CS 215: Assignment 2 Report

Poorna Teja and Karthikeya

October 2021

Contents

1 Sampling within a Euclidean Plane

1.1 Algorithm to generate uniformly distributed random points(UDRPs)

1.1.1 In an ellipse

Consider an ellipse in a polar coordinate system(r, θ).A point (r, θ)'s occurrence in the UDRP generator has a probability density proportional to r which can be intuitively understood from the fact that the infinitesimally small area around it is $\sqrt{\frac{b}{a}} r dr d\theta$.The random point generator can be divided into two independent parts,

- The θ generator which is $U(0, 2\pi)$.
- The r generator has the range $(0, a)$ and probability density proportional to r.

The random point is calculated from the output of the θ generator(θ_1), the r generator(r_1) as $(r_1 cos(\theta_1), \frac{b}{a} r_1 sin(\theta_1))$.The task is to construct the r generator.Consider two distributions R_1 and R_2 where R_1 has a range [0,1] and it's probability density is proportional to x, R_2 is $U(0, 1)$.Let $y = f(x)$ be an unknown function from $[0, 1]$ to $[0, 1]$ such that applying the transformation of f on R_1 results in R_2.Let's assume that f is monotonic and use the formula for transformation of RV's $R_1(x) = R_2(f(x))\frac{d}{dx}(f(x))$.
$f(x) = x^2$ is a valid transformation from R1 to R2 so we can construct the r generator using $\mathbf{a} * \mathbf{sqrt}(\mathbf{U}(\mathbf{0}, \mathbf{1}))$.

1.1.2 In a triangle

Consider a rectangle encompassing the triangle in consideration.The idea behind generating UDRPs in a rectangle is elementary, consider the rectangle (0,0),(a,0),(0,b),(a,b) the UDRP generator is simply (U(0,a),U(0,b)).We can construct the UDRP generator of the triangle using the generator of the rectangle as

Algorithm 1: UDRP$generator_{trl}$

```
point p = UDRPgenerator_rect()
if p is outside triangle then
    return UDRPgenerator_trl()
return p
```

All the points have the same probability density in UDRP$generator_{rect}()$ and $\frac{A_{trl}}{A_{rect}}$ is the probability for that point to be inside the triangle, clearly the UDRP generator constructed for the triangle has the same probability density for all the points inside the triangle proving it's correctness.

1.2 Histograms of the sample points

1.2.1 Ellipse

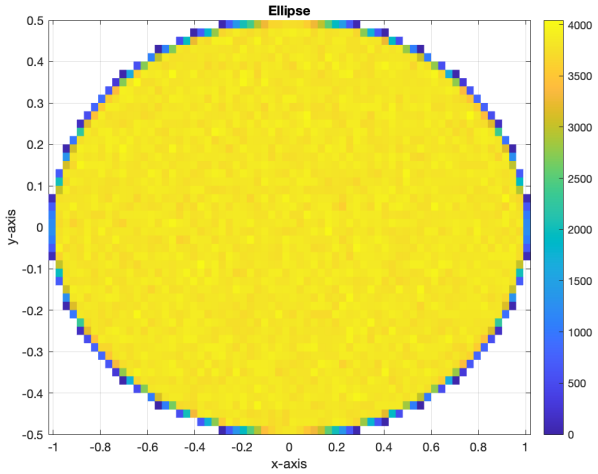

Figure 1: Histogram of the sample points

1.2.2 Triangle

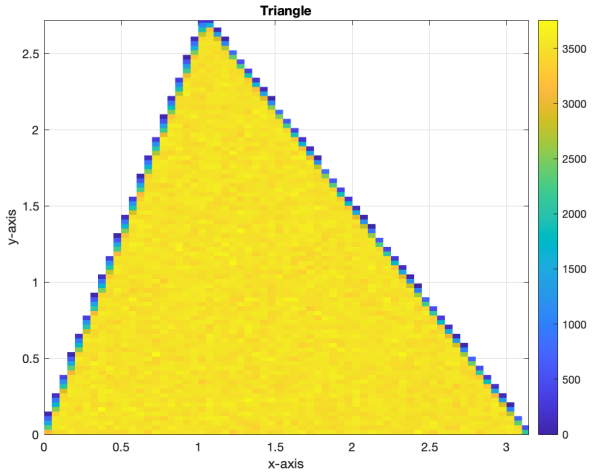

Figure 2: Histogram of the sample points

2 Multivariate Gaussian

2.1 Method to generate sample points

We know that the Co-variance matrix C is symmetric and positive definite hence it has a unique[1] Cholesky decomposition.From the given co-variance matrix C we can calculate the matrix A where $C = AA^T$.The multivariate Gaussian distribution has the probability density function $Y = AW + \mu$.To generate a random point from this distribution we need two random outcomes from a uni-variate Gaussian distribution r_1, r_2 using which we can get the point as the elements of the column vector Y where $Y = AW_1 + \mu$, $W_1 = [r_1, r_2]^T$.

2.2 Error measure of mean and co-variance

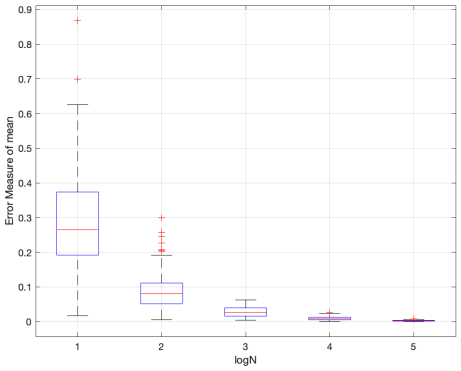

Figure 3: Error measure of mean vs $log(N)$

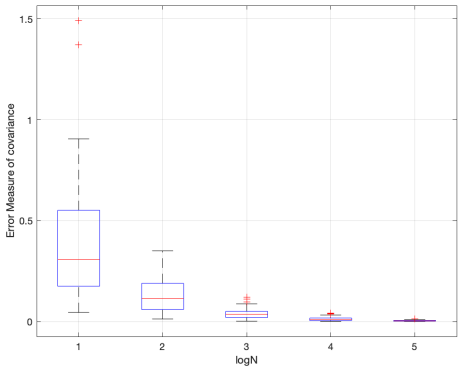

Figure 4: Error measure of co-variance vs $log(N)$

The error measure gets closer to 0 as the size (N) of the data is increased in both the cases, which is the expected behaviour.

2.3 Principal modes of variation

The principle modes of variation are depicted in the below scatter plots, note that the two lines are not perpendicular as the scale is not 1:1.

[1] Cholesky decomposition's wikipedia page

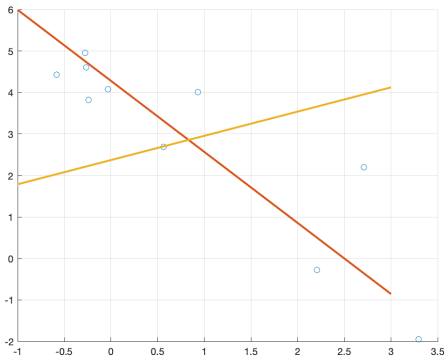

Figure 5: $N = 10$

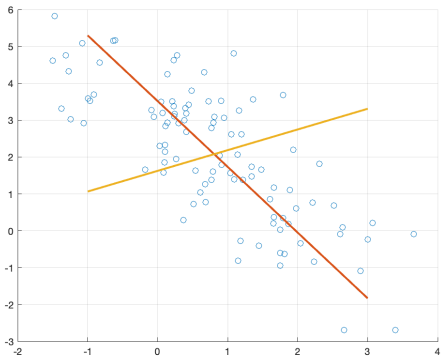

Figure 6: $N = 100$

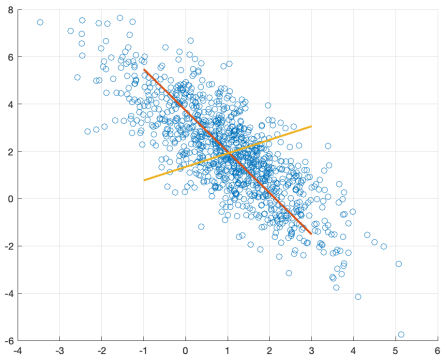

Figure 7: $N = 1000$

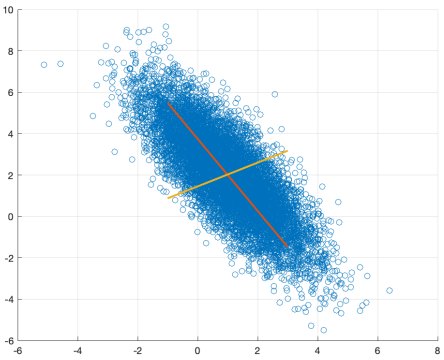

Figure 8: $N = 100000$

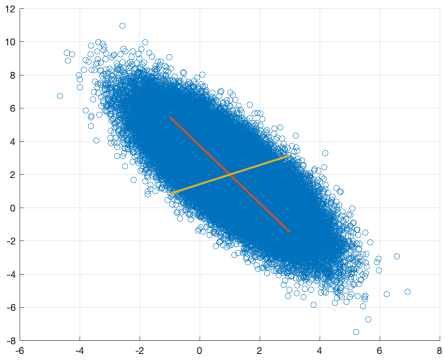

Figure 9: $N = 100000$

3 PCA and Hyperplane Fitting

3.1 Running the code

3.2 Approximate linear relationship between X and Y

Linear relationship between X and Y can be found by linear dimensionality reduction.

$$R = \begin{bmatrix} X & Y \end{bmatrix}^T$$

$$\mu = \begin{bmatrix} X\_mean & Y\_mean \end{bmatrix}^T$$

$$R = R - \mu$$

$$C = Cov(R)$$

Let V be the eigen vector of C with highest eigen value.(Direction of maximal variance)

$$V = \begin{bmatrix} v_x & v_y \end{bmatrix}^T \tag{1}$$

Now, the approximate linear relation between X and Y is given by

$$Y = \frac{(X - X\_mean)v_y}{v_x} + Y\_mean$$

3.3 Plots

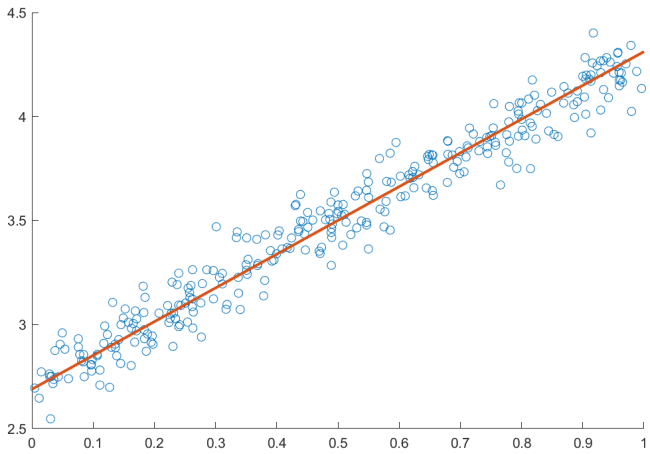

Figure 10: Set1

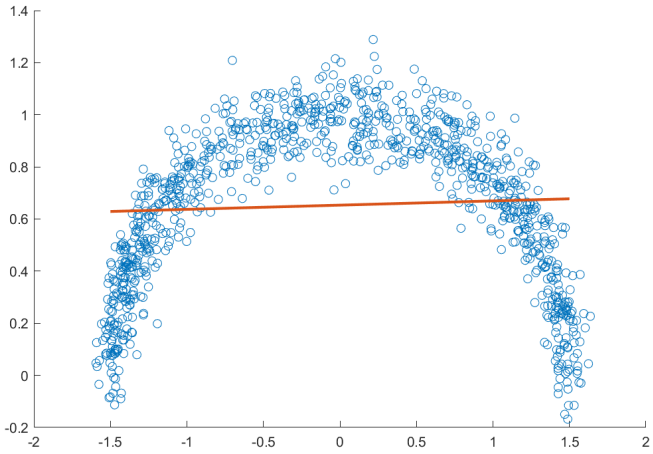

Figure 11: Set2

3.4 Comparision

From the figures, it's clear that quality of approximation in figure 1 is much better when compared to figure2. The error we get by fitting all the scattered points on the line will be less in the first case because points are very closer to our line. Due to the non-linear nature of the 2nd scattered plot, many points are very far away from the line.

4 Principal Component Analysis

4.1 Plots of eigen values

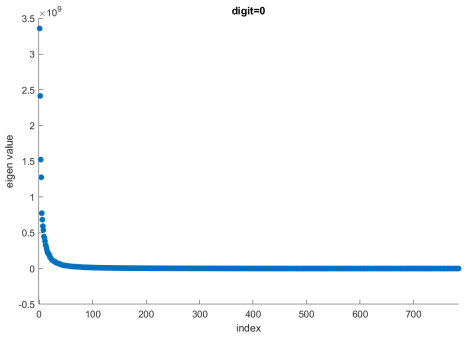

Figure 12: digit 0

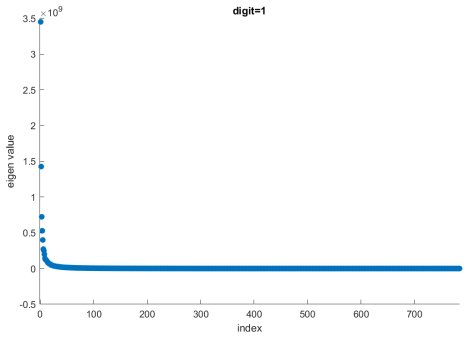

Figure 13: digit 1

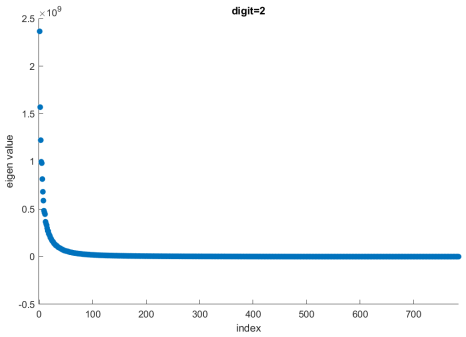

Figure 14: digit 2

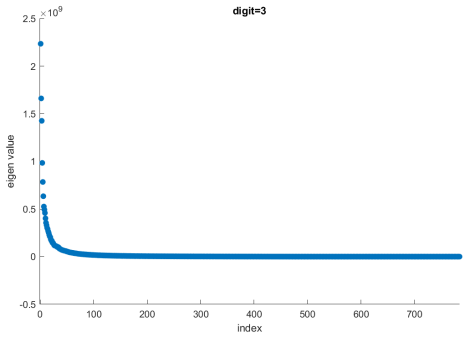

Figure 15: digit 3

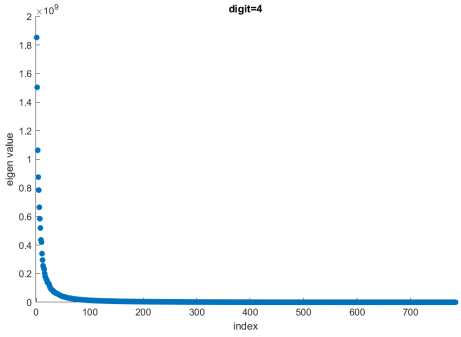

Figure 16: digit 4

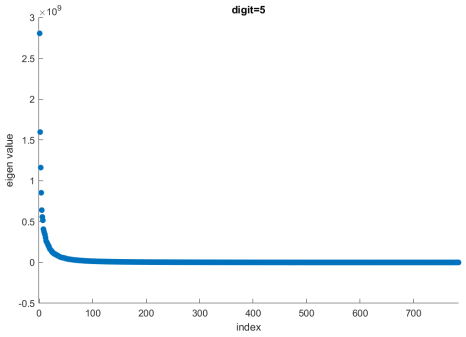

Figure 17: digit 5

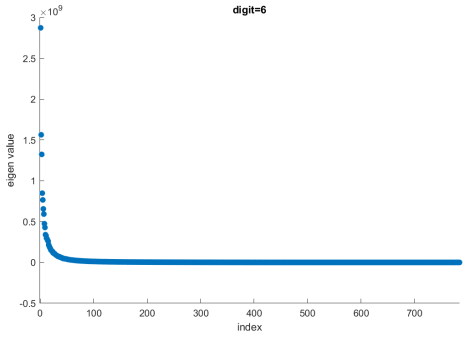

Figure 18: digit 6

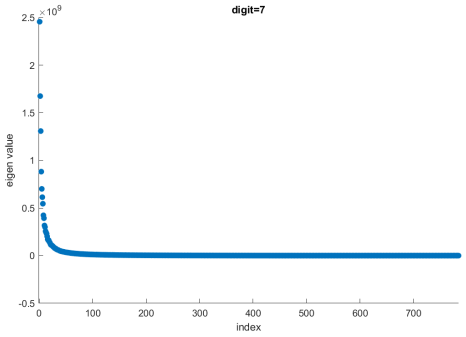

Figure 19: digit 7

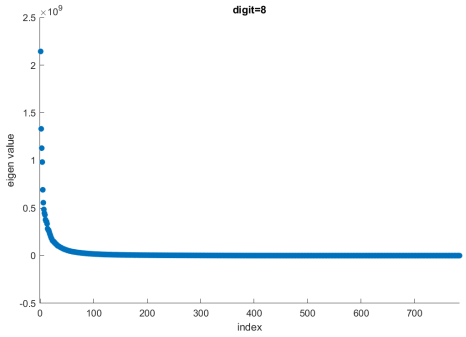

Figure 20: digit 8

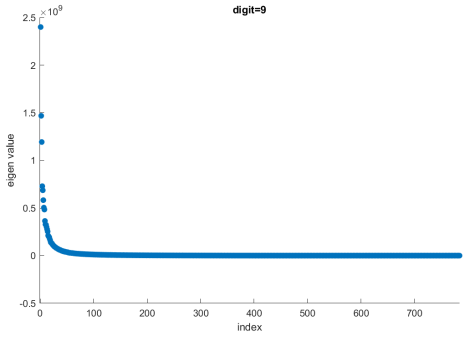

Figure 21: digit 9

4.2 explantaion

Number of eigen values which are greater that 10% of the total variance for all the digits are $\begin{bmatrix}2 & 2 & 1 & 1 & 1 & 1 & 2 & 1 & 1\end{bmatrix}$ (where value in i^{th} index is corresponding to i^{th} digit) Number of eigen values which are greater than 1% of the total variance for all the digits are $\begin{bmatrix}19 & 13 & 21 & 20 & 20 & 19 & 19 & 18 & 21 & 19\end{bmatrix}$ Clearly number of significant modes of variation is far less than 784 for all the digits. This is because acquired data is corrupted with errors. Like measurement errors. Such errors makes the signal representation seem to of a dimension higher than intrinsic dimension

4.3 Principle mode of variation of the digits around their mean

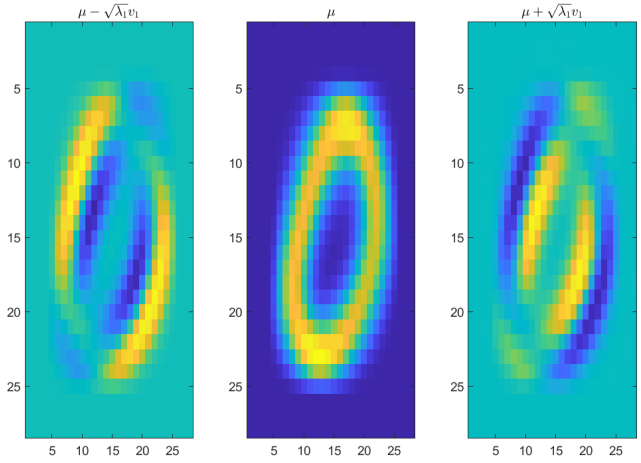

Figure 22: digit 0

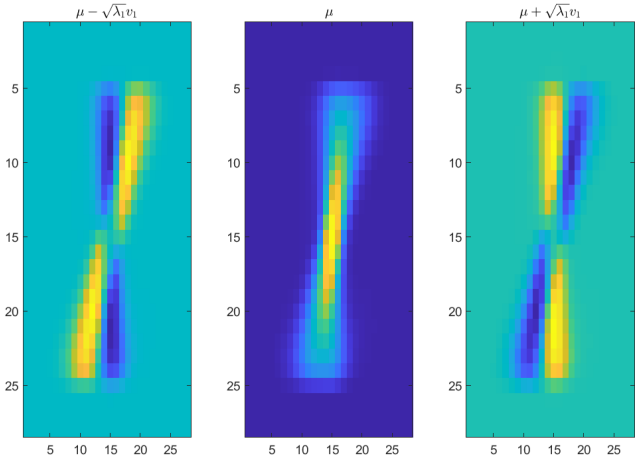

Figure 23: digit 1

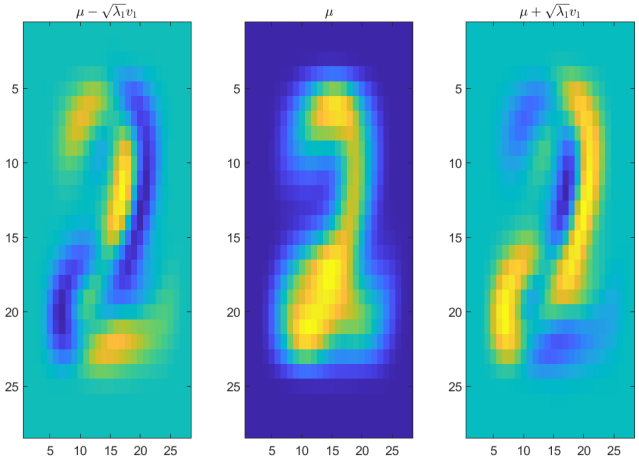

Figure 24: digit 2

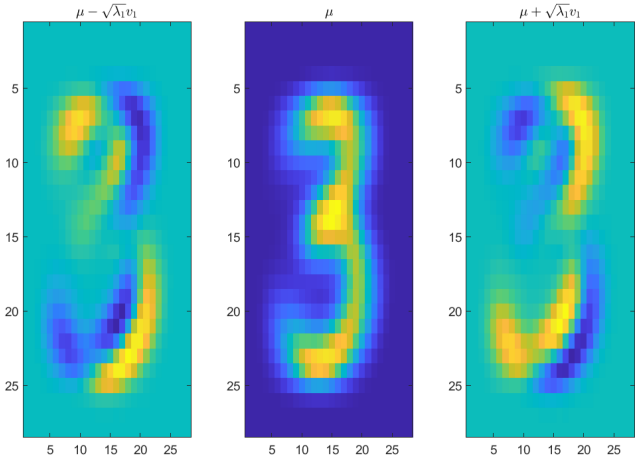

Figure 25: digit 3

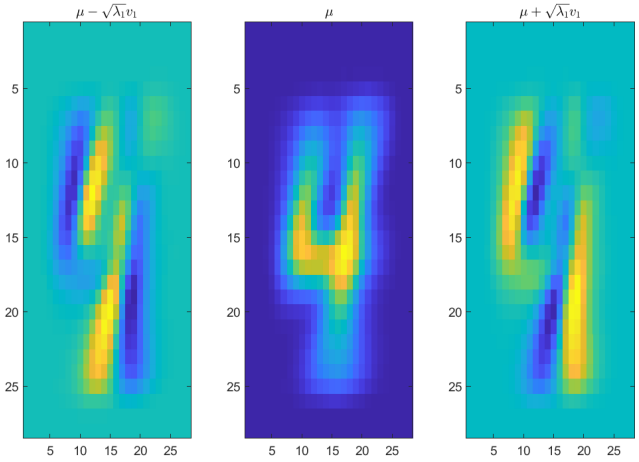

Figure 26: digit 4

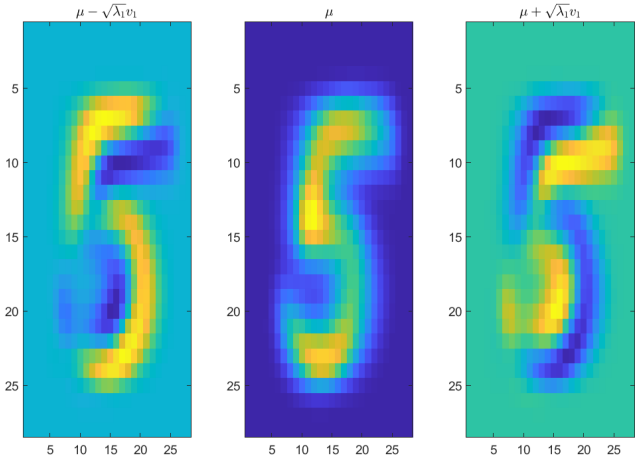

Figure 27: digit 5

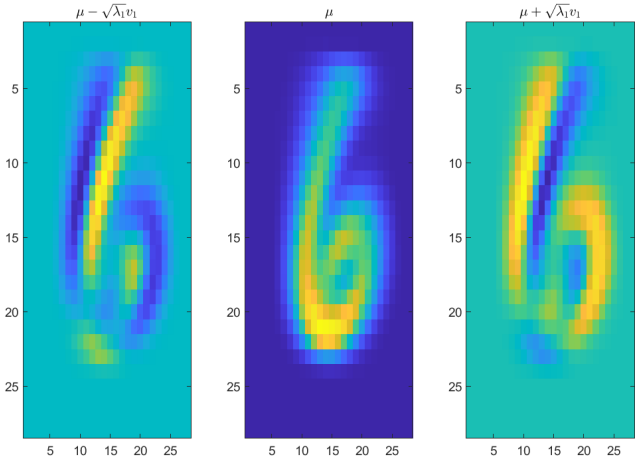

Figure 28: digit 6

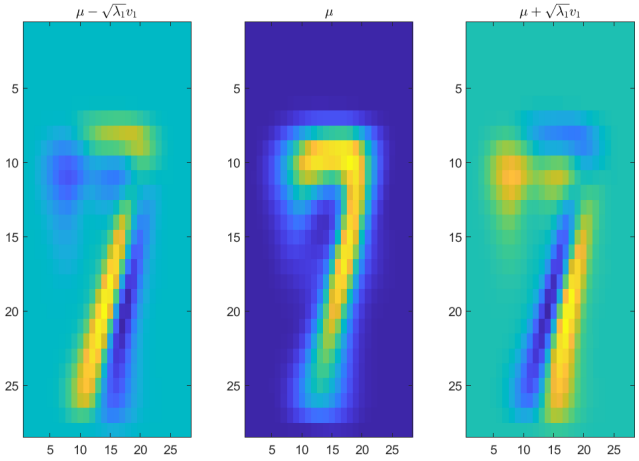

Figure 29: digit 7

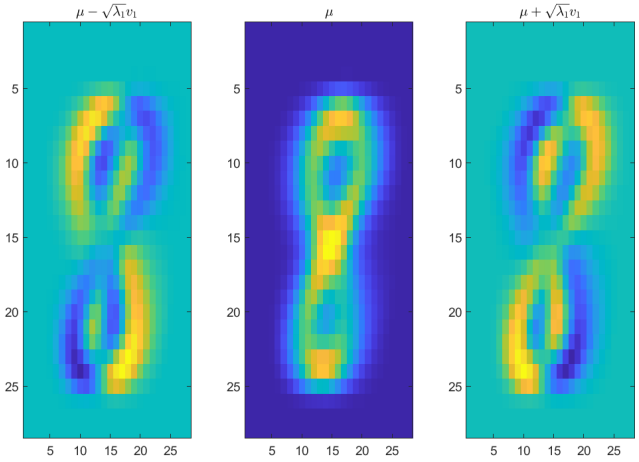

Figure 30: digit 8

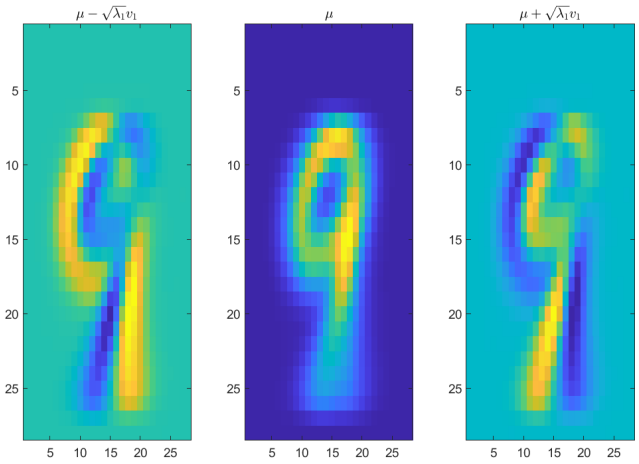

Figure 31: digit 9

4.4 comment on the plots

In the images, yellowish or orangish region represents how the digit is written by people. These three vectors($\mu - \sqrt{\lambda}v_1, \mu, \mu + \sqrt{\lambda}v_1$) depict the variation in the way of writing the digits by different people. Most of the variations of the digits fall in between 1st and 3rd image(along -v1 and along +v1 respectively). For ex, take 1, first image is inclined right and the third image is kinda straight. And the mean is roughly the intermediate state of these two(slightly inclined towards right). So most of the people are writing 1 a bit right tilted and that inclination varied with person to person.

5 Principal Component Analysis (PCA) for Dimensionality Reduction

5.1 Running the code

5.2 Comparision of original and compressed images

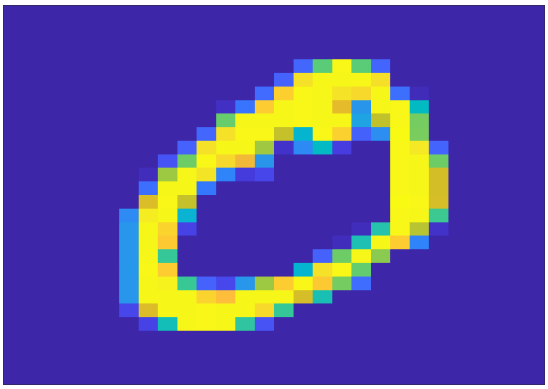

Figure 32: Original

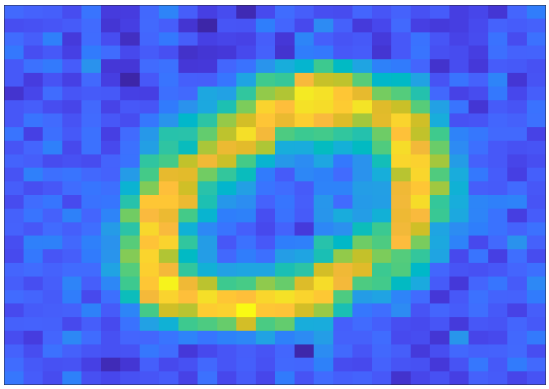

Figure 33: Compressed

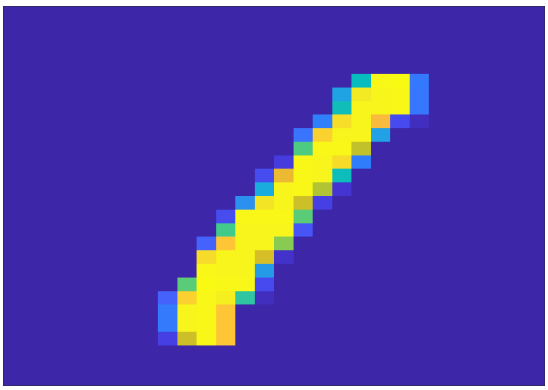

Figure 34: Original

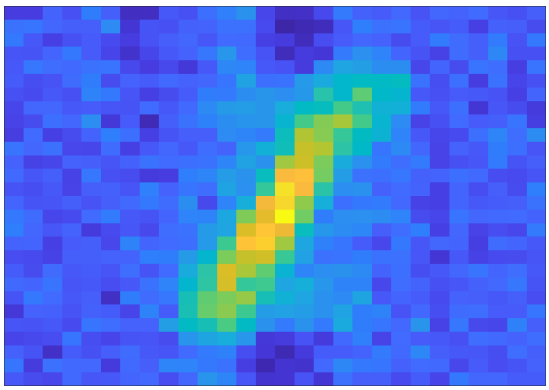

Figure 35: Compressed

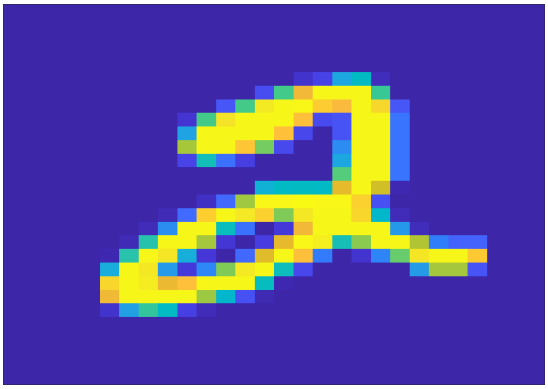

Figure 36: Original

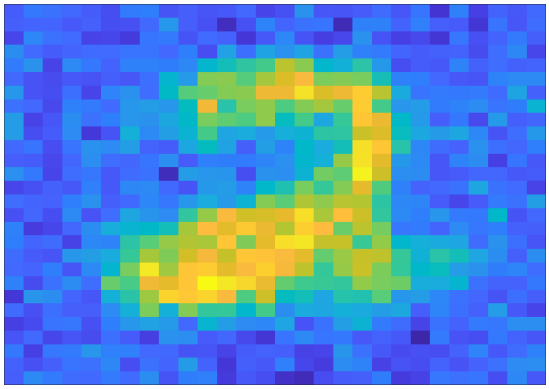

Figure 37: Compressed

Figure 38: Original

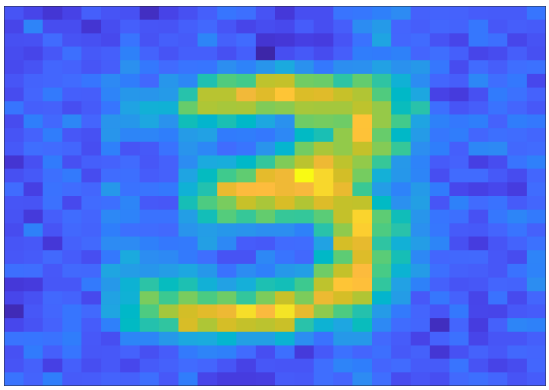

Figure 39: Compressed

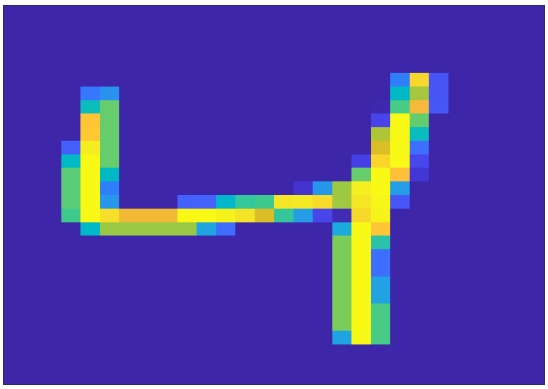

Figure 40: Original

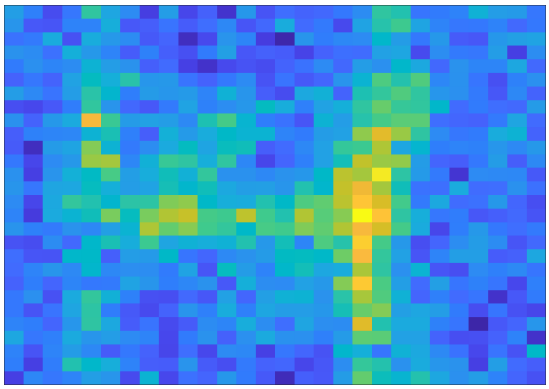

Figure 41: Compressed

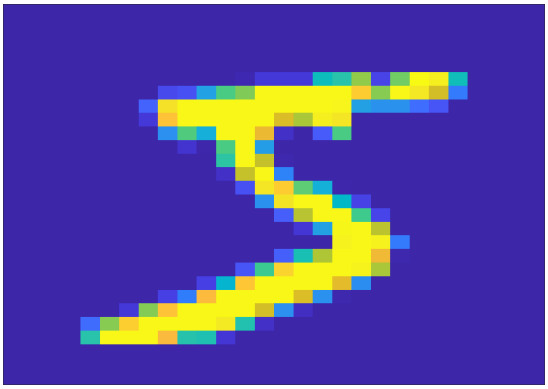

Figure 42: Original

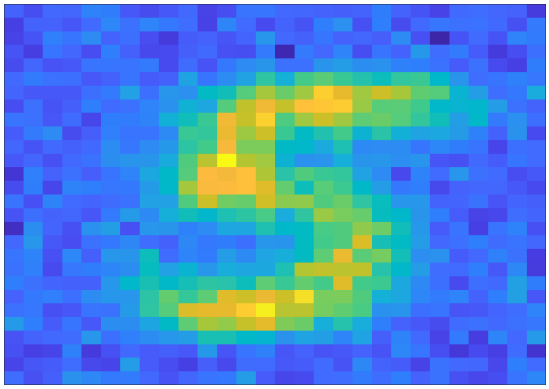

Figure 43: Compressed

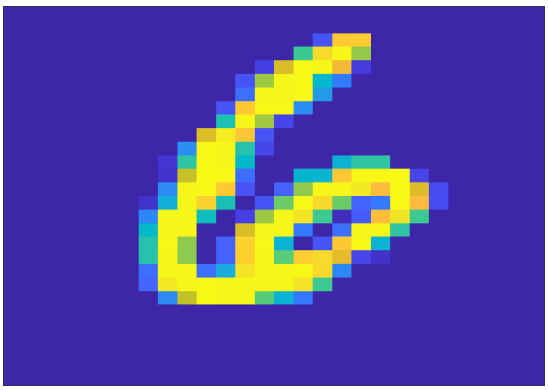

Figure 44: Original

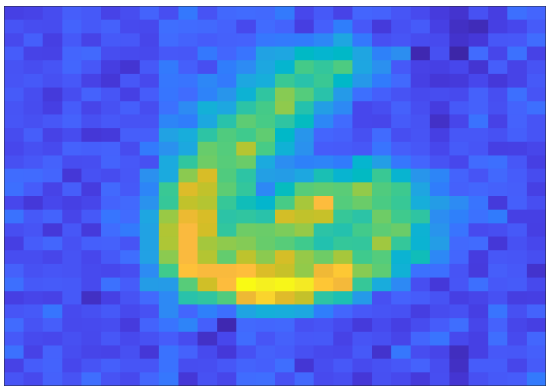

Figure 45: Compressed

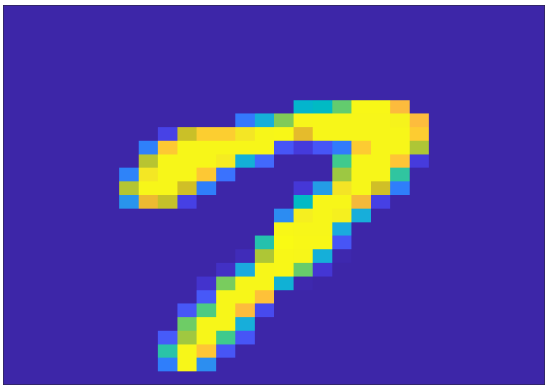

Figure 46: Original

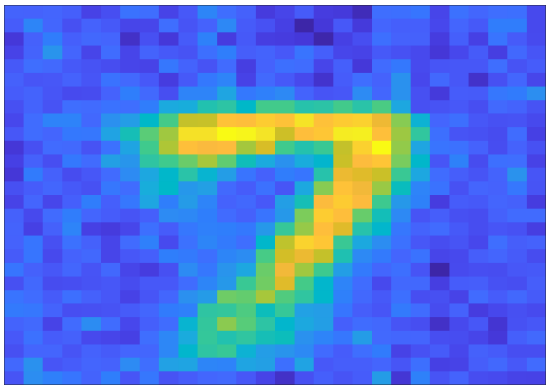

Figure 47: Compressed

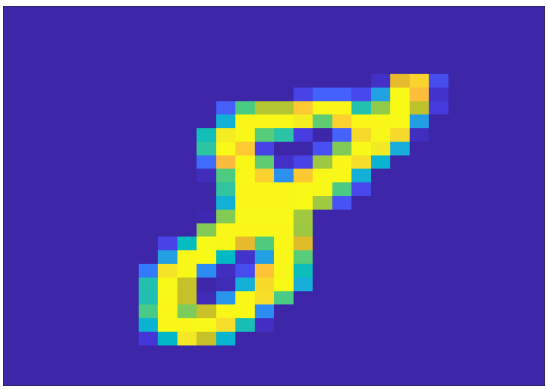

Figure 48: Original

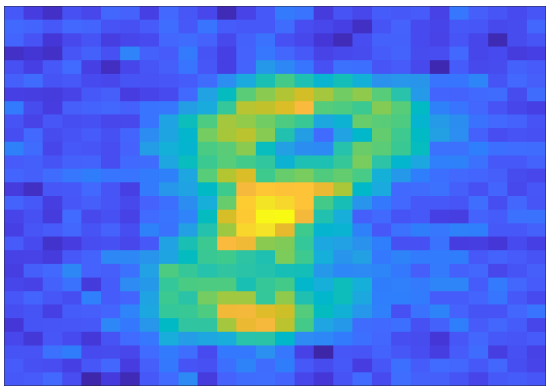

Figure 49: Compressed

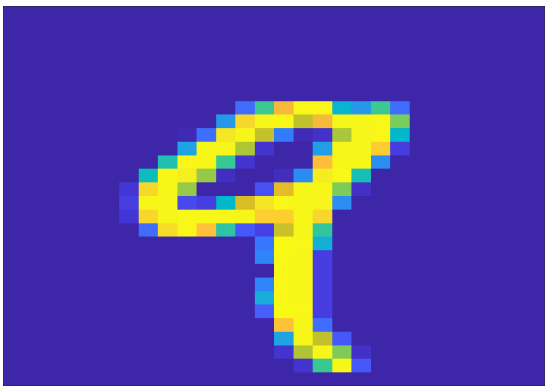

Figure 50: Original

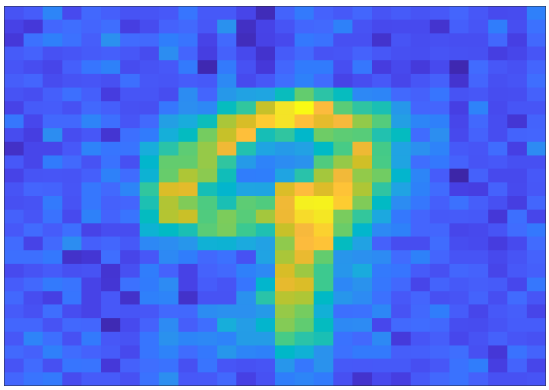

Figure 51: Compressed

6 Principal Component Analysis (PCA) for Another Image Dataset

6.1 Running the code

6.2 Mean and eigen vectors

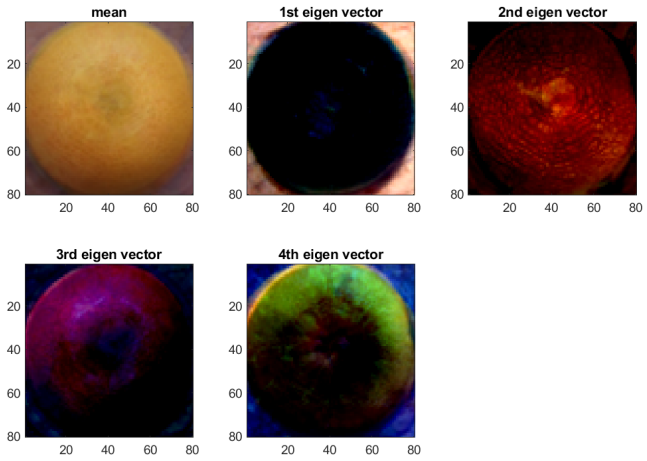

Figure 52: Mean and eigen vectors

6.3 Top 10 eigen vectors

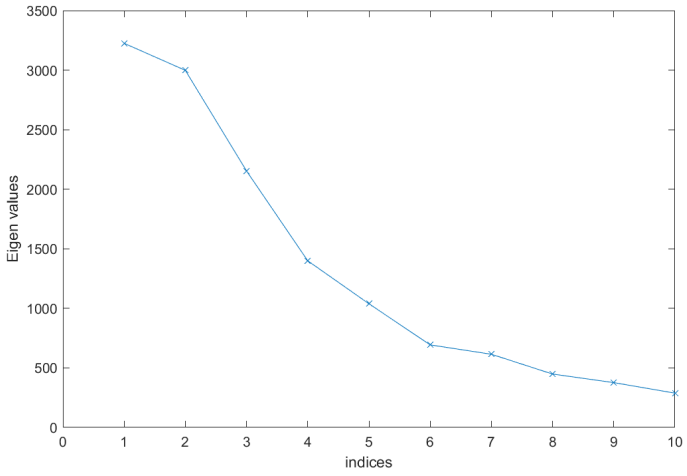

Figure 53: Eigen values

6.4 Closest representations

6.4.1 Algorithm

6.4.2 Results

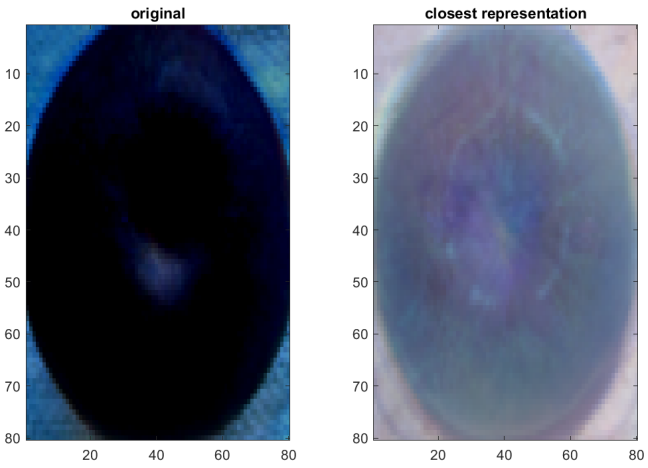

Figure 54: Image 1

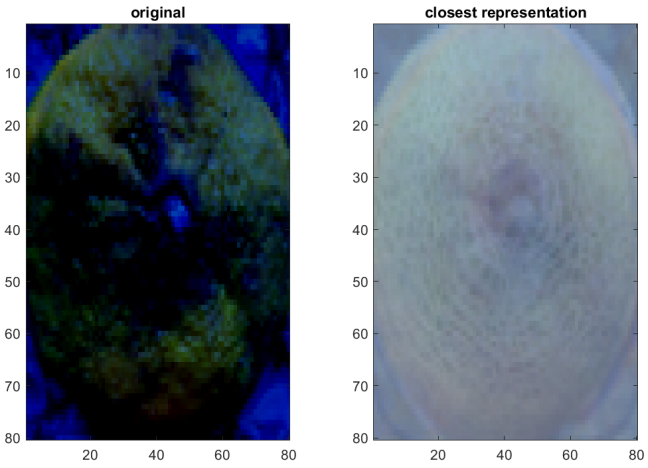

Figure 55: Image 2

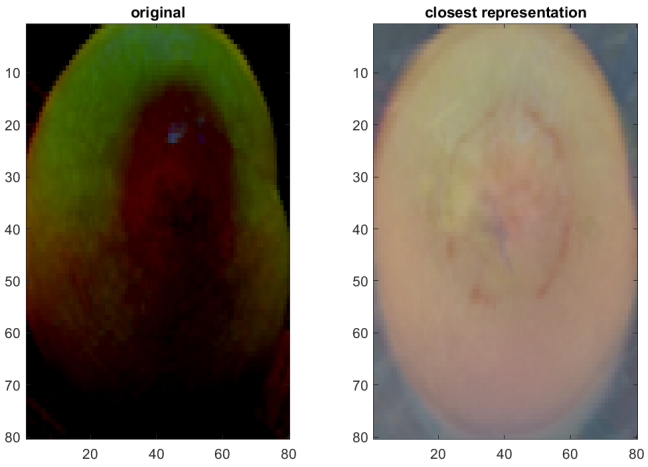

Figure 56: Image 3

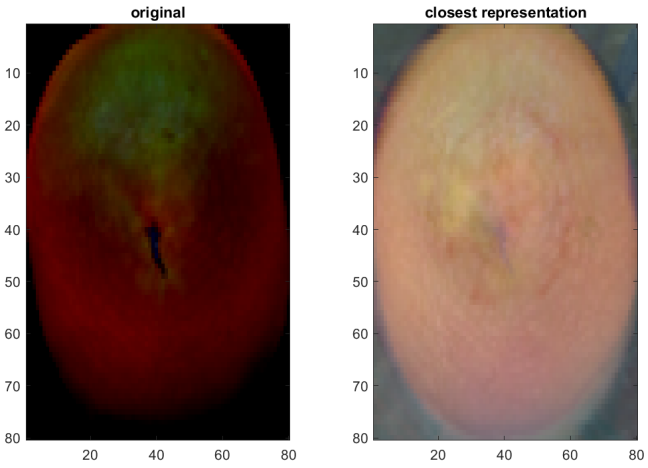

Figure 57: Image 4

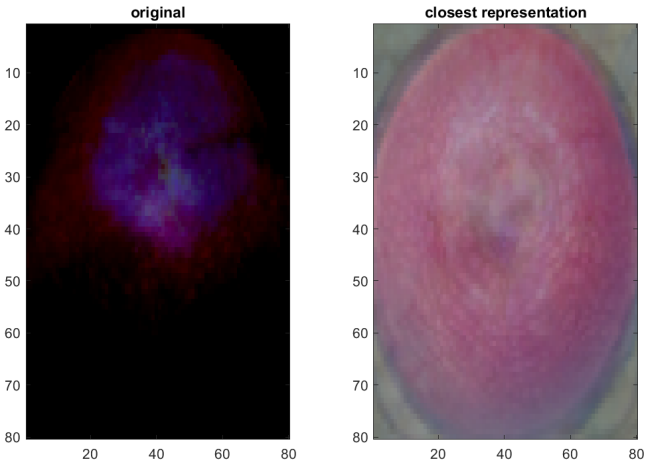

Figure 58: Image 5

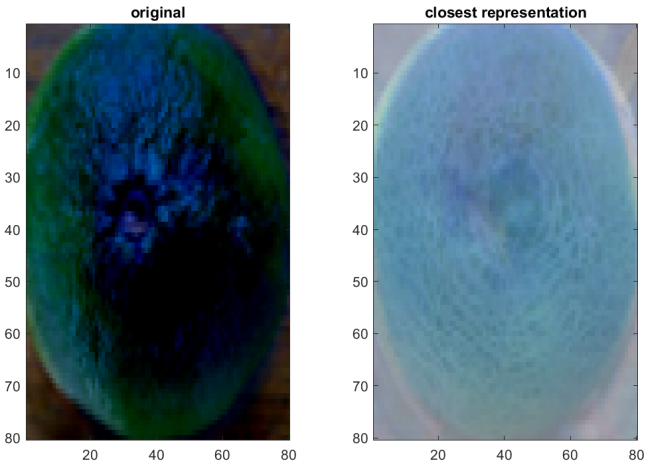

Figure 59: Image 6

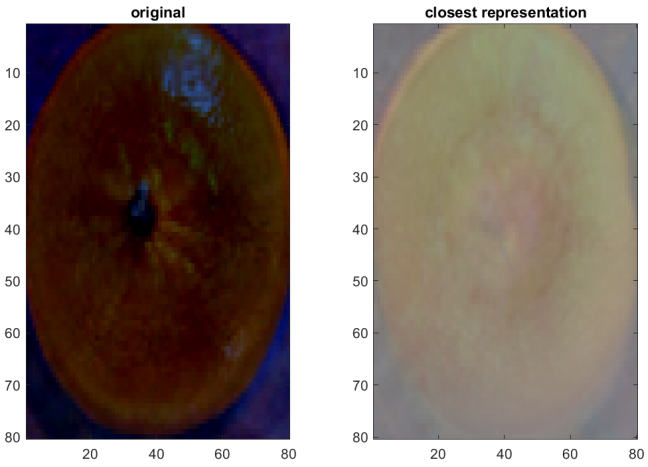

Figure 60: Image 7

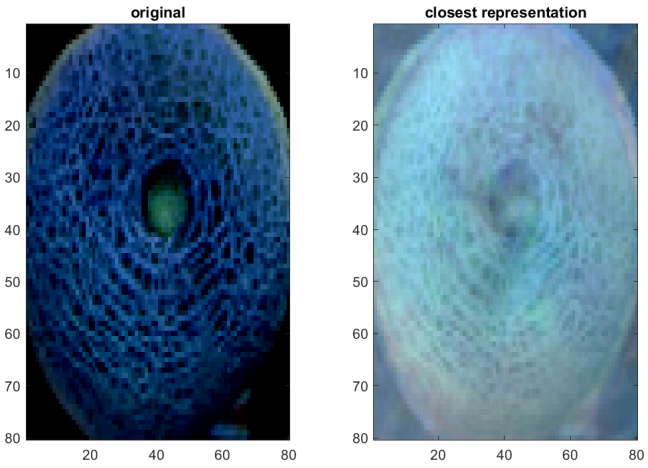

Figure 61: Image 8

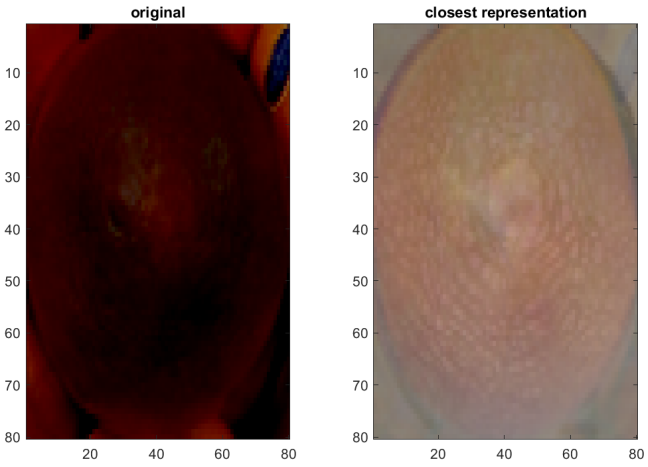

Figure 62: Image 9

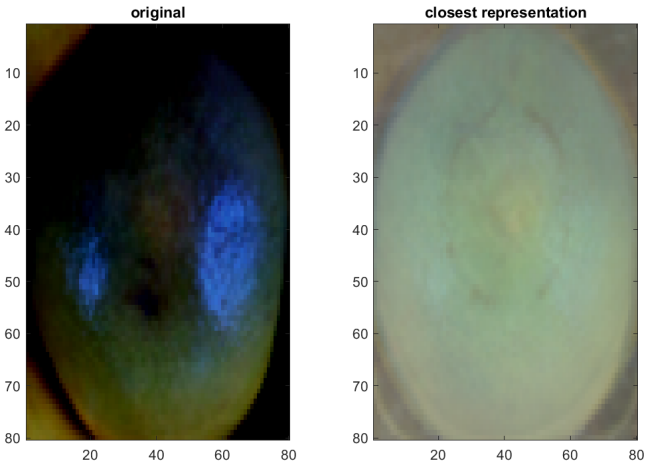

Figure 63: Image 10

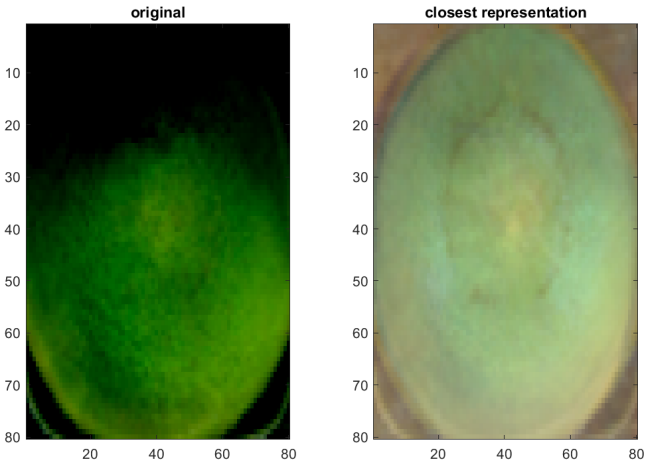

Figure 64: Image 11

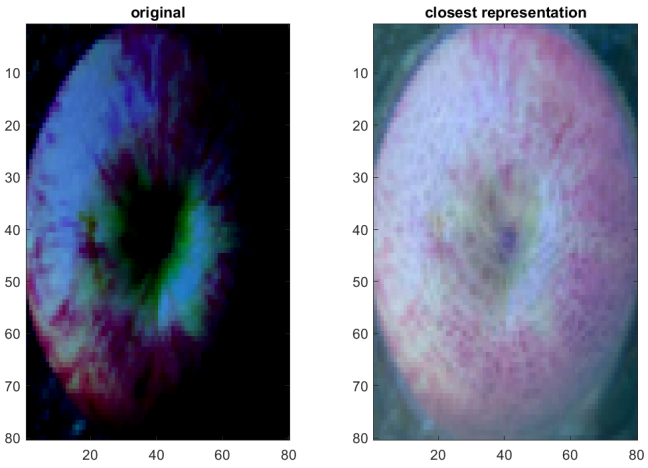

Figure 65: Image 12

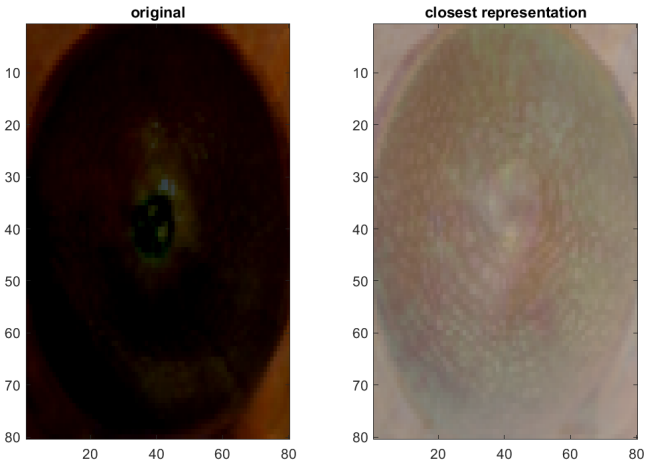

Figure 66: Image 13

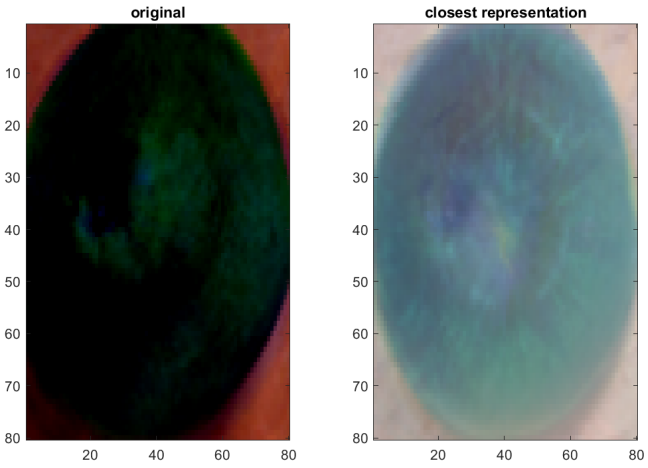

Figure 67: Image 14

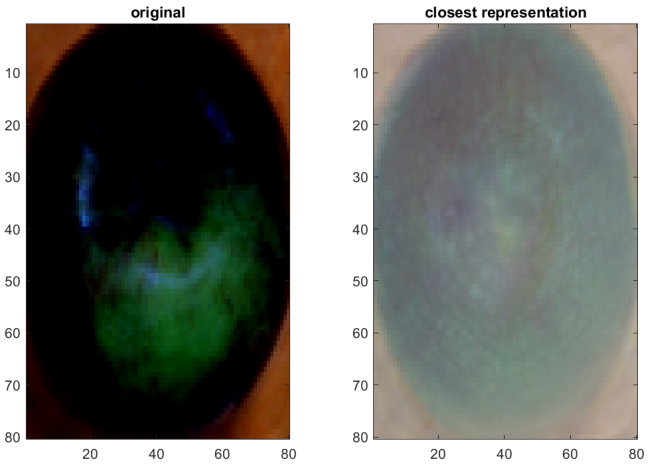

Figure 68: Image 15

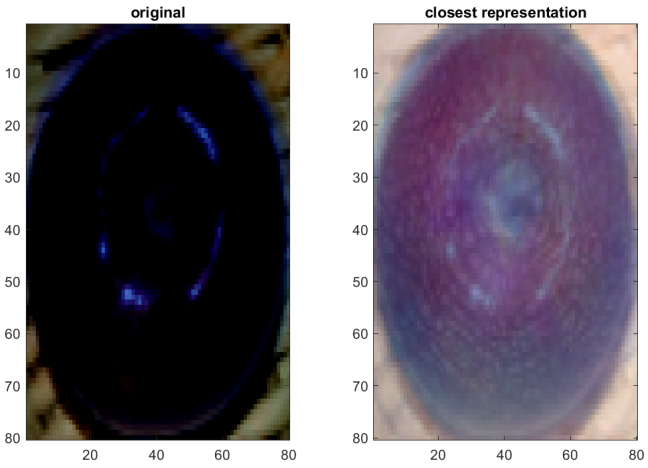

Figure 69: Image 16

6.5 New fruits

6.5.1 Algorithm

We convert data in each datum of size $80 \times 80 \times 3$ into vector of size 19200×1 and let them be $x_1, x_2, ...x_{16}$.

$$\mu = \frac{1}{16}\left(\sum_{i=1}^{16} x_i\right)$$

$$a_i = x_i - \mu$$

$$A = \begin{bmatrix} a_1 & \dots & a_{16} \end{bmatrix}$$

Now let $u_1, ...u_{19200}$ be the eigen vectors of Cov(A). Consider these eigen vectors as sorted in descending order with respect to their eigen values. Each datum in a dataset can be represented as linear combination of these eigen vectors(as the eigen vectors span the whole space of it's dimension) plus mean. To generate a sample random image we can just make all the coefficients 0 except for the top 4 eigen vectors, assign random numbers to the first 4 coefficients(using uniform random number generator rand()) and rescale it after adding mean.

$$NewImage = reshape(rescale(c_1 * u_1 + c_2 * u_2 + c3 * u3 + c4 * u4 + \mu))$$

where (c_1, c_2, c_3, c_4) are the coefficients. Rescaling makes sure that every value in the data is between 0 and 1. Reshaping will generate a $80 \times 0\times$ matrix with the input data which is what we require to visualize the image.

6.5.2 Images

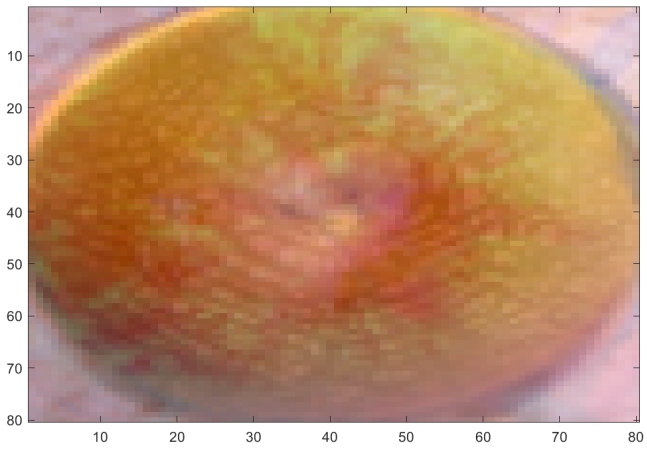

Figure 70: NewFruit 1

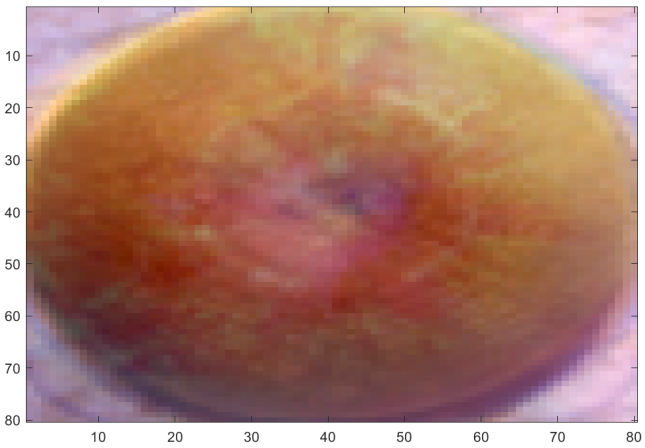

Figure 71: NewFruit 2

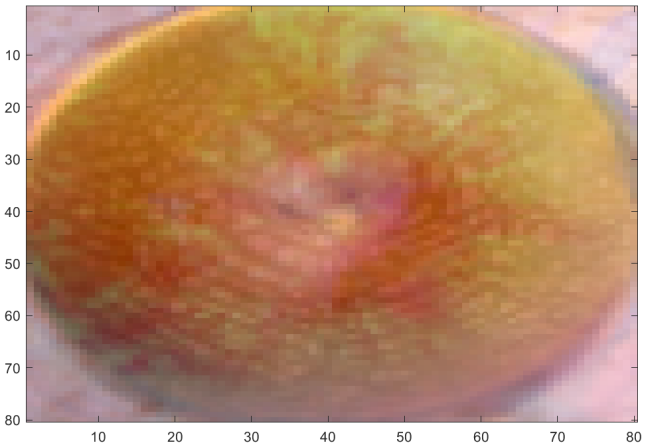

Figure 72: NewFruit 3