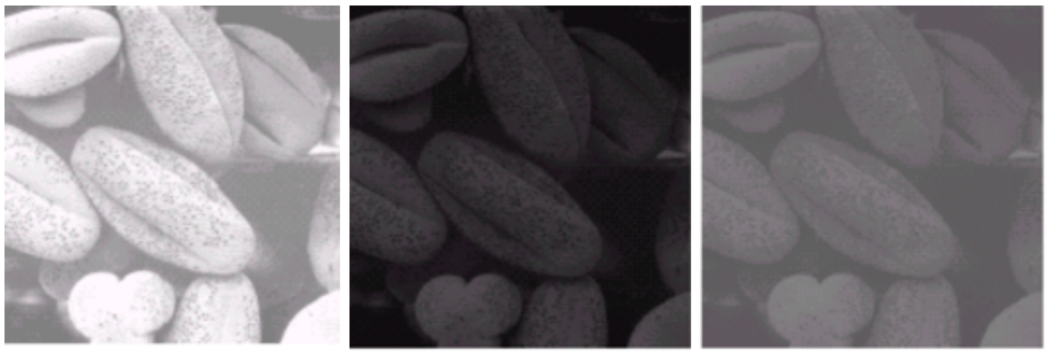
Exam

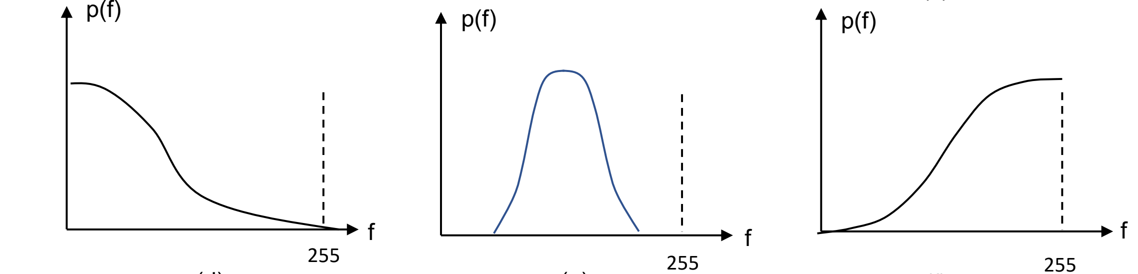
Open book. But **you are not allowed to discuss with any other people**. Failure to observe this rule will result in a grade of Fail. Please start to scan and upload your solution (preferably in a single pdf file) no later than 6:00PM. The submission site will close at 6:15. Please make sure your writing is legible. Make sure that you write your name and ID on your answer sheets.
If you have questions, you can send chat messages via Zoom or call:
Yao Wang 732-939-3399, Jacky Yuan: 718-312-9126, Bolin Liu: 332-201-2787.

Solution

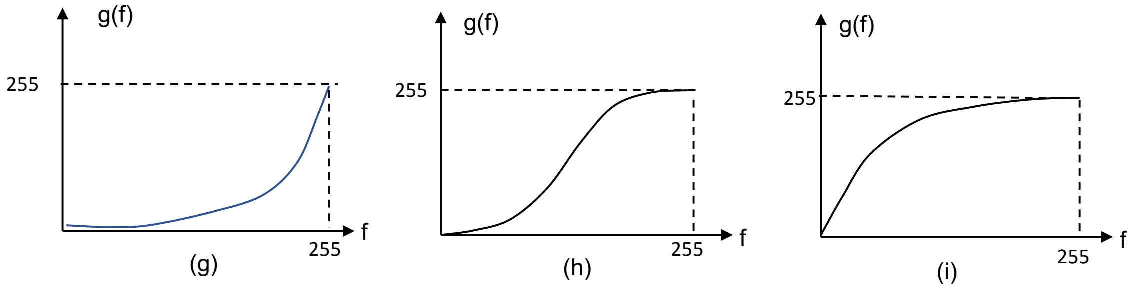
1. (6pt) In the figure below, three images are given in Figs. (a-c), three histograms are given in Figs. (d-f), and three transformations for contrast enhancement are given in Figs. (g-i). For each image, specify which histogram corresponds to this image, and which transformation is best to enhance its contrast.



(a) (b) (c)



(d) (e) (f)



(g) (h) (i)

Solution:
The overall correspondences:

Image	Histogram	Transform
(a)	(f)	(g)
(b)	(d)	(i)
(c)	(e)	(h)

2. (15 pt) Three filters are given below:

$$H_1 = \frac{1}{8}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \qquad H_2 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad H_3 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

(a) (9pt) For each filter, just based on the filter coefficients, can you say what is the function of the filter? Is the filter separable? If yes, please write the corresponding vertical and horizontal filters.

(b) (6pt) For the first filter, derive its discrete space Fourier Transform *H(u,v)*. Please simplify your result so that it is represented as a real function only. Plot *H(u,0)* and *H(0,v)*. Based on the the plot of frequency response, can you tell what is the function of the filter? Is this observation consistent with what you get in part (b)?

Solution:

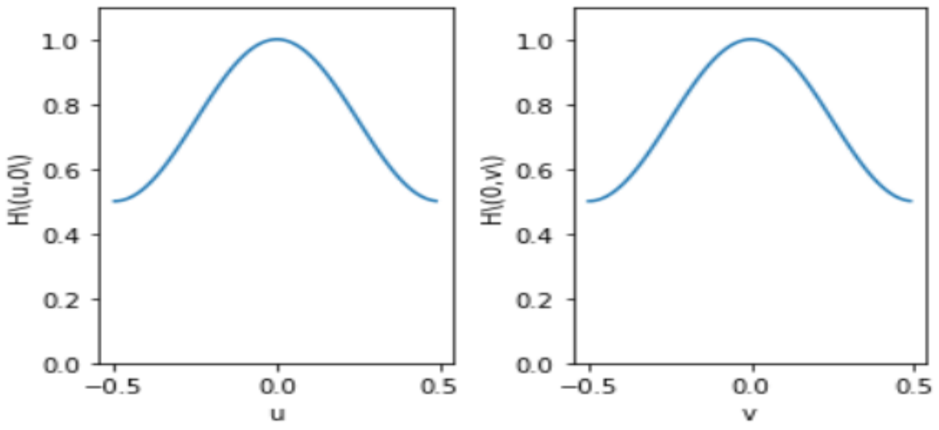
(a) H1 is a low pass filter and is used for smoothing. This is because all coefficients are positive and sum to 1. Not separable

H2 is Separable with $h_x = [1\ 2\ 1]$, $h_y = [-1\ 0\ 1]$. This filter can be used for detecting vertical edge. This is because the vertical filter h_x coefficients are all positive and sum to 1, and hence it is lowpass and performs smoothing vertically, while the horizontal filter h_y coefficients have both positive and negative values and sum to 0, and hence detect the changes. Note that h_y is technically bandpass, but if you said it is high pass, we will not deduct points.

H3 is a high pass filter because the coefficients are both positive and negative coefficients and sum to 0. It is used for detecting isolated points (local extrema) because it takes the difference between the center and all its surrounding pixels. (if you said that it detects edges in all directions, we did not deduct points). Not separable. **Note that this is NOT a high emphasis or sharpening filter. A high emphasis filter will sum to 1, but have both positive and negative coefficients.**

(b) H1:$H(u,v) = \frac{1}{8}\left(4 + e^{j2\pi u} + e^{-j2\pi u} + e^{j2\pi v} + e^{-j2\pi v}\right) = \frac{1}{4}(2 + \cos 2\pi u + \cos 2\pi v)$

$\mathrm{H(u,0)} = \frac{1}{4}(3 + \cos 2\pi u)$; $\mathrm{H(0,v)} = \frac{1}{4}(3 + \cos 2\pi v)$, plots shown below. We can see that it is low-pass in both horizontal and vertical direction. In fact, it is low pass in all directions. This is consistent with the observation of H1 in (a) based on filter coefficients.



Following are not required:

H2: $H(u,v) = \left(2 + e^{j2\pi u} + e^{-j2\pi u}\right)\left(-e^{j2\pi v} + e^{-j2\pi v}\right) = 2(1 + \cos 2\pi u)(-2\mathrm{j}) \sin 2\pi v$

$= -4j(1 + cos2\pi u) \sin 2\pi v$

$\mathrm{H(u,0)} = 0$; $\mathrm{H(0,v)} = -8j \sin 2\pi v$

H3: $H(u,v) = \left(4 - e^{j2\pi u} - e^{-j2\pi u} - e^{j2\pi v} - e^{-j2\pi v}\right) = 2(2 - \cos 2\pi u - \cos 2\pi v)$

$\mathrm{H(u,0)} = 2(1 - \cos 2\pi u)$; $\mathrm{H(0,v)} = 2(1 - \cos 2\pi v)$

3. (10pt) For the image below, determine its 2-level Gaussian Pyramid and Laplacian Pyramid. Use averaging of 2x2 pixels for downsampling, use nearest neighbor replication for interpolation.

(a) (4pt) Determine its 2-level Gaussian Pyramid and Laplacian Pyramid. Use averaging of 2x2 pixels for downsampling, use nearest neighbor replication for interpolation.

(b) (2pt) One way to remove noise from the original image is by assuming the Laplacian image (not the top level image, which is a low-passed image) is sparse and apply soft-thresholding. Show the soft-thresholed Laplacian image with a threshold of 1.

(c) (4pt) Show the final reconstructed image from the soft-thresholded Laplacian pyramid.

7	5	4	3
5	3	3	2
4	3	2	1
3	2	1	0

Solution:

(a) $G_2 = \begin{bmatrix} 7 & 5 & 4 & 3 \\ 5 & 3 & 3 & 2 \\ 4 & 3 & 2 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix}$ Downsampling by averaging every 2x2 samples, we get:

$G_1 = \begin{bmatrix} 5 & 3 \\ 3 & 1 \end{bmatrix}$. Then interpolating by nearest neighbor (which means to replicate each pixel to 2x2), we generate :

$$R_2 = \begin{bmatrix} 5 & 5 & 3 & 3 \\ 5 & 5 & 3 & 3 \\ 3 & 3 & 1 & 1 \\ 3 & 3 & 1 & 1 \end{bmatrix}$$

$L_1 = G_1 = \begin{bmatrix} 5 & 3 \\ 3 & 1 \end{bmatrix}$ $L_2 = G_2 - R_2 = \begin{bmatrix} 2 & 0 & 1 & 0 \\ 0 & -2 & 0 & -1 \\ 1 & 0 & 1 & 0 \\ 0 & -1 & 0 & -1 \end{bmatrix}$

(b) $L_1' = L_1 = \begin{bmatrix} 5 & 3 \\ 3 & 1 \end{bmatrix}$. Applying softthreshold to each element in L2 yields $L_2' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

(c) $R_2' = Up(L_1') = \begin{bmatrix} 5 & 5 & 3 & 3 \\ 5 & 5 & 3 & 3 \\ 3 & 3 & 1 & 1 \\ 3 & 3 & 1 & 1 \end{bmatrix}$ $G_2' = R_2' + L_2' = \begin{bmatrix} 6 & 5 & 3 & 3 \\ 5 & 4 & 3 & 3 \\ 3 & 3 & 1 & 1 \\ 3 & 3 & 1 & 1 \end{bmatrix}$

Note that for this problem, you are asked to only decompose to two levels. Some of you did 3 levels. As long as the solution is correct, no points are deducted.

4. (12pt) An image captured by a camera is generally perturbed by camera noise. If you reorder the image row-by-row into a vector, and denote the original image by a vector x, the captured image by a vector y, you can write y=x+n. One way to remove the noise is by assuming i) The noise has a small L2 norm; and ii) The image is generally smoothly changing except at the edges. This second assumption can be mathematically formulated as requiring the sum of the absolute difference between two horizontally adjacent pixels, denoted by $\|D_h x\|_1$, is small, and similarly, the sum of the absolute difference between two vertically adjacent pixels, denoted by $\|D_v x\|_1$, is small. The second assumption can also be cast as $\|Dx\|_1$ is small, where $D = \begin{bmatrix} D_h \\ D_v \end{bmatrix}$. Under these assumptions, denoising problem can be formulated as an optimization problem as described below:

$$x = \operatorname{argmin} \{ \|x-y\|_2^2 + \lambda \|Dx\|_1 \} \quad (*)$$

(a) (4pt) Formulate the matrix D_h and D_v.
(b) (4pt) Formulate the optimization problem in (*) into a form that can be solved by ADMM (show the original constrained optimization problem and the augmented Lagrangian cost function).
(c) (4pt) Show the ADMM iterations to solve the problem.

Solution:

(a) D_h should be a sparse matrix, and in each row, it should have only two non-zero values in two adjacent elements, 1 and -1, indicating the difference between two adjacent samples in the same row. Note that if you have N samples in each row, you can only take (N-1) differences. So if the image has a dimension of M by N, the matrix D_h has a dimension of M(N-1) by MN. For example, if the image dimension is 3 by 3,

$$D_h = \begin{bmatrix} 1 & -1 & & & & & & & \\ & 1 & -1 & & & & & & \\ & & & 1 & -1 & & & & \\ & & & & 1 & -1 & & & \\ & & & & & & 1 & -1 & \\ & & & & & & & 1 & -1 \end{bmatrix}$$

D_v is also a sparse matrix and in each row, it should have only two non-zero values separated by N elements, 1 and -1, indicating the difference between two pixels in the same column but two adjacent rows. Note that you have N rows, you can only take (M-1) differences. So if the image has a dimension of M by N, the matrix D_v has a dimension of (M-1)N by MN. For example, if the image dimension is 3 by 3,

$$D_v = \begin{bmatrix} 1 & & & & & & -1 & & & & & \\ & 1 & & & & & & -1 & & & & \\ & & 1 & & & & & & -1 & & & \\ & & & 1 & & & & & & -1 & & \\ & & & & 1 & & & & & & -1 & \\ & & & & & 1 & & & & & & -1 \end{bmatrix}$$

Note as long as your solution has 1 and -1 next to each other in D_h and has 1 and -1 separated by N elements in D_v, you get full points.

(b) To solve the problem using ADMM, we introduce another variable $z = Dx$, so that the L1 norm is applied to z. The optimization problem is equivalent to:

$$\text{minimize } \|x-y\|_2^2 + \lambda\|z\|_1$$
$$\text{subject to } Dx - z = 0$$

To solve the problem using ADMM, we first formulate the augmented Lagrangian, where θ represents the Lagrangian multiplier (dual variable):

$$L = \|x-y\|_2^2 + \lambda\|z\|_1 + \theta^T(Dx-z) + \frac{\rho}{2}[\|Dx-z\|_2^2\,]$$

Note that if you chose to add a factor of $\frac{1}{2}$ in front of the term $\|x-y\|_2^2$, we did not deduct points. The solution in (c) would be slightly different, but as long as you did correctly, it is fine.

Note that some of you used y to represent the Lagrangian variable, which is also used to denote the noisy measurement. This is a serious problem and 2pts are taken away. However, if your solution in Part (c) is correct with this wrong formulation, we do not further deduct points.

Some of you changed the measurement variable to b, and then used y to indicate the Lagrangian variable. That is fine.

(c) In each ADMM iteration, we update x, z, and θ sequentially as follows:

x minimization step can be solved by setting:

$$\frac{dL}{dx} = 2(x-y) + D^T\theta + \rho D^T(Dx-z) = 2(x-y) + \rho D^T\left(\frac{1}{\rho}\theta + Dx - z\right) = 0$$

This leads to

$$x^{k+1} = (2I + \rho D^T D)^{-1}(2y + \rho D^T z^k - D^T\theta^k)$$

To solve the z-minimization problem, we can rewrite L :

$$L = \left\|x-y\right\|^2 + \lambda\left\|z\right\|_1 + \frac{\rho}{2}\left[\frac{2}{\rho}\theta^T(Dx-z) + \left\|Dx-z\right\|^2\right]$$

The last part can be rewritten by completing the square, leading to

$$L = \left\|x-y\right\|^2 + \lambda\left\|z\right\|_1 + \frac{\rho}{2}\left\|\frac{1}{\rho}\theta + Dx - z\right\|^2 - \frac{\rho}{2}\left\|\frac{1}{\rho}\theta\right\|^2$$

Ignoring terms which do not depend on z, and rescale by $1/\frac{\rho}{2}$, we get

$$z^{k+1} = argmin\left\{\frac{2\lambda}{\rho}\|z\|_1 + \left\|z - Dx^{k+1} - \frac{\theta^k}{\rho}\right\|^2\right\} = soft(Dx^{k+1} + \frac{\theta^k}{\rho}, \frac{\lambda}{\rho})$$

θ update step is simply:

$$\theta^{k+1} = \theta^k + \rho(Dx^{k+1} - z^{k+1})$$

5. (12pt) You are given two images taken under different view angles. Assume that the two images are related by a bilinear mapping.
   (a) (6pt) Given 4 pairs of corresponding feature point positions: (u_n, v_n) in image 1 and (x_n, y_n) in image 2, *n=1,2,…,4*. How do you determine the bilinear mapping parameters? Formulate a matrix equation to solve the mapping parameters. (You can assume that the matrix you try to invert is non-singular)
   (b) (6pt) Suppose you have detected *N* (N>>4) corresponding pairs of features in these two images, but not all the correspondences are correct. Describe how would you determine the bilinear parameters using the RANSAC method. Write it as an algorithm flow chart

Solution:

(a) Suppose the bilinear mapping of point pair i (i=1,2,3,4) is:

$$\boldsymbol{z_i} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} a_0 + a_1u_i + a_2v_i + a_3u_iv_i \\ b_0 + b_1u_i + b_2v_i + b_3u_iv_i \end{bmatrix} = \begin{bmatrix} 1 & u_i & v_i & u_iv_i & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & u_i & v_i & u_iv_i \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \boldsymbol{A_i c}$$

then we can stack all 4 point pairs equation vertically :

$$\begin{bmatrix} \boldsymbol{z_1} \\ \boldsymbol{z_2} \\ \boldsymbol{z_3} \\ \boldsymbol{z_4} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A_1} \\ \boldsymbol{A_2} \\ \boldsymbol{A_3} \\ \boldsymbol{A_4} \end{bmatrix} \boldsymbol{c} \Rightarrow \boldsymbol{z} = \boldsymbol{Ac} \Rightarrow \boldsymbol{c} = \boldsymbol{A}^{-1}\boldsymbol{z}$$

A more efficient solution is to solve **a** and **b** separately, since the equations for them are independent and you only need to invert a 4x4 matrix. Specifically, you can set up two equations:

$$\begin{bmatrix} 1 & u_1 & v_1 & u_1v_1 \\ & & & \\ & & & \\ 1 & u_4 & v_4 & u_4v_4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \quad or\ \boldsymbol{Ba} = \boldsymbol{x},\ \ \boldsymbol{a} = \boldsymbol{B}^{-1}\boldsymbol{x}$$

$$\begin{bmatrix} 1 & u_1 & v_1 & u_1v_1 \\ & & & \\ & & & \\ 1 & u_4 & v_4 & u_4v_4 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \quad or\ \boldsymbol{Bb} = \boldsymbol{y},\ \ \boldsymbol{b} = \boldsymbol{B}^{-1}\boldsymbol{y}$$

(b) The RANSAC algorithm would consist of the following steps

1. Randomly select (or sample) 4 pairs from all candidate pairs
2. Compute the bilinear mapping parameters X using selected pairs with the method in question(a)
3. Apply resulting mapping to all candidate pairs and evaluate the mapping error (Euclidean distance) at each pair $E_l = \|\boldsymbol{z}_l - h(\mathbf{u}_l, \mathbf{a}, \mathbf{b})\|_2$ if $E_l > T$ (threshold), consider pair *l* outlier.
4. Count the number of inliers for this trial
5. Repeat the process S times
6. Return the mapping with the largest inlier count
7. Refit the mapping using least square fit with all inlier pairs

6. (15pt) Given two images of the same scene taken from two different view angles, we would like to stitch them into a single image based on feature correspondences. Describe the steps (in words) that you would take to accomplish this task. Be sure to include all the steps necessary, and briefly describe the method that you can use in each step.

Solution:
When the two images are taken from different angles of the same scene, the geometric relations between the two images can usually be approximated well by the homography mapping, as long as the observed scene is quite far from the camera so that all the observed points in the scene can be considered to line on a plane. We could fuse the following steps to stitch them together.

1. Detect features in each image (e.g. Harris or SIFT feature or multiscale Harris)
2. Determine a descriptor for each feature point (e.g. SIFT feature or HoG a block surrounding the feature point)
3. For each feature point in image one, find the point in another image with most similar feature descriptor
   - For SIFT descriptor, L2 distance is often used to measure similarity
   - Using Brute force or advance algorithms like K-D tree to get the point with shortest distance.
   - Reject points which have a large distance even with the closest matching point or have two matching points with very similar distance.
4. Determine the homography parameters **a** using RANSAC using the candidate pairs of corresponding features determined from step 3. This is necessary, as some of the candidate pairs may be incorrect. For this step, it is important to know which frame you want to use as a reference frame. Let us say image F is the reference, and you want to warp G to F. Let us say pixels in F are denoted by (u,v), and pixels in G are denoted by (x,y). You would want to use the inverse mapping $\mathbf{x} = \mathbf{h}(\mathbf{u}, \mathbf{a})$, so that for every pixel u in F, you find its corresponding position x in G.
5. Warp G to F' using the homography mapping determined in step 4, $\mathbf{x} = \mathbf{h}(\mathbf{u}, \mathbf{a})$.

6. Blend the image F' and F together. In the area whether the two images overlap, you can use average or weighted average. A more sophisticated approach is Laplacian blending. You should also perform gain compensation if there are contrast difference between F and G.

Some more specifics about the warping and blending steps: (specifics not required, but if you did not mention how to treat overlapping region or gain compensation, -1pt. If you did not mention the direction of the mapping function relative to the reference frame, -1pt)

1. Warp G to G' so that G' aligns with F. Use the transform from 2 as the inverse mapping function
   a) For each point (u, v) in the image G' to be obtained, find its corresponding point (x, y) in the original image G using the mapping function $\mathbf{x} = \mathbf{h}(\mathbf{u}, \mathbf{a})$, and let G'(u, v) = G(x, y)
   b) if the mapped point (x,y) is not an integer sample, interpolate from nearby integer samples in G.

2. Blend G' and F
   a) Determine the size of the stitched image. Ideally, one should find out the appropriate size based on the corresponding positions of the 4 corner points of image G after warping to image F. For simplicity, we can set width = sum of widths of the two images; height=sum of the heights of the two images.
   b) Find area of overlapping between F and G'
   c) Blend in the overlapping area. This can be done by either a simple approach (Average), Distance based weighting or using pyramid representation
   d) If there are contrast difference between F and G, perform gain normalization

7. (15pt) Consider the Lucas-Kanade (LK) method for flow estimation.
   (a) (10pt) Consider two successive frames shown below. Using the Lucas-Kanade (LK) method to determine the optical flow vector of the center pixel. To determine the horizontal and vertical gradient image, you should simply use the difference of two horizontally and vertically adjacent pixels. That is, Ix(m,n)=I(m,n)-I(m,n-1); Iy(m,n)=I(m,n)-I(m-1,n). Your solution should show the horizontal, vertical and temporal gradient image, the moment matrix and the final estimated flow vector. You should use a 3x3 neighborhood block surrounding the center pixel.
   (b) (5pt) Under what conditions, the LK method may fail? Will the EBMA method also fail in those cases?

Frame t

0	0	5	5	5
0	0	5	5	5
5	5	5	5	5
5	5	5	5	5
5	5	5	5	5

Frame t+1

0	0	0	5	5
0	0	0	5	5
0	0	0	5	5
5	5	5	5	5
5	5	5	5	5

Solution:

(a) Horizontal gradient image of $F = F(t)$ $I_x(m, n)$:

0	0	5	0	0
0	0	5	0	0
5	0	0	0	0
5	0	0	0	0
5	0	0	0	0

vertical gradient image of $F = F(t)$ $I_y(m, n)$:

0	0	5	5	5
0	0	0	0	0
5	5	0	0	0

0	0	0	0	0
0	0	0	0	0

The temporal gradient image is $I_t = F(t) - F(t+1)$:

0	0	5	0	0
0	0	5	0	0
5	5	5	0	0
0	0	0	0	0
0	0	0	0	0

The Lukas-Kanade method solves the motion at each pixel based on the spatial and temporal gradient in a small neighborhood using the following equations

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

Using the gradient images determined above and using the values within the 3x3 neighborhood surrounding the center pixel, we obtain

$$\begin{bmatrix} 25 & 0 \\ 0 & 25 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} 25 \\ 25 \end{bmatrix}$$

Which has a solution of $u = -1, v = -1$.

(b) i) LK would fail if the block surrounding a pixel has edge in one direction, because the moment matrix would be non-invertible. But EBMA would still work.

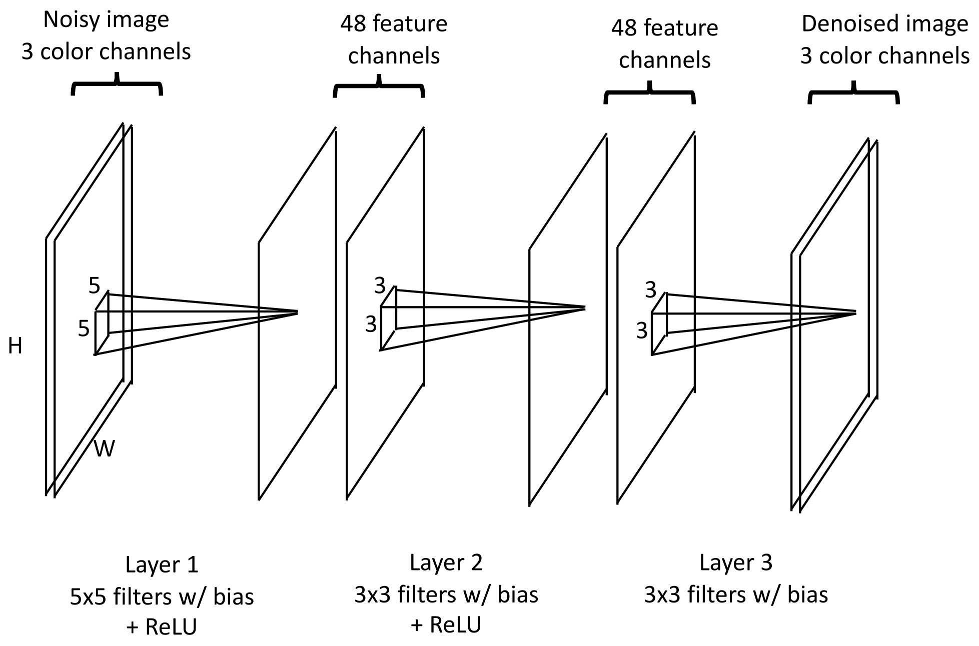
ii) LK would fail is motion is large, but EBMA would still work as long as the motion is within the search range.

iii) Both methods assume all pixels in a block has the same translational motion, and would fail if this assumption does not hold.

iv) Both methods would not work well in flat regions. LK method will have a singular moment matrix, while EBMA can find arbitrary match.

Grading: -1 point if missing any one of the above.

8. (15 pt) Consider the following simple convolutional network architecture used for image denoising
   (a) (3pt) Determine the number of trainable parameters. Write down clearly the number of parameters in each layer, so that you can get partial credits.
   (b) (3pt) What is the perceptive field sizes (relative to the input image) of the filters in each layer?
   (c) (2pt) A simple loss function for denoising is the mean squares error between the ground-truth (noise-free) image and the denoised image. Define the total loss function for all samples in a batch. Use $\hat{y}_{i,k}(m,n)$ to represent the k-th color component of the i-th denoised image, and $y_{i,k}(m,n)$ the corresponding noise-free image.
   (d) (2pt) Let the j-th feature map for the i-th input image after Layer 2 be described by $z_{i,j}(m,n)$. The filters in Layer 3 be denoted by $h_{j,k}(m,n)$ and the biases by b_k, where j is the index of the input feature map, and k is the index of the output color channels. Express the output $\hat{y}_{i,k}(m,n)$ as a function of $z_{i,j}(m,n), h_{j,k}(m,n), b_k$.
   (e) (5pt) Define the gradient of the loss function with respect to the parameters $h_{j,k}(m,n), b_k$.



Solution:

(a) Trainable parameters:
Layer 1: $48 \times (3 \times 5 \times 5 + 1) = 3648$
Layer 2: $48 \times (48 \times 3 \times 3 + 1) = 20784$
Layer 3: $3 \times (48 \times 3 \times 3 + 1) = 1299$
Total : 3648+20784+1299 = 25731

(b) Perceptive field sizes
Layer 1: 5×5
Layer 2: 7×7
Layer 3: 9×9

(c) $L = \text{MSE} = \sum_{i,k,m,n}\left(\hat{y}_{i,k}(m,n) - y_{i,k}(m,n)\right)^2$.
Ideally you should have a scale factor of $\frac{1}{3\,I\,M\,N}$ in front of the sum, where I is the number of images in a batch, M and N are the numbers of rows and columns of each image. 3 indicate 3 color components. Then your solution for the gradients below will have the same scale factor. Your solution would be considered correct regardless whether you added the scale factor.

(d) $\hat{y}_{i,k}(m,n) = \sum_{j=1,..,48,s=-1,0,1,l=-1,0,1} z_{i,j}(m+s,n+l)h_{j,k}(s,l) + b_k$
(OK if you write s=0,1,2; l=0,1,2, or if you do not specify the range of j,s,l)

(e) Using the chain rule

$$\frac{dL}{d\,h_{j,k}(s,l)} = \sum_{i,m,n} \frac{dL}{d\hat{y}_{i,k}(m,n)} \frac{d\hat{y}_{i,k}(m,n)}{dh_{j,k}(s,l)} \text{ and } \frac{dL}{d\,b_k} = \sum_{i,m,n} \frac{dL}{d\hat{y}_{i,k}(m,n)} \frac{d\hat{y}_{i,k}(m,n)}{db_k}$$

in which $\frac{dL}{d\hat{y}_{i,k}(m,n)} = 2\left(\hat{y}_{i,k}(m,n) - y_{i,k}(m,n)\right)$

$$\frac{d\hat{y}_{i,k}(m,n)}{dh_{j,k}(s,l)} = z_{i,j}(m+s,n+l) \text{ and } \frac{d\hat{y}_{i,k}(m,n)}{db_k} = 1$$

therefore we get:

$$\frac{dL}{d\,h_{j,k}(s,l)} = \sum_{i,m,n} 2\left(\hat{y}_{i,k}(m,n) - y_{i,k}(m,n)\right) z_{i,j}(m+s,n+l)$$

$$\frac{dL}{d\,b_k} = \sum_{i,m,n} 2\left(\hat{y}_{i,k}(m,n) - y_{i,k}(m,n)\right)$$