Multiple Choice/Fill in the Blank Questions

For multiple choice questions, circle ALL the correct answers. (Note: Partial marks will not be given for partially correct answers.)

1. (4 points) In Figure 1, grayscale images are in the top row and fourier transform magnitude images are in the bottom row. Write the correct pairing below.

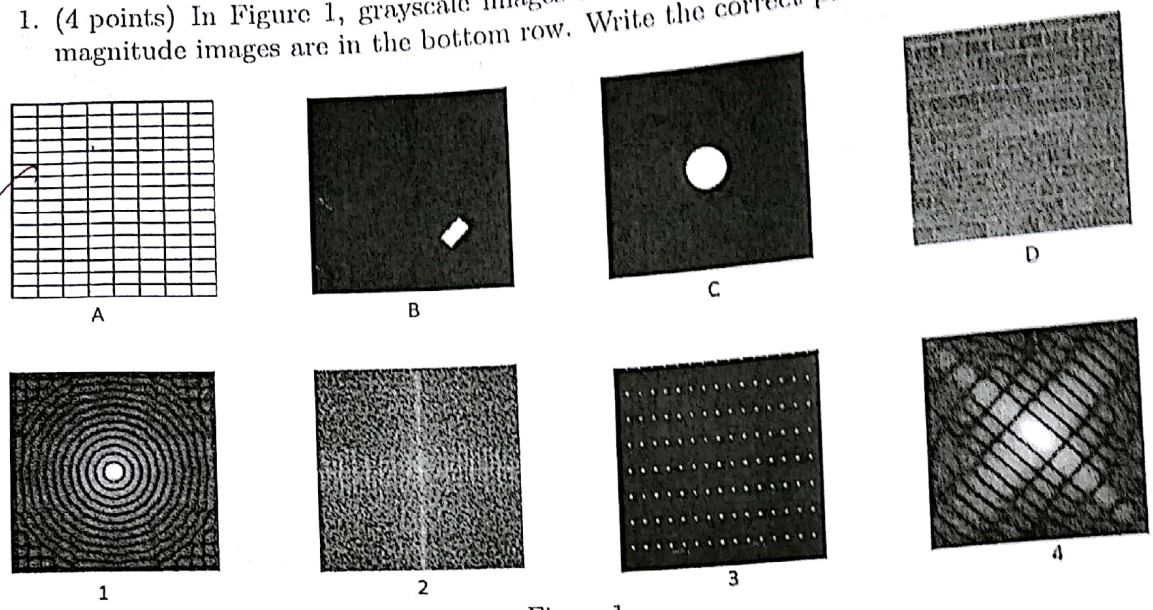


Figure 1

A-3, B-4, C-1, D-2

2. (2 points) The difference between highest and lowest intensity levels in an image is referred to as

(A) Contrast B. Saturation C. Differential D. Brightness

3. (2 points) The effect caused by the using an insufficient number of intensity levels in smooth areas of a digital image is called

A. Contouring B. Interpolation C. Shrinkage (D.) False Contouring

4. (2 points) Which of these image processing operations involves floating point operations ?

A. Filtering a grayscale image with a 3×3 min filter

B. Connected component labeling

C. Huffman coding

(D.) Arithmetic coding

5. (2 points) Suppose I is a 1024×1024 grayscale image, G corresponds to a Gaussian pyramidal representation of I. If I corresponds to the lowest level (Level 0) in G, what does the highest level (1×1) in the pyramid correspond to ?

A. Level 9 (B.) Level 10 C. Level 11 D. None of the previous choices

6. (3 points) Suppose I is an image with k bits per pixel. Suppose J represents the 'negative image' of I. The formula for $J(r, c)$ (intensity at row r and column c) in terms of $I(r, c)$ is $(2^k - 1) - I(r, c)$

7. (4 points) Which of the following image processing operations to the 8-bit grayscale apple image on the left in Figure 2 could result in image on the right in the same figure ? NOTE: The grayscale image is shown with a shadow effect at the base of the apple. Assume that you have access to a binary mask such that the image processing operation mentioned above is performed only on the pixels corresponding to the apple and not on the pixels corresponding to the shadow.

Figure 2

A. Histogram equalization
B. Intensity thresholding
C. Extracting the most significant bit and multiplying the result by 255
D. Negating the image (i.e. subtracting intensity of each pixel in the image from 255)

For the next two questions, Let $f(x, y)$ denote an image and $f_G(x, y)$ denote the image obtained by applying a Gaussian filter $g(x, y)$ to $f(x, y)$. The high-boost filtered version of f can be generated as $f_B(x, y) = af(x, y) - f_G(x, y)$ where $a \geqslant 1$.

8. (3 points) Suppose $J = \mathcal{F} * I$ represents the effect of spatial filter $\mathcal{F}$ convolved on image I, resulting in image J. Which of the following spatial filters corresponds to highboost filtering ? (Note: δ below refers to the impulse function)

A. $\mathcal{F} = a(1 - g(x, y))$ B. $\mathcal{F} = \delta(x, y) - g(x, y)$ C. $\mathcal{F} = a\,\delta(x, y) - g(x, y)$
D. None of the above

9. (2 points) For $a = 1$, highboost filtering is equivalent to unsharp masking. $f_U(x, y) = f(x, y) - f_G(x, y)$. If G is the frequency domain response of g, the frequency response of the unsharp masking is given by

A. $1 + G$ B. $1 - G$ C. $\frac{1}{G}$ D. None of the previous choices

10. (3 points) Suppose we wish to use Hough Transform and identify ellipses in an edge image. Consider an ellipse given by the equation $\frac{(x-c_1)^2}{a} + \frac{(y-c_2)^2}{b} = 1, a > 0, b > 0$. What is the dimensionality of the Hough accumulator ?

A. 2 B. 3 C. 4 D. 6

11. (3 points) We know that a point in (edge) image space corresponds to a line in Hough transform space. What does a parabola $y = ax^2 + bx + c$ in edge image correspond to in the Hough transform space ?

A. Plane B. Point C. Sphere D. Circle

Long Questions

Write detailed answers. Adequately explain your assumptions and thought process.

12. (30 points) Imagine you are a developer in LOOTLAB[1], tasked with writing code for geometric mean averaging filter, `f = GeometricMeanAverage(g,m,n)`, implying that image g is filtered over a $m \times n$ neighborhood, resulting in image f. The formula for geometric mean filter is given in Equation 1, where g is input image and f is the filtered output image. S_{xy} refers to the $m \times n$ neighborhood centered on pixel at location (x, y) (assume m, n are odd).

$$f(x,y) = \left[\prod_{(s,t) \in S_{xy}} g(s,t) \right]^{\frac{1}{mn}} \tag{1}$$

Although your first thought is to write code which loops over the neighborhood indices (s, t), you get a brainwave ! You realize that a better and more efficient approach would be to reuse an already existing LOOTLAB function `f = ArithmeticMeanAverage(g,m,n)` which performs arithmetic mean averaging (Equation 2).

$$f(x,y) = \frac{1}{mn} \sum_{(s,t) \in S_{xy}} g(s,t) \tag{2}$$

1. (20 points) Now write this efficient code for `f = GeometricMeanAverage(g,m,n)`. For simplicity, assume that filtering is done only on 'valid' locations, i.e. filter is centered only on pixels such that filter's spatial extents stay within image boundaries. Hint: If your code involves writing loops, it is not efficient !

2. (10 points) Your code will need to handle certain input image intensity values properly. What is this/are these value(s) ? Make sure this handling is included in the code you write.

13. (30 points) In Figure 3, I is a grayscale image.

1. (5 points) Write code for a function `SumRect(I,a,b,c,d)` which takes as input arguments image I and four row, column indices a, b, c, d (see I in Figure 3) and returns the sum of all the pixel intensities in the rectangle whose top-left corner is (a, c) and bottom-right corner is (b, d).

[1]The founder of company is honest enough to admit that their software is ridiculously expensive !

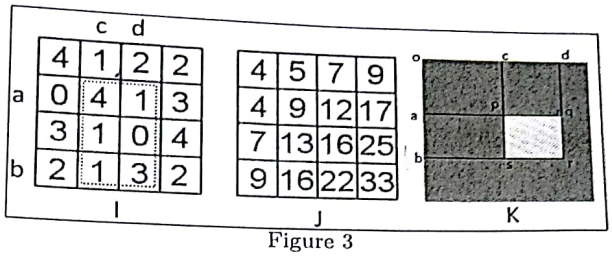


Figure 3

12. (25 points) J, called an integral image, is constructed from I such that $J(p,q) = \sum_{r=p}^{r=1} \sum_{c=q}^{c=1} I(r,c)$. In other words $J(p,q)$ represents the sum of intensities of all the pixels above and to the left of location (p,q) in image I. In Figure 3, for example, 13 in third row and second column of J is obtained from I as $4+1+0+4+3+1$. Suppose you have access to a function `IntgrImage(I)` which computes and returns the integral image of I. Write code which assumes you can call the function `IntgrImage` and write code for a more efficient version of function `SumRect`. Hint: Look at K, situated rightmost in the figure. Area of region $pqrs$ = Area($odrb$) - Area($ocsb$) - Area($odqa$) + Area($ocpa$).

14. (15 points) Histograms of three images are given below in Figure 4. For each image, draw the transformation function $g(f)$ that will help to histogram-equalize the image. NOTE: Don't worry about lack of actual numbers on y-axis in the figures. Full marks will be given as long as the transformation function has the correct shape *and* the function curve changes appropriately at certain key grayscale values on x-axis.

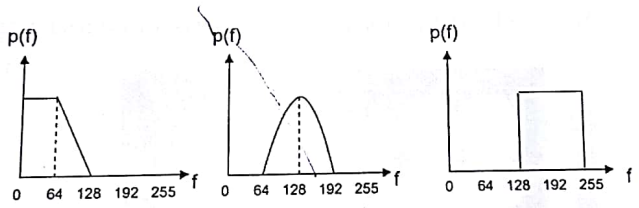


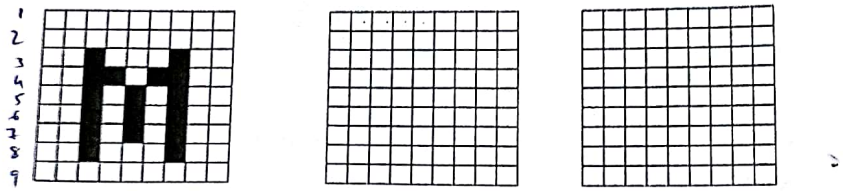
Figure 4

15. (10 points) Suppose you are given an image A. You are asked to generate an image B that is A shifted 4 pixels to the right, 2 pixels up and then rotated by 60 degrees in clockwise direction. Assume A uses image coordinates (u,v) and B uses image coordinate (x,y). NOTE: For the trignometric values, just use their fractional forms, i.e. $sin(\frac{\pi}{3}) = \frac{\sqrt{3}}{2}$. You need not actually calculate the values.

1. (5 points) Write down the forward mapping function from A to B.
2. (5 points) Write down the inverse mapping function from B to A.

16. (20 points) For the 9×9 binary image labeled 'original image' in Figure 5, consider the following compression method. For each new pixel, you predict its value (black or white) using the following method: if its top and left pixels are both white, you predict it to be white. Otherwise, you predict it to be black. You assume the first

row and first column are all white and start your processing only in the second row and second column. You create a prediction error image with 0 representing correct prediction, WB representing the event that white is predicted to be black, and BW representing the event that black is predicted to be white.



Original image Predicted image Prediction error

Figure 5

7. (5 points) In Figure 5, fill out the predicted image and the prediction error image as per above algorithm. NOTE: Fill out these images in your answer booklet and not on the question paper since the grid size is not large enough.

2. (10 points) Create the run-length representation for each row of the prediction error image, which starts with the run length of '0' followed by the non-zero value (WB or BW). The last runlength in a row is followed by EOL. If a row has only zeros, it is represented simply by EOL as well. NOTE: For clarity, write the run-length representation for each row separately.

3. (5 points) Draw the histogram of symbols present in prediction error image.

11. (20 points) Describe an algorithm for removing salt-and-pepper noise in Figure 6.

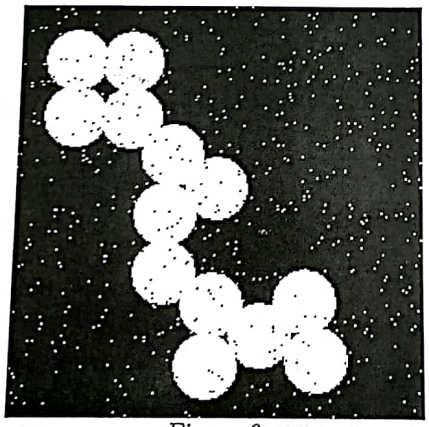
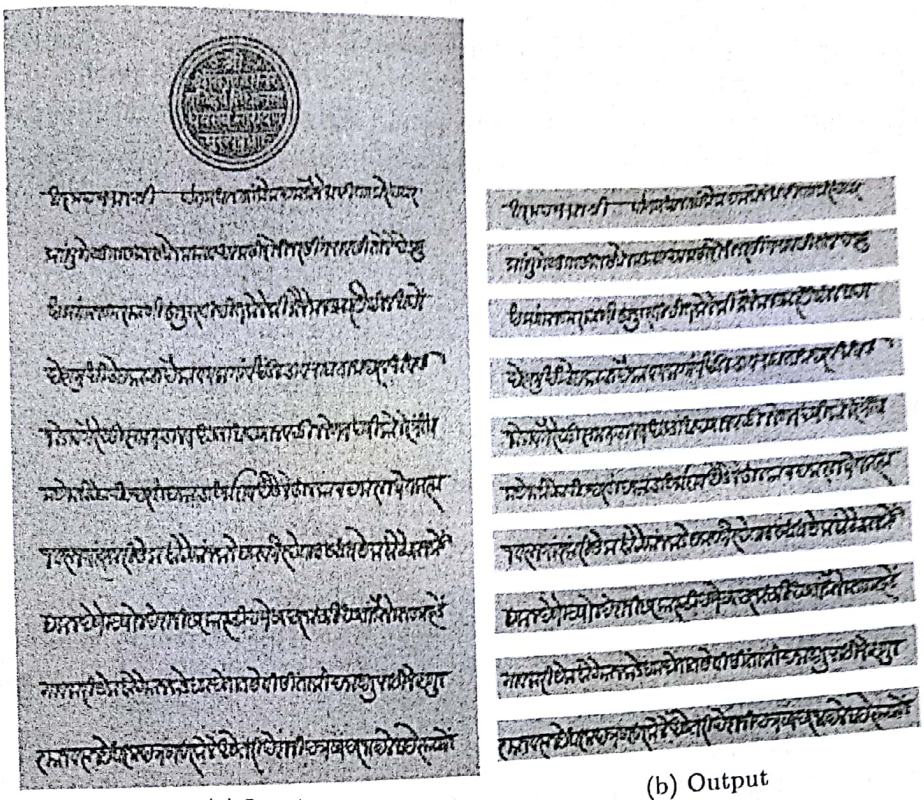


Figure 6

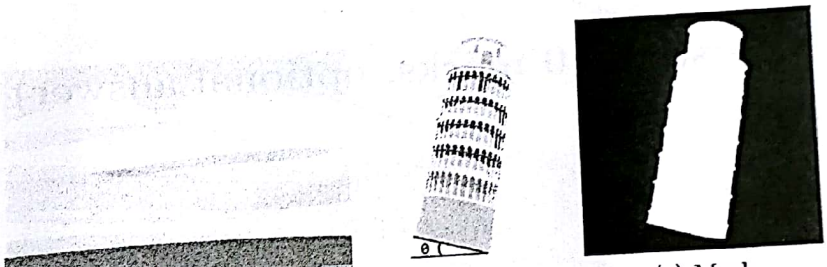
18. (25 points) Describe an algorithm which takes the historical handwritten grayscale document image in Figure 7a and outputs individual line images as in Figure 7b. Outline each step clearly and describe your reasoning.



(a) Input (b) Output

Figure 7

19. (20 points) Consider the background and foreground images shown in Figure 8. You can assume to have access to a mask image corresponding to the foreground. NOTE: The foreground picture shows the tilt of the tower with respect to the horizontal as θ degrees for clarity. Assume that only the tower is present in the foreground. Hopefully, the mask image makes this clear !



(a) Background (H × W) (b) Foreground (1.25H ×0.75W) (c) Mask

Figure 8

1. (5 points) Write a step-by-step procedure to create a composite image by placing the Leaning Tower of Pisa (Foreground (b) in figure) on the given background (Background (a) in figure) with alpha blending. You can clip the parts which are out of background image bounds (height and width).

2. (10 points) We need a composite in which the tower fits within the background image nicely (assume scale factor of s), it is placed in the center of the back-

ground image (assume x-translation of t and the Leaning Tower appears to be straight (assume lean angle to be θ from ground to base). Write the appropriate transform matrix to apply to the foreground image to achieve desired composite. What would be the size of the transformed foreground image?

c. (5 points) Write a procedure to create effect of shadow of the tower on the grass.

20. (20 points) Consider the following pedestrian detection pipeline.

    1. Create an exhaustive list of windows (corner point indices) to test.
    2. Compute Histogram of Oriented Gradients features for each window.
    3. Pass the features for each window to a pedestrian detector.
    4. Combine the detector responses using non-maxima suppression.
    5. Output final detections as indices of windows from the initial list.

    Assume image size to be 500×1000 and detector to be trained on input image of size 50×100. We start with minimum window size of 5×10 and expand the size by 20% in every pass till window size $\leq$ image size. Keeping this system in mind, answer the questions given below.

    1. (10 points) For overlap of 5 pixels in both x and y directions, how many test windows would be there in all passes? Write a formula to find the number of windows for any arbitrary overlap.
    2. (5 points) Write the steps to compute Histogram of Oriented gradients features.
    3. (5 points) Can you re-use any part of the feature computation done for a window A for its adjacent window B for efficiency? If yes, which parts and how?
    4. (5 points) How would you leverage a pyramidal structure to accelerate the detection process?

Bonus Question (0 marks, optional answer)

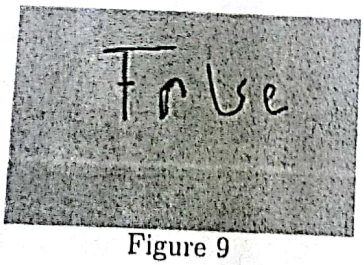
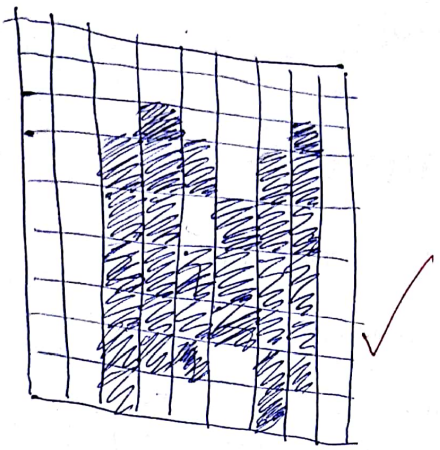


Figure 9

Do you think the person who wrote this intended to write 'True' or 'False' ? On a scale of 0 to 1, what is your confidence that the person wrote 'False' ? What do you think a state-of-the-art optical character recognition system would recognize the text as ?

Q16) i) Representing black by shading the pixel, if pixel is not shaded then it has white color.

1 → white, 0 → Black



⇒

1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 1 0 1 1 1 0 1
1 1 1 0 0 0 1 0 0 1
1 1 0 0 1 0 0 0 1
1 1 0 0 0 0 0 0 1
1 1 0 0 0 0 0 0 1
1 1 0 0 0 1 0 0 1
1 1 0 1 1 1 0 1 1

Prediction error ⇒

0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 BW WB 0 0 BW WB 0
0 0 0 0 WB BW 0 WB 0
0 0 0 WB BW WB 0 WB 0
0 0 0 WB 0 WB 0 WB 0
0 0 0 WB 0 WB 0 WB 0
0 0 0 WB WB 0 0 WB 0
0 0 WB 0 0 0 WB 0 0

given image ⇒

1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 0 1 1 1 0 1 1
1 1 0 0 1 0 0 1 1
1 1 0 1 0 1 0 1 1
1 1 0 1 0 1 0 1 1
1 1 0 1 0 1 0 1 1
1 1 0 1 1 1 0 1 1
1 1 1 1 1 1 1 1 1

(5)

Runlength representation

2.) 1st row: (EOL)

2nd row: (EOL)

3rd row: (2, BW), (0, WB), (2, BW), (0, WB), (1, EOL)

4th row: (4, WB), (0, BW), (1, WB), (1, EOL)

5th row: (3, WB), (0, BW), (0, WB), (1, WB), (1, EOL)

6th row: (3, WB), (1, WB), (1, WB), (1, EOL)

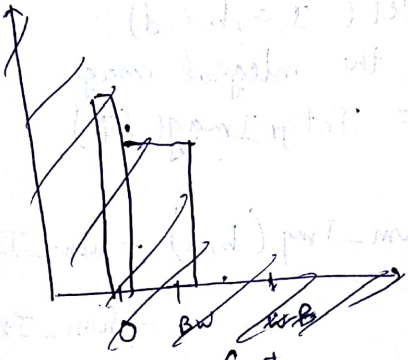
7th row: (3, WB), (1, WB), (1, WB), (1, EOL)

8th row: (3, WB), (0, WB), (2, WB), (1, EOL)

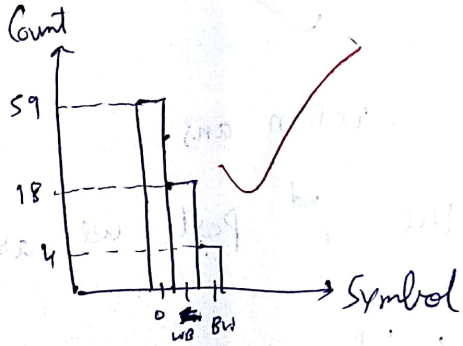
9th row: (2, WB), (3, WB), (2, EOL)

(10)

3.)



Histogram:



(5)

Counts: '0' → 59

'WB' → 18

'BW' → 4

Using MATLAB Syntax

Q13) 1)

```
function ans = SumRect(I, a, b, c, d)
% // Base Cases
if (a > c) || (b > d)
    return 0
end
ans = 0
for i = a : b
    for j = c : d
        ans = ans + I(i, j)
    end
end
return ans
end
```

(5)

→ Iterate over Pixels in the rectangle

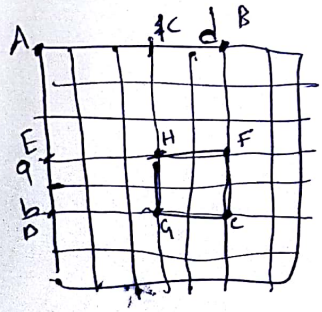
2)

```
function SumRect(I, a, b, c, d):
% Getting the integral image
Sum_Img = IntgrImage(I)
ans = Sum_Img(b, d) - Sum_Img(a-1, d)
      - Sum_Img(b, c-1)
      + Sum_Img(a-1, c-1)
return ans.
```

(22)

Edge cases

→ In the 2nd part we are using the concept of areas :-



Sum_Img

⇒ Sum_Img(b,d) will store the sum of values in the rectangle ABCD

⇒ Sum_Img(a-1,d) will store sum of values in the rectangle AEFB

⇒ Sum_Img(b,c-1) will store sum of values in rect. ACGB

⇒ Sum_Img(a-1,c-1) will store sum of

values in the rect. ACHE

To get sum of values in the rectangle HFGC we can use the formula:

∴ Sum of values in HFGC

= Sum_Img(b,d) + Sum_Img(a-1,c-1) − Sum_Img(b,c-1) − Sum_Img(a-1,d)

Q15) i) forward mapping:

Step 1: $(u,v) \xrightarrow{C \wedge} (u, v+4)$ (shifted 4 pixels to right)

Step 2: Shifted 2 pixels up ⇒ $(u, v+4) \rightarrow (u-2, v+4)$

Step 3: Rotated by 60° clockwise

$$(u-2, v+4) \longrightarrow \big((u-2)\cos(-60°) - (v+4)\sin(-60°),\ (u-2)\sin(-60°) + (v+4)\cos(-60°)\big)$$

4

Using rotation matrix $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

∴ The pixel's value at (u,v) is now mapped to $(x,y) = \left(\frac{u-2}{2} + \frac{(v+4)\sqrt{3}}{2},\ -\frac{(u-2)\sqrt{3}}{2} + \frac{v+4}{2} \right)$

↳ forward mapping

ii) Backward / Inverse mapping:

Step 1: $(x,y) \in B \longrightarrow (x\cos(60°) - y\sin(60°),\ x\sin(60°) + y\cos(60°))$

↳ Rotation by 60° anticlockwise

$$\left(\frac{x}{2} - \frac{\sqrt{3}y}{2},\ \frac{\sqrt{3}x}{2} + \frac{y}{2} \right)$$

Step 2:- Translate 2 pixels down

$$\left(\frac{x-\sqrt{3}y}{2}, \frac{\sqrt{3}x+y}{2}\right) \rightarrow \left(2+\frac{x-\sqrt{3}y}{2}, \frac{\sqrt{3}x+y}{2}\right)$$

Step 3:- Translate 4 pixels to left

(4) $$\left(2+\frac{x-\sqrt{3}y}{2}, \frac{\sqrt{3}x+y}{2}\right) \rightarrow \left(2+\frac{x-\sqrt{3}y}{2}, \frac{\sqrt{3}x+y}{2}-4\right)$$

∴ The pixel value of (x,y) corresponds to

$$(u,v) = \left(2+\frac{x-\sqrt{3}y}{2}, \frac{\sqrt{3}x+y}{2}-4\right)$$

Q17)

Naive Approach:- (Median Filter)

→ Take a window of size 3x3 over each pixel in the image.

→ The output value at the pixels position will be the median the values in the window around the pixel.

(90)

Improved Algorithm (Using Opening):-

→ The drawbacks in naive method are:-

i) The circles lose out their shape i.e they get shaped out.

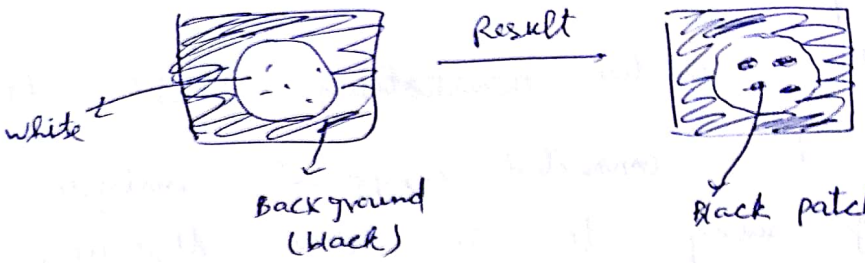
ii) Noise pixels on the boundary / near the boundary will still remain as most of the values in the window around them will be zero.

Steps:- → First we erode the image using a structuring element (disc) of radius greater than the radius of noise pixels. This can be slightly done by getting the maximum radius of a noise pixel. Use a struc. element

(disc) with radius = Max. radius of noise pixel + δ

(δ : small positive quantity)

→ After the erosion the noise pixels will be completly removed. But the image will now have more black patches (inside the circles also).



→ To get back the circles we perform a dilation operation with a disc structuring element which has a radius slightly greater than the previous structuring element (used for dilation).

→ The above step will completly close the black patches. Hence the resulting image will have the circles with the shape preserved and the noise removed.

→ Because of dilation the circles radius may increase by a small quantity which will not look up significantly

∥

Hence both drawbacks of naive algo. are resolved.

Q18) Steps:-

1) We first binarize the image using adaptive thresholding. We cannot use global thresholding as letters change a lot, and each letter has a different variance in a window around it. (from a letter to letter)

↓

Hence adaptive thresholding will give better results

2) We do a connected component analysis on the image using the Two-Pass Algorithm.

3) Once we get the connected components we get combine some of them together i.e for example

H: , $\frac{U}{E}$ → letters of these type will get split into two connected comps.

4) Hence we compare pairs of components and if the distance between components is below a set threshold we label them as same components. Distance can be calculated as we know the label of each pixel.

↳ This will result in H: being a single component.

Circle!

5) Now to get the different connected components as different images we can use image masks.

Ex:- As we know the positions of all pixels with label 1, we can create a mask in

which the positions with pixels of that label have value 1 and rest are zeros.

→ If we multiply this mask with the image we get only one line as the output.

→ Hence we create masks for each component and iteratively multiplying them we get the line images.

→ If we want exactly as shown in the output drop the pixels which are above and below the line. → This can be done as we know the pixels at which the line is present.

Q19) 1.) Steps:-

→ From the mask given create the negative of the mask by using:-

$$Neg\_Mask = 255 - Mask$$

→ Clip the foreground such that it now has the dimensions $H \times 0.75W$. This can be done by copying the required pixels into output image.

→ Similarly clip the mask & Neg_mask also.

→ Now the required composite image will be:-

$$Composite\ image = Mask \times Foreground + Neg\_Mask \times Background$$

Here $\times$ means point-to-point multiplication

→ Mask x Foreground: mask needs to multiplied point to point-

Neg_Mask x Background :- Neg_Mask is placed in the window where we want the tower to be and is multiplied point-to-point.

→ Next images are added such that the pixel values in Foreground x Mask get placed in the window that is specified above (M) blending??

2) Assuming scale along both axes to be S :-

→ Size of transformed foreground image is

$1.25 \times SH \times 0.75 \times SW$

Transformation matrix is :-

$$T = \begin{bmatrix} S\cos\theta & -S(\sin\theta) & t_x \\ S\sin\theta & S(\cos\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix}$$ (&)

↓

Translation required t_x, t_y along x, y axes respectively

→ θ ⇒ Angle the image needs to rotated

→ S → Scale factor

→ This matrix can be used by multiplying with Pixel's homogenous co-ordinate

$$T \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

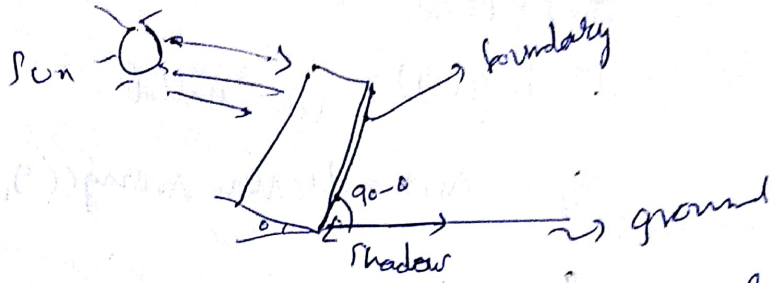
↓ (that is where this pixel will be in transformed image)

This gives resultant pixels position

→ Fill the pixels value in the output pixel

position.

3) To create the shadow we can start the tower's bottom right corner and do a DFS to get to the top right corner. Now we get to know the number of pixels on the boundary.



Now we need color [No. of pixels on boundary × cos(90-θ)]

no. of pixels on the ground to black. This will give the shadow effect.

→ Depending on the sun's position we need to change our defition of boundary.

Sun of left of tower → Bottom-right to Top-right edge

Sun of right of tower → Bottom-left to top-left edge

5

Q20) b) Input Image

↓

Gamma correction and Color space transform

↓

Compute Gradients → Populate the orientation Bins

↑

Normalize the contrast

↑

Combine HOG's from all windows → Combined HOG feature

4

Using the above steps we can get HOG features from each of windows we took on the image. Continued in next page

---

Q12) 1) Algo:-

f = GeometricMeanAverage(g, m, n):

~~g = exp(g)~~ Set zero to Gamma(g) → To handle special cases written below

$g_1 = \log(g)$ → like Matlab

g_2 = ArithmeticMeanAverage(g_1, m, n)

$g_3 = \exp(g_2)$ ✓ (2)

return g_3

→ Here exp, log are functions which ~~are~~ are vectorized to give the exponent and log of each element in the image in very less time

→ Less than running two loops

$$\exp\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = \begin{bmatrix} e^a & e^b \\ e^c & e^d \end{bmatrix}$$ ✓

$$\log\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = \begin{bmatrix} \ln a & \ln b \\ \ln c & \ln d \end{bmatrix}$$

2) Cases:- If input intensity value is 0 then log fn. will not work hence we need to take care of it. ✓ (10)

set zero to Gamma(g)

Matlab syntax ← g(g==0) = ε → small positive quantity

return g.

(Q20) 4.) We can use the pyramidal structure by doing an algo. similar to fast Template Matching.

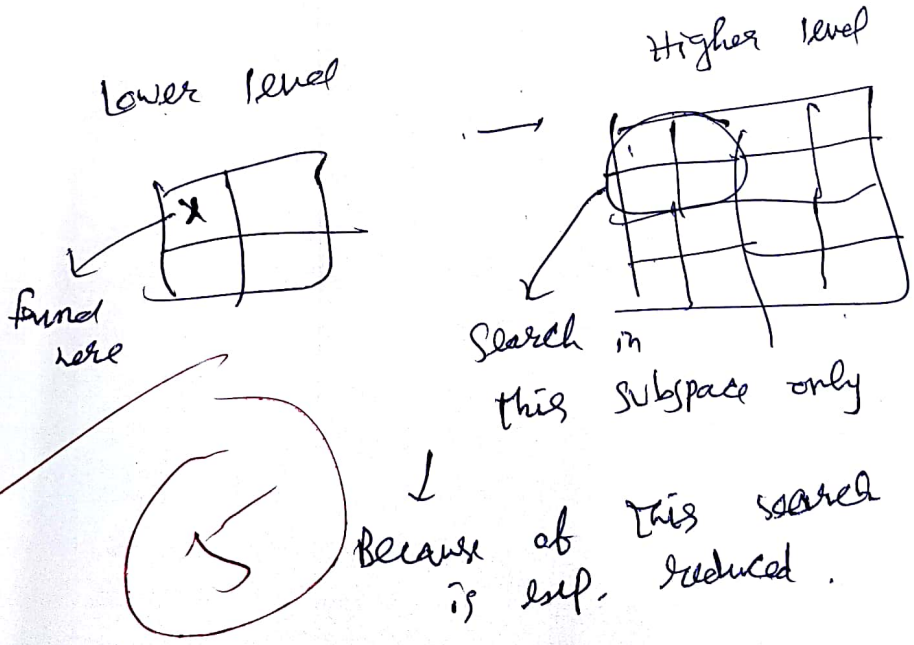
Algo:-

→ Build image pyramid of the input image and the descriptor

→ Search the highest level i.e least size descriptor in the highest level of the image.

→ Now we get a rough estimate of where the descriptor is.

→ Now go to the lower levels and localize the search depending on answer of higher level.



Because of this search time is esp. reduced.