c_j) P(c_j)
\]

Because the \( p(x_i|c_j) \) are assumed to take the form of Gaussian multivariate distribution functions, then

\[
p(x_i|c_j) = \frac{e^{-\frac{1}{2}([x_i-\mu_j]^T \Sigma_j^{-1} [x_i-\mu_j])}}{(2\pi)^{\frac{n}{2}} |\Sigma_j|^{\frac{1}{2}}}
\]

by definition.

The EM method is used because it is difficult to maximize \( L(\Theta|X) \) using standard analytic methods. Ordinarily one would take derivatives \( \frac{\partial L}{\partial \mu_j} \) and \( \frac{\partial L}{\partial \Sigma_j} \) and set them to zero; but solving the problem this way is intractable because 2.2 is a complicated expression, as the clusters that the points come from are not provided. Instead, EM can be used to estimate the parameters \( \mu_j \) and \( \Sigma_j \).

### 2.2.1 Description

EM is a two-step iterative algorithm which attempts to maximize the complete data likelihood function. Given a current estimation of the Gaussian parameters \( \Theta^t \), we
can find new parameters $\Theta^{t+1}$ such that $L(\Theta^{t+1}|X) \geq L(\Theta^t|X)$, or $\log L(\Theta^{t+1}|X) \geq \log L(\Theta^t|X)$, since the log function is smooth and monotonically increasing. The change in the likelihood per iteration is:

$$\log \left( L(\Theta^{t+1}|X) - L(\Theta^t|X) \right) = \sum_{i=1}^{N} \log \frac{\sum_{j=1}^{k} p(x_i|c_j^{t+1}) P(c_j^{t+1})}{P(X|\Theta^t)}$$

Now if we multiply the numerator and denominator of the fraction by $P(c_j^{t}|x_i)$ we get

$$\sum_{i=1}^{N} \log \left( \frac{\sum_{j=1}^{k} p(x_i|c_j^{t+1}) P(c_j^{t+1}) P(c_j^{t}|x_i)}{P(X|\Theta^t) P(c_j^{t}|x_i)} \right)$$

We can simplify the above expression using Jensen's inequality. Jensen's inequality states that if $\sum_j \lambda_j = 1$ and $\lambda_j \geq 0$, then we can conclude $\log \left( \sum_j \lambda_j y_j \right) \geq \sum_j \lambda_j \log(y_j)$. By setting $\lambda_j = P(c_j^{t}|x_i)$, the above expression becomes

$$\log \left( L(\Theta^{t+1}|X) - L(\Theta^t|X) \right) \geq \sum_{i=1}^{N} \sum_{j=1}^{k} P(c_j^{t}|x_i) \log \left( \frac{p(x_i|c_j^{t+1}) P(c_j^{t+1})}{P(X|\Theta^t) P(c_j^{t}|x_i)} \right)$$

We want to maximize the change in $L$ with respect to $P(c_j^{t+1})$ and $\Theta^{t+1}$. If we were to expand the log in the division, we would find that $p(X|\Theta^t)$ and $P(c_j^{t}|x_i)$ are grouped with $P(c_j^{t}|x_i)$; because this term depends only on the values of the parameters at time $t$, it is constant and can be dropped from the expression. Hence we can define

$$Q = \sum_{i=1}^{N} \sum_{j=1}^{k} P(c_j^{t}|x_i) \log \left( P(c_j^{t+1}) p(x_i|c_j^{t+1}) \right)$$

to be the lower bound of the change in likelihood when the parameters are updated. We can separate $Q$ into the terms

$$Q = \sum_{i=1}^{N} \sum_{j=1}^{k} P(c_j^{t}|x_i) \log P(c_j^{t+1}) + \sum_{i=1}^{N} \sum_{j=1}^{k} P(c_j^{t}|x_i) \log p(x_i|c_j^{t+1})$$

(2.5)

and maximize $Q$ separately with respect to $P(c_j^{t+1})$ and $\Theta^{t+1}$ because the mixing parameters, means, and covariance matrices are not inherently related.

We solve for the value of $P(c_j^{t+1})$ that would maximize $Q$ by using Lagrange
multipliers. By introducing the multiplier $\lambda$ with the constraint $\sum_{j=1}^{k} P(c_j^{t+1}) = 1$, and taking the derivative of equation 2.5 and setting to 0, we obtain

$$\frac{\partial}{\partial P(c_j^{t+1})} \left[ \lambda((\sum_{j=1}^{k} P(c_j^{t+1})) - 1) + \sum_{i=1}^{N} \sum_{j=1}^{k} p(c_j^t | x_i) \log P(c_j^{t+1}) \right] = 0$$

Evaluating the partial derivatives yields

$$\lambda + \sum_{i=1}^{N} \frac{P(c_j^t | x_i)}{P(c_j^{t+1})} = 0$$

We can pull the $P(c_j^{t+1})$ out of the sum because it does not depend on $i$. After rearranging the terms, we get

$$\sum_{i=1}^{N} P(c_j^t | x_i) = -\lambda P(c_j^{t+1})$$

Now we can sum over all $j$ to get

$$\sum_{i=1}^{N} \sum_{j=1}^{k} P(c_j^t | x_i) = -\lambda \sum_{j=1}^{k} P(c_j^{t+1})$$

Using the fact the sum of the probabilities $P(c_j^t | x_i)$ and $P(c_j^{t+1})$ over all $j$ is 1, we get

$$\lambda = -N.$$ Therefore, the value of $P(c_j^{t+1})$ that maximizes $Q$ is:

$$P(c_j^{t+1}) = P(c_j^{t+1}) \frac{1}{N} \sum_{i=1}^{N} P(c_j^t | x_i)$$

We can likewise take partial derivatives to obtain an equation for $\mu_j^{t+1}$. Starting with equation 2.5 and taking the term which has $c_j^{t+1}$, we plug in the definition of the Gaussian distribution function 2.4 into $p(x_i | \Theta_j^t)$ and get $Q =$

$$\frac{1}{2} \sum_{j=1}^{k} \sum_{i=1}^{N} p(c_j^t | x_i) \left(-\log |\Sigma_j^{t+1}| - [x_i - \mu_j^{t+1}]^T(\Sigma_j^{t+1})^{-1}[x_i - \mu_j^{t+1}] - d \log(2\pi) \right)$$

where $(\Sigma_j^{t+1})^{-1}$ is the inverse of $\Sigma_j^{t+1}$. To get the best $\mu_j^{t+1}$, we take partial derivatives with respect to $\mu_j^{t+1}$ and set to 0 to obtain
This equation can be rearranged to find $\mu_j^{t+1}$:

$$
\mu_j^{t+1} = \frac{\sum_{i=1}^{N} x_i p(c_j^t | x_i)}{\sum_{i=1}^{N} p(c_j^t | x_i)}
$$

(2.7)

To obtain the value of $\Sigma_j^{t+1}$, we first simplify the matrix product in 2.5

$$
[x_i - \mu_j^t]^T (\Sigma_j^{t+1})^{-1} [x_i - \mu_j] = \text{tr} \left( (\Sigma_j^t)^{-1} [x_i - \mu_j] [x_i - \mu_j]^T \right)
$$

by multiplying out the vectors and using the matrix algebra result that $\text{tr}(AB) = \text{tr}(BA)$. Plugging into equation 2.5 gives:

$$
\frac{1}{2} \sum_{j=1}^{k} p(c_j^t | x_i) (\log(\vert \Sigma_j^{t+1} \vert)^{-1}) - \text{tr}((\Sigma_j^{t+1})^{-1} [x_i - \mu_j] [x_i - \mu_j]^T)
$$

Now we need some properties of symmetric matrices to proceed. For all symmetric matrices $A$, it is known that $\frac{\partial \log \vert A \vert}{\partial A} = 2A^{-1} - \text{diag}(A^{-1})$ and $\frac{\partial \text{tr}(AB)}{\partial A} = B + b^T - \text{diag}(B)$. Because covariance matrices are always symmetric, we can substitute $\Sigma_j^{t+1}$ for $A$. If we take the derivative of the above expression by $(\Sigma_j^{t+1})^{-1}$ we get:

$$
\frac{1}{2} \sum_{i=1}^{N} p(c_j^t | x_i) \left[ (2\Sigma_j^{t+1} - \text{diag}(\Sigma_j^{t+1}))^{-1} - \left( 2([x_i - \mu_j^t][x_i - \mu_j^t]^T) \right) \right.
$$

$$
- \text{diag}([x_i - \mu_j^t][x_i - \mu_j^t]^T) \right] = \sum_{i=1}^{N} p(c_j^t | x_i)(\Sigma_j^{t+1} - [x_i - \mu_j^t][x_i - \mu_j^t]^T),
$$

Setting this result to zero and solving for $\Sigma_j^{t+1}$, we get

$$
\Sigma_j^{t+1} = \frac{\sum_{i=1}^{N} p(c_j^t | x_i) [x_i - \mu_j^t][x_i - \mu_j^t]^T}{\sum_{i=1}^{N} p(c_j^t | x_i)}
$$

(2.8)

Now that we have the update equations for $P(c_j^{t+1} | \mu_j^{t+1}$, and $\Sigma_j^{t+1}$, we can de-
scribe EM in algorithm form.

**Expectation maximization algorithm**

1. For \( t = 0 \), initialize the mean \( \mu_j^t \), covariance matrix \( \Sigma_j^t \), and mixing parameter \( P(c_j^t) \) of each cluster. One common way to initialize the means is to randomly divide the points into \( k \) partitions and find the means and covariance matrices of these partitions, and to set the mixing parameters to \( \frac{1}{k} \).

2. Loop until the termination condition is met by updating according to:

\[
P(c_{j}^{t+1}) = \frac{1}{N} \sum_{i=1}^{N} P(c_{j}^{t}|x_{i})
\]

\[
\mu_{j}^{t+1} = \frac{\sum_{i=1}^{N} x_{i}p(c_{j}^{t}|x_{i})}{\sum_{i=1}^{N} p(c_{j}^{t}|x_{i})}
\]

\[
\Sigma_{j}^{t+1} = \frac{\sum_{i=1}^{N} p(c_{j}^{t}|x_{i})[x_{i} - \mu_{j}^{t}][x_{i} - \mu_{j}^{t}]^{T}}{\sum_{i=1}^{N} p(c_{j}^{t}|x_{i})}
\]

The \( p(c_{j}^{t}|x_{i}) \) in the above equations are evaluated using Bayes rule and then expanding the denominator:

\[
p(c_{j}^{t}|x_{i}) = \frac{p(x_{i}|c_{j}^{t})P(c_{j}^{t})}{p(x_{i})} = \frac{p(x_{i}|c_{j}^{t})P(c_{j}^{t})}{\sum_{i=1}^{N} \sum_{j=1}^{k} p(x_{i}|c_{j}^{t})P(c_{j}^{t})}
\]

The \( p(x_{i}|c_{j}) \) can be found by plugging \( x_{i} \) and the values for \( \mu_{j}^{t} \) and \( \Sigma_{j}^{t} \) into the definition of the Gaussian distribution function at equation 2.4. \( P(c_{j}^{t}) \) are the mixing parameters.

Some common termination conditions are until some set number of iterations, or until the Gaussians no longer move by a particular amount. The algorithm is called expectation maximization because at each step, the expected value of the
log-likelihood is computed as $Q$ in equation 2.5. The expectation is then maximized via the update equations listed above.

2.2.2 Example

The following figure shows the EM algorithm clustering seven points into two groups. The two ellipses shown are the one-sigma boundaries of the Gaussians; that is, the distance from the center of a Gaussian to the edge of the ellipse is exactly one standard deviation. For a Gaussian distribution function, this means that roughly 78% of the points generated by this Gaussian fall within the one-sigma boundary.

At the algorithm’s initialization, the points are randomly assigned to the two Gaussians, and the Gaussian means are taken to be the means of the points assigned to them. The covariance matrix of each Gaussian is likewise calculated. Because there are two clusters, the weight for each cluster is initialized to one-half. (If there were three clusters, then each weight would initially be one-third; if there were four clusters, each weight would be one-fourth, and so on.) Note that at initialization, each Gaussian has a large standard deviation; this signifies that the clusters are not compact.

After step 1, the Gaussian means have begun to separate. By step 7, the algorithm has fully converged. The bottom cluster has a round boundary, so the points generated from the bottom cluster are distributed equally in every direction. The top cluster is stretched in the x-direction, so points from the top Gaussian vary more in the x-direction. Also of note are the weights of each Gaussian: 3/7 for the bottom Gaussian, and 4/7 for the top Gaussian. This shows that three out of the seven points are derived from the bottom Gaussian, and four of the seven were derived from the top Gaussian.

2.2.3 Advantages and disadvantages of using EM

There are many reasons why EM is a good classifier to use. As seen above, EM is a complicated algorithm to derive; however, the results are easy to use and implement.
Figure 2-2: The EM algorithm at initialization and after several iterations.
In the mixture model case, the re-estimation procedure is a simple iterative formula. Because of this simplicity, EM, like K-means, is a very good algorithm to embed in other algorithms and systems [9]. Another reason EM is a good algorithm is that it has a small representation for the groupings. The clusters are represented as just two values - the means and covariances matrices of each cluster. Like k-means, EM is greedy, and it is guaranteed to converge at a local minimum. Unlike k-means, which requires that each data point be either a member or non-member of a particular cluster, EM gives each data point a non-zero probability of being assigned to any of the clusters. Because EM makes softer assignments, EM is less susceptible to local minima than k-means. Mixture model EM is also more adaptable than k-means in that the clusters generated by the algorithm are of any ellipsoidal shape at any orientation; k-means just generates spherical clusters.

Using EM does have its drawbacks, however. Its most important drawback is its relatively slow convergence. There are ways to speed up EM by using Aitken’s approximation or other numerical methods[10], but these increase the complexity of the algorithm to the point that the simplicity methods over other clustering methods is lost[9]. Another drawback is that EM, like k-means, converges at a local minimum; it does not find a global minimum. This drawback, coupled with its slow convergence, makes a good initialization of the algorithm extremely critical. EM also has the same flaw as k-means in that k must be supplied as an argument to the algorithm; this is a problem for stereo vision systems, since generally k is unknown.

2.2.4 Recent work with the EM algorithm

EM is an immensely popular algorithm, and its clustering abilities are employed for a variety of uses. Earlier in this paper, it was mentioned in passing that EM can be used to analyze data from positron emission tomography. The EM algorithm is also the backbone for the Baum-Welsh algorithm for training hidden Markov models in speech recognition[12]. Another common application of EM is the identification of discrete-time stochastic systems [13].

On the theoretical front, there have been several attempts to tailor EM so as to
determine the value of k; this problem is more evasive than it looks, however. The most intuitive method from a statistics point of view is to use the likelihood ratio statistic to guide the search for the value of k: starting with an initial guess, use EM to evaluate $P(X|\Theta, k_{clusters})$, then guess there are k+1 parameters and evaluate $P(X|\Theta, k+1_{clusters})$; if the ratio of the first to the second probability is small, then it is more likely there are k+1 parameters. However, the usual regularity conditions for chi-squared distributions do not hold for mixture models, so the above method generates too many clusters. McLachlan [11] describes a complicated algorithm which generates bootstrapped samples at each value of k and then computing likelihood statistics for the new samples, but it is computationally very expensive.
Chapter 3

An In-depth Example: K-means and EM within a Stereo System

This section describes how K-means and EM can be used within a stereo system for the purposes of person tracking. Person tracking is the task of identifying people and their locations. A complex procedure for obtaining a set of images will be described; then it will be shown how these images can be processed by the clustering algorithm to locate people in the images. The results of the tracking will be evaluated, and possible improvements to the tracker will be suggested.

3.1 Description of Person Tracking System

The main focus of this section is how the clustering procedure can affect the quality of the clusters obtained. However, we will briefly describe the method for data collection.

In this example, the clustering algorithms perform person tracking when processed on the plan-view images obtained from following system. (Note that this is exactly the same system as [15], except for the choice of clustering strategy.)

Each of the modules in the above system will be described in further detail. The module that performs clustering will be described in a separate section.
Figure 3-1: The modules in a person tracking system.
3.1.1 Scene

The scene consists of everything that the cameras can see. It is a marked-off section of a room. People that appear in the scene are to be identified and located by the system.

3.1.2 Stereo Cameras

Each stereo camera module consists of two cameras which compute depth. Since the translation and orientation relating the left and right camera is known, depth can be calculated by matching features in the left and right camera and applying geometry.

3.1.3 Dense Background Model

Dense background data is obtained by performing stereo under a variety of gain conditions. Under one brightness level, pixels are often difficult to classify as either the foreground or background. By illuminating the scene under different brightness levels, it is possible to compute a smooth disparity distribution for the scene than a sparse one. The depth is stored in a histogram to allow for online background modeling.

3.1.4 Background Subtraction

Background subtraction extracts the foreground from a scene. For each pixel in the image we read off the median disparity value from the histogram. If the disparity at that pixel is greater than the median, then the pixel qualifies as part of the foreground.

By keeping a background histogram and subtracting it from the scene, we can identify the potentially mobile parts of the scene. These elements are assigned to the foreground.
3.1.5 Conversion to Plan-View

Each foreground point is transformed to three-dimensional world coordinates using the calibration characteristics for each camera, where two orthogonal axes are on the ground plane and the third axis is normal to the plane. By assuming that objects of different heights do not overlap in the scene (that is, assume that people do not stand on top of each other), we can ignore the height dimension to reduce the points to an overhead, two-dimensional plan-view representation of the scene.

When this step is reached, the image to be analyzed consists of just the foreground pixels of a two-dimensional, overhead view of the scene. The foreground consists of people, though any shifting object would also appear as foreground in the image. We can reduce the appearance of chairs or other non-persons in the scene by rejecting any pixels shorter than a certain height.

3.1.6 Merge

The plan-view foreground points from each camera are integrated into one plan-view using each cameras' calibration characteristics. The result is one image which can be segmented into clusters.

3.1.7 Clustering

Now we have just one image, consisting hopefully of pixels clusters which represent people. Here we can use a clustering technique of choice to group the pixels.

3.2 Optimizing the Clustering Algorithms for Tracker Performance

In [15], the clustering step was performed by a program that estimated the trajectories of moving objects based on dynamic programming principles. In this section, we run K-means and EM on the same images to show that using the K-means or EM
algorithm to cluster the plan-view foreground points would result in a suitable person tracker.

3.2.1 K-means

If we use a K-means clustering technique, then clustering is done in the two-dimensional space defined by the image; that is, the distance from one point to another is simply the Euclidean distance measured in the number of pixels. The clustering technique used for the person tracker is a K-means algorithm that dynamically chooses K based on the image. The pseudocode follows:

**Dynamic K-means algorithm**

1. Initialize K=1; set the mean of the cluster to be the mean of all of the points. Let $CurrentBlobs$ be the one cluster defined above.

2. Do:

   (a) Make a copy of $CurrentBlobs$. Calculate the compactness of the clusters as defined in 1.1.

   (b) Find the outlier points of each of the clusters in $CurrentBlobs$, where an outlier point is defined as a point some multiple of standard deviations away from the mean of that cluster. Take the union of these outlier points and assign them to a new K+1-th cluster; remove the outlier points from the original clusters. Run K-means on the K+1 clusters for some fixed number of iterations, and calculate the compactness of the resulting clusters.

   (c) If the sum of the compactness measurements for the k+1 clusters is smaller (i.e. better) than the sum of compactness measurements for the k clusters, repeat. Otherwise, return the k clusters from the copy of $CurrentBlobs$ from step 2.

The compactness measurements for each cluster is simply the sum of the distances from the mean to all points within the cluster. The idea behind using the outliers
is that the outliers are by definition isolated from the original clusters; this suggests
that these points may actually belong to their own cluster. At each iteration of the
loop, we evaluate quality of the clustering of some $k$ clusters versus the quality of
$k + 1$ clusters, where the last of the $k + 1$ clusters is formed from the outlier points
of the first $k$ clusters. If quality with more clusters is worse, then we stick with our
original guess of $k$ clusters; otherwise we keep searching for more clusters.

Time can be saved in this algorithm by saving the sum of the compactness values
from one loop iteration to another. This way, the compactness values do not have to
be re-computed.

It may happen that after running k-means on the $k + 1$ clusters, one of the clusters
has zero or an otherwise small number of points. Because the compactness for a one
point cluster is zero, having small clusters breaks our evaluation. We can fix this by
applying a penalty for all cluster sets with a small number of points.

In the person tracking system, the values chosen for the parameters were ten
iterations of k-means per loop; because there are so few foreground points (in the
order of one to two hundred), there is time to run this many iterations. The number of
standard deviations threshold for identifying outlier points is two standard deviations.
A penalty is of positive infinity is applied to cluster sets with empty clusters, so any
clustering with empty groups are immediately rejected.

Results

Presented below are some examples of the K-means tracker on a sequence of images.
A few representative samples show how clusters are found by the K-means tracker;
the different clusters are denoted by different colors in each figure. Afterwards, a
three-dimensional plot in (x, y, time) space displays the means found by the tracker
in each image.

Figures 3.6 - 3.8 show the most likely trajectories from running the dynamic k-
means on the images in [15]. The trajectories connect each mean in frame $t$ to the
nearest mean in frame $t + 1$, where nearness is measured by Euclidian distance. If
points were too far from any of the clusters, they were registered as new clusters.
Figure 3-2: Sample 1. The clusterer identified two clusters: one with 96 pixels and a mean at (154, 159), and one with 29 pixels and a mean at (29, 93).

Figure 3-3: Sample 1. The clusterer identified two clusters: one with 179 pixels and a mean at (163, 156), and one with 41 pixels and a mean at (72, 92).
Figure 3-4: Sample 3. The clusterer identified three clusters: one with 197 pixels and a mean at (117, 148), one with 74 pixels and a mean at (50, 95), one with 7 pixels and a mean at (32, 180).

Figure 3-5: Sample 4. The clusterer identified six clusters: one with 190 pixels and a mean at (118, 147), one with 22 pixels and a mean at (57, 94), one with 10 pixels and a mean at (73, 100), one with 19 pixels and a mean at (53, 91), one with 13 pixels and a mean at (61, 88), and one with 25 pixels and a mean at (93, 129).
Figure 3-6: K-means trajectories of the images in [15]. Each cluster represents a different color, and the path taken through x,y space. Clusters were tracked in consecutive frames by matching the mean in each cluster to the closest mean in the next frame. The cluster appeared in the sequence.
Figure 3-7: Same graph as above, from a different view.
Figure 3-8: Another view of the K-means trajectories, extracted from the images in [15].
Discussion

The dynamic clustering algorithm works well for many individual cases, especially when the groups are far away and unconnected. It correctly finds that there are two and three clusters in samples 1 and 2. However, the algorithm can run into trouble in some cases:

1. **Outlier points are in all directions.** If the outlier points surround the means in all directions, then the new mean is likely to be near the old means; as a result, the points near the center of the old means become a new cluster instead of the far away points. This problem can be fixed by taking note of the variances of each cluster. If the variance of a cluster is large, where large is some value predetermined to be too large to represent the same person, then that cluster should be separated. The new mean can be initialized correctly by assigning the new mean to a random outlier point of that cluster.

2. **Imbalance of clusters.** In many cases, the number of points within a cluster varies greatly from cluster to cluster. This is a side effect of using outlier points; because there are few points in a cluster which are two standard deviations away, the k+1-th cluster only represents a few points. Isolated clusters are more likely to be noise when they contain just a few number of points; for example, the cluster with only seven points in sample 3 above has too few points to represent a person. Lowering the number of standard deviations would balance the cardinality of the clusters more, but at the expense of creating extra, possibly extraneous clusters.

3. **Separating long strips into multiple clusters.** This is a flaw of the k-means algorithm: because a Euclidian distance measurement is used, the clusters are biased to be circle-shaped. As a result, long strips of points are divided into several circular clusters. To fix this problem, we can either add an isolation measurement to the quality clustering comparison, or we can use a clustering procedure that can accommodate non-circular shapes (such as EM).
The algorithm can be further modified to accommodate online tracking more smoothly. For each image at time $t$ for $t > 1$, instead of initializing k-means from one cluster, we can initialize k-means with the same means obtained from the image at time $t - 1$. We could also place a limit on how far a cluster can travel from one frame to another, where the limit is defined in terms of how far a person can travel in that time. There are two possible explanations for a cluster moving a long distance:

1. A new person entered the room, causing the k means from time $t - 1$ to shift to accommodate $k + 1$ means at time $t$.

2. A person left the room, causing the $k$th means from time $t - 1$ to jump to the $k - 1$ clusters in time $t$.

Both cases can be detected by counting the number of points from image to image. If the number of points increases dramatically in combination to a mean moving a large distance, then a new person has entered the room. K-means should be run again with $k + 1$ clusters, where the first $k$ clusters are initialized to the means at time $t$. If the number of points decreases dramatically in combination to a mean moving a large distance, then a person has exited the room. K-means should be run again on $k - 1$ clusters, and the $k - 1$ clusters should be initialized such as to minimize the total distance the clusters travel from time $t - 1$ to time $t$.

Another way to detect changes in the number of clusters is with person-creation zones. This concept will be discussed further in the following section on EM.

### 3.2.2 EM

The version of EM used to cluster points was the same EM described earlier at (2.2.1). Means and covariance matrices were initialized by randomly partitioning the points into $k$ groups and then calculating the mean and covariance matrix of each group. The clusters were run on different numbers of iterations for each sample. In general the larger the number clusters, the more iterations of EM are needed to obtain accurate results.
The number of clusters \( k \) was given as an input to the algorithm; the optimum \( k \) was chosen by inspection after running on several values of \( k \). However, it is possible to automate the selection of \( k \) by building person-creation zones into the system as in [14]. These zones are located at the entrances and exits to the room. When new foreground points appear or disappear in the sections of the plan-view image corresponding to the person-creation zones, then it is assumed that a new cluster was added or subtracted from the image. Having the system keep track of the number of entrances into the room guides the clusterer to group points into that number of clusters.

Results

Figures 3.13 - 3.15 show the results of using the EM clustering algorithm on the plan-view foreground images. The images are the same images used in section (3.2.1); though the output images are slightly centered and enlarged to show the neighborhoods of the ellipses better. The different ellipses represent the different clusters; the interior of the ellipse contains all of the points of Mahalanobis distance less than or equal to one. The Mahalanobis distance from a point \( x_i \) to a cluster \( j \)

\[
distance = (x_i - m_j)^T \Sigma_j^{-1} (x_i - m_j)
\]

(3.1)

where \( m_j \) and \( \Sigma_j^{-1} \) are the mean and inverse covariance matrix of cluster \( j \). (Mahalanobis distances are often used for measuring distance between points and Gaussian clusters because they have a built-in scale for non-round distributions, as defined by the covariance matrices.)

Figures 3.13 - 3.15 were calculated by running EM on the images in [15]. The number of clusters to run EM on was predetermined to be two or three, as suggested in [15].
Discussion

As expected, EM did better than K-means at identifying clusters which were not circle-shaped. Most notable was the way EM grouped the one long strip of pixels at the top left of the sample 4 image into one cluster while K-means grouped the strip into three circular clusters. The ability to identify clusters which are not circle-shaped is important, since point clouds for people in plan-view need not be circular. For example, if a person in the scene stretches out his or her arms, the point cloud representing the person becomes more elliptical in shape.

To improve tracking tracking results, EM can be strengthened in a similar way to the K-means algorithm. Initialize the means and covariance matrices to the same means and covariance from the last frame. Gaussians can be added or destroyed based on whether an upper bound of the values of the covariance matrix is reached or by using person-creation zones.

The one disadvantage of using EM instead of K-means is the extra iterations required when there are a large number of clusters. However, it is assumed that the number of people that can fit in any one scene is small; having too many people in a room would cause point clouds to merge on top of each other, and then no clustering algorithm would be useful. Because of this limitation, the time penalty for using EM instead of K-means is almost negligible.
Figure 3-9: Sample 1. The clusterer identified two clusters after ten iterations: one with weight .769841 and a mean at (161, 80), and one with weight .230159 and a mean at (34, 146).

Figure 3-10: Sample 1. The clusterer identified two clusters after ten iterations: one with weight .186364 and a mean at (72, 148), and one with weight .813636 and a mean at (163, 84).
Figure 3-11: Sample 3. The clusterer identified three clusters after twenty iterations: one with weight .692488 and a mean at (118, 94), one with weight .078121 and a mean at (94, 114), one with weight .229391 and a mean at (56, 147).

Figure 3-12: Sample 4. The clusterer identified two clusters after ten iterations: one with weight .761029 and a mean at (142, 85), one with weight .238971 and a mean at (67, 143).
Figure 3-13: EM trajectories of the images in [15]. Points of the same color represent the most likely path taken by the cluster.
Figure 3-14: Same image as above, from another view.
Figure 3-15: Same image as above, from another view.
Clustering using K-means or EM can be slow in cases with many data samples or dimensions, but these weaknesses can be minimized with proper filtering and dimension-reducing methods. The main advantages of using K-means and EM for clustering are that they are theoretically sound, simple to implement, and easy to embed in systems. I care, they can be applied to a large variety of problems, such as person tracking.

The method described above is just one approach to person tracking, based on clustering range data. A person tracker can be built by applying the clustering techniques of K-means and EM in the color domain. Person tracking can also be accomplished by identifying trajectories, by using pattern recognition systems, or a combination of the above.

While the K-means and EM trackers work well in many cases, the success of the K-means and EM trackers is not all-encompassing. Cases where people are situated near each other can confuse the trackers, as the point clouds are merged together in the image. Long point clouds may erroneously be broken down into too many circular clusters by K-means, or connected into too few ellipsoid point clouds by EM. Extra information, such as person-detection zones to supply the number of clusters, or any combination of pattern recognition, color, or trajectory information, is needed to support the clustering module.
Bibliography


[8] Jeff A. Bilmes, ”A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden


[15] "Plan-view Trajectory Estimation with Dense Stereo Background Models".