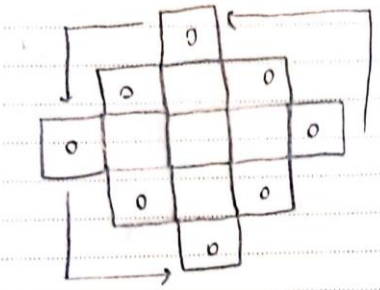
گزارش تمرین دهم بینایی ماشین

نام تهیه‌کننده: ملیکا نوبختیان

شماره دانشجویی: ۹۷۵۲۲۰۹۴

نسخه: ۱

۱- سوال اول



چون ۲۷۰ درجه سایکلرد چرخیده است
اگر بخواهیم آن را به حالت اول برگردانیم
باید ۶ خانه به سمت راست جابه جا
داشته باشیم.

0 : 00000000 → 0 : 00000000

34 : 00100010 → 136 : 10001000

143 : 1000 1111 → 227 : 11100011

247 : 1111 0111 → 253 : 11111101

بیشترین : 59 که داریم

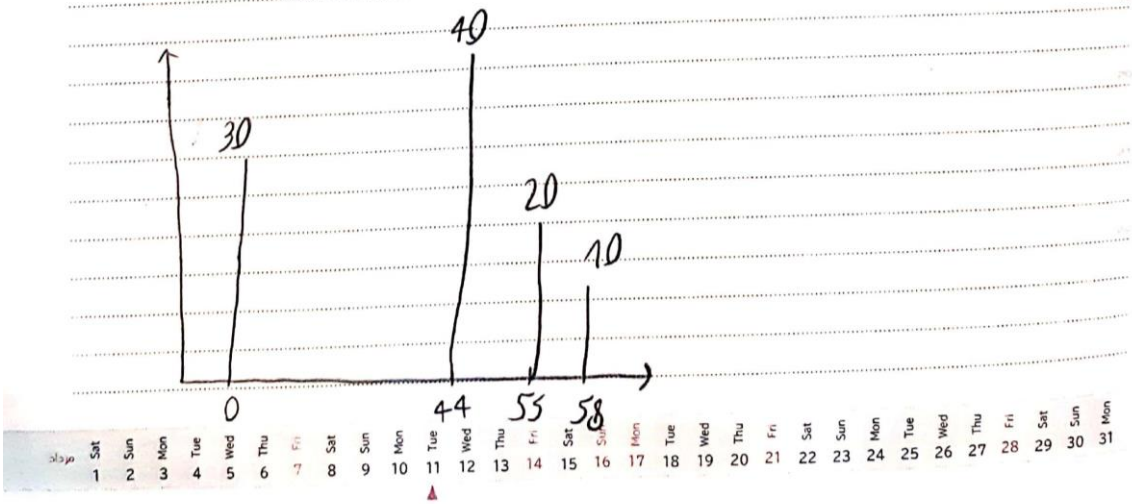
0 → 0

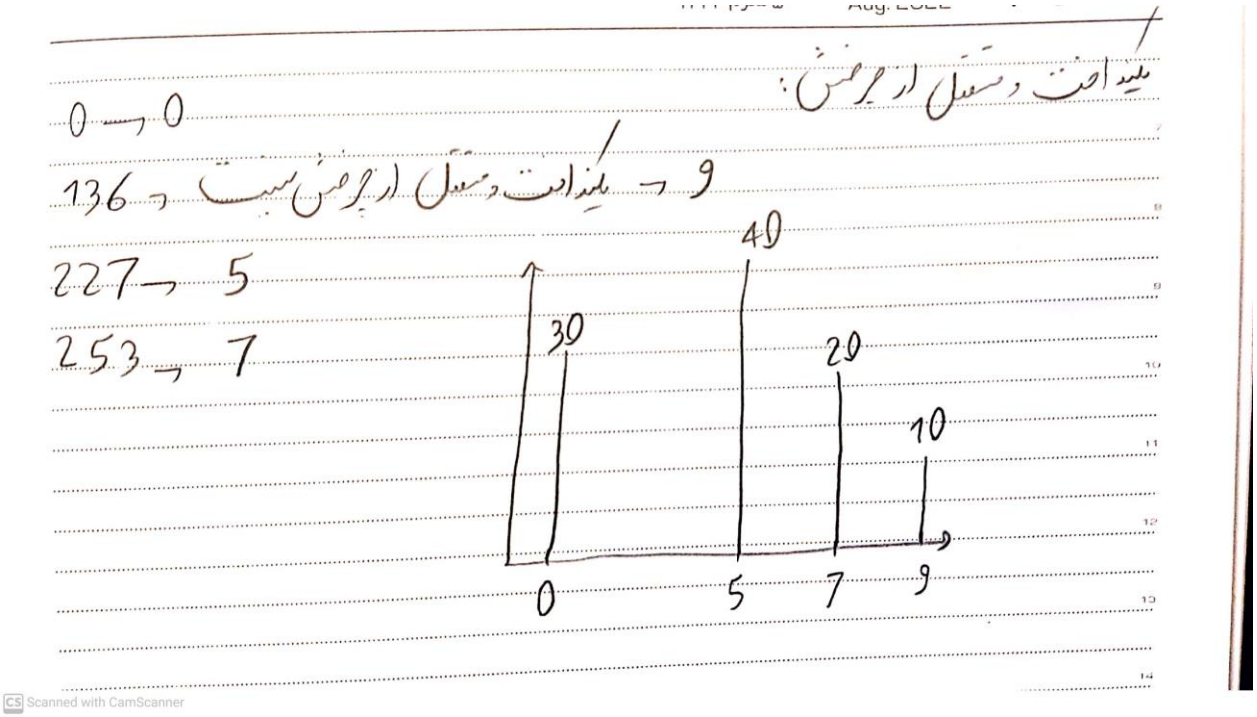
136 → بیشترین است → 58

227 → 44

253 → 55

* نصف کردن شدت روشنایی تصویری





۲- سوال دوم

در ابتدا برای اینکه بتوانیم contour موردنظر object را به دست بیاوریم تابع largest contour را تعریف می‌کنیم:

```
def largest_contour(image):
  gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
  blurred = cv2.GaussianBlur(gray, (5, 5), 0)
  thresh = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv
2.THRESH_BINARY, 21, 15)
  kernelSize = (5, 5)
  kernel = cv2.getStructuringElement(cv2.MORPH_RECT, kernelSize)
  opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel)
  contours, hierarchy= cv2.findContours(opening.copy(), cv2.RETR_EXTERNAL,
 cv2.CHAIN_APPROX_NONE)
  all_areas= []
  for cnt in contours:
      area= cv2.contourArea(cnt)
      all_areas.append(area)
  sorted_contours= sorted(contours, key=cv2.contourArea, reverse= True)

  return sorted_contours[0]
```

در ابتدا تصویر را به gray تبدیل می‌کنیم. سپس برای اینکه در مراحل بعد بتوانیم بهتر اشیا را تشخیص دهیم تصویر را blur می‌کنیم. سپس تصویر را با استفاده از adaptiveThreshold به یک تصویر با مقادیر ۰ و ۲۵۵ تبدیل

می‌کنیم. برای اینکه نویزهای تصویر به شکل بهتری حذف شوند یک عملگر opening به تصویر اعمال می‌کنیم تا نویزهای آن حذف شوند. سپس contour های تصویر را پیدا می‌کنیم و contour ها با توجه به مساحتی که دارند مرتب می‌کنیم. contour ای که بیشترین مساحت را داشته باشد به عنوان contour موردنظر ما انتخاب خواهد شد.

برای محاسبه compactness از فرمول زیر استفاده می‌کنیم:

$$Compactness = \frac{4\pi\ Area}{Perimeter^2}$$

تابع compactness به شکل زیر خواهد بود:

```
def compatness(image):
  contour = largest_contour(image)
  area = cv2.contourArea(contour)
  perimeter = cv2.arcLength(contour, True)
  compactness_score = (4 * np.pi * area) / (perimeter * perimeter)
  return compactness_score
```

در ابتدا contour را به دست می‌آوریم. سپس مساحت و محیط آن را به دست می‌آوریم و طبق فرمول مقدار compactness را به دست می‌آوریم.

برای محاسبه ecenticity از فرمول زیر استفاده می‌کنیم:

$$Eccentricity = \sqrt{1 - \left(\frac{MinorAxisLength}{MajorAxisLength}\right)^2}$$

تابع آن نیز به صورت زیر تعریف می‌شود:

```
def eccenticity(image):
  contour = largest_contour(image)
  (x, y), (minor_axis, major_axis), angle = cv2.fitEllipse(contour)
  eccentricity_score = math.sqrt(1 - (minor_axis / major_axis) * (minor_ax
is / major_axis))
  return eccentricity_score
```

برای اینکه minorAxisLength و MajorAxisLength را به دست آوریم از fitEllipse استفاده می‌کنیم. سپس طبق فرمول مقدار آن را حساب می‌کنیم.

برای محاسبه solidity از فرمول زیر استفاده می‌کنیم:

$$Solidity = \frac{Area}{ConvexArea}$$

تابع آن به صورت زیر خواهد بود:

```
def solidity(image):
  contour = largest_contour(image)
  area = cv2.contourArea(contour)
  hull = cv2.convexHull(contour)
  hull_area = cv2.contourArea(hull)
  solidity_score = float(area)/hull_area
  return  solidity_score
```

ابتدا مساحت contour را به دست می‌آوریم. سپس convex ای که contour ما را احاطه می‌کند به دست می‌آوریم و مساحت آن را نیز به دست می‌آوریم و امتیاز solidity را به دست می‌آوریم.

برای محاسبه histogram از LBP تابع را به شکل زیر تعریف می‌کنیم:

```
def histogram_of_LBP(image, numPoints, radius, eps=1e-7):
  gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
  lbp = local_binary_pattern(gray, numPoints, radius, method='uniform')
  n_bins = int(lbp.max() + 1)
  hist_values, _ = np.histogram(lbp, density=True, bins=n_bins, range=(0,
n_bins))
  return hist_values
```

تصویر را تبدیل به gray می‌کنیم و با local binary pattern ، lbp را به دست می‌آوریم. برای اینکه در هیستوگرام تعداد bin ها را به دست آوریم از بیشترین مقدار lbp به علاوه یک استفاده می‌کنیم. سپس histogram را به دست می‌آوریم و به عنوان خروجی برمی‌گردانیم.

برای اینکه ویژگی داده‌ها را به دست آوریم تابع موردنظر را به شکل زیر تعریف می‌کنیم:

```
def get_featureMatrix(data):
  train_count = len(data)
  feature_matrix = [None] * train_count
  for i in range(train_count):
    compactness_score = compatness(data[i])
    eccenticity_score = eccenticity(data[i])
    solidity_score = solidity(data[i])
    lbp = histogram_of_LBP(data[i], 16, 2)
    feature_vec = np.concatenate((np.array([compactness_score, eccenticity
_score, solidity_score]), lbp), axis=None)
    feature_matrix[i] = feature_vec

  return feature_matrix
```

برای هر داده مقدار سه امتیاز compactness, ecenticity و solidity را به دست می‌آوریم و سپس histogram lbp آن را نیز به دست می‌آوریم. سپس همه این ویژگی‌ها را با هم concat می‌کنیم و به عنوان ویژگی آن داده در نظر می‌گیریم.

ویژگی‌های داده train را به دست می‌آوریم و به شکل زیر svm را روی آن آموزش می‌دهیم:

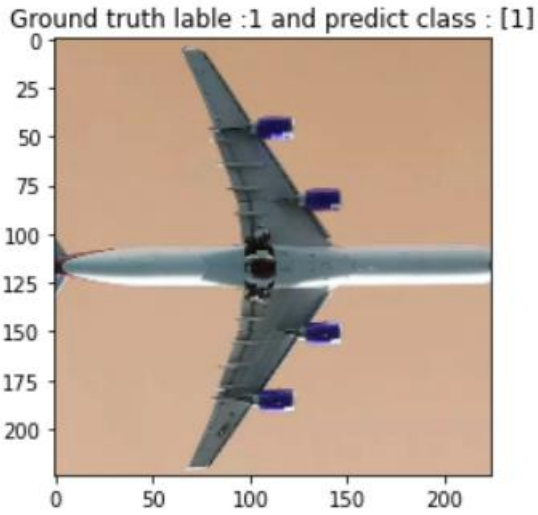
```
# model 1
feature_matrix_train = get_featureMatrix(x_train)
#determine classifier and train
clf = svm.SVC()
clf.fit(feature_matrix_train, y_train)
```

سپس مدلی که آموزش دادیم را روی داده‌های test آزمایش می‌کنیم و مقدار accuracy را به دست می‌آوریم:

```
1 #test on test dataset
2 feature_matrix_test = get_featureMatrix(x_test)
3 y_pred = clf.predict(feature_matrix_test)
4 accuracy_score(y_test, y_pred)
```

```
0.78125
```

در زیر نیز عملکرد دسته‌بند روی یکی از تصاویر را مشاهده می‌کنید:



منابع:

https://www.tutorialspoint.com/how-to-compute-the-area-and-perimeter-of-an-image-contour-using-opencv-python
https://docs.opencv.org/4.x/d1/d32/tutorial_py_contour_properties.html