Computer vision

Homework 10

Parsa Eissazadeh 97412364

سوال 1

ابتدا کد باینری این اعداد را میسازیم :

34 : 0010 0010

143 : 1000 1111

247 : 1111 0111

حال بر حسب این کد ها الگوهایشان را استخراج میکنیم و 90 درجه می چرخانیم تا به الگو های عکس اول برسیم .

شدت روشنایی تاثیری در LBP ندارد (؟؟) در نتیجه شدت روشنایی نیازی به بررسی ندارد .

الگو های عکس چرخیده شده :

1	1	1
1		1
1	1	0

247

1	0	0
1		0
1	1	1

143

0	0	1
0		0
1	0	0

34

الگو های عکس اصلی :

1	1	1
1		1
0	1	1

247

1	1	1
1		0
1	0	0

143

1	0	0
0		0
0	0	1

34

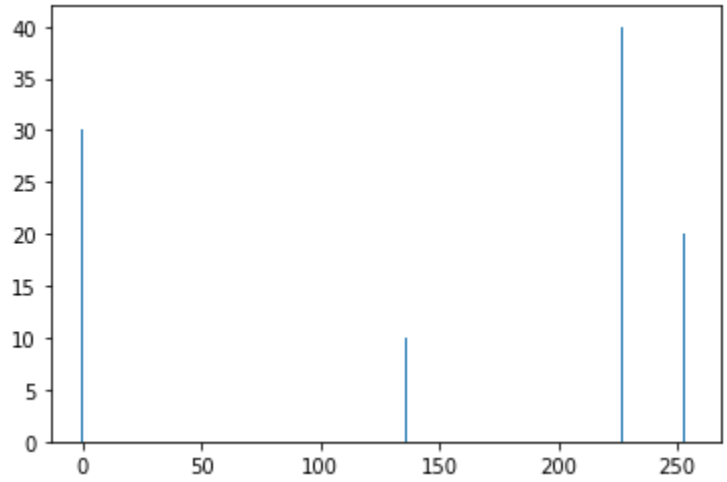
در این الگوهای عکس اصلی ، اعداد تفاوت میکنند .

253 ← 247

227 ← 143

136 ← 34

0 هم در اثر هر چرخشی همچنان 0 باقی خواهد ماند در نتیجه LBP عکس اول بدین شکل خواهد بود :



LBP مستقل از چرخش و یکنواخت

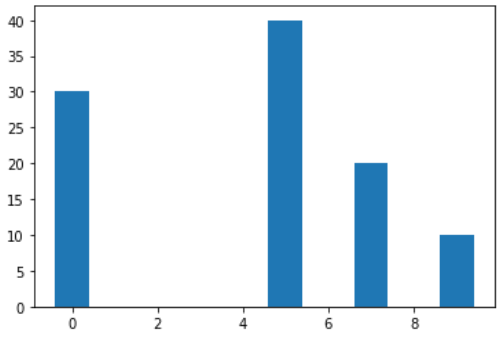
باید به الگوهایی که در اثر چرخش شبیه هم می شوند یک کد اختصاص دهیم . هیچ کدام از این الگو ها در اثر چرخش به هم تبدیل نمی شوند در نتیجه کد یکسان نخواهند داشت . کد هر کدام را بر اساس تعداد پیکسل های 1 ای که دارند حساب میکنیم .

7 ← 247

5 ← 143

34 ← یکنواخت نیست در نتیجه به کد دهم map می شود .

0 ← 0



از آنجایی که این کد گذاری مستقل از چرخش است ، در دو حالت عکس یکسان است .

سوال 2

محاسبه فشردگی : برای محاسبه این مقدار نیاز داریم که محیط و مساحت یک شکل را پیدا کنیم که هر دو توسط متد های openCV قابل محاسبه هستند .

1. contourArea : برای مساحت
2. arcLength : برای محیط

شعاع دایره هم محیط با شکل را به دست میاوریم و بر حسب آن مساحت دایره را . سپس مساحت شکل را به آن تقسیم میکنیم :

```
[29] def compatness(cnt):
       area = cv2.contourArea(cnt)
       perimeter = cv2.arcLength(cnt, False)

       compactness_score = np.divide(4 * np.pi * area , np.square(perimeter))
       return compactness_score
```

این کد را روی برگ اعمال کردیم و نتیجه شد عدد 0.20 که به مقدار واقعی فشردگی عکس برگ (طبق اسلاید ها) بسیار نزدیک است .

محاسبه صلبیت :

برای محاسبه صلب بودن عکس باید مساحت شکل عکس را به مساحت کوچکترین شکل محدب حاطی بر شکل تقسیم کنیم .کوچکترین شکل محدب محیط بر شکل هم توسط متد cv2.convexHull قابل به دست آوردن است .

```python
def solidity(cnt):

    area = cv2.contourArea(cnt)
    hull = cv2.convexHull(cnt)
    hull_area = cv2.contourArea(hull)
    if hull_area > 0 :
      solidity = float(area)/hull_area
    else :
      solidity = 0
    equi_diameter = np.sqrt(4*area/np.pi)

    return  solidity
```

محاسبه کشیدگی :

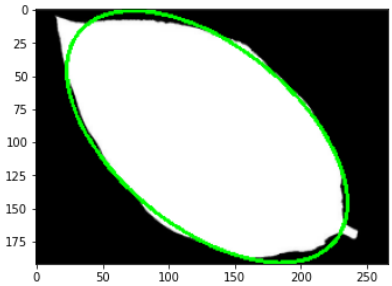
نیاز داریم محور های اصلی و فرعی عکس را حساب کنیم و بر هم تقسیم کنیم (کوچکتر را به بزرگتر تا نتیجه کمتر از 1 باشد) به کمک توابع openCV شی را به یک بیضی باید تبدیل کنیم که محور های اصلی و فرعی را درونش دارد .

```python
def eccenticity(cnt):

    ell = cv2.fitEllipse(cnt)
    centers , axes,orientation = ell

    cv2.ellipse(im ,ell,(0,255,0),2)
    plt.imshow(im)


    eccentricity_score = axes[0] / axes[1]
    if eccentricity_score > 1 :
      eccentricity_score = 1 / eccentricity_score
    return eccentricity_score
```



نتیجه همه این توابع با عددهای اسلاید چک شده است .

منابع :

https://www.tutorialspoint.com/how-to-find-the-solidity-and-equivalent-diameter-of-an-object-in-an-image-using-opencv-python

کشیدن lbp : از متدی که در متن سوال به آن اشاره شد استفاده میکنیم . سپس توسط متد هیستوگرام numpy ، هیستوگرامش را حساب میکنیم .

هر عکس در دیتاست باید :

1. تک کاناله بشود
2. باینری بشود
3. بزرگترین کانتورش پیدا شود

برای ساختن feature matrix ، ابتدا یک آرایه به نام row میسازیم ، مقادیر کشیدگی ، صلب بودن و فشردگی را درون آن میریزیم . سپس آنرا با هیستوگرام lbp عکس ، concat میکنیم . مجموعه ای از این row ها feature_matrix ما را میسازد .

در نهایت یک شی از `svm.LinearSVC` میسازیم و این feature matrix را به constructor اش پاس میدهیم.

```
# model 1
feature_matrix_train = get_featureMatrix(x_train)
#determine classifier and train

lin_clf = svm.LinearSVC(multi_class='ovr', max_iter=1000)
lin_clf.fit(feature_matrix_train,y_train)
```

```
#test on test dataset
prediction = lin_clf.predict(get_featureMatrix(x_test))

accuracy_score(y_test , prediction)
```

```
0.71875
```

همانطور که مشاهده میشود به دقت 70 درصد رسیده است .