به نام خدا

پردازش تصویر

تمرین شماره‌ی ۶

بخش‌بندی و انطباق تصاویر

تاریخ تحویل: ۱۴۰۰/۳/۲۹

ارشین سلطان بایزیدی

۹۷۳۳۰۳۷

استاد درس: دکتر حامد آذرنوش

نیمسال بهار ۹۹-۰۰

سؤال ۱

سؤال (۱) :

$T_0 = 4 \qquad T_1 = \frac{1}{2}(m_{01} + m_{02})$

شدت‌های کمتر از 4

(تعداد)

$$m_{01} = \frac{4(3) + 6(2) + 2(1) + 1(0)}{13} = 2$$

$$m_{02} = \frac{2(5) + 1(6) + 2(8) + 3(9)}{8} = 7.35$$ شدت‌های بیشتر از 4

$$\Rightarrow T_1 = \frac{1}{2}(2 + 7.35) = 4.675$$

شدت‌های کمتر از 4.6

$$m_{11} = \frac{3(4) + 4(3) + 6(2) + 2(1) + 1(0)}{13} = 2.375$$

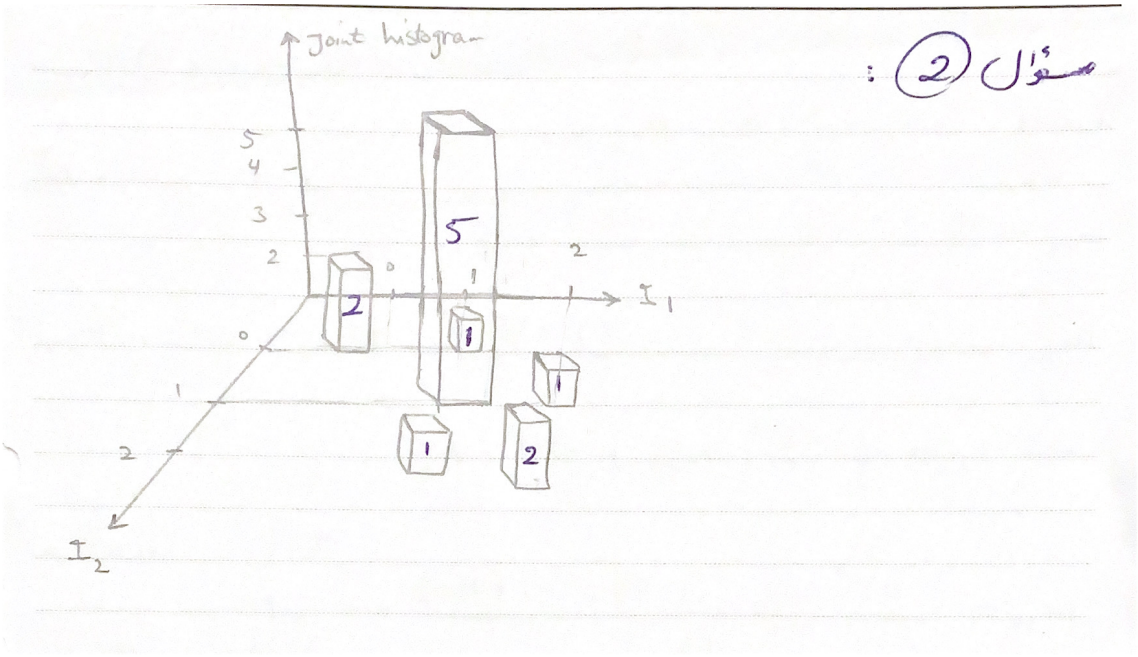
شدت‌های بیشتر از 4.6

$$m_{12} = \frac{2(5) + 1(6) + 2(8) + 3(9)}{8} = 7.35$$

$$T_2 = \frac{1}{2}(2.375 + 7.35) = 4.86$$

سپس دوباره مانند آستانه‌ی قبلی تا مساوی شود و دیگر تغییری نکند.

سؤال ۲

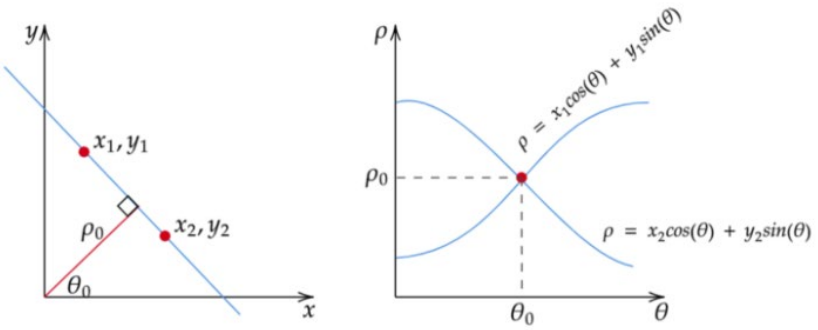


سؤال ۳

الف) Hough Transform روشی برای یافتن خطوط در تصویر است. هنگام شناسایی‌کردن لبه‌ها در تصویر، خطوط معمولاً به‌طور پیوسته نشان داده نمی‌شوند و نقص‌هایی در یافتن خطوط صاف و پیوسته وجود دارد. ایده‌ی اولیه‌ی این تبدیل، بردن معادله‌ی خط y = ax + b از صفحه‌ی x-y به صفحه‌ی a-b است (b = -ax + y). این‌گونه دو نقطه‌ای که در معادله‌ی خط y صدق می‌کنند، در صفحه‌ی a-b دو خط هستند که در یک نقطه اشتراک دارند و تمام نقاطی هم که روی خط y قرار دارند خطوطی‌اند که در صفحه‌ی a-b در آن نقطه برخورد می‌کنند. اما برای خطوطی که عمودی هستند و شیب بی‌نهایت دارند این روش کار نمی‌کند؛ پس به‌جای استفاده از دستگاه مختصات x-y از دستگاه $\theta - \rho$ استفاده می‌شود و معادلات به این شکل نوشته می‌شوند:

$$\rho = xcos\theta + ysin\theta$$

که ρ خط عمود بر خط اصلی در دستگاه مختصات x-y است و θ زاویه‌ی بین خط عمود و محور x است:



در این صورت در دستگاه جدید تعدادی موج کسینوسی خواهیم داشت. وقتی در تصویرمان دو لبه داشته باشیم که نقاطشان روی یک خط باشند، موج‌های کسینوسی هردو در یک نقطه با هم اشتراک خواهند داشت و تبدیل Hough با یافتن چنین نقاطی که از آستانه‌ی مشخصی تعداد اشتراک‌های‌شان بیشتر باشد، خطوط را می‌یابد.

برای یافتن **دایره‌ها** نیز الگوریتم نسبتاً مشابهی داریم، با این تفاوت که به‌جای معادله‌ی خط از معادله‌ی دایره استفاده می‌کنیم:

$$R^2 = (x - a)^2 + (y - b)^2$$

دایره را می‌توانیم این‌گونه نشان دهیم ((a, b) مختصات مرکز دایره‌ی موردنظر است):

$$x = a + R\,cos(\theta)$$
$$y = b + R\,sin(\theta)$$

و θ از صفر تا ۳۶۰ درجه تغییر می‌کند تا دایره به شعاع R تشکیل شود. اگر R دایره ثابت باشد و به‌دنبال دایره‌هایی با آن شعاع بگردیم، می‌توانیم معادلات را این‌گونه بنویسیم:

$$a = x - Rcos\theta$$
$$b = y - Rsin\theta$$

پس پارامترهای تبدیل Hough، a, b, R هستند.هر نقطه در صفحه‌ی x-y برابر با یک دایره در صفحه‌ی a-b است.

ب) برای اینکه تبدیل Hough به‌درستی کار کند، یافتن خطوط و لبه‌های اجسام در تصویر لازم است. زیرا این تبدیل با یافتن اختلاف شدت زیاد (یعنی تشخیص شدت‌های صفر و یک از یکدیگر) بین هر پیکسل و پیکسل‌های همسایه‌اش قادر به شناسایی خطوط/دایره‌ها است. تبدیل Hough با انتقال مختصات کل پیکسل‌های سفید به مختصات قطبی و باقی‌گذاشتن پیکسل‌های سیاه، خطوط را جدا می‌کند و نقاطی را که در یک خط (در اینجا دایره) مشترک هستند به‌هم وصل کرده و خطوط پیوسته/دایره‌ها را تشکیل می‌دهد. برای یافتن لبه‌ها چند روش وجود دارد که می‌توان گفت Canny رایج‌ترین و بهترین روش برای تبدیل Hough است. همچنین طبق درسنامه، روش HOUGH_GRADIENT در تابع آماده‌ی این تبدیل، برای تشخیص لبه از Canny استفاده می‌کند.

روش Canny از چند مرحله تشکیل شده است که علاوه‌بر تشخیص لبه‌ها، کنتراست تصویر را افزایش می‌دهد و نویز را از بین می‌برد. پس برای ازبین‌بردن نویزهایی هم که ممکن است در تشخیص دایره‌ها اشتباهی دایره شناسایی شوند، مناسب است.

فرایند Canny از چند مرحله تشکیل شده است:

۱- تبدیل عکس به خاکستری

۲- استفاده از فیلتر گوسی (که به ازبین‌بردن نویز کمک می‌کند)

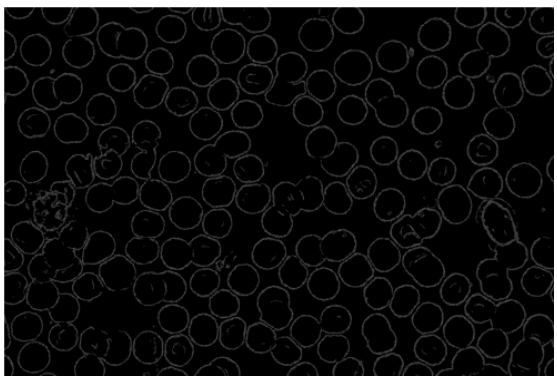
۳- محاسبه‌ی گرادیان (برای تشخیص شدت و جهت لبه‌ها)

۴- باریک‌کردن لبه‌ها (Non-maximum suppression)

۵- ترشهولد دوتایی (با تعیین دو نوع آستانه‌ی زیاد و کم برای تشخیص اینکه چه شدت‌های به‌طور قطعی لبه خواهند بود)

۶- تشخیص لبه با پسماند عملیات قبلی (Hysteresis) (انتقال پیکسل‌های با شدت ضعیف به قوی تنها درصورتی‌که یکی از پیکسل‌هایی که در اطراف پیکسل موردنظر است، شدت قوی داشته باشد)

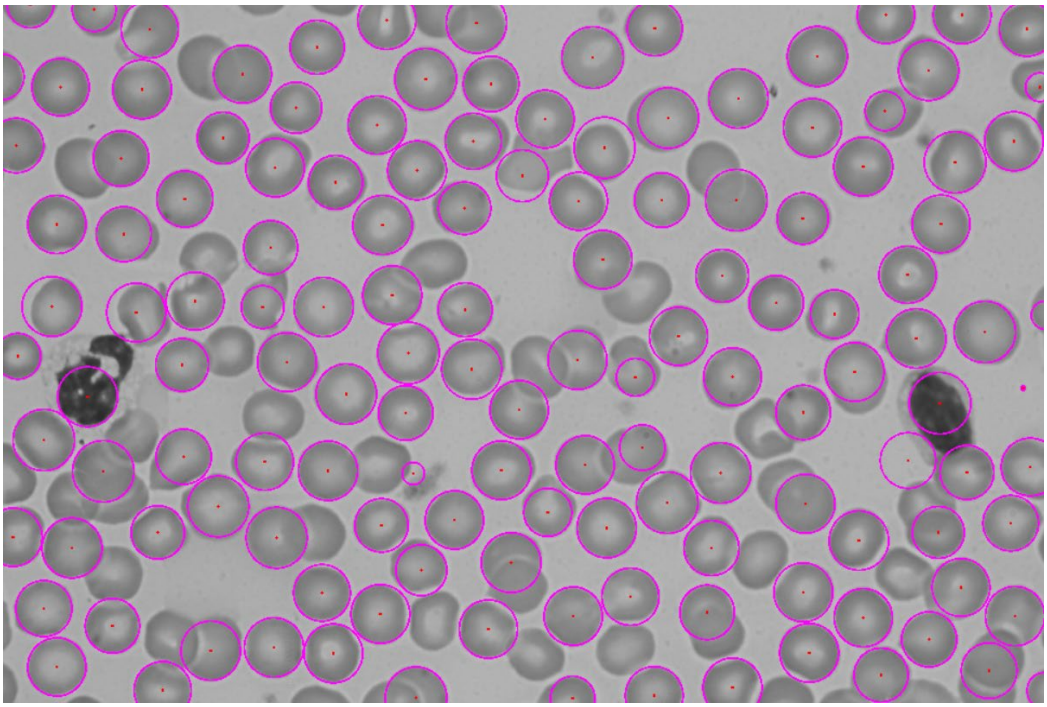
ج) برای اینکه دقت تابع HoughCircles را تا حد امکان بیشتر کنیم، روی تصویر خاکستری‌شده‌ی CBC ابتدا فیلتر Median با شعاع کرنل ۵ اعمال می‌کنیم تا نویزهای تصویر را از بین ببریم. تابع آماده‌ی تبدیل Hough در OpenCV خود برای تشخیص لبه یک فیلتر نرم‌کردن اعمال می‌کند؛ اما اعمال یک فیلتر مناسب قبل از انجام فرایند، دقت نتیجه را بیشتر می‌کند. می‌توانستیم از فیلتر گوسی یا blur عادی نیز استفاده کنیم، ولی طبق درسنامه‌ی اشاره‌شده در سؤال و همچنین با آزمون‌وخطا، فیلتر Median مناسب‌تر است. حال برای تعیین paramهای تابع HoughCircles از Canny استفاده می‌کنیم. با بالاوپایین کردن اعداد آستانه‌ی شدت، از نظر من اعداد ۱۵ و ۳۵ به‌خوبی لبه‌های دایره‌ها را نشان می‌دادند و لکه‌های ریز درون دایره‌ها تا حد زیادی از بین رفته بود:



د) تابع HoughCircles را می‌نویسیم. مقدار minDist که کمترین فاصله‌ی بین مراکز دایره‌های شناسایی‌شده است، به‌طور میانگین یک‌دهم عرض تصویر است و با آزمون‌وخطا دریافتم که عدد ۲۲۰ مناسب است؛ زیرا مقدار کمتر از آن باعث ایجاد دایره‌های بیش‌ازحد و بیشتر از آن باعث جاماندن برخی از دایره‌ها می‌شود. minRadius را صفر می‌گیریم و maxRadius را نیز با حدس و آزمون‌وخطا عددی حدود ۱۱۵ تعیین کردم که دایره‌های شناسایی‌شده تقریباً هم‌اندازه‌ی دایره‌های اصلی تصویر باشند.

اکنون طبق درسنامه، ابتدا برای خواندن و انجام عملیات روی پارامترهای تابع HoughCircles (a, b, r) آن‌ها را به uint16 تبدیل کرده و گرد می‌کنیم. حلقه‌ای می‌نویسیم که با خواندن هر دایره به‌مرکز (a, b) و شعاع r دایره‌ای رسم کرده و همچنین دایره‌ای کوچک‌تر به‌شعاع ۵ نیز برای نشان‌دادن مرکز آن‌ها ایجاد کند. از آنجا که تصویر بزرگ است، برای نمایش آن از تابع namedWindow استفاده می‌کنیم تا اندازه‌ی آن هنگام نمایش قابل‌تنظیم باشد.

در خروجی می‌بینیم که بیشتر دایره‌ها به‌درستی شناسایی و رسم شده‌اند؛ اما در برخی جاها دایره‌هایی که به دایره‌های دیگر برخورد کرده‌اند شناسایی نشده‌اند یا یک جای خالی اشتباهی دایره درنظر گرفته شده است.



سؤال ٤

تبدیل Affine مجموعه‌ای از تغییرات روی تصویر است که هم‌راستایی و نسبت فاصله‌ی بین نقاط مختلف در تصویر حفظ شود؛ مانند چرخش، translation، scaling و غیره. ماتریس‌های تبدیل Affine به این شکل هستند:

$$A = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix}_{2\times2} \quad B = \begin{bmatrix} b_{00} \\ b_{10} \end{bmatrix}_{2\times1}$$

$$M = [\,A \quad B\,] = \begin{bmatrix} a_{00} & a_{01} & b_{00} \\ a_{10} & a_{11} & b_{10} \end{bmatrix}_{2\times3}$$

که M ماتریس نهایی تبدیل Affine است و معمولاً به‌فرم ماتریس ۲در۳ نیز نمایش داده می‌شود.

در این سؤال برای اینکه تبدیل Affine را روی تصویر MRI انجام دهیم، باید سه نقطه از هر دو تصویر بگیریم تا ماتریس را تشکیل دهیم.

ابتدا دو لیست خالی ایجاد می‌کنیم تا مختصاتی را که کاربر وارد می‌کند در آن‌ها ذخیره کنیم. با استفاده از دستورها و توابع آماده، کد مناسب را برای گرفتن کلیک از کاربر و ثبت مختصات می‌نویسیم. سپس با دستور putText مقداری را که کاربر کلید کرده روی تصویر و در خروجی می‌نویسیم و در یک حلقه‌ی while تعیین می‌کنیم که با فشردن کلید Enter تصویر بعدی نمایش داده شود. در آخر با استفاده از دو دستور getAffineTransform و warpAffine طبق مختصاتی که از کاربر گرفته‌ایم روی تصاویر تبدیل را انجام می‌دهیم و نمایش می‌دهیم.

برای مثال اگر مختصات زیر را کلیک کنیم (۳ مختصات اول برای تصویر MRI2 وکه درواقع مانند ۳ رأس مثلث روی MRI سر هستند، خروجی تقریباً شبیه تصویر اصلی می‌شود.

```
Click on 3 coordinates and press Enter
37   96
214   99
119   247
Click on 3 coordinates and press Enter
171   43
207   128
131   122
```