Computer Vision
Degree in Information Technology
2º Semester 2021/2022

Worksheet 6

Goals:

- Object detection

Exercises

Part I – Shape detection in images and videos

1- Using the file shape.jpg and adopting computer vision techniques, identify the multiples shapes in the image. Figure 1 illustrates a possible output.

a. Read the image
b. Convert image to gray
c. Apply the blurring method
d. Apply Canny method from
e. Apply get_contours method
f. Apply draw_contour function. Analyse the code.
g. Instead of Canny, try with another segmentation algorithm (*e.g.*, threshold) and see the differences. Adopt your favourite in the next exercises.

```
def draw_contour(image, contours):
    for contour in contours:
        #cv.drawContours(image, contour, -1, (255,0,0), 1)
        perimeter = cv.arcLength(contour, True)
        approx = cv.approxPolyDP(contour, 0.02*perimeter, True)
        connors = len(approx)
        area = cv.contourArea(contour)
        x1, y1, x2, y2 = cv.boundingRect(approx)
        if connors > 2 and area > 500:
            cv.rectangle(image, (x1, y1), (x1+x2, y1+y2), (0,255,0),
thickness=2)
```

h. Do the show image including the number of shapes founded.

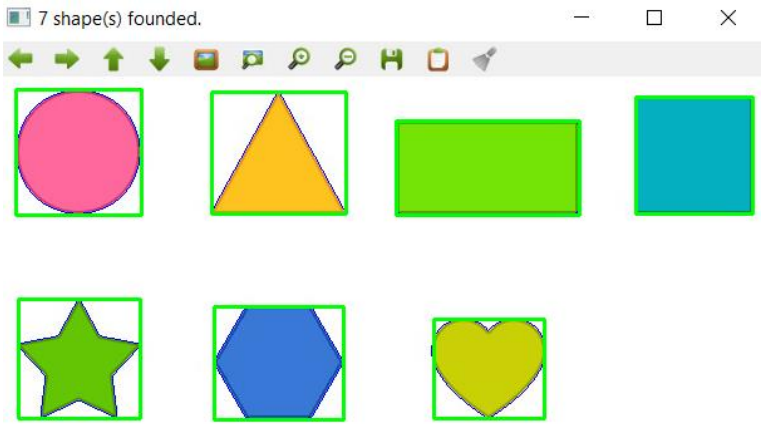


Figure 1 - Shape recognition

2- Experiment the previous code using the video file objects.mp4.

```
video = cv.VideoCapture("objects.mp4")
while True:
    success, frame = video.read()

    if success:
        height, width = frame.shape[:2]
        frame = cv.resize(frame, (width - (width * 65 // 100),
height - (height * 65 // 100)))
        frame_copy = frame.copy()
        continue …
```

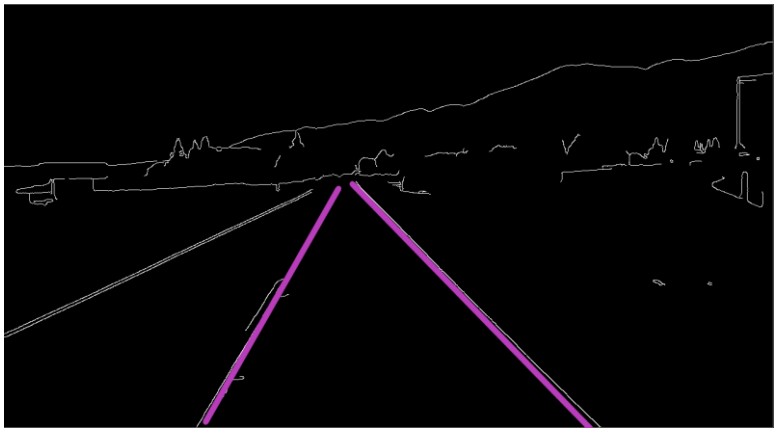
a. Use now the camera from your laptop. For that, add 0 as the parameter of VideoCapture - `cv.VideoCapture(0)`

Part II – Use case: Lane detector for self-driving cars

1. Watch the video

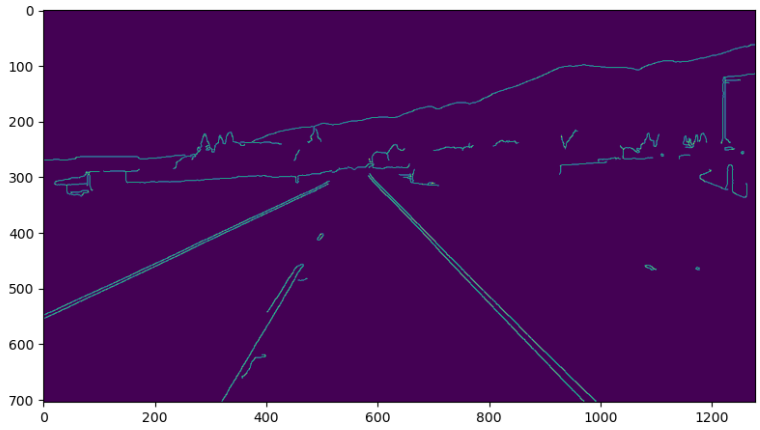
2. Using the computer vision techniques, try to detect lanes following the next steps:

   a. Start with the image lane.jpg
   b. Load and display the original image
   c. Apply: gray-scaling, noise-reduction and a segmentation or edge detection method
   d. Isolate the region of interest



e. To clarify the exact position of our region of interest, use matplotlib library to spot the coordinates and isolate that region.

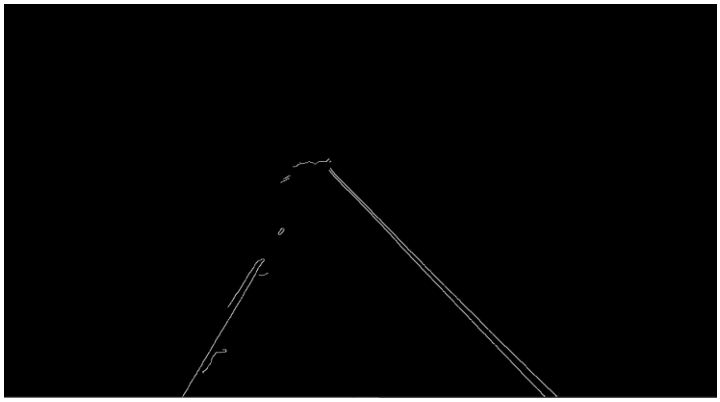
```
plt.imshow(image)
plt.show()
```



f. Define a function which will create a mask for the region of interest. As starting point, consider the next piece of code. Analyse the coordinates of the region of interest.

```
def region_of_interest(image):
    height = image.shape[0]
    polygons = np.array([
    [(200, height), (1100, height), (550, 300)]
    ])
    mask = np.zeros_like(image)
    cv2.fillPoly(mask, polygons, 255)
    return mask
```

g. Use a logic operation with the mask and the original image to only show the region of interest.



h. Apply hough transform to detect the lanes

i. Draw the lane line of the region of interest detected by the hough transform in the original image

j. Download the video. Apply the same steps in the video to detect the lanes