Computer Vision
Dr. Mohammadi
Fall 2022
Hoorieh Sabzevari - 98412004
HW10

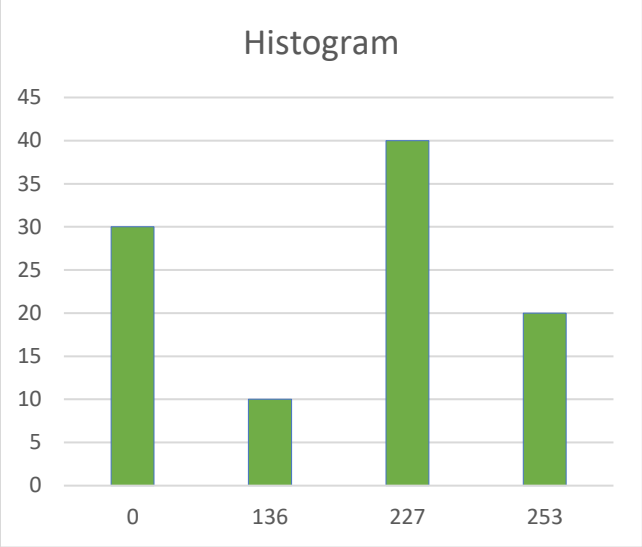
۱. در این سوال ابتدا مقادیر باینری ۴ عدد دسیمال داده شده را بدست می‌آوریم. چون تا ۸ نقطه در همسایگی را بررسی می‌کنیم، لذا مقدار باینری را تا هشت رقم می‌نویسیم.

$$(0)_{10} = (00000000)_2$$
$$(34)_{10} = (00100010)_2$$
$$(143)_{10} = (10001111)_2$$
$$(247)_{10} = (11110111)_2$$

همانطور که در صورت سوال ذکرشده، پیکسل‌ها به اندازه‌ی ۲۷۰ درجه به صورت ساعتگرد چرخانده شده‌اند، لذا برای رسیدن به حالت اولیه باید ۲۷۰ درجه به صورت پادساعتگرد بچرخیم یا می‌توانیم ۹۰ درجه به صورت ساعتگرد حرکت کنیم تا به جای اولیه برسیم. همچنین میزان روشنایی هر پیکسل نیز نصف شده که این مورد تاثیری در عملکرد LBP ندارد، چون بر اساس اختلاف میزان شدت روشنایی‌ها کار می‌کند که با نصف شدن باز هم همان نتیجه حاصل می‌شود.۹۰ درجه به صورت ساعتگرد معادل دوبار شیفت اعداد به سمت راست است. لذا داریم:

$$(0)_{10} = (00000000)_2 \rightarrow (00000000)_2 = (0)_{10}$$
$$(34)_{10} = (00100010)_2 \rightarrow (10001000)_2 = (136)_{10}$$
$$(143)_{10} = (10001111)_2 \rightarrow (11100011)_2 = (227)_{10}$$
$$(247)_{10} = (11110111)_2 \rightarrow (11111101)_2 = (253)_{10}$$

لذا هیستوگرام تصویر اولیه به صورت زیر بوده است:



طبق تعریف، کد یکنواخت مستقل از چرخش باید دو یا کمتر میزان تغییر را داشته باشد.

لذا کدهای ۰ ، ۲۲۷ و ۲۵۳ یکنواخت هستند. همچنین در هیستوگرام پس از چرخش نیز چون تعداد تغییرات ثابت می‌ماند، کدهای ۰ ، ۱۴۳ و ۲۴۷ یکنواخت هستند.

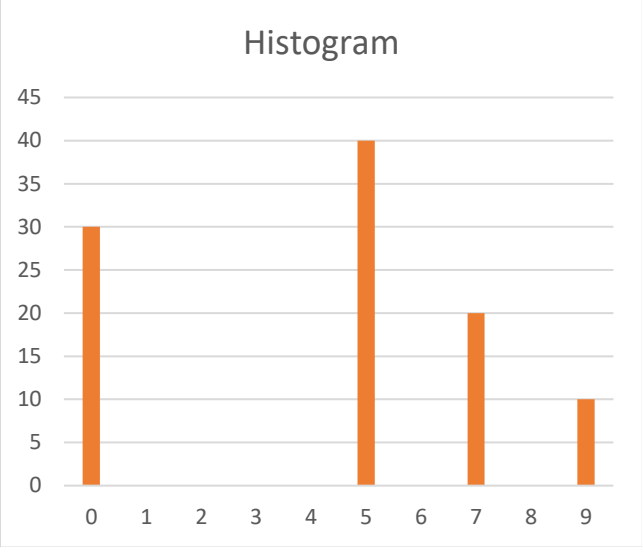
کد ۳۴ و ۱۳۶ نیز غیریکنواخت هستند.

در اینجا برای کدهای یکنواخت با تعداد ۱های متفاوت، کدی برابر با همان تعداد ۱ها قرار دادیم. مثلا ۲۲۷ که در باینری ۵ تا یک دارد کد ۵ می‌گیرد. بدین ترتیب هشت کد یا ۹ حالت پر می‌شود. یک حالت هم برای غیریکنواخت‌ها در نظر می‌گیریم که کد آن‌ها ۹ باشد. در مجموع کدهای ما از صفر تا ۹ خواهد بود که ۱۰ حالت مختلف را پوشش می‌دهد.

لذا برای تصویر بعد از چرخش داریم:

$$(0)_{10} = (00000000)_2 \rightarrow code0$$
$$(34)_{10} = (00100010)_2 \rightarrow code9$$
$$(143)_{10} = (10001111)_2 \rightarrow code5$$
$$(247)_{10} = (11110111)_2 \rightarrow code7$$

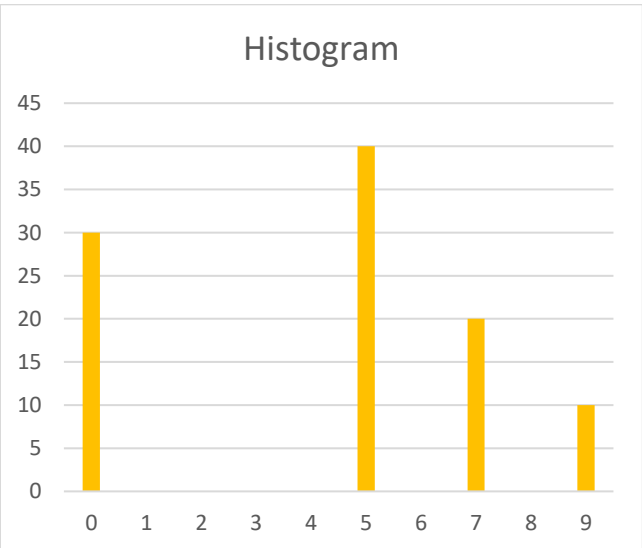
هیستوگرام LBP یکنواخت و مستقل از چرخش به صورت زیر خواهد شد:



همچنین برای تصویر اصلی نیز داریم:

$$(0)_{10} = (00000000)_2 \rightarrow code0$$
$$(136)_{10} = (10001000)_2 \rightarrow code9$$
$$(227)_{10} = (11100011)_2 \rightarrow code5$$
$$(253)_{10} = (11111101)_2 \rightarrow code7$$

هیستوگرام LBP یکنواخت و مستقل از چرخش به صورت زیر خواهد شد:



همانطور که واضح است هردو نمودار یکسان شد. زیرا نمودار مستقل از چرخش عمل کرده و با چرخشی که ما اعمال کردیم مقادیر آن تغییر نمی‌کند. همچنین نصف کردن شدت روشنایی هم در اینجا بی‌تاثیر است.

۲. الف) در این سوال ابتدا یک تابع shape_detector() تعریف می‌کنیم تا ابتدا تصویر را یک‌کاناله و باینری کرده و با استفاده از تابع findContours() شکل دارای بزرگ‌ترین مساحت موجود در تصویر را ریترن کند.

```
def shape_detector(image):
  # gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
  thresh = cv2.threshold(image, 100, 255, cv2.THRESH_BINARY)[1]
  cnts,_ = cv2.findContours(thresh, 1, 1)
  cnt = cnts[0]
  for c in cnts:
    area = cv2.contourArea(c)
    if area > cv2.contourArea(cnt):
      cnt = c
  return cnt
```

سپس تابع فشردگی را به صورت زیر تعریف می‌کنیم. مساحت شکل را با استفاده از تابع contourArea() و محیط آن را با استفاده از تابع arcLength() بدست آورده و در فرمول فشردگی قرار می‌دهیم.

```
def compatness(image):
  shape = shape_detector(image)
  area = cv2.contourArea(shape)
  primeter = cv2.arcLength(shape, True)
  compactness_score = np.divide((4 * np.pi * area), (np.square(primeter)))
  return compactness_score
```

تابع کشیدگی یا گریز از مرکز را طبق فرمول داخل اسلاید به صورت زیر تعریف می‌کنیم. برای پیدا کردن محور اصلی و محور فرعی شکل از تابع fitEllipse() استفاده می‌کنیم. (لینک کمکی)

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

```
def eccenticity(image):
  shape = shape_detector(image)
  (x,y),(MA, ma),angle = cv2.fitEllipse(shape)
  if MA > ma:
    MA, ma = ma, MA
  eccentricity_score = np.sqrt( 1 - np.square(MA / ma))
  return eccentricity_score
```

تابع صلب بودن را نیز به صورت زیر تعریف می‌کنیم. از تابع convexHull() برای پیدا کردن چندضلعی محدبی که بر شکل ما محیط شده است، استفاده می‌کنیم و مساحت آن را بدست آورده و مساحت شکل را بر مساحت چندضلعی تقسیم می‌کنیم. (لینک کمکی)

```
def solidity(image):
  shape = shape_detector(image)
  area = cv2.contourArea(shape)
  convex_hull = cv2.convexHull(shape)
  convex_hull_area = cv2.contourArea(convex_hull)
  solidity_score = float(area) / convex_hull_area
  return solidity_score
```

ب) با استفاده از تابع local_binary_pattern() و متد LBP یکنواخت و تابع histogram() هیستوگرام LBP را بدست آورده و سپس آن را نرمالایز کرده و مقدار اپسیلون را به آن اضافه می‌کنیم.

```
def histogram_of_LBP(image, numPoints, radius, eps=1e-7):
  lbp = local_binary_pattern(image, numPoints, radius, method="uniform")
  (hist, _) = np.histogram(lbp.ravel(), density=True, bins=256, range=(0, 256))
  hist = hist.astype("float")
  hist /= (hist.sum() + eps)
  return hist
```

تابع اعتبارسنجی برای تست کردن توابع پیشین با استفاده از ۲ نمونه‌ی دلخواه:

```
def validating_func(image_ship_path, image_airplane_path):
  ship_img = cv2.imread(image_ship_path, 0)
  airplane_img = cv2.imread(image_airplane_path, 0)
  eccenticity_ship, eccenticity_airplane = eccenticity(ship_img), eccenticity(airplane_img)
  compatness_ship, compatness_airplane = compatness(ship_img), compatness(airplane_img)
  solidity_ship, solidity_airplane = solidity(ship_img), solidity(airplane_img)
  print(f"Result for ship image:\ncompatness is : {compatness_ship}\t\teccenticity is : {eccenticity_ship}\t\tsolidity is : {solid
  print(f"Result for airplane image:\ncompatness is : {compatness_airplane}\t\teccenticity is : {eccenticity_airplane}\t\tsolidity
```

```
validating_func('/content/1412936.jpg', '/content/airplane153.jpg')
```

```
Result for ship image:
compatness is : 0.3900744545613469              eccenticity is : 0.8901555055181084             solidity is : 0.9471560558472225
Result for airplane image:
compatness is : 0.29043100592552473             eccenticity is : 0.8066844957099251             solidity is : 0.9043478216799644
```

ج) دیتاست را لود و split می‌کنیم.

د) سپس برای هر تصویر موجود در مجموعه‌ی آموزشی ویژگی‌های فشردگی، کشیدگی، صلب بودن و هیستوگرام LBP را محاسبه کرده و به ترتیب به هر سطر ماتریس ویژگی‌ها اضافه می‌کنیم.

```
def get_featureMatrix(data):
  feature_matrix = []
  for im in data:
    x = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    c = compatness(x)
    e = eccenticity(x)
    s = solidity(x)
    h = histogram_of_LBP(x, 8, 1, eps=1e-7)[0]
    feature_matrix.append([c, e, s, h])
  return feature_matrix
```

سپس از دسته‌بند svm() استفاده کرده و ماتریس ویژگی‌های x_train را بعنوان ورودی و y_train را بعنوان کلاس هر داده به دسته‌بند می‌دهیم تا با استفاده از این ویژگی‌ها آموزش ببیند.

```
# model 1
x_train = np.array(x_train)
feature_matrix_train = np.array(get_featureMatrix(x_train))

#determine classifier and train
y_train = np.array(y_train)
svm = LinearSVC()
svm.fit(feature_matrix_train, y_train)
```

ح) حال ماتریس ویژگی داده‌های تست را به ورودی تابع svm.predict() داده تا کلاس مربوط به هر x_test را برای ما پیش‌بینی کند. این مقادیر را در متغیر predicted_labels ذخیره می‌کنیم. در انتها با استفاده از تابع accuracy_score() برچسب‌های پیش‌بینی‌شده توسط دسته‌بند و برچسب‌های واقعی را با هم مقایسه کرده و دقت خود را بدست می‌آوریم. در این مسئله دقت ما حدود ۷۲٪ گزارش شد که نسبتا خوب است.

```
#test on test dataset
x_test = np.array(x_test)
y_test = np.array(y_test)
feature_matrix_test = np.array(get_featureMatrix(x_test))
predicted_labels = svm.predict(feature_matrix_test)
print("accuracy: ",accuracy_score(y_test, predicted_labels))
```

```
accuracy:  0.71875
```

خ) در انتها عملکرد دسته‌بند را بر روی یک نمونه‌ی آزمایشی دلخواه مشاهده می‌کنیم.

```
#test visualize
index = random.randint(0, len(x_test)-1)
# prediction = clf.predict(get_featureMatrix(np.array([x_test[index]])))
plt.title(f"Ground truth lable :{y_test[index]} and predict class : {predicted_labels[index]}")
plt.imshow(x_test[index])
plt.show()
```