FINGERPRINT RECOGNITION

#importing packages

import math

import numpy as np

import cv2 as cv

import matplotlib.pyplot as plt

from utils import \*

from ipywidgets import interact

#reading 1st fingerprint image

fingerprint = cv.imread('samples/sample\_1\_1.png', cv.IMREAD\_GRAYSCALE)

show(fingerprint, f'Fingerprint with size (w,h): {fingerprint.shape[::-1]}')

#reading 2nd fingerprint image

fingerprints = cv.imread('samples/sample\_1\_2.png', cv.IMREAD\_GRAYSCALE)

show(fingerprints, f'Fingerprint with size (w,h): {fingerprints.shape[::-1]}')

# Calculate the local gradient (using Sobel filters)

gx, gy = cv.Sobel(fingerprint, cv.CV\_32F, 1, 0), cv.Sobel(fingerprint, cv.CV\_32F, 0, 1)

show((gx, 'Gx'), (gy, 'Gy'))

gx1, gy1 = cv.Sobel(fingerprints, cv.CV\_32F, 1, 0), cv.Sobel(fingerprints, cv.CV\_32F, 0, 1)

show((gx1, 'Gx'), (gy1, 'Gy'))

# Calculate the magnitude of the gradient for each pixel

gx3, gy3 = gx1\*\*2, gy1\*\*2

gm3 = np.sqrt(gx3 + gy3)

show((gx3, 'Gx\*\*2'), (gy3, 'Gy\*\*2'), (gm, 'Gradient magnitude'))

gx2, gy2 = gx\*\*2, gy\*\*2

gm = np.sqrt(gx2 + gy2)

show((gx2, 'Gx\*\*2'), (gy2, 'Gy\*\*2'), (gm, 'Gradient magnitude'))

# Integral over a square window

sum\_gm3 = cv.boxFilter(gm3, -1, (25, 25), normalize = False)

show(sum\_gm3, 'Integral of the gradient magnitude')

sum\_gm = cv.boxFilter(gm, -1, (25, 25), normalize = False)

show(sum\_gm, 'Integral of the gradient magnitude')

thr = sum\_gm.max() \* 0.2

mask = cv.threshold(sum\_gm, thr, 255, cv.THRESH\_BINARY)[1].astype(np.uint8)

show(fingerprint, mask, cv.merge((mask, fingerprint, fingerprint)))

# Use a simple threshold for segmenting the fingerprint pattern

thr3 = sum\_gm3.max() \* 0.2

mask3 = cv.threshold(sum\_gm3, thr, 255, cv.THRESH\_BINARY)[1].astype(np.uint8)

show(fingerprints, mask, cv.merge((mask, fingerprints, fingerprints)))

#Pointing out the orientation of first finger print image

W = (23, 23)

gxx = cv.boxFilter(gx2, -1, W, normalize = False)

gyy = cv.boxFilter(gy2, -1, W, normalize = False)

gxy = cv.boxFilter(gx \* gy, -1, W, normalize = False)

gxx\_gyy = gxx - gyy

gxy2 = 2 \* gxy

orientations = (cv.phase(gxx\_gyy, -gxy2) + np.pi) / 2 # '-' to adjust for y axis direction

sum\_gxx\_gyy = gxx + gyy

strengths = np.divide(cv.sqrt((gxx\_gyy\*\*2 + gxy2\*\*2)), sum\_gxx\_gyy, out=np.zeros\_like(gxx), where=sum\_gxx\_gyy!=0)

show(draw\_orientations(fingerprint, orientations, strengths, mask, 1, 16), 'Orientation image')

#Pointing out the orientation of second finger print image

W = (23, 23)

gxx = cv.boxFilter(gx3, -1, W, normalize = False)

gyy = cv.boxFilter(gy3, -1, W, normalize = False)

gxy = cv.boxFilter(gx1 \* gy1, -1, W, normalize = False)

gxx\_gyy = gxx - gyy

gxy2 = 2 \* gxy

orientations = (cv.phase(gxx\_gyy, -gxy2) + np.pi) / 2 # '-' to adjust for y axis direction

sum\_gxx\_gyy = gxx + gyy

strengths = np.divide(cv.sqrt((gxx\_gyy\*\*2 + gxy2\*\*2)), sum\_gxx\_gyy, out=np.zeros\_like(gxx), where=sum\_gxx\_gyy!=0)

show(draw\_orientations(fingerprints, orientations, strengths, mask, 1, 16), 'Orientation image')

#checking out the first image with second orientation

W = (23, 23)

gxx = cv.boxFilter(gx3, -1, W, normalize = False)

gyy = cv.boxFilter(gy3, -1, W, normalize = False)

gxy = cv.boxFilter(gx1 \* gy1, -1, W, normalize = False)

gxx\_gyy = gxx - gyy

gxy2 = 2 \* gxy

orientations = (cv.phase(gxx\_gyy, -gxy2) + np.pi) / 2 # '-' to adjust for y axis direction

sum\_gxx\_gyy = gxx + gyy

strengths = np.divide(cv.sqrt((gxx\_gyy\*\*2 + gxy2\*\*2)), sum\_gxx\_gyy, out=np.zeros\_like(gxx), where=sum\_gxx\_gyy!=0)

show(draw\_orientations(fingerprint, orientations, strengths, mask, 1, 16), 'Orientation image')

#fingerprint panaroma

import cv2

image\_paths=['C:/Users/Hp/Pictures/3.png','C:/Users/Hp/Pictures/4.png','C:/Users/Hp/Pictures/5.png' ]

imgs = []

for i in range(len(image\_paths)):

imgs.append(cv2.imread(image\_paths[i]))

imgs[i]=cv2.resize(imgs[i],(0,0),fx=0.4,fy=0.4)

cv2.imshow('1',imgs[0])

cv2.imshow('2',imgs[1])

cv2.imshow('3',imgs[2])

stitchy=cv2.Stitcher.create()

(dummy,output)=stitchy.stitch(imgs)

if dummy != cv2.STITCHER\_OK:

print("stitching ain't successful")

else:

print('Your Panorama is ready!!!')

cv2.imshow('3',output)

cv2.waitKey(0)

OUTPUT

|  |  |
| --- | --- |
| 1ST FINGERPRINT IMAGE | 2ST FINGERPRINT IMAGE |
| SOBEL EDGE DETECTION | SOBEL EDGE DETECTION |
| GRADIENT MAGNITUDE | GRADIENT MAGNITUDE |
|  |  |

|  |
| --- |
| FIRST FRINGERPRINT COMPERED WITH SECOND FINGERPRINT ORIENTATION |

FINGERPRINT PANAROMA

|  |  |  |
| --- | --- | --- |
| 1ST IMAGE | 2ND IMAGE | OUTPUT IMAGE |