#Blurring

import cv2

import numpy as np

image = cv2.imread('c:\\users\\egek\\ders\\images\\elephant.jpg')

cv2.imshow('Original Image', image)

cv2.waitKey(0)

kernel\_3x3 = np.ones((3, 3), np.float32) / 9

blurred = cv2.filter2D(image, -1, kernel\_3x3)

cv2.imshow('3x3 Kernel Blurring', blurred)

cv2.waitKey(0)

kernel\_7x7 = np.ones((7, 7), np.float32) / 49

blurred2 = cv2.filter2D(image, -1, kernel\_7x7)

cv2.imshow('7x7 Kernel Blurring', blurred2)

cv2.waitKey(0)

cv2.destroyAllWindows()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import cv2

import numpy as np

image = cv2.imread('c:\\users\\egek\\ders\\images\\elephant.jpg')

cv2.imshow('Original Image', image)

blur = cv2.blur(image, (3,3))

cv2.imshow('Averaging', blur)

cv2.waitKey(0)

median = cv2.medianBlur(image, 5)

cv2.imshow('Median Blurring', median)

cv2.waitKey(0)

Gaussian = cv2.GaussianBlur(image, (7,7), 0)

cv2.imshow('Gaussian Blurring', Gaussian)

cv2.waitKey(0)

bilateral = cv2.bilateralFilter(image, 9, 75, 75)

cv2.imshow('Bilateral Blurring', bilateral)

cv2.waitKey(0)

cv2.destroyAllWindows()

#Sharpening

import cv2

import numpy as np

image = cv2.imread('images/input.jpg')

cv2.imshow('Original', image)

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)

cv2.destroyAllWindows()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Thresholding, Binarization & Adaptive Thresholding

import cv2

import numpy as np

# Greyscale

image = cv2.imread('images/soduku.jpg',0)

#gradient.jpg Origin\_of\_Species.jpg sudoku.jpg

cv2.imshow('Original', image)

# Values below 127 goes to 0 (black, everything above goes to 255 (white)

ret,thresh1 = cv2.threshold(image, 127, 255, cv2.THRESH\_BINARY)

cv2.imshow('1 Threshold Binary', thresh1)

# Values below 127 go to 255 and values above 127 go to 0 (reverse of above)

ret,thresh2 = cv2.threshold(image, 127, 255, cv2.THRESH\_BINARY\_INV)

cv2.imshow('2 Threshold Binary Inverse', thresh2)

# Values above 127 are truncated (held) at 127 (the 255 argument is unused)

ret,thresh3 = cv2.threshold(image, 127, 255, cv2.THRESH\_TRUNC)

cv2.imshow('3 THRESH TRUNC', thresh3)

# Values below 127 go to 0, above 127 are unchanged

ret,thresh4 = cv2.threshold(image, 127, 255, cv2.THRESH\_TOZERO)

cv2.imshow('4 THRESH TOZERO', thresh4)

# Resever of above, below 127 is unchanged, above 127 goes to 0

ret,thresh5 = cv2.threshold(image, 127, 255, cv2.THRESH\_TOZERO\_INV)

cv2.imshow('5 THRESH TOZERO INV', thresh5)

cv2.waitKey(0)

cv2.destroyAllWindows()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Dilation, Erosion, Opening and Closing

import cv2

import numpy as np

image = cv2.imread('images/opencv\_inv.png', 0)

cv2.imshow('Original', image)

cv2.waitKey(0)

kernel = np.ones((5,5), np.uint8)

erosion = cv2.erode(image, kernel, iterations = 1)

cv2.imshow('Erosion', erosion)

cv2.waitKey(0)

dilation = cv2.dilate(image, kernel, iterations = 1)

cv2.imshow('Dilation', dilation)

cv2.waitKey(0)

opening = cv2.morphologyEx(image, cv2.MORPH\_OPEN, kernel)

cv2.imshow('Opening', opening)

cv2.waitKey(0)

closing = cv2.morphologyEx(image, cv2.MORPH\_CLOSE, kernel)

cv2.imshow('Closing', closing)

cv2.waitKey(0)

cv2.destroyAllWindows()