Analysis of a Spatio-Temporal Clustering Algorithm for Counting People in a Meeting

Yongjun Jeon, Paul Rybski

CMU-RI-TR-06-04

January 2006

School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

This paper proposes an algorithm that, given a time interval and the positions of people's faces located by a face detector, automatically determines the number of people present at a meeting. It should be noted that such a face detector often times produces noise and false positives, rendering the analysis of its results increasingly difficult. In any given frame, false positives may appear, and legitimate faces can go unnoticed, which calls for the use of statistical methods in the algorithm.

Exploiting clustering patterns based on temporal and spatial alignments of the detected faces, our algorithm employs the expectation-maximization (EM) algorithm [4] for mixture models and K-Means clustering algorithm [8]. The Gaussian mixture model [2] is used to estimate the probability density function of the data points; its parameters are then optimized using the EM algorithm, whose performance is in turn enhanced by its joint use with the K-Means algorithm. Also, by performing random restarts in the final model verification stage of the algorithm, different estimates are sampled using different parameters, and the most consistent result is chosen, under the assumption that an incorrect parameter set will have inconsistent fitting.

The results from this combination of algorithms and the sample training data set indicate the existence of the optimal set of parameters that produces estimates with locally minimum standard deviation and percentage error.

Finally, a stand-alone module will first be trained with a data set for which the ground truth is available for calculation of percentage errors. It will also implement an automatic, but simplified, model verification procedure with the parameters obtained from the data set.

Contents

List of Tables

List of Figures

Part I

Introduction

CAMEO (Camera Assisted Meeting Event Observer) is a hardware and software physical awareness system to record and monitor people's activities in meetings using a person-specific facial appearance model (PSFAM) for rapid tracking and identification of people [5]. While generating a omnidirectional video clip of the meeting at 15 fps, CAMEO tracks human faces by correlating the candidate locations in the current frame with those in the previous frames.

The problem being addressed in this project, however, is to determine the number of people present at a meeting, not to track human faces, which motivates an economical alternative approach without tracking and person-specific face recognition capabilities. It must be noted, though, that the task is not trivial, taking into account the imperfections within any face detection algorithm, which creates false positives. So simply counting the number of detected faces in each frame does not necessarily accomplish the task; on the other hand, an overlay of multiple consecutive frames is bound to display a pattern - namely, clusters of points - despite the presence of noise, under the assumption that people are seated and thus stay relatively stationary for the duration of the meeting. Then, in order to account for the noise and correlate back to the patterns displayed in previous frames, it becomes necessary to employ statistical methods in the analysis of the data. In particular, statistical methods capable of identifying the clusters and evaluating the goodness of the clustering, such as the K-means clustering algorithm [8] and expectation-maximization (EM) [4] algorithm, are examined. To illustrate this alternative approach, Schneiderman's face detection algorithm [9] is applied off-line after CAMEO has recorded the meeting on every single frame with no correlation between any frames.

Part II

Related Work

The advantage of the Gaussian mixture model is its ability to account for the outliers, whose presence is degrading to the overall quality of the clustering. Rather than investigating outliers as a byproduct of a clustering algorithm, Chiu and Fu [7] articulate the detection of outliers themselves, even in the case of multiple, overlapping clusters. However, their algorithm requires that the user specify the number of intervals to be partitioned in each dimension beforehand, which makes it unattractive for use in noise filtering in the current project, since its overhead may well be equivalent to the overhead of the entire model verification procedure.

Elkan and Hamerly [3] discuss several improvements on the k-means and the Gaussian expectation-maximization algorithms such as the fuzzy k-means and the k-harmonic means algorithms and present a new set of such algorithms that outperform both k-means and the Gaussian expectation-maximization algorithms. Nonetheless, the k-

harmonic means algorithm, by definition, receives a parameter p, typically $p \geq 2$. The algorithm in turn uses it as the exponent in computation of harmonic means, which can be computationally expensive for large p.

Part III
Approach

1 Visual Reconstruction of Data Points

Once all the points are overlaid and plotted on a 2D grid, different clusters of points indicate highly probable positions for different faces in a frame. Then the objective at hand becomes locating and counting these clusters.

It is necessary to visually reconstruct the data points in order to first verify the initial assumption on clustering pattern and then determine the types of algorithms to use. Hence, the following three diagrams are constructed:

1. An X-Histogram, with the x-coordinates of the detected faces on the x-axis and the frequency on the y-axis,
2. An Y-Histogram, with the y-coordinates of the detected faces on the x-axis and the frequency on the y-axis, and
3. A simple 2D plot of the face positions.

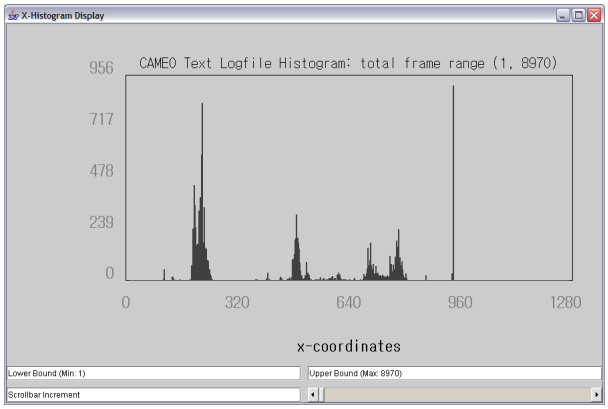


Figure 1: X-Histogram

The overlay of the data points from the entire clip, as shown in 3, does exhibit some clustering pattern, yet it is also useful to be able to observe changes in clustering patterns as the meeting proceeds. The scrollbars enable the user to specify the time interval, or the range of frames, to overlay.

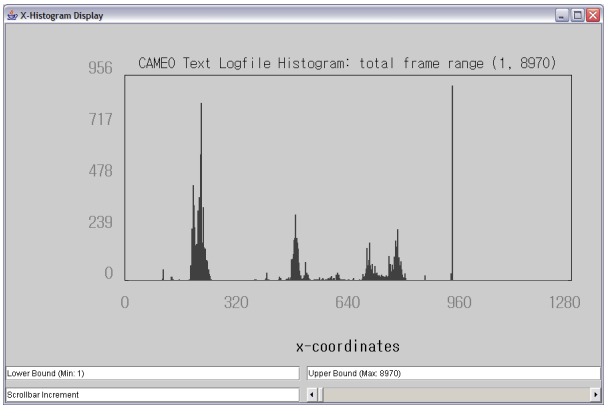


Figure 2: Y-Histogram

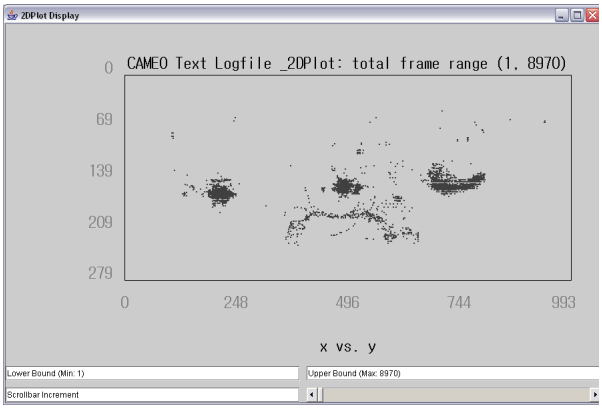


Figure 3: 2D Plot

2 Mixture Model

With the initial assumption verified, the Gaussian mixture model [2] is then employed to estimate the probability density function of the data points. For the sake of completeness, the algorithm is summarized below.

Probability density estimation is essentially searching the space of parameters to find the most likely probability density function $p(\mathrm{x})$ from a set of sample points $\mathrm{x}^n, n = 1, \cdots, N$ drawn from an unknown function. A mixture model is a particular form of semi-parametric estimation of probability density, creating a general class of functions with a set of parameters whose size is independent of the size of the data set.

For instance, consider a model with a probability density function which derives from a linear combination of basis functions and with M, the number of basis functions, as a parameter. Hence, the probability density function is expressed as a linear

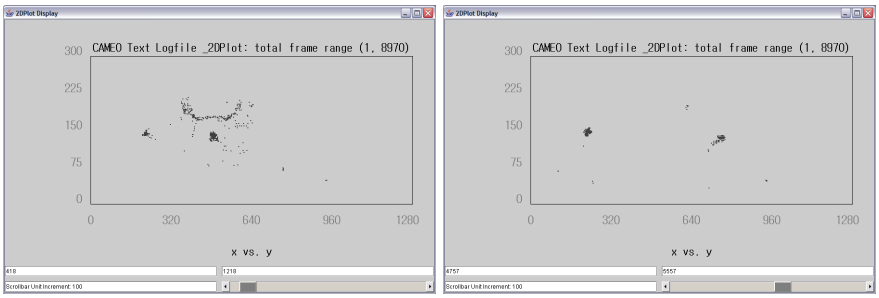


Figure 4: 2D Plot of face positions at different time intervals

combination of basis functions, or component densities $p(\mathrm{x}|j)$ in this case, as

$$p(\mathrm{x}) = \sum_{j=1}^{M} p(j)p(\mathrm{x}|j).$$

where $p(\mathrm{x})$ describes a *mixture distribution* and the coefficients $p(j)$ are the *mixing parameters*, or the *prior* probability that the data points originate from component j of the mixture. Intuitively, the following hold for the mixing parameters:

$$\sum_{j=1}^{M} p(j) = 1$$

$$0 \leq p(j) \leq 1$$

The component density functions $p(\mathrm{x}|j)$ are likewise normalized:

$$\int p(\mathrm{x}|j)d\mathrm{x} = 1$$

2.2.1 - Gaussian Mixture Models [2]

The basis functions are the components of a mixture density model, whose parameters must achieve maximum likelihood through optimization. We therefore model the density of the input data by a mixture model of the form

$$p(\mathrm{x}) = \sum_{j=1}^{M} p(j)\phi_j(\mathrm{x})$$

where the mixing coefficients $p(j)$ determine the proportion of $\phi_j(\mathrm{x})$ included in the mixture model and $\phi_j(\mathrm{x})$ are the basis functions.

The parameters of the basis functions can then be computed by re-estimation procedures based the application of the EM algorithm on maximization of the likelihood, which is discussed in the next section.

3 Expectation-Maximization (EM) Algorithm

EM is used to find the maximum likelihood estimates of parameters in probabilistic models, which depend on unobserved, or missing, latent variables. In particular, EM is an iterative approach to achieving the maximum likelihood of the parameters [2].

EM alternates between the two steps: The expectation step, where the expected values of latent variables are computed, and the maximization step, where the maximum likelihood of the parameters are computed from the given data and the latent variables are set to their expectations.

The general description of the algorithm is as follows:

Let y be the observed variables and let z be the latent variables.
Let $p(y, z|\theta)$ be the joint model of complete data with parameters θ.
Let $Q(\theta|\theta')$ be the expectation of the parameters θ given θ'; in other words, the expectation with respect to the probability density $p(y|z, \theta')$:

$$\begin{aligned} Q(\theta|\theta') &= E[\log p(y, z|\theta)|z, \theta'] \\ &= \sum_z [\log p(y, z|\theta) \cdot p(z|y, \theta')] \end{aligned}$$

Following the procedure below, the EM will then iteratively improve the initial estimate θ_0 and construct new estimates:

1. Set $i = 0$ and choose θ_i arbitrarily.

2. Compute $Q(\theta|\theta_i)$.

3. Choose θ_{i+1} so that it maximizes $Q(\theta|\theta_i)$; that is,

$$\theta_{i+1} = \arg\max_\theta [Q(\theta|\theta_i)]$$

   and then θ_{i+1} is the value maximizing the expectation of complete data log-likelihood with respect to the conditional distribution of the latent data.

4. If $\theta_i \neq \theta_{i+1}$, then set $\theta_i = \theta_{i+1}$ and return to step 2.

It is important to note the following two facts about the EM iteration:

- It can be shown that an EM iteration does not decrease the observed data likelihood function; that is,

$$p(y|\theta_{i+1}) > p(y|\theta_i)$$

- The fixed, data-dependent parameters of the function Q are calculated in the first step of the procedure. The function is fully determined once they are known and is maximized in the maximization step.

4 EM Algorithm for Gaussian Mixture Model

It is also important to note that EM is a description of a class of related algorithms and not a particular algorithm; EM is a skeletal structure for other more specific algorithms. Combining 2 and 3, an application of EM arises in determining the parameters of the basis functions of the Gaussian mixture model, given the number of classes to start with [2].

Assuming each Gaussian component has a covariance matrix that is a scalar multiple of the identity matrix (no correlation between any pair of different random variables), it has the form

$$p(\mathrm{x}|j) = \frac{1}{(2\pi\sigma_j^2)^{d/2}} \cdot \exp(-\frac{||\mathrm{x} - \mu_j||}{2\sigma_j^2})$$

And the negative log-likelihood for the data set is defined as follows:

$$E = -\ln L = -\sum_{n=1}^{N} \ln p(\mathrm{x}^n) = -\sum_{n=1}^{N} \ln [\sum_{j=1}^{M} p(\mathrm{x}^n|j)p(j)]$$

Remember that the EM is an iterative procedure; letting p^{new} be the probability under the new parameter values and p^{old} be the probability under the old,

$$E^{new} - E^{old} = -\sum_n \ln \frac{p^{new}(\mathrm{x}^n)}{p^{old}(\mathrm{x}^n)}$$

According to the definition of the mixture distribution given in 2,

$$\begin{aligned} \Delta E &= E^{new} - E^{old} \\ &= -\sum_n \ln \frac{\sum_j p^{new}(j)p^{new}(\mathrm{x}^n|j)}{p^{old}(\mathrm{x}^n)} \\ &= -\sum_n \ln [\frac{\sum_j p^{new}(j)p^{new}(\mathrm{x}^n|j)}{p^{old}(\mathrm{x}^n)} \frac{p^{old}(j|\mathrm{x}^n)}{p^{old}(j|\mathrm{x}^n)}] \end{aligned}$$

Jensen's inequality states that $\forall \lambda_j \geq 0$ such that $\sum_j \lambda_j = 1$,

$$\sum_j \lambda_j \ln(x_j) \leq \ln [\sum_j \lambda_j x_j]$$

Noting in the definition of E that $\sum_j p^{old}(j|\mathrm{x}) = 1$,

$$\Delta E = E^{new} - E^{old} \leq -\sum_n \sum_j [p^{old}(j|\mathrm{x}^n) \ln \frac{p^{new}(j)p^{new}(\mathrm{x}^n|j)}{p^{old}(\mathrm{x}^n)p^{old}(j|\mathrm{x}^n)}]$$

Letting Q be the right-hand side of the equation, the upper bound on is

$$E^{new} \leq E^{old} + Q$$

Recall that E represents the negative log-likelihood; in order to maximize the likelihood, E^{new} must be minimized, which necessarily follows a decrease in Q. For the sake of simplicity, consider the quantity $\widetilde{Q}$ which contains the terms of Q that do not depend only on the old parameters.

$$\widetilde{Q} = -\sum_n \sum_j p^{old}(j|\mathrm{x}^n) \ln [p^{new}(j) p^{new}(\mathrm{x}^n|j)]$$

Since the distribution p of interest is a Gaussian mixture model,

$$\widetilde{Q} = -\sum_n \sum_j p^{old}(j|\mathrm{x}^n)[\ln p^{new}(j) - d \cdot \ln \sigma_j^{new} - \frac{||\mathrm{x}^n - \mu_j^{new}||^2}{2(\sigma_j^{new})^2}] + const$$

It now remains to differentiate and minimize $\widetilde{Q}$ with respect to the new parameters μ_j^{new}, $(\sigma_j^{new})^2$, and $p(j)^{new}$ to get the EM update equations for the mixture model. Note that in minimizing $\widetilde{Q}$, the method of Lagrange multipliers is used with the constraint that $\sum_j p^{old}(j|\mathrm{x}^n) = 1$, leaving $\widetilde{Q} + \lambda(\sum_j p^{new}(j) - 1)$ for differentiation with respect to $p^{new}(j)$, which gives $\lambda = N$, the number of data points.

$$\begin{aligned}
\mu_j^{new} &= \frac{\sum_n p^{old}(j|\mathrm{x}^n)\mathrm{x}^n}{\sum_n p^{old}(j|\mathrm{x}^n)} \\
(\sigma_j^{new})^2 &= \frac{1}{d} \frac{\sum_n p^{old}(j|\mathrm{x}^n)||\mathrm{x}^n - \mu_j^{new}||^2}{\sum p^{old}(j|\mathrm{x}^n)} \\
p^{new}(j) &= \frac{1}{N} \sum_n p^{old}(j|\mathrm{x}^n)
\end{aligned}$$

where d is the number of components or variables considered in the model.

The actual implementation of 4 implements these update equations for each Gaussian component class. It also includes an additional class following uniform distribution to account for the outlying points that practically do not belong to any of the other classes - those having equally low likelihood in all the other classes - thereby reducing the negative effects of the noise which might otherwise deteriorate the quality of the outcome.

The implemented EM module is based on the K-Means demonstration applet by Akaho [1]. Although 4 intuitively seems well-suited for the problem, it soon becomes obvious that random initial distribution of classes often does not lead to an accurate localization of the clusters, not to mention the need to speed up the algorithm, which entails appropriately positioning the classes rather than just starting off at random locations. A second algorithm is needed; more specifically, a clustering algorithm which can partition data points into different clusters and return the representative points, or the mean points, to be used in the initialization of the EM classes.

Having specified the desired features of the new algorithm, the algorithm considered next is the K-Means clustering algorithm.

5 K-Means Clustering Algorithm

K-Means clustering algorithm [2, 8] is an algorithm for partitioning N data points $\mathrm{x}^i; 1 \leq i \leq N$ into K disjoint subsets $S_j; 1 \leq j \leq K$ containing N_j data points while minimizing the sum-of-the-squares error E:

$$E = \sum_{j=1}^{K} \sum_{n \in Sj} ||\mathrm{x}^n - \mu_j||^2$$

where μ_j is the mean, or the geometric centroid, of the set S_j, which is computed by

$$\mu_j = \frac{1}{N_j} \sum_{n \in S_j} \mathrm{x}^n$$

and is updated using

$$\Delta\mu_j = \eta \cdot (\mathrm{x}^n - \mu_j)$$

where η is the *learning rate parameter*.

The implemented K-Means module is based on the K-Means demonstration applet by Matteucci [6]. Now we are able to compute the mean points, but we still have no information about the approximate size of each cluster. In order to appropriately size the Gaussian mixture classes, the following values are needed - the variance of the x-coordinate, the variance of the y-coordinate, the covariance of the x- and y-coordinates, and the standard deviations of the x- and the y-coordinates of the data points in each cluster - which calls for the computation of a 2×2 covariance matrix (of x- and y-coordinates) in the K-Means module.

After the module has computed the mean points and the approximate size of each cluster, the information is retrieved and passed to the EM module in order to position and size the Gaussian mixture classes accordingly. The results are satisfactory; the proper positioning and sizing of the classes improves both the accuracy and the runtime significantly. It would be useful to color each point according to the cluster it belongs to. The implementation of this feature is relatively trivial, partly because it is partly a feature of the original K-Means module, and no further computation is needed to pass the necessary information to the EM module.

6 Model Verification

The objective of this project is to automatically determine the number of people present at a meeting. However, up until this point, despite the enhancements made to the K-Means and the EM modules, the user still has to specify the number of mean points K before running the K-Means module and re-run the algorithm until the result visually shows an acceptable partitioning of the points. In order to automatize this crucial step, a proper method for model verification must be devised. The following two terms are hereby explicitly defined to refer to the specific features of the verification procedure. The readers may also find it helpful to refer to the flowchart denoting the model verification process at the end of the paper (Figure 23).

```
For k k-means classes
    Do i k-means runs with k classes
    Pick the best set of mean points (maximum initial EM likelihood)

    n first EM Runs

    Take the run with the greatest likelihood
End For

Pick the best trial (maximum EM likelihood)
```

Table 1: Pseudocode for initial model verification

- An n^{th} *trial* refers to the collection of subroutines performed with n K-means classes.
- Each subroutine includes the notion of *runs*, or the iterations of the K-Means or EM module.

6.1 Initial Approach

Since, when automatized, the modules will have to run on their own and not necessarily report the results back to the user after every single run, a character-based user interface (CUI) was added in addition to the GUI for easier stand-alone, command-line operation.

The initial layout for model verification is as follows, and a pseudocode implementing the layout is presented afterwards:

1. Receive from the user the following: K, the maximum number of classes for the K-Means module to attempt up to; i, the number of K-Means runs; and n, the number of EM runs, and the desired EM accuracy.
2. For each run of K-Means, look at the number of points contained in each partition produced by K-Means and choose the trial with the *lowest standard deviation* (in the number of points in each partition).

Nonetheless, this approach is flawed because the K-Means module does not account for the outlying points, or the noise, like the EM module does with the additional uniform class, and low standard deviation - or low differences between the number of points in each partition - do not necessarily correlate to a better partitioning; even though it is generally the case that the K-Means algorithm does tend to partition the data into disjoint sets of approximately equal size, the inherent randomness in the initialization of the mean points in the K-Means module must be taken in to account, not to mention the fact that the final positions of the mean points are determined geometrically and the sizes of each set may very well be unequal, depending on the initialization the mean points. Also note that although on a given interval each cluster may contain similar number of points due to the facts that the same number of people are present in

the interval and that the face detection algorithm works frame-by-frame, the presence of noise in the face detection algorithm as well as the possibility of occlusion must be taken into account as well.

6.2 Devising an Improved Method for Model Verification

6.2.1 Evaluation of K-Means Partitioning by Initial EM Likelihood

Successful selection of the most appropriate K-Means partitioning is influential to the final answer that the program will produce; the evaluation of partitioning must incorporate the issue of whether the partitioning separates and captures each distinctive clusters with a high probability, which is precisely the question the Gaussian mixture model addresses, and the EM module in fact quantifies this quality as likelihood. Thus, an improved evaluation of K-Means partitioning is obtained by passing the partitioning information to the EM module and then retrieving the initial likelihood value that it computes. To illustrate the improved performance of the initial EM likelihood evaluation, a meeting clip is processed using both the standard deviation method and the initial EM likelihood evaluation. Figure 5 shows the orientation of the data points of the 5 people's faces throughout this particular meeting. Table 2 shows the results of the

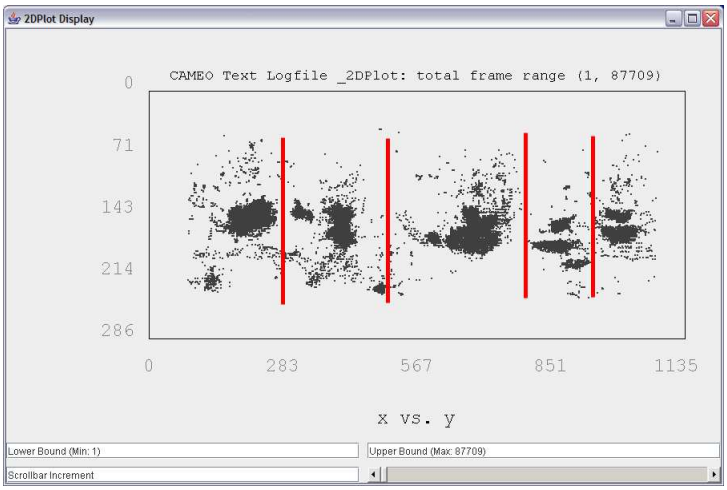


Figure 5: Meeting clip used for comparison, **5** clusters

comparison. The lowest standard deviation value appears in bold in standard deviation column. Likewise, the greatest initial likelihood value appears in bold in initial likelihood value column. As the table shows, the two boldfaced values do not appear in the same row. It also shows that, in this particular experiment, the standard deviation method actually led to the wrong conclusion by the program.

Run	Std Dev in # Points	Initial EM Likelihood	Correctly Spotted All 5 Clusters
1	26544.54	-11.7049049	FALSE
2	26445.77	**-11.5552135**	TRUE
3	**24044.43**	-11.6530598	FALSE
4	26444.75	-11.5553968	TRUE
5	34630.97	-11.8339914	FALSE

Table 2: Comparison of Standard Deviation and Initial EM Likelihood Evaluations

6.2.2 Controlling the Sampling Rate

Also crucial to the quality of the final answer is the presence of noise; a desirable procedure would effectively reduce the amount of noise present in the data while preserving the unique patterns in the orientation of the points, thereby preserving the clusters as well. One way to reduce the noise is to reduce the number of samples we take. Under the assumption that people in a meeting are generally stationary, a small decrease in the sampling rate does not significantly distort the clusters but reduces the frequency and the probability of sample noise.

For the purpose of this project, two different sampling rates - 1 fps and 15 fps - are experimented with.

Figures 6 and 7 illustrates the samples obtained from the same logfile with different sampling rate; left two and right two are obtained from the same images, respectively.

6.2.3 Dividing the Entire Clip into Subintervals

A question directly following 6.2.2 is whether the clustering pattern is lost when all the data points are overlaid onto a single frame. It is assumed that people at a meeting remain generally stationary; nonetheless, it is still possible that people move around and change seats so as to disturb the clustering pattern in the long run, not to mention the cases in which people leave the meeting or new people join the meeting. Hence, it makes intuitive sense to examine the whole clip in subintervals; a subinterval is bound to contain less moments of significant movements, or noise, and focusing on a subinterval rather than the entire clip corresponds to examining a smaller set of data points, thereby preserving the local characteristic of the data.

For the purpose of this project, five subintervals of different lengths - 10 seconds, 30 seconds, 60 seconds, 100 seconds and the entire clip - are experimented with. It should be noted that the number of frames each subinterval contains is dependent on the chosen sampling rate; that is, a 10-second subinterval at 1 fps contains 150 frames, whereas a 10-second subinterval at 15 ftp contains 10 frames.

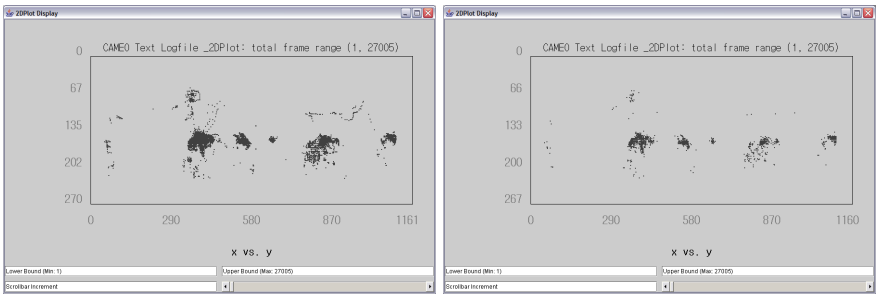


Figure 6: Clip sampled at 15 fps (left); at 1 fps (right)

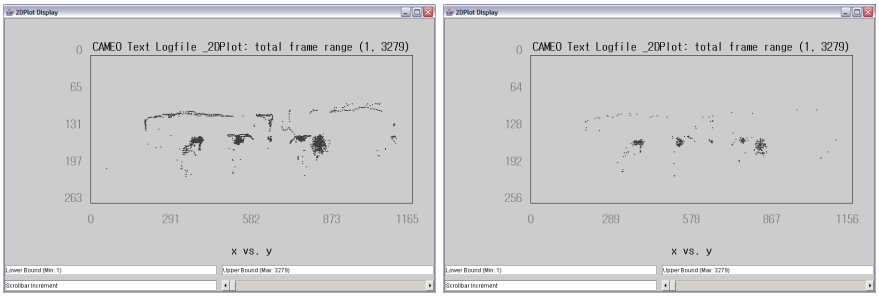


Figure 7: Different clip sampled at 15 fps (left); at 1 fps (right)

6.2.4 Running a Second EM without Overlapping Points

One thing that may not be apparent upon initial inspection of the K-Means partitioning is whether any of the clusters overlap with one another, as the K-Means module produces only the mean points of the clusters. Overlapping clusters correspond to an uncertainty in the identification of clusters and must accordingly be accounted for in the final answer. It must be noted that just a mere removal of the points contained in the intersections of clusters does not resolve the issue; as illustrated in Figures 8 and 9, removal of overlapping points can result in a change in the likelihood value or an EM failure, especially for a *subcluster* - a cluster that is completely contained within another.

Detection of Overlapping Points Each EM class is an ellipse, whose lengths of the axes are determined by the standard deviations of the x- and the y-coordinates of data points and whose orientation of the axes are determined by the covariance between the two coordinates.

The covariance matrix reveals the lengths (and the orientation) of the major and the minor axes, and, by the property of ellipses, c, the distance from the center to either foci, is

$$\begin{aligned} c^2 &= a^2 - b^2 \\ c &= \sqrt{a^2 - b^2} \end{aligned}$$

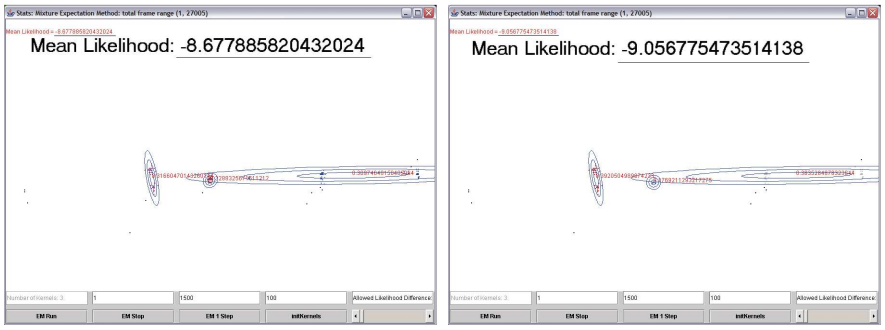
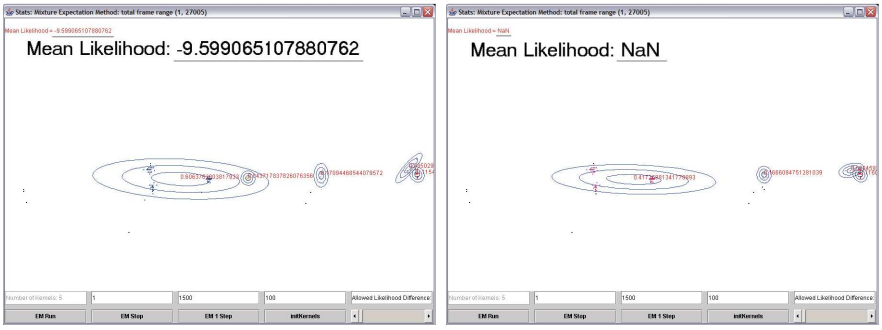
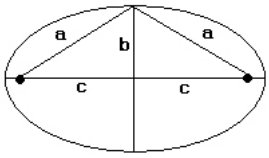


Figure 8: Before and after removal - Likelihood decrease



,

Figure 9: Before and after removal - EM failure (NaN likelihood)



where $2a$ and $2b$ are the lengths of the major and the minor axes, respectively. Further, it is possible to completely determine the locations of foci as well as the lengths of major and minor axes, and any point such that

$$(\texttt{Sum of distances from the foci}) \leq (\texttt{Length of major axis})$$

falls within the boundary of the ellipse or the class, in this case. The overlapping points can then be detected by keeping track of the number of classes that each point belongs to.

Weight Function In order to appropriately process the effect of the removal of overlapping points (since the removal of overlapping points does not always entail an increase in likelihood), the relative significance of the effect of the removal is quantified

by defining a weight function $p(i)$. Let N_{max} be the maximum number of overlapping points in an arbitrary trial, and let N_i be the number of overlapping points from i^{th} EM run. Then $p(i)$ is defined as follows:

$$p(i) = \begin{cases} \frac{N_i}{N_{max}} & : \quad N_{max} > 0 \\ 0 & : \quad otherwise \end{cases}$$

Note again that, if $N_i = 0$, then $p(i) = 0$, and if $N_i = N_{max}$, then $p(i) = 1$. Based on $p(i)$, a new likelihood value L_{new} is computed by mixing the likelihoods prior to (L_1) and after (L_2) the removal using the following equation:

$$L_{new} = (1 - p(i)) \cdot L_1 + p(i) \cdot L_2$$

Note that, if $p(i) = 0$, then $L_{new} = L_1$, and if $p(i) = 1$, then $L_{new} = L_2$.

Eliminating a Fixed Upper Bound for the K-Means Module As the number of K-Means classes increases, either an EM failure is more likely to occur, since each class is assigned less points, which degrades the performance of the inherently statistical algorithm, or the occurrence of overlapping points in forms of subcluster becomes more probable. A pre-set arbitrary constant f determines the maximum number of EM failure allowed in a trial; that is, a trial will stop upon confronting the n^{th} EM failure. For the purpose of this project, f has been experimentally determined to be 3.

7 Putting It All Together

The following pseudocode in Table 3 implements the improved model verification procedure, which is also illustrated in the flowchart at the end of the paper (Figure 23):

Part IV
Results and Findings

The logfile used in the first two parts of this experiment (detected_faces.txt, 2,197,222 bytes) has been prepared by Dr. Paul Rybski. Figure 10 is a 2D plot of all faces in the logfile.

As it appears in Figure 10, there are three readily identifiable clusters, with a considerable amount of noise present in the data set. We label the three clusters A, B and C, respectively, from the left.

To illustrate the improvements made on the performance of the EM module by initializing the EM classes to the mean points returned by K-Means module - which shall

```
For each time interval
    For k k-means classes
        Do i k-means runs with k classes
        Pick the best set of mean points (maximum initial EM likelihood)

        n first EM Runs
        Detect and remove overlapping points
        n second EM Runs

        Determine the two maximum EM likelihoods from the first
            and the second EM runs, respectively
        Compare and process the max likelihood values of the two using
            the formula and the weight function

        The final EM likelihood is then determined for this trial
    End For

    Pick the best trial (maximum final EM likelihood)
End For
```

Table 3: Pseudocode for improved model verification process

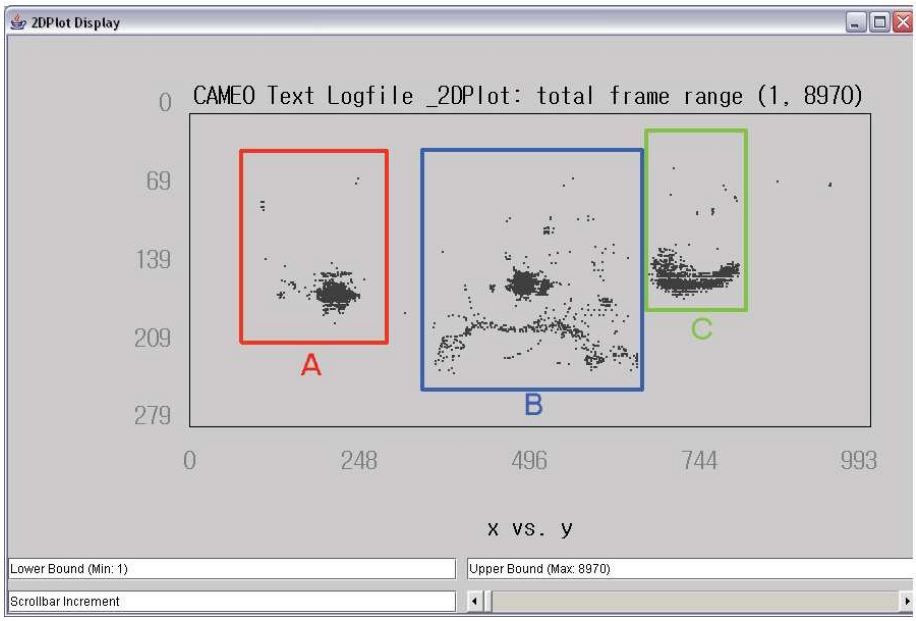


Figure 10: 2D Plot of the logfile with all three clusters labelled

Run	# EM Resets	# EM Iterations	Correctly spotted clusters	Mean likelihood
1	0	70	A ONLY	-9.76345
2	0	77	A ONLY	-9.76344
3	0	76	A ONLY	-9.76344
4	0	64	A ONLY	-9.76345
5	1	65	A ONLY	-9.76345
6	0	25	C ONLY	-10.39017
7	0	87	A ONLY	-9.76344
8	0	71	A ONLY	-9.76345
9	0	67	A ONLY	-9.76345
10	0	70	A ONLY	-9.76345
Average	.1	67.2	-	-9.82612
Std Dev	.3	15.5	-	.18802

Table 4: Result from Plain EM Procedure

now be called the *EM from K-Means procedure* - a comparison of the performances of the two procedures is presented: The *plain EM procedure* using only the EM algorithm and the *EM from K-Means procedure*. Also, the desired accuracy of the EM module in both procedures is empirically set to 1.0×10^{-5} and the number of classes to 3, the ground truth.

The EM module is designed to be robust, implying that it would reset upon encountering an invalid likelihood value such as infinity or NaN. The term *EM Resets* used in the tables below refers to this behavior of the program.

The *plain EM procedure* is examined first.

8 Plain EM Procedure

As it appears in Table 4, the plane EM procedure spots cluster A correctly with probability Pr[{A}] = $\frac{9}{10} = .9$, and the standard deviation of the likelihood values is $.188$, which is very large compared to the desired accuracy 1.0×10^{-5} specified. In the 10 runs conducted, it never spots all three clusters simultaneously. Also, the number of EM iterations fluctuates greatly at the average of about 67 iterations with the standard deviation of 15.5; the EM module, as per the definition of the EM algorithm, uses random initialization of classes, and it is clear from the experimental data that the behavior of the EM module, indicated by the number of iterations, depends heavily on the initialization of the classes.

Next, a similar set of experimental data is obtained from *the EM from K-Means procedure* and is analyzed. The data from the *EM from K-Means procedure* is twofold, consisting of the K-Means and the EM portions. To best illustrate the data, three interrelated tables are presented - one table containing the results from the K-Means portion of the procedure, a second one displaying the consistency in the mean points returned by the K-Means module, and another table containing the results from the EM portion of the procedure.

Run	# K-Means Resets	# K-Means Iterations	x_1	y_1	x_2	y_2	x_3	y_3
1	0	13	782	120	204	152	495	148
2	0	15	782	120	204	152	495	148
3	0	11	495	148	204	152	784	120
4	1	15	495	148	204	152	784	120
5	0	9	495	148	204	152	784	120
6	2	13	782	120	204	152	495	148
7	2	19	204	152	782	120	495	148
8	1	15	204	152	495	148	784	120
9	1	13	495	148	784	120	204	152
10	0	11	204	152	782	120	495	148
Average	.7	13.4	-	-	-	-	-	-
Std Dev	.8	2.7	-	-	-	-	-	-

Table 5: Initial Results of K-Means Algorithm

It is important to keep in mind that the K-Means module requires that the user specify the maximum k for the algorithm and that it is the user who determines whether the partitioning is acceptable upon visual inspection; if it is not, the user resets the K-Means and runs the algorithm again. The term *K-Means resets* refers to this action on the user's part. The k is 3 in this case, since there are 3 clusters of interest whose mean points need to be identified. The quantities $x_i, y_i \; \forall i$; $1 \leq i \leq k$ respectively refer to the x- and the y-coordinates of the mean point identified by the i^{th} class.

9 EM from K-Means

9.1 K-Means

9.1.1 Initial K-Means

The K-Means module, on average over the 10 runs, requires approximately 1 manual reset, and each run costs about 13 iterations with the standard deviation of 2.7. Although not as heavily dependent as the plain EM procedure, the performance of K-Means module, indicated by the number of iterations, does depend on the initialization of the class points, which is also randomly done according to the definition of the algorithm. Moreover, the module produces consistent results that are also correct, given that the produced result is visually acceptable; although it may be hard to notice at a glance with the same classes occasionally spotting different clusters, it is clear in Table (6), in which the mean points are rearranged according to the clusters A, B and C, that the results are indeed consistent, with 0 as the standard deviations of all the coordinates except that of the x-coordinate of the third class, which is 1.1.

Run	x_{A}	y_{A}	x_{B}	y_{B}	x_{C}	y_{C}
1	204	152	495	148	782	120
2	204	152	495	148	782	120
3	204	152	495	148	784	120
4	204	152	495	148	784	120
5	204	152	495	148	784	120
6	204	152	495	148	782	120
7	204	152	495	148	782	120
8	204	152	495	148	784	120
9	204	152	495	148	784	120
10	204	152	495	148	782	120
Average	204	152	495	148	783	120
Std Dev	0	0	0	0	1.1	0

Table 6: Mean points rearranged

Run	# EM Resets	# EM Iterations	Correctly spotted clusters	Mean likelihood
1	0	19	A, B, C	-9.43208
2	0	19	A, B, C	-9.43208
3	0	20	A, B, C	-9.43208
4	0	19	A, B, C	-9.43208
5	0	23	A, B, C	-9.43207
6	0	20	A, B, C	-9.43208
7	0	22	A, B, C	-9.43207
8	0	21	A, B, C	-9.43207
9	0	19	A, B, C	-9.43208
10	0	20	A, B, C	-9.43208
Average	0	20.2	-	-9.43208
Std Dev	0	1.3	-	4.58607×10^{-6}

Table 7: Results: EM from K-Means

9.2 EM

The results from the EM module consist Table 7, and the improvements on its performance, compared to the plain EM procedure, is readily observable. With no resets, the EM module is able to spot all the three clusters of interest with an average of approximately 20 iterations and the standard deviation of 1.3, along with a consistent mean likelihood, -9.43208, whose standard deviation, 4.58607×10^{-6}, is less than the desired accuracy, 1.0×10^{-5}, whereas the plain EM procedure is unable to spot all the three clusters at once, not to mention the fact that it costs more iterations and still produces less precise and less consistent results. In short, the *EM from K-Means procedure* produces results that are superior to those of the *plain EM procedure* with the only drawback of additional runs of K-Means module.

9.3 Images

Figures 11 and 12 are the screenshots of the K-Means and the EM modules before and after run, respectively.

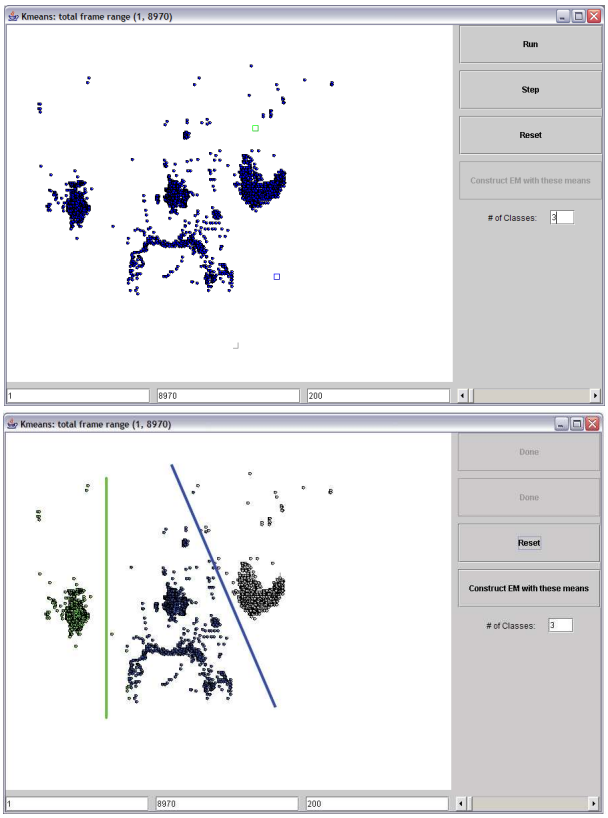


Figure 11: Images of K-Means module, before running (top) and after (bottom)

10 Demonstration of Initial Model Verification

The initial verification module described in 6 is called with the following parameters:

- 10 as the maximum K,
- 10 K-Means runs,
- 1 EM run, and
- 1.0×10^{-5} as the desired EM accuracy.

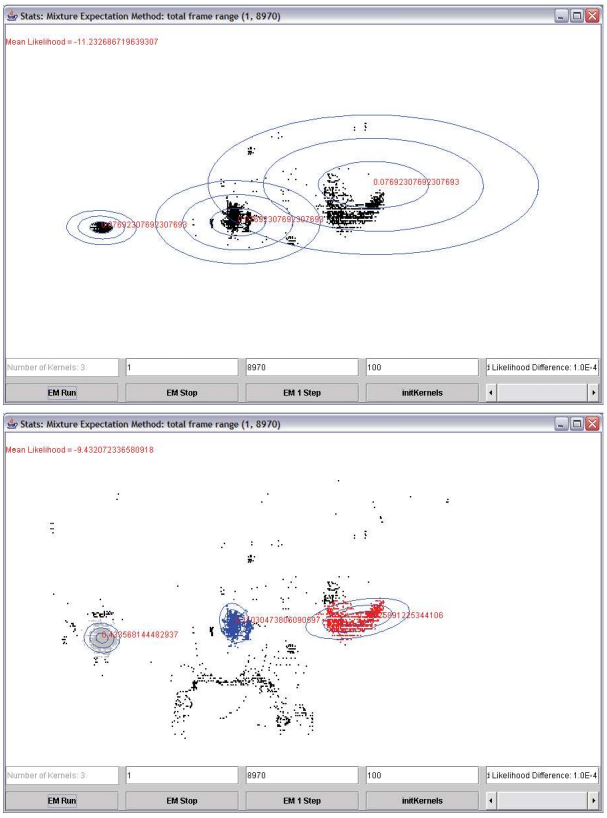


Figure 12: Images of EM module, before running (top) and after (bottom)

10 separate cases of model verification are examined and the results collected, which appear in Table 8.

It is obvious from Table 8 that the method is flawed; it predicts that, on average, there are approximately 9 people, or clusters, when only three are most obvious. The presence of a considerable amount of noise contributes to the failure of the method; Table 9 is the output of model verification run 5, and Table 13 is its graphical illustration, in which it appears that the likelihood values, as a function of the number of people (or clusters) is an increasing function, although not strictly increasing. It seems as though the increased number of classes help the module account for the noise by fitting to the noise in order to achieve a higher likelihood, for the K-Means module does not account for the noise with an additional class like the EM module does.

Case	Estimated # People	Likelihood
1	9	-9.07532
2	10	-9.09858
3	10	-9.08081
4	8	-9.12187
5	9	-9.04930
6	10	-9.05531
7	8	-9.05994
8	9	-9.04077
9	10	-9.04076
10	9	-9.07309
Average	9.2	-
Standard Deviation	.75	-
% Error	-206%	-

Table 8: Final Results

```
                    ⋮
Conclusion
==================
Likelihood of presence of 1 people : -11.175263033684622
Likelihood of presence of 2 people : -10.202015003943165
Likelihood of presence of 3 people : -9.713605941990616
Likelihood of presence of 4 people : -9.650842812581502
Likelihood of presence of 5 people : -9.63814376144801
Likelihood of presence of 6 people : -9.236791181913848
Likelihood of presence of 7 people : -9.639863491172887
Likelihood of presence of 8 people : -9.09046379303053
Likelihood of presence of 9 people : -9.049303546534903
Likelihood of presence of 10 people : -9.11586615347217

Estimated number of people present: 9

Procedure done. Exiting...
```

Table 9: Sample output of a model verification case (Case 5)

11 Improved Model Verification

The improved model verification procedure is designed to reduce this fallacy of the initial model verification method. It is invoked with the following parameters. Note that the maximum K K-Means classes is not given, since it is now automatically determined.

- Unchanged parameters

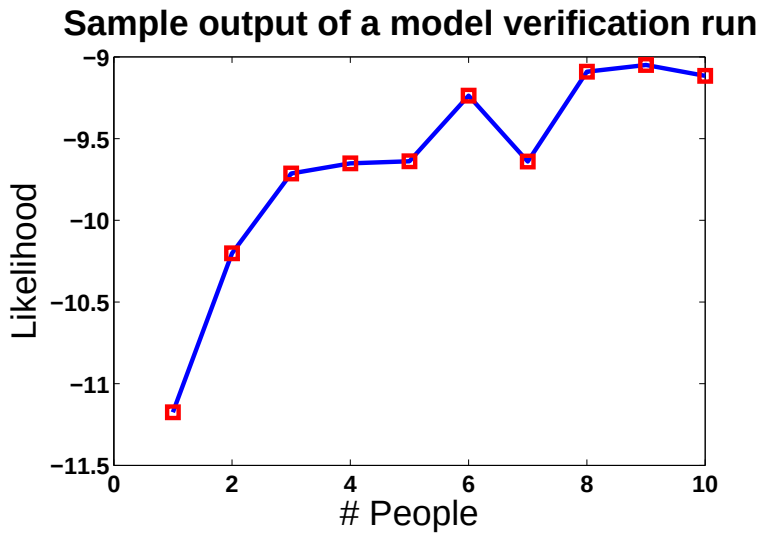


Figure 13: A Graphical Illustration of (2): # People vs. Likelihood

  - 10 K-Means runs,
  - 1 EM run, and
  - 1.0×10^{-5} as the desired EM accuracy.

- Varied parameters
  - Sampling rates: 1 fps, 15 fps
  - Interval lengths: 10 seconds, 30 seconds, 60 seconds, 100 seconds, and the whole clip

5 separate cases of model verification results are obtained for each sampling rate and interval length pair, and the results are collected for the next three new logfiles, followed by the sample screenshots of each video clip. Note that the absolute percentage error is computed as

$$|\frac{Result - GroundTruth}{GroundTruth} \times 100|$$

.

- Clip 1, 03m 47s, 5 participants
- Clip 2, 30m 38s, 5 participants
- Clip 3, 49m 40s, 8 participants

Figure 14: Screenshots: Clip 1, 03m 47s, 5 people

Figure 15: Screenshots: Clip 2, 30m 38s, 5 people

Figure 16: Screenshots: Clip 3, 49m 40s, 8 people

11.1 Clip 1

Table 10 shows the results from this clip. In addition, the following plots (Figures 17 and 18) show graphical analysis of the standard deviation and absolute percentage error of the estimated number of people (from the left) for varying interval lengths.

First note that this video clip is relatively short, spanning only about 4 minutes, and contains 5 people. For the simple averaging method, note that the most accurate results are observed with 60-second intervals for both sampling rates with the percentage error of 8%, and the estimation results are fairly consistent with the standard deviation of $.547722558$. Note also that the higher sampling rate of 15 fps does not display any clear indication of improved performance, compared to the results obtained at 1 fps sampling rate.

However, keep in mind that due to the presence of statistical algorithms, a sample of reasonable size is needed in order for the entire procedure to behave as it was originally designed to.

Interval Length (sec)	Estimated # People at Sampling Rate 1 fps *(Average)*	Estimated # People at Sampling Rate 15 fps *(Average)*
10	5	4
10	4	3
10	4	4
10	4	4
10	4	4
Average	4.2	3.8
Std Dev	0.447213595	0.447213595
% Error	-16%	-24%
30	7	4
30	6	4
30	6	4
30	7	4
30	6	4
Average	6.4	4
Std Dev	0.547722558	0
% Error	28%	-20%
60	6	5
60	5	5
60	5	6
60	6	6
60	5	5
Average	5.4	5.4
Std Dev	0.547722558	0.547722558
% Error	8%	8%
100	7	6
100	4	5
100	6	5
100	5	6
100	6	5
Average	5.6	5.4
Std Dev	1.140175425	0.547722558
% Error	12%	8%
Whole Clip	4	8
Whole Clip	2	11
Whole Clip	2	8
Whole Clip	4	9
Whole Clip	4	11
Average	3.2	9.4
Std Dev	1.095445115	1.516575089
% Error	-36%	-88%

Table 10: Result: Clip 1

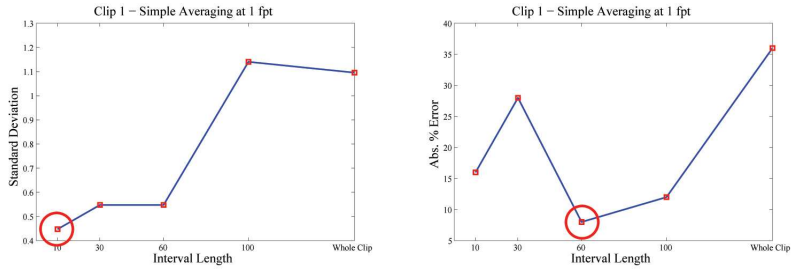


Figure 17: **Clip 1**, 03m 47s, **5 faces**, 1 fps - Std Dev and Absolute % Error

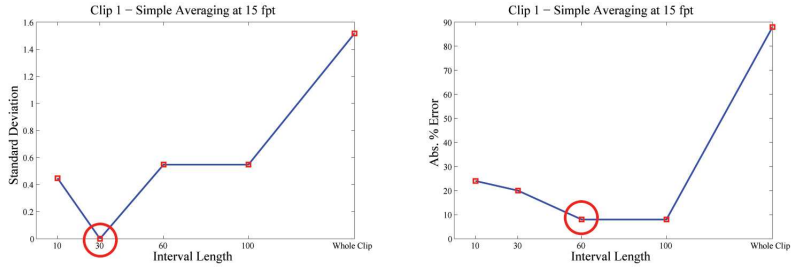


Figure 18: **Clip 1**, 03m 47s, **5 faces**, 15 fps - Std Dev and Absolute % Error

11.2 Clip 2

The improved verification procedure is performed this time with a larger clip, which is 30 minutes long. Table 11 shows the results from this clip. In addition, the following plots (Figures 19 and 20) show graphical analysis of the standard deviation and absolute percentage error of the estimated number of people (from the left) for varying interval lengths.

For the simple averaging method, note that the most accurate results are observed with 60-second intervals for the sampling rate of 1 fps and with 100-second intervals for 15 fps, and both are also the most precise results with the percentage error of 0%, as it appears in Figures 19 and 20. It is important to remark that now, with a larger set of data, it becomes clear that the absolute percentage error attains a local minimum (which is zero) at interval lengths of 60 and 100 seconds, where the standard deviation also remains 0. Also notice that, with 1 fps sampling rate, standard deviation becomes greater than zero and the absolute percentage error obtains the local minimum earlier (with respect to the interval length) than it is with 15 fps sampling rate.

11.3 Clip 3

Table 12 shows the results from this clip. In addition, the following plots (Figures 21 and 22) show graphical analysis of the standard deviation and absolute percentage error of the estimated number of people (from the left) for varying interval lengths.

Note that, once again, the standard deviation and absolute percentage error both

Interval Length (sec)	Estimated # People at Sampling Rate 1 fps *(Average)*	Estimated # People at Sampling Rate 15 fps *(Average)*
10	3	2
10	3	2
10	3	2
10	3	2
10	3	2
Average	3	2
Std Dev	0	0
% Error	-40%	-60%
30	4	3
30	4	3
30	4	3
30	4	3
30	4	3
Average	4	3
Std Dev	0	0
% Error	-20%	-40%
60	5	4
60	5	4
60	5	4
60	5	4
60	5	4
Average	5	4
Std Dev	0	0
% Error	0%	-20%
100	7	5
100	6	5
100	6	5
100	7	5
100	6	5
Average	6.4	5
Std Dev	0.547722558	0
% Error	28%	0%
Whole Clip	13	8
Whole Clip	14	5
Whole Clip	11	7
Whole Clip	14	6
Whole Clip	5	7
Average	11.4	6.6
Std Dev	3.78153408	1.140175425
% Error	128%	32%

Table 11: Result: Clip 2

Interval Length (sec)	Estimated # People at Sampling Rate 1 fps *(Average)*	Estimated # People at Sampling Rate 15 fps *(Average)*
10	6	5
10	6	5
10	6	5
10	6	5
10	6	5
Average	6	5
Std Dev	0	0
% Error	-25%	-37.5%
30	7	6
30	8	6
30	7	6
30	7	6
30	7	6
Average	7.2	6
Std Dev	0.447213595	0
% Error	-10%	-25%
60	8	7
60	8	7
60	8	7
60	8	7
60	8	7
Average	8	7
Std Dev	0	0
% Error	0%	-12.5%
100	9	7
100	8	7
100	9	7
100	9	7
100	8	7
Average	8.6	7
Std Dev	0.547722558	0
% Error	7.5%	-12.5%
Whole Clip	11	17
Whole Clip	16	18
Whole Clip	14	11
Whole Clip	15	15
Whole Clip	12	17
Average	13.6	15.6
Std Dev	2.073644135	2.792848009
% Error	70%	95%

Table 12: Result: Clip 3

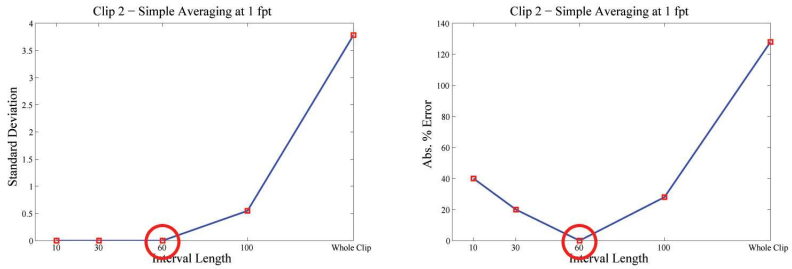


Figure 19: **Clip 2**, 30m 38s, **5 faces**, 1 fps - Std Dev and Absolute % Error

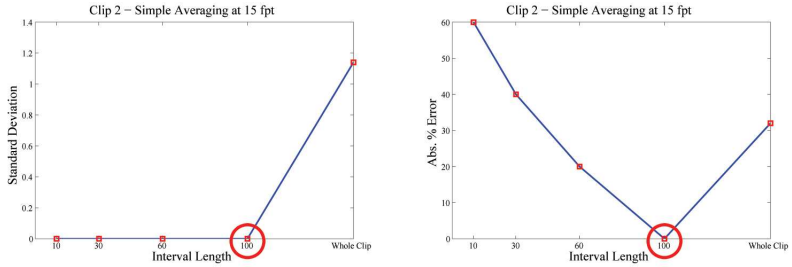


Figure 20: **Clip 2**, 30m 38s, **5 faces**, 15 fps - Std Dev and Absolute % Error

obtain local minima at the interval length of 60 seconds with 1 fps sampling rate; it indeed achieves 0 percentage error and 0 standard deviation simultaneously with simple averaging method. Although the results from 15 fps sampling rate, too, obtain local minima at 60-second interval length, its performance is inferior with -12.5% percentage error.

Part V
Conclusions and Future Works

We have thus far described an algorithm to automatically determine the number of people present at a meeting. As an alternative to the face detection approach of the current CAMEO project, the algorithm is a collection of statistical methods such as the EM and the K-Means algorithms combined together using a model verification procedure.

It operates on a set of face positions to estimate the number of distinct, recurring clusters of face positions while taking into account the presence of noise. Given that the ground truth for the data set is known (the true number of people present at the meeting is known), it is possible to evaluate the results of the algorithm and find the set of parameters that estimates the number with the lowest standard deviation and percentage errors.

With the improved model verification method, the algorithm estimates the number

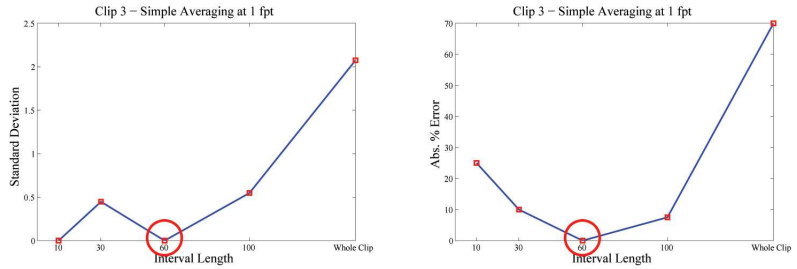


Figure 21: **Clip 3**, 49m 40s, **8 faces**, 1 fps - Std Dev and Absolute % Error

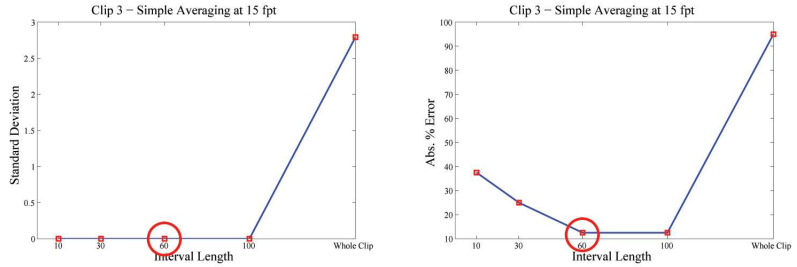


Figure 22: **Clip 3**, 49m 40s, **8 faces**, 15 fps - Std Dev and Absolute % Error

of people consistently with reasonable accuracy for a large set of data points for a given combination of parameters - 1 fps sampling rate and 60-second interval length in the above experiment - at which the standard deviation and the percentage error of the estimate attain local minima.

To put it differently, it is possible to obtain the optimal set of parameters to be used in future estimations in the training phase, in which the algorithm operates on the training data set for which the ground truth is known, by detecting the local minima of the standard deviation and the percentage errors of the estimates.

A stand-alone module for this purpose will implement an automatic, but simplified, training mode for model verification procedure by limiting the search space and adhering to 1 fps sampling rate and focusing mainly on the detection of such local minima in the percentage errors.

Ideas have been suggested for further noise filtering via examination of the y-coordinates, based on the observation that there are certain regions of each frame that are unlikely to contain any face. The exact locations of such regions are dependent on the specific orientation of the meeting and the camera. This technique may be implemented and experimented using different parameters in the future.

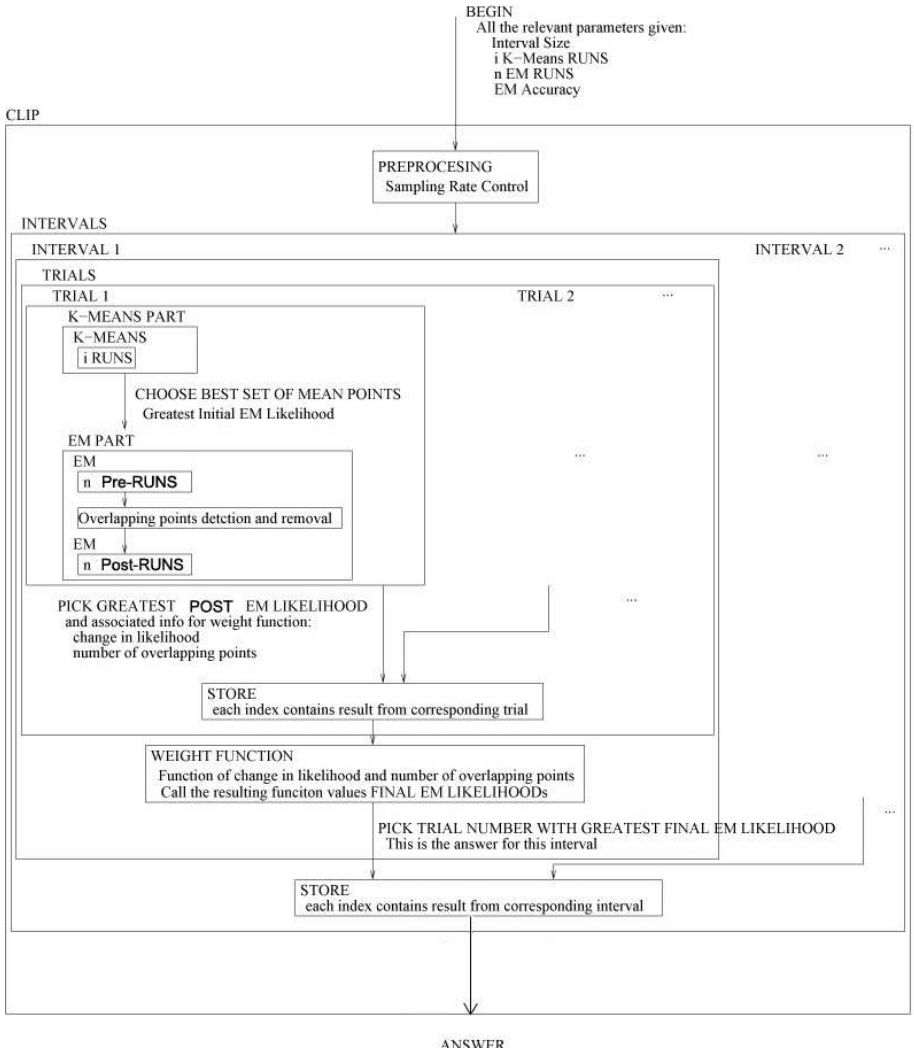


Figure 23: Flow of the algorithm

References

[1] S. Akaho. Em algorithm for mixture models. Internet, 2001.

[2] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, 2004.

[3] C. Elkan and G. Hamerly. Alternatives to the k-means algorithm that find better clusterings. tech. report CS2002-0702, Department of Computer Science and Engineering, University of California - San Diego, April 2002.

[4] A. P. Dempster et al. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, Series B, 39(1):1–38.

[5] F. De la Torre, C. Vallepsi, P. E. Rybski, M. Veloso, and T. Kanade. Omnidirectional video capturing, multiple people tracking and identification for meeting monitoring. tech. report CMU-RI-TR-05-04, Robotics Institute, Carnegie Mellon University, January 2005.

[6] Matteo Matteucci. Clustering - k-means demo. Internet, November 2004.

[7] Anny Lai mei Chiu and Ada Wai chee Fu. Enhancements on local outlier detection. In *Proceedings of the Seventh International Database Engineering and Applications Symposium*. IEEE, 2003.

[8] J. E. Moody and C. J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294.

[9] H. Schneiderman and T. Kanade. Probabilistic modeling of local appearance and spatial relationships for object recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 45–51. IEEE, 1998.