Computer Vision

Roll No: AA.SC.P2MCA2107434

CV LAB ASSIGNMENT -4

Morphological Image Processing

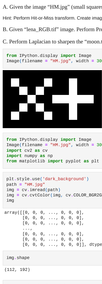
A. Given the image "HM.jpg" (small squares correspond to pixels). Find white pixels, that do not have 4-connected neighboring pixels.

Hint: Perform Hit-or-Miss transform. Create image matrix similar to HM.jpg and then process.

B. Given "lena_RGB.tif" image. Perform Prewitt, Sobel and Canny edge Detection. Compare the results.

C. Perform Laplacian to sharpen the "moon.tif" image.

In [1]:
```
from IPython.display import Image
Image(filename = 'HM.jpg', width = 300, height = 300)
```

Out[1]:


In [2]:
```
from IPython.display import Image
Image(filename = 'HM.jpg', width = 300, height = 300)
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
```

In [3]:
```
plt.style.use('dark_background')
path = 'HM.jpg'
img = cv.imread(path)
img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
img
```

Out[3]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)
```

In [4]:
```
img.shape
```

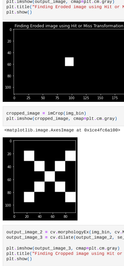
Out[4]:
```
(152, 192)
```

In [5]:
```
# convert into 0s and 1s - binary image
# threshold (image, thresh, maxval_to_use, thresh_type)
img_bin = cv.threshold(img, 127, 1, cv.THRESH_BINARY)[1]
se_15x15 = np.ones((15, 15), np.uint8)
erosion_15x15 = cv.erode(img_bin, se_15x15, iterations=1)
erosion_15x15
```

Out[5]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)
```

In [6]:
```
def imCrop(x):
    height,width = x.shape
    width_cutoff = width // 2
    s1 = x[:, :width_cutoff]
    s2 = x[:, width_cutoff:]
    return s1
```

In [7]:
```
plt.imshow(erosion_15x15, cmap=plt.cm.gray)
plt.title("Eroded Image")
plt.show()
```



In [8]:
```
# Apply hit-or-miss transformation
output_image = cv.morphologyEx(img_bin, cv.MORPH_HITMISS, erosion_15x15)

plt.imshow(output_image, cmap=plt.cm.gray)
plt.title("Finding Eroded image using Hit or Miss Transformation")
plt.show()
```

In [9]:
```
cropped_image = imCrop(img_bin)
plt.imshow(cropped_image, cmap=plt.cm.gray)
```

Out[9]:
```
<matplotlib.image.AxesImage at 0x1ccefc8a100>
```

In [10]:
```
output_image_2 = cv.morphologyEx(img_bin, cv.MORPH_HITMISS, cropped_image)
output_image_2 = cv.dilate(output_image_2, se_15x15, iterations=1)

plt.imshow(output_image_2, cmap=plt.cm.gray)
plt.title("Finding Cropped image using Hit or Miss transformation")
plt.show()
```

B. Given "lena_RGB.tif" image. Perform Prewitt, Sobel and Canny edge Detection. Compare the results.Prewitt Edge Detection

In [11]:
```
# Convert to grayscale
img_gray = cv.imread("lena_RGB.tif", cv.IMREAD_GRAYSCALE)
# Blur the image for better edge detection
img_blur = cv.GaussianBlur(img_gray, (9,9), 0)
```

In [12]:
```
kernelx = np.array([[1,1,1],[0,0,0],[-1,-1,-1]])
kernely = np.array([[1,0,-1],[1,0,-1],[1,0,-1]])
img_prewittx = cv.filter2D(img_gray, -1, kernelx)
img_prewitty = cv.filter2D(img_gray, -1, kernely)
```

In [13]:
```
fig = plt.figure(figsize=(20,20))
plt.subplot(1,4,1)
plt.imshow(img_gray, cmap="gray")
plt.title("Original Image")
plt.axis("off")
```

Out[13]:
```
(-0.5, 511.5, 511.5, -0.5)
```

In [14]:
```
plt.subplot(1,4,2)
plt.imshow(img_prewittx, cmap="gray")
plt.title("Prewitt Edge Detection X")
plt.axis("off")
```

Out[14]:
```
(-0.5, 511.5, 511.5, -0.5)
```

In [15]:
```
plt.subplot(1,4,3)
plt.imshow(img_prewitty, cmap="gray")
plt.title("Prewitt Edge Detection Y")
plt.axis("off")
```

Out[15]:
```
(-0.5, 511.5, 511.5, -0.5)
```

In [16]:
```
plt.subplot(1,4,4)
plt.imshow(img_prewitt, cmap="gray")
plt.title("Prewitt Edge Detection X+Y")
plt.axis("off")
plt.tight_layout()
fig.subplots_adjust(top=0.88)
plt.show()
```


Sobel Edge Detection

In [20]:
```
# Convert to grayscale
img_gray = cv.imread("lena_RGB.tif", cv.IMREAD_GRAYSCALE)
# Blur the image for better edge detection
img_blur = cv.GaussianBlur(img_gray, (19,19), 0)
```

In [24]:
```
# Sobel Edge Detection
def sobel_edge_detection(img_sobel, image_type):
    sobelx = cv.Sobel(src=img_sobel, ddepth=cv.CV_64F, dx=1, dy=0, ksize=3)
    # Sobel Edge Detection on the Y-axis
    sobely = cv.Sobel(src=img_sobel, ddepth=cv.CV_64F, dx=0, dy=1, ksize=3)
    # Sobel Edge Detection on the Y-axis
    sobelxy = cv.Sobel(src=img_sobel, ddepth=cv.CV_64F, dx=1, dy=1, ksize=3)

    #Combined X and Y Sobel Edge Detection
    abs_sobelx = cv.convertScaleAbs(sobelx)
    abs_sobely = cv.convertScaleAbs(sobely)
    abs_sobelxy = abs_sobelx + abs_sobely

    fig = plt.figure(figsize=(20,20))
    plt.subplot(1,4,1)
    plt.imshow(img_sobel,cmap="gray")
    plt.title(image_type+" Image")
    plt.axis("off")

    plt.subplot(1,4,2)
    plt.imshow(abs_sobelx,cmap="gray")

    plt.title("Sobel Edge Gradient in x-direction -- Gx")
    plt.axis("off")

    plt.subplot(1,4,3)
    plt.imshow(abs_sobely,cmap="gray")
    plt.title("Sobel Edge Gradient in y-direction -- Gy")
    plt.axis("off")

    plt.subplot(1,4,4)
    plt.imshow(abs_sobelxy,cmap="gray")
    plt.title("Sobel Edge Combined - Gx + Gy")
    plt.axis("off")

    fig.tight_layout()
    fig.subplots_adjust(top=0.88)
    plt.show()
```

In [25]:
```
sobel_edge_detection(img_gray, "Original")
```


In [26]:
```
sobel_edge_detection(img_blur, "Blurred")
```


Canny Edge Detection

In [27]:
```
# Canny Edge Detection
img_blur = cv.GaussianBlur(img_gray, (15,15), 0)
canny_edges = cv.Canny(image=img_blur, threshold1=10, threshold2=50)
fig = plt.figure(figsize=(20,20))
plt.subplot(1,3,1)
plt.imshow(img_blur, cmap="gray")
plt.title("Blurred Image")

plt.axis("off")

plt.subplot(1,3,2)
plt.imshow(canny_edges, cmap="gray")
plt.axis("off")

plt.title("Canny Edge Detected Image")
plt.show()
```


C. Perform Laplacian to sharpen the "moon.tif" image.

In [28]:
```
path = "moon.tif"
img = cv.imread(path)
img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
```

In [29]:
```
img_blur = cv.GaussianBlur(img_gray, (5,5), 0)
sharpen_filter = np.array([[-1,-1,-1], [-1,9,-1], [-1,-1,-1]])
sharped_img = cv.filter2D(img_blur, -1, sharpen_filter)
```

In [30]:
```
lap_2 = cv.Laplacian(img_blur, cv.CV_64F, ksize=5)
lap_2_abs = np.uint(np.absolute(lap_2))
```

In [32]:
```
fig = plt.figure(figsize=(15,15))
plt.subplot(1,3,1)
plt.imshow(img_blur, cmap="gray")
plt.title("Blurred Image")
plt.axis("off")

plt.subplot(1,3,2)
plt.imshow(sharped_img, cmap="gray")
plt.title("Sharped Image using Filter2D")

plt.axis("off")

plt.subplot(1,3,3)
plt.imshow(lap_2_abs, cmap="gray")
plt.title("Sharped Image After applying Laplacian Filter")
plt.axis("off")

fig.tight_layout()
fig.subplots_adjust(top=0.88)
plt.show()
```


In [ ]:
```

```