Problem 1:

Part (a):

\[ N = 16 \quad N_0 = 2 \quad N_1 = 9 \quad N_2 = 3 \quad N_3 = 2 \]

Part (b):

<table>
<thead>
<tr>
<th>Values</th>
<th>Prob ( \frac{3}{16} )</th>
<th>( \frac{2}{16} )</th>
<th>( \frac{9}{16} )</th>
<th>( \frac{3}{16} )</th>
<th>( \frac{2}{16} )</th>
<th>( \frac{3}{16} )</th>
<th>( \frac{1}{16} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \frac{2}{16} )</td>
<td>( \frac{1}{16} )</td>
<td>( \frac{9}{16} )</td>
<td>( \frac{3}{16} )</td>
<td>( \frac{2}{16} )</td>
<td>( \frac{3}{16} )</td>
<td>( \frac{1}{16} )</td>
</tr>
<tr>
<td></td>
<td>( \frac{3}{16} )</td>
<td>( \frac{2}{16} )</td>
<td>( \frac{3}{16} )</td>
<td>( \frac{2}{16} )</td>
<td>( \frac{3}{16} )</td>
<td>( \frac{2}{16} )</td>
<td>( \frac{3}{16} )</td>
</tr>
<tr>
<td></td>
<td>( \frac{2}{16} )</td>
<td>( \frac{3}{16} )</td>
<td>( \frac{2}{16} )</td>
<td>( \frac{3}{16} )</td>
<td>( \frac{2}{16} )</td>
<td>( \frac{3}{16} )</td>
<td>( \frac{1}{16} )</td>
</tr>
</tbody>
</table>

Part (c):
Problem 2.

Part (a): \[ H = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ -1 & -1 & -1 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} [1~1~1] \]

\[ h_x = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad h_y = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \Rightarrow \text{it is separable.} \]

Part (b): The filter \[ \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \] is a difference filter \[ \Rightarrow \text{high pass filter} \]

\[ \begin{bmatrix} 1 \\ 1 \end{bmatrix} \] is an averaging filter \[ \Rightarrow \text{low pass filter} \]

The overall filter is taking derivative in \( \downarrow \) direction and average in \( \rightarrow \) direction \[ \Rightarrow \text{detects horizontal edges.} \]

Part (c):

\[ \mathcal{F}\{ h_x \} = \frac{1}{\sqrt{3}} \left[ e^{j2\pi u} + e^{-j2\pi u} \right] = -\frac{2j}{\sqrt{3}} \sin(2\pi u) \]

\[ \mathcal{F}\{ h_y \} = \frac{1}{\sqrt{3}} \left[ 1 + 2 \cos(2\pi u) \right] \Rightarrow H(u, v) = -\frac{2j}{3} \sin(2\pi u) [1 + 2\cos(2\pi u)] \]

\[ H(u, 0) = -\frac{2j}{3} \sin(2\pi u) \]

\[ H(u, v) = \frac{2j}{3} \sin(2\pi u) \]

\[ H(0, v) = 0 \]

\[ H(\pm \frac{1}{2}, v) = 0 \]

Part (d): Filter acts as a horizontal edge detector.
Problem 3:

\[ F_1 = \begin{bmatrix} 6 & 5 & 4 & 3 \\ 5 & 4 & 3 & 2 \\ 4 & 3 & 2 & 1 \\ 3 & 2 & 1 & 1 \end{bmatrix} \Rightarrow \text{after averaging} \quad G_1 = \begin{bmatrix} 5 & 3 \\ 3 & 1 \end{bmatrix} \]

\[ \Rightarrow \text{interpolating} \quad I_1 = \begin{bmatrix} 5 & 5 & 3 & 3 \\ 5 & 5 & 3 & 3 \\ 3 & 3 & 1 & 1 \\ 3 & 3 & 1 & 1 \end{bmatrix} \Rightarrow \text{Subtract} \quad L_1 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \]

next level

\[ \text{averaging on } G_1 \Rightarrow \quad G_2 = \begin{bmatrix} 3 \\ 3 \end{bmatrix} \Rightarrow \quad I_2 = \begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix} \Rightarrow L_2 = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix} \]

For the construction we have:

\[ G_2 = \begin{bmatrix} 3 \\ 3 \end{bmatrix} \Rightarrow \text{interpolate} \quad I_2 = \begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix} \Rightarrow \text{add to } L_2 \Rightarrow G_3 = \begin{bmatrix} 5 & 3 \\ 3 & 1 \end{bmatrix} \]

\[ G_3 = \begin{bmatrix} 5 & 3 \\ 3 & 1 \end{bmatrix} \Rightarrow \text{interpolate} \quad I_1 = \begin{bmatrix} 5 & 5 & 3 & 3 \\ 5 & 5 & 3 & 3 \\ 3 & 3 & 1 & 1 \\ 3 & 3 & 1 & 1 \end{bmatrix} \Rightarrow \text{add to } L_1 \Rightarrow F = \begin{bmatrix} 6 & 5 & 4 & 3 \\ 5 & 5 & 3 & 3 \\ 3 & 3 & 1 & 1 \\ 3 & 3 & 1 & 1 \end{bmatrix} \]
Problem 4.6. \( Y_{m \times n} \) Observed gray values of \( N \) pixels.

Part (a): \( \min_x \|X\|_0 \quad \text{s.t.} \quad \|Y - MDX\|_2 \leq \varepsilon \)

This problem is not convex since \( L_0 \)-norm is not a convex norm and it is \( NP \)-Hard to solve.

Part (b): Using techniques in convex, an equivalent problem is

\[
\min_x \|Y' - MDX\|_2^2 \quad \text{s.t.} \quad \|X\|_0 \leq S
\]

which also is equal to \( \min_y \|Y' - MDX\|_2^2 + \lambda \|X\|_1 \)

Part (c): one relaxation to get convexity is changing \( L_0 \)-norm to \( L_1 \)-norm

\( \Rightarrow \min_x \|Y' - MDX\|_2^2 + \lambda \|X\|_1 \)

where \( L_1 \)-norm is sum of absolute values.

Part (d): when we calculate \( X^{opt} \) by solving the problem we can calculate the missing pixels by \( (I-M)DX^{opt} \) and we can replace them by text in image, assuming the observation is noisy we can have \( DX^{opt} \) as output to reduce the effect of noise too.
Problem 5:

Part (a):

\[ I_x(m,n) = I(m,n) - I(m,n-1) \quad \text{assuming outside is 0} \]

\[ \Rightarrow \theta = \tan^{-1} \left( \frac{I_x}{I_y} \right) \]

\[ \Rightarrow \text{HoG} \]

\[ \text{Should be given special symbol.} \]

This is created due to the fact that we padded zeros to the left.

Part (b): if we normalize (and threshold in the SIFT algorithm) then the HoG is invariant to slight change and intensity change.

This histogram is not rotation invariant since we did not circularly shift the histogram to get the main dominant direction in the first bin. But in the actual SIFT descriptor we actually calculate the dominant direction of the patch and shift the histograms of sub-patches in the first bin to get rotation invariant result.
Problem 6

\[ \alpha = a_0 + a_1 u + a_2 v \Rightarrow x + c_1 u + c_2 v = a_2 + a_1 u + a_2 v \]

\[ 1 + c_1 u + c_2 v \]

\[ \Rightarrow \begin{cases} x = a_0 - a_1 u - a_2 v - c_1 u - c_2 v \\ y = b_0 + b_1 u + b_2 v - c_1 y - c_2 y \end{cases} \]

\[
\begin{bmatrix}
1 & u_1 & v_1 & 0 & 0 & -u_1 x_1 - v_1 y_1 \\
0 & 0 & 0 & 1 & u_1 & v_1 - u_1 y_1 - v_1 y_1
\end{bmatrix}
\begin{bmatrix}
a_0 \\
a_1 \\
a_2 \\
b_0 \\
b_1 \\
b_2 \\
c_1 \\
c_2
\end{bmatrix} =
\begin{bmatrix}
x_1 \\
y_1
\end{bmatrix}
\]

we want to have \( A\alpha = x \) if \( N = 8 \Rightarrow \alpha = A^{-1}x \)

if \( N > 8 \) which is usually the case we solve least squares problem.

\[ N > 8 \Rightarrow \text{over-determined} \Rightarrow \min_{\alpha} \| A\alpha - x \|_2 \]

\[ \Rightarrow \frac{\partial}{\partial \alpha} \| A\alpha - x \|_2^2 = -2A^T(x - A\alpha) = 0 \Rightarrow \alpha_{\text{opt}} = (A^TA)^{-1}A^Tx \]

one other method is DLT:

\[
\begin{bmatrix}
1 & u_1 & v_1 & 0 & 0 & -u_1 x_1 - v_1 y_1 \\
0 & 0 & 0 & 1 & u_1 & v_1 - u_1 y_1 - v_1 y_1
\end{bmatrix}
\begin{bmatrix}
a_0 \\
a_1 \\
a_2 \\
b_0 \\
b_1 \\
b_2 \\
c_1 \\
c_2
\end{bmatrix} =
\begin{bmatrix}
x_1 \\
y_1
\end{bmatrix}
\]

\[ A\alpha = 0 \text{ least squares } \min_{\alpha} \| A\alpha \|_2^2 = \alpha^TA^TA\alpha \]

solution \( \alpha_{\text{opt}} \) = eigen vector of \( A^TA \) with the minimal evalue.
Problem 7.8

For this problem we choose to use Harris corners in laplacian as feature points and SIFT as descriptor.

Step 1: find the Laplacian scale images for both Image 1 and Image 2. This is done using filters $\sigma_{n-1} \cdot \sigma_{n}$ and down sampling and then subtracting adjacent images.

Step 2: extract Harris feature points in multiple scales and for each detect the characteristic scale.

Step 3: for each feature point in Images create the SIFT descriptor and save all feature points and their descriptor.

Step 4: try to find corresponding points between Image 1 and Image 2. This is achieved by comparing the descriptors and taking the one that has the nearest distance. Since there might be ambiguity we have:

$$d_1 < d_2 \quad \text{if} \quad d_1 < d_2 \iff \frac{d_1}{d_2} < T_v \implies \text{keep } d_1 \text{ as match else delete both.}$$

Step 5: after finding the matching points use least squares or RANSAC to find the homography or affine mapping that best describes the mapping between two images.

Step 6: use the mapping to warp one Image to other coordinate (this is done with inverse mapping and interpolation).

Step 7: stitch two images to get the result. This can be done using pyramid blending. Some smoothing might be required.
Problem 8:

Part (a):

\[ F(x+dx, y+dy) - G(x,y) = 0 \Rightarrow \frac{\partial F}{\partial x} \ dx + \frac{\partial F}{\partial y} \ dy + F(x,y) - G(x,y) = 0 \]  

\[ \Rightarrow \text{optical flow eq.} \quad \frac{\partial F}{\partial x} \ dx + \frac{\partial F}{\partial y} \ dy + F(x,y) - G(x,y) = 0 \]

\[
\begin{bmatrix}
\frac{dx}{dx}
\end{bmatrix}
= 
\begin{bmatrix}
a_0 + a_1 x + a_2 y \\
b_0 + b_1 x + b_2 y
\end{bmatrix}
= 
\begin{bmatrix}
1 & x & y & 0 & 0 & 0 \\
0 & 0 & 1 & x & y
\end{bmatrix}
\begin{bmatrix}
a_0 \\
a_1 \\
a_2 \\
b_0 \\
b_1 \\
b_2
\end{bmatrix}
\]

Simple case: Affine.

\[ du = [1 \ x \ y] \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ b_0 \\ b_1 \\ b_2 \end{bmatrix} \Rightarrow du = A \cdot \theta \Rightarrow \frac{\partial F}{\partial x} A(x,y) \cdot a + \frac{\partial F}{\partial y} A(x,y) \cdot b + F(x,y) - G(x,y) = 0 \]

(b)

Least squares problem:  
\[ E_{of} = \frac{1}{2} \sum_{i=1}^{N} \left( \frac{\partial F}{\partial x} A(x_i,y_i) \cdot a + \frac{\partial F}{\partial y} A(x_i,y_i) \cdot b + F(x_i,y_i) - G(x_i,y_i) \right)^2 \]

We try to minimize \( E_{of} \)  
\[ \frac{\partial E_{of}}{\partial a} = 0, \quad \frac{\partial E_{of}}{\partial b} = 0 \]

And we get the following equation:

\[
\begin{bmatrix}
\sum_{x \in X} \frac{\partial^2 F}{\partial x^2} A^T A \\
\sum_{x \in X} \frac{\partial^2 F}{\partial x \partial y} A^T A \\
\sum_{y \in Y} \frac{\partial^2 F}{\partial y^2} A^T A \\
\sum_{y \in Y} \frac{\partial^2 F}{\partial x \partial y} A^T A
\end{bmatrix}
\begin{bmatrix}
a \\
b
\end{bmatrix}
= 
\begin{bmatrix}
\sum_{x \in X} (G(x,y) - F(x,y)) \frac{\partial F}{\partial x} A^T \\
\sum_{y \in Y} (G(x,y) - F(x,y)) \frac{\partial F}{\partial y} A^T
\end{bmatrix}
\]

\[ Sa = t \Rightarrow a = S^{-1} t \]

In order to find the moving object, we can compensate for camera motion based on the mapping we found and use other motion estimation methods like EBMMA to find the moving object based on warped image and image 1.

You just need to detect pixels with large error after global motion compensation.
Problem 9:

Part (a):

When we have corresponding points in right image, we can find disparity and then we have:

\[ x_e = x + \frac{b}{2}, \quad x_r = x - \frac{b}{2} \Rightarrow x_e = \frac{F}{z} \left( x + \frac{b}{2} \right), \quad x_r = \frac{F}{z} \left( x - \frac{b}{2} \right) \]

\[ \Rightarrow d = x_e - x_r = \frac{Fb}{z} \Rightarrow \frac{z}{F} = \frac{d}{b} \Rightarrow z = \frac{Fb}{d} \]

\[ y = \frac{y}{F} = \frac{y}{d} \quad x = \frac{x_e + x_r}{2} = \frac{x_e + x_r}{2} \cdot \frac{b}{d} \]

Part (b):

First we find the mapping between \( I_{l,t} \) and \( I_{r,t} \) to find the corresponding points using these corresponding points and result of part (a) we can find \( z^t, x^t, y^t \) position 3D at time \( t \) using the same method we can find the corresponding points on \( I_{l,t+1} \), \( I_{r,t+1} \) and recover \( z^{t+1}, x^{t+1}, y^{t+1} \):

\[ \Delta z = z^t - z^{t+1} \]
\[ \Delta x = x^t - x^{t+1} \]
\[ \Delta y = y^t - y^{t+1} \]

Need to do motion estimation to find the corresponding pixel on frame \( t \) for a point \( \text{say } I_{l,t-1} \) (say \( I_{r,t-1} \)).
10. (10pt) The figure below shows 4 samples over a 2D plane (think of them as 4 pixels where each pixel is described by a feature vector of dimension 2). We would like to cluster them into two groups using the K-means method. Starting with the initial centroids illustrated in the top-left figure, show the results of several iterations of K-means in the figures provided until the iteration converges. You can use a big circle to include all samples in the same cluster in each iteration, and use triangles to indicate the cluster centroids. (Note that the lines in the figure are there to help you gauge the relative positions of the samples and initial centroids.)

- Samples
- Initial centroids