

# CM20220: Fundamentals of Pattern Analysis: GMM Classifier Coursework

## Plaragism Declaration

As per the coursework specification:

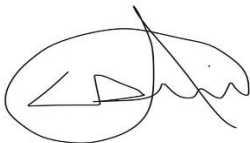
*“Plagiarism is the attempt to pass-off someone else’s work as your own, with the intention or expectation of receiving credit for it. It diminishes the degree you are on, the university you are in, and yourself; it is wholly unprofessional. Plagiarism can be intentional or un-intentional, but it is still plagiarism.*

*Regarding code. As mentioned above you are expected to write your code that relates directly to the classifier. However, we would prefer you to finish the coursework and so gain the experience of building and testing a classifier. So, if you do experience problems with coding be sure to let us know, and we may chose to let you use library code, or code we supply to you. You must not use another student’s code.*

*You must clearly declare – in your written document – any library code you have used that relates directly to the classifier; chain-code, Fourier transforms, and input/output functions excepted. If you are in any doubt, ask the unit leader.*

*The university web site explaining its position – and how to cite others so as to avoid plagiarism can be found at <http://www.bath.ac.uk/library/infoskills/>”*

I can confirm that all unaccredited MATLAB code that was not provided in course material such as lab sheets is my own work, however in cases where existing code is used from another source or third party, I have obtained permission and cited this source directly in my work. In cases where supporting code has been used from university tutors, as per the lab sheets, express permission to use this code is assumed. All sources used in the production of this report are cited in the bibliography. I am aware of the rules concerning plagarism.



Christopher Davies



# CM20220: Fundamentals of Pattern Analysis:

## GMM Classifier Coursework

### 1. Abstract

*This document demonstrates the stages involved in the development of a binary image shape classifier using MATLAB. More broadly, it describes pattern analysis –the classifier ‘learns’ a model from a large quantity of training data, assuming an underlying mathematical form, before testing the model to evaluate if the one chosen can make accurate predictions (‘inferences’) about a new piece of test data. Predictions are not certain meaning probabilities are used to reason about inferences made. In this paper, I explore the overall method in building and optimising the classifier, as well as potential improvements to the techniques used.*

### 2. Introduction to Theory

A **classifier** is a program that, when given an input, returns a label (a ‘class’). A class can be thought of as a grouping of datum which are logically related and share a common label. Ideally, the properties binding elements of a class together will not vary widely, yet will vary a lot between classes to make classes easy to distinguish. A classifier has 3 main stages: modelling, training and testing. Modelling involves describing features, choosing an appropriate statistical distribution and collecting data itself prior to the training stage. Once a model has been chosen, it is a challenge to find the best set of parameters for the model which best explain the data. Mathematically, a classifier is a function mapping the input feature vector  $\phi(X)$  to a class  $c$ , where the feature vector composes of some or all the data  $X$ . This means a **feature vector** is a tuple of values used to describe an object. A **confusion matrix** can assess a classifier’s accuracy- *more on this in the Method section*. **Modelling** assumes the data follows an underlying function or mathematical form. Using the overall training data itself as a feature vector is too sparse (does not allow for data to cluster enough to make confident assertions). This makes it difficult in cases where a piece of new test data does not correspond to any of the rows in the full joint distribution of random variables. A **model** is a compact representation allowing for assumptions about how data is distributed. Many distribution models exist – Bernoulli, Binomial, Multinomial, Gaussian, Uniform and so on, each with useful applications.

**Probabilistic classifiers** return the posterior probability of a given feature vector  $x$  belonging to a class  $\theta$  –  $p(\theta|x)$  –achieved by either directly learning the function that computes the posterior (**discriminative model**), or learning the class-conditional density  $p(x|\theta)$  for each  $\theta$ , and learning the class priors  $p(\theta)$  before applying Bayes’ rule to compute the posterior (**generative model**). **Parametric models** rely on a set of parameters and assume the data is generated from some probabilistic distribution. One such parametric model is a **Gaussian (or ‘Normal’) distribution** - a type of distribution whereby data adopts a ‘bell shaped’ curve - **See Appendix Fig 1d**. A Gaussian is parametrised by the **arithmetic mean** (‘average’ of all data points) and **covariance matrix** (a square  $(n \times n)$  matrix which characterises the relationship between all features with every other feature) of the data. The covariance matrix is symmetric - value  $(i,j)$  describes the relationship between features  $i$  and  $j$ , and is equal to value  $(j,i)$ . The diagonal captures the variance of a given feature. A high value on the diagonal of the covariance matrix for a given class means that for that class there is a lot of variance in that feature, suggesting that it might not be a very useful feature to use to classify that class. If its high for all classes, then it could be identified as a candidate for deletion from the feature vector. If using classes with fewer training data than  $N$  shapes, the covariance matrix will be rank deficient for classes with few training data. By increasing the very small (effectively zero) eigenvalues, the matrix is no longer rank deficient ensuring it is positive semi-definite. The Gaussian distribution generalises to  $N$  dimensions meaning Multivariate Gaussians are common. There are distinct types of classification: **supervised and unsupervised**. The former is where labels are supplied with the input training data, to describe the classes (or clusters) which the data is divided into. The latter is where no labels are given to the training data, and the number of clusters as well as which each data point is assumed to belong to is calculated. A supervised approach is arguably simpler and more common, but does require

human intervention in the form of labelling all training data. That said, unsupervised methods are more complex, but they do offer extra flexibility when the number of classes is unknown, or unbounded.

A **Gaussian Mixture Model (GMM)** can be thought of as a weighted sum of  $N$  multivariate Gaussians. The Gaussian components are scaled by weights, which sum to one - See **Appendix Fig 1e**. This can model the overall combined distribution for all classes in the training set, hence GMMs can be used for both supervised and unsupervised classification. The total set of parameters is decomposed into parts corresponding to different labels. For supervised classification, the number of Gaussians is equal to the number of distinct classes (labels) provided with the training data. The priors for each Gaussian is determined as the proportion of data points belonging to a given class. The total density of the data is expressed by summing the density of individual clusters (each modelled by their own Gaussian and weighted by the probability of that cluster). As such, when a new piece of test data arrives, it is possible to compare the likelihood of its feature vector belonging to any one of the existing clusters. The **prior probability** is the chance of seeing shape class:  $p(\theta_i) = \frac{N_i}{N}$  i.e. the number of shapes of that class, divided by the total number of shapes. If there is a large overlap between 2 given classes' Gaussians in the GMM and a lot of images misclassified as the other given class respectively (shown in the confusion matrix), it would suggest the method of feature extraction struggles to distinguish between these classes. The Multivariate Gaussian Distributions model the **Maximum Likelihood Estimation** – the probability density that a given class generates a given feature vector:  $p(x | \theta)$ . Ideally, we want to maximise the posterior probability, so we classify a new shape using **maximum a posteriori (MAP) classification** instead. For a new feature vector,  $x$ , we select the class,  $\theta$  which maximises the posterior  $p(\theta | x)$ . Bayes' rule can express this in terms of the likelihood,  $p(x | \theta)$ , and because we want the maximum value, we can remove terms which don't depend on  $\theta$ . The derivation of this equation is found in **Appendix, Fig 1f**. The **prior  $p(\theta)$**  can be thought of as the level of prior belief an image belongs to a given class – the proportion of images of that class in the training set. No matter what the specification is, the probability of all data can quickly approach zero, which in practice leads to **underflow errors**. In the case of a multivariate Gaussian, negative exponent means that probabilities become small. As such, the logarithm function is used, which avoids underflow and is monotonic, meaning it still preserves order between sets, since multiplications are replaced by additions and divisions are replaced by subtractions. **Alternatives to a supervised GMM approach** include supervised approaches such as Naïve Bayes Classification, Support Vector Machines (SVM) and k-Nearest Neighbour, or unsupervised approaches such as k-Means or GMM via Expectation Maximisation (EM). **Detail about the relative advantages and disadvantages of alternative methods to a supervised GMM can be found in Appendix Fig 1o and are discussed in the conclusion section of this report.**

Classifiers tend to **overfit** the training data in high dimensional spaces (i.e. when a large feature vector is chosen). The '**curse of dimensionality**' is where, as the dimensions of the feature space increase, the data becomes sparsely distributed within that space. The increase in feature space means a substantially larger set of training data is required. The way data distributes in a feature space is critical to the performance of a classifier, hence the classifier accuracy will decrease if an insufficient number of features is chosen. A **binary shape** is a white (or black) region in a picture of black and white pixels. The **chain code** of an image is the signal produced when an encoder traces its boundary. The directions moved are recorded as numbers which then form a signal. The chain code still enables the boundary to be reconstructed, but locational data is lost. The discrete **Fourier transform** transforms a signal from spatial (or time) domain to frequency domain, meaning a discrete image can be represented as the sum of sinusoids of various frequency coefficients ( $F[k] = \sum_{n=0}^{N-1} f[n]e^{-\frac{2\pi i n k}{N}}$  for  $k=0, 1, 2, \dots, N-1$ ). The Fourier transform contains both the amplitude and phase at each frequency, meaning it is periodic. The Fourier Transform can be used for **low pass filtering** which removes noise ('aliasing' - where small high frequency patterns appear as larger patterns, aliasing to low frequencies) in an image. Taking the absolute value removes complex arguments.

**Moments** describe numeric quantities at some distance from a reference point or axis (See **Appendix Fig 1h**). They summarise a shape, or more generally a continuous function  $f(x, y)$ . Given a scalar image  $I(x, y)$ , this can be approximated as:  $M_{ij} = \sum_x \sum_y x^i y^j I(x, y)$ . If  $I(x, y)$  is a binary image, then the 0<sup>th</sup> order moment is a count of the number, or area, of white pixels. This definition means the moment is sensitive of what the  $x$  and  $y$  values are. However, feature extraction needs to be addressed from the invariance property, irrespective of position and orientation (Rao, P et al. 2013). **Zernike Moments** offer rotation invariance meaning they allow for higher accuracy summarisation of detailed shapes, by centring an image on a disc at a fixed distance. They are also orthogonal which means there is less information redundancy - they can represent properties of an image with no redundancy or overlap of information between the moments. They project the image function onto orthogonal **Zernike polynomial** (set of orthogonal polynomials defined on the unit disk) basis functions, making them much more suited to image reconstruction than regular moments. That said, they are significantly dependent on the scaling and the translation of the object, but their magnitudes are independent of the rotation angle of the object. Scale and translation invariance are obtained by first normalising the image with respect to these parameters using its regular geometrical moments (Khotanzad A. et al. 1990). Taking the absolute value of Zernike Moments removes the complex argument.

**Principal Component Analysis (PCA)** and **t-distributed stochastic neighbour embedding (t-SNE)** are dimensionality reduction algorithms. t-SNE is generally better for visualisation but uses parameters and iterations hence scatters are stochastic and vary between attempts. t-SNE constructs a probability distribution over pairs of high-dimensional objects in such a way that similar objects have a high probability of being picked, whilst dissimilar points have an extremely small probability of being picked (LJP et al 2008, 2009, 2014). t-SNE builds a neighbourhood graph between feature vectors based on distance using a random walk, so plots can be likened to the London Underground map – they are topologically accurate and maintain local distances but lose information about the true distance between data points. Alternatively, PCA has no parameters and aims to capture the structure of data, but still suffers from a loss of information – as in a 2D floor plan of a building, for example, where distances between points locally are maintained but global distance information is still lost. Principal components are obtained by the eigen decomposition of the covariance matrix and dimensionality reduced by projecting the data onto the largest eigenvectors.

The **k-Means** algorithm is a way of attempting to find a user-specified number of clusters (**k**), which are represented by their centroids (normally the arithmetic mean of a cluster). For this reason, centroids don't usually correspond to an actual data point. The algorithm looks to minimise the inter-point distance in each cluster. More detail is in **Appendix, Fig 1k**. **Kullback–Leibler (KL) divergence** can be used to compare multivariate normal distributions for similarity, as well as being used for the objective function in t-SNE. See a detailed discussion in **Appendix, Fig 1i**. For a given data point, its **silhouette value**  $x$  ( $-1 \leq x \leq 1$ ) is a measure of how similar that point is to points in its own cluster, when compared to points in other clusters (Kaufman L et al. 1990). See a detailed discussion in **Appendix, Fig 1j**.

### 3. Method

*Initially the skeleton code supplied two libraries of binary shapes, a **training set** used to train the classifier and a **test set** to test the classifier's accuracy. The way the classifier was built is as follows:*

1. **Calculate the feature vectors** for all shapes of each class in the **training set**.
2. **Calculate the mean vector and covariance matrix** over all feature vectors in the **training set**.
3. Use the mean vector and covariance matrix as parameters to a **multivariate Gaussian distribution** for each class in the **training set** (this is the Maximum Likelihood value  $p(x|\theta)$  for parameters, however we want to classify by the MAP value  $p(\theta|x)$  instead).
4. **Model a GMM** based on the individual Gaussians for each class in the **training set**.
5. **For each shape in the test set:**
  - a. **Input a new test shape from the test set and calculate its feature vector.**

- b. Classify the shape as class,  $\theta$ , corresponding to the maximum:

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)} = \operatorname{argmax}_{\theta} p(x|\theta)p(\theta).$$

\*\*\* (we discard the  $p(x)$  term as it does not depend on  $\theta$ ) \*\*\*

- c. Increment the appropriate element of the confusion matrix.

6. Calculate the percentage of images classified in diagonal elements of the confusion matrix. This is the overall accuracy of the classifier (where an image of a given class has been correctly predicted as most likely to belong to that class).

## Step 1: Feature Vectors and Gaussian Distributions

A `getFeatures` function returns, as a row vector,  $N$  features from an input image - See Appendix Fig 1g. The feature vector needs to contain mostly useful information for classification with few redundancies - each dimension should be independent. Originally, code from Lab 3 extracted features from the input image by generating chain-codes for each image as in Appendix Fig 2h and features were extracted by taking the first  $N$  absolute values of the Fourier coefficients of the chain code with feature vectors of length 3. The lowest frequency (the zero frequency) was discarded. The absolute value is used to discard the complex argument. A variety of feature extraction methods were attempted to increase the accuracy of the classifier. Filtering 'noise' means the classifier is less likely to be too sensitive and misclassify an image. For example, two test "Star" images where one image has more jagged edges so may be misclassified if noise is not smoothed out during the extraction stage. Given that the boundary shape is smooth compared to at pixel level, a filter is required to remove higher frequency components. Low pass filtering creates a smoother signal by removing higher frequencies using the Fourier transform to filter in the frequency domain. This step also involved modelling a Gaussian for each class in the training set. The labelled set of training images was split into groups of the same class, and a Gaussian was modelled for each group. The whole training set (images) were input into the training script which calculated the mean and covariance for each class to model the parameters of each class' Gaussian distribution. Each Gaussian model gave a probability density of a feature vector given a class -  $p(x | \theta)$ . Originally, Feature Vector extraction and modelling Gaussians were tested visually – checking the output and ensuring that the scale and dimensions of the mean and covariance matrices made logical sense.

## Step 2: Modelling a Supervised Gaussian Mixture Model (GMM)

This involved combining the Gaussian for each class into a single GMM, using the proportion of images of that class in the training set as a prior for each one. See Appendix Fig 12a for the relevant code. Since data for each class was modelled as a multivariate Gaussian, the dimensionality of the data was too high to plot easily on a 2D figure. This meant that more complicated methods were required to reduce the number of dimensions of the data down to a reasonable number (i.e. 2D), making it easier to visualise the distribution of data for each class, requiring the use of PCA and t-SNE. The results of these plots are discussed in the results section.

## Step 3: Classifying Images with MAP Estimation

Once the supervised GMM had been trained and saved in 'models', each shape in the test set was compared against each class in the training set one by one to decide which class produced the maximum value of  $p(x|\theta)p(\theta) = p(\text{features}|\text{class})p(\text{class})$  – See Appendix Fig 12b for relevant code. The same method of feature extraction performed on the training set was now carried out on the test set. See Appendix Fig 1f for the MAP equation that the variable `score` uses to work out the class  $\theta$  that maximises this value. In this instance, we are looking for the value of `idx` which corresponds to the largest value of `score`. Then, the index of the class which maximises this value is returned (`bestidx`). This is the index of the class that the classifier has predicted most likely that the input image belongs to.



### Step 4: Testing Accuracy – the Confusion Matrix

To test the GMM had been trained correctly according to the training set provided, a confusion matrix **tested the quality of the classifier** - the chance that a test image belonging to a given class is correctly classified. **See Appendix Fig 12c for the relevant code.** This used the test set which was distinct from the training set, whilst keeping the same class divisions. Each row corresponded to each class in the test data, and each column corresponded to every unique predicted class. For each image in each class, a tally incremented the corresponding value of the confusion matrix based on the result of the classification. A new row for the next class began after all images of a given class had been classified. A diagonal matrix with only non-zero elements on the leading diagonal means every image has been correctly classified. In practice, images were misclassified meaning this was not the case. **See Appendix Fig 1a** for the original confusion matrix produced and **see Appendix Fig 12c for the relevant code.** As a metric to compare methods to optimise the classifier, I calculated the percentage of images classified in diagonal elements of the confusion matrix (where an image of a given class has been predicted as most likely to belong to that class). From this matrix, the percentage of correctly classified images from each class was also calculated. **Analysis of optimisation is in the results section.**

### Step 5: Optimising the Supervised GMM Classifier

The dimensionality of the data and feature extraction method was altered to optimise the classifier's accuracy. Feature extraction methods included the absolute values of the Fourier coefficients of an image's chain code (**See Appendix Fig 12d**), rotationally invariant Zernike Moments (**See Appendix Fig 12e**), binary image properties (**See Appendix Fig 12f**) and PCA dimensionality reduction (**See Appendix Fig 12g**). Optimisation of these methods mainly involved changing the number of features in the feature vector to see when the curse of dimensionality took hold / the optimum feature vector length occurred.

### Step 6: Unsupervised GMM using k-Means Clustering

Once optimised using a supervised GMM approach, I considered training an **unsupervised GMM**, namely via the k-Means clustering algorithm (**Appendix Fig 1k**), attempted using 4 feature extraction types, and comparing the accuracy of the unsupervised GMM to the supervised GMM just created.

1. Train the **supervised GMM** as before with a chosen feature extraction method on the **training set**.
2. Create a large data matrix of all features for all images in the **training set** (using the same extraction method as chosen above).
3. Run the k-Means algorithm on the set of all features to cluster data with  $k = 6$  (**See Appendix Fig 1k**).
4. Create a separate data matrix of features for each assigned k-Means cluster index and keep a count of the number of points belonging to each cluster. (**See Appendix Fig 12h (i) for relevant code**)
5. Calculate the mean and covariance matrix for each cluster's data matrix. Save an '*unsupervised\_models*' MATLAB struct to represent an unsupervised GMM (like the supervised GMM version '*models*'). **See Appendix Fig 12h (ii) for relevant code.**
6. Compare each cluster index's Gaussian distribution to each Gaussian distribution in '*models*' (the supervised GMM) using Kullback–Leibler divergence as a metric – note which supervised GMM class label is closest to the unsupervised cluster in question to give it a meaningful label. **See Appendix Fig 12h (iii) for relevant code.**
7. Classify each image in the **test set** according to the **closest label** of the cluster with the maximum '*score*' in the unsupervised GMM just created - create a **confusion matrix** based on this unsupervised model. **See Appendix Fig 12h (iv) for relevant code.**

### Step 7: Custom Test Images

A **custom test set** was experimented with to see how the classifier behaved when faced with images not belonging to any of the classes in the training set. Training using the supervised GMM meant it had training data for 6 classes – *Alien*, *Arrow*, *Butterfly*, *Face*, *Star*, *Toonhead*, but no data corresponding to other classes. The proportion of images of each class in the training and test data provided is very similar, so there is a real danger of '*over-fitting*'. If a new test set with the same class divisions but very different class representations was used, a class with small representation in the training set will have a low prior, so even with a sizeable proportion in the test set, the classifier's prior belief is that the class in question is unlikely to show up in test data and therefore less likely to be classified correctly unless the features chosen to distinguish this

class from the others effectively enough. I have been wary of this throughout. Test and training sets are distinct because performance on the training set only tells us that the model learns what it's supposed to learn- it may 'overfit' it rather than generalising to new data. The more training data, the more accurate the classifier due to more data being readily available to help distinguish between classes (Witten et al. 2000).

## 4. Results

See Appendix Fig 1m for a breakdown of the 3 image sets - the "training set" and "test set" were provided whereas the "custom set" was taken from the MPEG7 CE Shape-1 Part B database. Each classes' shapes vary due to a change of a viewpoint, distinct types of the same object (e.g. Alien), noise (e.g., digitization and segmentation) and non-rigid object motion (e.g. faces moving, stars rotating) (Latecki L.J et al. 2000). As such, the feature extraction stage should summarise the class independent of geometric transformations.

### Feature Extraction Method 1: Absolute Value of Fourier Coefficients of Chain Code

Rationale	This method of feature extraction takes the absolute values of $N$ Fourier Coefficients of the Chain Code produced from an input image. The contour is mapped to chain code, from which a shape descriptor is derived. The feature vector contains no redundancies as the signal contains no unnecessary duplicate data. It is also less sensitive to noise because the Fourier Transform applies a Low Pass Filter – the chain code has unwanted high frequency variants which are removed. Absolute locational data is lost because the signal is based on the path taken, meaning features are translationally invariant.
Tests	<ol style="list-style-type: none"> <li>1. The number of coefficients in the feature vector (<math>N</math>) is varied to determine the value of <math>N</math> yielding the maximum confusion matrix accuracy %.</li> <li>2. The <math>N</math> coefficients taken will be shifted along by various 'shifts' whilst keeping the value of <math>N</math> to determine the value of <math>Shift</math> yielding the maximum confusion matrix accuracy %.</li> <li>3. In code provided, the chain code searched for the first white spot in the image, however I experimented by altering the signal created by searching for the first black spot instead.</li> <li>4. Once optimised, a t-SNE plot is produced to visualise which classes were being confused.</li> </ol>
Findings	The confusion matrix and t-SNE plot (before optimisation) for 3 features with no shift [Appendix Fig 1a, 1b, 1c] showed an accuracy of 76.9231%. with a class accuracy distribution of {85.7143%, 0.0000%, 96.0000%, 66.6667%, 76.9231%, 0.0000%}, so 'Alien' and 'Butterfly' images were most accurately classified. 27/99 'Faces' are classified as 'Aliens'. 'Aliens' and 'Faces' are the most commonly predicted classes. Results were slightly lower in accuracy by finding the first black spot in the image, but the overall shape of the plot follows the same trend [Appendix Fig 2d, 2e]. The optimum number of features was 6 with no shift in Fourier spectrum and finding the first white spot. <b>Changing the part of spectrum extracted from whilst keeping the number of features constant lowered the accuracy no matter the number of features due to sampling higher frequency components more prone to noise</b> [Appendix Fig 2f, 2g]. The optimised confusion matrix and t-SNE plot [Appendix Fig 2a, 2b, 2c] showed an accuracy of 81.9005%. with a class accuracy distribution of {92.8571%, 0.0000%, 96.0000%, 73.7374%, 80.7692%, 0.0000%}. 'Aliens', 'Faces' and 'Stars' were classified with greater accuracy when optimised. 26/99 'Faces' are classified as 'Aliens'. 'Aliens' and 'Faces' are the most commonly predicted classes. Classes with small representations in the training set (and thus small priors) were misclassified no matter the length of the feature vector. The covariance matrices [Appendix Fig 1l] suggested the first feature (coefficient) showed the largest variance across all classes, suggesting it could be a candidate for deletion. A Gaussian instead of Top Hat filter yielded lower accuracy results.
Conclusion	The <b>curse of dimensionality</b> was demonstrated - too few features meant not enough "distinguishing information" was captured about each class – too many and the accuracy fell due to a lack of sufficient training data in this high dimensional sparse feature space. This method only works well with closed boundaries as it traces the outline – the chain code signal could be prematurely 'cut-off' by a discontinuous edge in an image. The zero frequency is not used in feature vectors because it does not specify anything more about the class – it is common to all meaning that classes are not distinguished using it. With chain code alone it is not possible to calculate image properties such as the size/area of the shape, allowing more features to specify more about the image – this is probably why a lot of distorted 'Faces' were classified as 'Aliens' before and still after optimisation and not vice versa, since the boundary of some images is so distorted due to noise that they do resemble Aliens visually – suggesting noise still has an effect even after filtering. The plot for shift in coefficients against accuracy decreases the accuracy because the higher frequency coefficients are more sensitive to noise (fine detail on a pixel basis). The Gaussian filter producing less accurate results suggests too much detail has been removed. t-SNE plots show impressive 'distinguishing' between class boundaries - even within clusters of different-class images, some images are correctly classified as being different despite having similar features. <b>Overall this method is not robust rotating shapes within the same class and so this is what I want to explore next.</b> Also, noise in the 'Faces' class causes images' boundaries to be summarised with a similar feature vector to, and consequently misclassified as, 'Aliens', backed up by the low KL divergence values between these classes' Gaussians in Appendix Fig 1i.

### Feature Extraction Method 2: Absolute Value of Zernike Moments (Degree 6)

Rationale	This method of feature extraction takes the absolute values of $N$ Zernike Moments produced from an input image. This offers rotation invariance meaning they allow for higher accuracy summarisation of detailed shapes such as 'Faces' which were commonly misclassified as 'Aliens' above, by centring an image on a disc at a fixed distance. They are also orthogonal which means they represent properties of an image with no redundancy or overlap of information between the moments, meaning feature vectors contain little redundant information. Taking the absolute value of Zernike Moments removes the complex argument and ensures rotational invariance.
Tests	<ol style="list-style-type: none"> <li>1. The number of coefficients in the feature vector (value of <math>N</math>) will be varied to determine the value of <math>N</math> yielding the maximum confusion matrix accuracy %.</li> <li>2. Once optimised, a t-SNE plot is produced to visualise which classes were being confused.</li> </ol>
Findings	The optimum number of features was 15 using degree 6 polynomial basis functions [Appendix Figure 3d]. The optimised confusion matrix and t-SNE plot [Appendix Fig 3a, 3b, 3c] showed an accuracy of 86.4253%. with a class accuracy distribution of {90.9091%, 100.0000%, 98.0000%, 97.9798%, 88.4615%, 0.0000%}, which is an improvement on Fourier for all classes except Aliens. <b>Compared to Fourier which misclassified a lot of Faces as Aliens, this is no longer the case due to rotational invariance. Now, however, less Aliens are classified accurately compared to Fourier - 19/42 'Aliens' are classified as 'Faces', presumably because before using Zernike, features relied on the difference in rotation of shapes between these two classes to distinguish class boundaries.</b> 'Faces' was by far the most commonly predicted class followed by 'Butterfly' due to them being the most represented in the training set and hence having the largest prior. Classes with small representations in the training set (and thus small priors) fared better compared to Fourier features – all 'Arrows' were classified correctly but 'Toonheads' were not no matter the length of the feature vector. <b>Toonhead was so far proving a difficult class to classify correctly using analytical methods such as chain code and Zernike Moments due to how similar the covariance matrices are to 'Face' and 'Star'.</b>
Conclusion	They Zernike Moments are significantly dependent on the scaling and the translation of the object which led to nearly half of the Aliens being misclassified as the class with the highest prior – 'Face', a definite loss of accuracy for this class compared to Fourier features. <b>That said, their magnitudes are independent of the rotation angle of the object meaning a lot less Faces are classed as Aliens as in the Fourier method, and all Arrows are now classified correctly due to rotation not being an issue.</b> The curse of dimensionality was again demonstrated - too few features meant not enough "distinguishing information" was captured about each class – too many and the accuracy fell due to a lack of sufficient training data in this high dimensional sparse feature space. t-SNE plots show impressive 'distinguishing' between class boundaries, particularly between Aliens and Butterflies - even within clusters of different-class images, some images are correctly classified as being different despite having similar features. <b>In the future, I would make this method scale and translation invariant by first normalising the image using its regular geometrical moments (Khotanzad A. et al. 1990). This change should optimise the classifier further. Next, I aim to try more of a feature engineering approach.</b>

### Feature Extraction Method 3: Black and White Image Properties

Rationale	This method of feature extraction takes important properties about the pixel values of the input image. This was <b>feature engineering</b> - the shape descriptor is a vector of parameters derived from the image instead of an analytical method such as chain code or Zernike moments, where it would be possible to vary $N$ over a greater range. The feature vector was [b w b/w w/b bwarea(image) bwarea(objects) aspRatio bwdistance] where image is the raw binary image, objects represents the convex hull image, aspRatio is the aspect ratio of the image, b represents the number of black pixels and w represents the number of white pixels. bwarea estimates the area of the objects in binary image whereas bwdistance computes the Euclidean distance transform of the image.
Test	<ol style="list-style-type: none"> <li>1. The number of coefficients in the feature vector (value of <math>N</math>) will be varied to determine the value of <math>N</math> yielding the maximum confusion matrix accuracy %.</li> <li>2. Once optimised, a t-SNE plot is produced to visualise which classes were being confused.</li> </ol>
Findings	Given that the maximum number of features for this extraction method is 8, I was a little restricted in analysing the importance of $N$ in this instance as I could only test from 1 to 8 features. The optimum number of features was 8 [Appendix Figure 4d]. The optimised confusion matrix and t-SNE plot [Appendix Fig 4a, 4b, 4c] showed an accuracy of 93.6652%. with a class accuracy distribution of {92.8571%, 100.0000%, 100.0000%, 90.9091%, 100.0000%, 0.0000%} which is a <b>drastic improvement such that all Arrows, Butterflies and Stars are classified correctly. Compared to Fourier, this method is equally as good for classifying Aliens, but considerably better than Zernike Moments, suggesting it is less susceptible to variance in translation and scaling.</b> 'Faces' was by far the most commonly predicted class followed by 'Butterfly' due to them being the most represented in the training set and hence having the largest prior. <b>'Faces' were the most 'confused' in the sense that some were classified as Aliens, Butterflies or even Stars – but looking at the test set and covariance matrices [Appendix Fig 6f], this seems to be because features vary for this class the most.</b> Classes with small representations in the training set (and thus small priors) fared better compared to Fourier features – all 'Arrows' were classified correctly but 'Toonheads' were not no matter the length of the feature vector.



Conclusion	Too few features meant not enough “distinguishing information” was captured about each class. The shape of the plot in Appendix Fig 4d suggests that adding some features such as <code>w/b</code> and <code>bwarea(objects)</code> can decrease the accuracy unless further features such as <code>bwarea(image)</code> and <code>aspRatio</code> are added respectively to specify more about the image and overcome redundant information. <b>Overall, this is the best method so far due having the best distribution of accuracy for each class.</b> t-SNE plots show impressive ‘distinguishing’ between class boundaries of ‘Alien’ and ‘Face’ - even within clusters of different-class images, some images are correctly classified as being different despite having similar features. <b>These features are mostly invariant of translation, and rotation, making them seemingly better choices than using Chain Code and Zernike Moments.</b>
------------	---

#### Feature Extraction Method 4: Principal Component Analysis (PCA)

Rationale	I decided to see if an approach which reduced the dimensions of the raw data would provide any more success. This technique reduced the dimensions of the image to $ND$ , then took the first $N$ principal components as features in the feature vector. This was inspired by research which suggested PCA was a common feature extraction method in data science and often used in facial recognition software because it constructs a smaller number of variables which still retain a substantial portion of the original information.
Tests	<ol style="list-style-type: none"> <li>1. The number of coefficients in the feature vector (value of <math>N</math>) will be varied to determine the value of <math>N</math> yielding the maximum confusion matrix accuracy %.</li> <li>2. Once optimised, a t-SNE plot is produced to visualise which classes were being confused.</li> </ol>
Findings	The optimum number of features was 15 [Appendix Figure 5d]. The optimised confusion matrix and t-SNE plot [Appendix Fig 5a, 5b, 5c] showed an accuracy of 93.6652%. with a class accuracy distribution of {97.6190%, 50.0000%, 100.0000%, 91.9192%, 92.3077%, 0.0000%}. <b>Compared to all other feature extraction methods this is the most successful in classifying Aliens – only 1 was misclassified, and equally as successful in classifying Butterflies as the binary image properties features.</b> It is comparatively worse to the previous method at classifying Arrows suggesting it is not as robust to rotated images. ‘Faces’ was by far the most commonly predicted class followed by ‘Butterfly’ due to them being the most represented in the training set and hence having the largest prior. Classes with small representations in the training set (and thus small priors) fared better compared to Fourier features – 1/2 ‘Arrows’ were classified correctly but ‘Toonheads’ were not no matter the length of the feature vector.
Conclusion	In practice, many features depend on each other or on an underlying unknown variable. A single feature could therefore represent a combination of multiple types of information by a single value. Removing such a feature would remove more information than needed. It turns out that removing too many eigenvectors removes valuable information from the feature space (not enough “distinguishing information” was captured about each class), whereas eliminating too few eigenvectors leaves us with the curse of dimensionality (the accuracy fell due to a lack of sufficient training data in this high dimensional sparse feature space). Although cross-validation techniques as in Appendix Fig 1n can obtain an estimate of the optimal number of dimensions, choosing this without over-fitting remains a problem. <b>In the future, I would check how much (as a percentage) of the variance of the original data is kept while eliminating eigenvectors to help assess this.</b> t-SNE plots show impressive ‘distinguishing’ between class boundaries for Alien and Faces <b>(which have been clustered together in most experiments so far)</b> - even within clusters of different-class images, some images are correctly classified as being different despite having similar features. <b>Robustness to rotation is not as good as with the binary image properties as shown - Arrows are classified with less accuracy. Despite having the same ‘overall’ accuracy as using binary image properties as features, the accuracy across all classes was better for the previous method, suggesting binary image properties is the best out of all attempted feature extraction methods.</b>

#### Unsupervised Classification using k-Means

Rationale	I chose k-Means to cluster data based on 4 feature extraction methods above, to compare which extraction method enabled data to be clustered into 6 classes most effectively, for an unsupervised GMM to be modelled, and compared with the supervised GMM with regards to its classification accuracy.
Tests	<ol style="list-style-type: none"> <li>1. Train the supervised GMM as before with each chosen feature extraction method on the training set and run the k-Means algorithm on the set of all features with <math>k = 6</math>.</li> <li>2. For each k-Means cluster produced, model a Gaussian and produce an unsupervised GMM.</li> <li>3. Compare each Gaussian in the unsupervised GMM to each Gaussian in the supervised GMM using Kullback–Leibler divergence – note which in supervised GMM’s class label is closest.</li> <li>4. Classify images in the test set according to the closest label of the cluster with the maximum ‘score’ in the unsupervised GMM - create a confusion matrix.</li> </ol>

Findings	<p>Accuracy was found to be low at times as k-Means isn't deterministic so will vary between runs – [Fig 6a, 7a, 8a, 9a]. For <b>Binary Image features</b> [6a] an overall accuracy of 69.6833% as Butterfly, Face and Star appeared to be distinct clusters with accuracies of {90.0000%, 94.9495%, 57.6923%}. K-Means appeared to cluster together and hence class data from the 'Alien' and 'Face' classes as the same 'cluster' [Appendix 6b, 6c] which is why the Alien label is not 'mapped to' using KL divergence and why the Silhouette plot has negative values in clusters 2 and 6 [Appendix 6d]. For <b>Fourier features</b> [7a] an overall accuracy of 47.9638% as Arrow, Butterfly, Face and Star appeared to be distinct clusters with accuracies of {100.0000%, 86.0000%, 37.3737%, 92.3077%}. K-Means appeared to cluster slightly more effectively than using binary properties [Fig 7b, 7c] but again the Alien label is not 'mapped to' using KL divergence which is why the Silhouette plot has negative values in clusters 1 and 4 where perhaps Aliens were 'misclustered' as Faces or Arrows [Fig 7d]. For <b>Zernike features</b> [8a] an overall accuracy of 80.0905% as Alien, Butterfly, Face and Star appeared to be distinct clusters with accuracies of {40.4762%, 82.0000%, 96.9697%, 88.4615%}. K-Means appeared to finally distinguish a cluster corresponding to the 'Alien' class [Fig 8b, 8c], however the Silhouette plot has negative values in clusters 1 and 2 suggesting Aliens and Faces are still confused [Fig 8d]. For <b>PCA features</b> [9a] an overall accuracy of 40.2715% as Arrow, Butterfly, Face and Toonhead appeared to be distinct clusters with accuracies of {100.0000%, 80.0000%, 46.4646%, 50.0000%}. Surprisingly, k-Means managed to distinguish clusters for the 'Arrow' and 'Toonhead' unlike previous methods [Fig 9b, 9c] and the Silhouette plot has fewer negative values in any clusters, suggesting this has clustered most effectively [Fig 9d].</p>
Conclusion	<p><b>The overall accuracy percentages are misleading as different classes have different proportions of images in the test set and so influence the overall percentage more, and not all classes were represented in each unsupervised GMM due to the way k-Means clustered the data as well as how the KL Divergence sometimes mapped more than one cluster to the same class label.</b> An alternative to using KL divergence would be to find the best-fitting permutation of the different supervised class labels which is computationally hard as there are <math>n!</math> permutations for <math>n</math> classes. In the future, I would use a different means of estimating the number of clusters, such as the 'gap statistic' (Tibshirani R. et al. 2001). This wouldn't be how unsupervised classification is done in industry as this method relied on supervised methods and knowledge about the training data to compare to. The disadvantage with k-Means is that the number of clusters is pre-determined and it does not guarantee a global maximum. <b>In future would use a different unsupervised method (such as expectation maximisation) or experiment more with the number of clusters, increasing the number of clusters until there is at least 1 cluster mapping to all six classes in the training set.</b> k-Means effectively picks separate ways to create class boundaries which shows that statistically these are the 'best' class boundaries, not necessarily the expected ones. Clustering does without about labels so judging accuracy requires other metrics which I would calculate given more time (e.g. inter-cluster-variation, intra-cluster-variation, purity).</p>

### Custom Test Images

Rationale	<p>To see how the classifier behaved when faced with images not belonging to any of the classes in the training set, this experiment allows analysis on how similar the characteristics of 'foreign' classes (not used in training data) were to existing classes. The confusion matrix was not a metric I could use quantifying success as the way it had been implemented relied on supervised class indexes – using a totally distinct set of classes in the test data meant I instead had to use a tally-based method [See Appendix 10a, 10b, 10c. 10d].</p>
Tests	<ol style="list-style-type: none"> <li><b>The custom test set is input into the supervised GMM classifier using each type of feature extraction method described above.</b></li> <li><b>Tally tables of the “predicted classes vs actual class in the test data” are produced.</b></li> </ol>
Findings	<p>Using Fourier features [Fig 10b], images were classified as Alien the most for all test classes, with Bats and Elephants showing almost as many Butterfly predictions as Alien. Using Zernike features [Fig 10c], Apples were mainly classed as Faces, Bats mainly Butterfly, Elephants were confused between Alien and Face. Using Image Property features [Fig 10a], most Apple, Bat, Cup, Elephant and Tree images were classified as Butterfly, with Hearts instead entirely classified as Aliens. Using PCA features [Fig 10d], Hearts were most distinguished, being classified as entirely Aliens.</p>
Conclusion	<p>For the feature extractions which rely on contour shapes, it was good to see that Bats were mainly predicted as Butterfly boundaries look visually very similar. The image property features seemed to class Hearts as Aliens due to the Aspect Ratio being similar to Alien test images. This stage emphasised the importance of having a large training set of many different classes whilst avoiding the curse of dimensionality. By having a small number of images of a certain class relative to the size of the training set, the prior probability for these classes are so small that the classifier is unlikely to correctly classify a new image belonging to that class. This meant the classes with small priors such as Arrow and Toonhead were very rarely distinguished and classified, whereas the likes of Alien and Butterfly were predicted many times.</p>

I attempted to produce meaningful plots of each class' covariance matrix's confidence/error ellipses (**Appendix Fig 11a**) to analyse the covariance between features (represented by the angle of the ellipse) and visualise the proportion of points within a confidence interval, however due to dimensionality reduction to 2D, too much meaningful information was lost – one way to get ameliorate this would be to plot all ellipses on a common axis for visual comparison, but this proved difficult. I also plotted a probability density estimation of the features for each class using MATLAB's `ksdensity` function (**Appendix Fig 11b**) to visualise whether the distribution followed the Gaussian 'bell curve' – distributions for Arrow and Toonhead suggested these did, but it would be nice to visualise on the same plot in the future.

## 5. Summary and Conclusion

I have created and optimised a binary image GMM classifier. Feature extraction create a feature vector for a given input image, then every image belonging to each class within the labelled training set has its features extracted and a Gaussian distribution is modelled for each class – a probability density function  $p(x|\theta)$ . A supervised GMM is constructed combining the Gaussians for each class in the training set with a prior for each Gaussian based on the class' representation in the training set. Images are classified using MAP estimation based on this GMM and a confusion matrix produced as a metric for comparing the accuracy. An unsupervised GMM is modelled using k-Means clustering to see how effectively training data with no labels is clustered using each feature extraction method, comparing the Gaussian for each cluster to each Gaussian in the supervised GMM and producing a meaningful cluster label. Finally, a custom test set belonging to none of the classes in the training set was tested.

I have learnt a lot building the classifier. First, the GMM model assumes the shape of the distribution is Gaussian, however in reality data will have a more complex distribution (Yegnanarayana et al. 2002). Whilst the supervised approach is simpler, it requires human intervention to label training data. With unsupervised methods, I found it harder to evaluate 'accuracy' or 'effectiveness' of clustering which is why comparison with the supervised model was used, but this approach offers flexibility when the number of classes is unknown. Visualising high dimensional data has proven to be difficult throughout experiments - t-SNE plots are stochastic and vary between runs despite being better for visualisation. Changing the number of features  $N$  has a profound effect on the accuracy – too few features didn't distinguish classes well enough causing low accuracy, too many required more training data than was available which also caused low accuracy. Whenever an image belonging to 'class 1' was misclassified as 'class 2' above, this is backed up by the relevant **Kullback–Leibler Divergence** value – the value of  $D_{KL}(\text{Class 1} || \text{Class 2})$  was generally smaller than  $D_{KL}(\text{Class 1} || \text{Class } n)$  where  $n \neq 2$ . **See Appendix 1i for all KL divergence results.** This meant that the Gaussians for class 1 and class 2 overlapped to some extent, meaning the chosen feature extraction did not separate the Gaussians for these classes enough to distinguish them effectively. If the divergence was small between two classes, I found that generally the two were confused in the confusion matrix. The higher the KL divergence between 2 classes, generally the better these two classes were distinguished. The best accuracy results across all classes (without over-fitting to the training data provided) was achieved by **feature engineering** - constructing application-dependent features with no redundancies – as such, **the binary image feature method tended to distinguish classes the best** according to its KL divergence results, and hence produced the best overall accuracy distribution across all the classes. Features sometimes depended on each other to boost overall accuracy - whilst `bwarea(objects)` alone decreased the accuracy, adding `aspRatio` to specify more about the image increased the accuracy even further than before. Dimensionality reduction such as PCA for feature extraction did help, because analysis with many variables generalises poorly to new samples. I also learnt the importance of using shape descriptors invariant to geometric transformations. Some "difficult" images may belong to different classes dependent on transformation – see **Appendix Fig 13a**. I have also noted that the **"overall accuracy" statistics need deeper analysis** to optimise the classifier - the overall percentage is misleading as different classes have different proportions of images in the test set and so their success may influence the overall percentage more than other classes.

Improvements to the adopted method include increasing the size of the training set to overcome the curse of dimensionality. Class imbalance in the training set is ameliorated by using priors, but for small data sets such as this, to help combat 'overfitting' to training sets if priors dominate (a problem I encountered), sometimes **k-Fold Cross-validation** is used to maximise use of the data – see **Appendix Fig 1n**. I would adopt this approach given more time and plot a graph of 'number of training images used' against the resultant 'classifier accuracy' for several types of feature extraction. I would also consider adopting a true **unsupervised GMM approach with EM** because mapping clusters to a 'closest' label as attempted above meant not all training class labels were represented in the unsupervised GMM, resulting in a low accuracy confusion matrix – I would evaluate cluster quality using **purity** given more time. Also, this wouldn't be how unsupervised classification is done in industry as this method relied on supervised methods and knowledge about the training data to compare to.

**As such, whilst the GMM classifier has been built and optimised with some success and interesting results along the way, there is still great scope for improvement in the future.**

## Bibliography

- [1] A. Tahmasbi, F. Saki, S. B. Shokouhi, Classification of Benign and Malignant Masses Based on Zernike Moments, *Comput. Biol. Med.*, vol. 41, no. 8, pp. 726-735, 2011.
- [2] Boland, M. (1999). *Zernike Feature Extraction and Image Reconstruction*. [online] Murphylab.web.cmu.edu. Available at: [http://murphylab.web.cmu.edu/publications/boland/boland\\_node32.html](http://murphylab.web.cmu.edu/publications/boland/boland_node32.html) [Accessed 14 Apr. 2017].
- [3] Chai, Y. (2011). *Recognition between a Large Number of Flower Species*. [online] Available at: [http://www.robots.ox.ac.uk/~vgg/research/flowers\\_demo/docs/Chai11.pdf](http://www.robots.ox.ac.uk/~vgg/research/flowers_demo/docs/Chai11.pdf) [Accessed 18 Apr. 2017].
- [4] Domingos, P. and Pazzani, M., 1997. *On the optimality of the simple Bayesian classifier under zero-one loss*. *Machine learning*, 29(2-3), pp.103-130.
- [5] F. Saki, A. Tahmasbi, H. Soltanian-Zadeh, S. B. Shokouhi, *Fast opposite weight learning rules with application in breast cancer diagnosis*, *Comput. Biol. Med.*, vol. 43, no. 1, pp. 32-41, 2013.
- [6] G. C. Cawley and N. L. C. Talbot, *Over-fitting in model selection and subsequent selection bias in performance evaluation*, *Journal of Machine Learning Research*, 2010. Research, vol. 11, pp. 2079-2107, July 2010.
- [7] Kaufman L., and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Hoboken, NJ: John Wiley & Sons, Inc., 1990.
- [8] Khotanzad A., Hong Y., 'Invariant Image Recognition by Zernike Moments' *IEEE Transactions on Pattern Analysis And Machine Intelligence*. Vol. 12. No. 5. May 1990
- [9] L.J.P. van der Maaten and G.E. Hinton. *Visualizing High-Dimensional Data Using t-SNE*. *Journal of Machine Learning Research* 9(Nov):2579-2605, 2008.
- [10] L.J.P. van der Maaten, E.O. Postma, and H.J. van den Herik. *Dimensionality Reduction: A Comparative Review*. Tilburg University Technical Report, TiCC-TR 2009-005, 2009.
- [11] L.J.P. van der Maaten. *Accelerating t-SNE using Tree-Based Algorithms*. *Journal of Machine Learning Research* 15(Oct):3221-3245, 2014.
- [12] L.J.P. van der Maaten. (c2017). *Matlab Toolbox for Dimensionality Reduction*. [online] Available at: <https://lvdmaaten.github.io/drtoolbox/> [Accessed 14 Apr. 2017].
- [13] Latecki L.J, Lakamper R., Eckhardt U. (2000). 'Shape Descriptors for Non-Rigid Shapes with a Single Closed Contour'. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 424-429
- [14] Mahi H, Isabaten H, Serief C. 'Zernike Moments and SVM for Shape Classification in Very High Resolution Satellite Images' *The International Arab Journal of Information Technology*, Vol. 11, No. 1, January 2014, pp 43.
- [15] Mitchell, Tom. M. 1997. *Machine Learning*. New York: McGraw-Hill
- [16] MPEG7 CE Shape-1 Part B Image Database. [online] Available at: [http://www.imageprocessingplace.com/root\\_files\\_V3/image\\_databases.htm](http://www.imageprocessingplace.com/root_files_V3/image_databases.htm) [Accessed 20 Apr. 2017]
- [17] Murphy K.P. (2006). 'Naïve Bayes classifiers' pp 1-2
- [18] Rao, P., Prasad, D. and Kumar, Ch. (2013). 'Feature Extraction Using Zernike Moments' Vol. 2 Issue 2 March 2013, pp 228.
- [19] Spruyt, V. (2014). 'How to draw an error ellipse representing the covariance matrix?'. [online] Available at: <http://www.visiondummy.com/2014/04/draw-error-ellipse-representing-covariance-matrix/> [Accessed 14 Apr. 2017]
- [20] Spruyt, V. (2014). 'Feature extraction using PCA'. [online] Available at: <http://www.visiondummy.com/2014/05/feature-extraction-using-pca/> [Accessed 14 Apr. 2017]
- [21] Tan, P.N., Steinbach, M. and Kumar, V., 2013. *Data mining cluster analysis: basic concepts and algorithms*. *Introduction to data mining*. Chapter 8, pp. 496-498
- [22] Tibshirani R., Walther G., Hastie T., (2001). 'Estimating the number of clusters in a data set via the gap statistic' *Stanford University. Royal Statistical Society*. Part 2, pp. 411-423.
- [23] Witten, Ian H., and Eibe Frank. 2000. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. San Diego, CA: Morgan Kaufmann.
- [24] Wolf C, Taylor G, Jolion JM. *Learning individual human activities from short binary shape sequences*. Technical Report RR-LIRIS-2011-018 Laboratoire d'Informatique en Images et Systèmes d'Information (LIRIS), INSA de Lyon, France, October 2011.
- [25] Wolf C., *Matlab code for Zernike moments*. [online] Available at: <http://liris.cnrs.fr/christian.wolf/software/zernike/> [Accessed 14 Apr. 2017].
- [26] Yegnanarayana, B. and Kishore, S. (2002). *AANN: an alternative to GMM for pattern recognition*. *Neural Networks*, 15(3), pp.459-469.



## Appendix

Figure 1a: Original Supervised Confusion Matrix (Fourier Before Optimisation)

Number of Features: 3

Training Directory: './images/train' (Provided)

Testing Directory: './images/test' (Provided)

Feature Type: Absolute Values of Fourier Coefficients of Chain Code

Shift in Fourier Spectrum (from N lowest frequencies): 0

		PREDICTED CLASSES						TOTAL
		Alien	Arrow	Butterfly	Face	Star	Toonhead	
ACTUAL CLASSES	Alien	36	0	4	1	1	0	42
	Arrow	0	0	2	0	0	0	2
	Butterfly	2	0	48	0	0	0	50
	Face	27	0	6	66	0	0	99
	Star	2	0	4	0	20	0	26
	Toonhead	2	0	0	0	0	0	2
	TOTAL	69	0	64	67	21	0	221

% Accuracy (Aliens): 85.7143%

% Accuracy (Arrow): 0.0000%

% Accuracy (Butterfly): 96.0000%

% Accuracy (Face): 66.6667%

% Accuracy (Star): 76.9231%

% Accuracy (Toonhead): 0.0000%

% Accuracy (Overall): 76.9231%

Figure 1b: Original Predicted Classifications t-SNE Plot (Fourier Before Optimisation)

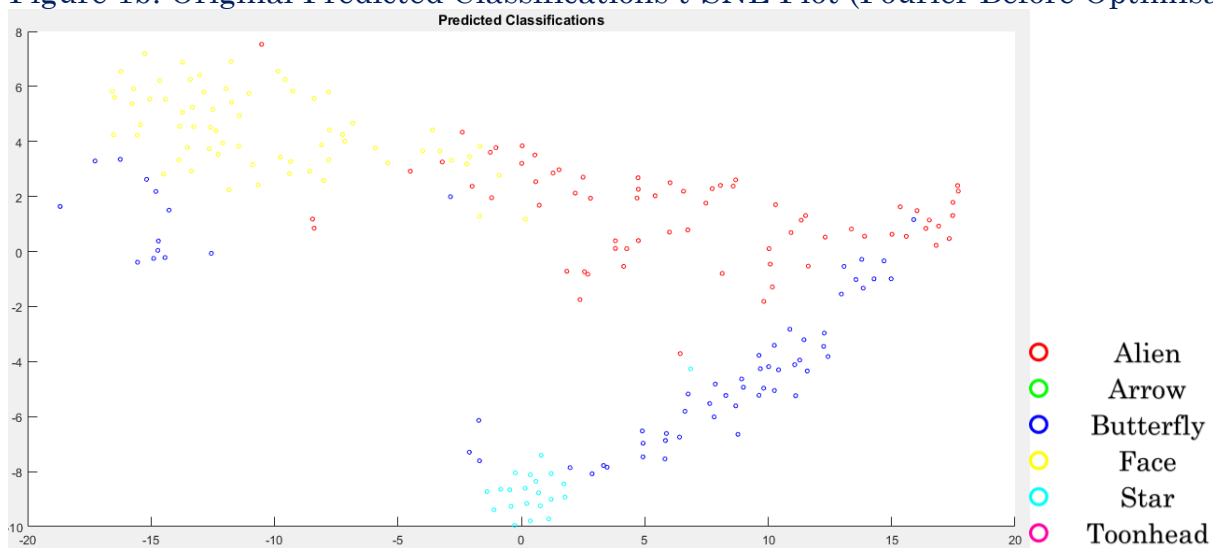




Figure 1c: Actual Classifications t-SNE Plot (Fourier Before Optimisation)

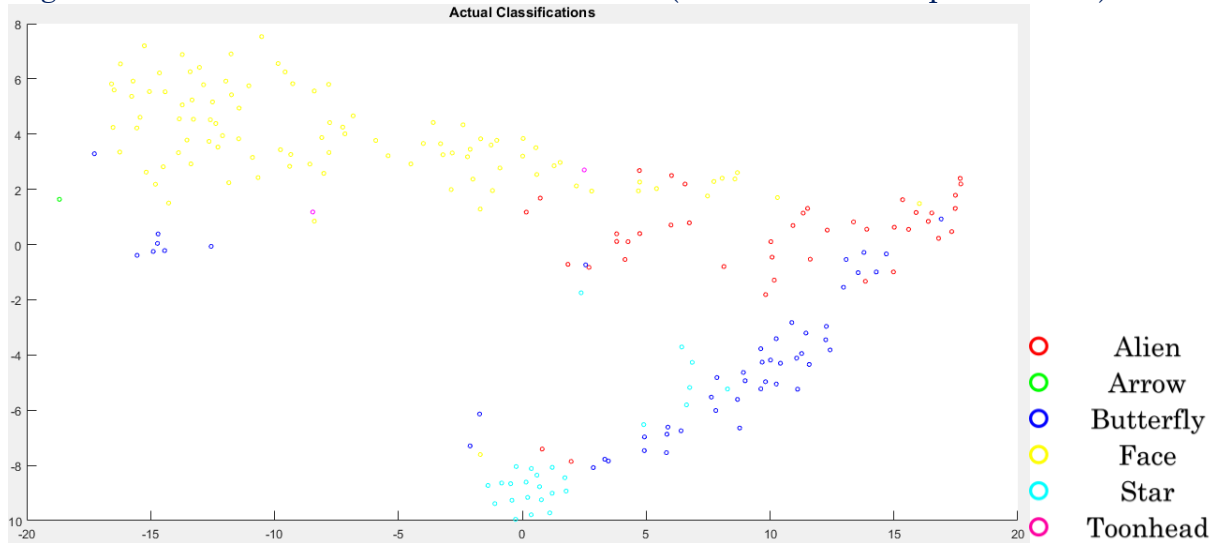


Figure 1d: 2D Gaussian Distribution

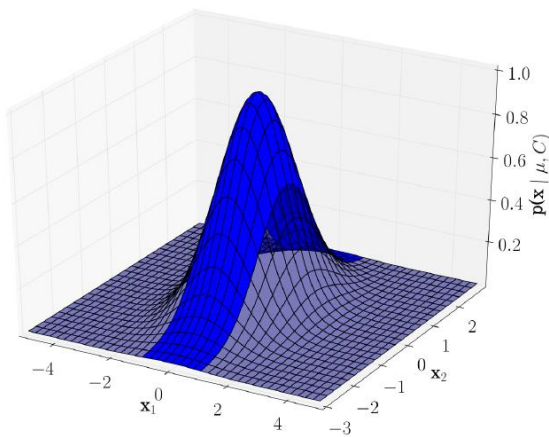


Figure 1e: Gaussian Mixture Model (GMM)

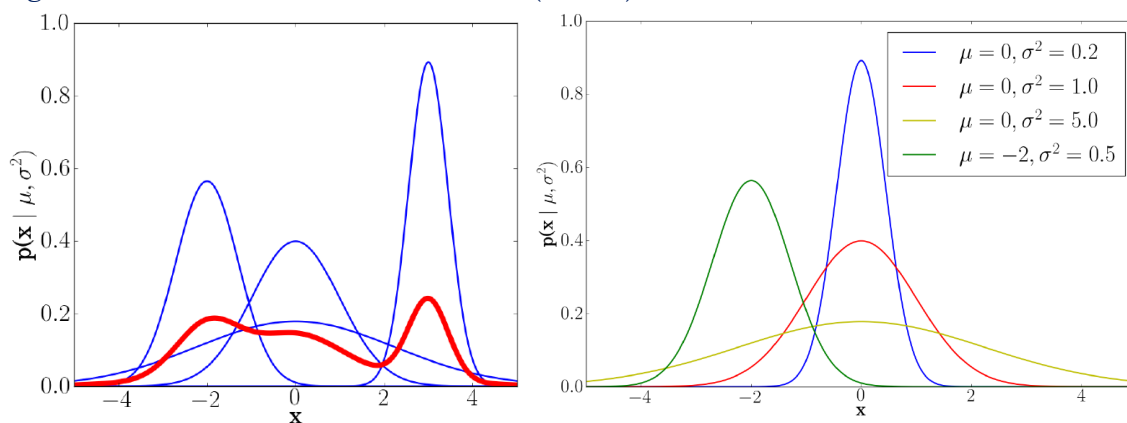


Figure 1f: Maximum A Posteriori (MAP) Equation Derivation

Maximum Likelihood  $p(\mathbf{x} | \theta_i)$ :

$$p(\mathbf{x} | \theta_i) \triangleq \frac{1}{(2\pi)^{K/2} |\mathbf{C}_i|^{1/2}} \exp \left( -\frac{1}{2} (\mathbf{x} - \mu_i)^T \mathbf{C}_i^{-1} (\mathbf{x} - \mu_i) \right)$$

**Maximum A Posteriori  $p(\theta_i | \mathbf{x})$ :**

$$\begin{aligned}\arg \max_{\theta} p(\theta | \mathbf{x}) &= \arg \max_{\theta} \frac{p(\mathbf{x} | \theta) p(\theta)}{p(\mathbf{x})} \\ &= \arg \max_{\theta} p(\mathbf{x} | \theta) p(\theta)\end{aligned}$$

$$\begin{aligned}\arg \max_{\theta} p(\theta | \mathbf{x}) &= \arg \max_{\theta} \log (p(\mathbf{x} | \theta) p(\theta)) \\ &= \arg \max_{\theta} [\log (p(\mathbf{x} | \theta)) + \log (p(\theta))]\end{aligned}$$

$$\arg \max_{\theta} p(\theta | \mathbf{x}) = \arg \max_{\theta} [\log(p(\theta)) - 0.5 \log(|\mathbf{C}_{\theta}|) - 0.5(\mathbf{x} - \mu_{\theta})^T \mathbf{C}_{\theta}^{-1}(\mathbf{x} - \mu_{\theta})]$$

Figure 1g: Feature Extraction of a Binary Image

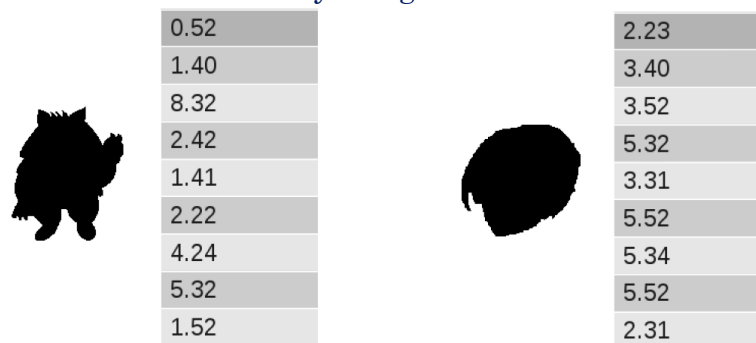


Figure 1h: General Moments Equation

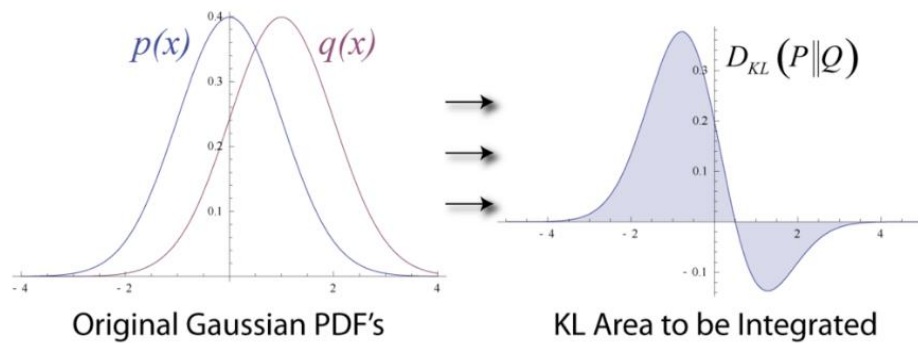
$$M_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y) dx dy$$

Figure 1i: Kullback–Leibler Divergence Equation for multivariate normal distributions

$$D_{\text{KL}}(\mathcal{N}_0 \| \mathcal{N}_1) = \frac{1}{2} \left( \text{tr}(\Sigma_1^{-1} \Sigma_0) + (\mu_1 - \mu_0)^T \Sigma_1^{-1} (\mu_1 - \mu_0) - k + \ln \left( \frac{\det \Sigma_1}{\det \Sigma_0} \right) \right)$$

- $D_{\text{KL}}(\mathcal{N}_0 \| \mathcal{N}_1)$  = KL divergence value between distribution  $\mathcal{N}_0$  and distribution  $\mathcal{N}_1$
- $\text{tr}(\cdot)$  = Matrix Trace (sum of diagonal elements)
- $\Sigma_0, \Sigma_1$  = Covariance matrices for distribution  $\mathcal{N}_0$  and distribution  $\mathcal{N}_1$  respectively.
- $\mu_0, \mu_1$  = Arithmetic mean vector for distribution  $\mathcal{N}_0$  and distribution  $\mathcal{N}_1$  respectively.
- $k$  = Dimensionality of data (i.e. number of features)
- $\det(\cdot)$  = Matrix determinant

The equation therefore gives a result measured in ‘nats’ (unit of information based on logarithm base e instead of base 2 [‘bits’])



### Kullback-Leibler Divergence between Gaussians – Fourier (6 Features)

#### Class: Alien

- $D_{KL}(\text{Alien} || \text{Arrow}) = 12.0723$
- $D_{KL}(\text{Alien} || \text{Butterfly}) = 5.2341$
- $D_{KL}(\text{Alien} || \text{Face}) = 6.6690$
- $D_{KL}(\text{Alien} || \text{Star}) = 3.2874$
- $D_{KL}(\text{Alien} || \text{Toonhead}) = 29.9308$

#### Class: Arrow

- $D_{KL}(\text{Arrow} || \text{Alien}) = 2.9230\text{e}+07$
- $D_{KL}(\text{Arrow} || \text{Butterfly}) = 8.6709\text{e}+06$
- $D_{KL}(\text{Arrow} || \text{Face}) = 1.1941\text{e}+07$
- $D_{KL}(\text{Arrow} || \text{Star}) = 2.4736\text{e}+07$
- $D_{KL}(\text{Arrow} || \text{Toonhead}) = 1.8742\text{e}+07$

#### Class: Butterfly

- $D_{KL}(\text{Butterfly} || \text{Alien}) = 26.0202$
- $D_{KL}(\text{Butterfly} || \text{Arrow}) = 2.0469$
- $D_{KL}(\text{Butterfly} || \text{Face}) = 4.1180$
- $D_{KL}(\text{Butterfly} || \text{Star}) = 14.4520$
- $D_{KL}(\text{Butterfly} || \text{Toonhead}) = 9.3590$

#### Class: Face

- $D_{KL}(\text{Face} || \text{Alien}) = 132.2096$
- $D_{KL}(\text{Face} || \text{Arrow}) = 322.5044$
- $D_{KL}(\text{Face} || \text{Butterfly}) = 127.3800$
- $D_{KL}(\text{Face} || \text{Star}) = 72.6798$
- $D_{KL}(\text{Face} || \text{Toonhead}) = 14.6207$

#### Class: Star

- $D_{KL}(\text{Star} || \text{Alien}) = 76.8020$
- $D_{KL}(\text{Star} || \text{Arrow}) = 203.2458$
- $D_{KL}(\text{Star} || \text{Butterfly}) = 68.0646$
- $D_{KL}(\text{Star} || \text{Face}) = 94.7515$
- $D_{KL}(\text{Star} || \text{Toonhead}) = 96.2027$

#### Class: Toonhead

- $D_{KL}(\text{Toonhead} || \text{Alien}) = 1.2392\text{e}+07$
- $D_{KL}(\text{Toonhead} || \text{Arrow}) = 2.7393\text{e}+07$
- $D_{KL}(\text{Toonhead} || \text{Butterfly}) = 1.9035\text{e}+07$
- $D_{KL}(\text{Toonhead} || \text{Face}) = 1.4448\text{e}+06$
- $D_{KL}(\text{Toonhead} || \text{Star}) = 6.7953\text{e}+06$

### Kullback-Leibler Divergence between Gaussians – Zernike (15 Features)

#### Class: Alien

- $D_{KL}(\text{Alien} || \text{Arrow}) = 35.6625$
- $D_{KL}(\text{Alien} || \text{Butterfly}) = 1.1755$
- $D_{KL}(\text{Alien} || \text{Face}) = 3.3414$

- $D_{KL}(\text{Alien} || \text{Star}) = 9.7921$
- $D_{KL}(\text{Alien} || \text{Toonhead}) = 2.6076$

**Class: Arrow**

- $D_{KL}(\text{Arrow} || \text{Alien}) = 45.1416$
- $D_{KL}(\text{Arrow} || \text{Butterfly}) = 48.5419$
- $D_{KL}(\text{Arrow} || \text{Face}) = 40.1120$
- $D_{KL}(\text{Arrow} || \text{Star}) = 37.9755$
- $D_{KL}(\text{Arrow} || \text{Toonhead}) = 42.6426$

**Class: Butterfly**

- $D_{KL}(\text{Butterfly} || \text{Alien}) = 1.3505$
- $D_{KL}(\text{Butterfly} || \text{Arrow}) = 33.5019$
- $D_{KL}(\text{Butterfly} || \text{Face}) = 8.8245$
- $D_{KL}(\text{Butterfly} || \text{Star}) = 19.9021$
- $D_{KL}(\text{Butterfly} || \text{Toonhead}) = 5.7813$

**Class: Face**

- $D_{KL}(\text{Face} || \text{Alien}) = 5.9805$
- $D_{KL}(\text{Face} || \text{Arrow}) = 37.5532$
- $D_{KL}(\text{Face} || \text{Butterfly}) = 7.9740$
- $D_{KL}(\text{Face} || \text{Star}) = 5.7595$
- $D_{KL}(\text{Face} || \text{Toonhead}) = 4.7796$

**Class: Star**

- $D_{KL}(\text{Star} || \text{Alien}) = 21.5832$
- $D_{KL}(\text{Star} || \text{Arrow}) = 36.8394$
- $D_{KL}(\text{Star} || \text{Butterfly}) = 24.9012$
- $D_{KL}(\text{Star} || \text{Face}) = 6.6655$
- $D_{KL}(\text{Star} || \text{Toonhead}) = 16.3677$

**Class: Toonhead**

- $D_{KL}(\text{Toonhead} || \text{Alien}) = 0.9752$
- $D_{KL}(\text{Toonhead} || \text{Arrow}) = 33.4204$
- $D_{KL}(\text{Toonhead} || \text{Butterfly}) = 2.1569$
- $D_{KL}(\text{Toonhead} || \text{Face}) = 0.0380$
- $D_{KL}(\text{Toonhead} || \text{Star}) = 2.8494$

**Kullback–Leibler Divergence between Gaussians – Binary Image Properties (8 Features)****Class: Alien**

- $D_{KL}(\text{Alien} || \text{Arrow}) = 7.5075\text{e}+03$
- $D_{KL}(\text{Alien} || \text{Butterfly}) = 52.1091$
- $D_{KL}(\text{Alien} || \text{Face}) = 11.1517$
- $D_{KL}(\text{Alien} || \text{Star}) = 4.6933$
- $D_{KL}(\text{Alien} || \text{Toonhead}) = 8.5493$

**Class: Arrow**

- $D_{KL}(\text{Arrow} || \text{Alien}) = 7.4384\text{e}+10$
- $D_{KL}(\text{Arrow} || \text{Butterfly}) = 6.7548\text{e}+10$
- $D_{KL}(\text{Arrow} || \text{Face}) = 9.6831\text{e}+10$
- $D_{KL}(\text{Arrow} || \text{Star}) = 5.5304\text{e}+10$
- $D_{KL}(\text{Arrow} || \text{Toonhead}) = 9.9127\text{e}+10$

**Class: Butterfly**

- $D_{KL}(\text{Butterfly} || \text{Alien}) = 90.1021$
- $D_{KL}(\text{Butterfly} || \text{Arrow}) = 2.3398\text{e}+03$
- $D_{KL}(\text{Butterfly} || \text{Face}) = 57.8471$
- $D_{KL}(\text{Butterfly} || \text{Star}) = 6.3235$
- $D_{KL}(\text{Butterfly} || \text{Toonhead}) = 18.1320$

**Class: Face**

- $D_{KL}(\text{Face} || \text{Alien}) = 304.4125$
- $D_{KL}(\text{Face} || \text{Arrow}) = 1.3916\text{e}+04$

- $D_{KL}(\text{Face}||\text{Butterfly}) = 746.9875$
- $D_{KL}(\text{Face}||\text{Star}) = 185.7297$
- $D_{KL}(\text{Face}||\text{Toonhead}) = 411.0724$

**Class: Star**

- $D_{KL}(\text{Star}||\text{Alien}) = 1.3561\text{e}+03$
- $D_{KL}(\text{Star}||\text{Arrow}) = 1.5774\text{e}+04$
- $D_{KL}(\text{Star}||\text{Butterfly}) = 576.3384$
- $D_{KL}(\text{Star}||\text{Face}) = 159.0304$
- $D_{KL}(\text{Star}||\text{Toonhead}) = 963.4133$

**Class: Toonhead**

- $D_{KL}(\text{Toonhead}||\text{Alien}) = 6.1254\text{e}+09$
- $D_{KL}(\text{Toonhead}||\text{Arrow}) = 1.8180\text{e}+10$
- $D_{KL}(\text{Toonhead}||\text{Butterfly}) = 1.4897\text{e}+09$
- $D_{KL}(\text{Toonhead}||\text{Face}) = 5.0564\text{e}+09$
- $D_{KL}(\text{Toonhead}||\text{Star}) = 1.8021\text{e}+08$

**Kullback–Leibler Divergence between Gaussians – PCA (15 Features)****Class: Alien**

- $D_{KL}(\text{Alien}||\text{Arrow}) = 25.8577$
- $D_{KL}(\text{Alien}||\text{Butterfly}) = 7.6499$
- $D_{KL}(\text{Alien}||\text{Face}) = 106.7580$
- $D_{KL}(\text{Alien}||\text{Star}) = 30.6776$
- $D_{KL}(\text{Alien}||\text{Toonhead}) = 22.5128$

**Class: Arrow**

- $D_{KL}(\text{Arrow}||\text{Alien}) = 1.8562\text{e}+03$
- $D_{KL}(\text{Arrow}||\text{Butterfly}) = 5.2659\text{e}+03$
- $D_{KL}(\text{Arrow}||\text{Face}) = 6.7682\text{e}+03$
- $D_{KL}(\text{Arrow}||\text{Star}) = 2.7101\text{e}+03$
- $D_{KL}(\text{Arrow}||\text{Toonhead}) = 5.2929\text{e}+03$

**Class: Butterfly**

- $D_{KL}(\text{Butterfly}||\text{Alien}) = 3.7673\text{e}+03$
- $D_{KL}(\text{Butterfly}||\text{Arrow}) = 55.4194$
- $D_{KL}(\text{Butterfly}||\text{Face}) = 2.5400\text{e}+03$
- $D_{KL}(\text{Butterfly}||\text{Star}) = 209.4882$
- $D_{KL}(\text{Butterfly}||\text{Toonhead}) = 10.2569$

**Class: Face**

- $D_{KL}(\text{Face}||\text{Alien}) = 1.9567\text{e}+03$
- $D_{KL}(\text{Face}||\text{Arrow}) = 29.3867$
- $D_{KL}(\text{Face}||\text{Butterfly}) = 58.9514$
- $D_{KL}(\text{Face}||\text{Star}) = 48.3946$
- $D_{KL}(\text{Face}||\text{Toonhead}) = 27.0621$

**Class: Star**

- $D_{KL}(\text{Star}||\text{Alien}) = 2.0079\text{e}+03$
- $D_{KL}(\text{Star}||\text{Arrow}) = 14.5048$
- $D_{KL}(\text{Star}||\text{Butterfly}) = 118.8074$
- $D_{KL}(\text{Star}||\text{Face}) = 487.6756$
- $D_{KL}(\text{Star}||\text{Toonhead}) = 285.7007$

**Class: Toonhead**

- $D_{KL}(\text{Toonhead}||\text{Alien}) = 9.9438\text{e}+03$
- $D_{KL}(\text{Toonhead}||\text{Arrow}) = 5.7102\text{e}+03$
- $D_{KL}(\text{Toonhead}||\text{Butterfly}) = 2.4253\text{e}+04$
- $D_{KL}(\text{Toonhead}||\text{Face}) = 4.4526\text{e}+03$
- $D_{KL}(\text{Toonhead}||\text{Star}) = 1.6221\text{e}+04$



### Figure 1j: Silhouette Value Equation

The **silhouette value** for the  $i$ th point,  $S_i = \frac{(b_i - a_i)}{\max(a_i, b_i)}$

- $a_i$  = average distance from  $i$ th point to other points in the same cluster as  $i$
- $b_i$  = minimum average distance from  $i$ th point to points in different cluster, minimized over clusters.

A high  $S_i$  indicates data point  $i$  is well-matched to its own cluster, and poorly-matched to neighbouring clusters. If most points have a high  $S_i$ , the clustering solution is appropriate. If many points have a low / negative  $S_i$ , the clustering solution has too many / too few clusters.

### Figure 1k: k-Means Algorithm

**The k-Means Algorithm looks to minimise the inter-point distance in each cluster.** Ideally, data points will be far apart, but for small  $k$ , the algorithm can cope with extremes. It is assumed that the data points  $\mathbf{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  (of any dimension) are provided as well as the number of clusters  $k$ .

It is desirable to minimise the objective function  $J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \mu_k\|^2$  (the sum of squared distances of each point from centroid  $\mu_k$ , a point which represents each cluster centre).

1. **Choose  $K$  points as centroids at a random  $\mu_k$**
2. **Label each point:**

$$r_{nk} = \begin{cases} 1 & \text{if } k = \operatorname{argmin}_j \|\mathbf{x}_n - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

i.e. for each point  $\mathbf{x}_n$  in training data, create a binary vector with an element for each cluster  $k$ , signifying whether point belongs to a given cluster.

3. **Update centroids  $\mu_k$ :**

Objective function is a quadratic function of  $\mu_k$ , so minimised by setting derivative with respect to  $\mu_k$  to zero:

$$2 \sum_{n=1}^N r_{nk} (\mathbf{x}_n - \mu_k) = 0$$

Solves to give:

$$\mu_k = \frac{\sum_n r_{nk} \mathbf{x}_n}{\sum_n r_{nk}}$$

4. **Loop from step 2 until labelling converges.**

Value of  $J$  decreases with each step, so k-Means guaranteed to find a minimum might be local minimum not global

*Tan, P.N et al. 2013.*

### Figure 1l: Original Supervised GMM Model (Fourier – 3 Features)

models				
1x6 struct with 4 fields				
Fields	name	mean	cov	prior
1	'Alien'	[1.4249e+03...	[5.0275e+04...	0.1892
2	'Arrow'	[645.0737;1...	[5.9310e+04...	0.0135
3	'Butterfly'	[1.2262e+03...	[3.5376e+04...	0.2252
4	'Face'	[968.9064;3...	[2.5576e+03...	0.4505
5	'Star'	[1.1329e+03...	[7.4648e+03...	0.1081
6	'Toonhead'	[1.1367e+03...	[1.4644e+04...	0.0135

**Covariance Matrix – Alien:**

1.0e+04 \*  
 5.0275 -0.2520 0.3342  
 -0.2520 0.1546 -0.0618  
 0.3342 -0.0618 0.3742

**Covariance Matrix – Arrow:**

1.0e+04 \*  
 5.9310 -0.3220 0.4779  
 -0.3220 0.7748 0.1861  
 0.4779 0.1861 0.0979

**Covariance Matrix – Butterfly:**

1.0e+04 \*  
 3.5376 0.0418 0.0527  
 0.0418 0.0883 -0.0054  
 0.0527 -0.0054 0.0781

**Covariance Matrix – Face:**

1.0e+03 \*  
 2.5576 0.1270 0.3923  
 0.1270 0.1171 0.0382  
 0.3923 0.0382 0.2738

**Covariance Matrix – Star:**

1.0e+03 \*  
 7.4648 0.7909 1.1941  
 0.7909 0.2320 0.1123  
 1.1941 0.1123 0.4954

**Covariance Matrix – Toonhead:**

1.0e+04 \*  
 1.4644 0.3133 -0.0932  
 0.3133 0.0682 -0.0139  
 -0.0932 -0.0139 0.0358

Figure 1m: Training Set vs Test Set vs Custom Set Breakdown

Training Set			Test Set		
Class	#	Proportion	Class	#	Proportion
Alien	42	42/223 = <b>0.1883</b>	Alien	42	42/221 = <b>0.1900</b>
Arrow	3	3/223 = <b>0.0135</b>	Arrow	2	2/221 = <b>0.0090</b>
Butterfly	50	50/223 = <b>0.2242</b>	Butterfly	50	50/221 = <b>0.2262</b>
Face	100	100/223 = <b>0.4484</b>	Face	99	99/221 = <b>0.4480</b>
Star	24	24/223 = <b>0.1076</b>	Star	26	26/221 = <b>0.1176</b>

<b>Toonhead</b>	3	$3/223 = 0.0135$	<b>Toonhead</b>	2	$2/221 = 0.0090$
<b>TOTAL</b>	<b>223</b>	<b>100%</b>	<b>TOTAL</b>	<b>221</b>	<b>100%</b>

Custom Set		
Class	#	Proportion
Apple	3	$3/28 = 0.1071$
Bat	5	$5/28 = 0.1786$
Cup	4	$4/28 = 0.1429$
Elephant	4	$4/28 = 0.1429$
Heart	6	$6/28 = 0.2143$
Tree	6	$6/28 = 0.2143$
<b>TOTAL</b>	<b>28</b>	<b>100%</b>

MPEG7 CE Shape-1 Part B Image Database. [online] Available at:

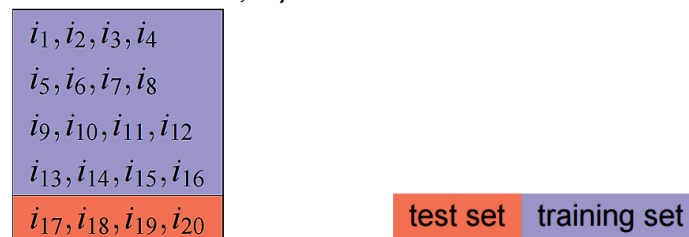
[http://www.imageprocessingplace.com/root\\_files\\_V3/image\\_databases.htm](http://www.imageprocessingplace.com/root_files_V3/image_databases.htm) [Accessed 20 Apr. 2017]

### Figure 1n: k-Fold Cross Validation Algorithm

1. Divide data randomly into  $k$  folds (subsets) of equal size.
2. Train the model on  $k - 1$  folds, use one-fold for testing.
3. Repeat this process  $k$  times so that all folds are used for testing.
4. Compute the average performance (*classifier accuracy*) on the  $k$  test sets.

**This effectively uses all the data for both training and testing. Typically,  $k = 10$  is used.**

*Example: data set with 20 instances, 5-fold cross validation*



- Compute accuracy for each fold then compute average accuracy (*Mitchell, T. 1997*)

### Figure 1o: Alternatives to a GMM Classifier

- **Supervised - k-Nearest-Neighbours (kNN)** - one of the most primitive classifiers as it is non-parametric – we only need to know the value for  $k$ . During training time, each feature vector from the training images is simply mapped onto the feature space. And for each test image, the "nearest  $k$ -neighbours" from the training samples in the feature space are found and the test image takes the label of most of the neighbours. A distance function compares samples for similarity. It is robust to noisy training data but costly as we need to compare to every sample so far. (*Chai, Y. 2011*).
- **Supervised - Naïve Bayes Classification** – very strong assumption about data distribution – that all features are statistically independent given a class. However, the classifier can be optimal even if the assumption is violated (*Domingos and Pazzani 1997*). For any possible value

of a feature, a likelihood value is obtained by a frequentist approach resulting in probabilities going towards 0 or 1, leading to numerical instabilities and worse results.

- **Supervised - Support Vector Machines (SVM)** –It has a regularisation parameter, which makes the user think about avoiding over-fitting. It uses the kernel trick, so you can build in expert knowledge about the problem via engineering the kernel. However, SVM moves the problem of over-fitting from optimising the parameters to model selection. (*G.C. Cawley et al. 2010*).
- **Unsupervised - k-Means** - one of the simplest unsupervised algorithms to solve clustering with a known number of clusters. There is always one or more data points per cluster. It is sensitive to outliers / noise and the number of clusters need to be specified beforehand. It is hard to compare the quality of clusters produced because different initial assignments or values of K affect the outcome. (*Tan, P.N et al. 2013*)
- **Unsupervised - GMM via Expectation Maximisation (EM)** - use the current labelling as a basis for selecting a better labelling by computing the posterior probability that a given datum belongs to a given component. If data in any one class are not Gaussian distributed, such a cluster cannot be properly approximated. Also, data clusters overlap making it difficult to separate them. The EM algorithm does not guarantee a global maximum - each time EM is run we may get different GMMs. Finally, the algorithm assumes the number of clusters is given but ideally, an algorithm will determine this automatically.

Figure 2a: Supervised Confusion Matrix (Optimised Fourier)

Number of Features: 6

Training Directory: './images/train' (Provided)

Testing Directory: './images/test' (Provided)

Feature Type: Absolute Values of Fourier Coefficients of Chain Code

		PREDICTED CLASSES						
		Alien	Arrow	Butterfly	Face	Star	Toonhead	TOTAL
ACTUAL CLASSES	Alien	39	0	3	0	0	0	42
	Arrow	0	0	2	0	0	0	2
	Butterfly	2	0	48	0	0	0	50
	Face	26	0	0	73	0	0	99
	Star	5	0	0	0	21	0	26
	Toonhead	2	0	0	0	0	0	2
	TOTAL	74	0	53	73	21	0	221

% Accuracy (Aliens): 92.8571%

% Accuracy (Arrow): 0.0000%

% Accuracy (Butterfly): 96.0000%

% Accuracy (Face): 73.7374%

% Accuracy (Star): 80.7692%

% Accuracy (Toonhead): 0.0000%

% Accuracy: 81.9005%

Figure 2b: Predicted Classifications t-SNE Plot (Optimised Fourier)

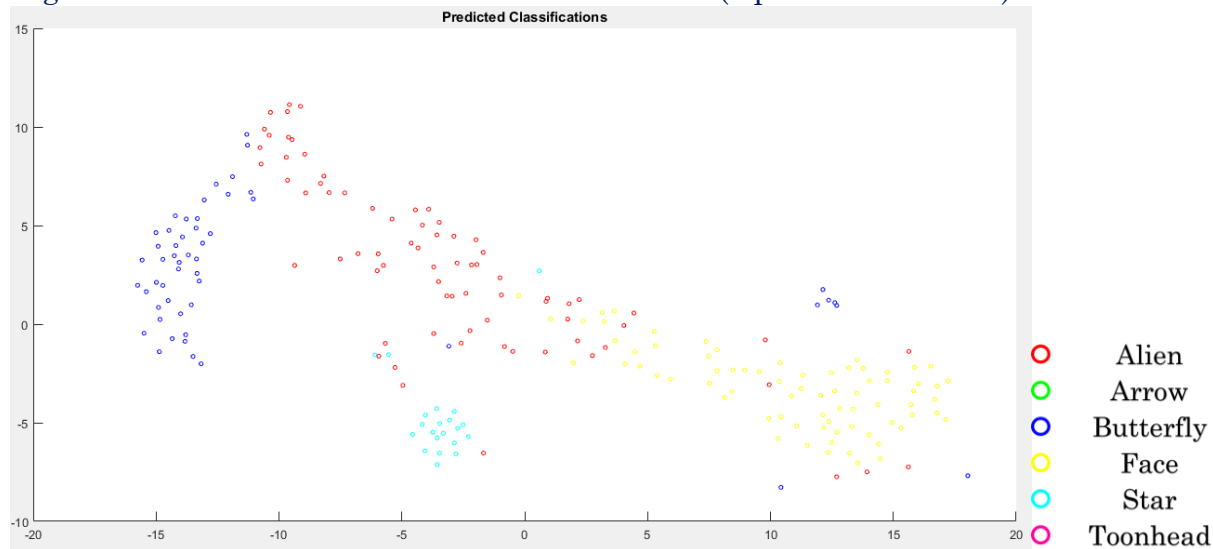


Figure 2c: Actual Classifications t-SNE Plot (Optimised Fourier)

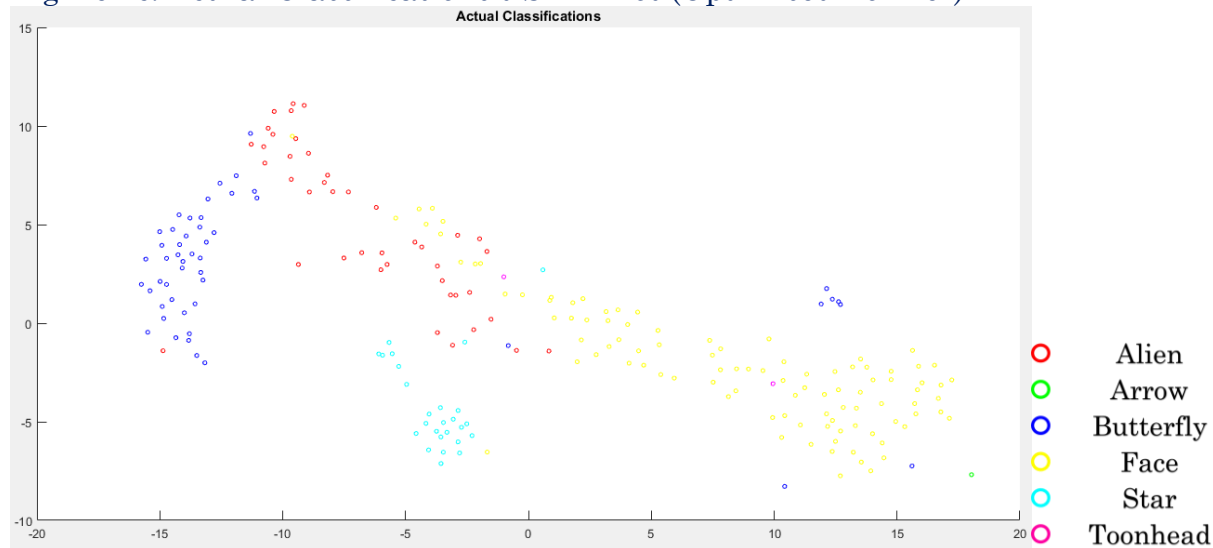




Figure 2d: Fourier Transform Feature Extraction Plot Using findBlackSpot.m

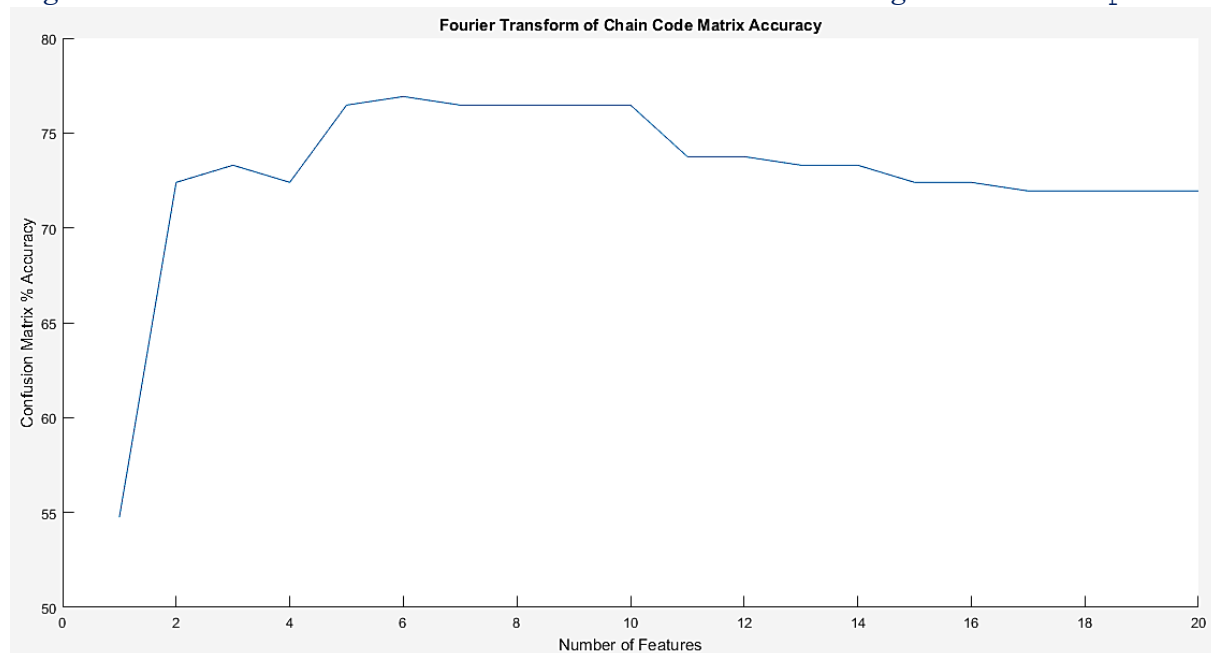


Figure 2e: Fourier Transform Feature Extraction Plot Using findWhiteSpot.m

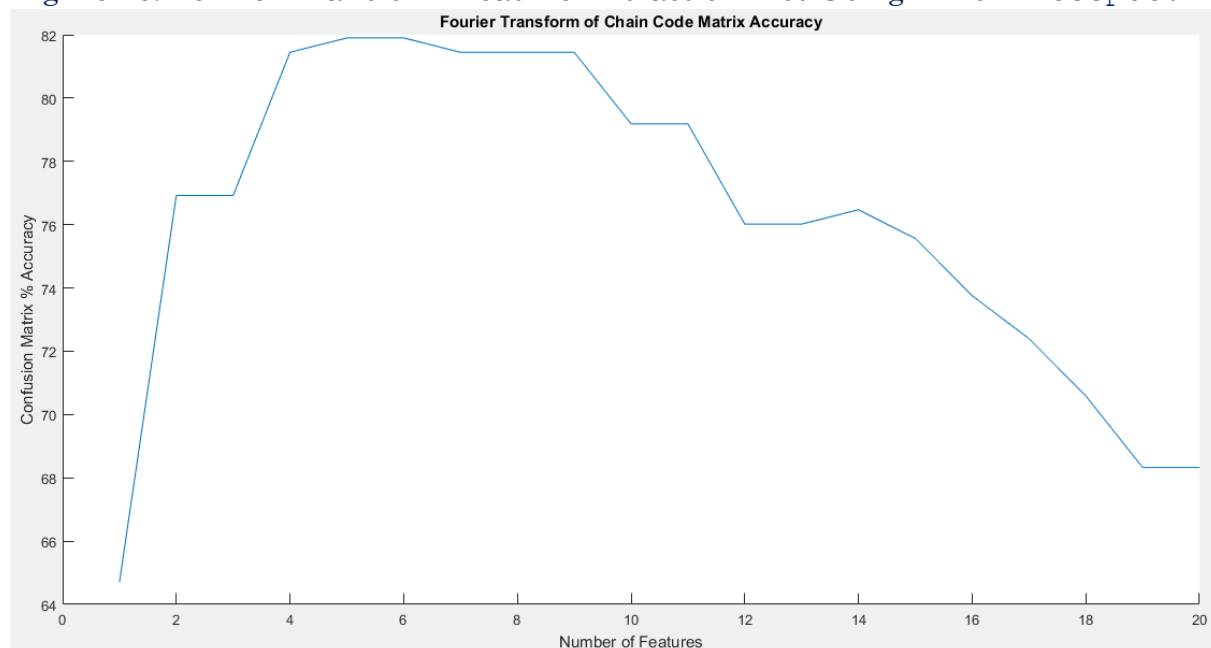


Figure 2f: Shift in Fourier Spectrum vs Confusion Matrix Accuracy (6 Features):

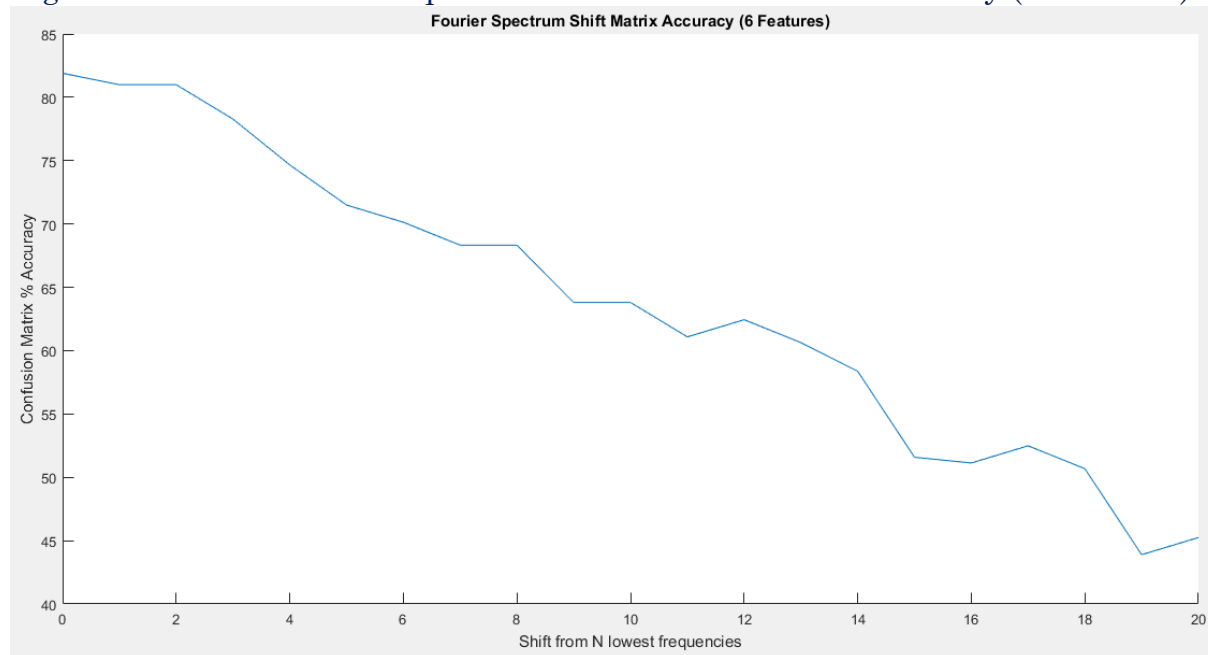


Figure 2g: Shift in Fourier Spectrum vs Confusion Matrix Accuracy (4 Features):

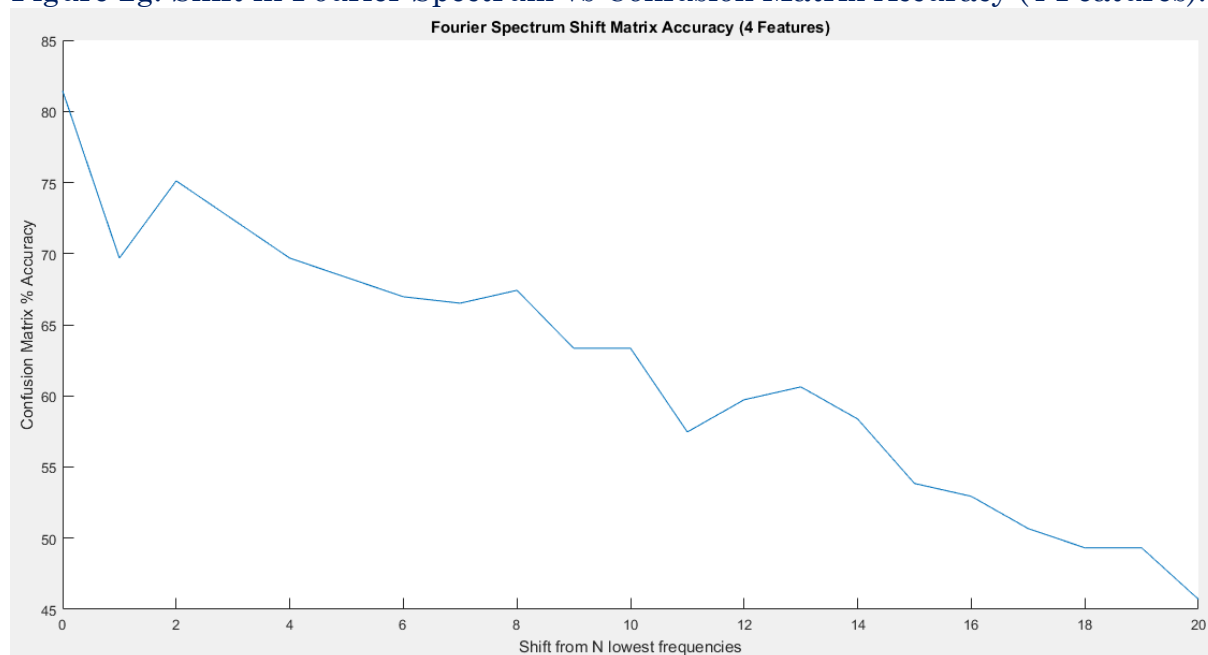


Figure 2h: Chain Code of a Binary Image

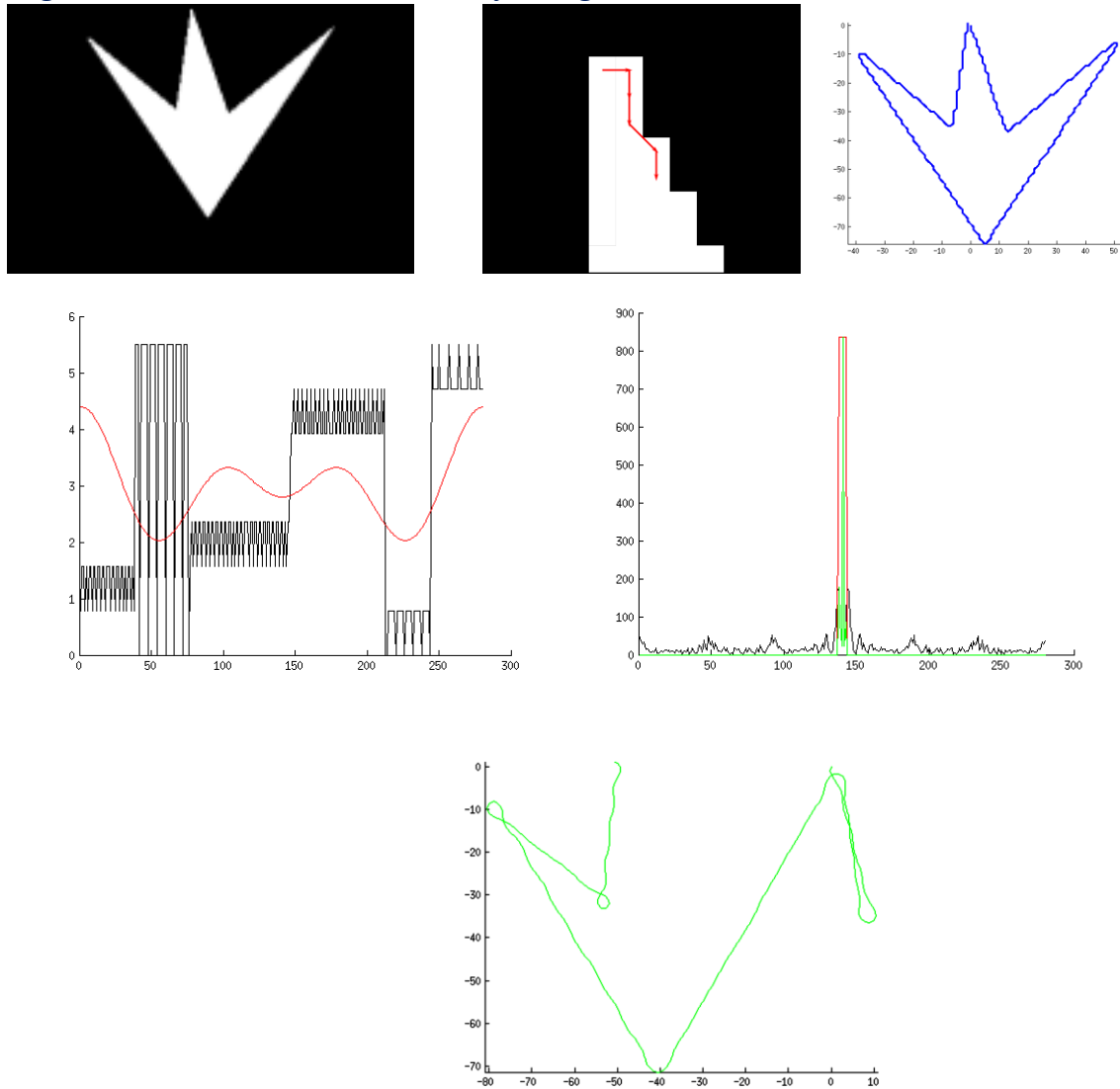


Figure 3a: Supervised Confusion Matrix (Optimised Zernike)

Number of Features: 15

Training Directory: './images/train' (Provided)

Testing Directory: './images/test' (Provided)

Feature Type: Absolute Values of Zernike Basis Functions

Image Size: 60

Degree: 6

		PREDICTED CLASSES						
		Alien	Arrow	Butterfly	Face	Star	Toonhead	TOTAL
ACTUAL CLASSES	Alien	20	0	1	19	2	0	42
	Arrow	0	2	0	0	0	0	2
	Butterfly	1	0	49	0	0	0	50
	Face	0	0	1	97	1	0	99
	Star	0	0	0	3	23	0	26

	<b>Toonhead</b>	0	0	0	1	1	0	2
	<b>TOTAL</b>	21	2	51	120	27	0	221

% Accuracy (Aliens): 90.9091%

% Accuracy (Arrow): 100.0000%

% Accuracy (Butterfly): 98.0000%

% Accuracy (Face): 97.9798%

% Accuracy (Star): 88.4615%

% Accuracy (Toonhead): 0.0000%

% Accuracy: 86.4253%

Figure 3b: Predicted Classifications t-SNE Plot (Zernike)

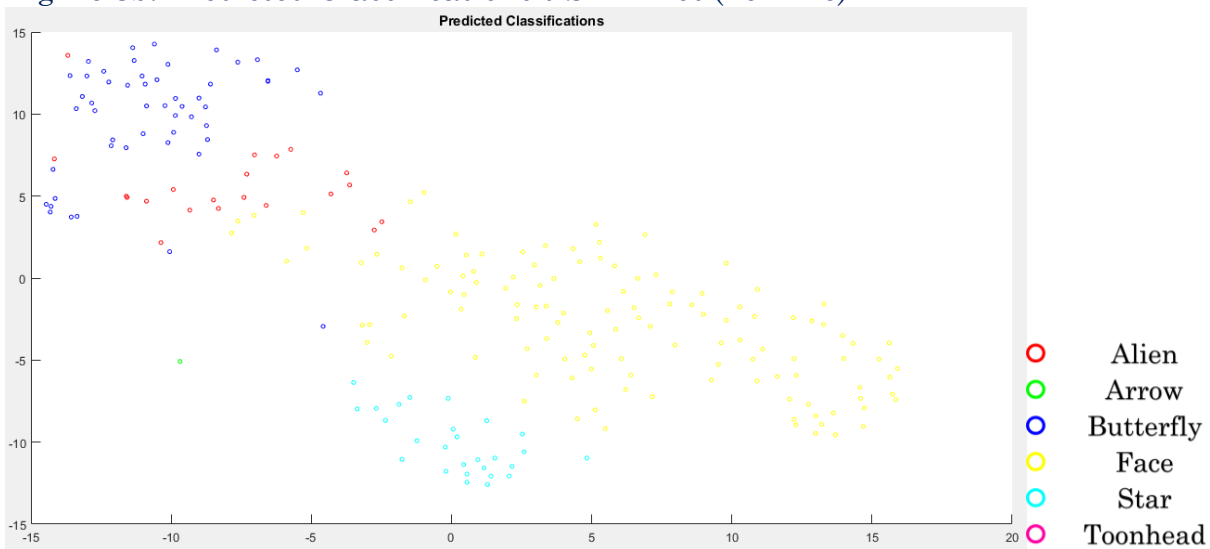


Figure 3c: Actual Classifications t-SNE Plot (Zernike)

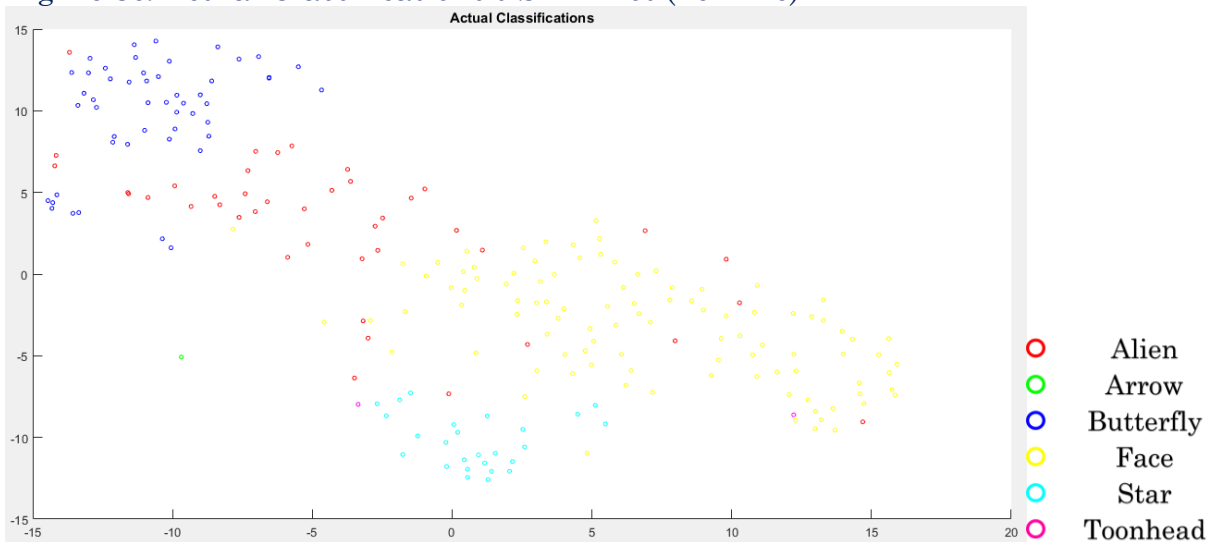


Figure 3d: Zernike Moments Feature Extraction Plot

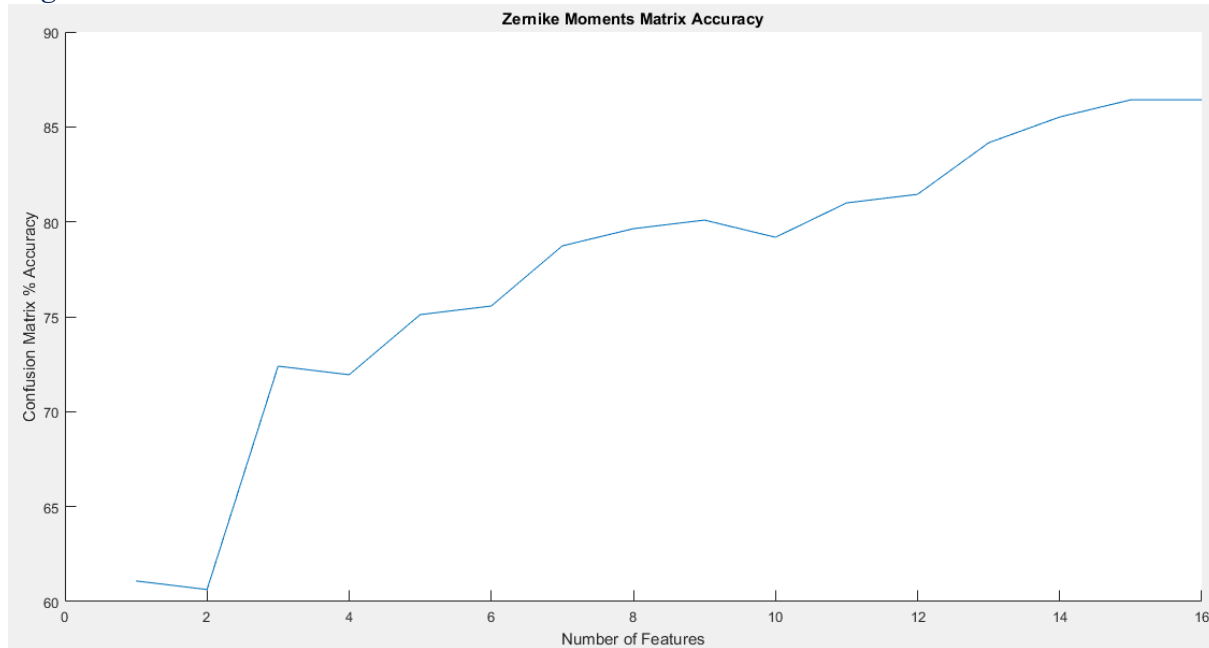


Figure 4a: Supervised Confusion Matrix (Optimised Black/White Image Properties)

**Number of Features:** 8**Training Directory:** './images/train' (Provided)**Testing Directory:** './images/test' (Provided)**Feature Type:** Properties of Binary Image

		PREDICTED CLASSES						TOTAL
		Alien	Arrow	Butterfly	Face	Star	Toonhead	
ACTUAL CLASSES	Alien	39	0	3	0	0	0	42
	Arrow	0	2	0	0	0	0	2
	Butterfly	0	0	50	0	0	0	50
	Face	3	0	2	90	4	0	99
	Star	0	0	0	0	26	0	26
	Toonhead	2	0	0	0	0	0	2
	TOTAL	44	2	55	90	30	0	221

% Accuracy (Aliens): 92.8571%

% Accuracy (Arrow): 100.0000%

% Accuracy (Butterfly): 100.0000%

% Accuracy (Face): 90.9091%

% Accuracy (Star): 100.0000%

% Accuracy (Toonhead): 0.0000%

% Accuracy (Overall): 93.6652%



Figure 4b: Predicted Classifications t-SNE Plot (Optimised Black/White Properties)

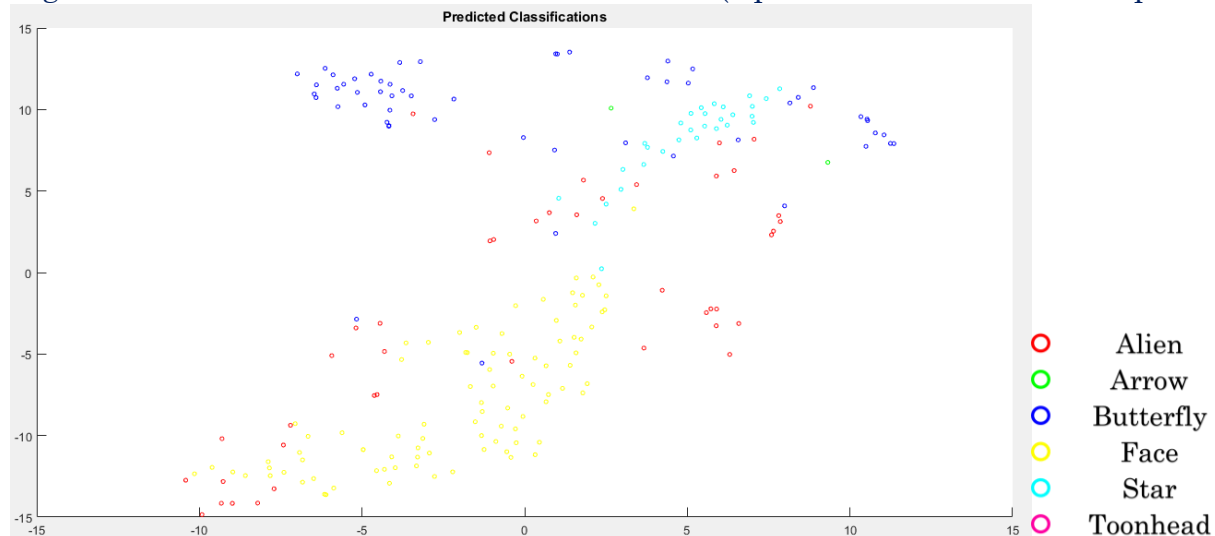


Figure 4c: Actual Classifications t-SNE Plot (Optimised Black/White Properties)

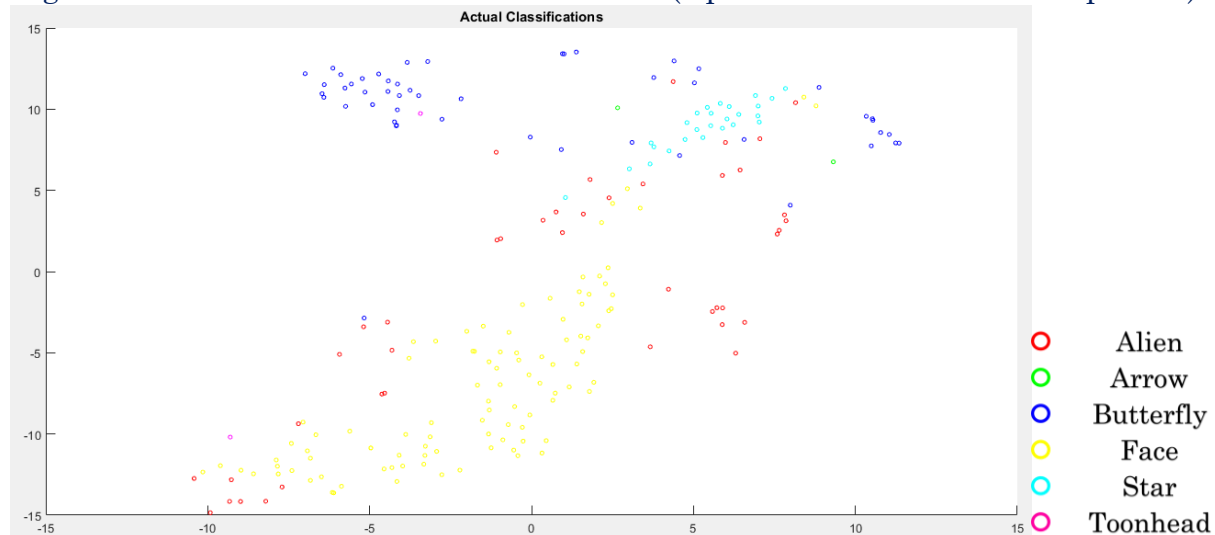


Figure 4d: Black / White Properties Feature Extraction Plot

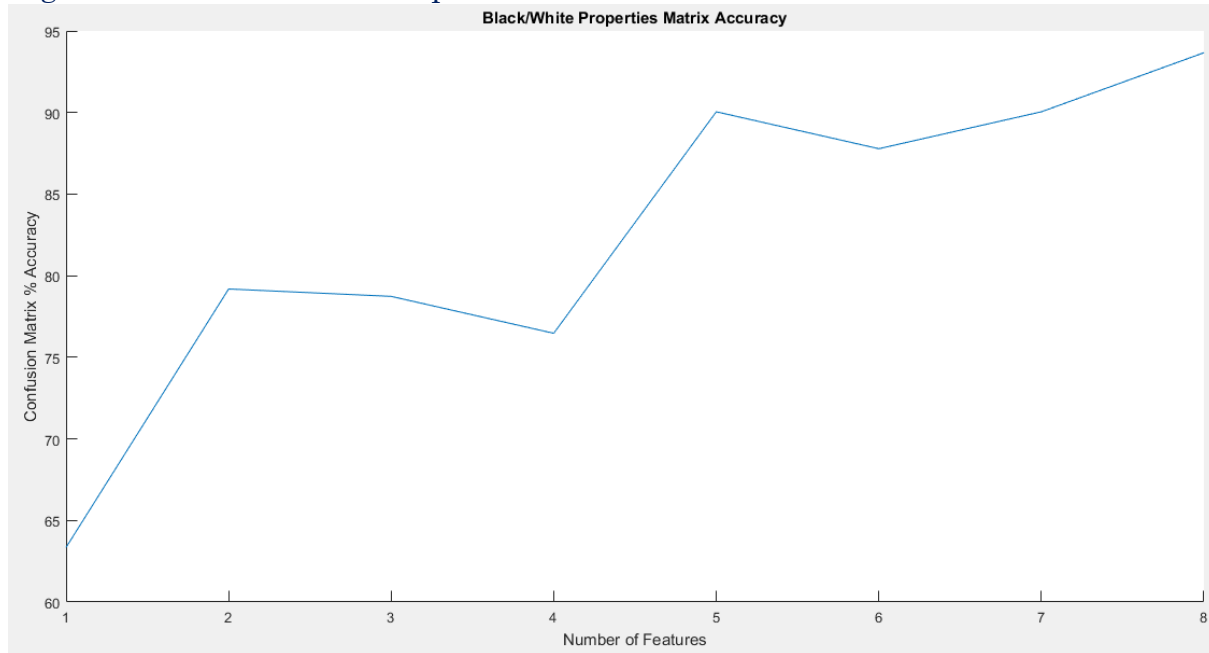


Figure 5a: Supervised Confusion Matrix (Optimised PCA)

**Number of Features:** 15**Training Directory:** './images/train' (Provided)**Testing Directory:** './images/test' (Provided)**Feature Type:** PCA for dimensionality reduction

		PREDICTED CLASSES						TOTAL
		Alien	Arrow	Butterfly	Face	Star	Toonhead	
ACTUAL CLASSES	Alien	41	0	0	0	1	0	42
	Arrow	0	1	1	0	0	0	2
	Butterfly	0	0	50	0	0	0	50
	Face	4	0	1	91	3	0	99
	Star	1	0	0	1	24	0	26
	Toonhead	0	0	1	1	0	0	2
	TOTAL	46	1	53	93	28	0	221

% Accuracy (Aliens): 97.6190%

% Accuracy (Arrow): 50.0000%

% Accuracy (Butterfly): 100.0000%

% Accuracy (Face): 91.9192%

% Accuracy (Star): 92.3077%

% Accuracy (Toonhead): 0.0000%

% Accuracy (Overall): 93.6652%

Figure 5b: Predicted Classifications t-SNE Plot (PCA)

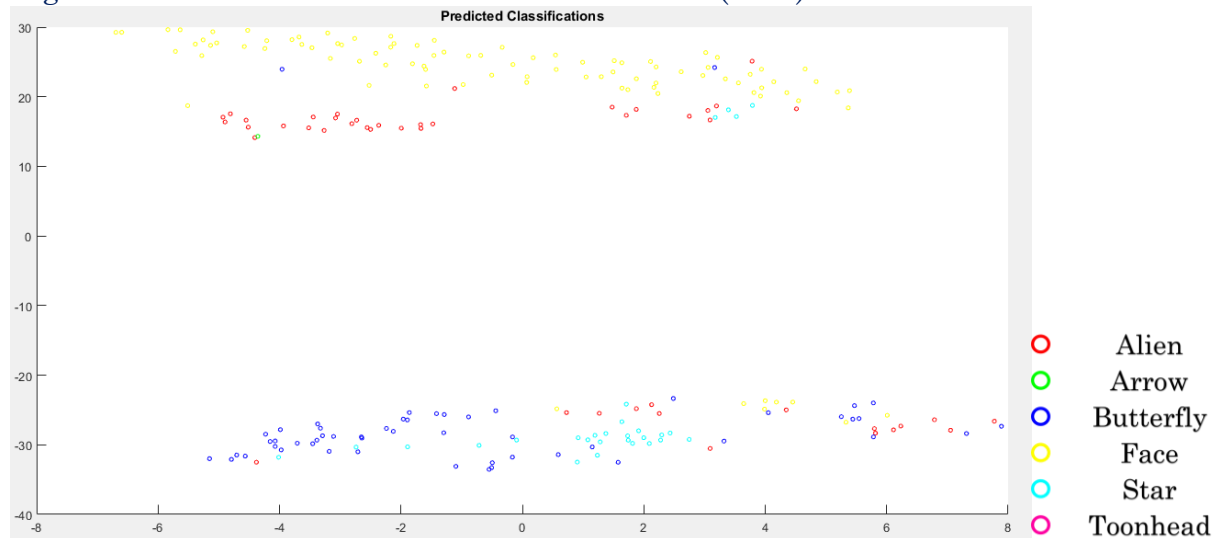


Figure 5c: Actual Classifications t-SNE Plot (PCA)

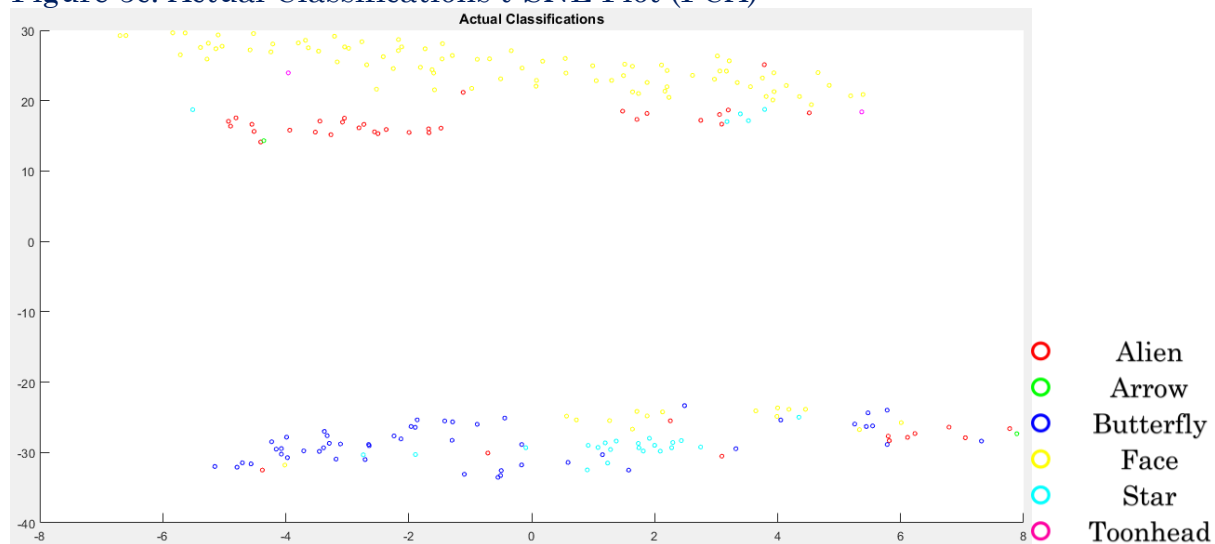


Figure 5d: PCA Feature Extraction Plot

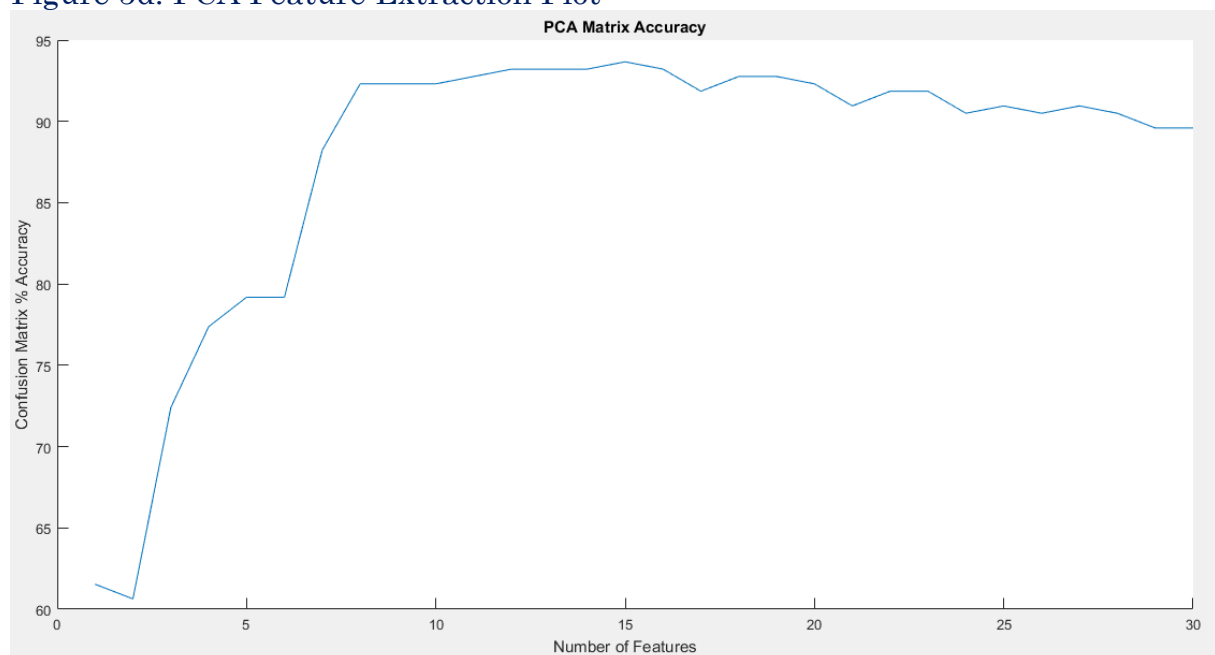


Figure 6a: Unsupervised Confusion Matrix 1 (k-Means – Black/White Properties – 8 Features)

**Number of Clusters:** 6**Training Directory:** './images/train' (Provided)**Testing Directory:** './images/test' (Provided)**Unsupervised Method:** GMM based on k-Means Clustering**Supervised Feature Type:** Black/White Properties – 8 Features

		PREDICTED CLASSES						
		Alien	Arrow	Butterfly	Face	Star	Toonhead	TOTAL
ACTUAL CLASSES	Alien	0	0	2	24	16	0	42
	Arrow	0	0	2	0	0	0	2
	Butterfly	0	0	45	0	5	0	50
	Face	0	0	2	94	3	0	99
	Star	0	0	11	0	15	0	26
	Toonhead	0	0	0	1	1	0	2
	TOTAL	0	0	62	119	40	0	221

% Accuracy (Aliens): 0.0000%

% Accuracy (Arrow): 0.0000%

% Accuracy (Butterfly): 90.0000%

% Accuracy (Face): 94.9495%

% Accuracy (Star): 57.6923%

% Accuracy (Toonhead): 0.0000%

% Accuracy (Overall): 69.6833%

Figure 6b: k-Means Clustering Plot 1 (Compared to Black and White Properties Supervised Model – 8 Features)

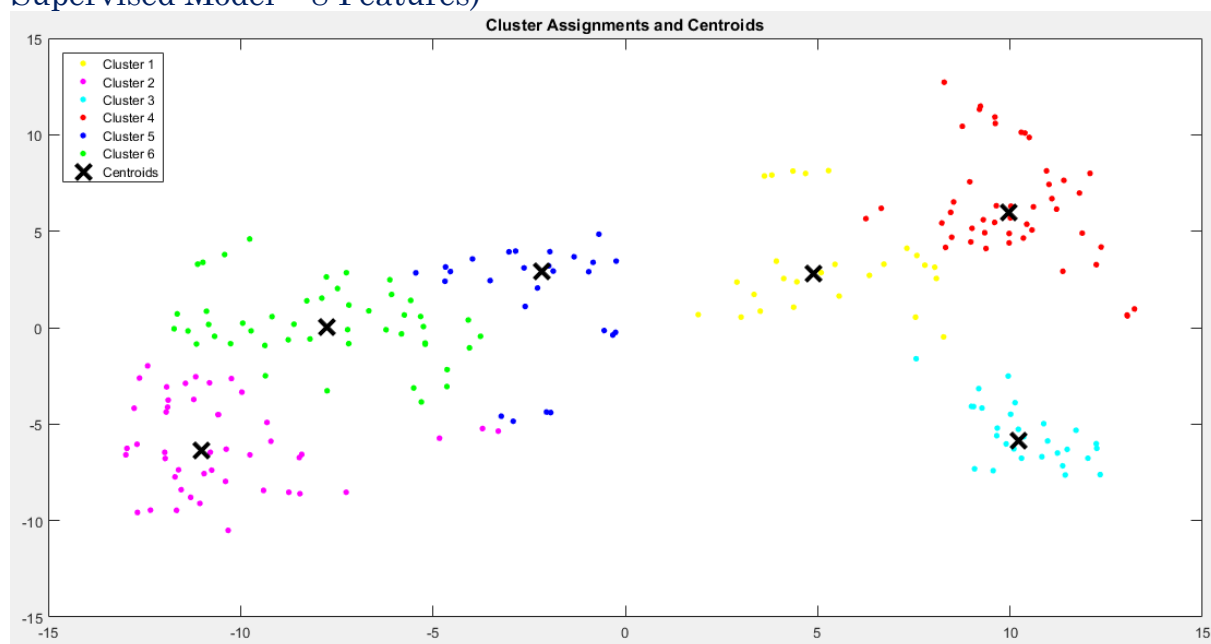


Figure 6c: k-Means Actual Classifications Plot 1 (For Visual Comparison)

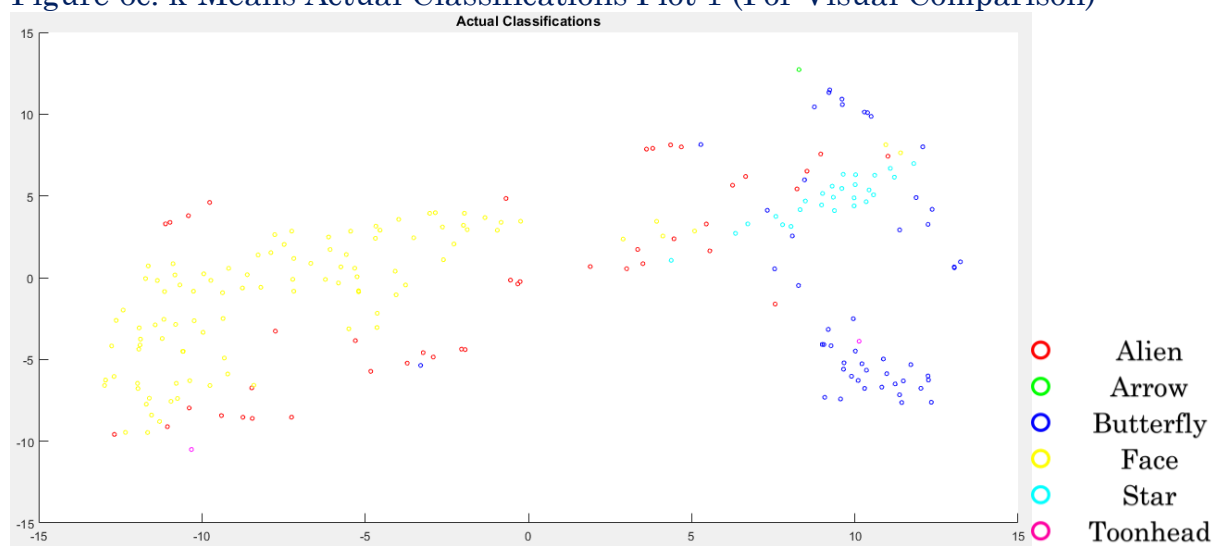


Figure 6d: k-Means Silhouette Value Plot 1

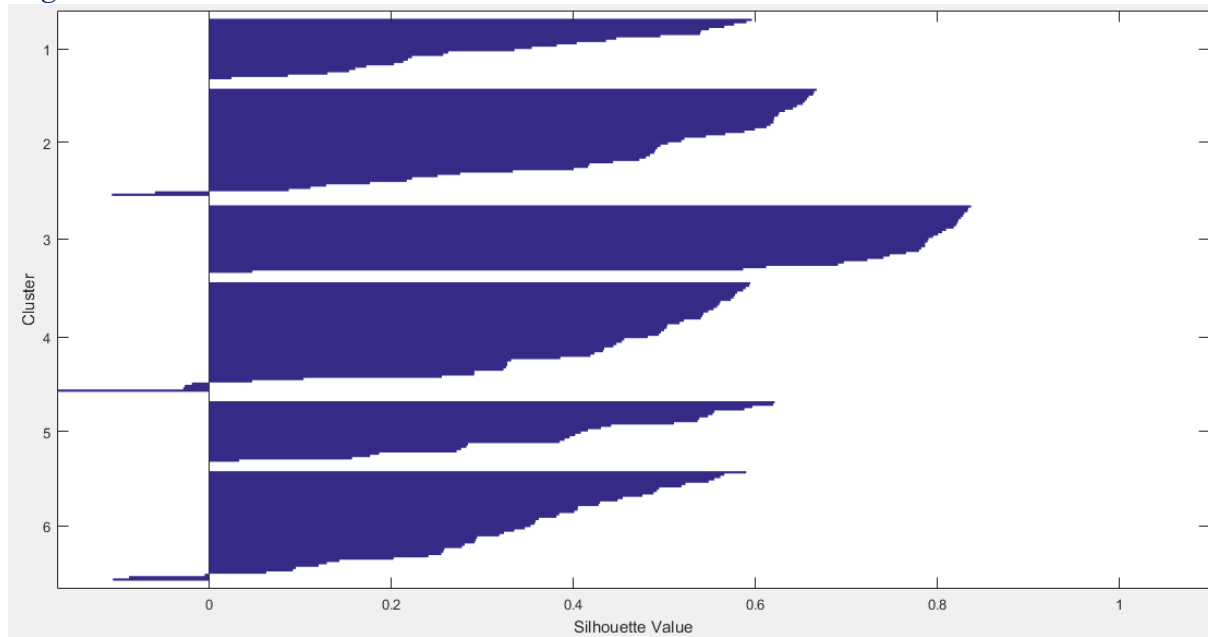


Figure 6e: k-Means Unsupervised GMM Model 1

unsupervised_models						
1x6 struct with 6 fields						
Fields	kMeansIdx	mean	cov	prior	closestLabel	closestClass
1	'1'	[7.4705e+03...	8x8 double	0.1176	5	'Star'
2	'2'	[5.9092e+03...	8x8 double	0.2081	4	'Face'
3	'3'	[9.0281e+03...	8x8 double	0.1312	3	'Butterfly'
4	'4'	[8.5401e+03...	8x8 double	0.2127	3	'Butterfly'
5	'5'	[6.6487e+03...	8x8 double	0.1176	4	'Face'
6	'6'	[6.1802e+03...	8x8 double	0.2127	4	'Face'

**Covariance Matrix – Cluster 1 (closest to Star):**

1.0e+05 \*

7.1038	1.9120	0.0007	-0.0008	1.9101	2.4227	0.0017	1.2681
1.9120	2.3180	-0.0000	0.0000	2.3169	1.6767	0.0003	0.3336
0.0007	-0.0000	0.0000	-0.0000	-0.0000	0.0001	0.0000	0.0001
-0.0008	0.0000	-0.0000	0.0000	0.0000	-0.0001	-0.0000	-0.0002
1.9101	2.3169	-0.0000	0.0000	2.3162	1.6863	0.0003	0.3328
2.4227	1.6767	0.0001	-0.0001	1.6863	2.8222	0.0006	0.3647
0.0017	0.0003	0.0000	-0.0000	0.0003	0.0006	0.0000	0.0005
1.2681	0.3336	0.0001	-0.0002	0.3328	0.3647	0.0005	0.3027

**Covariance Matrix – Cluster 2 (closest to Face):**

1.0e+05 \*

1.9841	-0.2061	0.0002	-0.0006	-0.2044	0.0543	0.0003	0.2466
-0.2061	1.0264	-0.0001	0.0003	1.0216	0.1663	0.0000	-0.0092
0.0002	-0.0001	0.0000	-0.0000	-0.0001	-0.0000	0.0000	0.0000
-0.0006	0.0003	-0.0000	0.0000	0.0003	0.0001	-0.0000	-0.0001
-0.2044	1.0216	-0.0001	0.0003	1.0171	0.1840	0.0000	-0.0095
0.0543	0.1663	-0.0000	0.0001	0.1840	2.5834	0.0000	-0.0915
0.0003	0.0000	0.0000	-0.0000	0.0000	0.0000	0.0000	0.0000



0.2466 -0.0092 0.0000 -0.0001 -0.0095 -0.0915 0.0000 0.0418  
**Covariance Matrix – Cluster 3 (closest to Butterfly):**  
 1.0e+05 \*

4.2355	-0.6002	0.0005	-0.0005	-0.5953	1.9678	-0.0002	0.5066
-0.6002	3.0340	-0.0004	0.0004	3.0319	2.2301	0.0000	-0.1821
0.0005	-0.0004	0.0000	-0.0000	-0.0004	0.0000	-0.0000	0.0001
-0.0005	0.0004	-0.0000	0.0000	0.0004	0.0000	0.0000	-0.0001
-0.5953	3.0319	-0.0004	0.0004	3.0297	2.2399	0.0000	-0.1820
1.9678	2.2301	0.0000	0.0000	2.2399	6.5249	-0.0003	-0.1230
-0.0002	0.0000	-0.0000	0.0000	0.0000	-0.0003	0.0000	0.0001
0.5066	-0.1821	0.0001	-0.0001	-0.1820	-0.1230	0.0001	0.1122

**Covariance Matrix – Cluster 4 (closest to Butterfly):**  
 1.0e+06 \*

2.7781	1.4897	-0.0006	0.0000	1.4949	3.3028	-0.0001	0.1800
1.4897	2.4966	-0.0012	0.0002	2.4996	2.9495	-0.0002	-0.0271
-0.0006	-0.0012	0.0000	-0.0000	-0.0012	-0.0012	0.0000	-0.0000
0.0000	0.0002	-0.0000	0.0000	0.0002	0.0002	-0.0000	-0.0000
1.4949	2.4996	-0.0012	0.0002	2.5026	2.9579	-0.0002	-0.0270
3.3028	2.9495	-0.0012	0.0002	2.9579	5.3569	-0.0004	0.0061
-0.0001	-0.0002	0.0000	-0.0000	-0.0002	-0.0004	0.0000	0.0001
0.1800	-0.0271	-0.0000	-0.0000	-0.0270	0.0061	0.0001	0.0606

**Covariance Matrix – Cluster 5 (closest to Face):**  
 1.0e+05 \*

1.5179	0.3449	0.0001	-0.0003	0.3390	-0.9426	0.0004	0.3396
0.3449	0.6995	-0.0000	0.0000	0.6918	-0.4960	0.0001	0.1001
0.0001	-0.0000	0.0000	-0.0000	-0.0000	-0.0001	0.0000	0.0000
-0.0003	0.0000	-0.0000	0.0000	0.0000	0.0001	-0.0000	-0.0001
0.3390	0.6918	-0.0000	0.0000	0.6843	-0.4708	0.0001	0.0982
-0.9426	-0.4960	-0.0001	0.0001	-0.4708	4.4019	0.0001	-0.3532
0.0004	0.0001	0.0000	-0.0000	0.0001	0.0001	0.0000	0.0001
0.3396	0.1001	0.0000	-0.0001	0.0982	-0.3532	0.0001	0.0935

**Covariance Matrix – Cluster 6 (closest to Face):**  
 1.0e+05 \*

2.5434	0.5733	0.0002	-0.0006	0.5712	0.1173	0.0005	0.3780
0.5733	0.5689	0.0000	-0.0001	0.5651	-0.0418	0.0002	0.1186
0.0002	0.0000	0.0000	-0.0000	0.0000	0.0000	0.0000	0.0000
-0.0006	-0.0001	-0.0000	0.0000	-0.0001	-0.0000	-0.0000	-0.0001
0.5712	0.5651	0.0000	-0.0001	0.5615	-0.0386	0.0002	0.1182
0.1173	-0.0418	0.0000	-0.0000	-0.0386	0.6097	0.0001	-0.0205
0.0005	0.0002	0.0000	-0.0000	0.0002	0.0001	0.0000	0.0001
0.3780	0.1186	0.0000	-0.0001	0.1182	-0.0205	0.0001	0.0683

Figure 6f: Supervised GMM Model 1 (Black and White Properties – 8 Features)

models				
1x6 struct with 4 fields				
Fields	name	mean	cov	prior
1	'Alien'	[5.6985e+03...	8x8 double	0.1892
2	'Arrow'	[9.4723e+03...	8x8 double	0.0135
3	'Butterfly'	[8.3890e+03...	8x8 double	0.2252
4	'Face'	[5.8447e+03...	8x8 double	0.4505
5	'Star'	[8.6007e+03...	8x8 double	0.1081
6	'Toonhead'	[8707;9851;...	8x8 double	0.0135

**Covariance Matrix – Alien:**

1.0e+06 \*

1.6008	-0.1589	0.0002	-0.0005	-0.1561	0.4309	0.0002	0.1960
-0.1589	1.1440	-0.0001	0.0002	1.1437	0.8959	0.0001	0.0049
0.0002	-0.0001	0.0000	-0.0000	-0.0001	-0.0000	0.0000	0.0000
-0.0005	0.0002	-0.0000	0.0000	0.0002	0.0000	-0.0000	-0.0001
-0.1561	1.1437	-0.0001	0.0002	1.1435	0.8974	0.0001	0.0051
0.4309	0.8959	-0.0000	0.0000	0.8974	1.0740	0.0001	0.0510
0.0002	0.0001	0.0000	-0.0000	0.0001	0.0001	0.0000	0.0000
0.1960	0.0049	0.0000	-0.0001	0.0051	0.0510	0.0000	0.0333

**Covariance Matrix – Arrow:**

1.0e+07 \*

3.0293	0.4665	0.0003	-0.0000	0.4698	0.8598	0.0001	0.6383
0.4665	0.0718	0.0001	-0.0000	0.0723	0.1324	0.0000	0.0983
0.0003	0.0001	0.0000	-0.0000	0.0001	0.0001	0.0000	0.0001
-0.0000	-0.0000	-0.0000	0.0000	-0.0000	-0.0000	-0.0000	-0.0000
0.4698	0.0723	0.0001	-0.0000	0.0729	0.1333	0.0000	0.0990
0.8598	0.1324	0.0001	-0.0000	0.1333	0.2440	0.0000	0.1811
0.0001	0.0000	0.0000	-0.0000	0.0000	0.0000	0.0000	0.0000
0.6383	0.0983	0.0001	-0.0000	0.0990	0.1811	0.0000	0.1599

**Covariance Matrix – Butterfly:**

1.0e+06 \*

1.8464	-0.2520	0.0003	-0.0002	-0.2493	1.0063	-0.0002	0.2012
-0.2520	2.0145	-0.0004	0.0003	2.0164	2.2416	-0.0002	-0.2436
0.0003	-0.0004	0.0000	-0.0000	-0.0004	-0.0002	-0.0000	0.0001
-0.0002	0.0003	-0.0000	0.0000	0.0003	0.0002	-0.0000	-0.0000
-0.2493	2.0164	-0.0004	0.0003	2.0183	2.2469	-0.0002	-0.2438
1.0063	2.2416	-0.0002	0.0002	2.2469	3.7986	-0.0005	-0.2267
-0.0002	-0.0002	-0.0000	-0.0000	-0.0002	-0.0005	0.0000	0.0000
0.2012	-0.2436	0.0001	-0.0000	-0.2438	-0.2267	0.0000	0.0644

**Covariance Matrix – Face:**

1.0e+05 \*

1.8004	-1.3515	0.0003	-0.0007	-1.3480	-1.0149	0.0001	0.2779
-1.3515	1.8464	-0.0003	0.0007	1.8442	1.5874	0.0001	-0.1782
0.0003	-0.0003	0.0000	-0.0000	-0.0003	-0.0002	0.0000	0.0000
-0.0007	0.0007	-0.0000	0.0000	0.0007	0.0006	-0.0000	-0.0001
-1.3480	1.8442	-0.0003	0.0007	1.8421	1.5862	0.0001	-0.1776
-1.0149	1.5874	-0.0002	0.0006	1.5862	1.5308	0.0001	-0.1407

0.0001 0.0001 0.0000 -0.0000 0.0001 0.0001 0.0000 0.0000  
 0.2779 -0.1782 0.0000 -0.0001 -0.1776 -0.1407 0.0000 0.0475  
**Covariance Matrix – Star:**  
 1.0e+05 \*

1.2020 -1.2126 0.0003 -0.0003 -1.2124 -0.6512 0.0001 0.2227  
 -1.2126 1.2714 -0.0004 0.0003 1.2716 0.7879 -0.0001 -0.2419  
 0.0003 -0.0004 0.0000 -0.0000 -0.0004 -0.0002 0.0000 0.0001  
 -0.0003 0.0003 -0.0000 0.0000 0.0003 0.0002 -0.0000 -0.0001  
 -1.2124 1.2716 -0.0004 0.0003 1.2717 0.7903 -0.0001 -0.2422  
 -0.6512 0.7879 -0.0002 0.0002 0.7903 1.6864 -0.0001 -0.2971  
 0.0001 -0.0001 0.0000 -0.0000 -0.0001 -0.0001 0.0000 0.0000  
 0.2227 -0.2419 0.0001 -0.0001 -0.2422 -0.2971 0.0000 0.0686

**Covariance Matrix – Toonhead:**  
 1.0e+06 \*

0.1306 -0.1700 0.0000 -0.0000 -0.1708 -0.4206 -0.0000 0.0416  
 -0.1700 1.1280 -0.0001 0.0002 1.1246 0.7244 0.0001 0.0134  
 0.0000 -0.0001 0.0000 -0.0000 -0.0001 -0.0001 -0.0000 0.0000  
 -0.0000 0.0002 -0.0000 0.0000 0.0002 0.0001 0.0000 -0.0000  
 -0.1708 1.1246 -0.0001 0.0002 1.1213 0.7259 0.0001 0.0129  
 -0.4206 0.7244 -0.0001 0.0001 0.7259 1.3892 0.0000 -0.1208  
 -0.0000 0.0001 -0.0000 0.0000 0.0001 0.0000 0.0000 0.0000  
 0.0416 0.0134 0.0000 -0.0000 0.0129 -0.1208 0.0000 0.0183

Figure 7a: Unsupervised Confusion Matrix 2 (k-Means – Fourier– 6 Features)

**Number of Clusters:** 6

**Training Directory:** './images/train' (Provided)

**Testing Directory:** './images/test' (Provided)

**Unsupervised Method:** GMM based on k-Means Clustering

**Supervised Feature Type:** Fourier – 6 Features

		PREDICTED CLASSES						
		Alien	Arrow	Butterfly	Face	Star	Toonhead	TOTAL
ACTUAL CLASSES	Alien	0	0	20	22	0	0	42
	Arrow	0	2	0	0	0	0	2
	Butterfly	0	6	43	1	0	0	50
	Face	0	60	1	37	1	0	99
	Star	0	0	0	2	24	0	26
	Toonhead	0	0	0	2	0	0	2
	TOTAL	0	68	64	64	25	0	221

% Accuracy (Aliens): 0.0000%

% Accuracy (Arrow): 100.0000%

% Accuracy (Butterfly): 86.0000%

% Accuracy (Face): 37.3737%

% Accuracy (Star): 92.3077%

% Accuracy (Toonhead): 0.0000%

% Accuracy (Overall): 47.9638%

Figure 7b: k-Means Clustering Plot 2 (Compared to Fourier Supervised Model – 6 Features)

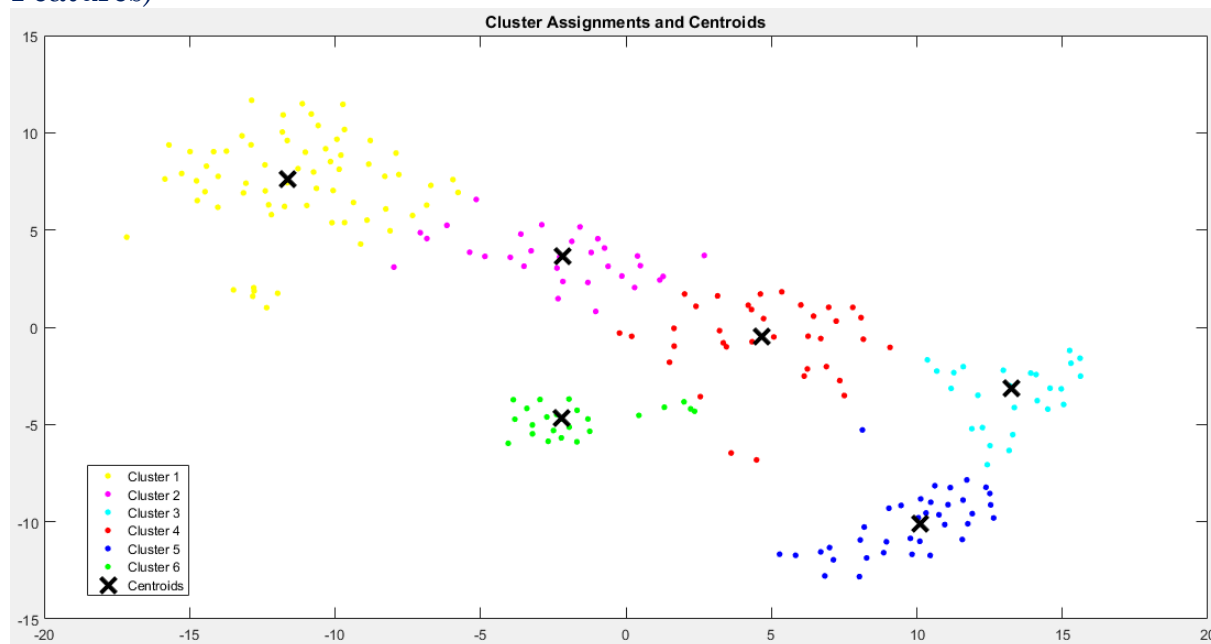


Figure 7c: k-Means Actual Classifications Plot 2 (For Visual Comparison)

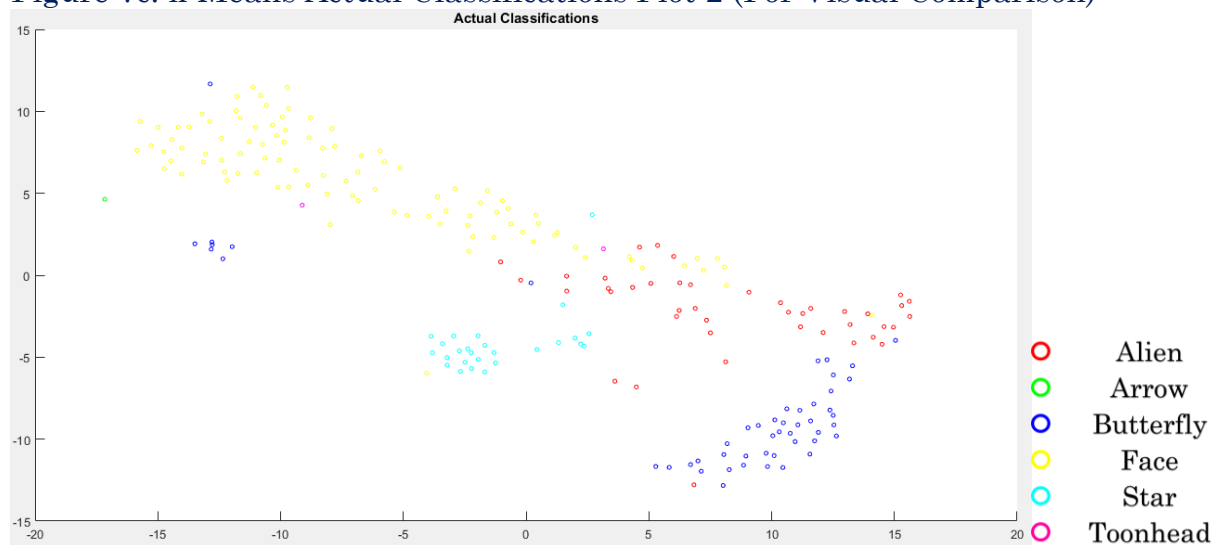


Figure 7d: k-Means Silhouette Value Plot 2

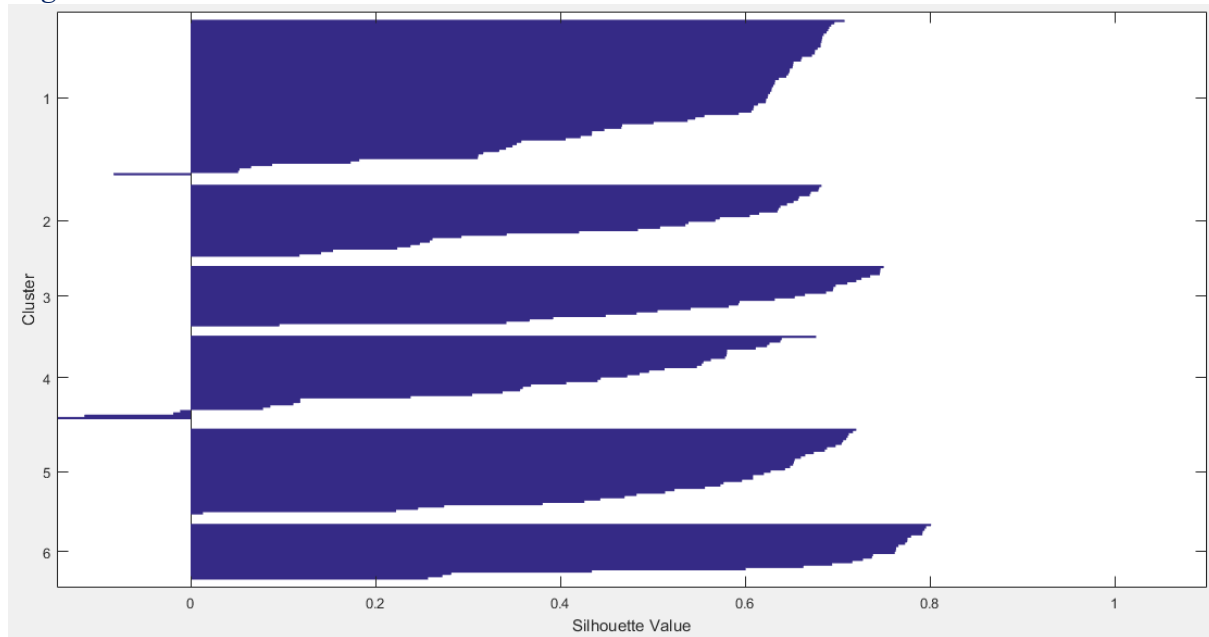


Figure 7e: k-Means Unsupervised GMM Model 2

unsupervised_models						
1x6 struct with 6 fields						
Fields	kMeansIdx	mean	cov	prior	closestLabel	closestClass
1	'1'	[920.9555;3...	6x6 double	0.3032	2 'Arrow'	
2	'2'	[1.0643e+03...	6x6 double	0.1403	4 'Face'	
3	'3'	[1.6499e+03...	6x6 double	0.1176	3 'Butterfly'	
4	'4'	[1.2667e+03...	6x6 double	0.1629	4 'Face'	
5	'5'	[1.2722e+03...	6x6 double	0.1674	3 'Butterfly'	
6	'6'	[1.1282e+03...	6x6 double	0.1086	5 'Star'	

**Covariance Matrix – Cluster 1 (closest to Arrow):**

1.0e+03 \*

9.4682	3.6529	0.9984	0.2382	0.5825	0.0213
3.6529	2.7265	0.8891	-0.4876	-0.3033	-0.3348
0.9984	0.8891	0.7105	-0.2282	-0.0349	-0.1241
0.2382	-0.4876	-0.2282	1.3518	0.3171	0.3974
0.5825	-0.3033	-0.0349	0.3171	0.7018	0.1779
0.0213	-0.3348	-0.1241	0.3974	0.1779	0.4192

**Covariance Matrix – Cluster 2 (closest to Face):**

1.0e+03 \*

1.9354	-0.2508	0.5481	-0.4516	0.4001	0.0111
-0.2508	0.4427	0.0112	0.1293	0.0216	-0.0394
0.5481	0.0112	2.0562	0.0244	0.7272	0.0608
-0.4516	0.1293	0.0244	0.8491	-0.2178	-0.0216
0.4001	0.0216	0.7272	-0.2178	0.8395	0.0234
0.0111	-0.0394	0.0608	-0.0216	0.0234	0.2928

**Covariance Matrix – Cluster 3 (closest to Butterfly):**

1.0e+04 \*

```

3.2448 -0.1580 0.3415 -0.0740 0.5031 -0.0913
-0.1580 0.2183 -0.0061 -0.1686 -0.0640 0.0493
0.3415 -0.0061 0.4719 0.0464 0.0270 -0.1573
-0.0740 -0.1686 0.0464 0.8260 -0.2403 0.0282
0.5031 -0.0640 0.0270 -0.2403 0.6962 -0.0459
-0.0913 0.0493 -0.1573 0.0282 -0.0459 0.3866

```

**Covariance Matrix – Cluster 4 (closest to Face):**

1.0e+03 \*

```

7.4903 -0.1983 1.9835 0.1574 0.2231 -1.9477
-0.1983 1.0691 -1.2919 -0.1856 -0.3124 -0.7173
1.9835 -1.2919 5.1100 0.2515 1.2603 -0.2884
0.1574 -0.1856 0.2515 1.5771 -0.0316 0.1854
0.2231 -0.3124 1.2603 -0.0316 2.1863 -0.2800
-1.9477 -0.7173 -0.2884 0.1854 -0.2800 2.8075

```

**Covariance Matrix – Cluster 5 (closest to Butterfly):**

1.0e+03 \*

```

9.3852 0.7437 -0.0988 1.9711 -0.0420 1.8711
0.7437 0.5877 -0.2899 -0.2235 0.1643 0.1361
-0.0988 -0.2899 0.7838 0.2521 -0.0484 -0.1775
1.9711 -0.2235 0.2521 1.4785 -0.0820 0.2496
-0.0420 0.1643 -0.0484 -0.0820 0.4775 -0.3543
1.8711 0.1361 -0.1775 0.2496 -0.3543 2.4267

```

**Covariance Matrix – Cluster 6 (closest to Star):**

1.0e+03 \*

```

4.3227 -0.1293 0.3168 -0.1184 0.8774 -0.0837
-0.1293 0.1808 -0.1521 0.0926 0.0333 -0.0154
0.3168 -0.1521 0.2384 -0.1106 -0.0202 -0.0205
-0.1184 0.0926 -0.1106 0.6231 0.1802 0.1451
0.8774 0.0333 -0.0202 0.1802 0.7513 0.2454
-0.0837 -0.0154 -0.0205 0.1451 0.2454 0.4222

```

Figure 7f: Supervised GMM Model 2 (Fourier – 6 Features)

models				
1x6 struct with 4 fields				
Fields	name	mean	cov	prior
1	'Alien'	[1.4249e+03...	6x6 double	0.1892
2	'Arrow'	[645.0737;1...	6x6 double	0.0135
3	'Butterfly'	[1.2262e+03...	6x6 double	0.2252
4	'Face'	[968.9064;3...	6x6 double	0.4505
5	'Star'	[1.1329e+03...	6x6 double	0.1081
6	'Toonhead'	[1.1367e+03...	6x6 double	0.0135

**Covariance Matrix – Alien:**

1.0e+04 \*

```

5.0275 -0.2520 0.3342 0.6810 0.9004 1.1306
-0.2520 0.1546 -0.0618 -0.0902 -0.0688 -0.0464
0.3342 -0.0618 0.3742 -0.0114 0.0132 0.1030
0.6810 -0.0902 -0.0114 0.3422 0.0617 0.2612
0.9004 -0.0688 0.0132 0.0617 0.6392 0.2092

```



1.1306 -0.0464 0.1030 0.2612 0.2092 0.5604  
**Covariance Matrix – Arrow:**  
 1.0e+04 \*

5.9310 -0.3220 0.4778 1.2991 1.5605 0.4943  
 -0.3220 0.7748 0.1861 -0.1309 -0.6554 0.0834  
 0.4778 0.1861 0.0979 0.0878 -0.0341 0.0707  
 1.2991 -0.1309 0.0878 0.2893 0.3873 0.0995  
 1.5605 -0.6554 -0.0341 0.3873 0.8407 0.0470  
 0.4943 0.0834 0.0707 0.0995 0.0470 0.0573

**Covariance Matrix – Butterfly:**  
 1.0e+04 \*

3.5376 0.0418 0.0527 0.8100 -0.0637 0.5525  
 0.0418 0.0883 -0.0054 -0.0341 0.0282 -0.0239  
 0.0527 -0.0054 0.0781 0.0280 -0.0058 -0.0534  
 0.8100 -0.0341 0.0280 0.3410 -0.0084 0.0577  
 -0.0637 0.0282 -0.0058 -0.0084 0.1046 -0.0826  
 0.5525 -0.0239 -0.0534 0.0577 -0.0826 0.4133

**Covariance Matrix – Face:**  
 1.0e+03 \*

2.5576 0.1270 0.3923 0.2354 0.4265 0.2587  
 0.1270 0.1171 0.0382 0.0266 0.0479 0.0253  
 0.3923 0.0382 0.2738 -0.0260 0.0467 0.0165  
 0.2354 0.0266 -0.0260 0.3580 0.0280 0.1346  
 0.4265 0.0479 0.0467 0.0280 0.3145 0.0690  
 0.2587 0.0253 0.0165 0.1346 0.0690 0.2278

**Covariance Matrix – Star:**  
 1.0e+03 \*

7.4648 0.7909 1.1941 0.8763 1.3853 0.6805  
 0.7909 0.2320 0.1123 -0.0494 0.2731 0.1040  
 1.1941 0.1123 0.4954 0.2529 0.2962 0.2010  
 0.8763 -0.0494 0.2529 0.4206 0.1966 0.1360  
 1.3853 0.2731 0.2962 0.1966 0.9765 0.3041  
 0.6805 0.1040 0.2010 0.1360 0.3041 0.4810

**Covariance Matrix – Toonhead:**  
 1.0e+04 \*

1.4644 0.3133 -0.0932 -0.1268 0.2052 0.0896  
 0.3133 0.0682 -0.0139 -0.0146 0.0521 0.0301  
 -0.0932 -0.0139 0.0358 0.0699 0.0277 0.0484  
 -0.1268 -0.0146 0.0699 0.1390 0.0666 0.1043  
 0.2052 0.0521 0.0277 0.0666 0.0843 0.0863  
 0.0896 0.0301 0.0484 0.1043 0.0863 0.1035

Figure 8a: Unsupervised Confusion Matrix 3 (k-Means – Zernike– 15 Features)

**Number of Clusters:** 6

**Training Directory:** './images/train' (Provided)

**Testing Directory:** './images/test' (Provided)

**Unsupervised Method:** GMM based on k-Means Clustering

**Supervised Feature Type:** Zernike – 15 Features

	PREDICTED CLASSES
--	-------------------

		Alien	Arrow	Butterfly	Face	Star	Toonhead	TOTAL
ACTUAL CLASSES	Alien	17	0	4	19	2	0	42
	Arrow	0	0	0	2	0	0	2
	Butterfly	9	0	41	0	0	0	50
	Face	1	0	0	96	2	0	99
	Star	0	0	0	3	23	0	26
	Toonhead	0	0	0	1	1	0	2
	TOTAL	27	0	45	121	28	0	221

% Accuracy (Aliens): 40.4762%

% Accuracy (Arrow): 0.0000%

% Accuracy (Butterfly): 82.0000%

% Accuracy (Face): 96.9697%

% Accuracy (Star): 88.4615%

% Accuracy (Toonhead): 0.0000%

% Accuracy (Overall): 80.0905%

Figure 8b: k-Means Clustering Plot 3 (Compared to Zernike Supervised Model – 15 Features)

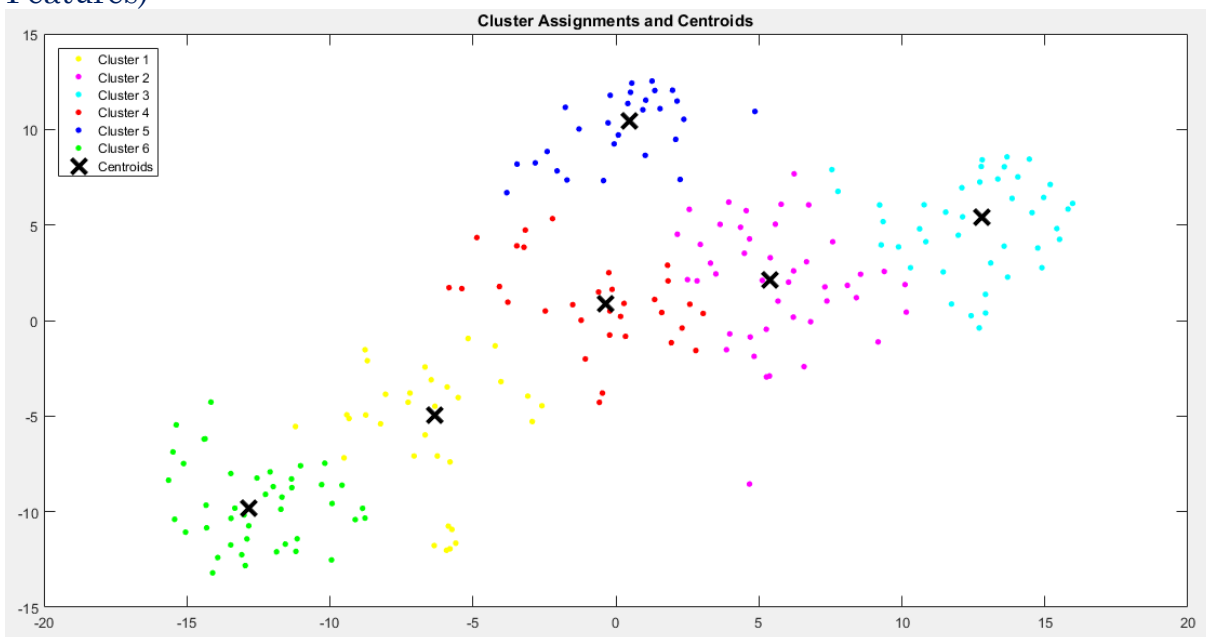


Figure 8c: k-Means Actual Classifications Plot 3 (For Visual Comparison)

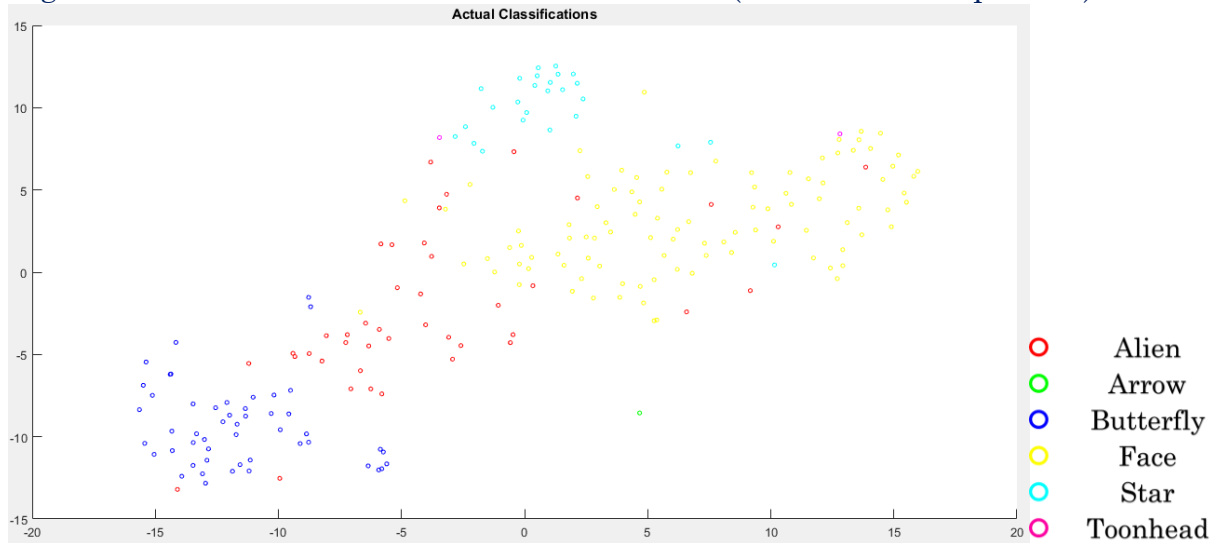


Figure 8d: k-Means Silhouette Value Plot 3

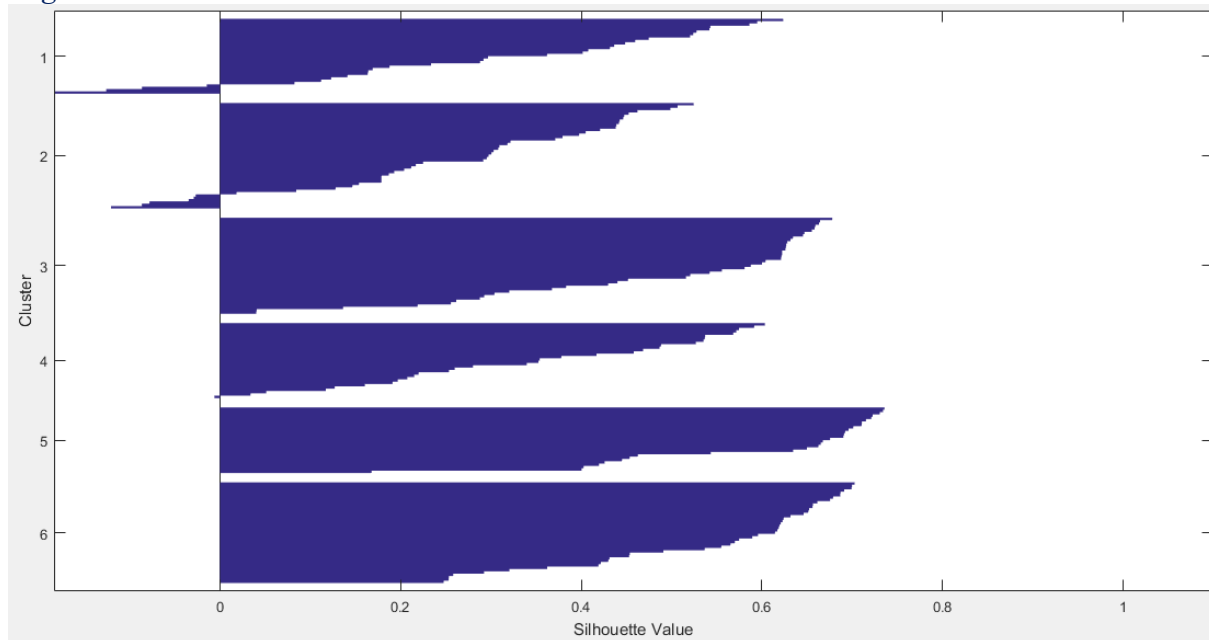


Figure 8e: k-Means Unsupervised GMM Model 3

unsupervised_models						
1x6 struct with 6 fields						
Fields	kMeansIdx	mean	cov	prior	closestLabel	closestClass
1	'1'	15x1 double	15x15 double	0.1448	1 'Alien'	
2	'2'	15x1 double	15x15 double	0.2036	4 'Face'	
3	'3'	15x1 double	15x15 double	0.1855	4 'Face'	
4	'4'	15x1 double	15x15 double	0.1448	4 'Face'	
5	'5'	15x1 double	15x15 double	0.1267	5 'Star'	
6	'6'	15x1 double	15x15 double	0.1946	3 'Butterfly'	

**Covariance Matrix – Cluster 1 (closest to Alien):**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.0013	6.1599e-05	4.5459e-04	4.4600e-05	1.2734e-04	-2.9205e-05	-1.4758e-04	2.2227e-06	1.0443e-05	2.0479e-05	-8.6918e-05	-1.2596e-05	-4.5997e-04	-3.2350e-04	-4.7597e-06
2	6.1599e-05	0.0010	8.0644e-05	9.8187e-06	2.7552e-05	-4.2416e-06	-3.8159e-05	3.3287e-06	1.4975e-06	5.6407e-06	-1.3411e-05	-1.5570e-06	-8.7084e-05	-5.9478e-05	-2.9502e-06
3	4.5459e-04	8.0644e-05	0.0016	5.8533e-05	1.6820e-04	-4.3285e-05	-1.8083e-04	-3.5155e-06	1.5520e-05	2.4312e-05	-1.2704e-04	-1.9286e-05	-6.3579e-04	-4.5117e-04	-1.9368e-06
4	4.4600e-05	9.8187e-06	5.8533e-05	0.0010	1.9838e-05	-3.1090e-06	-2.7310e-05	2.3213e-06	1.0984e-06	4.0293e-06	-9.7987e-06	-1.1520e-06	-6.3031e-05	-4.3103e-05	-2.0734e-06
5	1.2734e-04	2.7552e-05	1.6820e-04	1.9838e-05	0.0011	-9.1629e-06	-7.5560e-05	5.9558e-06	3.2429e-06	1.1090e-05	-2.8647e-05	-3.4754e-06	-1.7979e-04	-1.2335e-04	-5.4451e-06
6	-2.9205e-05	-4.2416e-06	-4.3285e-05	-3.1090e-06	-9.1629e-06	0.0010	6.9037e-06	1.5393e-06	-1.2061e-06	-7.5192e-07	9.4745e-06	1.6037e-06	4.0525e-05	2.9556e-05	-8.0338e-07
7	-1.4758e-04	-3.8159e-05	-1.8083e-04	-2.7310e-05	-7.5560e-05	6.9037e-06	0.0011	-1.5610e-05	-2.3726e-06	-1.8222e-05	2.4496e-05	1.6103e-06	2.1050e-04	1.3917e-04	1.2461e-05
8	2.2227e-06	3.3287e-06	-3.5155e-06	2.3213e-06	5.9558e-06	1.5393e-06	-1.5610e-05	0.0010	-5.7713e-07	2.6492e-06	3.4806e-06	1.0449e-06	-4.1129e-06	-4.1990e-07	-2.9080e-06
9	1.0443e-05	1.4975e-06	1.5520e-05	1.0984e-06	3.2429e-06	-1.2061e-06	-2.3726e-06	-5.7713e-07	0.0010	2.5238e-07	-3.4145e-06	-5.8085e-07	-1.4484e-05	-1.0580e-05	3.0614e-07
10	2.0479e-05	5.6407e-06	2.4312e-05	4.0293e-06	1.1090e-05	-7.5192e-07	-1.8222e-05	2.6492e-06	2.5238e-07	0.0010	-2.9166e-06	-8.9361e-08	-2.9329e-05	-1.9103e-05	-2.0704e-06
11	-8.6918e-05	-1.3411e-05	-1.2704e-04	-9.7987e-06	-2.8647e-05	9.4745e-06	2.4496e-05	3.4806e-06	-3.4145e-06	-2.9166e-06	0.0010	4.4673e-06	1.2088e-04	8.7485e-05	-1.6135e-06
12	-1.2596e-05	-1.5570e-06	-1.9286e-05	-1.1520e-06	-3.4754e-06	1.6037e-06	1.6103e-06	1.0449e-06	-5.8085e-07	-8.9361e-08	4.4673e-06	0.0010	1.7385e-05	1.2914e-05	-6.1564e-07
13	-4.5997e-04	-8.7084e-05	-6.3579e-04	-6.3031e-05	-1.7979e-04	4.0525e-05	2.1050e-04	-4.1129e-06	-1.4484e-05	-2.9329e-05	1.2088e-04	1.7385e-05	0.0016	4.5317e-04	7.3776e-06
14	-3.2350e-04	-5.9478e-05	-4.5117e-04	-4.3103e-05	-1.2335e-04	2.9556e-05	1.3917e-04	-4.1990e-07	-1.0580e-05	-1.9103e-05	8.7485e-05	1.2914e-05	4.5317e-04	0.0013	3.4420e-06
15	-4.7597e-06	-2.9502e-06	-1.9368e-06	-2.0734e-06	-5.4451e-06	-8.0338e-07	1.2461e-05	-2.9080e-06	3.0614e-07	-2.0704e-06	-1.6135e-06	-6.1564e-07	7.3776e-06	3.4420e-06	0.0010

**Covariance Matrix – Cluster 2 (closest to Face):**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.0013	-8.7709e-06	2.8316e-04	7.4721e-05	-7.4254e-05	1.8574e-05	-3.8200e-04	1.7889e-04	1.1106e-05	-1.6325e-04	3.8704e-05	1.1050e-05	-3.6672e-04	9.1545e-05	2.3267e-05
2	-8.7709e-06	0.0010	-9.6322e-06	-2.5417e-06	2.5259e-06	-6.3181e-07	1.2994e-05	-6.0851e-06	-3.7780e-07	5.5530e-06	-1.3166e-06	-3.7589e-07	1.2475e-05	-3.1140e-06	-7.9147e-07
3	2.8316e-04	-9.6322e-06	0.0013	8.2058e-05	-8.1546e-05	2.0397e-05	-4.1951e-04	1.9645e-04	1.2197e-05	-1.7928e-04	4.2504e-05	1.2135e-05	-4.0274e-04	1.0053e-04	2.5552e-05
4	7.4721e-05	-2.5417e-06	8.2058e-05	0.0010	-2.1518e-05	5.3825e-06	-1.1070e-04	5.1840e-05	3.2185e-06	-4.7307e-05	1.1216e-05	3.2022e-06	-1.0627e-04	2.6529e-05	6.7427e-06
5	-7.4254e-05	2.5259e-06	-8.1546e-05	-2.1518e-05	0.0010	-5.3489e-06	1.1001e-04	-5.1517e-05	-3.1985e-06	4.7012e-05	-1.1146e-05	-3.1823e-06	1.0561e-04	-2.6363e-05	-6.7006e-06
6	1.8574e-05	-6.3181e-07	2.0397e-05	5.3825e-06	-5.3489e-06	0.0010	-2.7517e-05	1.2886e-05	8.0004e-07	-1.1759e-05	2.7880e-06	7.9599e-07	-2.6417e-05	6.5944e-06	1.6761e-06
7	-3.8200e-04	1.2994e-05	-4.1951e-04	-1.1070e-04	1.1001e-04	-2.7517e-05	0.0016	-2.6502e-04	-1.6454e-05	2.4185e-04	-5.7340e-05	-1.6371e-05	5.4331e-04	-1.3563e-04	-3.4471e-05
8	1.7889e-04	-6.0851e-06	1.9645e-04	5.1840e-05	-5.1517e-05	1.2886e-05	-2.6502e-04	0.0011	7.7055e-06	-1.1326e-04	2.6852e-05	7.6665e-06	-2.5443e-04	6.3513e-05	1.6143e-05
9	1.1106e-05	-3.7780e-07	1.2197e-05	3.2185e-06	-3.1985e-06	8.0004e-07	-1.6454e-05	7.7055e-06	0.0010	-7.0317e-06	1.6671e-06	4.7598e-07	-1.5796e-05	3.9433e-06	1.0022e-06
10	-1.6325e-04	5.5530e-06	-1.7928e-04	-4.7307e-05	4.7012e-05	-1.1759e-05	2.4185e-04	-1.1326e-04	-7.0317e-06	0.0011	-2.4504e-05	-6.9961e-06	2.3218e-04	-5.7959e-05	-1.4731e-05
11	3.8704e-05	-1.3166e-06	4.2504e-05	1.1216e-05	-1.1146e-05	2.7880e-06	-5.7340e-05	2.6852e-05	1.6671e-06	-2.4504e-05	0.0010	1.6587e-06	-5.5047e-05	1.3741e-05	3.4925e-06
12	1.1050e-05	-3.7589e-07	1.2135e-05	3.2022e-06	-3.1823e-06	7.9599e-07	-1.6371e-05	7.6665e-06	4.7598e-07	-6.9961e-06	1.6587e-06	0.0010	-1.5716e-05	3.9233e-06	9.9715e-07
13	-3.6672e-04	1.2475e-05	-4.0274e-04	-1.0627e-04	1.0561e-04	-2.6417e-05	5.4331e-04	-2.5443e-04	-1.5796e-05	2.3218e-04	-5.5047e-05	-1.5716e-05	0.0015	-1.3020e-04	-3.3093e-05
14	9.1545e-05	-3.1140e-06	1.0053e-04	2.6529e-05	-2.6363e-05	6.5944e-06	-1.3563e-04	6.3513e-05	3.9433e-06	-5.7959e-05	1.3741e-05	3.9233e-06	-1.3020e-04	0.0010	8.2609e-06
15	2.3267e-05	-7.9147e-07	2.5552e-05	6.7427e-06	-6.7006e-06	1.6761e-06	-3.4471e-05	1.6143e-05	1.0022e-06	-1.4731e-05	3.4925e-06	9.9715e-07	-3.3093e-05	8.2609e-06	0.0010

**Covariance Matrix – Cluster 3 (closest to Face):**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.0010	-1.4206e-06	7.9256e-06	-7.3360e-09	-2.0495e-06	-2.8135e-06	-2.3547e-06	-4.9472e-08	-1.5085e-06	-1.5380e-06	-6.7142e-06	-4.8199e-07	1.1351e-05	-1.4615e-06	-3.5822e-06
2	-1.4206e-06	0.0010	-2.2516e-06	2.0841e-09	5.8225e-07	7.9928e-07	6.6895e-07	1.4055e-08	4.2854e-07	4.3694e-07	1.9074e-06	1.3693e-07	-3.2247e-06	4.1521e-07	1.0177e-06
3	7.9256e-06	-2.2516e-06	0.0010	-1.1627e-08	-3.2485e-06	-4.4593e-06	-3.7322e-06	-7.8412e-08	-2.3909e-06	-2.4377e-06	-1.0642e-05	-7.6395e-07	1.7991e-05	-2.3165e-06	-5.6777e-06
4	-7.3360e-09	2.0841e-09	-1.1627e-08	0.0010	3.0068e-09	4.1276e-09	3.4545e-09	7.2579e-11	2.2130e-09	2.2564e-09	9.8503e-09	7.0712e-10	-1.6653e-08	2.1442e-09	5.2554e-09
5	-2.0495e-06	5.8225e-07	-3.2485e-06	3.0068e-09	0.0010	1.1532e-06	9.6513e-07	2.0277e-08	6.1827e-07	6.3039e-07	2.7520e-06	1.9756e-07	-4.6524e-06	5.9904e-07	1.4683e-06
6	-2.8135e-06	7.9928e-07	-4.4593e-06	4.1276e-09	1.1532e-06	0.0010	1.3249e-06	2.7835e-08	8.4873e-07	8.6536e-07	3.7777e-06	2.7119e-07	-6.3865e-06	8.2233e-07	2.0155e-06
7	-2.3547e-06	6.6895e-07	-3.7322e-06	3.4545e-09	9.6513e-07	1.3249e-06	0.0010	2.3296e-08	7.1033e-07	7.2425e-07	3.1617e-06	2.2697e-07	-5.3451e-06	6.8824e-07	1.6869e-06
8	-4.9472e-08	1.4055e-08	-7.8412e-08	7.2579e-11	2.0277e-08	2.7835e-08	2.3296e-08	0.0010	1.4924e-08	1.5216e-08	6.6427e-08	4.7686e-09	-1.1230e-07	1.4460e-08	3.5441e-08
9	-1.5085e-06	4.2854e-07	-2.3909e-06	2.2130e-09	6.1827e-07	8.4873e-07	7.1033e-07	1.4924e-08	0.0010	4.6397e-07	2.0254e-06	1.4540e-07	-3.4241e-06	4.4089e-07	1.8086e-06
10	-1.5380e-06	4.3694e-07	-2.4377e-06	2.2564e-09	6.3039e-07	8.6536e-07	7.2425e-07	1.5216e-08	4.6397e-07	0.0010	2.0651e-06	1.4825e-07	-3.4913e-06	4.4953e-07	1.1018e-06
11	-6.7142e-06	1.9074e-06	-1.0642e-05	9.8503e-09	2.7520e-06	3.7777e-06	3.1617e-06	6.6427e-08	2.0254e-06	2.0651e-06	0.0010	6.4718e-07	-1.5241e-05	1.9624e-06	4.8099e-06
12	-4.8199e-07	1.3693e-07	-7.6395e-07	7.0712e-10	1.9756e-07	2.7119e-07	2.2697e-07	4.7686e-09	1.4540e-07	1.4825e-07	6.4718e-07	0.0010	-1.0941e-06	1.4088e-07	3.4529e-07
13	1.1351e-05	-3.2247e-06	1.7991e-05	-1.6653e-08	-4.6524e-06	-6.3865e-06	-5.3451e-06	-1.1230e-07	-3.4241e-06	-3.4913e-06	-1.5241e-05	-1.0941e-06	0.0010	-3.3176e-06	-8.1315e-06
14	-1.4615e-06	4.1521e-07	-2.3165e-06	2.1442e-09	5.9904e-07	8.2233e-07	6.8824e-07	1.4460e-08	4.4089e-07	4.4953e-07	1.9624e-06	1.4088e-07	-3.3176e-06	0.0010	1.0470e-06
15	-3.5822e-06	1.0177e-06	-5.6777e-06	5.2554e-09	1.4683e-06	2.0155e-06	1.6869e-06	3.5441e-08	1.0806e-06	1.1018e-06	4.8099e-06	3.4529e-07	-8.1315e-06	1.0470e-06	0.0010

**Covariance Matrix – Cluster 4 (closest to Face):**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.0010	-4.7551e-06	9.3219e-06	3.6403e-06	-1.2654e-05	-3.6406e-06	-3.4012e-06	1.1381e-05	-3.5144e-06	-1.4276e-05	-7.7838e-06	-7.5651e-07	2.0054e-05	1.1175e-05	-7.9883e-06
2	-4.7551e-06	0.0010	-1.0530e-05	-4.1122e-06	1.4295e-05	4.1126e-06	3.8421e-06	-1.2857e-05	3.9699e-06	1.6126e-05	8.7928e-06	8.5458e-07	-2.2654e-05	-1.2623e-05	9.0238e-06
3	9.3219e-06	-1.0530e-05	0.0010	8.0615e-06	-2.8023e-05	-8.0622e-06	-7.5319e-06	2.5204e-05	-7.7826e-06	-3.1614e-05	-1.7237e-05	-1.6753e-06	4.4410e-05	2.4746e-05	-1.7690e-05
4	3.6403e-06	-4.1122e-06	8.0615e-06	0.0010	-1.0943e-05	-3.1484e-06	-2.9413e-06	9.8424e-06	-3.0392e-06	-1.2346e-05	-6.7313e-06	-6.5422e-07	1.7343e-05	9.6637e-06	-6.9082e-06
5	-1.2654e-05	1.4295e-05	-2.8023e-05	-1.0943e-05	0.0010	1.0944e-05	1.0225e-05	-3.4214e-05	1.0565e-05	4.2916e-05	2.3400e-05	2.2742e-06	-6.0287e-05	-3.3593e-05	2.4014e-05
6	-3.6406e-06	4.1126e-06	-8.0622e-06	-3.1484e-06	1.0944e-05	0.0010	2.9416e-06	-9.8434e-06	3.0395e-06	1.2347e-05	6.7320e-06	6.5429e-07	-1.7344e-05	-9.6646e-06	6.9089e-06
7	-3.4012e-06	3.8421e-06	-7.5319e-06	-2.9413e-06	1.0225e-05	2.9416e-06	0.0010	-9.1959e-06	2.8396e-06	1.1535e-05	6.2892e-06	6.1125e-07	-1.6204e-05	-9.0289e-06	6.4544e-06
8	1.1381e-05	-1.2857e-05	2.5204e-05	9.8424e-06	-3.4214e-05	-9.8434e-06	-9.1959e-06	0.0010	-9.5020e-06	-3.8598e-05	-2.1045e-05	-2.0454e-06	5.4222e-05	3.0213e-05	-2.1598e-05
9	-3.5144e-06	3.9699e-06	-7.7826e-06	-3.0392e-06	1.0565e-05	3.0395e-06	2.8396e-06	-9.5020e-06	0.0010	1.1919e-05	6.4985e-06	6.3159e-07	-1.6743e-05	-9.3294e-06	6.6692e-06
10	-1.4276e-05	1.6126e-05	-3.1614e-05	-1.2346e-05	4.2916e-05	1.2347e-05	1.1535e-05	-3.8598e-05	1.1919e-05	0.0010	2.6398e-05	2.5656e-06	-6.8012e-05	-7.3897e-05	2.7091e-05
11	-7.7838e-06	8.7928e-06	-1.7237e-05	-6.7313e-06	2.3400e-05	6.7320e-06	6.2892e-06	-2.1045e-05	6.4985e-06	2.6398e-05	0.0010	1.3989e-06	-3.7083e-05	-2.0663e-05	1.4771e-05
12	-7.5651e-07	8.5458e-07	-1.6753e-06	-6.5422e-07	2.2742e-06	6.5429e-07	6.1125e-07	-2.0454e-06	6.3159e-07	2.5656e-06	1.3989e-06	0.0010	-3.6041e-06	-2.0083e-06	1.4356e-06
13	2.0054e-05	-2.2654e-05	4.4410e-05	1.7343e-05	-6.0287e-05	-1.7344e-05	-1.6204e-05	5.4222e-05	-1.6743e-05	-6.8012e-05	-3.7083e-05	-3.6041e-06	0.0011	5.3237e-05	-3.8057e-05
14	1.1175e-05	-1.2623e-05	2.4746e-05	9.6637e-06	-3.3593e-05	-9.6646e-06	-9.0289e-06	3.0213e-05	-3.8597e-05	-2.0663e-05	-2.0083e-06	5.3237e-05	0.0010	-2.1206e-05	-2.1206e-05
15	-7.9883e-06	9.0238e-06	-1.7690e-05	-6.9082e-06	2.4014e-05	6.9089e-06	6.4544e-06	-2.1598e-05	6.6692e-06	2.7091e-05	1.4771e-05	1.4356e-06	-3.8057e-05	-2.1206e-05	0.0010



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.0010	-6.0033e-06	5.1460e-05	-1.0230e-05	-1.8147e-05	-5.5483e-06	-4.9693e-05	-2.6531e-05	-4.9792e-06	-2.0468e-05	-1.4916e-05	-2.2163e-06	1.2342e-04	-3.5381e-05	-1.6845e-05
2	-6.0033e-06	0.0010	-7.5437e-06	1.4996e-06	2.6603e-06	8.1333e-07	7.2846e-06	3.8893e-06	7.2992e-07	3.0005e-06	2.1866e-06	3.2490e-07	-1.8093e-05	5.1866e-06	2.4694e-06
3	5.1460e-05	-7.5437e-06	0.0011	-1.2855e-05	-2.2804e-05	-6.9719e-06	-6.2443e-05	-3.3339e-05	-6.2569e-06	-2.5720e-05	-1.8744e-05	-2.7850e-06	1.5509e-04	-4.4460e-05	-2.1168e-05
4	-1.0230e-05	1.4996e-06	-1.2855e-05	0.0010	4.5332e-06	1.3859e-06	1.2413e-05	6.6274e-06	1.2438e-06	5.1130e-06	3.7261e-06	5.5363e-07	-3.0831e-05	8.8382e-06	4.2079e-06
5	-1.8147e-05	2.6603e-06	-2.2804e-05	4.5332e-06	0.0010	2.4586e-06	2.2021e-05	1.1757e-05	2.2065e-06	9.0703e-06	6.6100e-06	9.8214e-07	-5.4694e-05	1.5679e-05	7.4648e-06
6	-5.5483e-06	8.1333e-07	-6.9719e-06	1.3859e-06	2.4586e-06	0.0010	6.7324e-06	3.5945e-06	6.7459e-07	2.7731e-06	2.0209e-06	3.0027e-07	-1.6722e-05	4.7935e-06	2.2822e-06
7	-4.9693e-05	7.2846e-06	-6.2443e-05	1.2413e-05	2.2021e-05	6.7324e-06	0.0011	3.2194e-05	6.0420e-06	2.4837e-05	1.8100e-05	2.6894e-06	-1.4977e-04	4.2933e-05	2.0441e-05
8	-2.6531e-05	3.8893e-06	-3.3339e-05	6.6274e-06	1.1757e-05	3.5945e-06	3.2194e-05	0.0010	3.2258e-06	1.3261e-05	9.6636e-06	1.4359e-06	-7.9961e-05	2.2922e-05	1.0913e-05
9	-4.9792e-06	7.2992e-07	-6.2569e-06	1.2438e-06	2.2065e-06	6.7459e-07	6.0420e-06	3.2258e-06	0.0010	2.4887e-06	1.8136e-06	2.6948e-07	-1.5007e-05	4.3019e-06	2.0482e-06
10	-2.0468e-05	3.0005e-06	-2.5720e-05	5.1130e-06	9.0703e-06	2.7731e-06	2.4837e-05	1.3261e-05	2.4887e-06	0.0010	7.4553e-06	1.1077e-06	-6.1689e-05	1.7684e-05	8.4194e-06
11	-1.4916e-05	2.1866e-06	-1.8744e-05	3.7261e-06	6.6100e-06	2.0209e-06	1.8100e-05	9.6636e-06	1.8136e-06	7.4553e-06	0.0010	8.0727e-07	-4.4956e-05	1.2887e-05	6.1357e-06
12	-2.2163e-06	3.2490e-07	-2.7850e-06	5.5363e-07	9.8214e-07	3.0027e-07	2.6894e-06	1.4359e-06	2.6948e-07	1.1077e-06	8.0727e-07	0.0010	-6.6797e-06	1.9148e-06	9.1166e-07
13	1.2342e-04	-1.8093e-05	1.5509e-04	-3.0831e-05	-5.4694e-05	-1.6722e-05	-1.4977e-04	-7.9961e-05	-1.5007e-05	-6.1689e-05	-4.4956e-05	-6.6797e-06	0.0014	-1.0663e-04	-5.0769e-05
14	-3.5381e-05	5.1866e-06	-4.4460e-05	8.8382e-06	1.5679e-05	4.2933e-05	2.2922e-05	4.3019e-06	1.7684e-05	1.2887e-05	1.9148e-06	-1.0663e-04	0.0010	1.4554e-05	
15	-1.6845e-05	2.4694e-06	-2.1168e-05	4.2079e-06	7.4648e-06	2.2822e-06	2.0441e-05	1.0913e-05	2.0482e-06	8.4194e-06	6.1357e-06	9.1166e-07	-5.0769e-05	1.4554e-05	0.0010

**Covariance Matrix – Cluster 6 (closest to Butterfly):**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.0010	2.0449e-06	1.7970e-06	-2.9878e-06	1.0801e-06	1.8047e-06	4.3226e-06	-7.5239e-06	2.3465e-06	-3.2644e-06	5.0474e-06	2.1999e-08	-5.8863e-06	-5.5587e-06	6.2509e-06
2	2.0449e-06	0.0010	-1.0601e-05	-2.5725e-05	4.3632e-05	8.7630e-06	-1.1884e-06	-7.2608e-05	2.5760e-05	-8.1642e-06	2.0859e-05	-6.7774e-07	-4.5890e-06	-7.4183e-05	6.7247e-05
3	1.7970e-06	-1.0601e-05	0.0010	4.6612e-06	-2.3538e-05	1.4972e-06	1.7703e-05	1.6719e-05	-7.1975e-06	-7.6013e-06	6.5106e-06	5.1763e-07	-2.0156e-05	2.5427e-05	-1.8298e-05
4	-2.9878e-06	-2.5725e-05	4.6612e-06	0.0010	-3.4754e-05	-9.8968e-06	-7.0481e-06	6.8354e-05	-2.3658e-05	1.2129e-05	-2.4939e-05	4.6861e-07	1.4260e-05	6.5928e-05	-6.1990e-05
5	1.0801e-06	4.3632e-05	-2.3538e-05	-3.4754e-05	0.0011	9.4439e-06	-1.5180e-05	-1.0086e-04	3.6765e-05	-3.9808e-06	2.0193e-05	-1.2221e-06	1.0092e-05	-1.0952e-04	9.5595e-05
6	1.8047e-06	8.7630e-06	1.4972e-06	-9.8968e-06	9.4439e-06	0.0010	7.7558e-06	-2.6259e-05	8.7219e-06	-7.4056e-06	1.2858e-05	-7.5297e-08	-1.1622e-05	-2.2910e-05	2.3000e-05
7	4.3226e-06	-1.1884e-06	1.7703e-05	-7.0481e-06	-1.5180e-05	7.7558e-06	0.0010	-1.3707e-05	2.6659e-06	-1.7999e-05	2.3577e-05	4.9967e-07	-3.7687e-05	4.7938e-07	7.8130e-06
8	-7.5239e-06	-7.2608e-05	1.6719e-05	6.8354e-05	-1.0086e-04	-2.6259e-05	-1.3707e-05	0.0012	-6.6006e-05	3.0450e-05	-6.5349e-05	1.4200e-06	3.2434e-05	1.8555e-04	-1.7279e-04
9	2.3465e-06	2.5760e-05	-7.1975e-06	-2.3658e-05	3.6765e-05	8.7219e-06	2.6659e-06	-6.6006e-05	0.0010	-9.4599e-06	2.1395e-05	-5.3840e-07	-8.7313e-06	-6.5645e-05	6.0529e-05
10	-3.2644e-06	-8.1642e-06	-7.6013e-06	1.2129e-05	-3.9808e-06	-7.4056e-06	-1.7999e-05	3.0450e-05	-9.4599e-06	0.0010	-2.0755e-05	-9.9502e-08	2.4437e-05	2.2255e-05	-2.5217e-05
11	5.0474e-06	2.0859e-05	6.5106e-06	-2.4939e-05	2.0193e-05	1.2858e-05	2.3577e-05	-6.5349e-05	2.1395e-05	-2.0755e-05	0.0010	-9.8690e-08	-3.4127e-05	-5.4968e-05	5.6547e-05
12	2.1999e-08	-6.7774e-07	5.1763e-07	4.6861e-07	-1.2221e-06	-7.5297e-08	4.9967e-07	1.4200e-06	-5.3840e-07	-9.9502e-08	-9.8690e-08	0.0010	-4.9010e-07	1.6789e-06	-1.3921e-06
13	-5.8863e-06	-4.5890e-06	-2.0156e-05	1.4260e-05	1.0092e-05	-1.1622e-05	-3.7687e-05	3.2434e-05	-8.7313e-06	2.4437e-05	-3.4127e-05	-4.9010e-07	0.0010	1.4840e-05	-2.3872e-05
14	-5.5587e-06	-7.4183e-05	2.5427e-05	6.5928e-05	-1.0952e-04	-2.2910e-05	4.7938e-07	1.8555e-04	-6.5645e-05	2.2255e-05	-5.4968e-05	1.6789e-06	1.4840e-05	0.0012	-1.7144e-04
15	6.2509e-06	6.7247e-05	-1.8298e-05	-6.1990e-05	9.5595e-05	2.3000e-05	7.8130e-06	-1.7279e-04	6.0529e-05	-2.5217e-05	5.6547e-05	-1.3921e-06	-2.3872e-05	-1.7144e-04	0.0012

Figure 8f: Supervised GMM Model 3 (Zernike – 15 Features)

models				
1x6 struct with 4 fields				
Fields	name	mean	cov	prior
1	'Alien'	15x1 double	15x15 double	0.1892
2	'Arrow'	15x1 double	15x15 double	0.0135
3	'Butterfly'	15x1 double	15x15 double	0.2252
4	'Face'	15x1 double	15x15 double	0.4505
5	'Star'	15x1 double	15x15 double	0.1081
6	'Toonhead'	15x1 double	15x15 double	0.0135

**Covariance Matrix – Alien:**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.0013	9.5259e-05	3.8036e-04	-1.8972e-05	1.1679e-04	2.9366e-05	-2.8674e-04	-1.5657e-04	-1.7921e-05	5.7131e-05	5.1901e-05	4.2949e-05	-1.1761e-04	-3.6843e-04	-1.6224e-04
2	9.5259e-05	0.0011	1.9243e-05	4.7560e-06	1.2517e-04	2.2955e-05	-2.4132e-04	-3.0401e-05	8.1728e-06	8.8628e-05	4.1625e-05	3.2731e-05	-1.2093e-04	-9.9727e-05	-4.4366e-05
3	3.8036e-04	1.9243e-05	0.0016	-3.6016e-05	-8.5185e-06	1.0667e-05	-7.7721e-05	-1.9402e-04	-4.0035e-05	-4.5194e-05	1.7510e-05	1.6716e-05	-1.2371e-06	-4.1538e-04	-1.8313e-04
4	-1.8972e-05	4.7560e-06	-3.6016e-05	0.0010	1.0066e-05	1.0401e-06	-1.2091e-05	1.2696e-05	2.9389e-06	1.0028e-05	2.4637e-06	1.1019e-06	-1.3082e-05	2.2699e-05	8.2996e-06
5	1.1679e-04	1.2517e-04	-8.5185e-06	1.0066e-05	0.0012	3.3728e-05	-3.5583e-04	-3.0796e-05	1.4603e-05	1.3620e-04	6.1697e-05	4.7736e-05	-1.8493e-04	-1.1958e-04	-5.4850e-05
6	2.9366e-05	2.2955e-05	1.0667e-05	1.0401e-06	3.3728e-05	0.0010	-6.5763e-05	-9.7443e-06	1.7393e-06	2.3547e-05	1.1512e-05	8.9565e-06	-3.3462e-05	-3.0584e-05	-1.3920e-05
7	-2.8674e-04	-2.4132e-04	-7.7721e-05	-1.2091e-05	-3.5583e-04	-6.5763e-05	0.0017	9.4367e-05	-2.1572e-05	-2.5035e-04	-1.1921e-04	-9.3810e-05	3.4449e-04	3.0076e-04	1.3387e-04
8	-1.5657e-04	-3.0401e-05	-1.9402e-04	1.2696e-05	-3.0796e-05	-9.7443e-06	9.4367e-05	0.0011	1.0219e-05	-7.6898e-06	-1.5635e-05	-1.5276e-05	2.0541e-05	1.7497e-04	7.2031e-05
9	-1.7921e-05	8.1728e-06	-4.0035e-05	2.9389e-06	1.4603e-05	1.7393e-06	-2.1572e-05	1.0219e-05	0.0010	1.3160e-05	3.0298e-06	2.5279e-06	-1.1074e-05	1.9093e-05	9.3448e-06
10	5.7131e-05	8.8628e-05	-4.5194e-05	1.0028e-05	1.3620e-04	2.3547e-05	-2.5035e-04	-7.6898e-06	1.3160e-05	0.0011	4.3416e-05	3.3089e-05	-1.3486e-04	-5.5923e-05	-2.6934e-05
11	5.1901e-05	4.1625e-05	1.7510e-05	2.4637e-06	6.1697e-05	1.1512e-05	-1.1921e-04	-1.5635e-05	3.0298e-06	4.3416e-05	0.0010	1.6120e-05	-6.4265e-05	-5.2683e-05	-2.5322e-05
12	4.2949e-05	3.2731e-05	1.6716e-05	1.1019e-06	4.7736e-05	8.9565e-06	-9.3810e-05	-1.5276e-05	2.5279e-06	3.3089e-05	1.6120e-05	0.0010	-4.5484e-05	-4.5586e-05	-1.9917e-05
13	-1.1761e-04	-1.2093e-04	-1.2371e-06	-1.3082e-05	-1.8493e-04	-3.3462e-05	3.4449e-04	2.0541e-05	-1.1074e-05	-1.3486e-04	-6.4265e-05	-4.5484e-05	0.0012	1.0935e-04	6.1723e-05
14	-3.6843e-04	-9.9727e-05	-4.1538e-04	2.2699e-05	-1.1958e-04	-3.0584e-05	3.0076e-04	1.7497e-04	1.9093e-05	-5.5923e-05	-5.2683e-05	-4.5586e-05	1.0935e-04	0.0014	1.7281e-04
15	-1.6224e-04	-4.4366e-05	-1.8313e-04	8.2996e-06	-5.4850e-05	-1.3920e-05	1.3387e-04	7.2031e-05	9.3448e-06	-2.6934e-05	-2.5322e-05	-1.9917e-05	6.1723e-05	1.7281e-04	0.0011

**Covariance Matrix – Arrow:**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.0010	3.5140e-19	1.3198e-20	1.9521e-19	-3.0567e-19	1.3267e-19	-3.0638e-19	1.4113e-19	5.8431e-20	-5.3216e-20	3.6799e-19	2.5898e-20	1.2734e-19	3.2876e-20	8.4850e-20
2	3.4004e-19	1.0000e-03	-2.4844e-19	-2.4408e-19	1.7889e-19	3.6516e-19	-6.4393e-20	-1.8903e-19	-2.7527e-19	-2.5636e-19	3.4809e-19	-1.0598e-19	1.3067e-19	2.9935e-19	1.4471e-19
3	7.1758e-20	-2.7513e-19	1.0000e-03	2.6778e-21	2.0892e-19	2.4289e-20	3.1016e-19	-8.7105e-20	-5.1235e-19	-6.6925e-20	3.0162e-19	-6.0559e-20	-1.1083e-19	-9.7902e-20	1.1035e-20
4	2.1837e-19	-2.3011e-19	-4.3230e-20	0.0010	1.2276e-19	-5.2232e-20	-1.5771e-20	-5.0340e-19	-2.3870e-19	5.4489e-21	-4.4339e-20	3.9789e-21	4.0402e-20	-1.3225e-20	-1.8478e-20
5	-2.8621e-19	1.8066e-19	1.6533e-19	1.0238e-19	1.0000e-03	-2.2981e-19	2.6478e-19	4.8052e-22	4.5956e-20	-2.1417e-19	2.2782e-19	-3.5515e-19	-8.7811e-20	2.1764e-19	9.3834e-20
6	1.6572e-19	3.9956e-19	3.6015e-20	2.6495e-21	-2.3129e-19	1.0000e-03	4.8183e-20	2.9992e-19	2.7943e-19	-7.5147e-20	-1.0785e-19	3.3376e-19	-9.5951e-20	5.7642e-20	7.7873e-23
7	-3.0766e-19	-1.0469e-19	3.1700e-19	-1.3006e-21	2.4397e-19	3.6036e-20	1.0000e-03	-1.7105e-19	9.4227e-20	2.2447e-19	-6.3746e-20	-1.9005e-20	1.3202e-19	1.8047e-19	1.8939e-19
8	1.2924e-19	-1.6725e-19	-4.8260e-20	-4.4966e-19	7.0935e-22	2.9073e-19	-1.7316e-19	1.0000e-03	-6.7211e-20	-5.8909e-21	2.0531e-19	1.7159e-19	2.5983e-21	-2.4745e-20	1.7597e-20
9	5.6391e-20	-2.7907e-19	-5.2198e-19	-2.4628e-19	5.5057e-20	3.6749e-19	8.3131e-20	-7.0520e-20	1.0000e-03	-6.5541e-20	1.8813e-19	2.5503e-19	-2.2576e-19	6.2572e-21	-2.4294e-21
10	-4.9043e-20	-2.4589e-19	-7.4479e-20	-6.9442e-21	-2.3739e-19	-7.8307e-20	1.8499e-19	-2.8693e-20	-6.5065e-20	1.0000e-03	-2.5793e-19	2.3208e-20	-8.5109e-20	-1.5419e-19	1.6190e-19
11	3.5195e-19	3.5694e-19	3.0130e-19	5.7936e-22	2.2186e-19	-1.2097e-19	-4.5652e-20	2.6204e-19	2.3654e-19	-2.5210e-19	1.0000e-03	-1.1963e-20	4.8848e-20	3.3999e-19	8.3958e-20
12	3.0817e-20	-1.1316e-19	-6.4807e-20	1.8766e-20	-3.5458e-19	3.2455e-19	-1.9693e-20	1.7002e-19	2.8559e-19	2.1173e-20	-9.8633e-21	0.0010	-1.0435e-20	2.3485e-19	3.0516e-20
13	1.4114e-19	1.1211e-19	-1.3242e-19	3.4038e-20	-8.9943e-20	-9.2530e-20	1.7649e-19	-6.4578e-21	-2.2807e-19	-7.8555e-20	4.8933e-20	-8.1377e-21	1.0000e-03	-3.3110e-19	2.5329e-20
14	1.9076e-20	2.8904e-19	-9.3387e-20	-8.5612e-21	1.9745e-19	5.5674e-20	1.4375e-19	-4.5046e-20	6.6637e-21	-1.7560e-19	3.3000e-19	2.3454e-19	-3.4521e-20	1.0000e-03	2.7182e-20
15	9.4237e-20	1.2390e-19	5.4207e-21	-1.2575e-20	1.1828e-19	-5.5433e-22	1.8718e-19	2.5054e-20	-2.9289e-21	1.7095e-19	8.6507e-20	3.1043e-20	3.2275e-20	3.7604e-20	1.0000e-03

**Covariance Matrix – Butterfly:**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.0011	2.6833e-05	1.0740e-04	-4.2821e-05	6.2979e-05	1.5991e-05	-2.2336e-04	-1.4255e-04	8.6093e-05	3.0921e-05	2.5165e-06	-2.1066e-06	-1.1579e-04	-2.0431e-04	2.3119e-04
2	2.6833e-05	0.0010	-1.5077e-06	-3.3789e-05	4.2601e-05	1.1349e-05	-6.3548e-05	-1.0159e-04	4.3729e-05	-2.4127e-07	1.7892e-05	8.7876e-07	2.8061e-05	-1.1778e-04	1.1772e-04
3	1.0740e-04	-1.5077e-06	0.0011	-2.0148e-06	1.1369e-05	2.2564e-06	-1.4403e-04	-1.9616e-05	3.2727e-05	3.0647e-05	-1.8726e-05	-3.1096e-06	-1.4694e-04	-6.1080e-05	8.7534e-05
4	-4.2821e-05	-3.3789e-05	-2.0148e-06	0.0010	-6.1773e-05	-1.6426e-05	9.7107e-05	1.4702e-04	-6.4231e-05	-8.4869e-07	-2.5027e-05	-1.1437e-06	-3.4663e-05	1.7194e-04	-1.7289e-04
5	6.2979e-05	4.2601e-05	1.1369e-05	-6.1773e-05	0.0011	2.0955e-05	-1.3463e-04	-1.8751e-04	8.3908e-05	3.5939e-06	3.0105e-05	1.1904e-06	3.1743e-05	-2.2243e-04	2.2581e-04
6	1.5991e-05	1.1349e-05	2.2564e-06	-1.6426e-05	2.0955e-05	0.0010	-3.4798e-05	-4.9763e-05	2.2097e-05	7.3665e-07	8.1465e-06	3.3912e-07	9.5022e-06	-5.8759e-05	5.9469e-05
7	-2.2336e-04	-6.3548e-05	-1.4403e-04	9.7107e-05	-1.3463e-04	-3.4798e-05	0.0014	3.1070e-04	-1.6731e-04	-4.1687e-05	-2.4061e-05	1.8451e-06	1.2477e-04	4.1319e-04	-4.4960e-04
8	-1.4255e-04	-1.0159e-04	-1.9616e-05	1.4702e-04	-1.8751e-04	-4.9763e-05	3.1070e-04	0.0014	-1.9761e-04	-6.4277e-06	-7.3022e-05	-3.0524e-06	-8.5853e-05	5.2563e-04	-5.3184e-04
9	8.6093e-05	4.3729e-05	3.2727e-05	-6.4231e-05	8.3908e-05	2.2097e-05	-1.6731e-04	-1.9761e-04	0.0011	9.7089e-06	2.7448e-05	6.2193e-07	4.0606e-06	-2.4181e-04	2.5048e-04
10	3.0921e-05	-2.4127e-07	3.0647e-05	-8.4869e-07	3.5939e-06	7.3665e-07	-4.1687e-05	-6.4277e-06	9.7089e-06	0.0010	-5.2090e-06	-8.8187e-07	-4.1782e-05	-1.8412e-05	2.5974e-05
11	2.5165e-06	1.7892e-05	-1.8726e-05	-2.5027e-05	3.0105e-05	8.1465e-06	-2.4061e-05	-7.3022e-05	2.7448e-05	-5.2090e-06	0.0010	1.1696e-06	4.5166e-05	-7.8365e-05	7.3984e-05
12	-2.1066e-06	8.7876e-07	-3.1096e-06	-1.1437e-06	1.1904e-06	3.3912e-07	1.8451e-06	-3.0524e-06	6.2193e-07	-8.8187e-07	1.1696e-06	0.0010	5.1847e-06	-2.4462e-06	1.6905e-06
13	-1.1579e-04	2.8061e-05	-1.4694e-04	-3.4663e-05	3.1743e-05	9.5022e-06	1.2477e-04	-8.5853e-05	4.0606e-06	-4.1782e-05	4.5166e-06	5.1847e-06	0.0012	-4.7596e-05	1.1702e-05
14	-2.0431e-04	-1.1778e-04	-6.1080e-05	1.7194e-04	-2.2243e-04	-5.8759e-05	4.1319e-04	5.2563e-04	-2.4181e-04	-1.8412e-05	-7.8365e-05	-2.4462e-06	-4.7596e-05	0.0016	-6.5060e-04
15	2.3119e-04	1.1772e-04	8.7534e-05	-1.7289e-04	2.2581e-04	5.9469e-05	-4.4960e-04	-5.3184e-04	2.5048e-04	2.5974e-05	7.3984e-05	1.6905e-06	1.1702e-05	-6.5060e-04	0.0017

**Covariance Matrix – Face:**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.0011	-2.0316e-06	9.0908e-05	-2.7274e-05	-2.2275e-05	6.7401e-06	-1.8886e-04	-1.0663e-04	3.1514e-06	2.5902e-06	4.7171e-06	3.5506e-06	2.7780e-04	-1.6389e-04	1.0357e-05
2	-2.0316e-06	0.0010	-1.7406e-06	5.2220e-07	4.2650e-07	-1.2905e-07	3.6160e-06	2.0416e-06	-6.0338e-08	-4.9594e-08	-9.0317e-08	-6.7982e-08	-5.3190e-06	3.1379e-06	-1.9830e-07
3	9.0908e-05	-1.7406e-06	0.0011	-2.3367e-05	-1.9084e-05	5.7746e-06	-1.6181e-04	-9.1353e-05	2.6999e-06	2.2192e-06	4.0414e-06	3.0420e-06	2.3801e-04	-1.4041e-04	8.8733e-06
4	-2.7274e-05	5.2220e-07	-2.3367e-05	0.0010	5.7256e-06	-1.7325e-06	4.8544e-05	2.7407e-05	-8.1002e-07	-6.6579e-07	-1.2125e-06	-9.1264e-07	-7.1407e-05	4.2125e-05	-2.6621e-06
5	-2.2275e-05	4.2650e-07	-1.9084e-05	5.7256e-06	0.0010	-1.4150e-06	3.9647e-05	2.2384e-05	-6.6157e-07	-5.4377e-07	-9.9026e-07	-9.9026e-07	5.8320e-05	3.4405e-05	-2.1742e-06
6	6.7401e-06	-1.2905e-07	5.7746e-06	-1.7325e-06	-1.4150e-06	0.0010	-1.1997e-05	-6.7731e-06	2.0018e-07	1.6454e-07	2.2964e-07	2.2554e-07	1.7647e-05	-1.0410e-05	6.5789e-07
7	-1.8886e-04	3.6160e-06	-1.6181e-04	4.8544e-05	3.9647e-05	-1.1997e-05	0.0013	1.8978e-04	-5.6091e-06	-4.6103e-06	-8.3959e-06	-6.3196e-06	-4.9446e-04	2.9170e-04	-1.8434e-05
8	-1.0663e-04	2.0416e-06	-9.1353e-05	2.7407e-05	2.2384e-05	-6.7731e-06	1.8978e-04	0.0011	-3.1668e-06	-2.6029e-06	-4.7402e-06	-3.5680e-06	-2.7917e-04	1.6469e-04	-1.0408e-05
9	3.1514e-06	-6.0338e-08	2.6999e-06	-8.1002e-07	6.6157e-07	2.0018e-07	-5.6091e-06	-3.1668e-06	0.0010	7.6929e-08	1.4010e-07	1.0545e-07	8.2507e-06	-4.8674e-06	3.0760e-07
10	2.5902e-06	-4.9594e-08	2.2192e-06	-6.6579e-07	-5.4377e-07	1.6454e-07	-4.6103e-06	-2.6029e-06	7.6929e-08	0.0010	1.1515e-07	8.6675e-08	6.7816e-06	-4.0007e-06	2.5283e-07
11	4.7171e-06	-9.0317e-08	4.0414e-06	-1.2125e-06	-9.9026e-07	2.9964e-07	-8.3959e-06	-4.7402e-06	1.4010e-07	1.1515e-07	0.0010	1.5784e-07	1.2350e-05	-7.2857e-06	4.6042e-07
12	3.5506e-06	-6.7982e-08	3.0420e-06	-9.1264e-07	-7.1407e-05	2.2554e-07	-6.3196e-06	-3.5680e-06	1.0545e-07	8.6675e-08	1.5784e-07	0.0010	9.2960e-06	-5.4840e-06	3.4656e-07
13	2.7780e-04	-5.3190e-06	2.3801e-04	-7.1407e-05	-5.8320e-05	1.7647e-05	-4.9446e-04	-2.7917e-04	8.2507e-06	6.7816e-06	1.2350e-05	9.2960e-06	0.0017	-4.2908e-04	2.7116e-05
14	-1.6389e-04	3.1379e-06	-1.4041e-04	4.2125e-05	3.4405e-05	-1.0410e-05	2.9170e-04	1.6469e-04	-4.8674e-06	-4.0007e-06	-7.2857e-06	-5.4840e-06	-4.2908e-04	0.0013	-1.5996e-05
15	1.0357e-05	-1.9830e-07	8.8733e-06	-2.6621e-06	-2.1742e-06	6.5789e-07	-1.8434e-05	-1.0408e-05	3.0760e-07	2.5283e-07	4.6042e-07	3.4656e-07	2.7116e-05	-1.5996e-05	0.0010

**Covariance Matrix – Star:**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.0010	1.0162e-06	-5.6558e-06	1.6802e-06	-4.4162e-06	2.4695e-06	-3.6485e-05	2.0042e-06	1.1110e-06	-7.2860e-06	5.2473e-06	1.9551e-06	3.2742e-05	-2.6794e-06	1.8332e-06
2	1.0162e-06	0.0010	-9.6938e-07	2.8797e-07	-7.5693e-07	4.2327e-07	-6.2535e-06	3.4351e-07	1.9042e-07	-1.2488e-06	8.9937e-07	3.3510e-07	5.6118e-06	-4.5924e-07	3.1420e-07
3	-5.6558e-06	-9.6938e-07	0.0010	-1.6027e-06	4.2127e-06	-2.3557e-06	3.4804e-05	-1.9118e-06	-1.0598e-06	6.9502e-06	-5.0054e-06	-1.8650e-06	-3.1233e-05	2.5559e-06	-1.7487e-06
4	1.6802e-06	2.8797e-07	-1.6027e-06	0.0010	-1.2515e-06	6.9981e-07	-1.0339e-05	5.6794e-07	3.1483e-07	-2.0647e-06	1.4870e-06	5.5403e-07	9.2783e-06	-7.5928e-07	5.1948e-07
5	-4.4162e-06	-7.5693e-07	4.2127e-06	-1.2515e-06	0.0010	-1.8394e-06	2.7176e-06	-1.4928e-06	-8.2753e-07	5.4270e-06	-3.9084e-06	-1.4563e-06	-2.4388e-05	1.9957e-06	-1.3654e-06
6	2.4695e-06	4.2327e-07	-2.3557e-06	6.9981e-07	-1.8394e-06	0.0010	-1.5197e-05	8.3477e-07	4.6275e-07	-3.0347e-06	2.1856e-06	8.1433e-07	1.3637e-05	-1.1160e-06	7.6355e-07
7	-3.6485e-05	-6.2535e-06	3.4804e-05	-1.0339e-05	2.7176e-05	-1.5197e-05	0.0012	-1.2333e-05	-8.8367e-06	4.4836e-05	-3.2290e-05	-1.2031e-05	-2.0148e-04	1.6488e-05	-1.1281e-05
8	2.0042e-06	3.4351e-07	-1.9118e-06	5.6794e-07	-1.4928e-06	8.3477e-07	-1.2333e-05	0.0010	3.7555e-07	-2.4629e-06	1.7737e-06	6.6088e-07	1.1068e-05	-9.0570e-07	6.1966e-07
9	1.1110e-06	1.9042e-07	-1.0598e-06	3.1483e-07	-8.2753e-07	4.6275e-07	-6.8367e-06	3.7555e-07	0.0010	-1.3653e-06	9.8325e-07	3.6635e-07	6.1353e-06	-5.0207e-07	3.4351e-07
10	-7.2860e-06	-1.2488e-06	6.9502e-06	-2.0647e-06	5.4270e-06	-3.0347e-06	4.4836e-05	-2.4629e-06	-1.3653e-06	0.0010	-6.4482e-06	-2.4026e-06	-4.0236e-05	3.9267e-06	-2.2527e-06
11	5.2473e-06	8.9937e-07	-5.0054e-06	1.4870e-06	-3.9084e-06	2.1856e-06	-3.2290e-05	1.7737e-06	9.8325e-07	-6.4482e-06	0.0010	1.7303e-06	2.8977e-05	-2.3713e-06	1.6224e-06
12	1.9551e-06	3.3510e-07	-1.8650e-06	5.5403e-07	-1.4563e-06	8.1433e-07	-1.2031e-05	6.6088e-07	3.6635e-07	-2.4026e-06	1.7303e-06	0.0010	1.0797e-05	-8.8353e-07	6.0449e-07
13	3.2742e-05	5.6118e-06	-3.1233e-05	9.2783e-06	-2.4388e-05	1.3637e-05	-2.0148e-04	1.1068e-05	6.1353e-06	-4.0236e-05	2.8977e-05	1.0797e-05	0.0012	-1.4796e-05	1.0123e-05
14	-2.6794e-06	-4.5924e-07	2.5559e-06	-7.5928e-07	1.9957e-06	-1.1160e-06	1.6488e-05	-9.0570e-07	-5.0207e-07	3.9267e-06	-2.3713e-06	-8.8353e-07	-1.4796e-05	0.0010	-8.2843e-07
15	1.8332e-06	3.1420e-07	-1.7487e-06	5.1948e-07	-1.3654e-06	7.6355e-07	-1.1281e-05	6.1966e-07	3.4351e-07	-2.2527e-06	1.6224e-06	6.0449e-07	1.0123e-05	-8.2843e-07	0.0010



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.0020	-9.9453e-05	0.0015	-1.6927e-04	-5.0563e-04	3.7174e-05	-2.3187e-04	-7.1254e-04	-4.9760e-05	-4.5661e-04	5.5844e-06	-1.7610e-05	0.0012	-0.0013	-1.9169e-04
2	-9.9453e-05	0.0010	-1.5476e-04	1.7440e-05	5.2097e-05	-3.8302e-06	2.3890e-05	7.3416e-05	5.1270e-06	4.7047e-05	-5.7538e-07	1.8144e-06	-1.2644e-04	1.3673e-04	1.9750e-05
3	0.0015	-1.5476e-04	0.0033	-2.6340e-04	-7.8683e-04	5.7847e-05	-3.6081e-04	-0.0011	-7.7432e-05	-7.1055e-04	8.6900e-06	-2.7403e-05	0.0019	-0.0021	-2.9829e-04
4	-1.6927e-04	1.7440e-05	-2.6340e-04	0.0010	8.8669e-05	-6.5189e-06	4.0661e-05	1.2495e-04	8.7260e-06	8.0072e-05	-9.7929e-07	3.0881e-06	-2.1519e-04	2.3271e-04	3.3615e-05
5	-5.0563e-04	5.2097e-05	-7.8683e-04	8.8669e-05	0.0013	-1.9473e-05	1.2146e-04	3.7326e-04	2.6066e-05	2.3919e-04	-2.9253e-06	9.2248e-06	-6.4282e-04	6.9516e-04	1.0041e-04
6	3.7174e-05	-3.8302e-06	5.7847e-05	-6.5189e-06	-1.9473e-05	0.0010	-8.9298e-06	-2.7442e-05	-1.9164e-06	-1.7585e-05	2.1507e-07	-6.7820e-07	4.7260e-05	-5.1108e-05	-7.3823e-06
7	-2.3187e-04	2.3890e-05	-3.6081e-04	4.0661e-05	1.2146e-04	-8.9298e-06	0.0011	1.7116e-04	1.1953e-05	1.0969e-04	-1.3415e-06	4.2302e-06	-2.9478e-04	3.1878e-04	4.6046e-05
8	-7.1254e-04	7.3416e-05	-0.0011	1.2495e-04	3.7326e-04	-2.7442e-05	1.7116e-04	0.0015	3.6733e-05	3.3707e-04	-4.1224e-06	1.3000e-05	-9.0587e-04	9.7962e-04	1.4150e-04
9	-4.9760e-05	5.1270e-06	-7.7432e-05	8.7260e-06	2.6066e-05	-1.9164e-06	1.1953e-05	3.6733e-05	0.0010	2.3539e-05	-2.8788e-07	9.0782e-07	-6.3261e-05	6.8411e-05	9.8818e-06
10	-4.5661e-04	4.7047e-05	-7.1055e-04	8.0072e-05	2.3919e-04	-1.7585e-05	1.0969e-04	3.3707e-04	2.3539e-05	0.0012	-2.6417e-06	8.3305e-06	-5.8050e-04	6.2776e-04	9.0678e-05
11	5.5844e-06	-5.7538e-07	8.6900e-06	-9.7929e-07	-2.9253e-06	2.1507e-07	-1.3415e-06	-4.1224e-06	-2.8788e-07	-2.6417e-06	0.0010	-1.0188e-07	7.0996e-06	-7.6776e-06	-1.1090e-06
12	-1.7610e-05	1.8144e-06	-2.7403e-05	3.0881e-06	9.2248e-06	-6.7820e-07	4.2302e-06	1.3000e-05	9.0782e-07	8.3305e-06	-1.0188e-07	0.0010	-2.2388e-05	2.4211e-05	3.4971e-06
13	0.0012	-1.2644e-04	0.0019	-2.1519e-04	-6.4282e-04	4.7260e-05	-2.9478e-04	-9.0587e-04	-6.3261e-05	-5.8050e-04	7.0996e-06	-2.2388e-05	0.0026	-0.0017	-2.4370e-04
14	-0.0013	1.3673e-04	-0.0021	2.3271e-04	6.9516e-04	-5.1108e-05	3.1878e-04	9.7962e-04	6.8411e-05	6.2776e-04	-7.6776e-06	2.4211e-05	-0.0017	0.0028	2.6354e-04
15	-1.9169e-04	1.9750e-05	-2.9829e-04	3.3615e-05	1.0041e-04	-7.3823e-06	4.6046e-05	1.4150e-04	9.8818e-06	9.0678e-05	-1.1090e-06	3.4971e-06	-2.4370e-04	2.6354e-04	0.0010

Figure 9a: Unsupervised Confusion Matrix 4 (k-Means – PCA– 15 Features)

Number of Clusters: 6

Training Directory: './images/train' (Provided)

Testing Directory: './images/test' (Provided)

Unsupervised Method: GMM based on k-Means Clustering

Supervised Feature Type: PCA – 15 Features

		PREDICTED CLASSES						
		Alien	Arrow	Butterfly	Face	Star	Toonhead	TOTAL
ACTUAL CLASSES	Alien	0	31	3	0	0	8	42
	Arrow	0	2	0	0	0	0	2
	Butterfly	0	10	40	0	0	0	50
	Face	0	12	3	46	0	38	99
	Star	0	2	20	0	0	4	26
	Toonhead	0	0	0	1	0	1	2
	TOTAL	0	57	66	47	0	51	221

% Accuracy (Aliens): 0.0000%

% Accuracy (Arrow): 100.0000%

% Accuracy (Butterfly): 80.0000%

% Accuracy (Face): 46.4646%

% Accuracy (Star): 0.0000%

% Accuracy (Toonhead): 50.0000%

% Accuracy (Overall): 40.2715%

Figure 9b: k-Means Clustering Plot 4 (Compared to PCA Supervised Model – 15 Features)

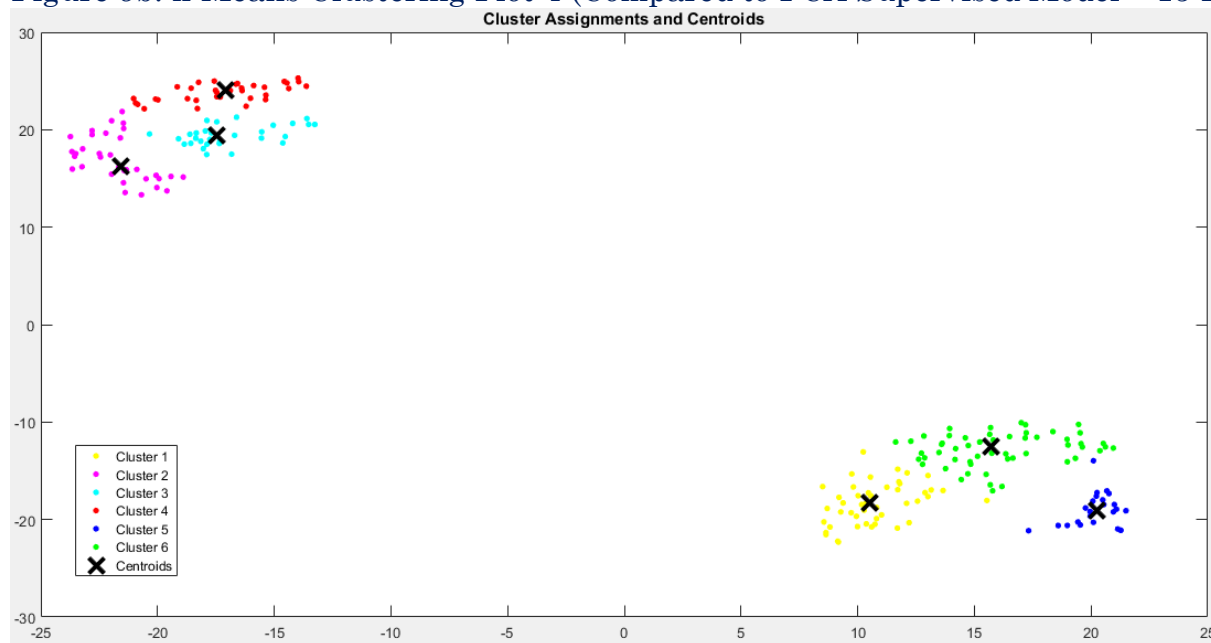


Figure 9c: k-Means Actual Classifications Plot 4 (For Visual Comparison)

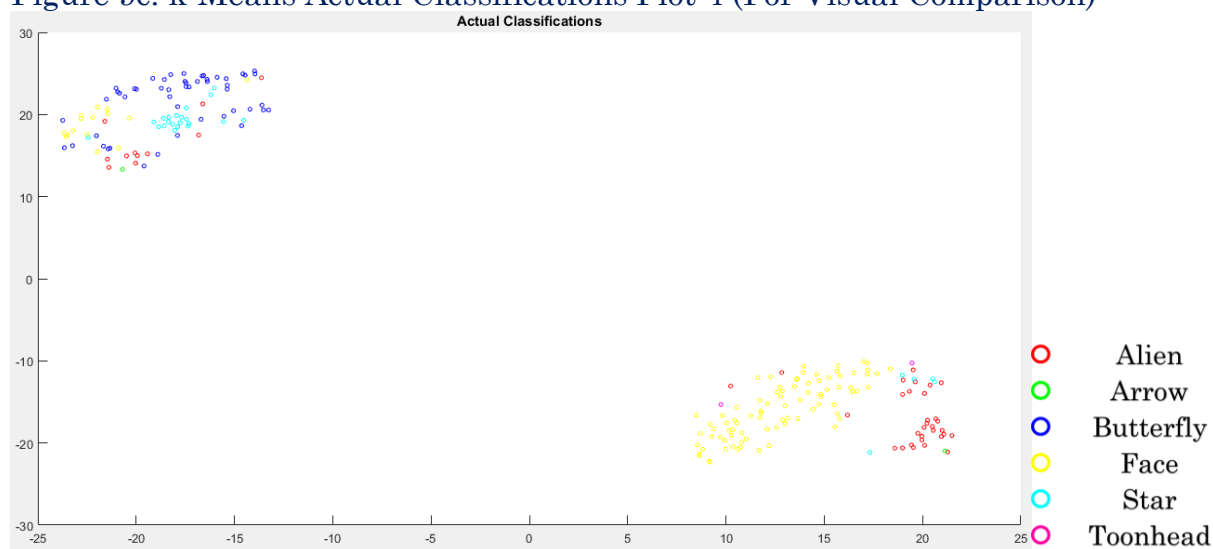


Figure 9d: k-Means Silhouette Value Plot 4

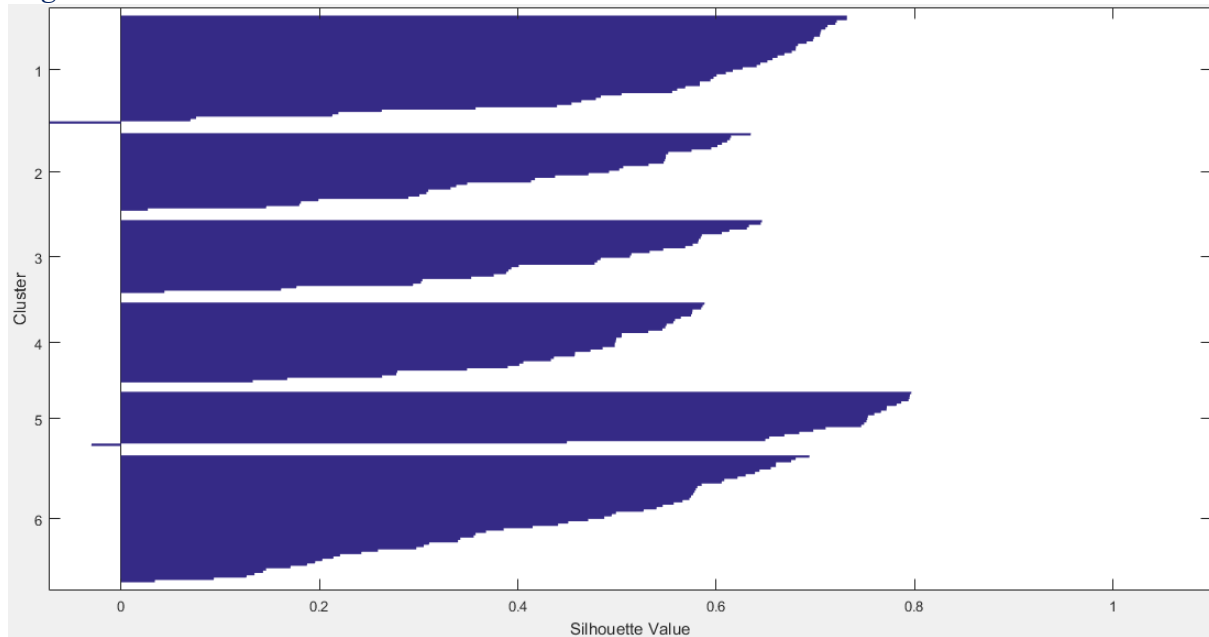


Figure 9e: k-Means Unsupervised GMM Model 4

unsupervised_models						
1x6 struct with 6 fields						
Fields	kMeansIdx	mean	cov	prior	closestLabel	closestClass
1	'1'	15x1 double	15x15 double	0.2081	4 'Face'	
2	'2'	15x1 double	15x15 double	0.1493	2 'Arrow'	
3	'3'	15x1 double	15x15 double	0.1403	3 'Butterfly'	
4	'4'	15x1 double	15x15 double	0.1538	3 'Butterfly'	
5	'5'	15x1 double	15x15 double	0.1041	2 'Arrow'	
6	'6'	15x1 double	15x15 double	0.2443	6 'Toonhead'	

Covariance Matrix – Cluster 1 (closest to Face):

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.2065	0.2055	0.2366	0.2553	0.2637	0.2699	0.1996	0.0757	8.0111e-04	-0.0201	-0.0585	-0.0659	-0.0840	-0.1038	-0.1305
2	0.2055	0.2065	0.2366	0.2553	0.2637	0.2699	0.1996	0.0757	8.0111e-04	-0.0201	-0.0585	-0.0659	-0.0840	-0.1038	-0.1305
3	0.2366	0.2366	0.2770	0.2996	0.3103	0.3182	0.2440	0.0916	0.0046	-0.0258	-0.0721	-0.0850	-0.1076	-0.1329	-0.1631
4	0.2553	0.2553	0.2996	0.3272	0.3383	0.3471	0.2706	0.1011	0.0069	-0.0292	-0.0802	-0.0964	-0.1218	-0.1502	-0.1826
5	0.2637	0.2637	0.3103	0.3383	0.3519	0.3602	0.2826	0.1054	0.0079	-0.0308	-0.0839	-0.1016	-0.1282	-0.1581	-0.1914
6	0.2699	0.2699	0.3182	0.3471	0.3602	0.3708	0.2914	0.1086	0.0087	-0.0319	-0.0866	-0.1054	-0.1329	-0.1639	-0.1979
7	0.1996	0.1996	0.2440	0.2706	0.2826	0.2914	0.8020	0.0028	-0.0790	-0.1290	-0.1664	-0.1940	-0.2551	-0.3170	-0.3253
8	0.0757	0.0757	0.0916	0.1011	0.1054	0.1086	0.0028	0.7445	0.5627	0.3664	0.3012	0.2085	0.1676	0.1458	0.0956
9	8.0111e-04	8.0111e-04	0.0046	0.0069	0.0079	0.0087	-0.0790	0.5627	0.6840	0.5475	0.4816	0.4072	0.3690	0.3352	0.3041
10	-0.0201	-0.0201	-0.0258	-0.0292	-0.0308	-0.0319	-0.1290	0.3664	0.5475	0.6142	0.5462	0.4725	0.4246	0.3734	0.3472
11	-0.0585	-0.0585	-0.0721	-0.0802	-0.0839	-0.0866	-0.1664	0.3012	0.4816	0.5462	0.5833	0.5216	0.4801	0.4489	0.4163
12	-0.0659	-0.0659	-0.0850	-0.0964	-0.1016	-0.1054	-0.1940	0.2085	0.4072	0.4725	0.5216	0.5142	0.4809	0.4515	0.4283
13	-0.0840	-0.0840	-0.1076	-0.1218	-0.1282	-0.1329	-0.2551	0.1676	0.3690	0.4246	0.4801	0.4809	0.4920	0.4590	0.4313
14	-0.1038	-0.1038	-0.1329	-0.1502	-0.1581	-0.1639	-0.3170	0.1458	0.3352	0.3734	0.4489	0.4515	0.4590	0.5640	0.5254
15	-0.1305	-0.1305	-0.1631	-0.1826	-0.1914	-0.1979	-0.3253	0.0956	0.3041	0.3472	0.4163	0.4283	0.4313	0.5254	0.5199

Covariance Matrix – Cluster 2 (closest to Arrow):

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1.6849	1.6839	1.7641	1.7842	1.7945	1.8054	1.4128	1.0162	0.6159	0.3342	0.2705	0.1464	0.0449	0.0382	0.0626
2	1.6839	1.6849	1.7641	1.7842	1.7945	1.8054	1.4128	1.0162	0.6159	0.3342	0.2705	0.1464	0.0449	0.0382	0.0626
3	1.7641	1.7641	1.9057	1.9532	1.9777	1.9996	1.5889	1.1665	0.7366	0.4302	0.3446	0.2027	0.0859	0.0798	0.1005
4	1.7842	1.7842	1.9532	2.1099	2.1932	2.2550	1.8280	1.3598	0.8848	0.5234	0.4007	0.2186	0.0810	0.0781	0.0891
5	1.7945	1.7945	1.9777	2.1932	2.3155	2.4010	1.9647	1.4685	0.9669	0.5718	0.4261	0.2203	0.0671	0.0618	0.0637
6	1.8054	1.8054	1.9996	2.2550	2.4010	2.5052	2.0607	1.5463	1.0265	0.6092	0.4479	0.2255	0.0609	0.0541	0.0495
7	1.4128	1.4128	1.5889	1.8280	1.9647	2.0607	2.0712	1.4806	0.9490	0.5749	0.4102	0.1570	0.0283	-0.0076	-0.0214
8	1.0162	1.0162	1.1665	1.3598	1.4685	1.5463	1.4806	1.6978	1.1246	0.7396	0.5224	0.3083	0.1458	0.0430	0.0070
9	0.6159	0.6159	0.7366	0.8848	0.9669	1.0265	0.9490	1.1246	0.9892	0.7699	0.5848	0.4033	0.2212	0.1317	0.0883
10	0.3342	0.3342	0.4302	0.5234	0.5718	0.6092	0.5749	0.7396	0.7699	0.8527	0.7509	0.6042	0.4848	0.4195	0.3924
11	0.2705	0.2705	0.3446	0.4007	0.4261	0.4479	0.4102	0.5224	0.5848	0.7509	0.8580	0.7812	0.7473	0.7554	0.7668
12	0.1464	0.1464	0.2027	0.2186	0.2203	0.2255	0.1570	0.3083	0.4033	0.6042	0.7812	0.8640	0.9059	0.9863	1.0223
13	0.0449	0.0449	0.0859	0.0810	0.0671	0.0609	0.0283	0.1458	0.2212	0.4848	0.7473	0.9059	1.0802	1.2351	1.3111
14	0.0382	0.0382	0.0798	0.0781	0.0618	0.0541	-0.0076	0.0430	0.1317	0.4195	0.7554	0.9863	1.2351	1.4741	1.5748
15	0.0626	0.0626	0.1005	0.0891	0.0637	0.0495	-0.0214	0.0070	0.0883	0.3924	0.7668	1.0223	1.3111	1.5748	1.7087

**Covariance Matrix – Cluster 3 (closest to Butterfly):**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.2170	0.2160	0.2165	0.2186	0.2192	0.2189	0.1655	0.1274	-0.0101	-0.0984	-0.1497	-0.1680	-0.2992	-0.3396	-0.3399
2	0.2160	0.2170	0.2165	0.2186	0.2192	0.2189	0.1655	0.1274	-0.0101	-0.0984	-0.1497	-0.1680	-0.2992	-0.3396	-0.3399
3	0.2165	0.2165	0.2181	0.2194	0.2201	0.2199	0.1664	0.1282	-0.0093	-0.0976	-0.1491	-0.1676	-0.2995	-0.3401	-0.3380
4	0.2186	0.2186	0.2194	0.2233	0.2233	0.2232	0.1697	0.1312	-0.0062	-0.0948	-0.1463	-0.1665	-0.2997	-0.3401	-0.3320
5	0.2192	0.2192	0.2201	0.2233	0.2259	0.2250	0.1709	0.1324	-0.0050	-0.0938	-0.1457	-0.1653	-0.3009	-0.3422	-0.3288
6	0.2189	0.2189	0.2199	0.2232	0.2250	0.2265	0.1708	0.1322	-0.0053	-0.0940	-0.1464	-0.1646	-0.3021	-0.3443	-0.3282
7	0.1655	0.1655	0.1664	0.1697	0.1709	0.1708	0.1570	0.1295	0.0201	-0.0583	-0.0970	-0.1204	-0.2275	-0.2511	-0.2417
8	0.1274	0.1274	0.1282	0.1312	0.1324	0.1322	0.1295	0.1180	0.0287	-0.0447	-0.0755	-0.0945	-0.1647	-0.1735	-0.1594
9	-0.0101	-0.0101	-0.0093	-0.0062	-0.0050	-0.0053	0.0201	0.0287	0.1446	0.1475	0.1322	0.0723	0.0359	0.0144	0.0133
10	-0.0984	-0.0984	-0.0976	-0.0948	-0.0938	-0.0940	-0.0583	-0.0447	0.1475	0.2466	0.2633	0.2107	0.1770	0.1556	0.1517
11	-0.1497	-0.1497	-0.1491	-0.1463	-0.1457	-0.1464	-0.0970	-0.0755	0.1322	0.2633	0.3145	0.2788	0.2781	0.2777	0.2843
12	-0.1680	-0.1680	-0.1676	-0.1665	-0.1653	-0.1646	-0.1204	-0.0945	0.0723	0.2107	0.2788	0.2936	0.3206	0.3332	0.3541
13	-0.2992	-0.2992	-0.2995	-0.2997	-0.3009	-0.3021	-0.2275	-0.1647	0.0359	0.1770	0.2781	0.3206	0.5766	0.6586	0.7072
14	-0.3396	-0.3396	-0.3401	-0.3401	-0.3422	-0.3443	-0.2511	-0.1735	0.0144	0.1556	0.2777	0.3332	0.6586	0.8232	0.9006
15	-0.3399	-0.3399	-0.3380	-0.3320	-0.3288	-0.3282	-0.2417	-0.1594	0.0133	0.1517	0.2843	0.3541	0.7072	0.9006	1.0829

**Covariance Matrix – Cluster 4 (closest to Butterfly):**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.1245	0.1235	0.1235	0.1157	0.1115	0.1105	0.1144	0.1102	0.0276	-0.1049	-0.1571	-0.1951	-0.2239	-0.2229	-0.2079
2	0.1235	0.1245	0.1235	0.1157	0.1115	0.1105	0.1144	0.1102	0.0276	-0.1049	-0.1571	-0.1951	-0.2239	-0.2229	-0.2079
3	0.1235	0.1235	0.1245	0.1157	0.1115	0.1105	0.1144	0.1102	0.0276	-0.1049	-0.1571	-0.1951	-0.2239	-0.2229	-0.2079
4	0.1157	0.1157	0.1157	0.1114	0.1075	0.1068	0.1116	0.1100	0.0304	-0.0982	-0.1470	-0.1820	-0.2077	-0.2042	-0.1889
5	0.1115	0.1115	0.1115	0.1075	0.1063	0.1048	0.1101	0.1099	0.0318	-0.0945	-0.1415	-0.1749	-0.1988	-0.1941	-0.1786
6	0.1105	0.1105	0.1105	0.1068	0.1048	0.1053	0.1097	0.1099	0.0322	-0.0936	-0.1402	-0.1732	-0.1967	-0.1916	-0.1761
7	0.1144	0.1144	0.1144	0.1116	0.1101	0.1097	0.1216	0.1235	0.0443	-0.0876	-0.1389	-0.1775	-0.2041	-0.1893	-0.1637
8	0.1102	0.1102	0.1102	0.1100	0.1099	0.1099	0.1235	0.1311	0.0519	-0.0801	-0.1319	-0.1715	-0.1987	-0.1785	-0.1507
9	0.0276	0.0276	0.0276	0.0304	0.0318	0.0322	0.0443	0.0519	0.1526	0.1056	0.1050	0.0859	0.0746	0.0919	0.1114
10	-0.1049	-0.1049	-0.1049	-0.0982	-0.0945	-0.0936	-0.0876	-0.0801	0.1056	0.2564	0.3281	0.3657	0.3854	0.3992	0.3973
11	-0.1571	-0.1571	-0.1571	-0.1470	-0.1415	-0.1402	-0.1389	-0.1319	0.1050	0.3281	0.4765	0.5518	0.5948	0.6123	0.6001
12	-0.1951	-0.1951	-0.1951	-0.1820	-0.1749	-0.1732	-0.1775	-0.1715	0.0859	0.3657	0.5518	0.6725	0.7395	0.7517	0.7315
13	-0.2239	-0.2239	-0.2239	-0.2077	-0.1988	-0.1967	-0.2041	-0.1987	0.0746	0.3854	0.5948	0.7395	0.8375	0.8495	0.8253
14	-0.2229	-0.2229	-0.2229	-0.2042	-0.1941	-0.1916	-0.1893	-0.1785	0.0919	0.3992	0.6123	0.7517	0.8495	0.8901	0.8740
15	-0.2079	-0.2079	-0.2079	-0.1889	-0.1786	-0.1761	-0.1637	-0.1507	0.1114	0.3973	0.6001	0.7315	0.8253	0.8740	0.8989

**Covariance Matrix – Cluster 5 (closest to Arrow):**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2.6540	2.6530	2.1050	1.5767	1.3241	1.0961	0.8201	0.5315	0.3929	0.2408	0.1159	0.0638	-0.0447	-0.1086	-0.1452
2	2.6530	2.6540	2.1050	1.5767	1.3241	1.0961	0.8201	0.5315	0.3929	0.2408	0.1159	0.0638	-0.0447	-0.1086	-0.1452
3	2.1050	2.1050	1.7869	1.3728	1.1688	0.9891	0.7366	0.4197	0.2895	0.1555	0.0514	0.0101	-0.0696	-0.0970	-0.1042
4	1.5767	1.5767	1.3728	1.2932	1.1735	1.0534	0.8036	0.4475	0.3508	0.2348	0.1730	0.1285	0.0648	0.0884	0.1085
5	1.3241	1.3241	1.1688	1.1735	1.1149	1.0274	0.7878	0.4046	0.3179	0.2141	0.1772	0.1367	0.0823	0.1306	0.1662
6	1.0961	1.0961	0.9891	1.0534	1.0274	0.9856	0.7815	0.3870	0.3154	0.2297	0.2124	0.1816	0.1410	0.2094	0.2600
7	0.8201	0.8201	0.7366	0.8036	0.7878	0.7815	0.7348	0.5572	0.5153	0.4532	0.4356	0.4019	0.3542	0.3849	0.4095
8	0.5315	0.5315	0.4197	0.4475	0.4046	0.3870	0.5572	0.8558	0.8548	0.8162	0.7786	0.7269	0.6583	0.5827	0.5235
9	0.3929	0.3929	0.2895	0.3508	0.3179	0.3154	0.5153	0.8548	0.8820	0.8717	0.8477	0.8061	0.7488	0.6780	0.6212
10	0.2408	0.2408	0.1555	0.2348	0.2141	0.2297	0.4532	0.8162	0.8717	0.9391	0.9230	0.9097	0.8876	0.8245	0.7702
11	0.1159	0.1159	0.0514	0.1730	0.1772	0.2124	0.4356	0.7786	0.8477	0.9230	0.9491	0.9455	0.9352	0.8943	0.8479
12	0.0638	0.0638	0.0101	0.1285	0.1367	0.1816	0.4019	0.7269	0.8061	0.9097	0.9455	0.9614	0.9726	0.9456	0.9050
13	-0.0447	-0.0447	-0.0696	0.0648	0.0823	0.1410	0.3542	0.6583	0.7488	0.8876	0.9352	0.9726	1.0218	1.0132	0.9809
14	-0.1086	-0.1086	-0.0970	0.0884	0.1306	0.2094	0.3849	0.5827	0.6780	0.8245	0.8943	0.9456	1.0132	1.0546	1.0474
15	-0.1452	-0.1452	-0.1042	0.1085	0.1662	0.2600	0.4095	0.5235	0.6212	0.7702	0.8479	0.9050	0.9809	1.0474	1.0688

**Covariance Matrix – Cluster 6 (closest to Toonhead):**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.2433	0.2423	0.2559	0.2636	0.2677	0.2649	0.2307	0.0509	-0.0061	-0.0393	-0.1127	-0.1558	-0.2011	-0.2126	-0.2685
2	0.2423	0.2433	0.2559	0.2636	0.2677	0.2649	0.2307	0.0509	-0.0061	-0.0393	-0.1127	-0.1558	-0.2011	-0.2126	-0.2685
3	0.2559	0.2559	0.2786	0.2887	0.2944	0.2925	0.2561	0.0474	-0.0162	-0.0540	-0.1307	-0.1737	-0.2215	-0.2382	-0.2965
4	0.2636	0.2636	0.2887	0.3071	0.3163	0.3166	0.2829	0.0595	-0.0089	-0.0514	-0.1322	-0.1803	-0.2321	-0.2562	-0.3195
5	0.2677	0.2677	0.2944	0.3163	0.3314	0.3325	0.3012	0.0630	-0.0097	-0.0513	-0.1351	-0.1856	-0.2385	-0.2659	-0.3317
6	0.2649	0.2649	0.2925	0.3166	0.3325	0.3369	0.3060	0.0625	-0.0138	-0.0546	-0.1381	-0.1866	-0.2369	-0.2637	-0.3285
7	0.2307	0.2307	0.2561	0.2829	0.3012	0.3060	0.5344	0.1318	0.0395	-0.0216	-0.0992	-0.1686	-0.2538	-0.2897	-0.3375
8	0.0509	0.0509	0.0474	0.0595	0.0630	0.0625	0.1318	0.4449	0.3866	0.2703	0.1942	0.1796	0.1589	0.1083	0.0431
9	-0.0061	-0.0061	-0.0162	-0.0089	-0.0097	-0.0138	0.0395	0.3866	0.4581	0.4032	0.3180	0.3138	0.2946	0.2502	0.1775
10	-0.0393	-0.0393	-0.0540	-0.0514	-0.0513	-0.0546	-0.0216	0.2703	0.4032	0.4849	0.4244	0.4397	0.4423	0.3996	0.2997
11	-0.1127	-0.1127	-0.1307	-0.1322	-0.1351	-0.1381	-0.0992	0.1942	0.3180	0.4244	0.5277	0.5709	0.5863	0.5414	0.4704
12	-0.1558	-0.1558	-0.1737	-0.1803	-0.1856	-0.1866	-0.1686	0.1796	0.3138	0.4397	0.5709	0.7418	0.8079	0.7763	0.7344
13	-0.2011	-0.2011	-0.2215	-0.2321	-0.2385	-0.2369	-0.2538	0.1589	0.2946	0.4423	0.5863	0.8079	0.9579	0.9669	0.9449
14	-0.2126	-0.2126	-0.2382	-0.2562	-0.2659	-0.2637	-0.2897	0.1083	0.2502	0.3996	0.5414	0.7763	0.9669	1.0703	1.0879
15	-0.2685	-0.2685	-0.2965	-0.3195	-0.3317	-0.3285	-0.3375	0.0431	0.1775	0.2997	0.4704	0.7344	0.9449	1.0879	1.2862

Figure 9f: Supervised GMM Model 4 (PCA – 15 Features)

models				
1x6 struct with 4 fields				
Fields	name	mean	cov	prior
1	'Alien'	15x1 double	15x15 double	0.1892
2	'Arrow'	15x1 double	15x15 double	0.0135
3	'Butterfly'	15x1 double	15x15 double	0.2252
4	'Face'	15x1 double	15x15 double	0.4505
5	'Star'	15x1 double	15x15 double	0.1081
6	'Toonhead'	15x1 double	15x15 double	0.0135

**Covariance Matrix – Alien:**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	20.1685	19.9106	17.3664	15.5369	14.1889	13.1732	11.9467	11.0042	9.8322	8.7892	7.7775	6.8870	6.0111	5.3581	4.5581
2	19.9106	19.6701	17.1684	15.3512	14.0177	13.0146	11.8028	10.8705	9.7074	8.6747	7.6670	6.7833	5.9137	5.2655	4.4701
3	17.3664	17.1684	15.3437	13.8743	12.7719	11.9607	10.9172	10.0845	9.0525	8.1191	7.1873	6.3847	5.5786	4.9954	4.2543
4	15.5369	15.3512	13.8743	12.7236	11.7993	11.1378	10.2454	9.5280	8.6215	7.8029	6.9771	6.2508	5.5220	5.0004	4.3135
5	14.1889	14.0177	12.7719	11.7993	11.0707	10.4939	9.7126	9.0696	8.2403	7.5047	6.7421	6.0402	5.3449	4.8667	4.2104
6	13.1732	13.0146	11.9607	11.1378	10.4939	10.0304	9.3363	8.7466	7.9800	7.3048	6.6025	5.9613	5.3173	4.8702	4.2441
7	11.9467	11.8028	10.9172	10.2454	9.7126	9.3363	8.8028	8.3023	7.6430	7.0598	6.4360	5.8434	5.2477	4.8407	4.2542
8	11.0042	10.8705	10.0845	9.5280	9.0696	8.7466	8.3023	7.9122	7.3409	6.8485	6.3153	5.7660	5.2337	4.8722	4.3199
9	9.8322	9.7074	9.0525	8.6215	8.2403	7.9800	7.6430	7.3409	6.9092	6.5077	6.0439	5.5464	5.0612	4.7343	4.2179
10	8.7892	8.6747	8.1191	7.8029	7.5047	7.3048	7.0598	6.8485	6.5077	6.2232	5.8643	5.4291	5.0094	4.7339	4.2569
11	7.7775	7.6670	7.1873	6.9771	6.7421	6.6025	6.4360	6.3153	6.0439	5.8643	5.6395	5.2926	4.9600	4.7472	4.3169
12	6.8870	6.7833	6.3847	6.2508	6.0402	5.9613	5.8434	5.7660	5.5464	5.4291	5.2926	5.0554	4.7954	4.6306	4.2465
13	6.0111	5.9137	5.5786	5.5220	5.3449	5.3173	5.2477	5.2337	5.0612	5.0094	4.9600	4.7954	4.6345	4.5156	4.2085
14	5.3581	5.2655	4.9954	5.0004	4.8667	4.8702	4.8407	4.8722	4.7343	4.7339	4.7472	4.6306	4.5156	4.4439	4.1747
15	4.5581	4.4701	4.2543	4.3135	4.2104	4.2441	4.2542	4.3199	4.2179	4.2569	4.3169	4.2465	4.2085	4.1747	4.0150

**Covariance Matrix – Arrow:**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	4.3432	4.2567	4.1693	4.1693	4.1693	3.9769	3.9769	3.8804	3.8949	3.9042	3.8109	3.7218	3.7503	3.6652	3.5731
2	4.2567	4.1764	4.0923	4.0923	4.0923	3.9093	3.9093	3.8176	3.8314	3.8402	3.7515	3.6667	3.6938	3.6130	3.5215
3	4.1693	4.0923	4.0145	4.0135	4.0135	3.8402	3.8402	3.7533	3.7664	3.7747	3.6907	3.6104	3.6360	3.5595	3.4687
4	4.1693	4.0923	4.0135	4.0145	4.0135	3.8402	3.8402	3.7533	3.7664	3.7747	3.6907	3.6104	3.6360	3.5595	3.4687
5	4.1693	4.0923	4.0135	4.0135	4.0145	3.8402	3.8402	3.7533	3.7664	3.7747	3.6907	3.6104	3.6360	3.5595	3.4687
6	3.9769	3.9093	3.8402	3.8402	3.8402	3.6891	3.6881	3.6119	3.6233	3.6307	3.5569	3.4865	3.5090	3.4418	3.3527
7	3.9769	3.9093	3.8402	3.8402	3.8402	3.6891	3.6881	3.6119	3.6233	3.6307	3.5569	3.4865	3.5090	3.4418	3.3527
8	3.8804	3.8176	3.7533	3.7533	3.7533	3.6119	3.6119	3.5420	3.5516	3.5585	3.4899	3.4243	3.4453	3.3828	3.2945
9	3.8949	3.8314	3.7664	3.7664	3.7664	3.6233	3.6233	3.5516	3.5634	3.5693	3.5000	3.4337	3.4549	3.3917	3.3032
10	3.9042	3.8402	3.7747	3.7747	3.7747	3.6307	3.6307	3.5585	3.5693	3.5773	3.5064	3.4397	3.4610	3.3973	3.3088
11	3.8109	3.7515	3.6907	3.6907	3.6907	3.5569	3.5569	3.4899	3.5000	3.5064	3.4425	3.3796	3.3994	3.3403	3.2525
12	3.7218	3.6667	3.6104	3.6104	3.6104	3.4865	3.4865	3.4243	3.4337	3.4397	3.3796	3.3232	3.3405	3.2857	3.1988
13	3.7503	3.6938	3.6360	3.6360	3.6360	3.5090	3.5090	3.4453	3.4549	3.4610	3.3994	3.3405	3.3603	3.3032	3.2159
14	3.6652	3.6130	3.5595	3.5595	3.5595	3.4418	3.4418	3.3828	3.3917	3.3973	3.3403	3.2857	3.3032	3.2522	3.1647
15	3.5731	3.5215	3.4687	3.4687	3.4687	3.3527	3.3527	3.2945	3.3032	3.3088	3.2525	3.1988	3.2159	3.1647	3.0816

**Covariance Matrix – Butterfly:**



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	5.7928	5.7918	5.7918	5.7918	5.7918	5.7918	5.7918	5.7918	5.3201	4.6554	4.1159	3.5485	3.0837	2.6410	2.2898
2	5.7918	5.7928	5.7918	5.7918	5.7918	5.7918	5.7918	5.7918	5.3201	4.6554	4.1159	3.5485	3.0837	2.6410	2.2898
3	5.7918	5.7918	5.7928	5.7918	5.7918	5.7918	5.7918	5.7918	5.3201	4.6554	4.1159	3.5485	3.0837	2.6410	2.2898
4	5.7918	5.7918	5.7918	5.7928	5.7918	5.7918	5.7918	5.7918	5.3201	4.6554	4.1159	3.5485	3.0837	2.6410	2.2898
5	5.7918	5.7918	5.7918	5.7918	5.7928	5.7918	5.7918	5.7918	5.3201	4.6554	4.1159	3.5485	3.0837	2.6410	2.2898
6	5.7918	5.7918	5.7918	5.7918	5.7918	5.7928	5.7918	5.7918	5.3201	4.6554	4.1159	3.5485	3.0837	2.6410	2.2898
7	5.7918	5.7918	5.7918	5.7918	5.7918	5.7918	5.7928	5.7918	5.3201	4.6554	4.1159	3.5485	3.0837	2.6410	2.2898
8	5.7918	5.7918	5.7918	5.7918	5.7918	5.7918	5.7918	5.7928	5.3201	4.6554	4.1159	3.5485	3.0837	2.6410	2.2898
9	5.3201	5.3201	5.3201	5.3201	5.3201	5.3201	5.3201	5.3201	5.2701	4.7742	4.3018	3.7927	3.3388	2.9277	2.5734
10	4.6554	4.6554	4.6554	4.6554	4.6554	4.6554	4.6554	4.6554	4.7742	4.5364	4.2140	3.8062	3.4248	3.0924	2.7713
11	4.1159	4.1159	4.1159	4.1159	4.1159	4.1159	4.1159	4.1159	4.3018	4.2140	4.0607	3.7589	3.4504	3.2036	2.9394
12	3.5485	3.5485	3.5485	3.5485	3.5485	3.5485	3.5485	3.5485	3.7927	3.8062	3.7589	3.5794	3.3443	3.1805	2.9891
13	3.0837	3.0837	3.0837	3.0837	3.0837	3.0837	3.0837	3.0837	3.3388	3.4248	3.4504	3.3443	3.1936	3.0872	2.9603
14	2.6410	2.6410	2.6410	2.6410	2.6410	2.6410	2.6410	2.6410	2.9277	3.0924	3.2036	3.2036	3.1805	3.0606	2.9831
15	2.2898	2.2898	2.2898	2.2898	2.2898	2.2898	2.2898	2.2898	2.5734	2.7713	2.9394	2.9891	2.9603	2.9831	2.9825

**Covariance Matrix – Face:**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1.8791	1.8781	1.8781	1.8781	1.8781	1.8781	1.8608	1.2617	1.0353	0.9333	0.8101	0.6175	0.5223	0.4303	0.3156
2	1.8781	1.8791	1.8781	1.8781	1.8781	1.8781	1.8608	1.2617	1.0353	0.9333	0.8101	0.6175	0.5223	0.4303	0.3156
3	1.8781	1.8781	1.8791	1.8781	1.8781	1.8781	1.8608	1.2617	1.0353	0.9333	0.8101	0.6175	0.5223	0.4303	0.3156
4	1.8781	1.8781	1.8781	1.8791	1.8781	1.8781	1.8608	1.2617	1.0353	0.9333	0.8101	0.6175	0.5223	0.4303	0.3156
5	1.8781	1.8781	1.8781	1.8781	1.8791	1.8781	1.8608	1.2617	1.0353	0.9333	0.8101	0.6175	0.5223	0.4303	0.3156
6	1.8781	1.8781	1.8781	1.8781	1.8781	1.8791	1.8608	1.2617	1.0353	0.9333	0.8101	0.6175	0.5223	0.4303	0.3156
7	1.8608	1.8608	1.8608	1.8608	1.8608	1.8608	1.9928	1.2923	1.0606	0.9504	0.8446	0.6390	0.5333	0.4356	0.3236
8	1.2617	1.2617	1.2617	1.2617	1.2617	1.2617	1.2923	1.1247	0.9872	0.9039	0.8011	0.6554	0.5833	0.4917	0.3885
9	1.0353	1.0353	1.0353	1.0353	1.0353	1.0353	1.0606	0.9872	0.9210	0.8434	0.7518	0.6334	0.5765	0.4931	0.4118
10	0.9333	0.9333	0.9333	0.9333	0.9333	0.9333	0.9504	0.9039	0.8434	0.8330	0.7511	0.6455	0.5964	0.5171	0.4343
11	0.8101	0.8101	0.8101	0.8101	0.8101	0.8101	0.8446	0.8011	0.7518	0.7511	0.7205	0.6178	0.5702	0.5008	0.4269
12	0.6175	0.6175	0.6175	0.6175	0.6175	0.6175	0.6390	0.6554	0.6334	0.6455	0.6178	0.6013	0.5703	0.5092	0.4392
13	0.5223	0.5223	0.5223	0.5223	0.5223	0.5223	0.5333	0.5833	0.5765	0.5964	0.5702	0.5703	0.6080	0.5592	0.4957
14	0.4303	0.4303	0.4303	0.4303	0.4303	0.4303	0.4356	0.4917	0.4931	0.5171	0.5008	0.5092	0.5592	0.5704	0.5149
15	0.3156	0.3156	0.3156	0.3156	0.3156	0.3156	0.3236	0.3885	0.4118	0.4343	0.4269	0.4392	0.4957	0.5149	0.5144

**Covariance Matrix – Star:**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	23.3692	23.3682	23.3682	23.3682	23.3682	23.3682	21.7252	21.2078	20.5577	19.7172	18.9837	18.0769	16.0571	13.1699	11.5667
2	23.3682	23.3692	23.3682	23.3682	23.3682	23.3682	21.7252	21.2078	20.5577	19.7172	18.9837	18.0769	16.0571	13.1699	11.5667
3	23.3682	23.3682	23.3692	23.3682	23.3682	23.3682	21.7252	21.2078	20.5577	19.7172	18.9837	18.0769	16.0571	13.1699	11.5667
4	23.3682	23.3682	23.3682	23.3692	23.3682	23.3682	21.7252	21.2078	20.5577	19.7172	18.9837	18.0769	16.0571	13.1699	11.5667
5	23.3682	23.3682	23.3682	23.3682	23.3692	23.3682	21.7252	21.2078	20.5577	19.7172	18.9837	18.0769	16.0571	13.1699	11.5667
6	23.3682	23.3682	23.3682	23.3682	23.3682	23.3692	21.7252	21.2078	20.5577	19.7172	18.9837	18.0769	16.0571	13.1699	11.5667
7	21.7252	21.7252	21.7252	21.7252	21.7252	21.7252	20.2390	19.7457	19.1406	18.3649	17.6770	16.8176	14.9224	12.2077	10.6987
8	21.2078	21.2078	21.2078	21.2078	21.2078	21.2078	19.7457	19.2734	18.6840	17.9288	17.2600	16.4322	14.5899	11.9424	10.4770
9	20.5577	20.5577	20.5577	20.5577	20.5577	20.5577	19.1406	18.6840	18.1244	17.3919	16.7478	15.9534	14.1967	11.6312	10.2167
10	19.7172	19.7172	19.7172	19.7172	19.7172	19.7172	18.3649	17.9288	17.3919	16.7325	16.1126	15.3523	13.6674	11.2284	9.8742
11	18.9837	18.9837	18.9837	18.9837	18.9837	18.9837	17.6770	17.2600	16.7478	16.1126	15.5260	14.8019	13.2029	10.8752	9.5887
12	18.0769	18.0769	18.0769	18.0769	18.0769	18.0769	16.8176	16.4322	15.9534	15.3523	14.8019	14.1727	12.6904	10.4892	9.2968
13	16.0571	16.0571	16.0571	16.0571	16.0571	16.0571	14.9224	14.5899	14.1967	13.6674	13.2029	12.6904	11.5288	9.6256	8.6246
14	13.1699	13.1699	13.1699	13.1699	13.1699	13.1699	12.2077	11.9424	11.6312	11.2284	10.8752	10.4892	9.6256	8.3377	7.6063
15	11.5667	11.5667	11.5667	11.5667	11.5667	11.5667	10.6987	10.4770	10.2167	9.8742	9.5887	9.2968	8.6246	7.6063	7.0638

**Covariance Matrix – Toonhead:**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.9930	0.9920	0.9920	0.9920	0.9920	0.9920	0.9920	0.9920	1.5041	1.6808	1.4103	1.5025	1.1550	1.0775	1.1581
2	0.9920	0.9930	0.9920	0.9920	0.9920	0.9920	0.9920	0.9920	1.5041	1.6808	1.4103	1.5025	1.1550	1.0775	1.1581
3	0.9920	0.9920	0.9930	0.9920	0.9920	0.9920	0.9920	0.9920	1.5041	1.6808	1.4103	1.5025	1.1550	1.0775	1.1581
4	0.9920	0.9920	0.9920	0.9930	0.9920	0.9920	0.9920	0.9920	1.5041	1.6808	1.4103	1.5025	1.1550	1.0775	1.1581
5	0.9920	0.9920	0.9920	0.9920	0.9930	0.9920	0.9920	0.9920	1.5041	1.6808	1.4103	1.5025	1.1550	1.0775	1.1581
6	0.9920	0.9920	0.9920	0.9920	0.9920	0.9930	0.9920	0.9920	1.5041	1.6808	1.4103	1.5025	1.1550	1.0775	1.1581
7	0.9920	0.9920	0.9920	0.9920	0.9920	0.9920	0.9930	0.9920	1.5041	1.6808	1.4103	1.5025	1.1550	1.0775	1.1581
8	0.9920	0.9920	0.9920	0.9920	0.9920	0.9920	0.9920	0.9930	1.5041	1.6808	1.4103	1.5025	1.1550	1.0775	1.1581
9	1.5041	1.5041	1.5041	1.5041	1.5041	1.5041	1.5041	1.5041	3.3801	3.8058	3.2985	3.3068	2.4819	2.1603	2.1491
10	1.6808	1.6808	1.6808	1.6808	1.6808	1.6808	1.6808	1.6808	3.8058	4.2880	3.7175	3.7232	2.7933	2.4283	2.4122
11	1.4103	1.4103	1.4103	1.4103	1.4103	1.4103	1.4103	1.4103	3.2985	3.7175	3.2314	3.2226	2.4138	2.0880	2.0616
12	1.5025	1.5025	1.5025	1.5025	1.5025	1.5025	1.5025	1.5025	3.3068	3.7232	3.2226	3.2400	2.4336	2.1251	2.1222
13	1.1550	1.1550	1.1550	1.1550	1.1550	1.1550	1.1550	1.1550	2.4819	2.7933	2.4138	2.4336	1.8318	1.6048	1.6099
14	1.0775	1.0775	1.0775	1.0775	1.0775	1.0775	1.0775	1.0775	2.1603	2.4283	2.0880	2.1251	1.6048	1.4237	1.4464
15	1.1581	1.1581	1.1581	1.1581	1.1581	1.1581	1.1581	1.1581	2.1491	2.4122	2.0616	2.1222	1.6099	1.4464	1.4938



Figure 10a: Custom Test Set Results (Optimised Black/White Image Properties)

**Number of Features:** 8**Training Directory:** './images/train' (Provided)**Testing Directory:** './images/custom'**Feature Type:** Properties of Binary Image

Test Class	Apple
<b>Prediction Tally</b>	
Alien	1
Arrow	0
Butterfly	2
Face	0
Star	0
Toonhead	0

Test Class	Bat
<b>Prediction Tally</b>	
Alien	0
Arrow	0
Butterfly	5
Face	0
Star	0
Toonhead	0

Test Class	Cup
<b>Prediction Tally</b>	
Alien	0
Arrow	0
Butterfly	4
Face	0
Star	0
Toonhead	0

Test Class	Elephant
<b>Prediction Tally</b>	

<b>Alien</b>	1
<b>Arrow</b>	0
<b>Butterfly</b>	3
<b>Face</b>	0
<b>Star</b>	0
<b>Toonhead</b>	0

<b>Test Class</b>	<b>Heart</b>
<b>Prediction Tally</b>	
<b>Alien</b>	6
<b>Arrow</b>	0
<b>Butterfly</b>	0
<b>Face</b>	0
<b>Star</b>	0
<b>Toonhead</b>	0

<b>Test Class</b>	<b>Tree</b>
<b>Prediction Tally</b>	
<b>Alien</b>	0
<b>Arrow</b>	0
<b>Butterfly</b>	6
<b>Face</b>	0
<b>Star</b>	0
<b>Toonhead</b>	0

Figure 10b: Custom Test Set Results (Optimised Fourier)

**Number of Features:** 6**Training Directory:** './images/train' (Provided)**Testing Directory:** './images/custom'**Feature Type:** Absolute Values of Fourier Coefficients of Chain Code

<b>Test Class</b>	<b>Apple</b>
<b>Prediction Tally</b>	
<b>Alien</b>	3
<b>Arrow</b>	0

<b>Butterfly</b>	0
<b>Face</b>	0
<b>Star</b>	0
<b>Toonhead</b>	0

<b>Test Class</b>	<b>Bat</b>
<b>Prediction Tally</b>	
<b>Alien</b>	3
<b>Arrow</b>	0
<b>Butterfly</b>	2
<b>Face</b>	0
<b>Star</b>	0
<b>Toonhead</b>	0

<b>Test Class</b>	<b>Cup</b>
<b>Prediction Tally</b>	
<b>Alien</b>	3
<b>Arrow</b>	0
<b>Butterfly</b>	1
<b>Face</b>	0
<b>Star</b>	0
<b>Toonhead</b>	0

<b>Test Class</b>	<b>Elephant</b>
<b>Prediction Tally</b>	
<b>Alien</b>	2
<b>Arrow</b>	0
<b>Butterfly</b>	2
<b>Face</b>	0
<b>Star</b>	0
<b>Toonhead</b>	0

<b>Test Class</b>	<b>Heart</b>
-------------------	--------------

Prediction Tally	
Alien	5
Arrow	0
Butterfly	1
Face	0
Star	0
Toonhead	0

Test Class	Tree
Prediction Tally	
Alien	6
Arrow	0
Butterfly	0
Face	0
Star	0
Toonhead	0

Figure 10c: Custom Test Set Results (Optimised Zernike)

Number of Features: 15

Training Directory: './images/train' (Provided)

Testing Directory: './images/custom'

Feature Type: Absolute Value of Zernike Moments

Test Class	Apple
Prediction Tally	
Alien	0
Arrow	0
Butterfly	0
Face	2
Star	1
Toonhead	0

Test Class	Bat
Prediction Tally	
Alien	1

<b>Arrow</b>	0
<b>Butterfly</b>	4
<b>Face</b>	0
<b>Star</b>	0
<b>Toonhead</b>	0

<b>Test Class</b>	<b>Cup</b>
<b>Prediction Tally</b>	
<b>Alien</b>	0
<b>Arrow</b>	1
<b>Butterfly</b>	0
<b>Face</b>	1
<b>Star</b>	2
<b>Toonhead</b>	0

<b>Test Class</b>	<b>Elephant</b>
<b>Prediction Tally</b>	
<b>Alien</b>	2
<b>Arrow</b>	0
<b>Butterfly</b>	0
<b>Face</b>	2
<b>Star</b>	0
<b>Toonhead</b>	0

<b>Test Class</b>	<b>Heart</b>
<b>Prediction Tally</b>	
<b>Alien</b>	6
<b>Arrow</b>	0
<b>Butterfly</b>	0
<b>Face</b>	0
<b>Star</b>	0
<b>Toonhead</b>	0

Test Class	Tree
Prediction Tally	
Alien	3
Arrow	2
Butterfly	1
Face	0
Star	0
Toonhead	0

Figure 10d: Custom Test Set Results (Optimised PCA)

Number of Features: 15

Training Directory: './images/train' (Provided)

Testing Directory: './images/custom'

Feature Type: PCA

Test Class	Apple
Prediction Tally	
Alien	2
Arrow	0
Butterfly	1
Face	0
Star	0
Toonhead	0

Test Class	Bat
Prediction Tally	
Alien	2
Arrow	2
Butterfly	0
Face	0
Star	1
Toonhead	0

Test Class	Cup
Prediction Tally	

<b>Alien</b>	1
<b>Arrow</b>	0
<b>Butterfly</b>	3
<b>Face</b>	0
<b>Star</b>	0
<b>Toonhead</b>	0

<b>Test Class</b>	<b>Elephant</b>
<b>Prediction Tally</b>	
<b>Alien</b>	1
<b>Arrow</b>	0
<b>Butterfly</b>	3
<b>Face</b>	0
<b>Star</b>	0
<b>Toonhead</b>	0

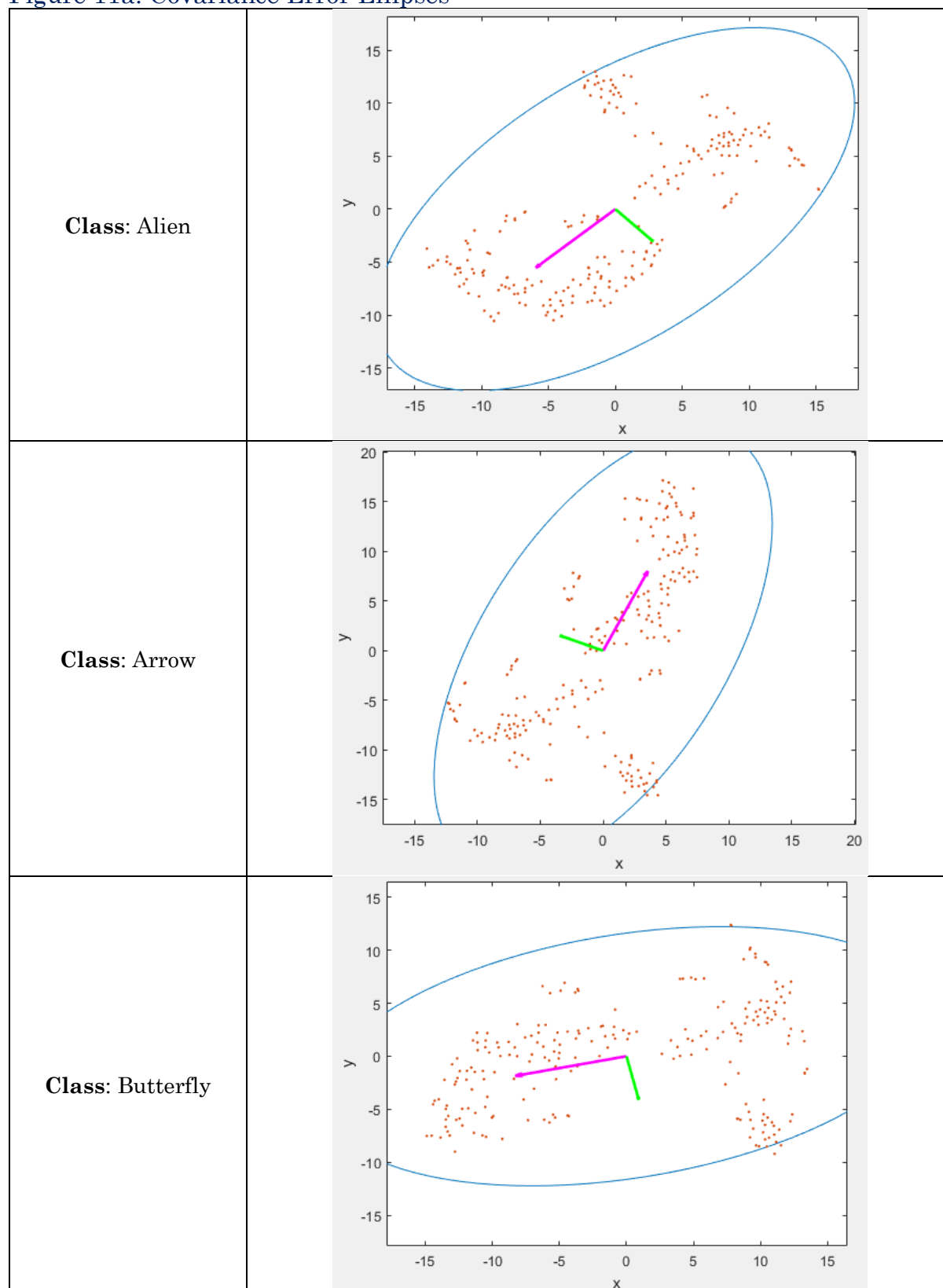
<b>Test Class</b>	<b>Heart</b>
<b>Prediction Tally</b>	
<b>Alien</b>	6
<b>Arrow</b>	0
<b>Butterfly</b>	0
<b>Face</b>	0
<b>Star</b>	0
<b>Toonhead</b>	0

<b>Test Class</b>	<b>Tree</b>
<b>Prediction Tally</b>	
<b>Alien</b>	0
<b>Arrow</b>	3
<b>Butterfly</b>	2
<b>Face</b>	0
<b>Star</b>	1



<b>Toonhead</b>	0
-----------------	---

Figure 11a: Covariance Error Ellipses



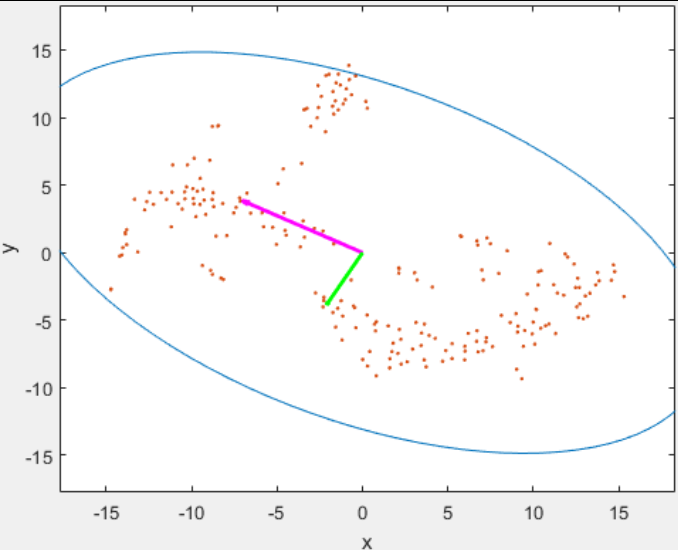
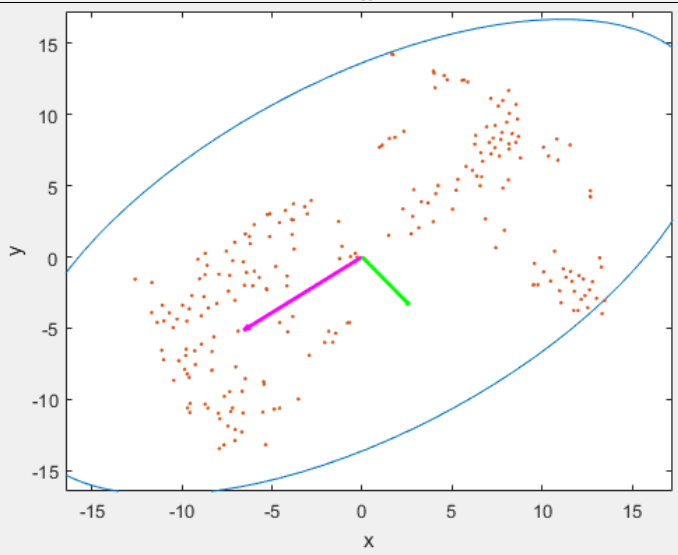
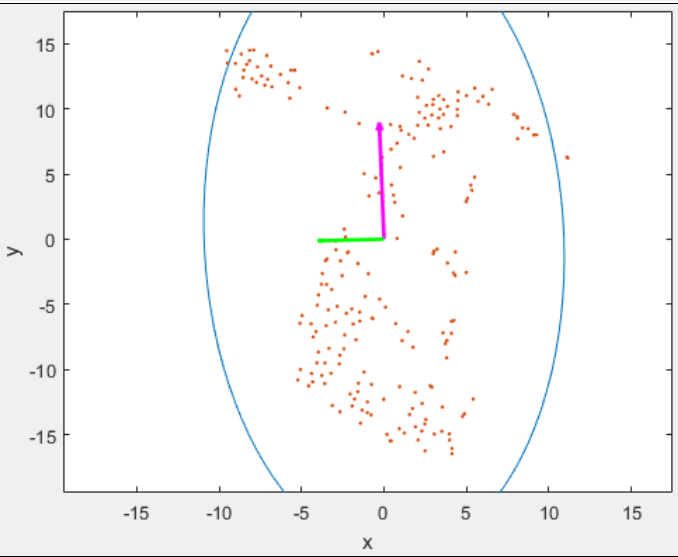
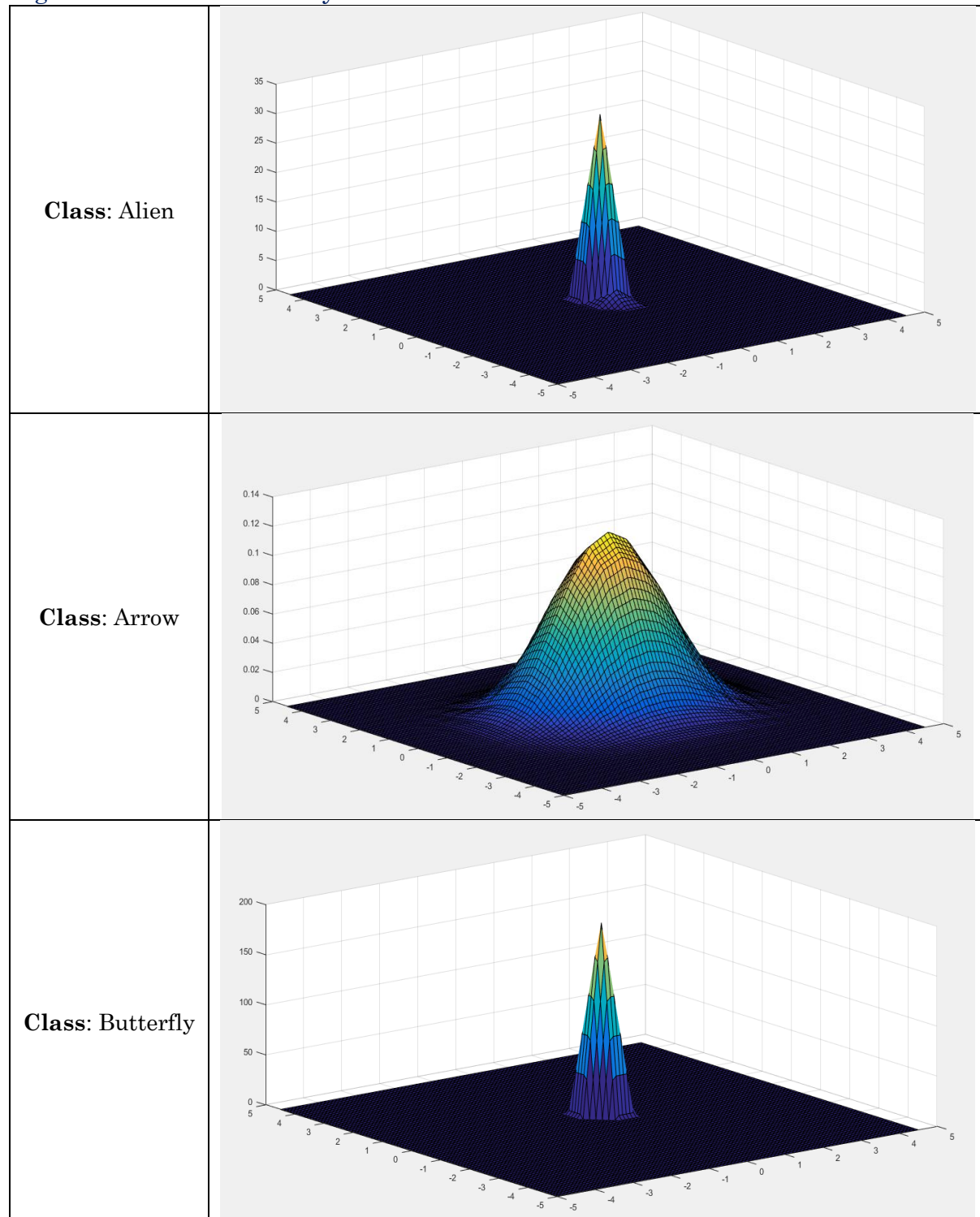
<b>Class: Face</b>	 A scatter plot showing data points for the 'Face' class. The points are orange dots clustered in the upper-left and upper-right regions of the plot. A blue ellipse represents the error ellipse. A magenta line segment connects the center of the ellipse (approximately at x=-2, y=4) to the origin (0,0). A green line segment connects the origin (0,0) to the center of the ellipse (approximately at x=-2, y=4).
<b>Class: Star</b>	 A scatter plot showing data points for the 'Star' class. The points are orange dots clustered in the lower-left and lower-right regions of the plot. A blue ellipse represents the error ellipse. A magenta line segment connects the center of the ellipse (approximately at x=-2, y=-4) to the origin (0,0). A green line segment connects the origin (0,0) to the center of the ellipse (approximately at x=-2, y=-4).
<b>Class: Toonhead</b>	 A scatter plot showing data points for the 'Toonhead' class. The points are orange dots clustered in the upper-left and upper-right regions of the plot. A blue ellipse represents the error ellipse. A magenta line segment connects the center of the ellipse (approximately at x=-2, y=4) to the origin (0,0). A green line segment connects the origin (0,0) to the center of the ellipse (approximately at x=-2, y=4).
<p><b>Code to produce these plots was courtesy of Vincent Spruyt:</b> <a href="http://www.visiondummy.com/2014/04/draw-error-ellipse-representing-covariance-matrix/">http://www.visiondummy.com/2014/04/draw-error-ellipse-representing-covariance-matrix/</a> "How to draw an error ellipse representing the covariance matrix?" (2014)</p>	

Figure 11b: Kernel Density Estimation Plots



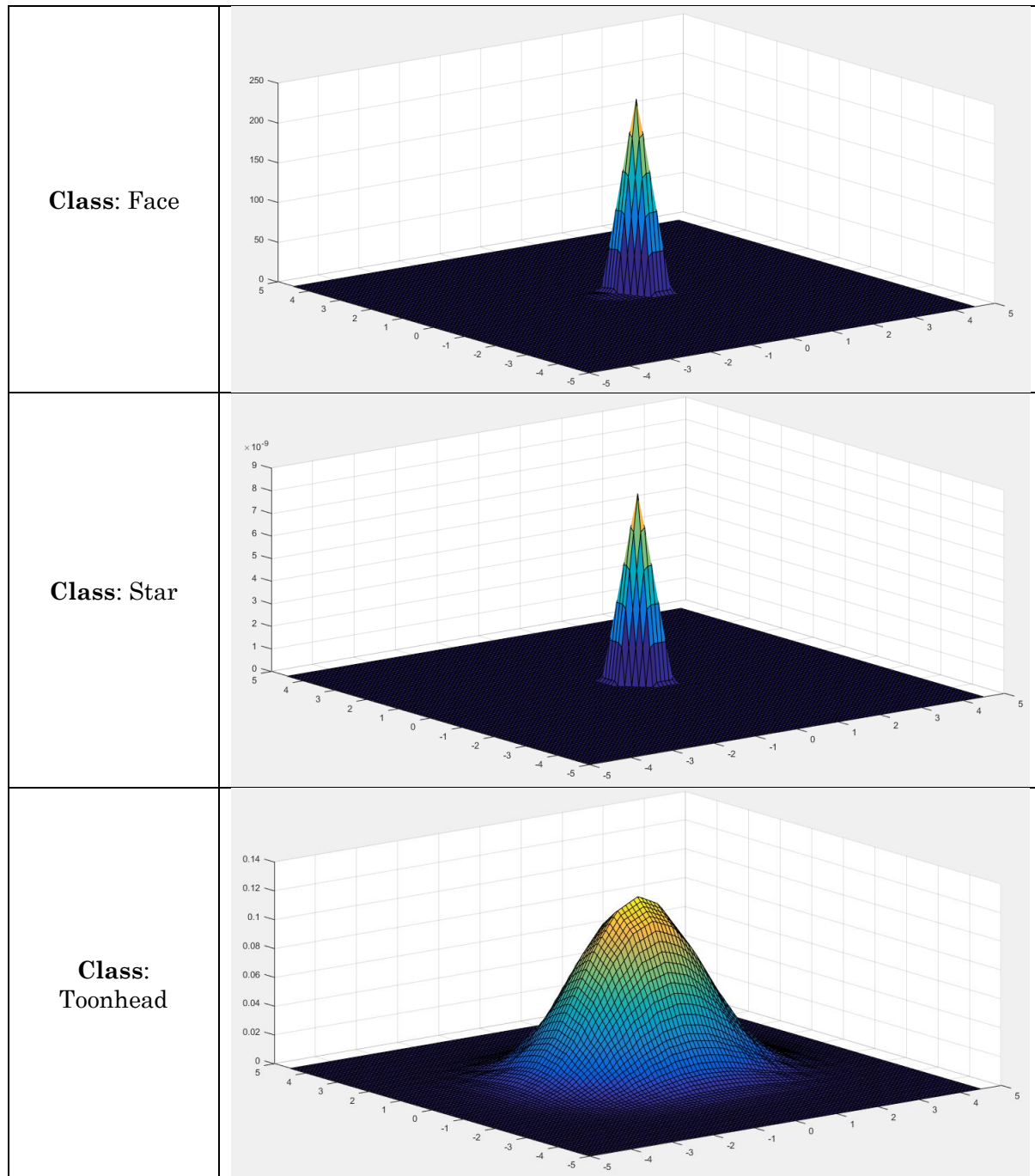


Figure 12a: Code Snippet: Training the Supervised GMM

```

classes = getClasses(imagedir);
totalImages = getNumImages(imagedir);
for idx = 1:length(classes)
    class = classes{idx};
    models(idx).name = class;
    dataMatrix = getDataMatrix(imagedir, class, N, featureType, Shift);
    models(idx).mean = transpose(calcMean(dataMatrix));
    models(idx).cov = ensurePSD(calcCov(dataMatrix));
    models(idx).prior = numImagesForClass(imagedir, class)/totalImages;
end
save('models');

```

Figure 12b: Code Snippet: Classifying an Image (Supervised)

```

load('models');
%Assume all models use the same number of features
N = length(models(1).mean);

```

```

features = getFeatures(imagepath, N, featureType, Shift)';
maxscore = -inf;
%Find out which class has the highest score
for idx = 1:length(models)
    model = models(idx);
    score = log(model.prior) - 0.5*log(det(model.cov)) - 0.5*(features -
model.mean)'*inv(model.cov)*(features - model.mean);
    if score > maxscore
        maxscore = score;
        bestidx = idx;
    end
end
end

```

Figure 12c: Code Snippet: Confusion Matrix

```

conMatrix = zeros(length(getClasses(trainindir)));
classes = getClasses(trainindir);

for idx = 1:length(classes)
    class = classes{idx};
    imagelist = dir(sprintf('%s/%s*.gif', testdir, class));
    for imgidx = 1:length(imagelist)
        imagepath = sprintf('%s/%s', testdir, imagelist(imgidx).name);
        conMatrix(idx, classify(imagepath, 'SUPERVISED')) = conMatrix(idx,
classify(imagepath, 'SUPERVISED')) + 1;
    end
end
end

```

### Calculating the Accuracy of the Classifier:

```

% Calculate the percentage of elements which lie on a matrix diagonal
% For confusion matrices, this is the accuracy of classification
function spread = matrixSpread(cMatrix)
    count = sum(sum(cMatrix));
    spread = 100*trace(cMatrix)/count;
end

```

Figure 12d: Code Snippet: Feature Extraction: Absolute Value of Fourier Coefficients of Chain Code

```

% Get the chain code for the image
c = chainCode(image);
angles = c(3,:)*(2*pi/8);
anglesFFT = fft(angles); %fast fourier transform
% Filter using a 'top hat' filter.
filter = zeros(size(angles));
%Both the positive and negative low frequencies must be kept
%filter(1) is the zero (DC) frequency, so there will be (N*2)-1 ones in total
filter(1:N) = 1;
filter(end-N+2:end) = 0;
filteredFFT = anglesFFT .* filter;
% abs() gets rid of the phase component (the complex argument)
features = abs(filteredFFT(1+Shift:N+Shift));
features = abs(anglesFFT(1+Shift:N+Shift));

```

Figure 12e: Code Snippet: Feature Extraction: Absolute Value of Zernike Moments (Degree 6)

```

% Use the absolute value of the Zernike Moments up to degree 6 as a feature
vector
imageSize = 60;
% The image needs to be centred on a unit square
m = centersquare(image,imageSize);
% abs() gets rid of the phase component (the complex argument)
% Calculate the Zernike moments up to degree 6
featureVector = abs(zernike_mom(m,zernike_bf(imageSize,6)));
features = featureVector / sum(featureVector);
features = features(1:N)';

```

Wolf C., *Matlab code for Zernike moments*. [online] Available at:  
<http://iris.cnrs.fr/christian.wolf/software/zernike/> [Accessed 14 Apr. 2017].

The code from Christian Wolf at the link above allows the calculation of Zernike moments on a binary image and to reconstruct an image from its Zernike moments. The moments are calculated robustly which allows to use them even for the reconstruction with high order moments. All manipulations use a structure holding Zernike basis functions up to a certain maximum order, which are returned by the function `zernike_bf` (the arguments are (square) image size and highest order). The moments are extracted from an image using the function `zernike_mom`.

Figure 12f: Code Snippet: Feature Extraction: Black and White Image Properties

```
b = 0; % number of black pixels
w = 0; % number of white pixels
aspRatio = size(image,2)/size(image,1); % aspect ratio of the image
objects = bwconvhull(image,'objects'); % Generate convex hull image from
binary image
D = bwdist(image); % Distance transform of binary image
bwdistance = sum(calcMean(D));
% count number of black and white pixels respectively
for i = 1:size(image,1)
    for j = 1:size(image,2)
        if image(i,j) == 0
            b = b+1;
        else
            w = w+1;
        end
    end
end

features = [b w b/w w/b bwarea(image) bwarea(objects) aspRatio bwdistance];
features = features(1:N);
```

Figure 12g: Code Snippet: Feature Extraction: PCA

```
% reduce the dimensionality of the data to dimension N
[mapped_data, mapping] = compute_mapping(image, 'PCA', N);
features = mapped_data(1:N);
```

L.J.P. van der Maaten. (c2017). *Matlab Toolbox for Dimensionality Reduction*. [online]  
 Available at: <https://lvdmaaten.github.io/drtoolbox/> [Accessed 14 Apr. 2017].

Figure 12h (i): Code Snippet: k-Means Unsupervised GMM Training

```
for k=1:numImages
    if(idx(k)==1)
        allFeaturesCluster1(countCluster1, :) = allFeatures(k, :);
        countCluster1 = countCluster1 + 1;
    end
    if(idx(k)==2)
        allFeaturesCluster2(countCluster2, :) = allFeatures(k, :);
        countCluster2 = countCluster2 + 1;
    end
    if(idx(k)==3)
        allFeaturesCluster3(countCluster3, :) = allFeatures(k, :);
        countCluster3 = countCluster3 + 1;
    end
    if(idx(k)==4)
        allFeaturesCluster4(countCluster4, :) = allFeatures(k, :);
        countCluster4 = countCluster4 + 1;
    end
    if(idx(k)==5)
        allFeaturesCluster5(countCluster5, :) = allFeatures(k, :);
        countCluster5 = countCluster5 + 1;
    end
    if(idx(k)==6)
        allFeaturesCluster6(countCluster6, :) = allFeatures(k, :);
        countCluster6 = countCluster6 + 1;
    end
end
```



```
end
end
```

Figure 12h (ii): Code Snippet: k-Means Unsupervised GMM Training

```
unsupervised_models(1).kMeansIdx = '1';
dataMatrix = allFeaturesCluster1;
unsupervised_models(1).mean = transpose(calcMean(dataMatrix));
unsupervised_models(1).cov = ensurePSD(calcCov(dataMatrix));
unsupervised_models(1).prior = numClus1/numImages;
```

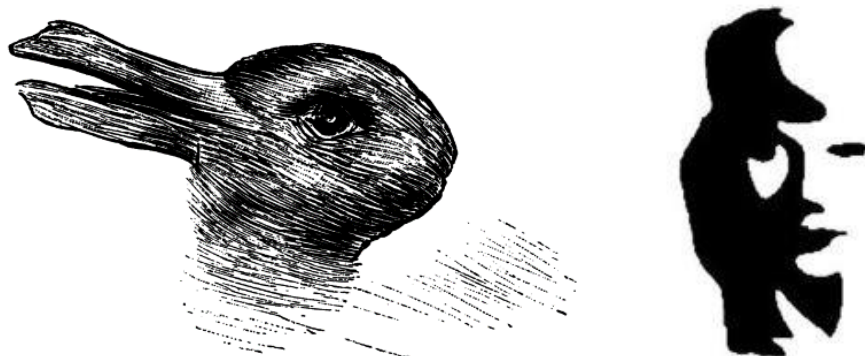
Figure 12h (iii): Code Snippet: k-Means Unsupervised GMM Training

```
min_klDivVal = inf;
closestClassIdx = 0;
for cluster = 1:6
    mu_0 = unsupervised_models(1).mean;
    mu_1 = models(cluster).mean;
    cov_0 = unsupervised_models(1).cov;
    cov_1 = models(cluster).cov;
    klDiv = klDivergence(mu_1, mu_0, cov_1, cov_0);
    if(abs(klDiv) < min_klDivVal)
        min_klDivVal = klDiv;
        closestClassIdx = cluster;
    end
end
unsupervised_models(1).closestLabel = closestClassIdx;
unsupervised_models(1).closestClass = classes(closestClassIdx);
```

Figure 12h (iv): Code Snippet: k-Means Unsupervised GMM Training

```
load('unsupervised_models');
%Assume all models use the same number of features
N = length(unsupervised_models(1).mean);
features = getFeatures(imagepath, N, featureType, Shift)';
maxscore = -inf;
%Find out which class has the highest score
for idx = 1:length(unsupervised_models)
    model = unsupervised_models(idx);
    score = log(model.prior) - 0.5*log(det(model.cov)) - 0.5*(features -
model.mean)'*inv(model.cov)*(features - model.mean);
    if score > maxscore
        maxscore = score;
        bestidx = model.closestLabel;
    end
end
```

Figure 13a: “Difficult Images”



**Left:** A binary image which appears to be of the class *Duck* at one angle, and class *Rabbit* at another.  
**Right:** A binary image that appears of the class *Musician* if examining just the black silhouette, but class *Face* if examining the whole image.