230962258_Aadiv_CV_Lab3

August 5, 2025

Lab 3

Implementation of Image Filtering Operations

[2]:
```
#generic import statements, pip3 list for all installations

import cv2 as cv
import pandas as pd
import numpy as np
import sklearn as sk
import matplotlib as plt
```

Lab Exercises

1. Write a program to read an image and perform unsharp masking.

[41]:
```
'''
Apply the Gaussian to the entire image and save that as  a new image. Subtract␣
 ↪original image from the Gaussian, and that is our mask.
Add the sharpening mask to the image, and that is our sharpened image.
'''

image = cv.imread('/home/cvl-5aiml-a2/Downloads/bowzer.jpg')

Gaussian_kernel = 0.0625 * np.array(
                  [[1,2,1],
                   [2,4,2],
                   [1,2,1]])

filtered_image = cv.filter2D(src = image, ddepth= -1, kernel= Gaussian_kernel)
sharpening_mask = cv.subtract(image, filtered_image)
final_image = cv.add(image, sharpening_mask)
cv.imshow("sharpeing", final_image)
cv.waitKey(0)
cv.destroyAllWindows()
```

2. Write a program to obtain gradient of an image.

[ ]:
```
'''
Four gradient filters side by side for comparison
```

```
'''

laplacian = cv.Laplacian(src= image, ddepth= -1)
sobelX = cv.Sobel(src= image, ddepth= -1, dx= 0, dy = 1)
sobelY = cv.Sobel(src= image, ddepth= -1, dx= 1, dy = 0)
completeSobel = cv.add(sobelX, sobelY)

stacked_images = cv.hconcat([laplacian, sobelX, sobelY, completeSobel])
cv.imshow("Left: Laplacian, Middle: sobelX, Right, sobelY", stacked_images)
cv.waitKey(0)
cv.destroyAllWindows()
```

3. Write a program to compare box filter and gaussian filter image outputs.

[62]:
```
box_blur = 0.112 * np.array([
[1,1,1],
[1,1,1],
[1,1,1]])

Gaussian_kernel = 0.0625 * np.array(
[[1,2,1],
[2,4,2],
[1,2,1]])


image = cv.imread('/home/cvl-5aiml-a2/Downloads/bowzer.jpg')

filtered_image_box = cv.filter2D(src = image, ddepth= -1, kernel= box_blur)
filtered_image_gaussian = cv.filter2D(src = image, ddepth= -1, kernel=␣
 ↪Gaussian_kernel)
boxGauss = cv.hconcat([filtered_image_box,filtered_image_gaussian])
cv.imshow("Left: Box Blur, Right Gaussian Blur", boxGauss)
cv.waitKey(0)
cv.destroyAllWindows()
```

4. Write a program to detect edges in a image.

[65]:
```
'''
Apply vertical and horizontal kernels and add the values as the final answer.␣
 ↪Trivial.
'''

image = cv.imread('/home/cvl-5aiml-a2/Downloads/bowzer.jpg',0)

h_edge_kernel = np.array(
                        [[1,1,1],
                        [0,0,0],
                        [-1,-1,-1]])
```

```
v_edge_kernel = np.array(
                        [[1,0,-1],
                        [1,0,-1],
                        [1,0,-1]])

filtered_image_v = cv.filter2D(src = image, ddepth= -1, kernel= v_edge_kernel)
filtered_image_h = cv.filter2D(src = image, ddepth= -1, kernel= h_edge_kernel)

final_image = cv.add(filtered_image_h, filtered_image_v)

cv.imshow("edge detection", final_image)
cv.waitKey(0)
cv.destroyAllWindows()
```

5. Implement Canny edge detection algorithm.

[84]:
```
'''
Procedure: (from wikipedia):
The process of Canny edge detection algorithm can be broken down to five␣
 ↪different steps:

    Apply Gaussian filter to smooth the image in order to remove the noise
    Find the intensity gradients of the image
    Apply gradient magnitude thresholding or lower bound cut-off suppression to␣
 ↪get rid of spurious response to edge detection
    Apply double threshold to determine potential edges
    Track edge by hysteresis: Finalize the detection of edges by suppressing␣
 ↪all the other edges that are weak and not connected to strong edges
'''

#Step 1:
image = cv.imread('/home/cvl-5aiml-a2/Downloads/bowzer.jpg',0)
blurred_image = cv.GaussianBlur(image,[7,7],0)

#step 2:
sol_x = cv.Sobel(blurred_image, cv.CV_64F, 1, 0, ksize=3)
sol_y = cv.Sobel(blurred_image, cv.CV_64F, 0, 1, ksize=3)
magnitude = cv.magnitude(sol_x, sol_y)

#step 3 onwards:
misc, thresholded = cv.threshold(magnitude, thresh= 90, maxval= 255, type= cv.
 ↪THRESH_BINARY)
cv.imshow("edge detection", thresholded)
cv.waitKey(0)
cv.destroyAllWindows()
```