Nama : Taufiiqulhakim

Kelas : 5 CF

NIM : 061830701128

PENGOLAHAN CITRA DIGITAL

**UTS**

1. Kode detect\_shapes.py

# USAGE

# python detect\_shapes.py --image shapes\_and\_colors.png

# by Taufiiqulhakim

# import the necessary packages

from pyimagesearch.shapedetector import ShapeDetector

import argparse

import imutils

import cv2

# construct the argument parse and parse the arguments

ap = argparse.ArgumentParser()

ap.add\_argument("-i", "--image", required=True,

    help="path to the input image")

args = vars(ap.parse\_args())

# load the image and resize it to a smaller factor so that

# the shapes can be approximated better

image = cv2.imread(args["image"])

resized = imutils.resize(image, width=300)

ratio = image.shape[0] / float(resized.shape[0])

# convert the resized image to grayscale, blur it slightly,

# and threshold it

gray = cv2.cvtColor(resized, cv2.COLOR\_BGR2GRAY)

blurred = cv2.GaussianBlur(gray, (5, 5), 0)

thresh = cv2.threshold(blurred, 60, 255, cv2.THRESH\_BINARY)[1]

# find contours in the thresholded image and initialize the

# shape detector

cnts = cv2.findContours(thresh.copy(), cv2.RETR\_EXTERNAL,

    cv2.CHAIN\_APPROX\_SIMPLE)

cnts = imutils.grab\_contours(cnts)

sd = ShapeDetector()

# loop over the contours

for c in cnts:

    # compute the center of the contour, then detect the name of the

    # shape using only the contour

    M = cv2.moments(c)

    cX = int((M["m10"] / M["m00"]) \* ratio)

    cY = int((M["m01"] / M["m00"]) \* ratio)

    shape = sd.detect(c)

    # multiply the contour (x, y)-coordinates by the resize ratio,

    # then draw the contours and the name of the shape on the image

    c = c.astype("float")

    c \*= ratio

    c = c.astype("int")

    cv2.drawContours(image, [c], -1, (0, 255, 0), 2)

    cv2.putText(image, shape, (cX, cY), cv2.FONT\_HERSHEY\_SIMPLEX,

        0.5, (255, 255, 255), 2)

    # show the output image

    cv2.imshow("Image", image)

    cv2.waitKey(0)

2. Kode shapedetector.py

# import the necessary packages

import cv2

class ShapeDetector:

    def \_\_init\_\_(self):

        pass

    def detect(self, c):

        # initialize the shape name and approximate the contour

        shape = "unidentified"

        peri = cv2.arcLength(c, True)

        approx = cv2.approxPolyDP(c, 0.04 \* peri, True)

        # if the shape is a triangle, it will have 3 vertices

        if len(approx) == 3:

            shape = "segitiga"

        # if the shape has 4 vertices, it is either a square or

        # a rectangle

        elif len(approx) == 4:

            # compute the bounding box of the contour and use the

            # bounding box to compute the aspect ratio

            (x, y, w, h) = cv2.boundingRect(approx)

            ar = w / float(h)

            # a square will have an aspect ratio that is approximately

            # equal to one, otherwise, the shape is a rectangle

            shape = "segiempat" if ar >= 0.95 and ar <= 1.05 else "persegipanjang"

        # if the shape is a pentagon, it will have 5 vertices

        elif len(approx) == 5:

            shape = "segilima"

        # otherwise, we assume the shape is a circle

        else:

            shape = "lingkaran"

        # return the name of the shape

        return shape

3. Hasil Running

































