Problem 1

(a)

I computed the change in phase angle in the FFT domain and added the magnified change to the phase angle of the image 1 to get the final angle of the magnified image.Finally I inverse transformed of the complex numbers corresponding to the final magnified phase angle to get the magnified image.

```
phaseShift = angle(im2Dft) - angle(im1Dft);
magnification = angle(im1Dft) + magnificationFactor * phaseShift;
[m,n] = pol2cart(magnification, abs(im1Dft))
magnified = ifft2(complex(m,n));
```

The resulting image from the magnification is given below.

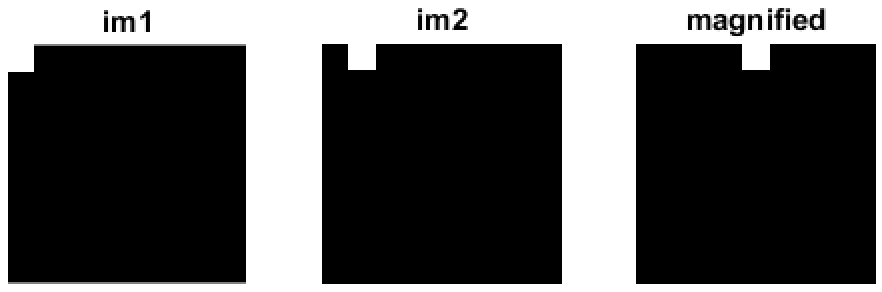

Figure 1: 1D motion magnification

(b)

The resulting image from running part b is given below. There is two direction of motion

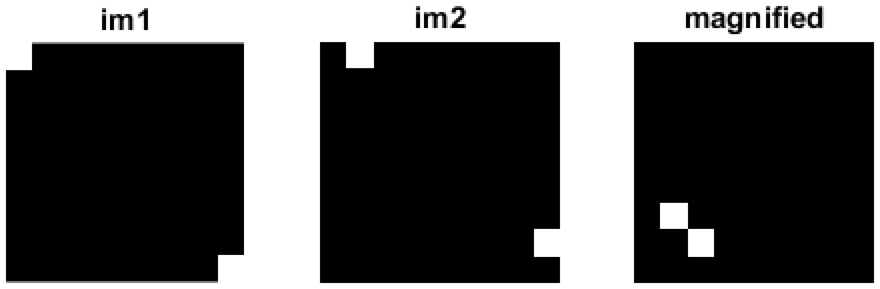

Figure 2: 2d motion magnification

in these images and phase shift due to each of the motion are applied to another one. The local changes in position is lost in the frequency domain and results in both horizontal and vertical shifts in the positions of both objects. Restricting the FFT to the local region and taking smaller patches of magnification helps in resolving this issue. This avoids changes due to motions in non local regions.

(c)

I multiplied both of the images with the gaussian mask and doing the magnificaiton in each iteration.The relevant code for the method is given below:

```
gaussianMask = exp((-(X-x).^2 - (Y- y).^2)/ (2 * sigma ^2 ));
im11 = im1.*gaussianMask;
im22 = im2.*gaussianMask;
windowMagnified = magnifyChange(im11, im22, magnificationFactor);
```

```
magnified = magnified + windowMagnified;
```

The resulting image from the magnification is given below.

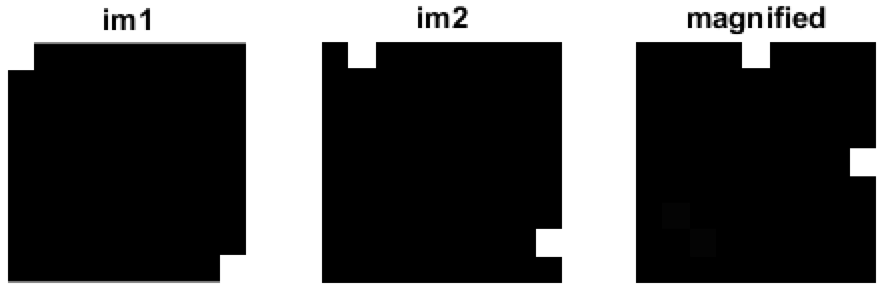

Figure 3: Corrected 2D motion magnification using local gaussian mask

(d)

The video magnification was achieved by following code.

```
%line 49
gaussianMask = exp((-(X-x).^2 - (Y- y).^2)/ (2 * sigma ^2 ));

%magnify phase shift
windowMagnifiedPhase = windowAveragePhase + windowPhaseShift * magnificationF
[m,n] = pol2cart(windowMagnifiedPhase, abs(windowDft));

windowMagnifiedDft = complex(m,n);
windowMagnified = abs(ifft2(windowMagnifiedDft));
```

(2)

(a)

(i) We know that

$$C_{RGB} = TC_{XYZ} \tag{1}$$

We can multiply the C_{XYZ} with T to getC_{RGB}. We get the following plot when we plot the result.

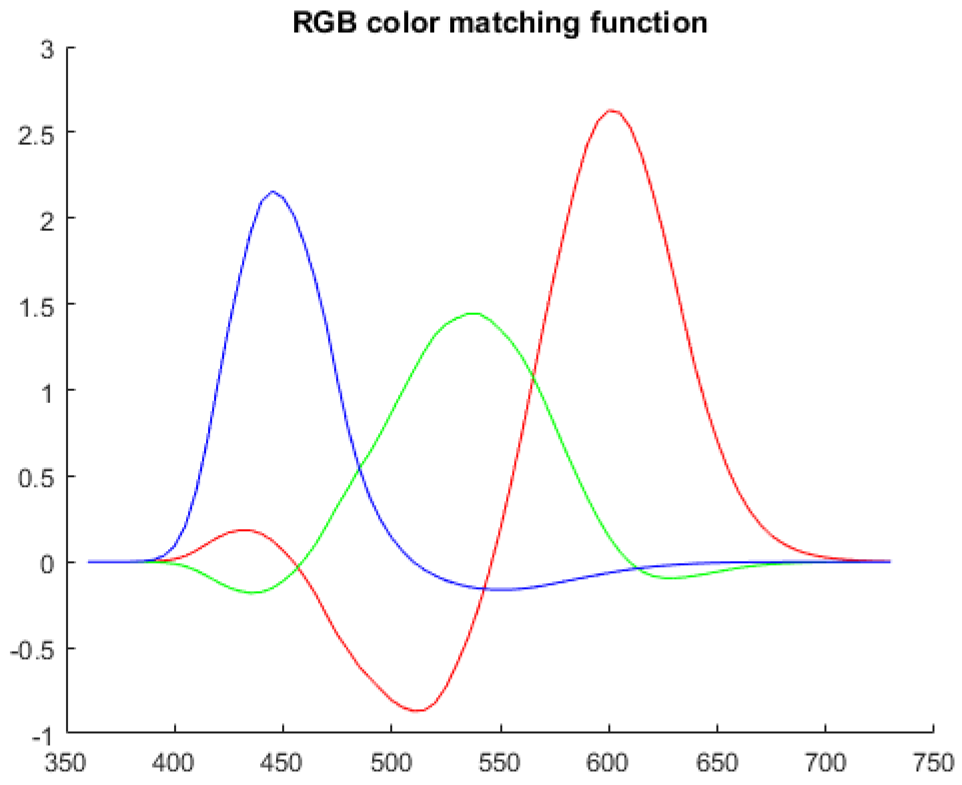

Figure 4: RGB color matching function

From the figure, we can see that the color matching functions are not always positive. Different curve show different range of negativeness. The red color matching function, for instance is negative from wavelengths of 450 to 550 nm. Similarly for blue and green functions, we can see different negative regions.

(ii) We know that $C_{RGB}.P = I$. Taking the pseudo inverse of the matrix C_{RGB} we can get the primary light spectra.

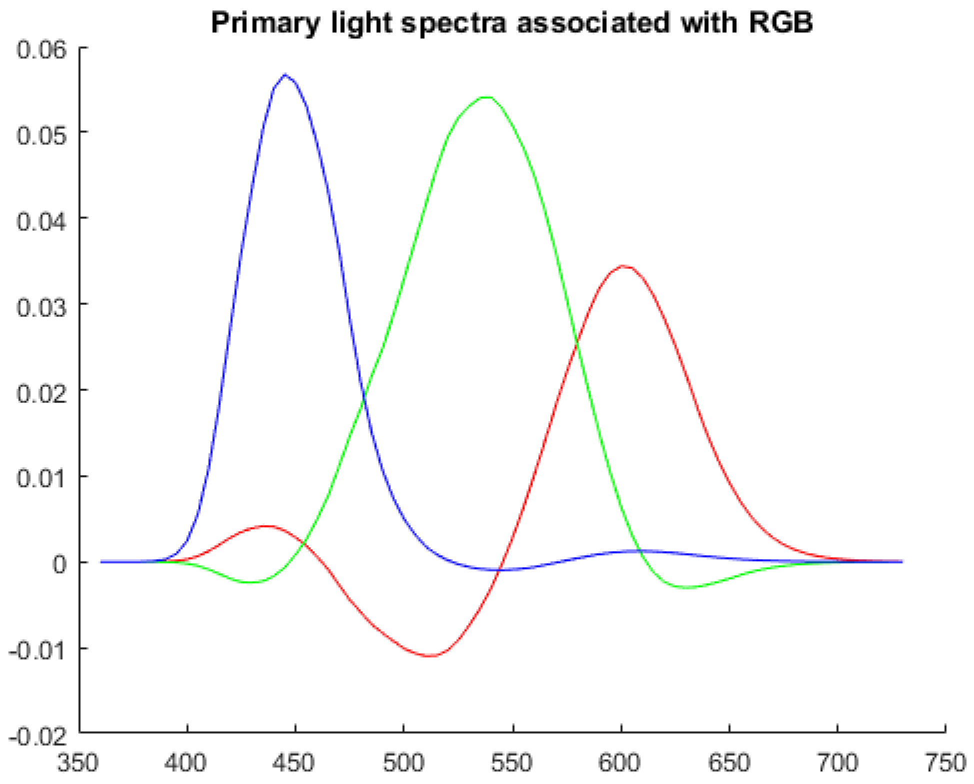

Figure 5: RGB primary light spectra

As like the color matching functions, these light spectra are also negative in some region. The red spectra is negative between 450nm and 550 nm. Similary, green and blue primary spectra also have their respective negative regions.

(iii) Plotting the CIE we get the following graph at figure 6.

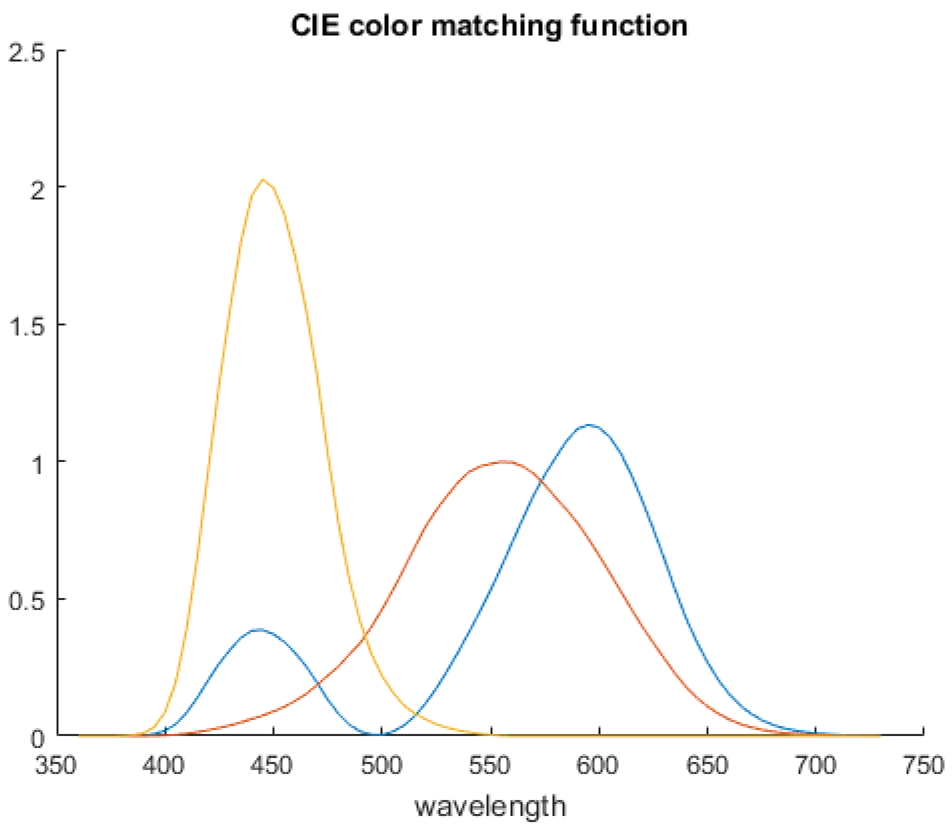

Figure 6: CIE color matching function

We can see that the curves do not have the negative regions. This is not a coincidence as it was specifically designed to be such.

(iv) As like before, we take the pseudo inverse of the CIE colormatching function. We get the following graph at figure 7. We can also see that the primaries are not always positive. We can see a large negative region on the blue spectrum between 450 and 550 nm wavelength. This is in contrast to negative region of red wavelength in RGB primary. Similarly, red and yellow also have some negative region as seen in the plot.

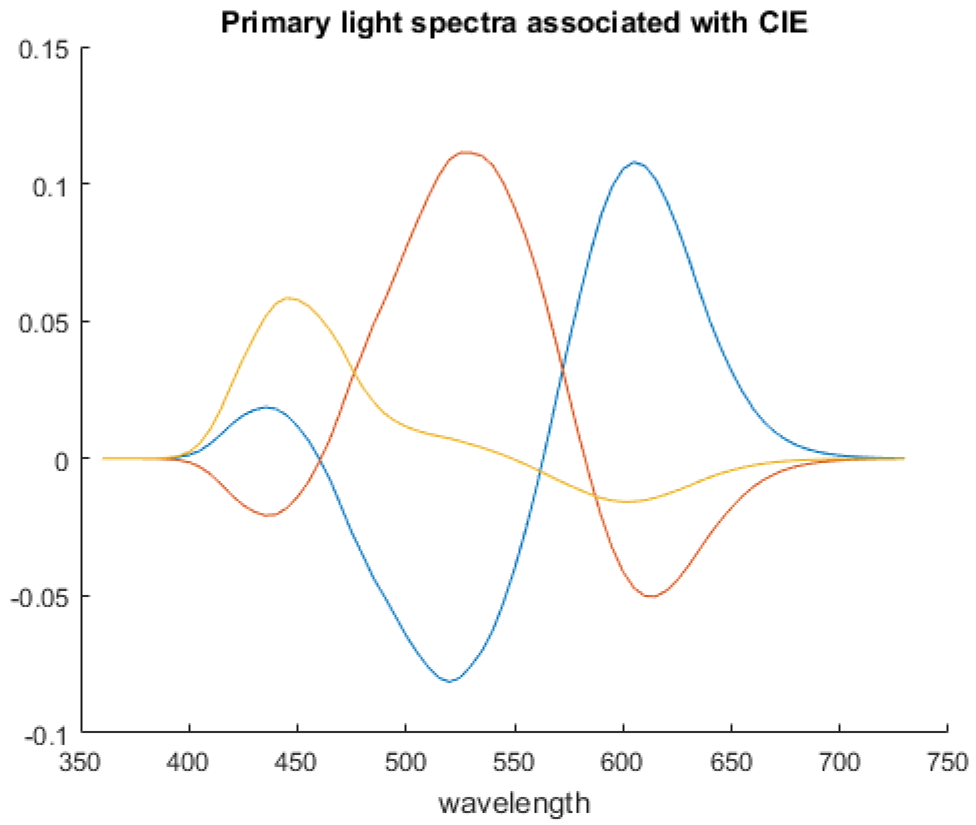

Figure 7: CIE color primary light spectra

(b)

Each primary spectrum in Figure 8 has negative region. For example, the blue curve has it's negative range from 450 nm to 550 nm whereas yellow has negative range from 550nm to 700nm.

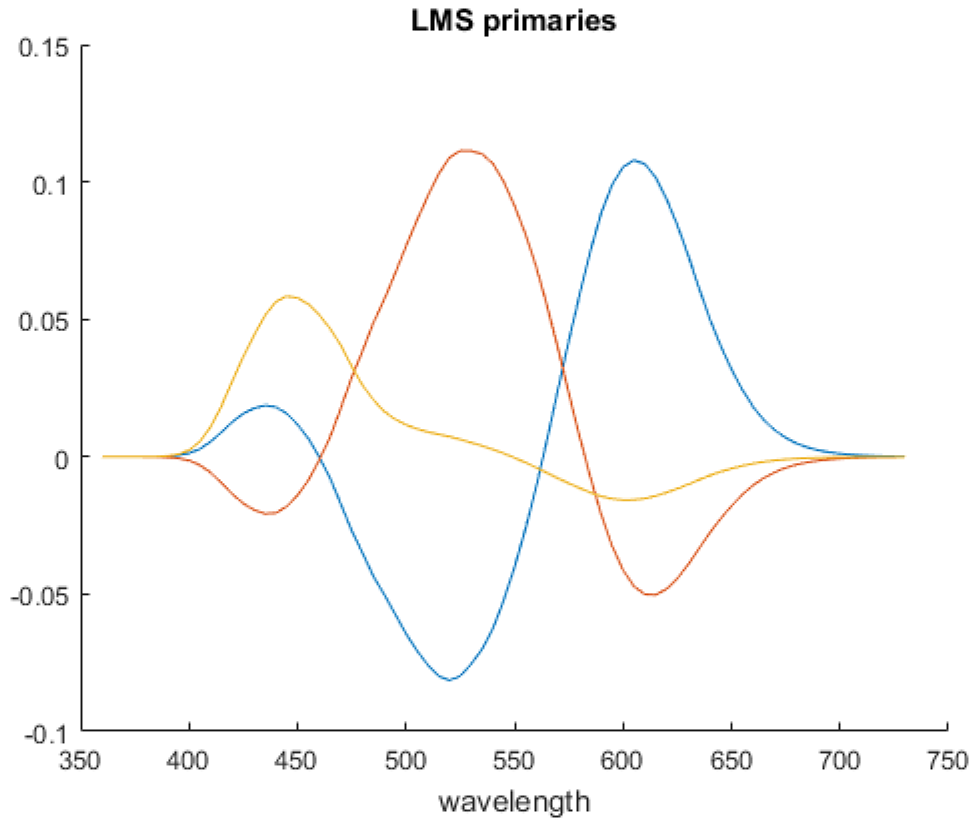

Figure 8: LMS Primaries

(C)

Let the non negative spectral response curves be L, M and S.We denote their components as $l_1, m_1, ...$ We define P as shown below which is also a non negative matrix as it contains all the positive numbers in C divided by a normalization constant. We showed that the particular arrangement when multiplied with C gives I, due to orthogonality properties of L, M and S.

d

We cannot uniquely determine the associated primaries P for a given color matching functions C.

$$C = \begin{bmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \\ s_1 & s_2 & s_3 \end{bmatrix} \quad ; \quad \begin{aligned} L &= (l_1 \; l_2 \; l_3) \\ M &= (m_1, m_2, m_3) \\ S &= (s_1, s_2, s_3) \end{aligned}$$

We define P as,

$$\vec{L} \cdot \vec{M} = \vec{M} \cdot \vec{S} = \vec{S} \cdot \vec{L} = 0.$$

$$P = \begin{bmatrix} \frac{l_1}{|L|^2} & \frac{m_1}{|M|^2} & \frac{s_1}{|S|^2} \\ \frac{l_2}{|L|^2} & \frac{m_2}{|M|^2} & \frac{s_2}{|S|^2} \\ \frac{l_3}{|L|^2} & \frac{m_3}{|M|^2} & \frac{s_3}{|S|^2} \end{bmatrix}$$

Since C is positive, P is also positive. We are just dividing by square of magnitude.

Now,

$$C \cdot P = \begin{bmatrix} \frac{1}{|L|^2} \cdot l_1^2 + l_2^2 + l_3^2 & 0 & 0 \\ 0 & \frac{1}{|M^2|}(m_1^2 + m_2^2 + m_3^2) & 0 \\ 0 & 0 & \frac{1}{|S|^2}(s_1^2 + s_2^2 + s_3^2) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$\Rightarrow C \cdot P = I$

$\Rightarrow$ This proves P is valid set of primaries.

If y is a vector in null space of C, then

$$C.(inv(C) + k * [y, y, y]) = I$$

for all scaler values of k. This means I can have more than one P that give I when left multiplied by C.

However, the opposite is true for the uniquely determined P. We can always find a unique C associated with P.From the above equation, it follows that every P has to be in form of

$$inv(C) + k * [y, y, y])$$

.Also, from uniqueness of left side inverse of matrix inv(C), we can conclude that only one C can reduce inv(C) to I and k*[y y y] to 0 to result on CP = I.