Unit-3 Assignment

PES1PG24CA 330
Thushar

1) Image filtering is the process of modifying or enhancing an image by applying a filter (a matrix @ algorithm) to emphasize certain features @ remove unwanted noise

Low pass filter	High pass filter
→ Smooths the image	→ Enhances edges & fine details
→ Remove high-frequency details like noise & sharp edges	→ Removes low frequency (Smooth / flat) areas.
→ Produces a blurred result	→ Produces a sharper, more detailed result.

2) The Sobel operator uses two 3x3 kernals to compute intensity changes in x(horizontal) & y(vertical) direction

Horizontal gradient (G_x) = $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$

Vertical gradient (G_y) = $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$

Applying the kernel for G_x & G_y, which Combined to detect edge Strength.

3) Canny edge detection first smooths the image with a Guassian filter, then Computes gradient magnitude & direction

Next, it performs non-maximum suppression, applies double thresholds to classify strong/weak edges, and uses edge tracking by hysteresis to keep only Connected true edges.

4) import cv2

```
import numpy as np
img = cv2.imread("imag.jpg", cv2.IMREAD_GRAYSCALE)
gx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3)
gy = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=3)
sobel = cv2.magnitude(gx, gy)
cv2.imshow("Sobel Edge Detection", sobel)
cv2.~~imshow~~ (WaitKey(0)
cv2.destroyAllWindows()
```

5) import cv2

```
img = cv2.imread("image.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray, 100, 200)
cv2.imshow("Canny Edges", edges)
cv2.waitkey(0)
cv2.destroyAllWindows()
```

PES1PG24CA330
Thushar

6) SIFT (Slowest but most accurate):
- -> Works well for scale / rotation changes
- -> Used in robust matching & research

SURF (Faster than SIFT slightly lower accuracy)
- -> Good for real-time-ish application but patented

ORB:
- -> Fastest & free
- -> Good for real-time Systems but less accurate than SIFT/SURF for complex Scenes.

7)
```
import cv2
img = cv2.imread("image.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
orb = cv2.ORB_create()
kp = orb.detect(gray, None)
kp, des = orb.compute(gray, kp)
out = cv2.drawKeypoints(img, kp, None)
cv2.imshow("ORB Keypoints", out)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

PES1PG24CA330

8) Similarity based Segmentation
- → Groups pixel with similar properties (intensity color)
- → Uses methods like region growing, threshold, clustering.
- → produces homogenous regions

Discontinuity - based Segmentation
- → Detect sudden intensity changes
- → Uses edge detectors like Sobel, Canny
- → Focuses on locating regional boundaries

9) Global thresholding:
- → Uses one threshold value for entire image
- → Eg: Otsu's method chooses a single optimal threshold

Adaptive thresholding:
- → Computes different threshold for small regions
- → Eg, Adaptive Mean / Guassian thresholding handles uneven Lighting.

10) Limitations in image segmentation

-> Weak boundaries : Hard to detect edges when object borders are blurry @ low-contrast

-> Noise : Image noise causes false edges

-> Color Variations

-> Occlusion

-> Complex backgrounds : Background textures can mimic object pattern.

-> Class Imbalance = Small object get under-represented.

-> Limited Labeled data

-> Generalization issue.

PE-1PE24CA332 Thushar

1) Region growing Segmentation:

A Method that starts from seed points & add neighboring pixels that have ~~pt~~ similar properties to form regions

prefered over threshold-based methods when:

-> objects have smooth intensity change instead of sharp thresholds.

-> you need spatial continuity

-> images have uneven lighting.

-> objects have share similar intensities.

2) Graph Cuts :

A Segmentation method that models an image as a graph & find the optimal Seperation (cut) between foreground & background by minimizing an energy function.

Application

-> Interactive foreground extension (GrabCut)

-> Medical image Segmentation.

13) Object detection:

→ Finds what objects are present & where they are in each frame / image

→ Output: Bounding boxes + class label.

→ Eg: Detecting Cars, people & traffic lights

Object tracking:

→ What Follows the same object across multiple frame over time

→ Output: Continous object ID + trajectory.

→ Eg: Tracking a Specific car

- Detection = Identify object per frame
- Tracking = Maintain Identity across frames

14) Optimal flow:

It represent the apparent motion of pixel b/w

- Concecutive vedio frames, based on changes in intensity patterns

How it helps in tracking:

- Optical flow computes a motion vector for each pixel
- These vectors show direction + speed of movement
- By following these motion vectors, the System can track objects as they move from one frame to next.

15)
```
import cv2
import numpy as np
img = cv2.imread("input.jpg")
(h,w) = img.shape[:2]
```

PB5194240B330
Jhuthag

```
center = (w//2, h//2)
M = cv2.getRotationMatrix2D(center, 45, 1.0)

cos = abs(M[0,0])
sin = abs(M[0,1])
new_w = int((h*sin) + (w*cos))
new_h = int((h*cos) + (w*sin))

M[0,2] += (new_w/2) - center[0]
M[1,2] += (new_h/2) - center[1]

rotated = cv2.warpAffine(img, M, (new_w, new_h))
cv2.imwrite("rotated_45.jpg", rotated)
```

16)
```
import cv2
import numpy as np
img = cv2.imread("input.jpg")
h,w = img.shape[:2]

#A) Center crop
ch, cw = h//2, w//2
crop = img[ch-100:ch+100, cw-100:cw+100]
cv2.imshow("Crop", crop)

#B) Resize to 300x300.
resized = cv2.resize(img, (300x300))

#C) Rotate without cropping (auto border)

angle = 45
center = (w//2, h//2)
M = cv2.getRotationMatrix2D(center, angle, 1)
cos, sin = abs(M[0,0]), abs(M[0,1])
new_w = int(h*sin + w*cos)
new_h = int(h*cos + w*sin)
```

PE51PG24CA330
Thushar

```
M[0,2] += new_w/2 - center[0]
M[1,2] += new_h/2 - center[1]
rotated = cv2.warpAffine(img, M, (new_w, new_h))

# Translation + scaling:
tx, ty = 50, 30
scale = 1.2
M2 = np.float32([[scale, 0, tx], [0, scale, ty]])
trans_scaled = cv2.warpAffine(img, M2, (w, h))

# Flip:
flip_h = cv2.flip(img, 1)   # horizontal
flip_v = cv2.flip(img, 0)   # Vertical.

# Convert to Grayscale + resize.
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
gray_resized = cv2.resize(gray, (300, 300))
cv2.waitKey(0)
cv2.destroyAllWindows()
```

PES1PG24CA330

Thushar