به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس رباتیک

مینی پروژه دوم

نام و نام خانوادگی	علی سبزه‌جو
شماره دانشجویی	810100372
تاریخ ارسال گزارش	1401/08/17

فهرست گزارش سوالات

بخش اول، سوال ۱ – شناسایی ماسک روی صورت

در این قسمت ایتدا مدل شبکه عصبی train شده را با استفاده از کتابخانه keras لود کرده و خلاصه مدل را با دستور model.summary() نمایش می‌دهیم که خروجی آن بصورت زیر است:

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_7 (Conv2D)           (None, 148, 148, 32)      896

 max_pooling2d_7 (MaxPooling  (None, 74, 74, 32)       0
 2D)

 conv2d_8 (Conv2D)           (None, 72, 72, 32)        9248

 max_pooling2d_8 (MaxPooling  (None, 36, 36, 32)       0
 2D)

 conv2d_9 (Conv2D)           (None, 34, 34, 32)        9248

 max_pooling2d_9 (MaxPooling  (None, 17, 17, 32)       0
 2D)

 flatten_3 (Flatten)         (None, 9248)              0

 dense_5 (Dense)             (None, 100)               924900

 dense_6 (Dense)             (None, 1)                 101

=================================================================
Total params: 944,393
Trainable params: 944,393
Non-trainable params: 0
_________________________________________________________________
```

شکل 1- خلاصه مدل آموزش داده شده برای این مسئله

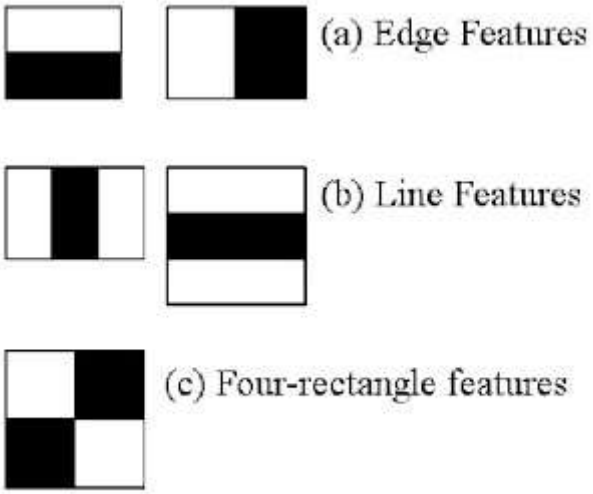
همچنین تشخیصگر صورت آموزش دیده را هم با دستور cv2.CascadeClassifier لود می‌کنیم.

سپس تابعی به نام mask_detection نوشته‌ایم که کد آن را در شکل زیر مشاهده می‌کنید:

```python
def mask_detection(img):
  img_RGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img_GRAY = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
  faces = face_cascade.detectMultiScale(img_GRAY, 1.3, 5, minSize = (100, 100))
  for (x,y,w,h) in faces:
    roi_BGR = img[y:y+h, x:x+w]
    roi_GRAY = img_GRAY[y:y+h, x:x+w]
    roi_RGB = img_RGB[y:y+h, x:x+w]
    dim = (150, 150)
    img_n = roi_RGB.astype('float32') / 255
    resized = cv2.resize(img_n, dim)
    resized1 = np.reshape(resized,(1,150,150,3))
    pred = mymodel.predict(resized1)
    if pred < 0.5:
      label = 'Mask'
      color = (0,255,0)
    else:
      label = 'No Mask'
      color = (0,0,255)
    cv2.putText(img, label, (x+10, y+h+15), cv2.FONT_HERSHEY_SIMPLEX,
                1, color, 4)
    cv2.rectangle(img,(x,y),(x+w,y+h),color,2)
  cv2.imshow("real image",img)
```

شکل 2- تابع mask_detection برای تشخصیص ماسک روی صورت

این تابع یک ورودی دارد که همان تصویر ماست. در ابتدا چون قرار است تصویر را به کمک کتابخانه opencv بخوانیم و opencv وقتی تصاویر را می‌خواند، آن‌ها را با فرمت رنگی BGR ذخیره می‌کند، نیاز به تغییر فرمت رنگی تصویر ذخیره شده به RGB داریم (چون ورودی شبکه عصبی آموزش دیده ما باید تصویر RGB باشد تا خروجی مناسب را دریافت کنیم؛ چون با تصاویر RGB آموزش دیده شده است) و برای این منظور از دستور cv2.cvtCOLOR(img, cv2.COLOR_BGR2RGB) استفاده می‌کنیم. همچنین برای استفاده از تشخیصگر صورت آموزش دیده شده، یک تصویر gray scale هم نیاز داریم و برای این منظور از دستور cv2.cvtCOLOR(img, cv2.COLOR_BGR2GRAY) استفاده می‌کنیم. تشخیصگر صورت haarcascade_frontface_default یک رویکرد مبتنی بر یادگیری ماشین است که در آن یک تابع cascade از تعداد زیادی تصاویر مثبت و منفی آموزش داده می‌شود. در ابتدا، الگوریتم به تعداد زیادی تصاویر مثبت (تصاویر چهره) و تصاویر منفی (تصاویر بدون چهره) برای آموزش کلاسیفایر نیاز دارد. سپس باید ویژگی‌هایی را از آن استخراج کنیم. برای این کار از ویژگی های haar که در تصویر زیر نشان داده شده، استفاده شده است که دقیقا مانند کرنل‌ها در CNN عمل می‌کنند.



شکل 3- ویژگی‌های **Haar** که برای استخراج ویژگی از تصویر استفاده می‌شوند.

هر ویژگی یک مقدار واحد است که با کم کردن مجموع پیکسل‌های زیر مستطیل سفید از مجموع پیکسل های زیر مستطیل سیاه به دست می‌آید.

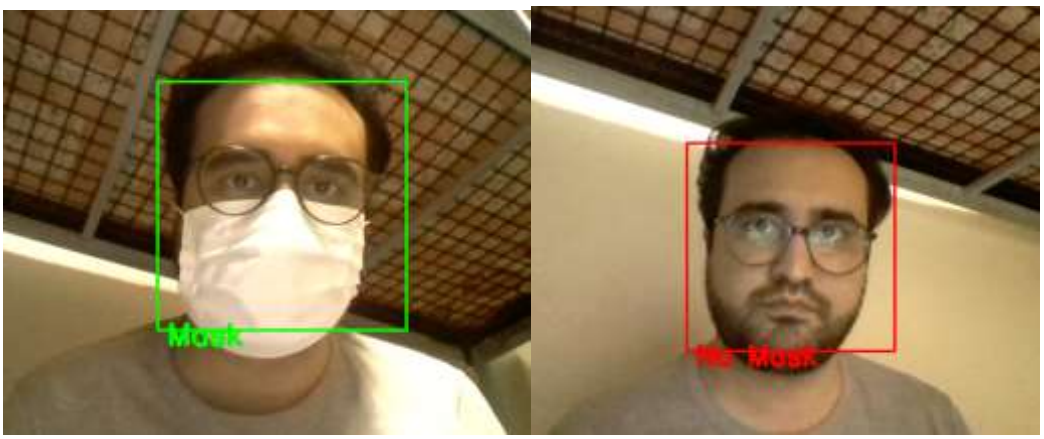
ما برای استفاده از این تشخیصگر از دستور زیر استفاده می‌کنیم:

detectMultiScale(image, objects,scaleFactor, minNeighbors, flags, minSize, maxSize)

که در آن image که تصویر ماست. objects بردار مستطیل‌هایی است که هر مستطیل شامل شیئ تشخیص داده شده است. scaleFactor پارامتری است که مشخص می‌کند اندازه تصویر در هر بار مقیاس کردن تصویر چقدر کاهش می‌یابد. minSize حداقل اندازه ممکن جسم را مشخص می‌کند. اشیاء کوچکتر از این سایز نادیده گرفته می‌شوند. minNeighbors پارامتری است که مشخص می‌کند هر مستطیل نامزد باید چند همسایه داشته باشد تا آن را حفظ کند. maxSize حداکثر اندازه ممکن جسم را مشخص می‌کند. اجسام بزرگتر از این سایز نادیده گرفته می شوند. ما در این مسئله scaleFactor را 1.3 و minNeighbors را 5 و minSize را (100,100) در نظر گرفتیم. همچنین maxSize را مشخص نکرده و محدودیتی برای آن درنظر نمی‌گیریم.

حال از صورت‌هایی که تشخیصگر از روی تصویر تشخیص داده، استفاده کرده و با توجه به مختصات صورت، روی تصویر RGB آن را crop کرده و سپس به ابعاد (150,150,3) ریسایز می‌کنیم و در نهایت هم تصویر را با تقسیم مقادیر همه درایه‌ها بر 255 نرمالایز می‌کنیم. به این ترتیب این تصویر آماده وارد شدن به مدل شبکه عصبی است تا prediction انجام شود و مدل شبکه عصبی پیش‌بینی کند که شخص دارای ماسک است یا خیر. خروجی مدل شبکه عصبی عددی بین 0 تا 1 است که 0 یعنی وجود ماسک و 1 یعنی

عدم وجود ماسک. به این ترتیب ما خروجی‌هایی که مقدار کمتر از 0.5 دارند را تصاویر با ماسک و خروجی‌هایی که مفدار بیشتر مساوی 0.5 دارند را تصاویر بدون ماسک درنظر می‌گیریم. سپس همه این مستطیل‌ها و لیبل‌ها را روی تصویر اصلی نمایش می‌دهیم. در ادامه ما فریم‌های تصویر را به صورت live از وبکم دریافت کرده و وجود یا عدم وجود ماسک را توسط کد نوشته شده پیشبینی می‌کنیم. در شکل زیر نمونه‌ای از خروجی کد نوشته شده را مشاهده می‌کنید که در ویدیوی موجود در همین فولدر هم فیلم آن قابل مشاهده است.



شکل 4- نمونه‌ای از خروجی کد نوشته شده برای فریم‌های تصاویر لایو گرفته شده از وبکم لپتاپ

همانطور که مشاهده می‌گردد، هم ناحیه صورت به خوبی تشخیص داده شده است و هم وجود و عدم وجود ماسک به درستی تشخیص داده شده است.

کد مربوط به این سوال در فایل MP2_Q1_S1.py موجود است.

بخش اول، سوال ۲ – تعیین دقت مدل شبکه عصبی روی تصاویر با ماسک و بدون ماسک

در این سوال باید تصاویر افراد با ماسک و افراد بدون ماسک را از فولدرهای مربوطه خوانده و دقت شبکه را تعیین کنیم. برای این منظور، از کتابخانه glob استفاده می‌کنیم و با استفاده از دستور زیر تصاویر داخل فولدرها را در درون یک لیست دریافت می‌کنیم:

```
images = [cv2.imread(file) for file in glob.glob (FOLDER PATH)]
```

همچنین آرایه‌هایی از جنس numpy که دارای لیبل‌های مربوط به دیتاست‌ها هستند، تعریف می‌کنیم. این آرایه‌ها برای داده‌های با ماسک و بدون ماسک بصورت زیر تعریف می‌شوند(چون لیبل 0 مربوط به داشتن ماسک و لیبل 1 مربوط به نداشتن ماسک است):

```
y_mask = np.zeros((np.shape(mask_images)[0],1)
y_no_mask = np.ones((np.shape(no_mask_images)[0],1)
```

سپس ابتدا مانند قسمت قبل با استفاده از تشخیصگر صورت Haar، چهره شخص را تشخیص داده و آن را از تصویر اصلی RGB شده جدا کرده(کراپ می‌کنیم) و سپس آن‌ها را به سایز (150,150,3) ریسایز می‌کنیم. همه تصاویر چهره کراپ و ریسایز شده را درون یک آرایه ریخته و با دستور model.evaluate(image,label) ، به ارزیابی مدل می‌پردازیم و دقت مدل را روی آن‌ها محاسبه می‌کنیم.

دقت شود که در این حالت تشخیصگر صورت گاهی نمی‌تواند در تصویری چهره را تشخیص دهد و به این ترتیب دیگر با آن تصویر کاری نداریم و فقط چهره‌های تشخیص داده شده را به شبکه می‌دهیم.

خروجی مربوط به این قسمت در شکل زیر آورده شده است.

```
Test_accuracy for mask: 80.51947951316833
Test_accuracy for no_mask: 78.87324094772339
```

شکل 5- دقت بدست آمده برای شبکه عصبی روی داده‌های با ماسک و بدون ماسک

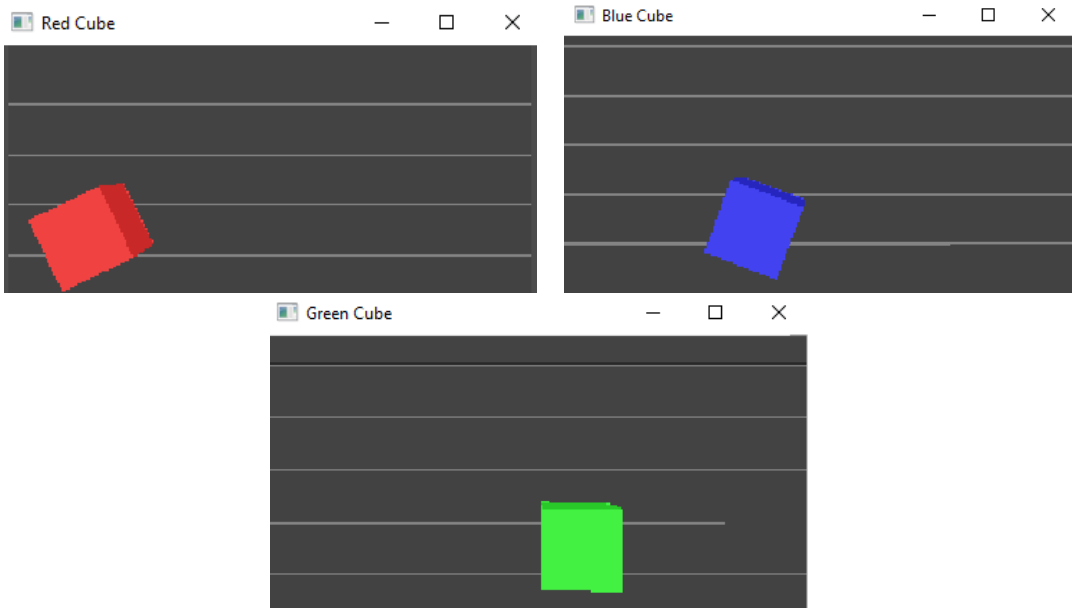
همانطور که مشاهده می‌گردد، دقت بدست آمده برای داده‌های با ماسک حدود 80% و برای داده‌های بدون ماسک حدود 79% می‌باشد.

کد مربوط به این سوال در فایل MP2_Q1_S2.py موجود است.

بخش دوم، سوال ۱ – تعیین رنگ، مکان و زاویه چرخش مکعب‌ها

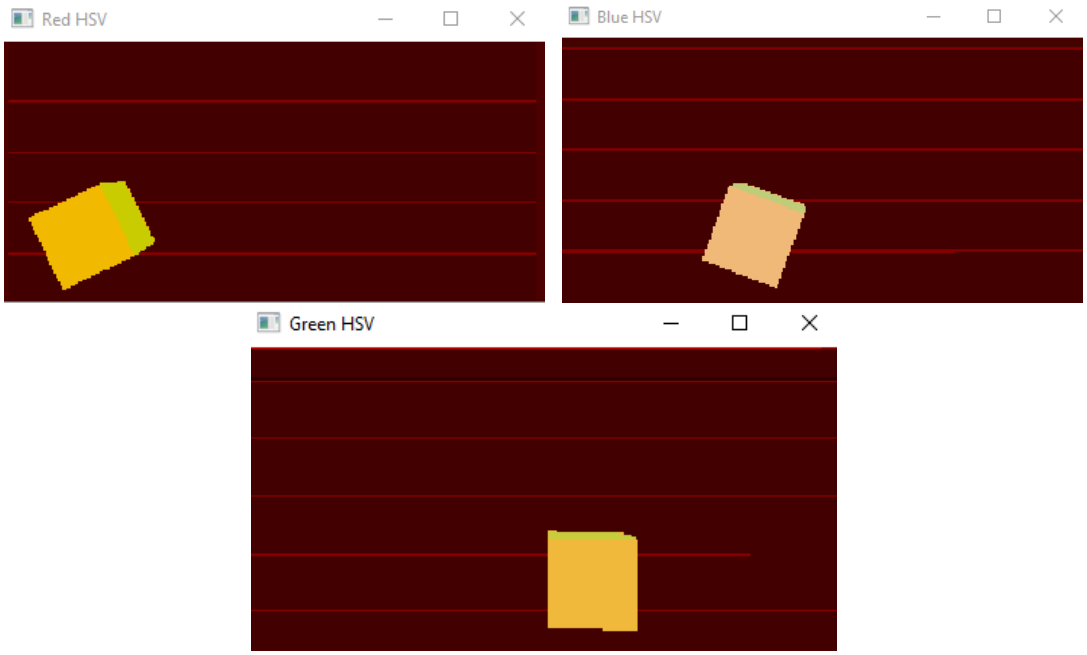
در این سوال به دنبال تعیین رنگ، مکان و زاویه چرخش مکعب‌های موجود در فایل پروژه هستیم. برای کار با تصاویرشان فضای رنگی BGR یا RGB مناسب نیستند زیرا شدن رنگ و میزان روشنایی را در نظر نمی‌گیرند. برای همین منظور با فضای رنگی HSV کار می‌کنیم که در آن H مخفف Hue نشان دهنده رنگ، S مخفف Saturation نشان دهنده میزان رنگی بودن و V مخفف Value نشان دهنده میزان brightness هر رنگ است. وقتی با این فضای رنگی کار می‌کنیم، هر رنگ دارای یم محدوده است و ما هر وقت قرار است با رنگ خاص کار کنیم، دیگر محدوده Hue همان رنگ را مورد بررسی قرار می‌دهیم و با محدوده‌های رنگ‌های دیگر کاری نداریم.

ابتدا تصاویر را خوانده و نمایش می‌دهیم که در شکل زیر آن‌ها را مشاهده می‌کنید:



شکل 6- نمایش تصاویر ورودی مکعب‌های مدنظر

برای تغییر فضای رنگی به HSV از دستور cv2.cvtCOLOR(image, cv2.COLOR_BGR2HSV) استفاده می‌کنیم. تصاویر HSV شده به صورت زیر بدست آمده اند.



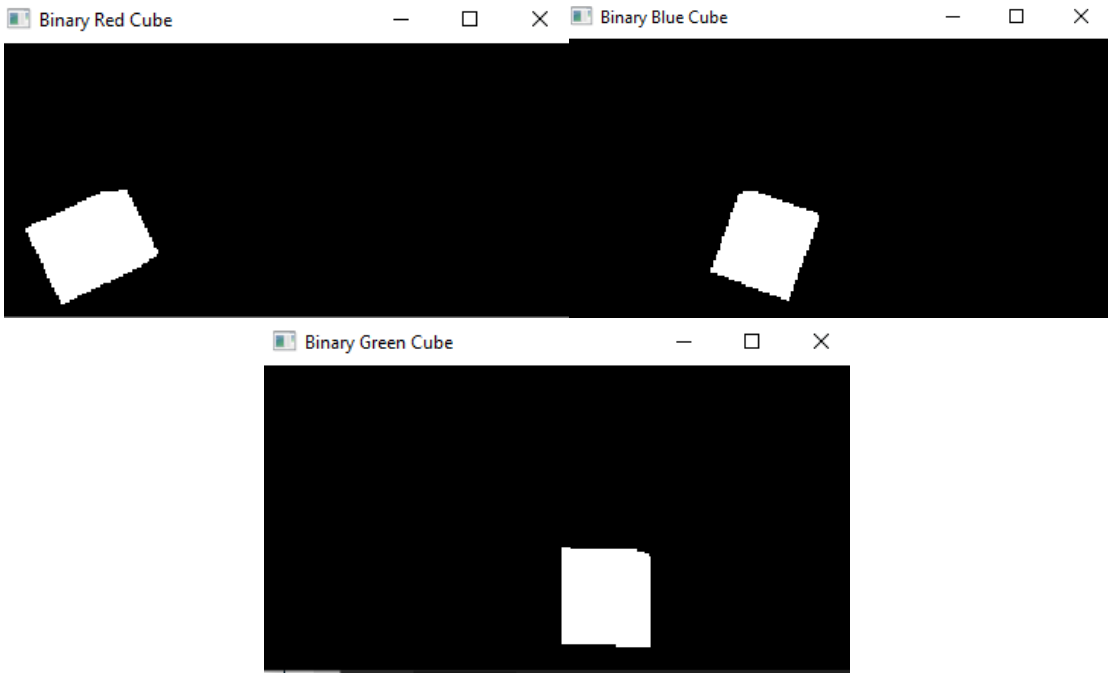
شکل 7- تصاویر مکعب‌ها وقتی که فضای رنگی‌شان به **HSV** تعییر کرده است.

حال برای تولید تصاویر باینری، باید مقادیر باند بالا و پایین HSV مربوط به رنگ‌های سبز و آبی و قرمز را بیابیم. برای این منظور، ابتدا برداری که نشان‌دهنده هر رنگ هست را تولید کرده (به عنوان مثال، برای رنگ آبی بردار [255,0,0]) و با دستور cvtColor مقادیر HSV را محاسبه می‌کنیم. خروجی این مقادیر به صورت زیر است.

```
hsv blue: [[[120 255 255]]]
hsv red: [[[  0 255 255]]]
hsv green: [[[ 60 255 255]]]
```

شکل 8- مقادیر **HSV** برای رنگ‌های آبی، قرمز و سبز

حال برای تعیین باند پایین Hue، مقدار HSV آن را منهای 10 و برای تعیین باند بالای آن، مقدار HSV آن را به اضافه 10 می‌کنیم. باند بالای Saturation و Value را 255 و باند پایین‌شان را 100 درنظر می‌گیریم و سپس آن را نمایش می‌دهیم که به صورت زیر است:



شکل 9- تصاویر باینری شده

حال تابعی به نام getOrientation تعریف می‌کنیم که دو ورودی تصویر و کانتور جسم را می‌گیرد و زاویه چرخش جسم را محاسبه و روی تصویر نمایش می‌دهد. برای این منظور، ابتدا بر اساس کانتور جسم و با استفاده از دستور cv2.PCACompte2 بردارهای ویژه را محاسبه کرده و سپس با دستور زیر زاویه جسم را به رادیان محاسبه می‌کنیم:

Angle = atan2(eigenvectors[0,1], eigenvector[0,0])

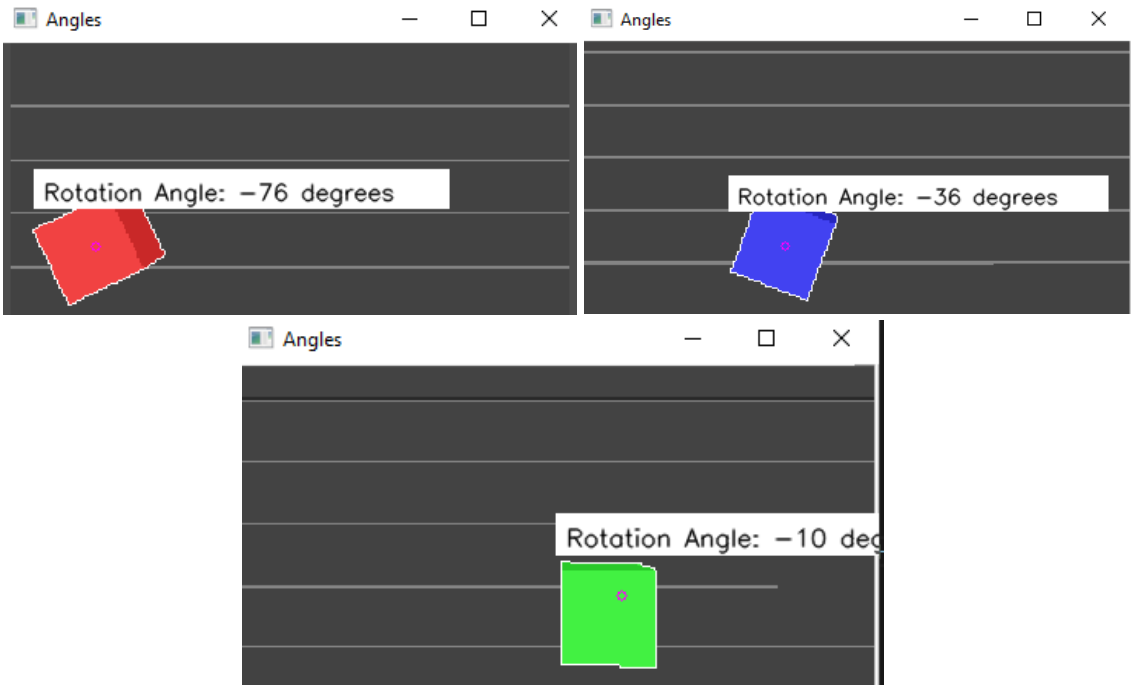
و سپس آن را با دستور زیر به درجه بین -180 تا 180 تغییر می‌دهیم:

Angle_deg = -int(np.rad2dwg(angle)) + 90

حال با د ستور cv2.rectangle تکت باک سی را نمایش داده و با د ستور cv2.putText مقدار زاویه را روی تکست باکس نمایش می‌دهیم. در نهایت هم این تابع مقدار زاویه به درجه را برمی‌گرداند.

سپس ما تابعی به ام orientation تعریف می‌کنیم که ورودی‌هایش تصویر و تصویر باینری شده هستند. سپس با دستور cv2.contours(binary image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE) کانتور مکعب‌ها را در ت صاویر باینری محا سبه کرده و سپس آن را ر سم کرده و تابع getOrientation را فراخوانی کرده تا هم تکست باکس حاوی متن مقدار زاویه را نمایش دهد و هم مقدار زاویه را برگرداند. در نهایت هم ت صویر ا صلی BGR را نمایش می‌دهیم تا مقدار زاویه روی آن م شخص شود. خروجی این تاع هم مدار زاویه چرخش است.

خروجی این قسمت را در شکل زیر مشاهده می‌کنید.



شکل 10- نمایش زاویه چرخش محاسبه شده به همراه تصاویر اصلی و کانتورشان

در ادامه تابع position را نوشتیم که ورودی‌هایش تصویر و تصویر باینری شده است. ایتدا با استفاده از دستور cv2.moment(binary image) مرکز جسم را مشخص می‌کنیم. Image Moment میانگین وزنی خاصی از شدت پیکسل‌های تصویر است که با کمک آن می‌توانیم برخی از ویژگی‌های خاص یک تصویر مانند شعاع، مساحت، مرکز و غیره را پیدا کنیم. برای یافتن مرکز تصویر، معمولاً آن را به فرمت باینری تبدیل می‌کنیم و سپس مرکز آن را پیدا می‌یابیم. حال پس از یافتن مرکز جسم، با یک دایره مکان آن را روی جسم نمایش داده و تکست باکسی برای نمایش مکان جسم تعیین کرده و روی تصویر اصلی نمایش می‌دهیم. خروجی این قسمت به صورت شکل زیر است:

شکل 11- نمایش مکان محاسبه شده برای هر مکعب روی تصویر **BGR** جسم

نکته مهم و قابل توجه این است که در اینجا چون سایز تصاویر یکسان نیست، مقدار y مربوط به مکان این مکعب‌ها با هم تفاوت ملموسی دارند؛ در حالی که با توجه به نگاه چشمی به آن‌ها درمیابیم که احتمالا مقدار yشان حدودا برابر است. برای این منظور می‌توان همه تصاویر را به یک سایز خاص ریسایز کرد و دید که مقادیر yشان تقریبا یکسان است.

بخش دوم، سوال ۲ – محاسبه ماتریس دوران

در این بخش با توجه به زوایای چرخش محاسبه شده در قسمت اول سوال، ماتریس دوران را برای هر مکعب محاسبه می‌کنیم. برای این منظور همانطور از تصاویر پیداست و در صورت پروژه هم ذکر شده، محور دوران را در راستای z در نظر گرفته و با توجه به فرمولی که در درس بدست آوردیم، ماتریس دوران را محاسبه می‌کنیم. فرم کلی ماتریس دوران حول محور z به صورت زیر است:

$$Q = \begin{bmatrix} \cos(phi) & -\sin(phi) & 0 \\ \sin(phi) & \cos(phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

فقط نکته مهم این است که باید زوایا به رادیان در این فرمول قرار بگیرند.

خروجی مربوط به این قسمت که ماتریس‌های دوران است، در شکل زیر آورده شده است:

```
Q_blue =
 [[ 0.80901699  0.58778525  0.        ]
 [-0.58778525  0.80901699  0.        ]
 [ 0.          0.          1.        ]]

Q_red =
 [[ 0.2419219   0.97029573  0.        ]
 [-0.97029573  0.2419219   0.        ]
 [ 0.          0.          1.        ]]

Q_green =
 [[ 0.98480775  0.17364818  0.        ]
 [-0.17364818  0.98480775  0.        ]
 [ 0.          0.          1.        ]]
```

شکل 12- ماتریس‌های دوران مجاسبه شده برای هر مکعب

کد مربوط به این دو سوال بخش دوم در فایل پایتون MP2_Q2.py موجود است.