۱. نصف کردن شدت روشنایی پیکسل‌ها تاثیری در LBP ندارد چون با فرض اینکه بعد از نصف کردن گردکردن انجام ندهیم ترتیب شدت روشنایی‌ها حفظ می‌شود و وضعیت شدت روشنایی دو پیکسل مختلف قبل و بعد از نصف کردن حفظ می‌شود. پس فقط تغییر حاصل از چرخش را به حالت اول برمی‌گردانیم. فرض می‌کنیم تصویر اولیه ۲۷۰ درجه در جهت عقربه‌های ساعت چرخیده باشد که معادل است با ۹۰ درجه چرخش در جهت عکس عقربه‌های ساعت پس ما کد LBPهای داده شده ۹۰ درجه در جهت چرخش عقربه‌های ساعت می‌چرخانیم تا به LBPهای موجود در تصویر اولیه برسیم.

0	0	0
0		0
0	0	0

$0 = (0000\ 0000)_2$

همان‌طور که مشخص است چرخش بر روی کد LBP صفر هیچ تاثیری ندارد.

0	0	1
0		0
1	0	0

$34 = (0010\ 0010)_2$

پس قبل از چرخش کد LBP این پیکسل‌ها از ماتریس زیر بدست می‌آید:

1	0	0
0		0
0	0	1

که برابر است با: $(1000\ 1000)_2 = \underline{136}$

1	0	0
1		0
1	1	1

$143 = (1000\ 1111)_2$

پس قبل از چرخش کد LBP این پیکسل‌ها از ماتریس زیر بدست می‌آید:

1	1	1
1		0
1	0	0

که برابر است با: $(1110\ 0011)_2 = \underline{227}$

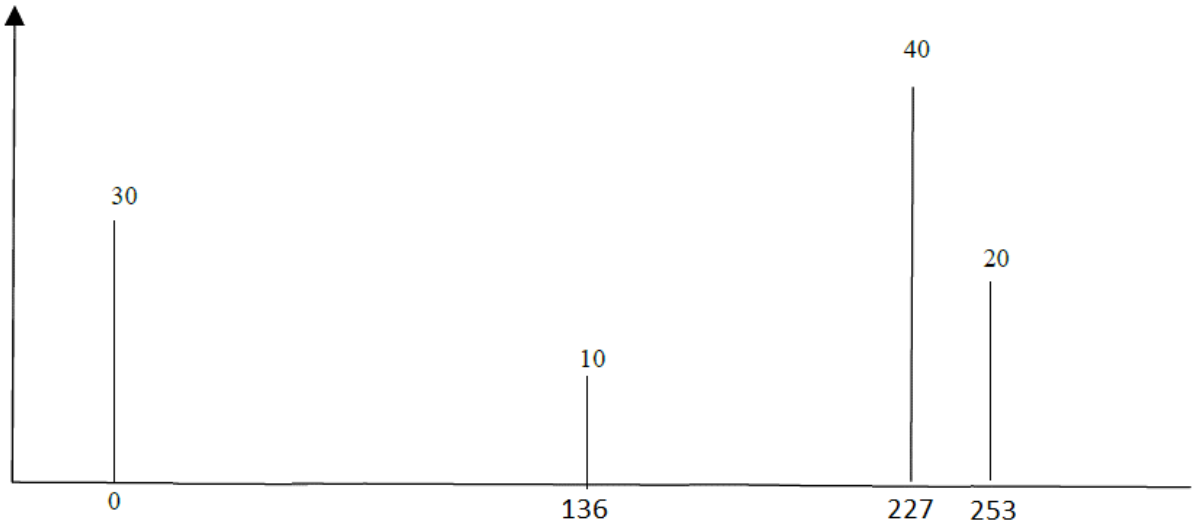
$247 = (1111\ 0111)_2$

1	1	1
1		1
1	1	0

پس قبل از چرخش کد LBP این پیکسل‌ها از ماتریس زیر بدست می‌آید:

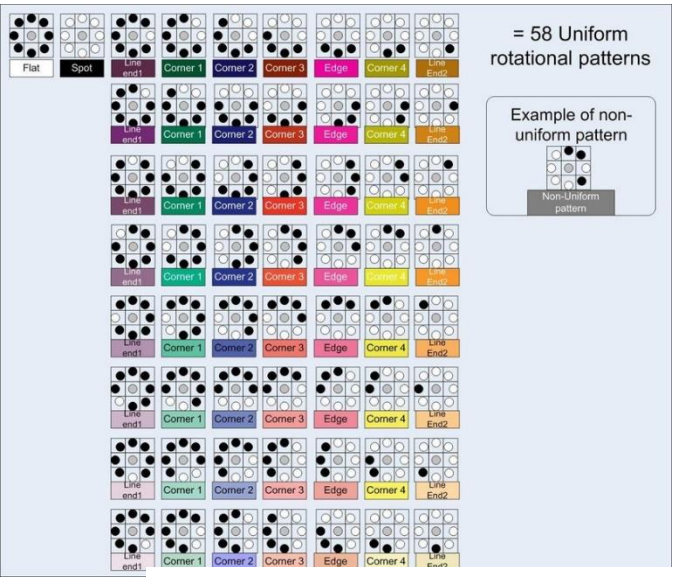
که برابر است با: $\underline{253} = (1111\ 1101)_2$

1	1	1
1		1
0	1	1



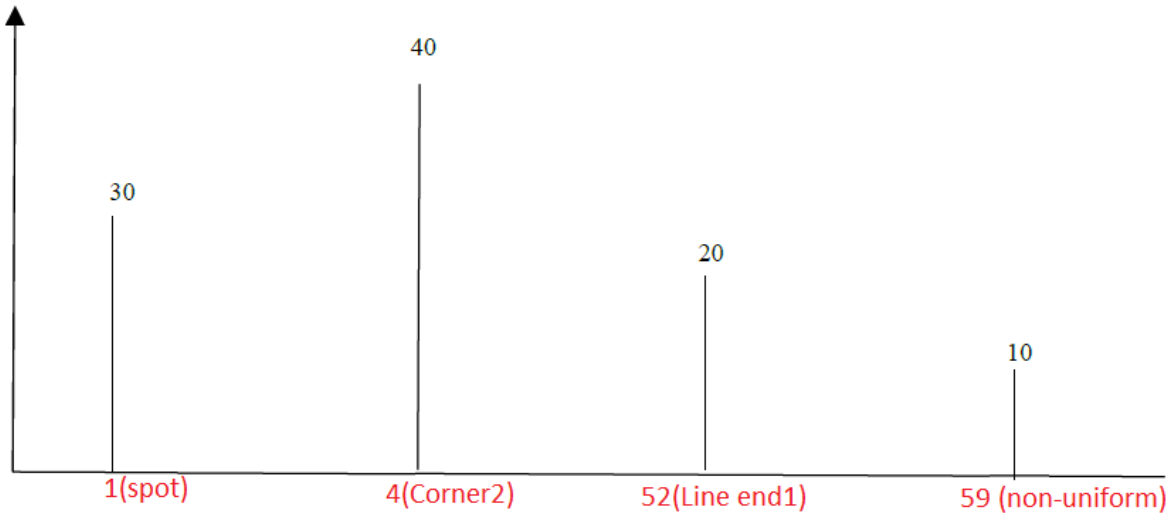
هیستوگرام LBP تصویر اصلی بدون چرخش و بدون تغییر شدت روشنایی

هیستوگرام LBP یکنواخت

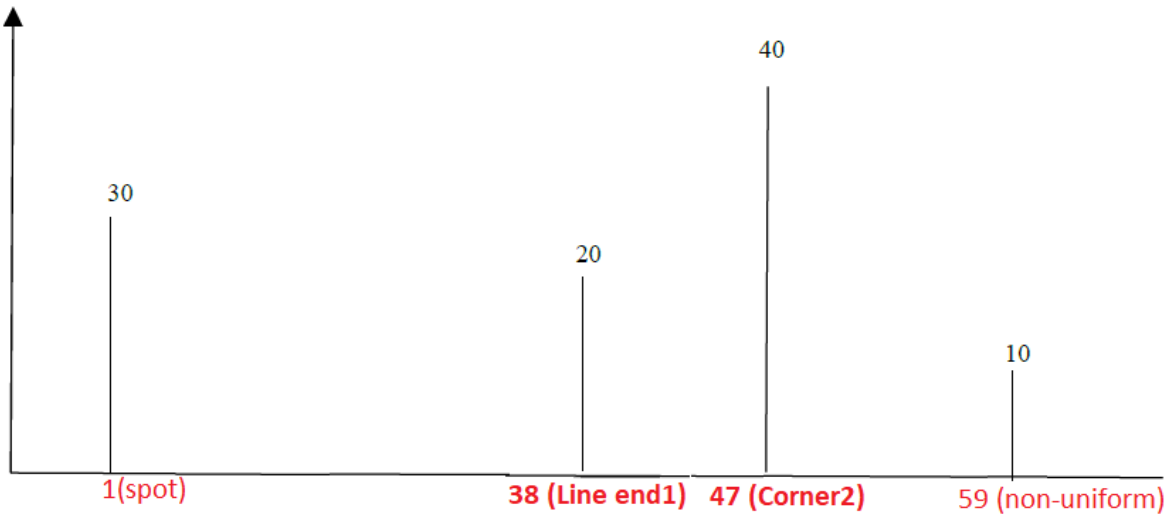


برای رسم هیستوگرام LBP یکنواخت در هر دو حالت فقط مقدارهای LBP ۳۴ و ۱۳۶ کد 59 یا غیریکنواخت می‌گیرند چون الگوی یکنواخت ندارند بقیه پیکسل‌ها که الگوی یکنواخت دارند هم متناسب با کد یکنواخت کد می‌گیرند و رسم می‌شوند. فرض می‌کنیم ترتیب کدهای یکنواخت به صورتی که در شکل روبرو آمده، باشند.

همانطور که در هیستوگرام‌های رسم شده مشخص است برای ۱ که یک الگوی متقارن است کد LBP متقارن تغییری در دو هیستوگرام نکرده همچنین کد ۵۹ که یک الگوی غیریکنواخت بود هم تغییری نکرده ولی دو الگوی دیگر ترتیبشان در دو هیستوگرام جابجا شده است.

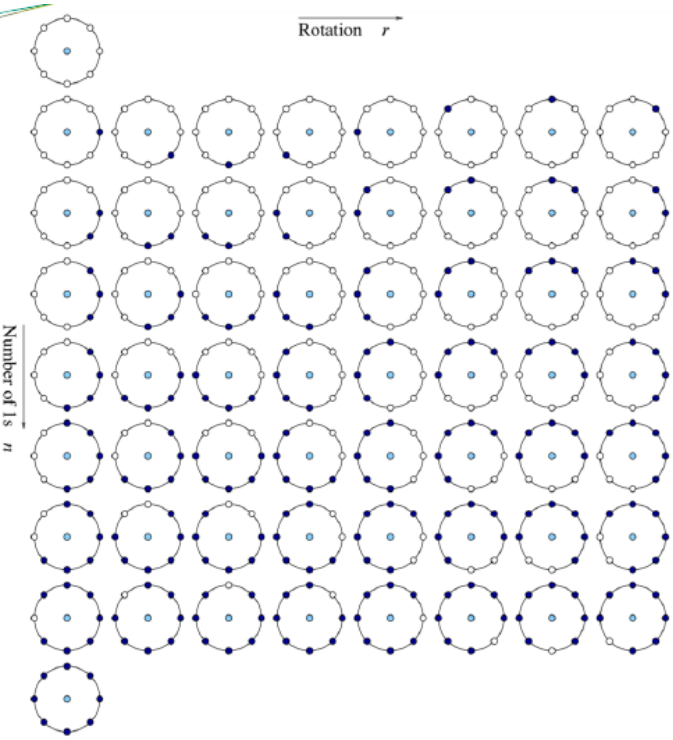


هیستوگرام LBP یکنواخت تصویر بعد از چرخش و تغییر شدت روشنایی



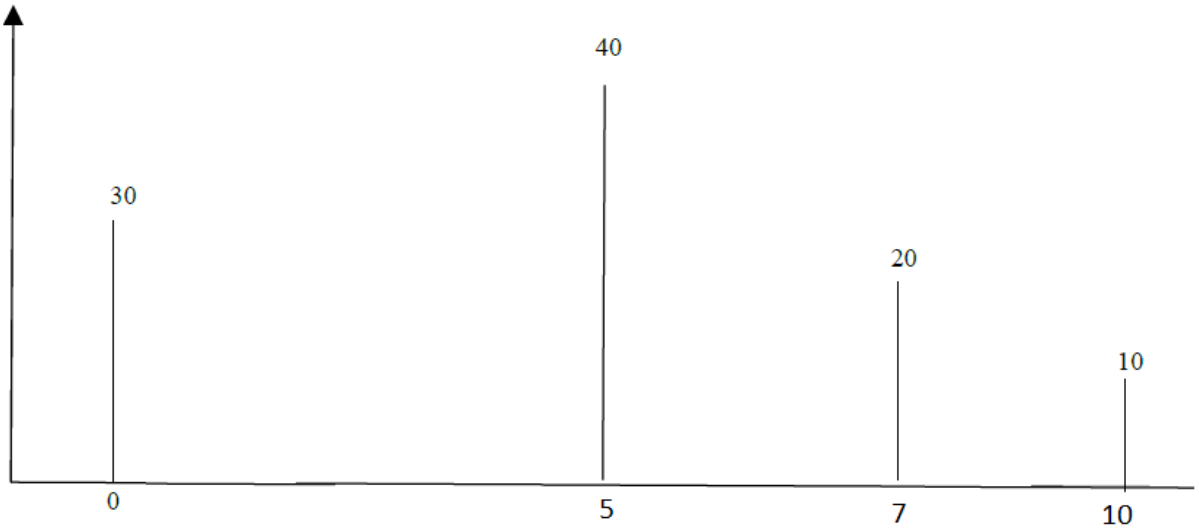
هیستوگرام LBP یکنواخت تصویر اصلی

کد LBP مستقل از چرخش



فرض می‌کنیم ترتیب کدهای مستقل از چرخش به صورتی که در شکل روبرو آمده، باشند و همچنین برای کدهای غیریکنواخت کد ۱۰ را در نظر می‌گیریم.

هیستوگرام LBP مستقل از چرخش برای هر دو حالت یکسان بدست خواهد آمد چون تغییر شدت روشنایی که تاثیری ندارد و چرخش تصویر هم در کد LBP مستقل از چرخش تاثیری ندارد پس:



کد LBP مستقل از چرخش برای تصویر اصلی و تصویر چرخانده شده

۲. الف) قبل از بدست‌آوردن هر یک از ویژگی‌ها ابتدا تصویر ورودی تابع را با روش adaptive باینری کردم و سپس با اعمال عملگرهای باز و بسته حفره‌ها و جزئیات اضافی کوچک را حذف کردم. سپس کانتورهای تصویر را ذخیره و برای بزرگترین کانتور پارامترهای خواسته شده را بدست آوردم. برای مثال محاسبه‌ی ویژگی اول (فشردگی) در تصویر زیر آمده است. کد بقیه ویژگی‌ها نیز در نوتبوک قرار دارد.

Part1.compatness

```
def compatness(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    adaptive_output = cv2.adaptiveThreshold(
        gray, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY_INV, 55, 15
    )
    opening_output = cv2.morphologyEx(adaptive_output, cv2.MORPH_OPEN, cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3)))
    closing_output = cv2.morphologyEx(opening_output, cv2.MORPH_CLOSE, cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3)))
    contours, hierarchy = cv2.findContours(closing_output, cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)
    biggest_contour = sorted(contours, key = cv2.contourArea, reverse = True)[0]
#     contours_drawing = np.zeros_like(closing_output)
#     cv2.drawContours(contours_drawing, [biggest_contour], -1, (255,0,0), 3)
#     plt.imshow(contours_drawing)
#     plt.show()
    compactness_score = 4 * math.pi * cv2.arcLength(biggest_contour, True) / cv2.contourArea(biggest_contour, False)
    return compactness_score
# compatness(cv2.imread("dataset\\ship\\613378.jpg"))
```

```
2.731171935420989
```

ب)

Part4.LBP

```
def histogram_of_LBP(image, numPoints, radius, eps=1e-7):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    LBPs = feature.local_binary_pattern(gray, numPoints, radius)
    n = int(LBPs.max() + 1)
    hist_values, _ = np.histogram(LBPs, bins=n, range=(0, n))
    #plt.plot(hist_values)
    return hist_values
#histogram_of_LBP(cv2.imread("dataset\\ship\\613378.jpg"), 16, 2)
```

بعد از سیاه و سفید کردن تصویر و استفاده از تابع local_binary_pattern بیشینه کد LBPهای مختلف را محاسبه و در n ریختم و در نهایت با تابع np.histogram هیستوگرام گرفتم و خروجی دادم.

ج) برای اینکه بصورت local در سیستم خودم اجرا کنم آدرس‌های این بخش را تغییر دادم.

```
x_data , y_data = [], []
for img_path in images_paths:
    x_data.append(cv2.resize(cv2.imread(img_path), (224,224)))
    if img_path.split("\\")[1] == "ship":
        y_data.append(0)
    else:
        y_data.append(1)
```

```
!gdown --id 1O_VkcGlWRR6h00VT8nXIgeI3RyvBpXbE
!unzip dataset.zip
classes = os.listdir("dataset")
images_paths = glob.glob("dataset\\*\\*")
```

د) ماتریس گفته شده را مطابق روش گفته شده و به ازای LBP^2_{16} ایجاد کردم. در نهایت از SVM استفاده کردم و داده‌های تست را به ازای ویژگی‌های استخراج شده fit کردم.

Part6. Exctract features and determine classifier

```
def get_featureMatrix(data):
    feature_matrix = [[compatness(img), eccenticity(img), solidity(img), *histogram_of_LBP(img, 16, 2)] for img in data]
    return feature_matrix
```

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

ج،خ)

دقت مدل آموزش دیده بیش از ۷۸٪ است و خوشبختانه روی اکثر تصاویر تست خروجی صحیح می‌دهد.

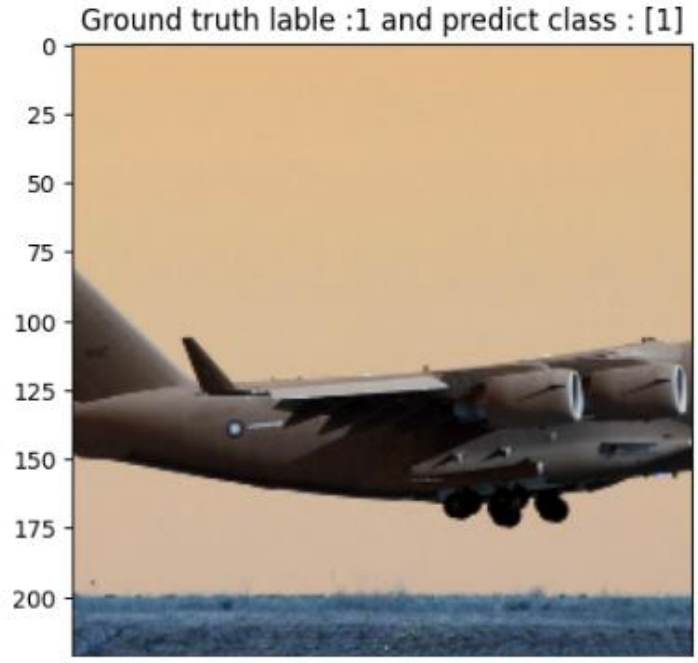
Part7

In [147]:
```
#test on test dataset
y_pred = clf.predict(get_featureMatrix(x_test))
accuracy_score(y_test, y_pred)
```

Out[147]: 0.78125

Part8

In [148]:
```
#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 : {prediction}")
plt.imshow(x_test[index])
plt.show()
```



منابع

سوال ۲)

https://docs.opencv.org/4.x/d1/d32/tutorial_py_contour_properties.html

https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_local_binary_pattern.html