
# پروژه سوم درس سامانه های چندرسانه ای

آنوشا شریعتی ۹۹۲۳۰۴۱
مهشاد اکبری سریزدی ۹۹۲۳۰۹۳

## سوال ۱ :

در این سوال قصد داریم تا اشکال را تشخیص دهیم . ابتدا تصویر مد نظرمان را لود کرده . حال با توجه به موضوع تشخیص اشکال از کانتور استفاده می کنیم تا اشکال مختلف را در تصویر تشخیص دهد . میدانیم قبل از اینکه بخواهیم کانتور ها را تشخیص دهیم باید ابتدا تصویر را سیاه سفید کنیم و سپس با استفاده از بلور کردن نویز تصویر را بگیریم و در نهایت با استفاده از threshold تصویر را باینری می کنیم . حال تصویر برای پیدا کردن کانتور ها آماده سازی شده است .

```
# Load the image
image_path = '/Users/digitcrom/Desktop/multi project /Multimedia_HW3/Shapes.jpg'
image = cv2.imread(image_path)
org_image = image.copy()
image_2 = image.copy()
cv2.imshow('Original Image', org_image)
cv2.waitKey(0) #wait until key pressed

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
#blurred = cv2.medianBlur(gray, 5)
blurred = cv2.GaussianBlur(gray, (5, 5), 0) # kernel size = 5 , sigma_x = 0 open cv will calculate it
#blurred = cv2.bilateralFilter(gray, d=9, sigmaColor=75, sigmaSpace=75)
#blurred = cv2.fastNlMeansDenoising(gray, None, h=10, templateWindowSize=7, searchWindowSize=21)
_, threshold = cv2.threshold(blurred, 60, 255, cv2.THRESH_BINARY)
```

حال با استفاده از دستور findcontour کانتور های تصویر را می یابیم . سپس یک تابع برای تشخیص اشکال طراحی می کنیم . در این جا approx نشان‌دهنده رأس‌های چندضلعی است که بهترین تقریب کانتور ورودی را نشان می‌دهد و peri درواقع محیط کانتور را نشان می دهد . با استفاده از دستور len(approx) می توانیم تعداد راس های هر کانتور را بیابیم و درواقع بر اساس تعداد راس ها نوع شکل کانتور را تعیین می کنیم . از آنجایی که تصویر شامل مربع و مستطیل می باشد و هر دو دارای ۴ راس هستند برای تمایز آن ها با استفاده از کشیدن یک مستطیل به دور کانتور آن ها ' عرض و طول را میتوانیم به دست بیاوریم ' اگر با تقریب خوبی نسبت عرض به طول یک باشد میدانیم شکل مربع است و در غیر این صورت شکل مستطیل می باشد . همچنین در نظر گرفتیم اگر شکل بیشتر از ۶ راس داشته باشد دایره می باشد .

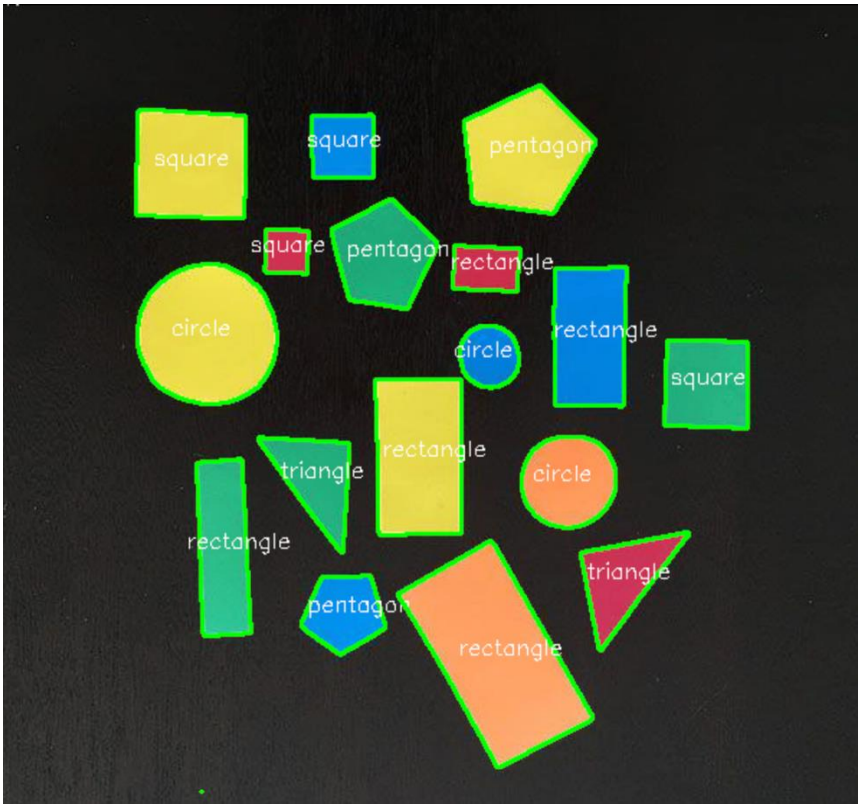
```python
contours, _ = cv2.findContours(threshold, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

def detect_shape(c):
    shape = ""
    peri = cv2.arcLength(c, True) #closed contours
    approx = cv2.approxPolyDP(c, 0.04 * peri, True)

    if len(approx) == 3:
        shape = "triangle"
    elif len(approx) == 4:
        (x, y, w, h) = cv2.boundingRect(approx)
        ar = w / float(h)
        shape = "square" if ar >= 0.95 and ar <= 1.05 else "rectangle"
    elif len(approx) == 5:
        shape = "pentagon"
    elif len(approx) == 6:
        shape = "hexagon"
    else:
        shape = "circle"

    return shape, len(approx) , approx
```

حال نوع هر شکل در تصویر تشخیص داده شده است و قصد داریم نوع هر کدام را بر روی آن بنویسیم .
برای این کار با استفاده از یک حلقه for تمامی کانتور های پیدا شده را طی می کنیم . با استفاده از مومنتوم ها میتوانیم مراکز هر کانتور (هر شکل ) را مشخص کنیم که به صورت زیر مختصات x و y مربوط به مرکز هر کانتور را به دست آوریم .
با استقاده از تابع detect_shape که در بالا توضیح دادیم شکل هر کدام را می دانیم و با استفاده از دستور putText بر روی تصویر اصلی کانتور و نام هر شکل را در مرکز آن ( در راستای محور x برای اینکه نام ها روی تصویر دچار تداخل نشوند کمی عقب تر از مرکز شروع به نوشتن نام می کند) نوشتیم

```python
for contour in contours:
    M = cv2.moments(contour)
    if M["m00"] != 0:
        cX = int((M["m10"] / M["m00"]))
        cY = int((M["m01"] / M["m00"]))
    else:
        cX, cY = 0, 0
    shape, vertices , approx = detect_shape(contour)
    cv2.drawContours(org_image, [contour], -1, (0, 255, 0), 2)
    cv2.putText(org_image, shape, (cX-25, cY), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 1)
```

.

تصویر حاصل از این فرآیند به صورت زیر می باشد ' که مشاهده می شود شکل ها به درستی تشخیص داده شده اند .



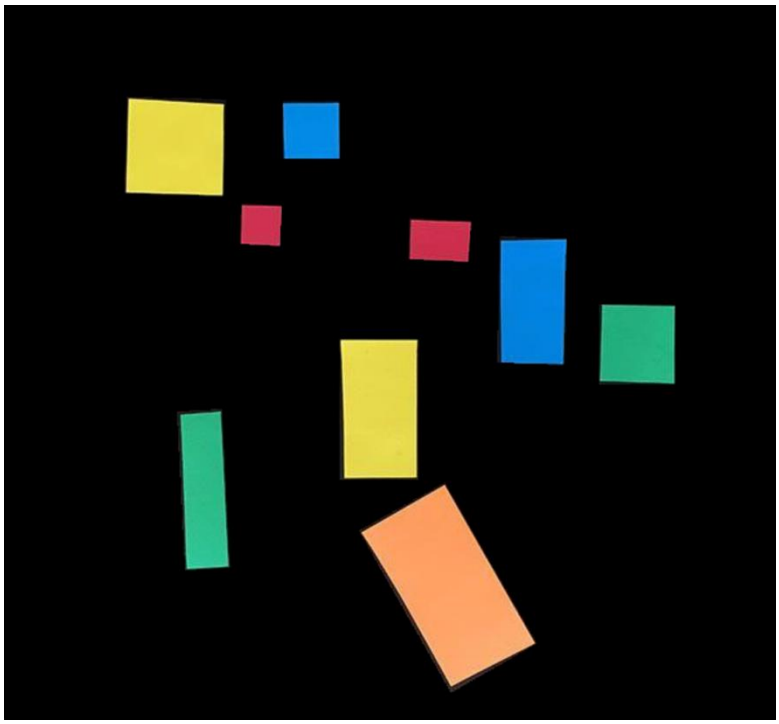

حال در این مرحله با توجه به خواسته سوال' قصد داریم تا اشکالی که چهار ضلعی نیستند را از تصویر اصلی حذف کنیم و اشکل چهار ضلعی را در جای خود باقی بگذاریم .

برای این کار ابتدا یک بک گراند سیاه که ابعادی برابر تصویر اصلی داشته باشد درست کردیم ( با استفاده از دستور zeros ) در حلقه for تعریف شده برای طی کردن تمام کانتور ها نیز به صورت زیر عمل کردیم. یک if قرار دادیم تا اگر تصویر مستطیل یا مربع باشد' یک مستطیل چرخیده محیط شده به کانتور تشکیل دهد .تابع boxpoint درواقع مختصات چهار نقطه مستطیل را میدهد و در خط بعدی از آنجایی که مقادیر پیکسل ها باید صحیح باشند ' این اعداد به اعداد صحیح تبدیل می شوند . حال یک ماسک سیاه با ابعاد تصویر اصلی ایجاد میکنیم و بعد کانتور تعریف شده در بالا را بر روی ماسک می کشد ( داخل کانتور نیز پر می شود ) حال تصویر با استفاده از ماسک تعریف شده به صورت بیت به بیت and می شود و درواقع با این کار تنها نواحی که پشت ماسک هستند باقی می ماند . حال در نهایت در خط آخر پیکسل هایی از بالایی که سفید هستند را بر روی موقعیت های متناطرشان بر روی پس زمینه مشکی ایجاد شده قرار می دهد .

```
black_background = np.zeros_like(image)
```

```
if shape in ["square", "rectangle"]:  # Only keep squares and rectangles
    rect = cv2.minAreaRect(contour)
    box = cv2.boxPoints(rect)
    box = np.int0(box)

    # Create a mask and draw the rotated rectangle
    mask = np.zeros_like(gray)
    cv2.drawContours(mask, [box], 0, 255, -1)
    # Bitwise-and to extract the region
    rotated_cropped = cv2.bitwise_and(image_2, image_2, mask=mask)

    # Place extracted part on black background
    black_background[mask == 255] = rotated_cropped[mask == 255]
```

نتیجه کد بالا به صورت رو به رو می شود :



## سوال ۲:

در این سوال از ما خواسته شده است با استفاده از روش های پردازش تصویر وپردازش لبه زمین های کشاورزی در تصویر هوایی را تشخیص داده و مساحت آنها را محاسبه کرده و رتبه آنها از لحاظ وسعت را روی تصویر نمایش دهیم.

در ابتدا کتابخانه های مورد نیاز را اضافه کرده وتوابعی مانند imshow, auto_canny که در ادامه استفاده میشود را تعریف میکنیم.

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

def imshow(title = "Image", image = None, size = 10):
    w, h = image.shape[0], image.shape[1]
    aspect_ratio = w/h
    plt.figure(figsize=(size * aspect_ratio,size))
    plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    plt.title(title)
    plt.show()
```

[16] ✓ 0.0s

```
def autoCanny(image):

  blurred_img = cv2.blur(image, ksize=(5,5))
  med_val = np.median(image)
  lower = int(max(0, 0.66 * med_val))
  upper = int(min(255, 1.33 * med_val))
  edges = cv2.Canny(image=image, threshold1=lower, threshold2=upper)
  return edges
```

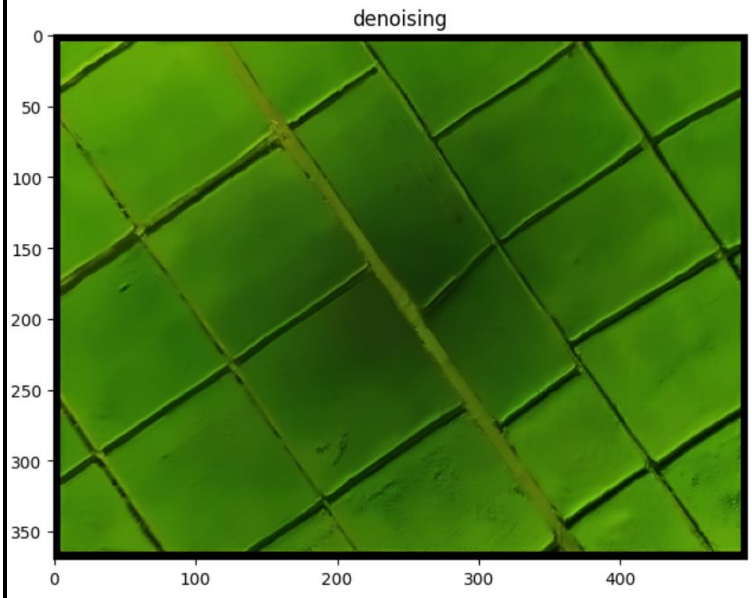
[18] ✓ 0.0s

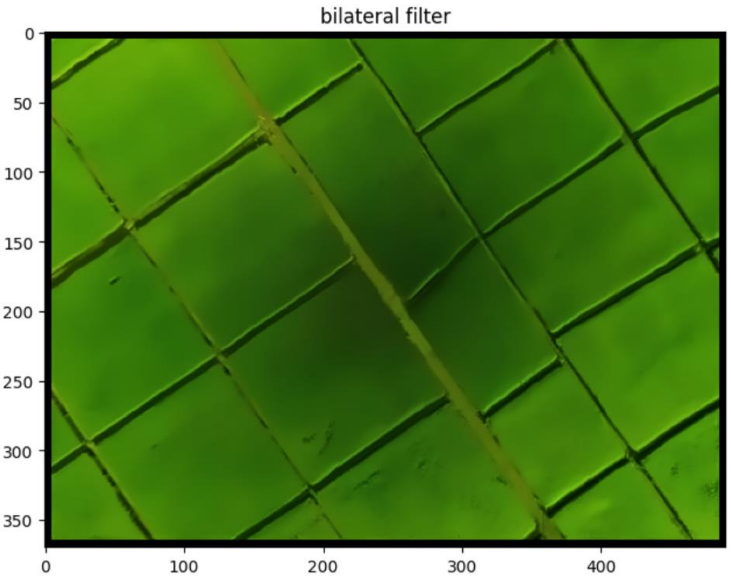
الگوریتم کلی این سوال به این صورت است که نویزتصویر را گرفته وتصویر رنگی را به سیاه و سفید تبدیل میکنیم. سپس از تابع تشخیص لبه canny استفاده کرده و لبه ها را تشخیص میدهیم و در ادامه با استفاده از dilation که یکی از روش های مورفولوژی است لبه ها را ضخیم تر میکنیم. و در نهایت کانتور های روی عکس را پیدا میکنیم. دقت شود که ترتیب انجام مراحل بالا و اعداد به کار گرفته شده میتواند متفاوت باشد ونتیجه با استفاده از سعی وخطا به ازای مقادیر مختلف و مراحل گوناگون به دست آمده است.

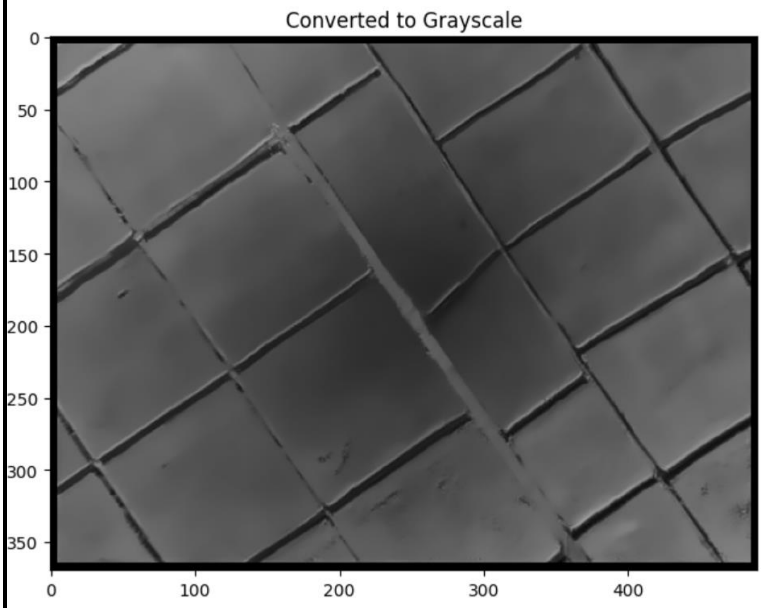
```python
image = cv2.imread('C:/Users/My/Desktop/HW3_MULTI/farms_2.jpeg')
imshow('image', image)

denoising = cv2.fastNlMeansDenoisingColored(image, None,11,11, 9,11)#Non-Local Means
imshow('denoising', denoising)

gray = cv2.cvtColor(denoising, cv2.COLOR_BGR2GRAY)
imshow("Grayscale", gray)

filtered_image = cv2.bilateralFilter(gray,9, 5, 5) #sigmaColor,sigmaSpace
imshow("bilateral filter", filtered_image)

edged=cv2.Canny(filtered_image, 20, 225)
imshow("canny", edged)

kernel1 = np.ones((9,9), np.uint8)
dilation = cv2.dilate(edged, kernel1, iterations = 1)
imshow('Dilation', dilation)

kernel=create_diamond_filter(9)
erosion = cv2.erode(dilation, kernel, iterations=1)
imshow('erosion', erosion)

 copy = image.copy()
 contours, hierarchy = cv2.findContours(image=cv2.bitwise_not(erosion), mode=cv2.RETR_EXTERNAL,method=cv2.CHAIN_APPROX_NONE)
 pic_con = cv2.drawContours(copy, contours, -1, color=(0, 0, 255), thickness=5)
 image = cv2.imread('C:/Users/My/Desktop/HW3_MULTI/farms_2.jpeg')
 cv2.drawContours(image, contours, -1, (0,0,255), 4)

 imshow('Contours', image)
```

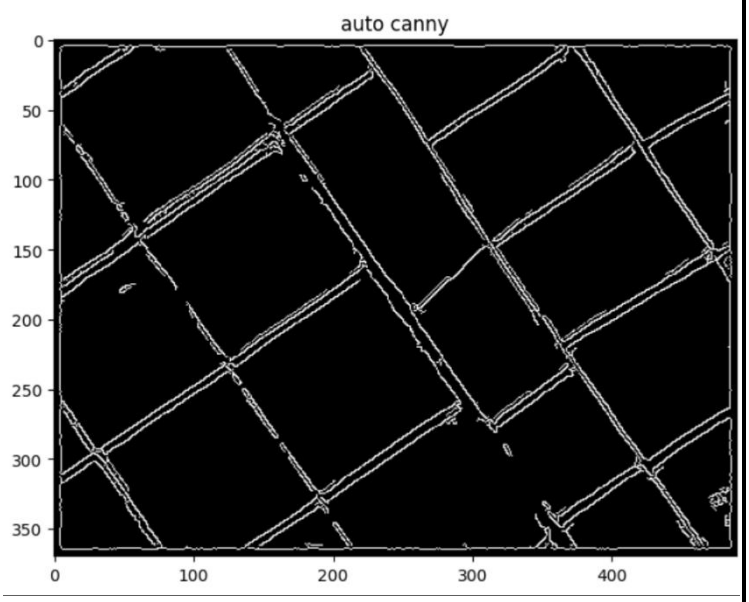
✓ 5.6s

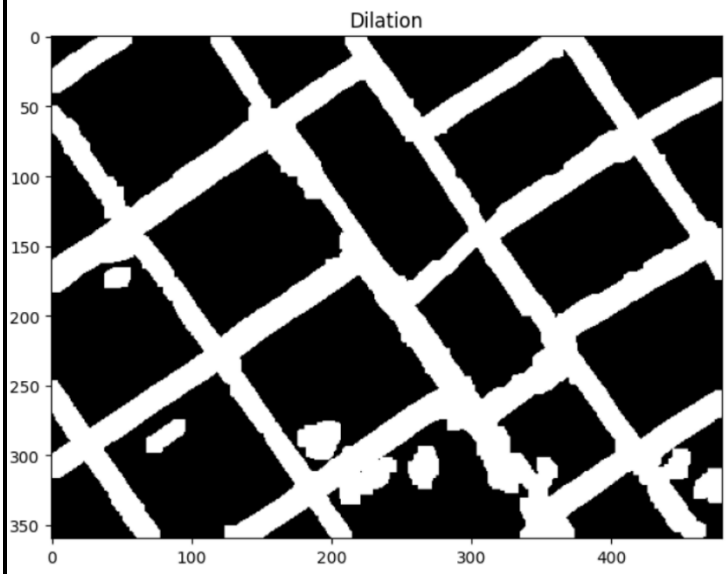
نتایج مراحل بالا به صورت زیر است:

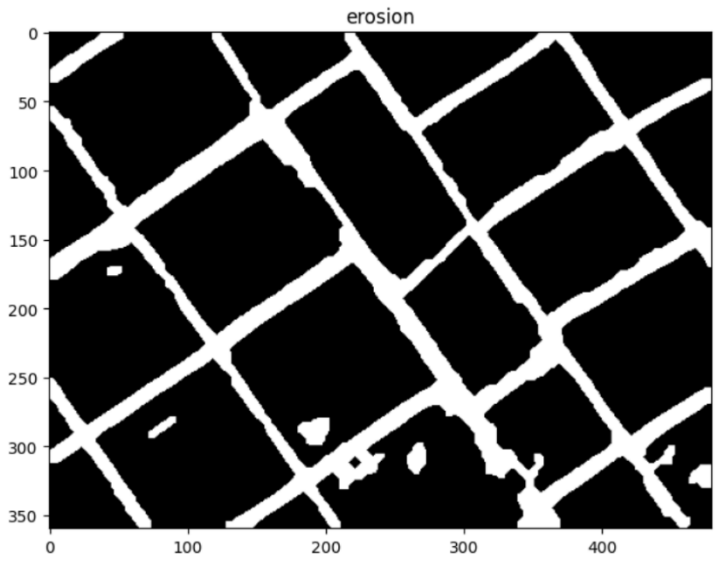


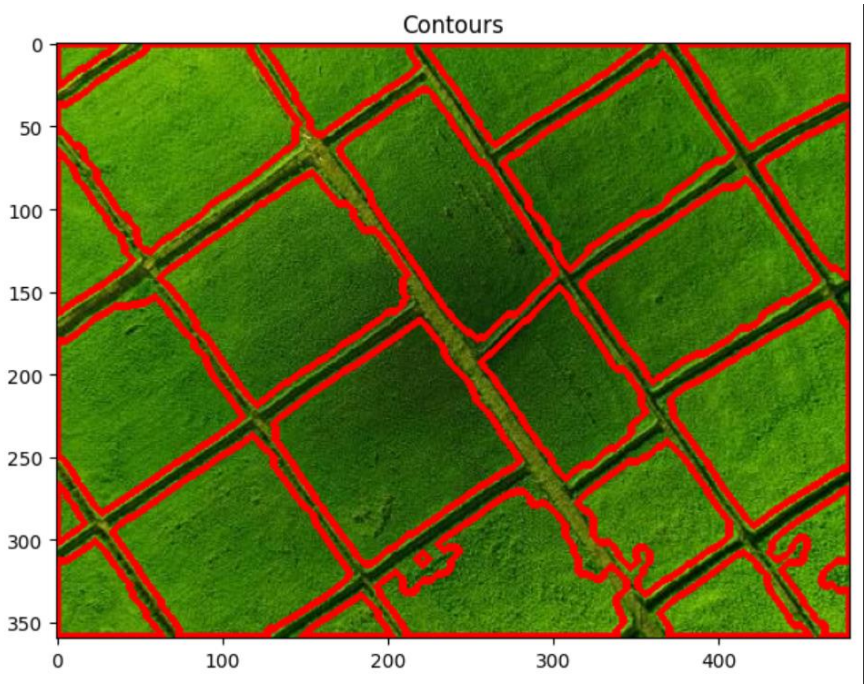




Converted to Grayscale
0
50
100
150
200
250
300
350
0
100
200
300
400


auto canny
0
50
100
150
200
250
300
350
0
100
200
300
400


Dilation
0
50
100
150
200
250
300
350
0
100
200
300
400


erosion
0
50
100
150
200
250
300
350
0
100
200
300
400


Contours
0
50
100
150
200
250
300
350
0
100
200
300
400

در ادامه با استفاده از دستور زیر تعداد کانتور ها چاپ میشود. برای محاسبه مساحت هر کانتور تابع زیر تعریف میشود. و در ادامه مساخحت ها بر حسب اندازه چیده میشوند و چاپ میشوند.

```
print("Number of Contours found = " + str(len(contours)))
```
✓ 0.0s

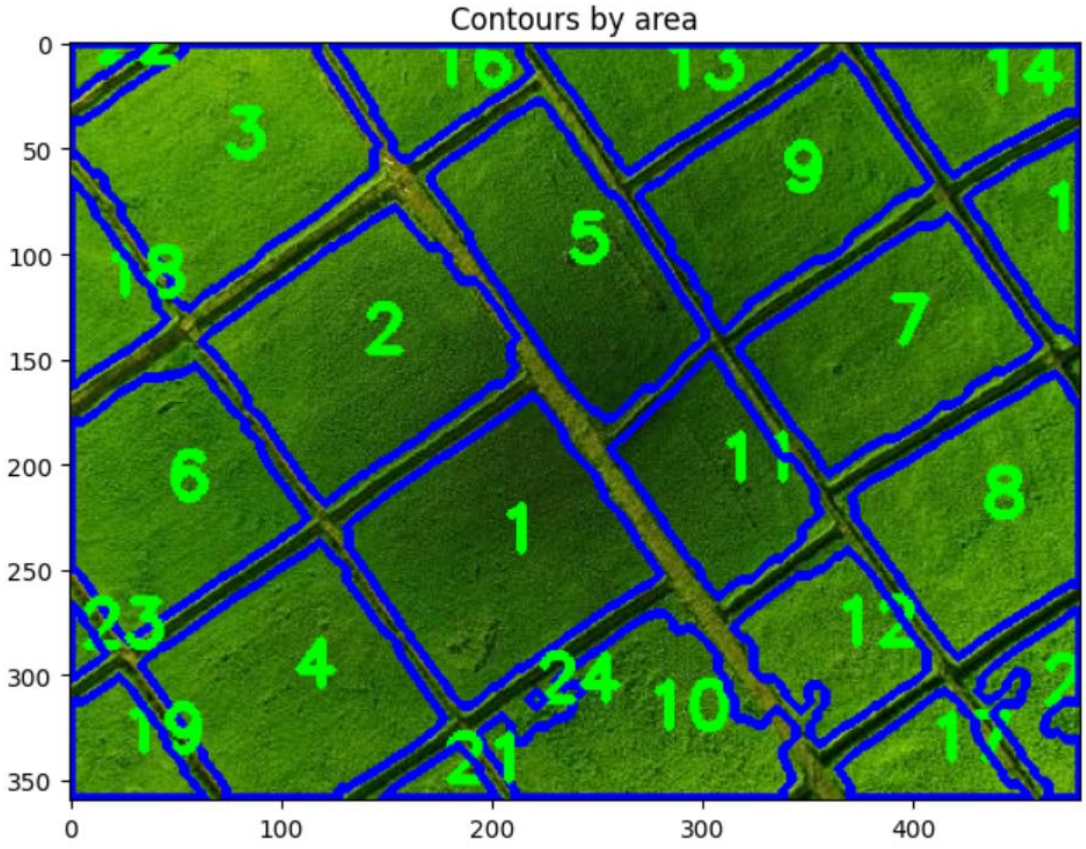
```
Number of Contours found = 30
```

```
def get_contour_areas(contours):
    """returns the areas of all contours as list"""
    all_areas = []
    for cnt in contours:
        area = cv2.contourArea(cnt)
        all_areas.append(area)
    return all_areas
```
✓ 0.0s

```
print("num of cnt = " + str(len(contours)))
print("Contor Areas before sorting...")
print(get_contour_areas(contours))

sorted_contours = sorted(contours, key=cv2.contourArea, reverse=True)
print("Contor Areas after sorting...")
print(get_contour_areas(sorted_contours))
```
✓ 0.0s

در مرحله آخر با استفاده از مومنت مرکز هر کانتور به دست می آید و رتبه بر حسب مساحت کانتور در وسط آن نوشته میشود که نتیجه نهایی زیر را میدهد.

```
image = cv2.imread('C:/Users/My/Desktop/HW3_MULTI/farms_2.jpeg')
for (i,c) in enumerate(sorted_contours):
    M = cv2.moments(c)
    cx = int(M['m10'] / M['m00'])
    cy = int(M['m01'] / M['m00'])
    cv2.putText(image, str(i+1), (cx, cy), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 3)
    cv2.drawContours(image, [c], -1, (255,0,0), 3)

imshow('Contours by area', image)
```
✓ 0.8s


Contours by area
0
50
100
150
200
250
300
350
0
100
200
300
400
3
16
13
14
9
1
5
18
2
7
6
1
8
4
24
10
19
21

## سوال سوم :

در اینجا ما میخواهیم برنامه ای طراحی کنیم تا یوزر بتواند عکسی از دیوایس لود کند و یا با استفاده از وبکم عکسی ثبت کند و در نهایت با استفاده از یک مدل از پیش train شده و imagenet بتوانیم کلاس عکس و میزان دقت در کلاس بندی آن را بدست آوریم و این مقادیر بر روی تصویر نمایش داده شوند .

در ابتدا مدل مدنظرمان را لود می کنیم' ما یک resnet50 از پیش train شده با وزن های Imagenet را لود کردیم .

در ابتدا تابعی تعریف می شود تا تصویر را برای کلاس بندی کردن آماده کندو درواقع پیش پردازش انجام شود. پیش‌پردازش معمولاً شامل تغییر اندازه و نرمال سازی تصویر بر اساس نیاز مدل است.

```python
# Load a pre-trained ResNet50 model from Keras
model = tf.keras.applications.ResNet50(weights='imagenet')

def preprocess_image(image_path):
    """ Preprocess the image for classification. """
    try:
        image = PIL.Image.open(image_path)
        image = image.resize((224, 224))
        image = np.array(image)
        image = tf.keras.applications.resnet50.preprocess_input(image)
        image = np.expand_dims(image, axis=0)  # Add batch dimension
        return image
    except Exception as e:
        print(f"Error opening image: {e}")
        raise
```

حال تابعی تعریف کردیم به نام classify_image تا تصویر تنظیم شده را از تابع قبلی دریافت کند و کلاس تصویر را predict کند ' از آنجایی که ممکن است چند کلاس برای تصویر پیش بینی شود ' تنها کلاس با احتمال بالاتر را انتخاب می کنیم . خروجی این تابع نام کلاس تصویر و میزان دقت تشخیص می باشد .

```python
def classify_image(image_path):
    """ Classify the image using the pre-trained model and return class label with confidence. """
    try:
        image = preprocess_image(image_path)
        predictions = model.predict(image)
        decoded_predictions = tf.keras.applications.resnet50.decode_predictions(predictions, top=1)[0][0]
        class_name, confidence = decoded_predictions[1], decoded_predictions[2]
        return class_name, confidence
    except Exception as e:
        print(f"Error classifying image: {e}")
        return "Classification Error", 0
```

در زیر تابع frame_updateرا داریم که وظیفه آن بروزرسانی فریم در ui سیستم می باشد . به طور کلی کار این تابع این است که وب کم را متصل کند و تصویر را بخواند و در صورت وجود تصویر آن را متناسب با لیبل ui ایجاد شده کند.

```python
def frame_update():
    """ Update the video frame in the GUI. """
    global video_label, cap, ret, frame
    cap = cv2.VideoCapture(0)
    ret, frame = cap.read()
    if ret:
        rgb_image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        h, w, ch = rgb_image.shape
        bytes_per_line = ch * w
        convert_to_Qt_format = QImage(rgb_image.data, w, h, bytes_per_line, QImage.Format_RGB888)
        pixmap = QPixmap.fromImage(convert_to_Qt_format).scaled(640, 480, Qt.KeepAspectRatio)
        video_label.setPixmap(pixmap)
```

تابع زیر را نیز تعریف می کنیم . این تابع تصویر گرفته شده را ذخیره می کند و تابع classify_image را با مسیر تصویر ذخیره شده فراخوانی می‌کند تا تصویر را طبقه‌بندی کند و نام کلاس و احتمال آن را برمی‌گرداند.

```python
def capture_and_classify():
    " Capture image from webcam, classify, and display. "
    global cap, frame
    if cap is None or not cap.isOpened():
        cap = cv2.VideoCapture(0)  # Ensure the camera is opened here
    ret, frame = cap.read()
    if ret:
        cv2.imwrite('captured_photo.jpg', frame)
        class_name, confidence = classify_image('captured_photo.jpg')
        display_classified_image('captured_photo.jpg', class_name, confidence)
    else:
        print("Error: Could not capture an image.")
    cap.release()
    cap = None  # Reset cap to ensure it's clean for next use
```

به توضیح تابع display_classified_image میپردازیم ' همانطور که از اسم تابع معلوم است تصویر کلاسیفای شده را نمایش میدهد .

این تابع مسیر تصویر و نام کلاس و احتمال آن را به عنوان ورودی دریافت می کند و سپس تصویر را از مسیر گرفته و می خواند . از آنجایی که قصد داریم تا نام کلاس تصاویر و احتمال آن ها بر روی خود تصویر نشان داده شود و هنگامی که تصاویر محتلف با رزولوشن مختلف و ابعاد مختلف نمایش داده می شود ' نسبت به اندازه این تصاویر متن نوشته شده نیز مکان و ابعاد متغییری دارد (چون بر اساس تعداد پیکسل فاصله از لبه مکان آن را مشخص می کنیم ). بنابراین اندازه تصاویر را به صورت ثابت در نطر می گیریم و برای اینکه تصویر خراب نشود در سایز جدید aspect ratio را در تصویر در نظر می گیریم . همچنین اگر تصویر هم اندازه ' اندازه تعریف شده ما نباشد دور آن را انگار پدینگ مشکی قرار می دهیم . که برای اینکار یک بک گراند مشکی با اندازه مشخصمان درست میکنیم و تصویر را روی آن می اندازیم و برای اینکه این تصویر تغییر سایز داده درست در وسط صفحه سیاه قرار گیرد مختصات مرکز را حساب می کنیم و تصویر تغییر سایز داده را در مرکز پس زمینه قرار می دهیم . برای نمایش تصویر در ui آن را تبدیل به فرمت مناسب می کنیم .

```python
def display_classified_image(image_path, class_name, confidence):
    """ Display the image with aspect ratio maintained and classification with confidence on the UI. """
    image = cv2.imread(image_path)
    if image is None:
        print(f"Failed to read image from {image_path}")
        return

    # Define the target display size
    target_width, target_height = 640, 480

    # Calculate the aspect ratio of the image
    height, width, channels = image.shape
    scaling_factor = min(target_width / width, target_height / height)
    new_width = int(width * scaling_factor)
    new_height = int(height * scaling_factor)

    # Resize the image to maintain aspect ratio
    resized_image = cv2.resize(image, (new_width, new_height), interpolation=cv2.INTER_AREA)

    # Create a black background to paste the resized image onto
    background = np.zeros((target_height, target_width, 3), dtype=np.uint8)

    # Calculate centering position
    x_offset = (target_width - new_width) // 2
    y_offset = (target_height - new_height) // 2

    # Place the resized image onto the center of the background
    background[y_offset:y_offset+new_height, x_offset:x_offset+new_width] = resized_image

    # Convert the background to QImage for display
    rgb_image = cv2.cvtColor(background, cv2.COLOR_BGR2RGB)
    qt_image = QImage(rgb_image.data, target_width, target_height, rgb_image.strides[0], QImage.Format_RGB888)
    pixmap = QPixmap.fromImage(qt_image)
```

در ادامه تابع که در پایین نشان داده شده است به نوشتن نام کلاس و احتمال بر روی تصویر می پردازیم. رنگ و نوع فونت و اندازه را مشخص می کنیم ’ همچنین ابعاد باکسی که برای نوشتن در نظر گرفتیم و موقعیت آن نسبت به تصویر را مشخص میکنیم .

```
# Prepare to draw the classification result and confidence on the image
painter = QPainter(pixmap)
painter.setPen(QColor(255, 255, 255))
painter.setFont(QFont('Arial', 20))

# Display text format
display_text = f"{class_name} - {confidence*100:.2f}"
text_rect = QtCore.QRect(0, pixmap.height() - 50, pixmap.width(), 50)
painter.drawText(text_rect, Qt.AlignCenter, display_text)
painter.end()

# Set the pixmap to the QLabel to display on the UI
image_label.setPixmap(pixmap)
```

در ادامه تابع open_file را داریم که با استفاده ار آن در رابط کاربری میتوانیم تصویر مد نظر را انتخاب کنیم. در این تابع اسم فایل تصویر مدنظر به صورت path به تابع classify_and_display داده میشود.

در تابع classify_and_display مسیر تصویر انتخاب شده ورودی گرفته میشود و با پاس داده شدن به تابع classify_image اسم کلاس و میزان شباهت تصویر به عنوان خروجی گرفته شده و دوباره به تابع display_classified_image پاس داده میشود تا نمایش داده شود.

```
def open_file():
    """ Open a file dialog to select an image, display it, and classify it. """
    options = QFileDialog.Options()
    options |= QFileDialog.DontUseNativeDialog
    file_name, _ = QFileDialog.getOpenFileName(None, "Open Image File", "", "Image Files (*.png *.jpg *.bmp *.jpeg)", options=options)
    if file_name:
        classify_and_display(file_name)


def classify_and_display(image_path):
    """ Display the selected image and classify it, showing the class label and confidence on the image. """
    global image_label
    try:
        class_name, confidence = classify_image(image_path)
        display_classified_image(image_path, class_name, confidence)
    except Exception as e:
        print(f"Error displaying image: {e}")
```

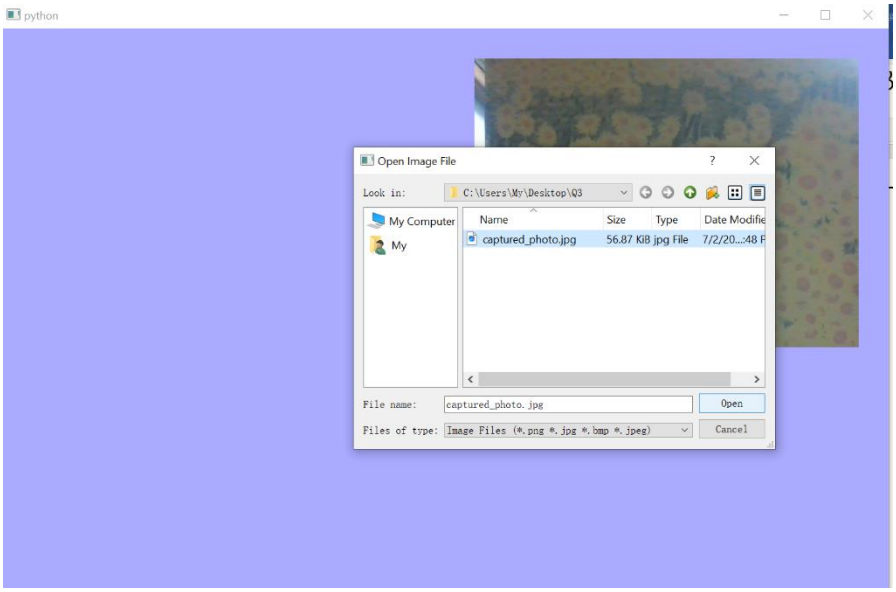
کد مربوط به رابط کاربری به صورت زیر زده شد.

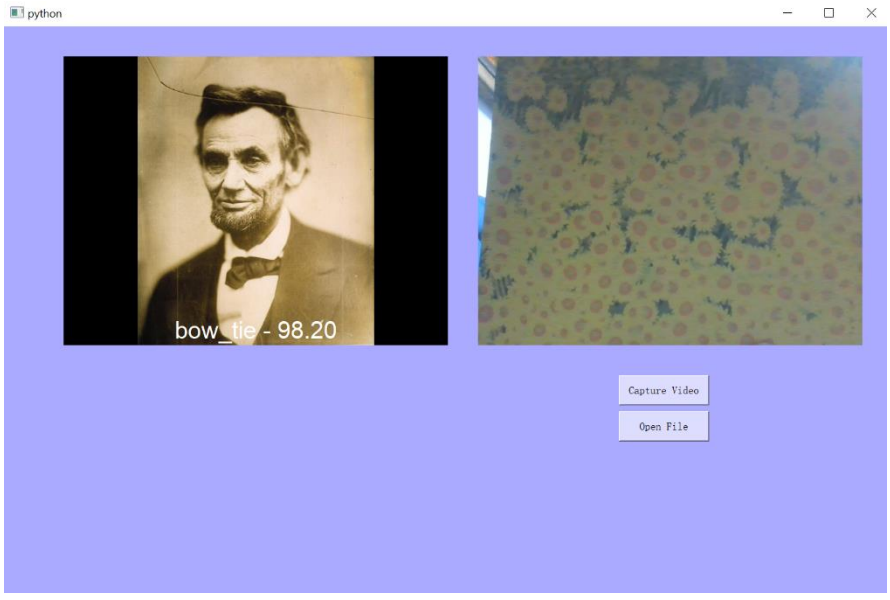
```
if __name__ == "__main__":
    app = QApplication(sys.argv)
    MainWindow = QWidget()
    MainWindow.setObjectName("MainWindow")
    MainWindow.resize(1480, 1000)
    MainWindow.setStyleSheet("background-color: rgb(170, 170, 255);")

    # Setup labels
    video_label = QLabel(MainWindow)
    video_label.resize(640, 480)
    video_label.move(790, 50)

    image_label = QLabel(MainWindow)
    image_label.resize(640, 480)
    image_label.move(100, 50)  # Adjust the position to not overlap with video_label

    # Setup buttons
    video_btn = QPushButton("Capture Video", MainWindow)
    video_btn.setGeometry(QtCore.QRect(1025, 580, 150, 50))
    video_btn.setStyleSheet("background-color: rgb(221, 222, 255);")
    video_btn.clicked.connect(capture_and_classify)

    open_file_btn = QPushButton("Open File", MainWindow)
    open_file_btn.setGeometry(QtCore.QRect(1025, 640, 150, 50))
    open_file_btn.setStyleSheet("background-color: rgb(221, 222, 255);")
    open_file_btn.clicked.connect(open_file)
```

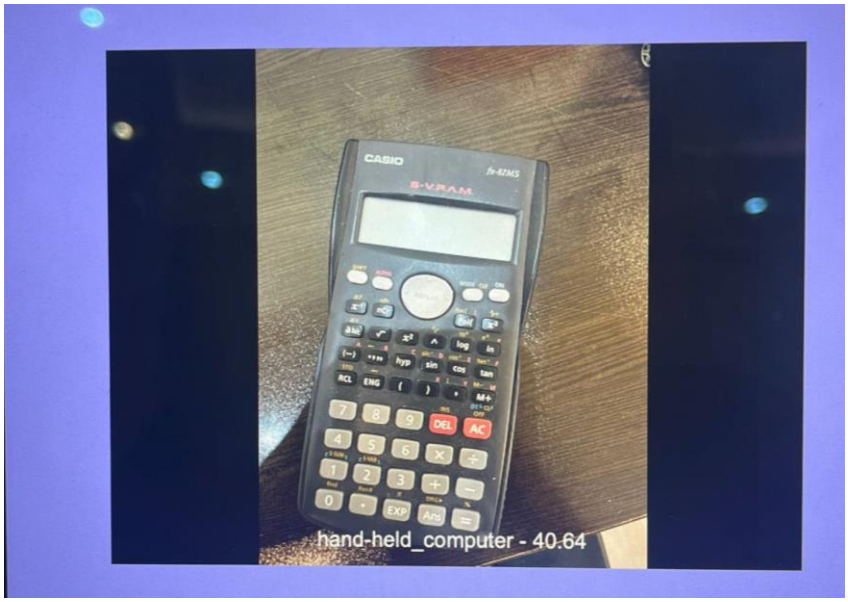
```
    # Timer to update video frames
    timer = QTimer(MainWindow)
    timer.start(1000//30)  # Update at about 30fps
    timer.timeout.connect(frame_update)

    MainWindow.show()
    sys.exit(app.exec_())
```

در ادامه تصاویری از نتایج کارکرد برنامه آورده شده است.

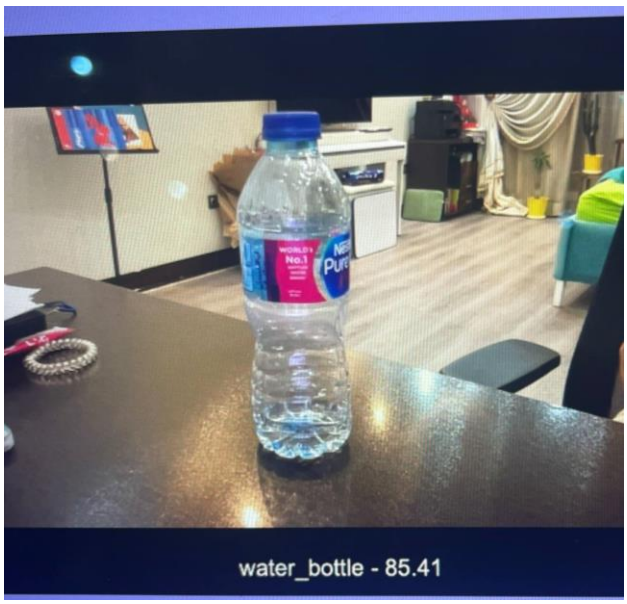

python
Capture Video
Open File


python
Open Image File
Look in:
C:\Users\My\Desktop\Q3
My Computer
My
Name
Size
Type
Date Modifie
captured_photo.jpg
56.87 KiB jpg File
7/2/20...:48 P
File name:
captured_photo. jpg
Open
Files of type:
Image Files (*.png *.jpg *.bmp *.jpeg)
Cancel


python
bow_tie - 98.20
Capture Video
Open File


CASIO
S-V.P.A.M.
hand-held_computer - 40.64


paper_towel - 55.80


water_bottle - 85.41