NOTE: I believe the instructions mistakenly forget to mention a script_2d and script_2e. I have assumed that I am supposed to have those files, as it's the only logical way I could interpret the instructions. However, if for some reason I'm not supposed to have them, then probably my output for 2a, 2b, and 2c would not be correct, so just a heads-up on that.

Question 1:

a) First I do the derivation for **Model 2**. When S = S1 = S2, we need to solve for the combined S. However finding the likelihood means finding the combined likelihood of both classes. We begin by writing this likelihood formulation down:

$$\prod_{i=1}^{2}\prod_{t=1}^{N} N(x^t|\mu_i, S)^{r_i^t}$$

Where r_i is the class selector for class i and N is the number of data points in the entire data set (do not get confused with the N inside the product, as this other N represents the normal distribution). Taking the log of this yields:

$$\sum_{t=1}^{N} r_1^t \log N(x^t|\mu_1, S) + \sum_{t=1}^{N} r_2^t \log N(x^t|\mu_2, S)$$

We must solve for all three parameters, so we start with mu_1. We take the derivative with respect to mu_1 and set the resulting expression to 0. Since the second summation disappears, we only need to make a small modification to the derivation for the normal distribution in the textbook:

$$\sum_{t=1}^{N} r_1^t (x^t - \mu_1) = 0$$

Removing mu_1 from the summation yields:

$$|C_1|\mu_1 = \sum_{t=1}^{N} r_1^t x^t$$

And finally, solving for mu_1 gives us:

$$\mu_1 = \frac{\sum_{t=1}^{N} r_1^t x^t}{|C_1|}$$

The result is the class-mean of class 1. Without loss of generality, we can say that mu_2 will be the class-mean of class 2. We now proceed to solve for the final parameter, S. We take the derivative of the log-likelihood with respect to S and set it to 0. However, we know how to take the derivative of the log normal with respect to S from the textbook, so again the result is only slightly modified:

$$\frac{1}{2}\sum_{t=1}^{N} r_1^t (\mathrm{S} - (x^t - \mu_1)(x^t - \mu_1)^T) + \frac{1}{2}\sum_{t=1}^{N} r_2^t (\mathrm{S} - (x^t - \mu_2)(x^t - \mu_2)^T) = 0$$

Dividing out the ½ term and pulling S out of the expression, we get:

$$|C_1|S - \sum_{t=1}^{N} r_1^t (x^t - \mu_1)(x^t - \mu_1)^T) + |C_2|S - \sum_{t=1}^{N} r_2^t (x^t - \mu_2)(x^t - \mu_2)^T) = 0$$

Combining the S terms (since |C1| + |C2| = N) and solving for S, we get.

$$S = \frac{\sum_{t=1}^{N} r_1^t (x^t - \mu_1)(x^t - \mu_1)^T + \sum_{t=1}^{N} r_2^t (x^t - \mu_2)(x^t - \mu_2)^T}{N}$$

But this is exactly:

$$S = P(C_1)S_1 + P(C_2)S_2$$

So we've derived the desired equation.

Here I give the derivation for **Model 3**.

Let N be the number of items within class i. Then the likelihood of C_i is given by:

$$\prod_{t=1}^{N} P(x^t|C_i)$$

Taking the log of this gives us:

$$\sum_{t=1}^{N} \log P(x^t|C_i)$$

Since we're dealing with a normal distribution, we expand as follows:

$$\sum_{t=1}^{N} \log\left(\frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_i|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(x^t - \mu_i)^T \Sigma_i^{-1}(x^t - \mu_i)\right]\right)$$

And simplifying by expanding the log gives us:

$$\sum_{t=1}^{N} -\frac{\mathrm{d}}{2}\log 2\pi - \frac{1}{2}\log|\Sigma_i| - \frac{1}{2}(x^t - \mu_i)^T \Sigma_i^{-1}(x^t - \mu_i)$$

Since we want the covariance to have only one value along the diagonal, we use the following two facts to simplify:

$$\det(\Sigma_i) = \alpha_i^d \; and \; \Sigma_i^{-1} = \frac{1}{\alpha_i} I$$

Plugging this in yields:

$$\sum_{t=1}^{N} -\frac{\mathrm{d}}{2}\log 2\pi - \frac{d}{2}\log \alpha_i - \frac{\left||x^t - \mu_i|\right|^2}{2\alpha_i}$$

Now taking the derivative with respect to alpha_i gives us (note that there should be a two in both denominators, but I've multiplied it out already for simplicity):

$$\sum_{t=1}^{N} -\frac{d}{\alpha_i} + \frac{\left||x^t - \mu_i|\right|^2}{\alpha_i^2} = 0$$

Reaching a common denominator:

$$\sum_{t=1}^{N} \frac{\left||x^t - \mu_i|\right|^2 - d\alpha_i}{\alpha_i^2} = 0$$

Multiplying out by alpha_i:

$$\sum_{t=1}^{N} \left||x^t - \mu_i|\right|^2 - d\alpha_i = 0$$

Removing alpha_i from the summation:

$$\sum_{t=1}^{N} \left||x^t - \mu_i|\right|^2 = dN\alpha_i$$

Finally, solving for alpha_i:

$$\alpha_i = \frac{\sum_{t=1}^{N} \left||x^t - \mu_i|\right|^2}{dN}$$

Now we must solve for mu_i. We take the derivate of the log-likelihood with respect to mu_i and set it to 0, which is equivalent to solving the following simplified equation:

$$\frac{dL}{d\mu_i} \sum_{t=1}^{N} (x^t - \mu_i)^T \Sigma_i^{-1} (x^t - \mu_i) = 0$$

Remembering what we know about know about the covariance, we can further simplify this equation to:

$$\frac{dL}{d\mu_i} \sum_{t=1}^{N} (x^t - \mu_i)^T (x^t - \mu_i) = 0$$

Multiplying out the expression in the summation gives:

$$\frac{dL}{d\mu_i} \sum_{t=1}^{N} (x^t)^T x^t - (x^t)^T \mu_i - \mu_i^T x^t + \mu_i^T \mu_i = 0$$

And computing the derivative gives:

$$\sum_{t=1}^{N} -x^t - x^t + 2\mu_i = 0$$

Which ultimately yields:

$$\mu_i = \frac{\sum_{t=1}^{N} x^t}{N}$$

So mu_i is the class-mean. Therefore, to find S1 and S2, we compute alpha1 and alpha2 for the two classes, and then just multiply by the identity matrix.

For Model 3, we are told that we must derive the classifier. We aren't told anywhere to write the formula or to actually derive it, but I think it couldn't hurt to include it:

$$g_i(x) = -\frac{d}{2}\log \alpha_i - \frac{1}{2\alpha_i}(x - m_i)^T(x - m_i) + \log P(C_i)$$

Deriving this is just a matter of plugging in the aforementioned facts about the determinant of the covariance and the inverse of the covariance directly into equation 5.19.

b) Below is the output of MultiGaussian for Model = 1, 2, and 3 respectively:

MODEL 1:

```
Model 1:
Classification error:
    0.1500

Prior of class 1: 0.3
Prior of class 2: 0.7
Mean of class 1:
    1.0554    2.5181    3.2967   -1.8927   -1.3918    4.0635   -4.3540   -5.8705

Mean of class 1:
    3.8052    5.3740    5.7333    1.1596    1.1777    6.8000   -2.0286   -2.5044

Covariance of class 1:
    0.9729    0.7135    0.4570    0.8938    0.3096    0.1975    0.7362    1.6629
    0.7135    3.0658    2.5982    0.3878    1.2994    0.1442    0.9168    4.9388
    0.4570    2.5982    6.6612    0.9084    1.6397    0.8148    0.0779    5.4168
    0.8938    0.3878    0.9084    5.0754    0.0963    1.0611    2.3978    4.5946
    0.3096    1.2994    1.6397    0.0963    2.3973   -0.0191    0.2175    2.5378
    0.1975    0.1442    0.8148    1.0611   -0.0191    1.0412   -0.0531    1.8604
    0.7362    0.9168    0.0779    2.3978    0.2175   -0.0531    6.5154    3.8609
    1.6629    4.9388    5.4168    4.5946    2.5378    1.8604    3.8609   17.0931

Covariance of class 2:
    1.3486    0.8658    0.1559    0.5847    0.9473    0.2543    0.3705    0.9435
    0.8658    2.8161   -0.1819    0.2152    0.7141    0.6797   -0.2383    2.4194
    0.1559   -0.1819    6.6734    1.7716    1.0318    0.6526    1.6795    3.8992
    0.5847    0.2152    1.7716    3.5433    0.3570    1.3665    2.1207    3.0899
    0.9473    0.7141    1.0318    0.3570    2.7517    0.1225    1.4342    2.4651
    0.2543    0.6797    0.6526    1.3665    0.1225    1.8107    0.3445    1.5113
    0.3705   -0.2383    1.6795    2.1207    1.4342    0.3445    7.1134    2.6246
    0.9435    2.4194    3.8992    3.0899    2.4651    1.5113    2.6246   13.9151
```

MODEL 2:

```
Model 2:
Classification error:
    0.1200

Prior of class 1: 0.3
Prior of class 2: 0.7
Mean of class 1:
    1.0554    2.5181    3.2967   -1.8927   -1.3918    4.0635   -4.3540   -5.8705

Mean of class 1:
    3.8052    5.3740    5.7333    1.1596    1.1777    6.8000   -2.0286   -2.5044

Covariance of class 1:
    1.2359    0.8201    0.2462    0.6774    0.7560    0.2373    0.4802    1.1593
    0.8201    2.8910    0.6521    0.2670    0.8897    0.5190    0.1082    3.1752
    0.2462    0.6521    6.6697    1.5126    1.2142    0.7013    1.1990    4.3545
    0.6774    0.2670    1.5126    4.0030    0.2788    1.2749    2.2039    3.5413
    0.7560    0.8897    1.2142    0.2788    2.6454    0.0800    1.0692    2.4869
    0.2373    0.5190    0.7013    1.2749    0.0800    1.5799    0.2252    1.6160
    0.4802    0.1082    1.1990    2.2039    1.0692    0.2252    6.9340    2.9955
    1.1593    3.1752    4.3545    3.5413    2.4869    1.6160    2.9955   14.8685

Covariance of class 2:
    1.2359    0.8201    0.2462    0.6774    0.7560    0.2373    0.4802    1.1593
    0.8201    2.8910    0.6521    0.2670    0.8897    0.5190    0.1082    3.1752
    0.2462    0.6521    6.6697    1.5126    1.2142    0.7013    1.1990    4.3545
    0.6774    0.2670    1.5126    4.0030    0.2788    1.2749    2.2039    3.5413
    0.7560    0.8897    1.2142    0.2788    2.6454    0.0800    1.0692    2.4869
    0.2373    0.5190    0.7013    1.2749    0.0800    1.5799    0.2252    1.6160
    0.4802    0.1082    1.1990    2.2039    1.0692    0.2252    6.9340    2.9955
    1.1593    3.1752    4.3545    3.5413    2.4869    1.6160    2.9955   14.8685
```

MODEL 3:

```
Model 3:
Classification error:
    0.2300

Prior of class 1: 0.31429
Prior of class 2: 0.68571
Mean of class 1:
    1.0554    2.5181    3.2967   -1.8927   -1.3918    4.0635   -4.3540   -5.8705

Mean of class 1:
    3.8052    5.3740    5.7333    1.1596    1.1777    6.8000   -2.0286   -2.5044

Alpha1:
    5.1744

Alpha2:
    4.9252
```

We can see that the learned values for alpha1 and alpha2 are 5.1744 and 4.9252 respectively.

c) Below are the classification error rates for model 1, 2, and 3 respectively:

```
Model 1:
Classification error:
    0.1500

Model 2:
Classification error:
    0.1200

Model 3:
Classification error:
    0.2300
```

We notice from these numbers that the classification rate of Model 2 is actually better than that of Model 1, even though Model 2 is a simpler model. This is likely due to the limited amount of data we're given; when the data is limited, pooling the data may actually be more effective than trying to classify the data as if both sets have separate covariance matrices. In my own testing, I had even better results when I set S = cov(data) – that is, I took the covariance of the entire data set instead of weighting the covariances by their priors, and I had even better results than Model 2. Model 3 performs the worst, but this is to be expected since both covariances depend on only one parameter. What's surprising is just how well it performs; even with the assumption that the covariances are diagonal matrices with only one variable on the entire diagonal, the classifier is still able to achieve 77% classification accuracy.

Question 2

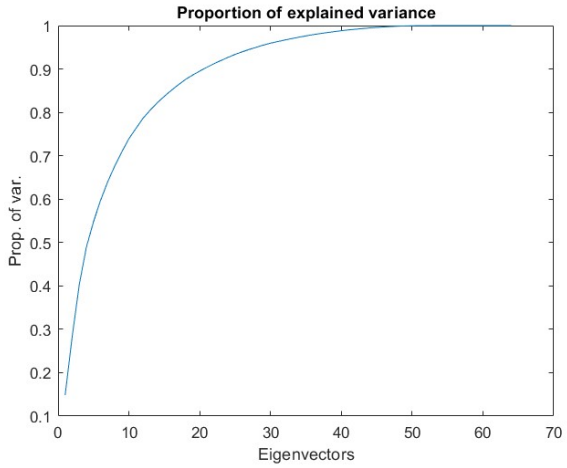
a) Below is the output of script_2a:

```
KNN Classification Errors
k = 1: 0.053872
k = 3: 0.040404
k = 5: 0.043771
k = 7: 0.053872
```

b) Below is the output of script_2b:

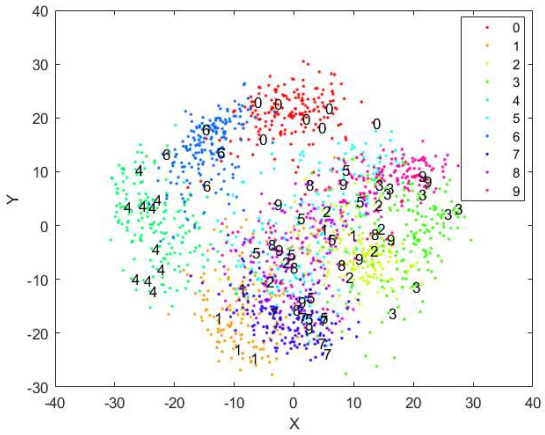
```
Number of dimensions required to explain 90% of variance: 21
KNN Classification Error for Projected Data
k = 1: 0.043771
k = 3: 0.040404
k = 5: 0.043771
k = 7: 0.040404
```

Note that the number of dimensions required to explain 90% of the variance corresponds to the K value we are asked to find. The script also generates this plot of proportion of variance:



We can see that the proportion of variance hits 90% at 21 principle components.

c) Below is the graph produced by script_2c:



To produce this graph, I first pooled all the data for the optdigits set (training and test data). I then ran myPCA(), projected the data onto the first two principle component, and used gscatter to display it. I took roughly 1/25 of the points and displayed text for them. I decided not to color the actual text because the data points are already colored; coloring the text would make it hard to read. Overall, even with only two dimensions we can observe some pretty good clustering here.

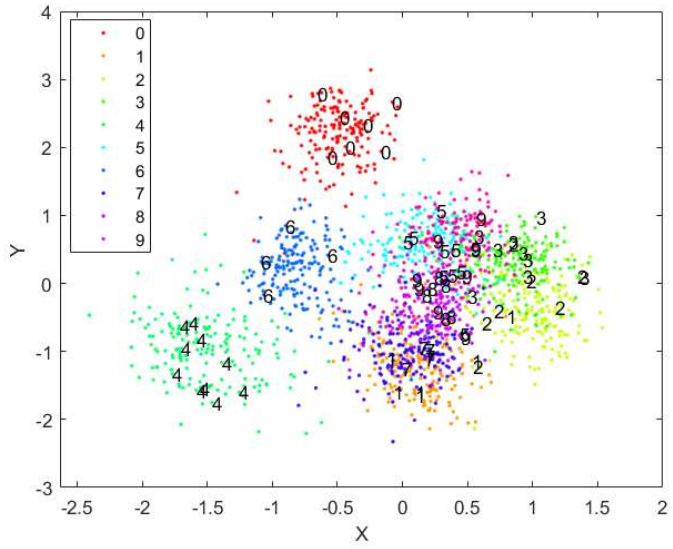
d) Below is the output of script_2d:

```
KNN error rates on data projected to 2 dimensions:
k = 1: 0.40067
k = 3: 0.41077
```

```
k = 5: 0.38047
k = 7: 0.37374
KNN error rates on data projected to 4 dimensions:
k = 1: 0.15152
k = 3: 0.16498
k = 5: 0.14815
k = 7: 0.13805
KNN error rates on data projected to 9 dimensions:
k = 1: 0.094276
k = 3: 0.084175
k = 5: 0.087542
k = 7: 0.077441
```

We can see LDA seems to perform very poorly with only 2 dimensions, and improves substantially as we add back dimensions. The reason for this can be seen in the graph I've provided for part E; there appears to be a lot of overlap between most of the classes in low dimensions. Note that for this problem, I projected both the training and test optdigit sets, and then ran KNN on both of them respectively. Though the instructions say to only use the training set, this must be a mistake, because you can't run KNN with one set being both the training and test set; it'll lead to nonsensical results (for example, 0 error when k = 1).

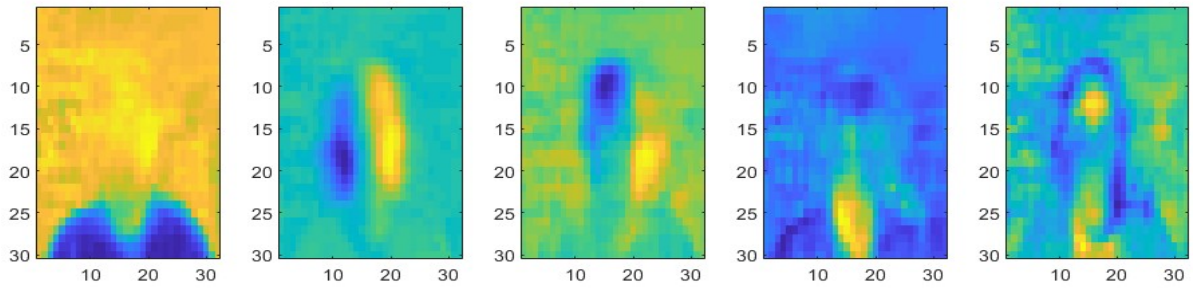
e) Below is the graph produced by script_2e:



To produce this graph, I first pooled all the data for the optdigits set (training and test data). I then ran myLDA(), projected the data onto the first two principle component, and used gscatter to display it. I took roughly 1/25 of the points and displayed text for them. I decided not to color the actual text because the data points are already colored; coloring the text would make it hard to read. We can see that the success of LDA vs PCA is not so clear-cut. While 0, 6, and 4 are very nicely separated with LDA, there seems to be lots of class overlap for the other values.

Question 3:

a) Below is the graph generated by script_3a:

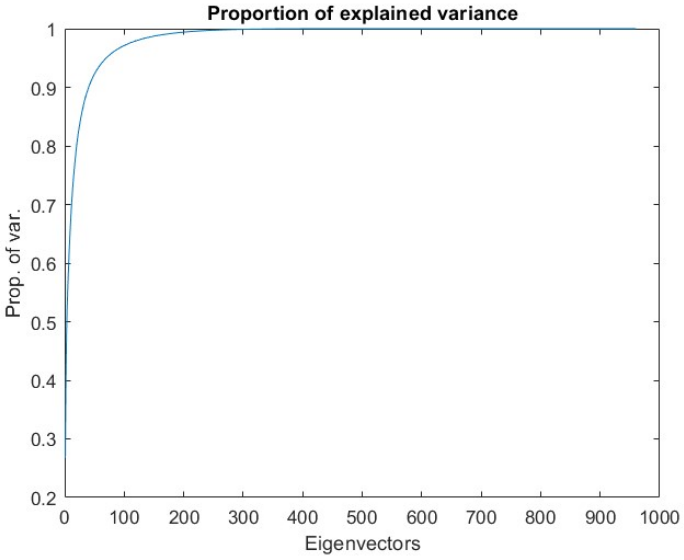


These eigenfaces are labeled in order from most significant to least significant (left to right). It is not entirely clear that these eigenfaces correspond to truly meaningful features (eyes, mouth, etc.), but we can still see resemblance to something human, which is promising.

b) Below is the output of script_3b:

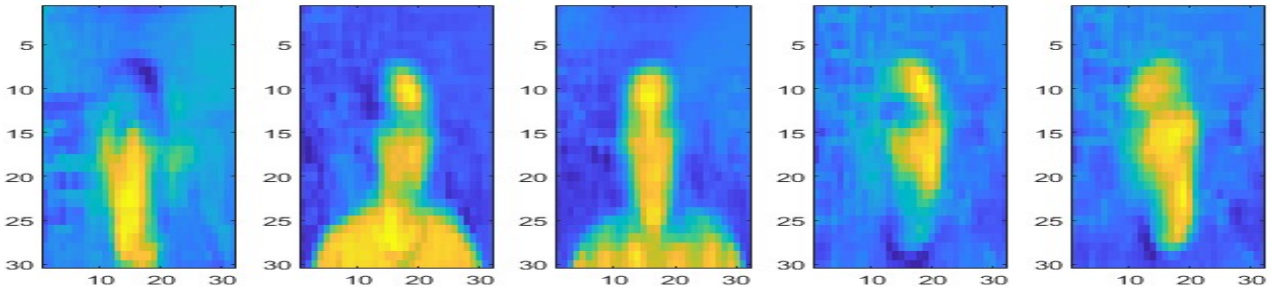
```
Num PCs that explain 90% of the variance: 41
KNN Classification Error for Projected Data
k = 1: 0.10484
k = 3: 0.24194
k = 5: 0.39516
k = 7: 0.39516
```

Note that the number of dimensions required to explain 90% of the variance corresponds to the K value we are asked to find. The script also generates this plot of proportion of variance:

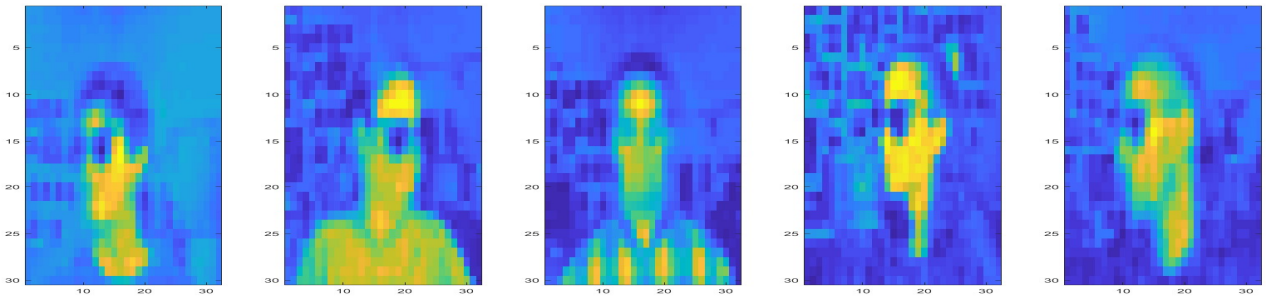


We can see that the proportion of variance hits 90% at 41 eigenvectors.

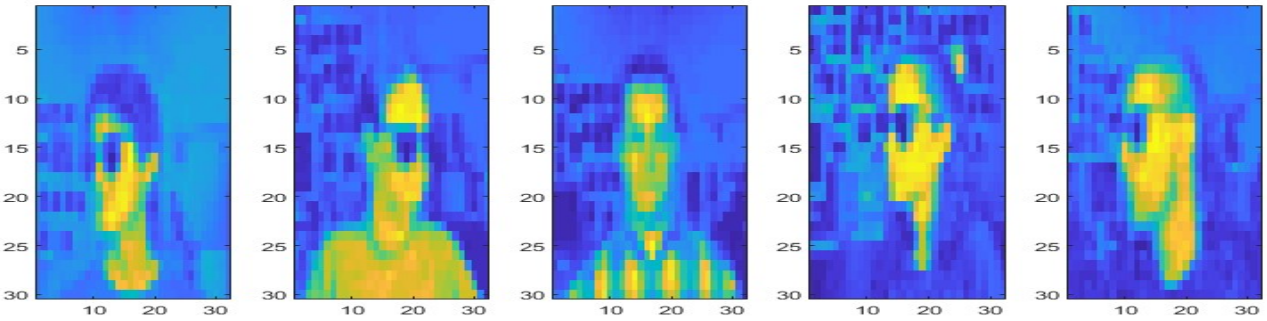
c) Below are the graphs generated by script_3c. These graphs correspond to projections to 10, 50, and 100 dimensions respectively. We can see a significant (but diminishing) quality increase as the dimensionality increases.



These first five faces correspond to a projection to the first 10 principle components.



These next five faces correspond to a projection to the first 50 principle components. We can see a noticeable increase in the sharpness of the image.



These last five faces correspond to a projection to the first 100 principle components. The quality improves yet again, but not as drastically as jumping from 10 to 50 principle components.