|  |
| --- |
| import cv2 |
|  | import numpy as np |
|  | import math |
|  |  |
|  |  |
|  | cap = cv2.VideoCapture(0) |
|  | while(cap.isOpened()): |
|  | # read image |
|  | ret, img = cap.read() |
|  | ret, img = cap.read() |
|  |  |
|  | cv2.rectangle(img, (450,350), (200,130), (0,255,0),5) |
|  | gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY) |
|  | #blur = cv2.GaussianBlur(gray,(5,5),0) |
|  | blur = cv2.GaussianBlur(gray,(35,35),0) |
|  | ret,thresh1 = cv2.threshold(blur,70,255,cv2.THRESH\_BINARY\_INV+cv2.THRESH\_OTSU) |
|  | thresh1 = thresh1[130:350, 200:450] |
|  | cv2.imshow('threshnow1',thresh1) |
|  | image, contours, hierarchy = cv2.findContours(thresh1.copy(), \ |
|  | cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_NONE) |
|  | #cnt = max(contours, key = lambda x: cv2.contourArea(x)) |
|  | #center |
|  | #dist=cv2.distanceTransform(thresh1,cv2.DIST\_L2,5) |
|  | #x, y, w, h = cv2.boundingRect(cnt) |
|  | thresh3=cv2.resize(thresh1,(249,199)) |
|  | cv2.imshow('final blur',blur) |
|  | #cv2.imshow('thresh3',thresh3) |
|  | thresh2=np.array(thresh3) |
|  | #cv2.imshow('thresh2',thresh2) |
|  | \_ ,contours, hierarchy = cv2.findContours(thresh1,cv2.RETR\_TREE,cv2.CHAIN\_APPROX\_NONE) |
|  |  |
|  | dist=cv2.distanceTransform(thresh1,cv2.DIST\_L2,3) |
|  |  |
|  | i=0 |
|  | j=0 |
|  | ii=0 |
|  | jj=0 |
|  |  |
|  | aa = 255\*np.ones((199,249), dtype = np.uint8) |
|  | bb=np.zeros((199,249),dtype=np.uint8) |
|  |  |
|  | thresh1=cv2.resize(thresh1,(249,199)) |
|  | dist=cv2.distanceTransform(thresh1,cv2.DIST\_L2,3) |
|  | i1=0 |
|  | j1=0 |
|  | maxx=0 |
|  | for i in range(199): |
|  | for j in range(249): |
|  | if(dist[i,j]>maxx): |
|  | maxx=dist[i,j] |
|  | for i in range(199): |
|  | for j in range(249): |
|  | if(dist[i,j]==maxx): |
|  | a11=i |
|  | b11=j |
|  | for i in range(199): |
|  | for j in range(249): |
|  | ii=i-a11 |
|  | jj=j-b11 |
|  | if(ii\*ii+jj\*jj<4000): |
|  | aa[i,j]=0 |
|  | #cv2.imshow('aa',aa) |
|  | for i in range(199): |
|  | for j in range(249): |
|  | ii=i-a11 |
|  | jj=j-b11 |
|  | if(ii\*ii+jj\*jj<6500): |
|  | bb[i,j]=255 |
|  | #cv2.imshow('bb',bb) |
|  | a2=np.array(aa) |
|  | #cv2.imshow('cir2',a2) |
|  | cc=cv2.bitwise\_and(a2,thresh2) |
|  | #cv2.imshow('cc',cc) |
|  | cc2=cv2.bitwise\_and(cc,bb) |
|  | cv2.imshow('cc2',cc2) |
|  |  |
|  | \_ ,contours, hierarchy = cv2.findContours(cc2,cv2.RETR\_TREE,cv2.CHAIN\_APPROX\_NONE) |
|  |  |
|  | count\_defects=len(contours)-1 |
|  | #if(result<=5): |
|  | drawing = np.zeros(cc2.shape,np.uint8) |
|  | cv2.drawContours(drawing , contours, 0, (0,255,0), 3) |
|  |  |
|  | if count\_defects == 2: |
|  | cv2.putText(img,"This is two!", (50, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 2, 2) |
|  | elif count\_defects == 3: |
|  | str = "This is three!" |
|  | cv2.putText(img, str, (50, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 2, 2) |
|  | elif count\_defects == 4: |
|  | cv2.putText(img,"This is four!", (50, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 2, 2) |
|  | elif count\_defects == 5: |
|  | cv2.putText(img,"This is five!", (50, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 2, 2) |
|  | elif count\_defects == 1: |
|  | cv2.putText(img,"This is one!", (50, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 2, 2) |
|  | else: |
|  | cv2.putText(img,"This is zero!", (50, 50),\ |
|  | cv2.FONT\_HERSHEY\_SIMPLEX, 2, 2) |
|  |  |
|  | print(count\_defects) |
|  | cv2.imshow('input',img) |
|  | k = cv2.waitKey(10) |
|  | if k == 27: |
|  | break |