Content:

Edge detection

```
Sobel
Laplacian
Canny
```

Morphological operations

```
Erosion
Dilation
```

```
1 import cv2
2 import numpy as np
3 from matplotlib import pyplot as plt
```

```
1 from google.colab import files
2 uploaded=files.upload()
```

Choose Files flower2.jpg

- **flower2.jpg**(image/jpeg) - 36251 bytes, last modified: 6/22/2021 - 100% done

```
Saving flower2.jpg to flower2.jpg
```

```
1 img = cv2.imread('flower2.jpg', cv2.IMREAD_GRAYSCALE)
2
3 from google.colab.patches import cv2_imshow
4
5 img1 = cv2.GaussianBlur(img, (3, 3), 0)
6 plt.subplot(1,2,1),plt.imshow(img,cmap = 'gray')
7 plt.title('Original'), plt.xticks([]), plt.yticks([])
8 plt.subplot(1,2,2),plt.imshow(img1,cmap = 'gray')
9 plt.title('blur'), plt.xticks([]), plt.yticks([])
10
```

```
(Text(0.5, 1.0, 'blur'),
 ([], <a list of 0 Text major ticklabel objects>),
 ([], <a list of 0 Text major ticklabel objects>))
```

Sobel Edge detector

Syntax:cv2.Sobel(original_image,ddepth,xorder,yorder,kernelsize)

the first parameter is the original image,

the second parameter is the depth of the destination image. When ddepth=-1/CV_64F, the destination image will have the same depth as the source.

The third parameter is the order of the derivative x.

The fourth parameter is the order of the derivative y. While calculating Sobelx we will set xorder as 1 and yorder as 0 whereas while calculating Sobely, the case will be reversed.

The last parameter is the size of the extended Sobel kernel; it must be 1, 3, 5, or 7.

```
1 img = cv2.imread('flower2.jpg', cv2.IMREAD_GRAYSCALE)
2 from google.colab.patches import cv2_imshow
3 cv2_imshow(img)
4 sobelx = cv2.Sobel(img, cv2.CV_64F, 1, 0)
5 cv2_imshow(sobelx)
6 sobely = cv2.Sobel(img, cv2.CV_64F, 0, 1)
7 cv2_imshow(sobely)
```

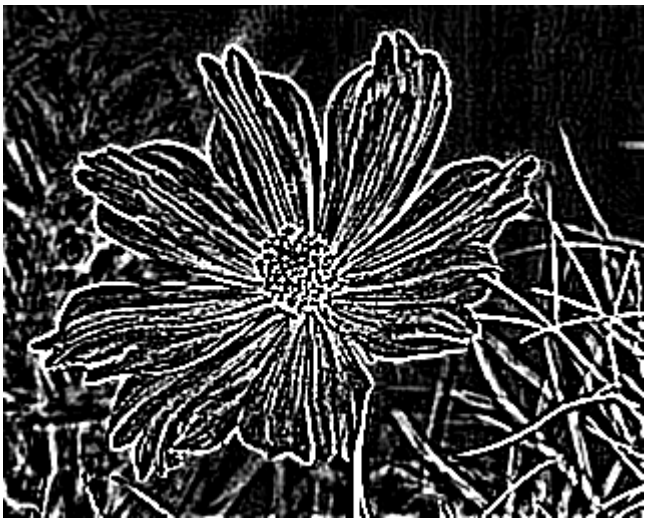
Laplacian edge detector:

Syntax: cv2.Laplacian(frame,cv2.CV_64F)

the first parameter is the original image

the second parameter is the depth of the destination image.When depth=-1/CV_64F, the destination image will have the same depth as the source.

```
1 laplacian = cv2.Laplacian(img, cv2.CV_64F, ksize=5)
2 cv2_imshow(laplacian)
```



Canny Edge Detector

cv2.canny()--

First argument is our input image.

Second and third arguments are our minVal and maxVal respectively.

```
canny = cv2.Canny(img, 120, 170)
cv2_imshow(canny)
```

Morphological Operation

```
from google.colab import files
uploaded=files.upload()
```

Choose Files block.JPG

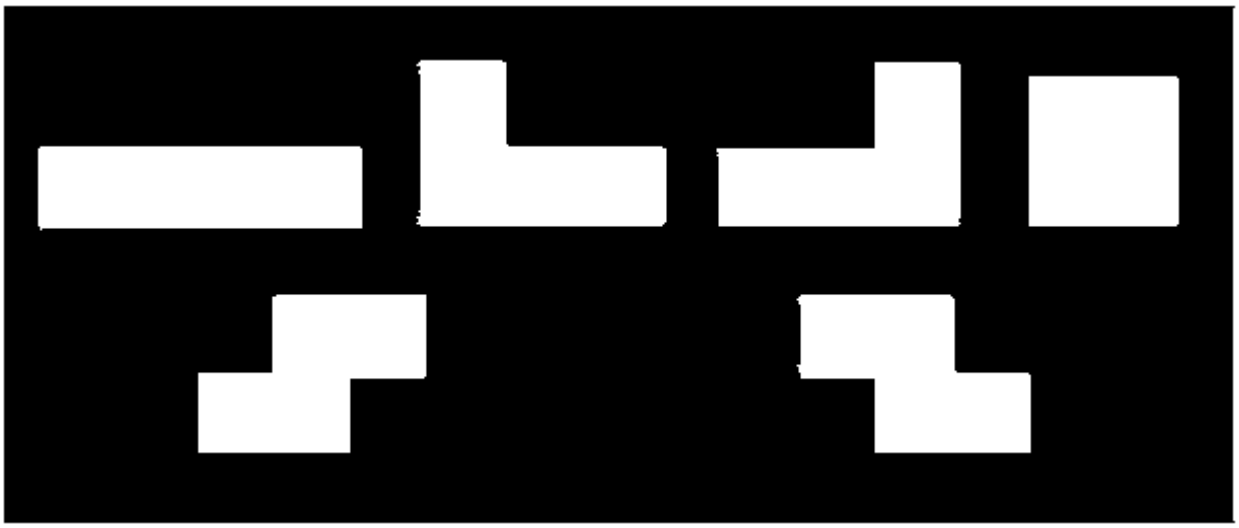
- **block.JPG**(image/jpeg) - 19315 bytes, last modified: 6/24/2021 - 100% done

```
Saving block.JPG to block.JPG
```

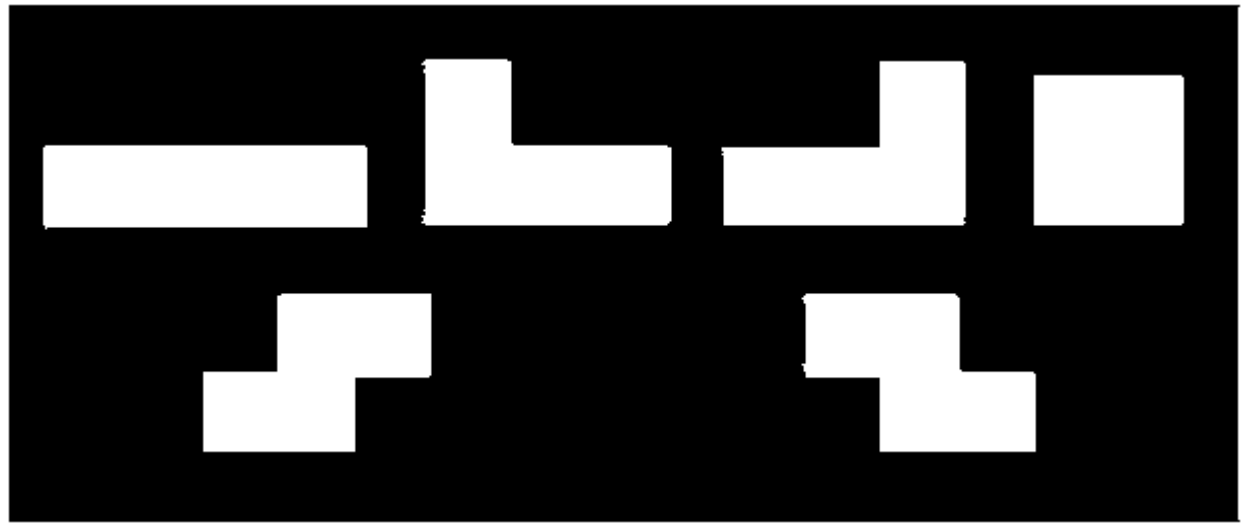
```
# Python program to demonstrate erosion and
# dilation of images.

# Reading the input image
img = cv2.imread('block.JPG', 0)
(height,width)=img.shape

# Taking a matrix of size 5 as the kernel
kernel = np.ones((5,5), np.uint8)

max_intensity = 255
output = img.copy()
for i in np.arange(height):
  for j in np.arange(width):
```

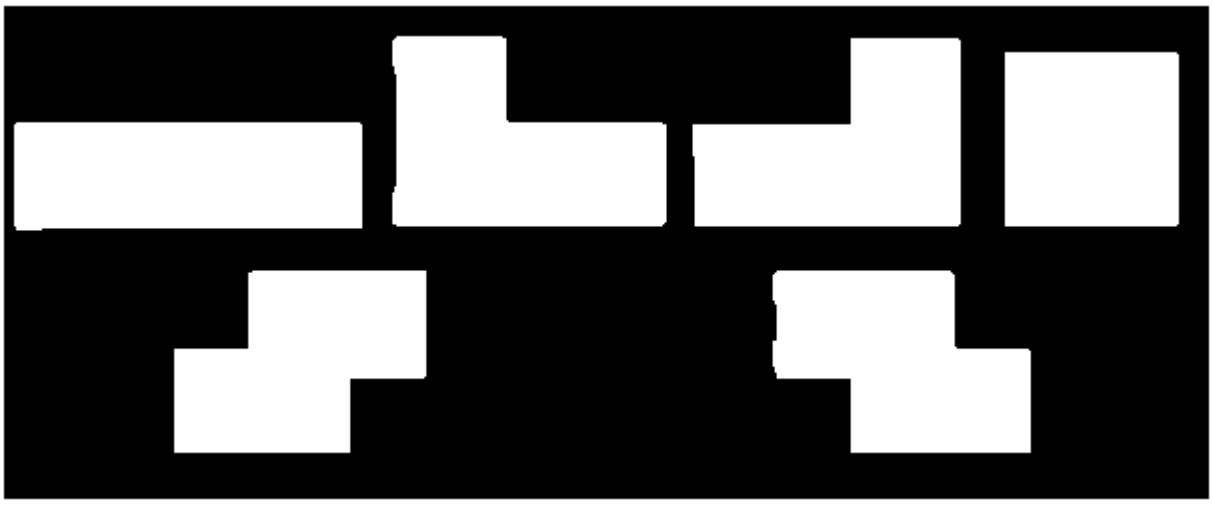
```
    a = output.item(i,j)
    b = max_intensity - a
    output.itemset((i,j), b)

```

```
r,bwimg = cv2.threshold(output, 24, 255, cv2.THRESH_BINARY_INV)
cv2_imshow(bwimg)
```



```
# The first parameter is the original image,
# kernel is the matrix with which image is
# convolved and third parameter is the number
# of iterations, which will determine how much
# you want to erode/dilate a given image.
img_erosion = cv2.erode(bwimg, kernel, iterations=4)
img_dilation = cv2.dilate(bwimg, kernel, iterations=3)

from google.colab.patches import cv2_imshow
cv2_imshow(bwimg)
cv2_imshow(img_erosion)
cv2_imshow(img_dilation)
```





✓ 0s completed at 12:52