Translation

import cv2

import numpy as np

image=cv2.imread("translation.jpg")

height ,width=image.shape[:2]

quarter\_height,quarter\_width=height/4,width/4

T=np.float32([[1,0,quarter\_width],[0,1,quarter\_height]])

img\_translation=cv2.warpAffine(image,T,(width,height))

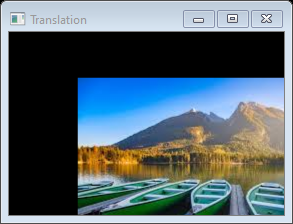
cv2.imshow("original image",image)

cv2.imshow("Translation",img\_translation)

cv2.waitKey()

cv2.destoryAllwindows()





Rotation

import cv2

import numpy as np

image=cv2.imread("rotation.jpg")

height ,width=image.shape[0:2]

rotationMatrix=cv2.getRotationMatrix2D((width/2,height/2),90,.5)

rotationImage=cv2.warpAffine(image,rotationMatrix,(width,height))

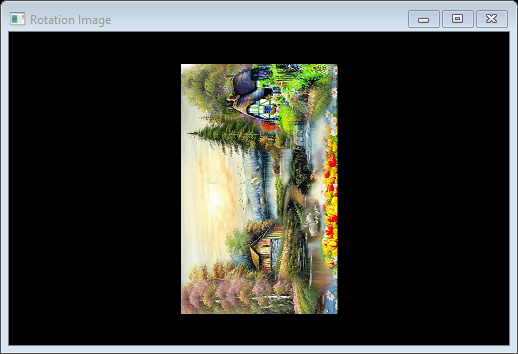
cv2.imshow("original image",image)

cv2.imshow("Rotation Image",rotationImage)

cv2.waitKey(0)

cv2.destoryAllWindows()





cropping

import cv2

import numpy as np

image=cv2.imread("cropping.jpg")

height,width=image.shape[0:2]

startRow=int(height\*.5)

startCol=int(width\*.5)

endRow=int(height\*.75)

endCol=int(width\*.75)

croppingImage=image[startRow:endRow,startCol:endCol]

cv2.imshow("original image",image)

cv2.imshow("Cropping image",croppingImage)

cv2.waitKey(0)

cv2.destoryAllWindows()





Sharping

import cv2

import numpy as np

#reading in and displaying our image

image=cv2.imread('sharp.jpg')

cv2.imshow('Original',image)

#create our sharpening kernel,it must equal to one eventually

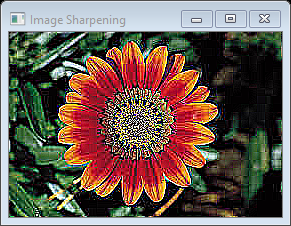
kernel\_sharpening=np.array([[-1,-1,-1],[-1,9,-1],[-1,-1,-1]])

sharpened=cv2.filter2D(image,-1,kernel\_sharpening)

cv2.imshow('Image Sharpening',sharpened)

cv2.waitKey(0)





Sobel and Laplacian Filter using OpenCv.

import cv2

import numpy as np

from matplotlib import pyplot as plt

img=cv2.imread('sharp.jpg',0)

laplacian=cv2.Laplacian(img,cv2.CV\_64F)

sobelx=cv2.Sobel(img,cv2.CV\_64F,1,0,ksize=5)

sobely=cv2.Sobel(img,cv2.CV\_64F,0,1,ksize=5)

plt.subplot(2,2,1),plt.imshow(img,cmap='gray')

plt.title('Original'),plt.xticks([]),plt.yticks([])

plt.subplot(2,2,2),plt.imshow(laplacian,cmap='gray')

plt.title('Laplacian'),plt.xticks([]),plt.yticks([])

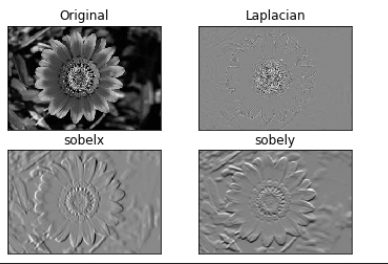
plt.subplot(2,2,3),plt.imshow(sobelx,cmap='gray')

plt.title('sobelx'),plt.xticks([]),plt.yticks([])

plt.subplot(2,2,4),plt.imshow(sobely,cmap='gray')

plt.title('sobely'),plt.xticks([]),plt.yticks([])

plt.show()



Absolute sharpening using OpenCv

import cv2

import numpy as np

from matplotlib import pyplot as plt

img=cv2.imread('Capture.jpg',0)

sobelx=cv2.Sobel(img,cv2.CV\_64F,1,0,ksize=5)

sobely=cv2.Sobel(img,cv2.CV\_64F,0,1,ksize=5)

abs\_sobelx64f=np.absolute(sobelx)

sobelx\_8u=np.uint8(abs\_sobelx64f)

plt.subplot(1,3,1),plt.imshow(img,cmap='gray')

plt.title('Original'),plt.xticks([]),plt.yticks([])

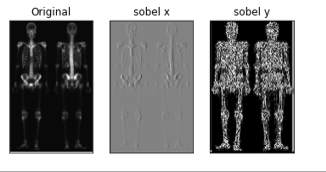
plt.subplot(1,3,2),plt.imshow(sobelx,cmap='gray')

plt.title('sobel x'),plt.xticks([]),plt.yticks([])

plt.subplot(1,3,3),plt.imshow(sobelx\_8u,cmap='gray')

plt.title('sobel y'),plt.xticks([]),plt.yticks([])

plt.show()



Color complement using OpenCv.

import cv2

img = cv2.imread('sharp.jpg')

comp\_image = 255 - img

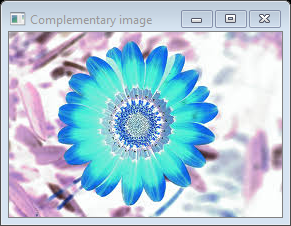
cv2.imshow('original image',img)

cv2.imshow("Complementary image",comp\_image)

cv2.waitKey(0)

cv2.destoryAllWindows()





Color Slicing using OpenCv.

3D Scatter plot of nemo fish using OpenCv.

Smoothing and Sharpening using OpenCv(Use Skeleton Image).