به نام خدا

نام: فاطمه زهرا بخشنده **استاد: دکتر محمدرضا محمدی**

شماره دانشجویی: 98522157 **درس مبانی بینایی کامپیوتر**

گزارش تمرین 10:

سوال اول:

با فرض این که اعداد محدودیت ندارند، دو برابر کردن روشنایی تاثیری در مقادیر LBP ندارد، اما چرخش تصویر در این مقادیر موثر است. وقتی تصویر را 270 درجه ساعتگرد بچرخانیم، کد ها به صورت 6 واحد شیفت به راست، تغییر پیدا می کنند. پس برای بدست آوردن هیستوگرام LBP_8^2 تصویر اصلی، باید کد ها را 6 واحد به چپ شیفت دهیم.

4 مقدار LBP در تصویر موجود است که روی هر کدام این تغییر را اعمال می کنیم:

عدد 0 که نسبت به چرخش مقاوم است و همان 0 می ماند.

$$0 = (00000000)_2 \rightarrow shifted\ to\ left\ 6\ times: (00000000)_2 = 0$$

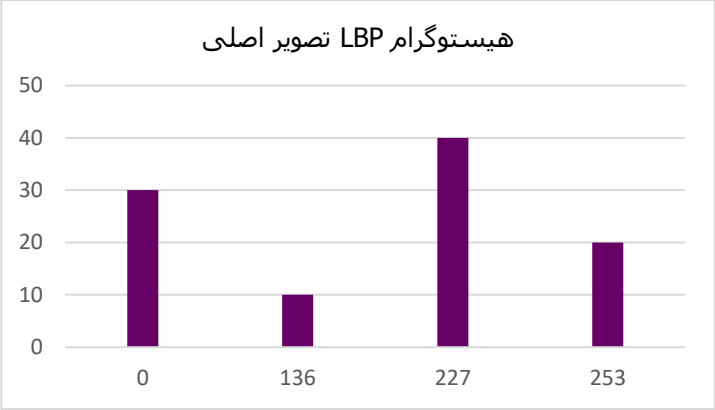
$$34 = (00100010)_2 \rightarrow shifted\ to\ left\ 6\ times: (10001000)_2 = 136$$

$$143 = (10001111)_2 \rightarrow shifted\ to\ left\ 6\ times: (11100011)_2 = 227$$

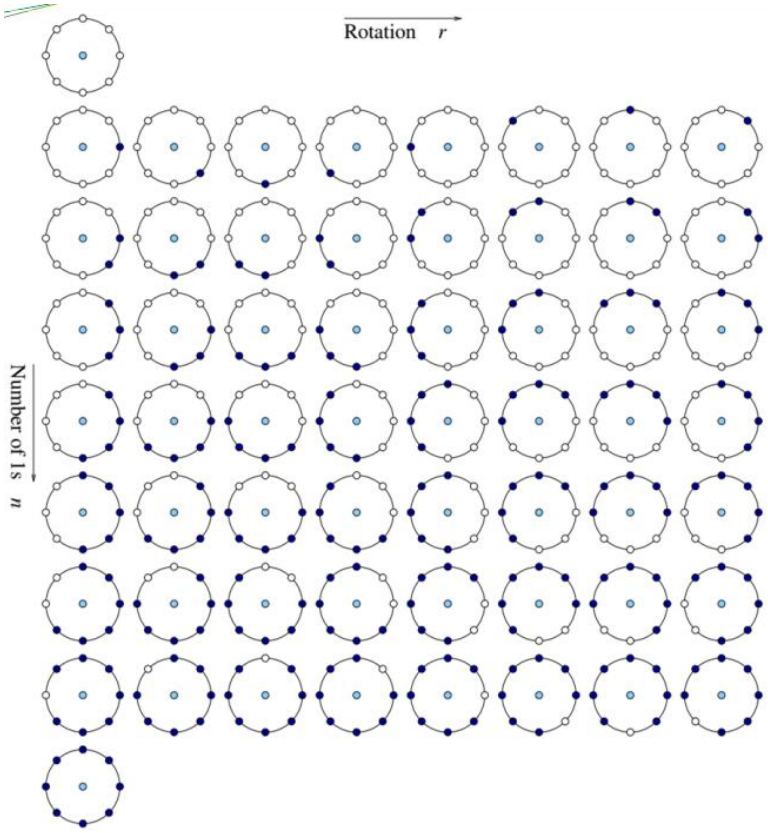
$$247 = (11110111)_2 \rightarrow shifted\ to\ left\ 6\ times: (11111101)_2 = 253$$

هیستوگرام LBP_8^2 تصویر اصلی به صورت زیر است:

مقدار LBP	0	136	227	253
تعداد	30	10	40	20



LBP یکنواخت: برخی از کدهای LBP مربوط به یک الگوی مشخص (مانند گوشه) هستند اما برخی الگوهای دیگر رفتار منظمی ندارند. به الگوهایی که بیش از 2 تغییر بین صفر و یک داشته باشند، غیریکنواخت گفته میشود. در LBP هشت نقطه ای تعداد الگوهای یکنواخت 58 عدد است و 198 الگو غیریکنواخت وجود دارد. برای LBP یکنواخت، بجای 256 کد، از 59 کد استفاده میشود (یک کد برای تمام الگوهای غیریکنواخت)، که می توان آن را با 6 بیت ذخیره کرد. هر 4 مقدار هیستوگرام را بررسی می‌کنیم و از روی شکل زیر عدد مربوط به کد LBP_8^2 یکنواخت آن ها را پیدا می کنیم.



برای تصویر چرخش یافته:

$0 = (00000000)_2 \rightarrow$ یکنواخت : $0 = (000000)_2$

$136 = (10001000)_2 \rightarrow$ غیریکنواخت : $58 = (111010)_2$

$227 = (11100011)_2 \rightarrow$ یکنواخت : $38 = (100110)_2$

$253 = (11111101)_2 \rightarrow$ یکنواخت : $56 = (111000)_2$

برای تصویر اصلی:

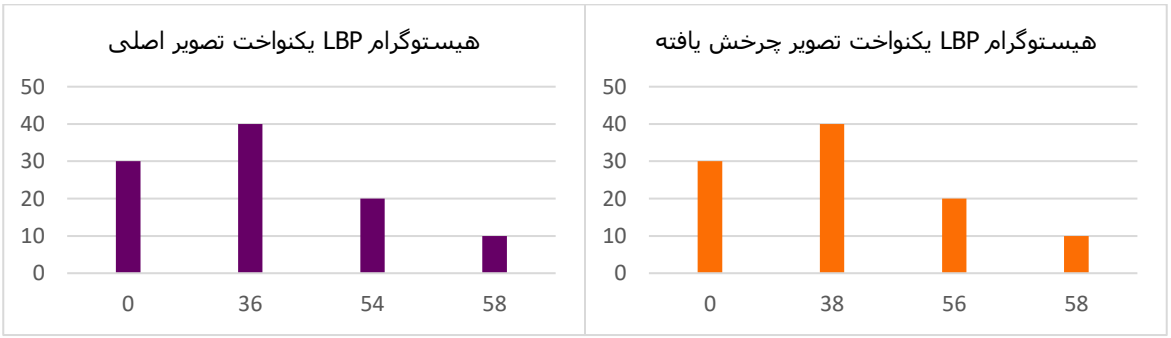
$0 = (00000000)_2 \rightarrow$ یکنواخت : $0 = (000000)_2$

$34 = (00100010)_2 \rightarrow$ غیریکنواخت : $58 = (111010)_2$

$143 = (10001111)_2 \rightarrow$ یکنواخت : $36 = (100100)_2$

$247 = (11110111)_2 \rightarrow$ یکنواخت : $54 = (110110)_2$

همانطور که می بینیم با چرخش تصویر، کد LBP پیکسل ها تغییر می کند اما یکنواخت یا غیر یکنواخت بودن الگوی آن ها عوض نمی شود.



LBP یکنواخت مستقل از چرخش: در مجموع 9 کد یکنواخت مستقل از چرخش در LBP با 8 همسایه خواهیم داشت، و یک کد نیز برای همه الگوهای غیریکنواخت در نظر گرفته می‌شود. که می توان با 4 بیت آن ها را ذخیره کرد.

برای تصویر چرخش یافته:

$0 = (00000000)_2 \rightarrow$ یکنواخت : $0 = (0000)_2$

$136 = (10001000)_2 \rightarrow$ غیریکنواخت : $9 = (1001)_2$

$227 = (11100011)_2 \rightarrow$ یکنواخت : $5 = (0101)_2$

$253 = (11111101)_2 \rightarrow$ یکنواخت : $7 = (0111)_2$

برای تصویر اصلی:

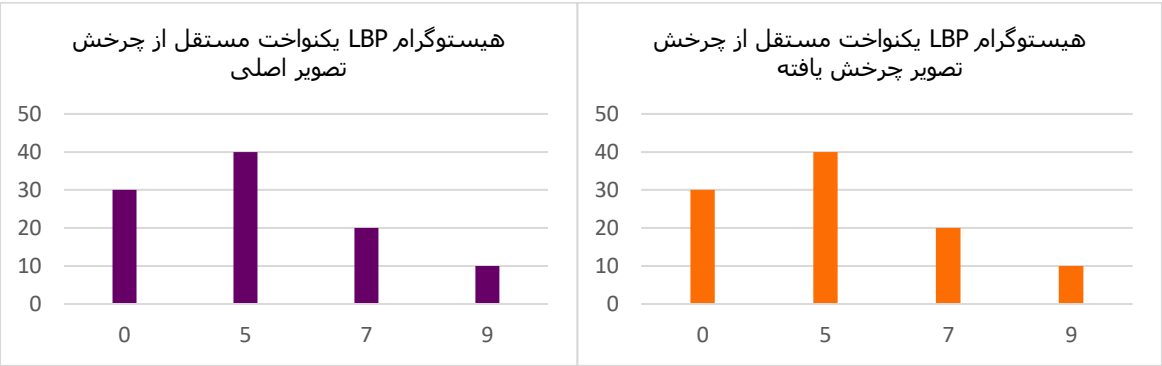
$0 = (00000000)_2 \rightarrow$ یکنواخت : $0 = (0000)_2$

$34 = (00100010)_2 \rightarrow$ غیریکنواخت : $9 = (1001)_2$

$143 = (10001111)_2 \rightarrow$ یکنواخت : $5 = (0101)_2$

$247 = (11110111)_2 \rightarrow$ یکنواخت : $7 = (0111)_2$

همانطور که می بینیم با چرخش تصویر، کد LBP یکنواخت مستقل از چرخش پیکسل ها تغییر نمی کند، چون تعداد 1 ها تغییر نمی‌کند، و در نتیجه هیستوگرام آن ها نیز مشابه است.



منابع: اسلاید

سوال دوم:

1- سه تابع مربوط به محاسبه compactness، eccentricity و solidity را پیاده سازی می کنیم.
ابتدا برای رفع نویز و هموارتر شدن، روی تصویر یک فیلتر ضعیف گاوسی می‌زنیم، سپس تصویر را grayscale کرده، و تصویر را باینری می کنیم. من از روش های thresholding مختلفی استفاده کردم، برای این تسک و روی این دیتاست، روش Otsu بهتر از همه جواب داد.
سپس برای اینکه findContours به شکل بهتری کشتی و هواپیما را از تصاویر باینری تشخیص دهد، از opening و closing با عنصر ساختاری های مختلف استفاده کردم، که تا حد امکان اشکال بی‌مربوط از تصاویر حذف شده و شکل های اصلی کشتی و هواپیما با دقت بیشتری از تصاویر استخراج شوند.
سپس روی نتیجه، findContours می زنیم و از بین کانتور ها کانتور با بیشترین مساحت را در نظر میگیریم.
قدم بعدی محاسبه ویژگی مورد نظر برای آبجکت استخراج شده است. این سه پارامتر برای هر آبجکت از فرمول های زیر محاسبه می شوند.

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

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

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

نتیجه validation function برای یک تصویر کشتی و یک تصویر هواپیما:

```
Result for ship image:
compatness is : 0.6817744565013821    eccenticity is : 0.981031097066142    solidity is : 0.9831235697940504
Result for airplane image:
compatness is : 0.1846583078551962    eccenticity is : 0.7420817333011284    solidity is : 0.8883087170918774
```

2- برای محاسبه هیستوگرام LBP تصویر، ابتدا تصویر را grayscale می کنیم. LBP به دو پارامتر نیاز دارد: شعاع الگوی همسایه پیکسل مرکزی (radius)، و تعداد نقاط در امتداد شعاع بیرونی (numPoints) که ورودی های تابع هستند. همچنین Method را uniform قرار می‌دهیم که LBP یکنواخت مستقل از چرخش را داشته باشیم. در این حالت، به تعداد numPoints + 1 کد یکنواخت مستقل از چرخش در LBP با numPoints همسایه خواهیم داشت، و یک کد نیز برای همه الگوهای غیریکنواخت در نظر گرفته می‌شود. پس در مجموع numPoints + 2 کد داریم.
برای ساختن بردار ویژگی، باید از np.histogram استفاده کنیم که تعداد دفعاتی که هر یک از نمونه های LBP ظاهر می شوند را می شمارد. هیستوگرام برگشتی، numPoints + 2 بعدی است.

سپس این هیستوگرام را را normal می کنیم تا مجموع آن برابر با 1 شود.

3- قدم بعدی آماد کردن دیتاست است. دیتاست را دانلود کرده، آن را unzip می کنیم و در پوشه dataset ذخیره می کنیم. بردار Label ها را نیز از روی نام تصاویر میسازیم. سپس تصاویر به دو بخش train و test تقسیم میشوند.

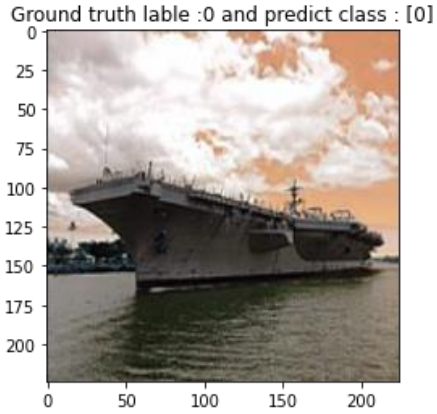
4- تابع get_feature_matrix را پیاده سازی می کنیم. به گونه ای که یک ماتریس دو بعدی با سه ستون compactness، eccentricity و solidity و 10 ستون مربوط به خروجی هیستوگرام LBP_8^1 (شعاع همسایگی 1 و تعداد نقاط 8) میسازد. یعنی در مجموع 13 ستون دارد. سطر های آن نیز مربوط به تصاویر مختلف dataset می باشد.

از LinearSVC به عنوان مدل خود استفاده کرده و feature_matrix_train و label ها را به آن می‌دهیم تا دسته بند آموزش ببیند.

5- در قدم بعدی، عملکرد دسته‌بند آموزش‌دیده را بر روی تصاویر تست می‌سنجیم. ابتدا feature_matrix را برای x_test میسازیم. سپس دسته بند را روی دیتای تست، predict می کنیم. و با استفاده از accuracy_score دقت مدل را روی داده تست بدست می آوریم که به صورت زیر است:

```
accuracy: 87.5 %
```

6- عملکرد دسته بند را روی یکی از تصاویر تست مشاهده می‌کنیم که به درستی predict کرده است:



نتیجه گیری: در این سوال، به صورت دستی ویژگی استخراج کرده، و سپس آن ها را به دسته بند دادیم تا آموزش ببیند. این روش چالش های بسیاری دارد. مخصوصا هنگام استخراج شکل از تصویر، باید تصویر به خوبی threshold گذاری شود و پیش بردازش هایی روی تصاویر انجام دهیم که findContours بتواند به خوبی شکل مورد نظر را از آن استخراج کند. خودم هم هنگام پیدا کردن کانتور مورد نظر از تصویر، به این چالش برخوردم و با تنظیم threshold و برخی عملگر های باز و بسته، آن را رفع کردم. در کل با توجه به چالش های این روش می توان نتیجه گرفت که استفاده از شبکه های عمیق خیلی می تواند مؤثر باشد.

منابع: لینک و لینک و لینک و لینک