import cv2, time, math

import numpy as np

video = cv2.VideoCapture(0)

x = 1

while True:

check, frame = video.read()

frame = frame

frame = cv2.cvtColor(frame,cv2.COLOR\_BGR2GRAY)

og = frame

frame = cv2.GaussianBlur(frame,(5,5),0)

frame = cv2.adaptiveThreshold(frame, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv2.THRESH\_BINARY, 15, 5)

image, contours, hierarchy = cv2.findContours(frame, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

maxA = -1

maxC = None

secA = -1

secC = None

for f in contours:

if cv2.contourArea(f) > maxA:

maxA = cv2.contourArea(f)

maxC = f

elif cv2.contourArea(f) > secA:

secA = cv2.contourArea(f)

secC = f

big = [1]

big[0] = secC

poly = cv2.approxPolyDP(secC, 1, True)

epsilon = 0.025\*cv2.arcLength(secC, True)

poly = cv2.approxPolyDP(secC, epsilon, True)

if len(poly) > 3:

topleft = min(secC, key=lambda x: x[0,0]+x[0,1])

bottomright = max(secC, key=lambda x: x[0,0]+x[0,1])

topright = max(secC, key=lambda x: x[0,0]-x[0,1])

bottomleft = min(secC, key=lambda x: x[0,0]-x[0,1])

corners = (topleft, topright, bottomleft, bottomright)

rect = cv2.minAreaRect(poly)

width = int(rect[1][0])

height = int(rect[1][1])

src\_pts = np.float32([corners[0],corners[1],corners[2],corners[3]])

dst\_pts = np.float32([[0,0],[width,0],[0,height],[width,height]])

per = cv2.getPerspectiveTransform(src\_pts, dst\_pts)

warp = cv2.warpPerspective(frame, per, (width, height))

warp = cv2.resize(warp, (max(width, height), max(width, height)), interpolation=cv2.INTER\_CUBIC)

dist = max(width, height)

sq = round(dist/9)

tiny = warp[0:sq,0:sq]

squares = [None]\*81

a = 0

for x in range(0,9):

for y in range(0,9):

squares[a] = warp[x\*sq:(x+1)\*sq,y\*sq:(y+1)\*sq]

squares[a] = cv2.GaussianBlur(squares[a],(5,5),0)

squares[a] = cv2.adaptiveThreshold(squares[a], 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv2.THRESH\_BINARY, 15, 5)

a += 1

##c\_list = [None]\*81

##for k in range(0,81):

check, frame2 = video.read()

contains = [None]\*81

for g in range(0,81):

image, contours2, hierarchy = cv2.findContours(squares[g], cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

if len(contours2) > 1:

cnts = sorted(contours2, key=cv2.contourArea, reverse=True)

number = cnts[1]

num = []

num.append(number)

##final = cv2.drawContours(frame2, biggest, -1, (0,0,255),1)

minarea = sq\*sq\*0.04

if cv2.contourArea(number) > minarea:

contains[g] = True

cur\_im = squares[g]

rect = cv2.minAreaRect(number)

box = cv2.boxPoints(rect)

box = np.int0(box)

if g == 0:

cv2.drawContours(squares[0],[box],0,(0,0,255),2)

else:

contains[g] = False

final = cv2.drawContours(frame2, contours, -1, (0,0,255),1)

##cv2.fillConvexPoly(final, poly, (0,255,255), 1)

cv2.imshow('final1',frame2)

cv2.imshow('final2',squares[0])

key = cv2.waitKey(1)

if key == ord('x'):

break

video.release()

cv2.destroyAllWindows