

Assignment NO.5 Solutions

Digital Image Processing | Fall 1400 | Dr.Mohammadi

Teacher Assistant : Ramin Kamali

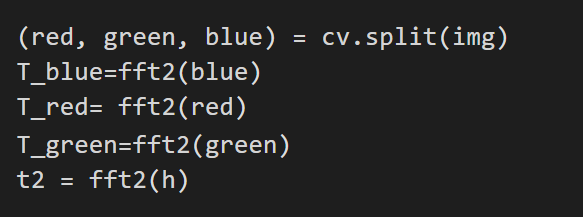
Student name : **Amin Fathi**

Student id : **400722102**

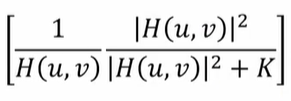
**Problem1**

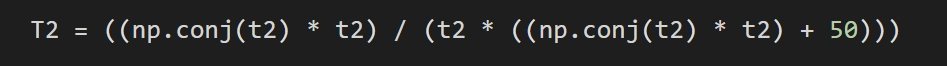
تصویر image1.jpgرا بصورت سطح خاکستری بخوانید. تصویر بدلیل حرکت دوربین در راستای عمودی تخریب شده است و مدل تخریب در فضای مکان به شکل h.bmpاست. با پیاده سازی فیلتر وینر تاری ناشی از حرکت را اصلاح کنید. در فایل HW5.ipynbتابع image\_restoration را بگونه ای تکمیل کنید که تصویر ورودی و مدل تخریب را گرفته و تصویر اصلاح شده را برگرداند (استفاده از کتابخانهای که فیلتر وینر را بصورت مستقیم اعمال کند مجاز نیست.)) 20امتیاز(

ابتدا در تابع image\_restoration تصویر ورودی را به صورت سه کانال red , blue , green جداسازی می کنیم و سپس از هر یک از کانال های به دست آمده تبدیل فوریه میگیریم. همچنین از مدل تخریب h هم تبدیل فوریه میگیریم و در متغییر t2 ذخیره می کنیم .



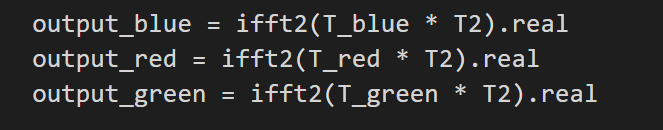
سپس تابع تبدیل وینر را که به صورت زیر است پیاده سازی میکنیم



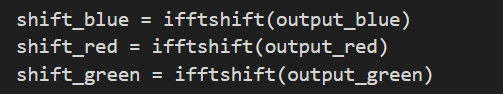


لازم به ذکر است مقدار K 50 انتخاب شده است

سپس مقدار تبدیل فوریه معکوس را برای حاصل ضرب تبدیل وینر در تبدیل فوریه کانال ها را برای هر کدام از کانال ها RGB به دست می آوریم



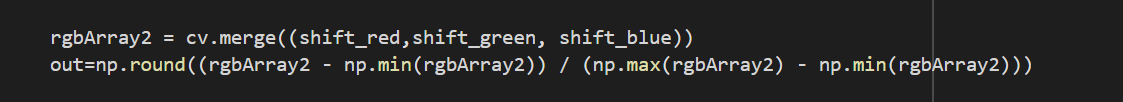
از آنجا که تابع تخریب ما تبعی است در مرکز تصویر h بنابراین برای انکه تصویر نهایی درست به دست بیایید . تبدیل فوریه معکوس را شیفت میدهیم .



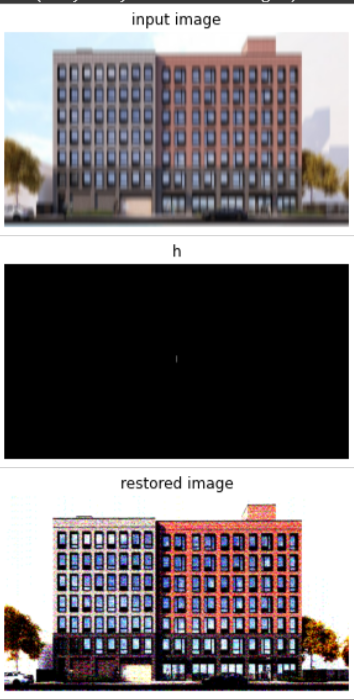
لازم به ذکر است چنانچه این کار را انجام ندهیم ، نتیجه نهایی به شکل زیر خواهد بود



حال تصویر شیفت یافته را با هم ترکیب کرده و پس از نرمال سازی به عنوان خروجی تحویل میدهیم .



نتیجه نهایی :



**Problem2.a**

تمام مراحل لبه یاب cannyرا نام برده و دلیل استفاده از هر مرحله را توضیح دهید.

- **هموار کردن تصویر با استفاده از فیلتر گاوسی** : به مانند سوبل برای حذف نویز با فیلتر گوسی برای کم کردن جزئیات و نویز که تصویر را اماده می کند برای مشتق گیری

- **محاسبه گرادیان** : با استفاده از سوبل مشتق تصویر را حساب میکنیم

- **حذف مقادیر غیر بیشیینه**: برای یکی از دو مشکل سوبل ( وجود لبه های ضخیم) برای حذف لبه های ضخیم از بین چند پیکسل که دارای مقدار گرادیان هستند و در راستای عمودی بر سطح مشتق هستند ، پیکسل های اضافی را حذف می کنیم تا لبه ها متعادل شوند ( این کار با توجه به جهت گرادیان انجام می شود و در جهت عمود بر لبه ، ضخامت رابه یک پیکسل کاهش می دهیم )

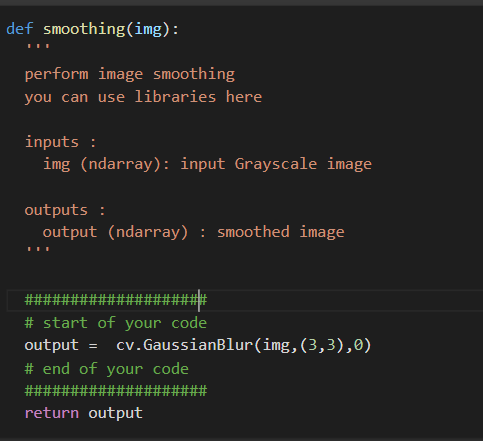
-**آستانه گذاری دو مرحله ای** : برای حل مشکل دوم سوبل ( گسستگی مقدار گرادیان ها ) که در واقع در بعضی نقاط گرادیان ضعیف بوده و در بعضی نقاط قوی ، ابتدا نقاط را به سه قسمت تقسیم میکنیم ( 1 - نقاطی که لبه هستند و مقدار آن ها از ماکسیمم ترشولد بیشتر می باشد 2 - نقاطی که لبه نیستند و مقدار آن ها از مینیمم ترشولد کم تر میباشد 3 – نقاط مشکوک که بزرگی گرادیان آن ها بین مینیمم ترشولد و ماکسیمم ترشولد قرار دارد ) برای تعیین وضعیت گروه سوم ، در صورتی که این نقاط در همسایگی یکی از نقاط گروه اول ( لبه ها ) باشند ، خود نیز لبه به حساب می آیند و این چرخه را برای تمام نقاط تکرار می کنیم . در نتیجه این کار گسستگی بین لبه ها از بین می رود .

منبع : توضیحات استاد در کلاس درس

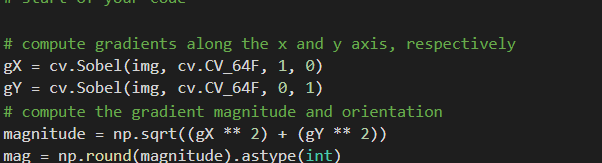
**Problem2.b**

الگوریتم cannyرا مرحله به مرحله پیاده سازی کنید و با استفاده از آن تصویر image2.jpgرا لبه یابی کنید. در HW5.ipynb برای هر مرحله یک تابع نوشته شده است که باید آنها را کامل کنید. در پیاده سازی توابع حذف مقادیر غیر بیشینه و آستانه گذاری دو مرحله ای استفاده از کتابخانه مجاز نیست. سپس تابع opencv\_cannyرا کامل کنید که با استفاده از کتابخانه OpenCVلبه یابی را انجام میدهد. نتایج بدست آمده را از نظر خروجی و زمان اجرا مقایسه کنید

ابتدا با استفاده از کتابخانه GaussianBlur و کرنل 3\*3 تصویر را هموار میکنیم



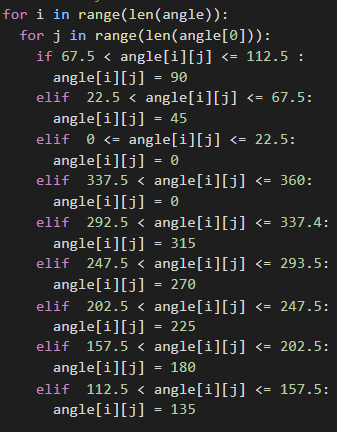
سپس با استفاده از کتابخانه open cv بزرگی مشتق را حساب کرده و مقادیر آن را به مقادیر اعداد طبیعی رند میکنیم .



همچنین برای محاسبه مقدار زاویه های گردایان تصویر از توابع np.gradient و cv.phase استفاده می کنیم تا در نهایت مقدایر زاویه ها در متریک درجه را در خروجی بدهد .

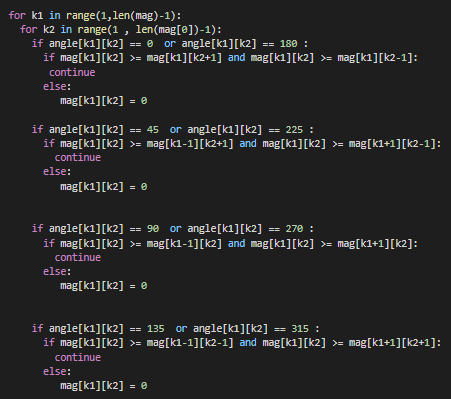


برای تابع NMS که وظیفه آن حذف مقادیر غیر بیشینه است ، ابتدا مقدایر زاویه ها را نرمالایز میکنیم :به این صورت که آن ها را به نزدیک ترین مقدار از مقادیر 0 و 45 و 90 و 135 و 180 و225 و 270 و315 درجه نگاشت میکنیم .



در قسمت بعد ، با توجه به اینکه حساسیتی در نقاط مرزی تصویر وجود نداشت ، به جای پدینگ دادن تصویر ، ادامه عملیات پردازش تصویر را بر روی لایه درونی تصیور که از هر طرف یک پیکسل با تصویر فاصله دارد انجام شده . هر درایه از ماتریس بزرگی گرادیان با توجه به مقدار زاویه گرادیان ای که دارد با درایه های همسایه اش ( در راستای زاویه گرادیانش ) مقایسه شده و چنانچه از آن دو درایه کمتر بوده ، مقدارش به صفر نگاشت می شود . در غیر این صورت مقدار خود را حفظ می کند .

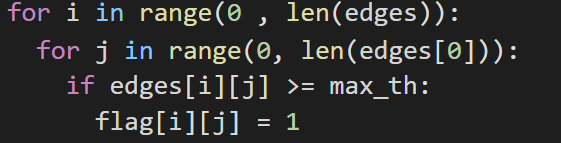
نتیجه نهایی به عنوان خروجی تابع معرفی می شود



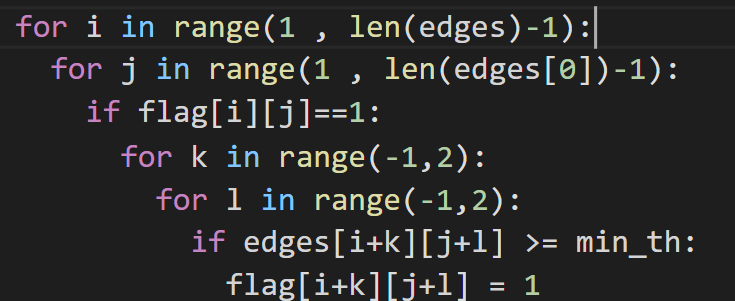
در قسمت بعد و در تابع hysteresis\_threshold که وظیفه آستانه گذاری دو مرحله ای را بر عهده دارد ، ابتدا یک آرایه دو بعدی تعریف میکنیم به نام flag که در نهایت برای درایه هایی از ماتریس بزرگی گردایان که مقدار flag شان برابر با صفر بود ، مقدار بزرگی گرادیانشان را صفر میکنیم ، و در غیر این صورت تغییری نمی دهیم .



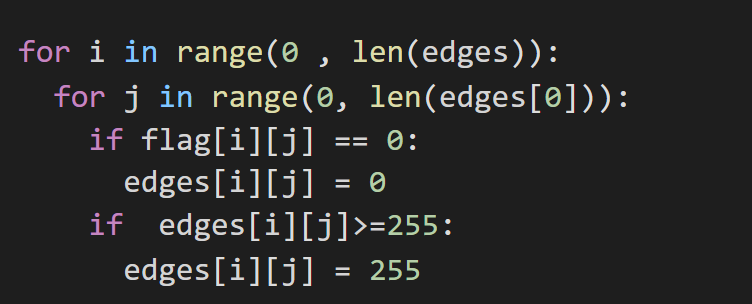
سپس برای همه ی درایه های ماتریس بزرگی گرادیان که مقدارشان از ماکسیمم ترشولد بیشتر بود ، مقدار flag را برابر 1 تنظیم میکنیم



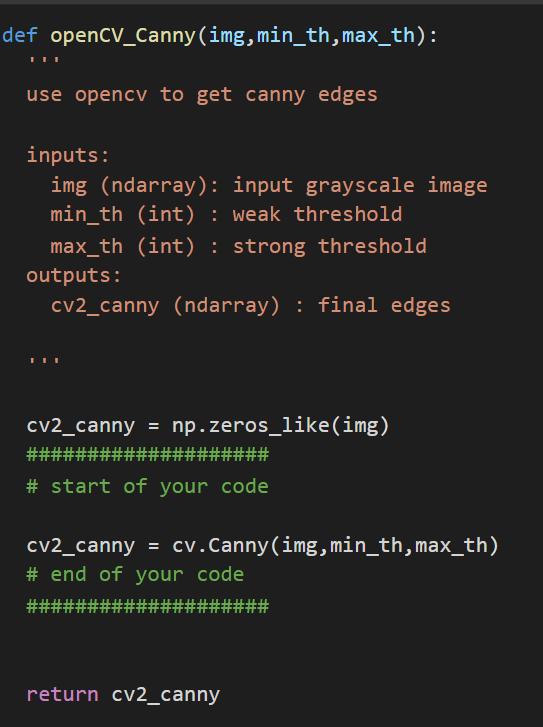
در ادامه تمام همسایگی های درایه هایی که مقدار flag آن ها برابر 1 است را را بررسی کرده و چنان که مقدار بزرگی گرادیان آن ها بیشتر ازmin\_t بوده را نیز بررسی کرده و مقدار flag آن ها را برابر 1 می کنیم .



در نهایت هم مقدار بزرگی گرادیان درایه هایی که flag آن ها برابر صفر می باشد را صفر کرده و همچنین از آنجا که درایه هایی دارای اندازه بیشتر از 255 می باشند و از اسکیل قابل نمایش خارج هستند ، بزرگی گرادیان آن ها را نیز برابر 255 ست میکنیم و نتیجه به در خروجی می دهیم

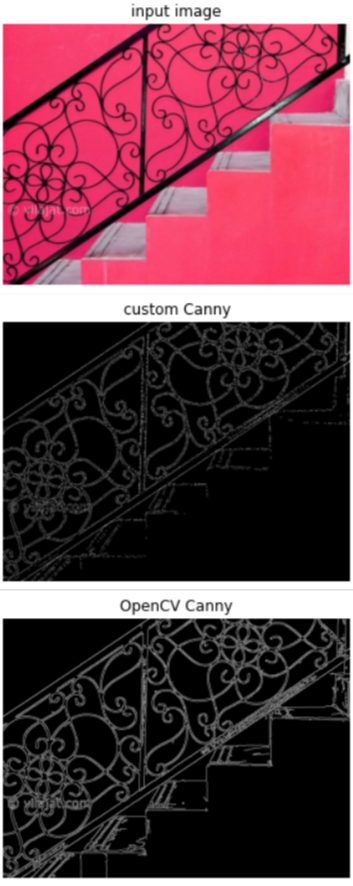
****

در قسمت بعدی تابع opencv\_Canny را پیاده سازی میکنیم



در نهایت نتیجه حاصل را مشاهده میکنید :

تصویر به دست آمده توسط توابعی که ما نوشتیم ، کند تر به دست آمده و همچنین دقت تصویر به دست آمده توسط تابع opencv را ندارد .



**Problem2.c**

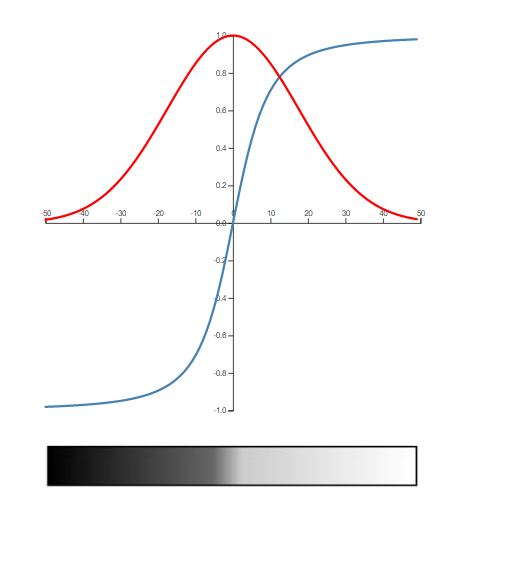
**لبه یاب sobelرا با لاپلاسین مقایسه کنید. آیا لاپلاسین لبه یاب مناسبی است؟**

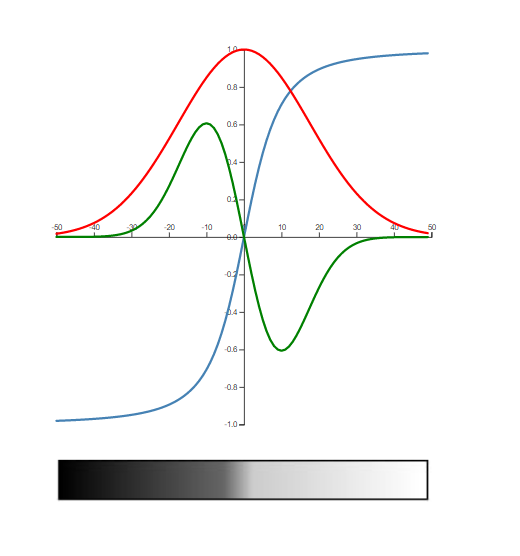
در لبه یاب sobel از دو کرنل برای تشخیص لبه ها به صورت افقی و عمودی استفاده می کنیم در حالی که در لبه یاب لاپلاسین از یک کرنل برای تشخیص گرادیان تصویر استفاده می کنیم .از انجا که لاپلاسین مشتق دوم است با حساسیت بیشتری نسبت به سوبل تغییرات تصویر را نشان می دهد.

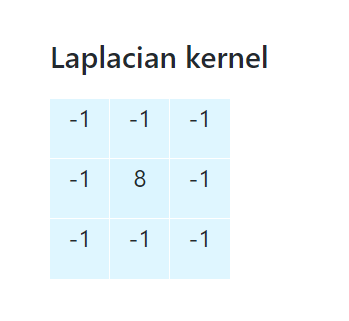
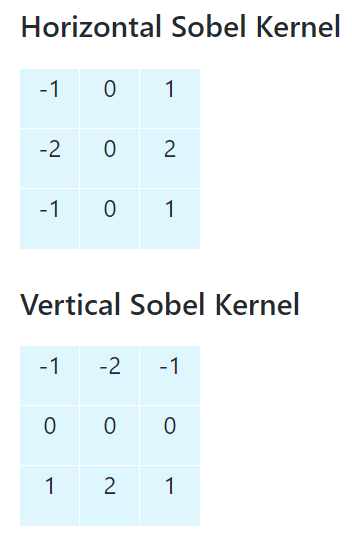
از دیگر تفاوت های این دو روش می توان به شکل موج مشتق و تصویر در این دو روش اشاره کرد.

منبع :

https://cse442-17f.github.io/Sobel-Laplacian-and-Canny-Edge-Detection-Algorithms/

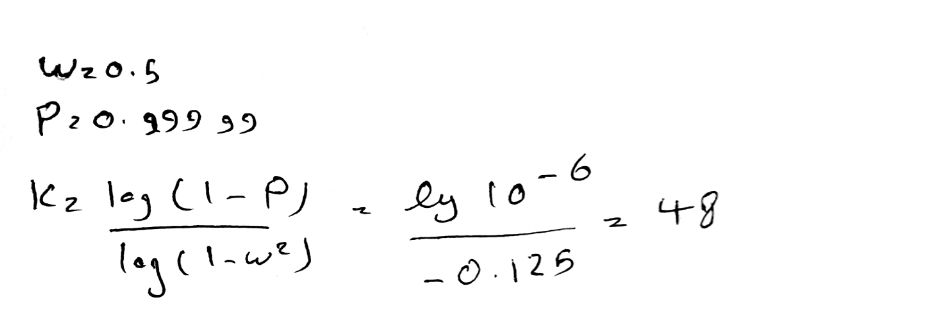


**سوبل لاپلاسین **

****

**Problem3.a**

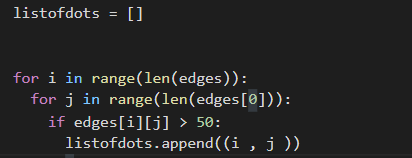
**مي خواهيم از الگوريتم RANSACبراي يافتن پارامترهاي يك دايره استفاده كنيم. در صورتي كه ٥٠ درصد از نقاط موجود مربوط به دايره و بقيه نقاط پرت باشند، براي اينكه دقت انتخاب پارامترها به ٠,٩٩٩٩٩٩ برسد به چند تكرار نياز داريم؟**



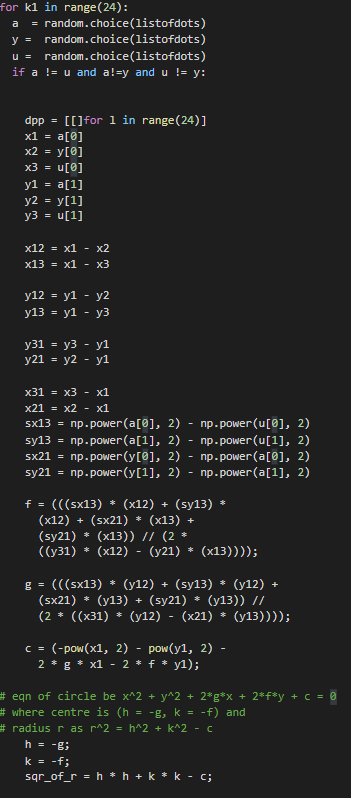
**Problem3.b**

در تصویر image3.jpgتعدادی لبه وجود دارد که 50درصد این لبه ها مربوط به دو دایره و مابقی نقاط پرت هستند. الگورتم RANSACرا بگونه ای پیاده سازی کنید تا پارامترهای این دو دایره (مختصات مرکز و شعاع) را پیدا کند. برای بدست آوردن پارامترهای دایره با استفاده از سه نقطه از این لینک میتوانید استفاده کنید. تابع circle\_RANSACرا کامل کنید. خروجی این تابع سه لیست از طول مرکزها، عرض مرکزها و شعاع های دایره های موجود باشد .

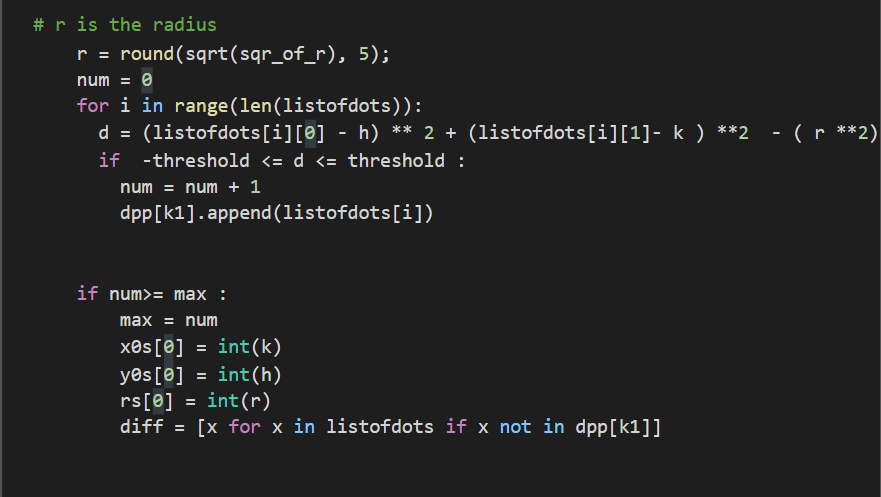
ابتدا با توجه به اینکه برخی نقاط دارای روشنایی های کوچک و در حد 3 و 4 هستند ، برای به دست اوردن نقاط سفید یک ترشولد (50 ) را تعیین میکنیم و مختصات نقاط روشن تر از ان ترشولد را به عنوان نقاط برجستگی ها در لیست listofdots ذخیره می کنیم

****

از انجا که در سوال مقدار p را برابر با 0.999 در نظر گرفته ام و با توجه به فرمول به کار رفته در سوال قبل ، مقدرا k را برابر 24 به دست اورده و به ازای هر بار تکرار 3 نقطه رندم از مجموعه نقاط لبه ها ( listofdots ) به دست آورده و با توجه به لینک روی سوال مختصات مرکز و شعاع دایره قابل ترسیم با این 3 نقطه را به دست می آوریم.

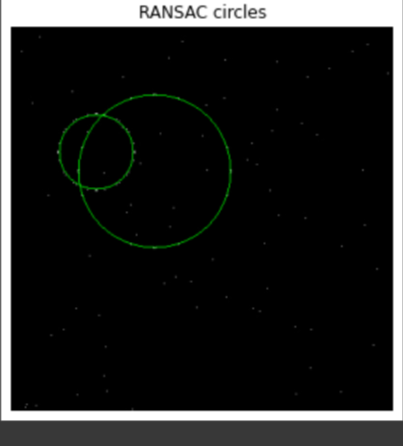


سپس برای دایره به دست آمده معادله دایره را حساب کرده وفاصله نقاط برجستگی را از معادله دایره حساب کرده و در متغیر d قرار می دهیم ، چنانچه مقدار d هر کدام از نقاط لبه از مقدار قدرمطلق ترشولد که در ابتدا برابر با 10 تعیین کرده ایم کمتر باشد ، مقدار رای آن دایره را ( num) یکی اضافه میکنیم ، در نهایت مختصات مرکز و شعاع دایره ای را که بیشترین رای را کسب کرده باشد را به خروجی پاس می دهیم . همچنین نقاطی که برای تشکیل این دایره دارای d کمتر از ترشولد تشخیص داده شده بودند را از لیست نقاط برجستگی حذف میکنیم تا در ادامه برای تشخیص دایره دوم به مشکل نخوریم و نقاط جدید را در لیست diff ذخیره می کنیم .



در ادامه این کار را مجددا برای تشخیص دایره دوم به کار میبریم ، با این تفاوت که لیست نقاط برجستگی که استفاده میکنیم diff می باشد نه listofdots .

نتیجه نهایی :



منابع :

<https://www.youtube.com/watch?v=nG5QC_WFdGU>

<https://github.com/anubhavparas/ransac-implementation>

**Problem3.c**

با مطالعه این منبع، Least Median of squares را توضیح دهید و با RANSAC مقایسه کنید

در این روش ما در واقع به دنبال اپتیمایز کردن تابع فرضی hcur بر اساس پارامتر های موجود است ، مقدار فاصله های نقاط از پیشبینی که مدل ما کرده را به دست اورده سپس میانه ان را حساب کرده و با rcur مقایسه می کنیم و اگر بیشتری نسبت به آن داشت ، آن را جایگزین rcur می کنیم و این کار را تا آنجا ادامه می دهیم که در نهایت بهترین h ممکن حاصل شود .مقدار تکرار در این روش همانند RANSAC تعیین میشود اما تفاوت اصلی این دو روش در این است که RANSAC به دنبال بیشترین چگالی ممکن برای نقاط هست اما Least Median of squares دنبال نازک ترین خطی که 50 درصد نقاط را پوشش دهد .

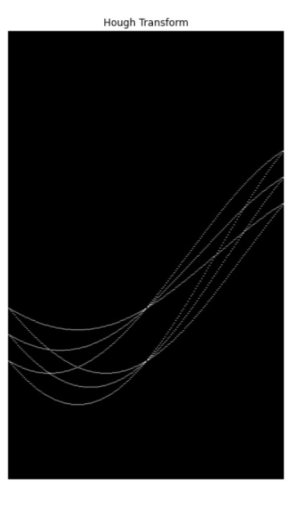
**Problem3.d**

در یک تصویر تعدادی لبه از یک دایره موجود است. در بین لبه ها داده پرت هم وجود دارد. برای اینکه پارامترهای دایره را خیلی دقیق به دست بیاوریم چگونه از RANSAC باید استفاده کنیم؟

برای حل این مشکل علاوه بر اینکه میتوان مقدار تکرار ( k )را افزایش داد با افزایش p ، می توان با کاهش ترشولد وهمچنین رای گیری را بر اساس فاصله نقاط از معادله دایره (x –xc)\*\*2 + (y-yc)\*\*2 – r\*\*2 که در آن r شعاع و (xc , yc) مرکز هستند و توسط انتخاب 3 نقطه تصادفی از نقاط لبه ها به دست می آیند ، برگزار کرد تا دایره دقیق تری به دست بیاید .

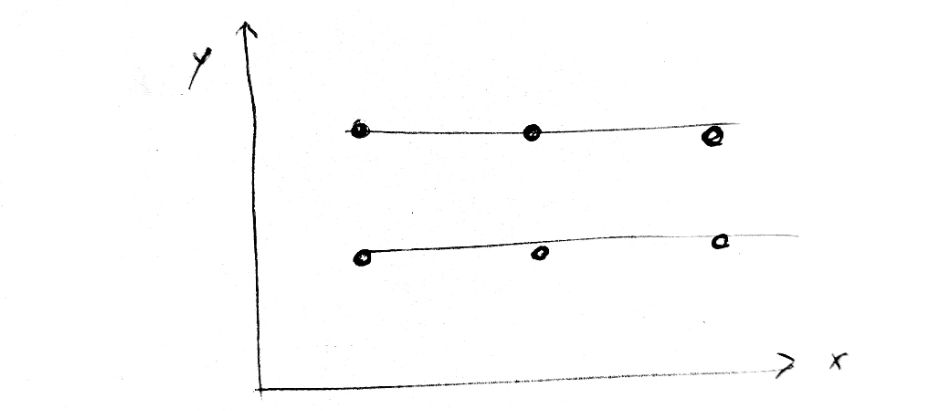
**Problem4.a**

**تبديل hough زیر مربوط به چه تصويري است؟**

****

همانطور که در تصویر می بینیم دو خط با زاویه یکسان ( موازی ) داریم که هر کدام از 3 لبه تشکیل شده اند ، از انجا که این لبه ها در تتایی به حدود 90 درجه ( وسط تصویر) به هم رسیده اند پس طبق فرمول زیر و با توجه به اینکه مقدار ƿ یکسانی دارند ، 3 نقطه تشکیل دهنده هر خط دارای y های یکسان بوده یا به قولی دو خط به صورت افقی با هم موازی هستند . همچنین با توجه به اینکه هر نقطه از خط اول با هر نقطه از خط دوم دارای ƿ یکسان در شروع و پایان ( تتای صفر و 180 ) هستند ، پس نقاط لبه در هر دو خط ، دو به دو با هم دارای x های یکسان هستند.





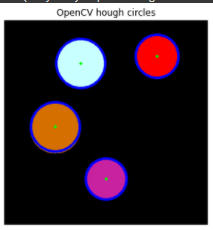
**Problem4.b**

**شبه كدي براي تبديل Houghدايره و با استفاده از جهت گراديان بنويسيد ) راهنمايي: هر نقطه بر روي دايره ميتواند به تعداد زيادي دايره نگاشت شود كه با استفاده از گراديان اين موضوع بايد حل شود(**

* Initialize the accumulator (H[a,b,r]) to all zeros
* Find the edge image using any edge detector
* For r= 0 to diagonal image length
  + For each edge pixel (x,y) in the image
    - For Θ = 0 to 360
      * a = x – r\*cosΘ
      * b = y – r\*sinΘ
      * H[a,b,r] = H[a,b,r] +1
* Find the [a,b,r] value(s), where H[a,b,r] is above a suitable threshold valu

**منبع :** [**https://theailearner.com/tag/hough-circle-transform-algorithm/**](https://theailearner.com/tag/hough-circle-transform-algorithm/)

**Problem4.c**

****توابع را با استفاده از کتابخانه پیاده سازی کرده و نیتجه نهایی به این شکل می باشد :

منبع :

https://www.pyimagesearch.com/2014/07/21/detecting-circles-images-using-opencv-hough-circles/