بسم الله الرحمن الرحیم

دیدکامپیوتری

خانم دکتر هدی محمدزاده

فاز دوم تمرین سری 1

امیرحسین رستمی 96101635

زمستان 98

فایل پایتون و فایل ژوپیتر نوت بوک در فایل آپلودی ضمیمه شده است.ترجیحا ژوپیتر نوت بوک را استفاده کنید.

سوال سوم:

ابتدا طبق خواسته ی سوال ابعاد تصویر سوال را به 300 در 400 پیکسل تبدیل می کنیم که این عمل به کمک دستور cv2.resize انجام می دهیم،توجه کنید که نحوه ی interpolation(درون یابی) خود را با بررسی های مکرر از نوع cubic انتخاب کردیم که با بررسی بین انواع موجود از کیفیت بیشتر برای این عکس برخوردار بود.

خروجی بخش اسکیل یافته:

عکس اصلی اسکیل شده

حال برویم سراغ خواسته ی بعدی سوال که اعمال فیلتر پایین گذر است.ابتدا نحوه ی عملکرد فیلتر میانگین گیر(mean) را توضیح می دهیم و سپس ادعای خود را مبنی بر این که میانگین گیر پایین گذر مناسبی است اثبات می کنیم. ابتدا می دانیم که اثر فیلتر mean به این صورت است که بسته به سایز کرنل موجود همسایگی هایی از پیکسل هر مرحله را انتخاب می کنیم و میانگین آن هارا محاسبه می کنیم و به جای آن پیکسل قرار می دهیم،حال توجه کنید که چون میانگین ذاتا تغییرات شدید را کاهش می دهد پس فرکانس بالا های تصویر با گرفتن میانگین از عکس حذف می شود،اینکه از چه حدود از فرکانس های بالا به بعد حذف می شود بسته به سایز کرنل ما دارد و هرچه سایز کرنل بیشتر باشد طبیعتا به علت زیاد تر شدن پیکسل هایی که در جمع و میانگین گیری شرکت می کنند لذا شدت حذف فرکانس های بالای تصویر بیشتر می شود لذا منطقی است که اعمال فیلتر mean همانند اعمال یک فیلتر low Pass روی تصویر است که فرکانس قطع فیلتر با سایز کرنل رابطه ی عکس دارد و این یعنی هرچه سایز کرنل بزرگتر باشد فرکانس قطع فیلتر low pass ما نیز کم تر می گردد که این مساله نیز در عکس های ضمیمه شده آورده شده است.

توجه کنید با توجه به خصلت mean و نیاز آن به وجود همه ی همسایه ها لذا نیاز که برای پیکسل های سطراول،آخر و ستون اول،آخر نیز چاره بیاندیشیم که این چاره با zero padding انجام می گردد(طبق مباحث آموزشی در درس).ذکر چند نکته حایز اهمیت است.

1 ـ فیلتر را به 2 روش اعمال کردیم.

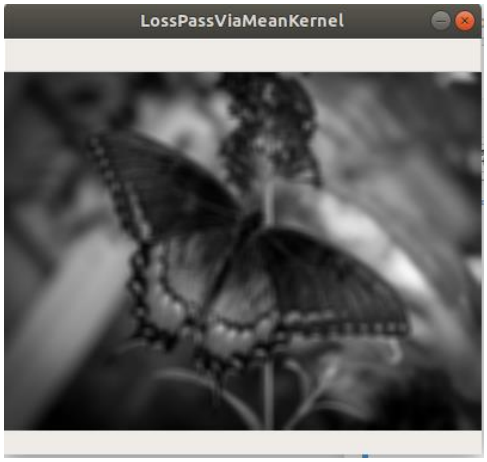
روش اول:با استفاده از فیتلر میانگین موجود در کتاب خانه ی opencv و اعمال آن روی عکس.

روش دوم:اعمال روش mean با محاسبه ی محتوای تک تک پیکسل ها به کمک دو for تو در تو. و سپس همه ی پیکسل هارا در 1/9 ضرب می کنیم که وزن میانگین گیری اعمال شود و در نهایت نیز به علت احتمال تولید پیسکل هایی با محتوای رنگی اعشاری نیاز است که ماتریس دو بعدی نهایی را به unsigned int 8 یا همان uint8،cast کنیم.

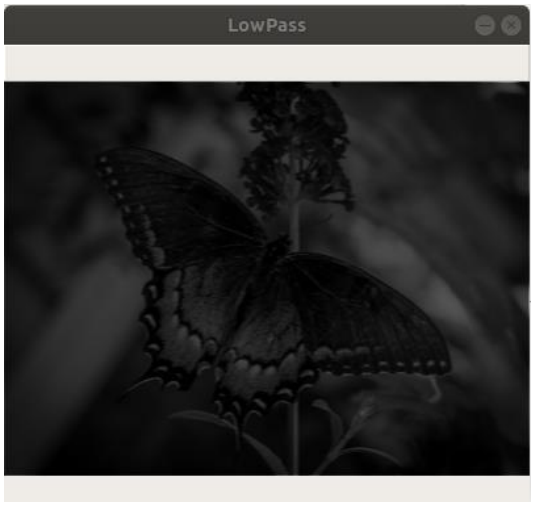
خروجی فیلتر میانگین گیر:

روش اول:اعمال کرنل ها و محاسبه ی فیلترینگ عکس از کرنل میانگین گیر.

با کرنل 3 در 3 و کرنل 7 در 7 به ترتیب:



روش دوم:محاسبه ی دستی به کمک دو تا for تودرتو



همانطور که مشخص است لبه ها کند شده اند و دیگر تغییرات شدید در تصویر دیده نمی شود،و فرکانس بالا های تصویر حذف شده است و عکس به اصطلاح پایین گذر شده است.

حال برویم سراغ Edge:

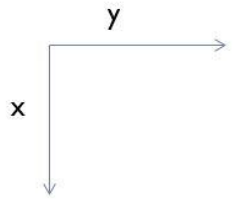
برای پیدا کردن لبه ها در دو جهت از کرنل های ساده ای که در اسلاید های درس اورده شده است استفاده میکنیم. در اسلاید ها کرنل ها به صورت زیر معرفی شده اند

$\frac{\partial f}{\partial x} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x,y)-f(x,y)}{\Delta x}$ $\frac{\partial f}{\partial x} \approx f(x+1,y) - f(x,y)$

and similarly $\frac{\partial f}{\partial y} \approx f(x,y+1) - f(x,y)$

پس کرنل ها را به صورت زیر تعریف می کنیم.

- $\begin{bmatrix}-1\\+1\end{bmatrix}$ for horizontal edges -> pixel[i+1,j] − pixel[i,j]
- $[-1 \quad +1]$ for vertical edges -> pixel[i,j+1] − pixel[i,j]



می توانیم برای بهتر شدن و این کرنل هارا فرد تایی درنظر گرفته و فقط یک سطر اول صفر به آنها اعمال کنیم.توجه کنید که این کرنل انگار به نوعی مشتق میگیرد! و بدیهیست که این کار فرکانس بالا های عکس را استخراج می کند چرا

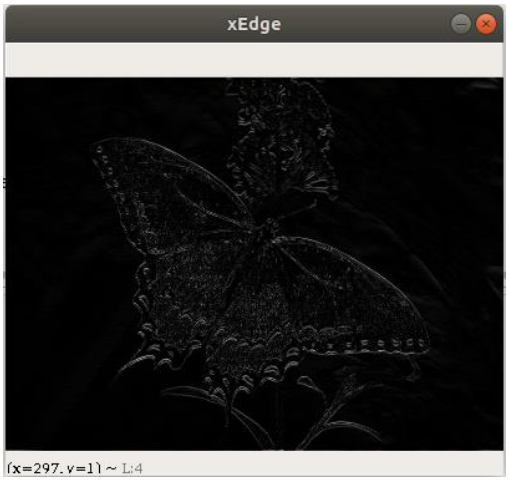
که اگر تغییرات شدید داشته باشیم زنده می ماند و تغییرات نرم و آهسته ی به نوعی حذف(یا بهتره بگیم ضعیف تر) می شود.توجه کنید که همانند فیتلرمیانگین گیر برای درنظر گرفتن رفتار ستون آخر،zero padding را درنظر میگیریم.و بدیهی است که باتوجه به تعریف کرنل،مقدارش برای ستون های آخر چون zeroPadding کرده ایم برابر منفی خودشان است(صفر منهای یک عدد برابر منفی آن عدد)

مقدار تغییرات -> هم ارز است با شدت لبه بودن

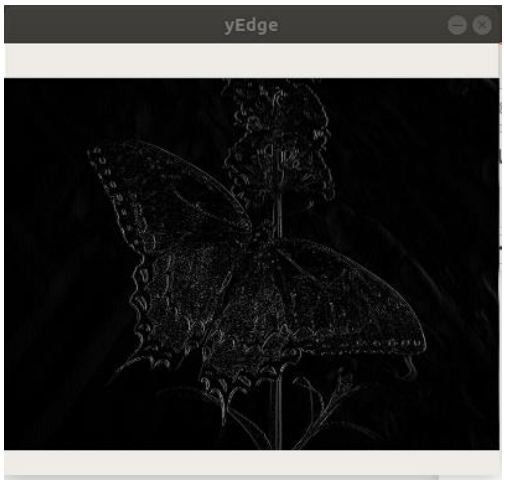
علامت تغییرات -> جهت لبه بودن

از آنجا که در این مرحله صرفا نمایش می خواهیم(و جهت لبه بودن را استفاده نمی کنیم)لذا absolute value(قدر مطلق) گرفته از ماتریس ها و آن هارا رسم می کنیم.

خروجی ها به شرح زیر اند:

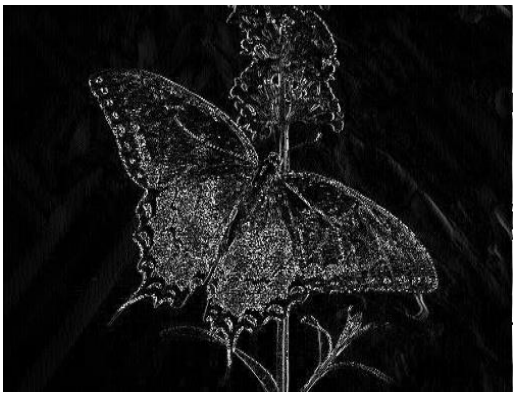


لبه عمودی



لبه های افقی

توجه کنید که الان لبه های پشت را(در پس زمینه پروانه) اینطور نیست که تشخیص داده نشده اند بلکه وزن کمی که دارند که باعث شده است اصلا تاثیر آنها را نبینیم، اگر خروجی اعمال کرنل هارا چند برابر کنیم داریم که این مرز ها لبه ها مشخص تر می شوند.



افزایش شدت رنگی پیکسلی

که لبه هایی از پس زمینه هم نیز وارد شده اند و قبلی هایش هم پر رنگ تر شده اند.

لبه ها، فرکانس های بالای تصویر ما میباشند و به نوعی بالاگذر شده تصویر ما میباشند پس باید از روی لبه های افقی و عمودی ما لبه ها را به صورت کلی پیدا کنیم،تا در نهایت جزییات لبه در شکل کلی به دست بیاید.و برای این کار کافی است که نرم دو آنها را حساب کنیم پس شکل نهایی که لبه یابی به صورت کلی میباشد به صورت شکل زیر می شود:

شکل نهایی لبه

که همانطور که مشاهده می کنید لبه های این حالت با جزییات دقیق تری نسبت به حالت x,y پیدا شده است و این عکس لبه ی کلی تصویر ورودی است.

نکته:در صورت سوال اشاره شده است که ما تصویر بالاگذر شده را به دست بیاوریم،همانطور که می دانید خود edge عه یک تصویر حاوی فرکانس بالا های تصویر است و در اصل همان بخش فرکانس بالای تصویر اولیه است اماطبق صحبتی که با تی ای شد هدف به دست آوردن فرکانس های بالا از روشی دیگر است و بنده روش زیر را درنظر گرفتم

1ـ محاسبه ی low pass تصویر اولیه

2ـ تفریق low pass از تصویر اولیه(همانطور که می دانید low pass فرکانس پایین های تصویر را دارد و تفریق آن از عکس اصلی غالبا شامل فرکانس بالا های تصویر خواهد شد که انگار عکس به دست آمده بالاگذر شده ی عکس اول است).

روش اول:

روش دوم:

سوال چهارم:

توضیح روش Sobel:

روش Sobel نوعی مشتق گیر گسسته است که به طریقی گرادیان تصویر را به صورت تقریبی از طریق کانولوشن با کرنل های مخصوص به دست می آورد،و هم چنین با توجه به توضیحات داکیومنت opencv داریم که این روش،روش Gaussian smoothing (فیلترینگ گوسی) و مشتق گیری(differentiation) را با یکدیگر ترکیب می کند.

توجه کنید که در روش Sobel ما دو مشتق عمودی و افقی از عکس میگیریم و سپس از روی این دو مشتق گرادیان تصویر را محاسبه می کنیم.

اگر چه opencv پیاده سازی داخلی Sobel را در درون خود دارد اما به نظرم خالی از لطف نیست اگر اشاره ای به انواع پیاده سازی های دیگری که می توانیم با سواد فعلی برای این متد بکنیم را بیان نماییم.

پیاده سازی الگوریتم Sobel:

برای پیاده سازی این الگوریتم از روش می توان استفاده کرد.

- روش openCV:استفاده از cv2.Sobel که تابع آماده پایتون است.
- روش اول: تولید ماتریس های Gx, Gy از طریق گرفتن کانولوشن دو بعدی آنها با کرنل ها.
- روش دوم:استفاده کردن از تبدیل فوریه ی دو بعدی و بردن ماتریس های Gx, Gy در فضای فوریه و هم چنین بردن ماتریس تصویر به فضای دو بعدی(تبدیل فوریه گرفتن از ماتریس تصویر) و در نهایت ضرب کردن دو تبدیل حاصل در کرنل ها جهت یافتن تبدیل فوریه ی پاسخ و در نهایت تبدیل فوریه ی وارون گرفتن از ماتریس نهایی و محاسبه ی G به کمک یکی از دو فرمول ذکر شده در پایین.
- روش سوم: استفاده از تابع 2Dfilter جهت فیلترینگ به کمک کرنل های ذکر شده.

هر سه روش مشابه هم بوده صرفا از ابزار های مختلف استفاده شده است.

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A$$

نکاتی چند:

*نماد کانولوشن است و داریم که Gx , Gx مشتق گرفته شده های ماتریس A در راستا های x و y است.

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & +2 & +1 \end{bmatrix} * A$$

جهت محاسبه ی ماتریس نهایی لبه به طور دقیق می توان از فرمول زیر می توان استفاده کرد.اما توجه کنید که گاهی وقتا از فرمول پایینی اش که پیچیدگی کمتری دارد و نتیجه ی به نسبت خوبی هم می دهد می توان استفاده کرد.

$$G = \sqrt{G_x^2 + G_y^2}$$

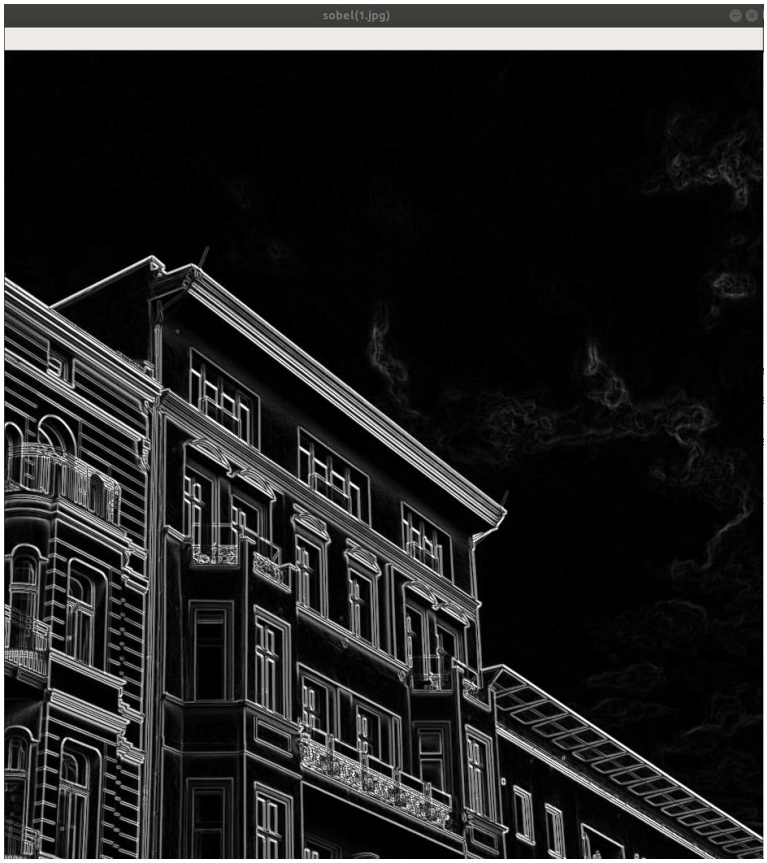
$$G = |G_x| + |G_y|$$

هشدار: از آنجا که Sobel تنها تقریبی از مشتق است، گاهی وقت ها رخ می دهد که کیفیت آن در کرنل سه تایی کاهش پیدا می کند و مقداری نارسایی در خود دارا می باشد در این مواقع از کرنل های scharr استفاده می کنیم که به شرح زیر اند و از دقت بالایی در محاسبه ی مشتق گیری برخوردار است.

$$G_x = \begin{bmatrix} -3 & 0 & +3 \\ -10 & 0 & +10 \\ -3 & 0 & +3 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ +3 & +10 & +3 \end{bmatrix}$$

حال به نمایش خروجی های الگوریتم Sobel می پردازیم:



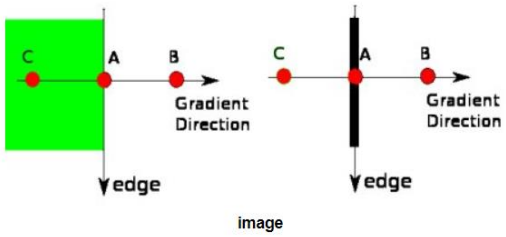
توجه کنید که عکس سمت چپ بخشی از کل عکس است(جهت جا شدن!)

توضیح روش Canny:

در این روش ما ابتدا با فیلتر های مشتق گاوسی اندازه و زاویه گرادیان ها راپیدا میکنید، کاری که انجام میدهیم بعد آن این است که هرپیکسل را باتوجه به جهت گرادیانش با همسایه های خود مقایسه میکنیم چون لبه در جهت گرادیان تنها یک نقطه باید بدهد و برای ما در لبه یابی ضخامت خود لبه خیلی مطرح نمیباشد که این ضخامت در همان جهت گرادیان میپاشد پس و وقتی با همسایه های در جهت گرادیان مقایسه کردیم از بین آنها ماکسیمم های محلی را انتخاب میکنیم و به عنوان کاندیداهای لبه بودن آن ها را در نظر میگیریم.

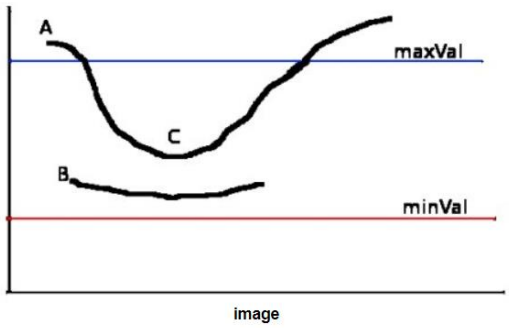
$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$
$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

پیدا کردن کاندید های لبه بودن به شرح تصویر:



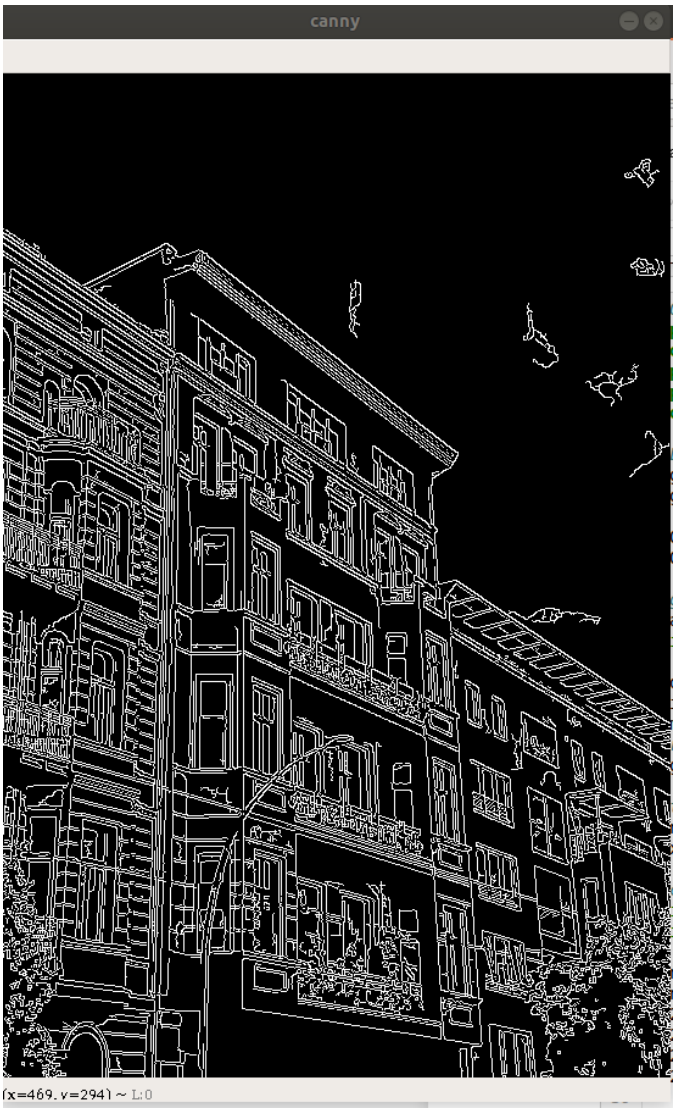
شکل 16 : حرکت برروی گرادیان

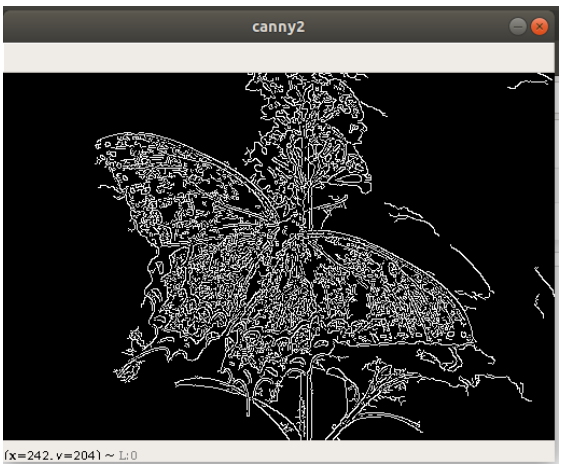
حال در داکیومنت canny به شرح روشی به نام double thresholding می پردازد که به نوعی پیکسل های رنگی را بین دو مجانب افقی بلوکیده می کند.به شکل زیر که در داکیومنت آمده است توجه کنید:



در این عملیات در ابتدا یک ترشلد قوی و یک ترشلد معمولی در نظر میگیریم، از بین ماکسیمم های محلی انتخاب شده گزینه ای را به عنوان لبه قطعی انتخاب میکنیم که بیشتر از ترشلد قوی ما باشد و اگر تنها بیشتر از ترشلد معمولی باشد انرا لبه احتمالی در نظر میگیریم و لبه های احتمالی در صورت اتصال به یک لبه ی قطعی یا ضعیف دیگر پذیرفته می شوند.

خروجی ها:



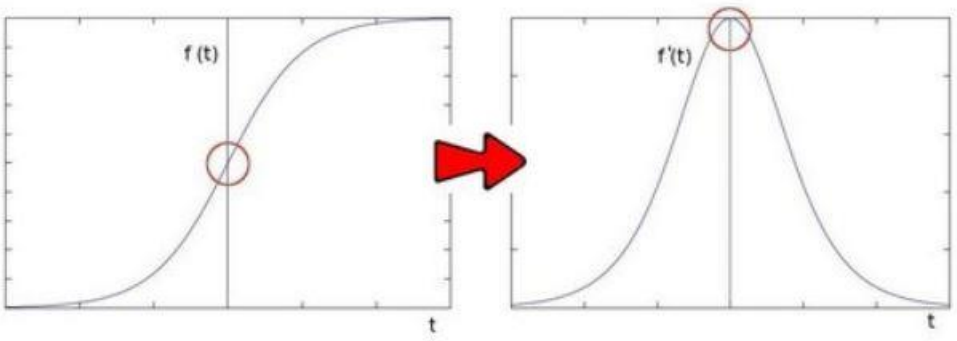


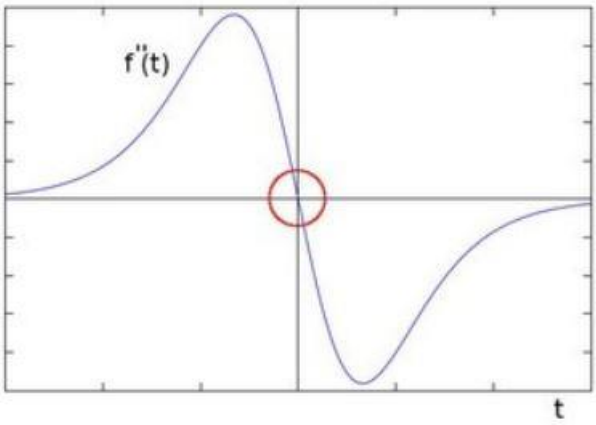
توضیح روش لاپلاسین(loG):

همانطور که در ریاضی مهندسی داشتیم(و هم چنین با توجه به داکیومنت openCV) داریم که رابطه لاپلاس به شرح زیر است:(که در ذات خود به نوعی مشتق گیری مرتبه 2 است)

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

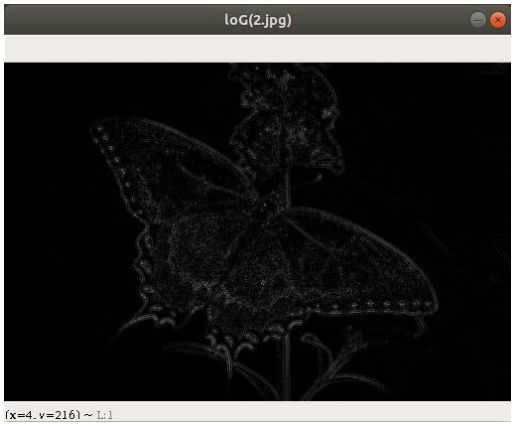
با توجه به گفته های داکیومنت openCV داریم که به کمک تشخیص محل zero crossing داریم که می توان محل لبه را تعیین کرد اما همانطور که در داکیومنت openCV شرح داده شده است این zero crossing در مواقع بی موردی ایجاد شود! که این موارد با الگوریتم laplacian عه openCV به طریقی با اعمال فیلتر هایی هندل می شود.علت اینکه zero crossing به کمک مشتق دوم به دست می آید را به راحتی از روی شکل های زیر می توان تشخیص داد.





تغییرات لبه له صورت شکل اول میباشد اگر مشتق را نگاه کنیم باید ماکزیمم آن را به عنوان معیار در نظر بگیریم باید نقطه ای که در آن به ماکزیمم خود میرسد را پیدا کنیم ولی برای مشتق دوم این نقطه نقطه ای میشود که مقدار آن صفر شده و محل تغییر علامت آن میباشد این را برای تصاویر دو بعدی نیز میتوانیم تعمیم دهیم ولی نکته ای که باید به آن دقت کنیم این میباشد که مشتق دوم به شدت نیز به نویز حساس میباشد پس در مرحله قبل آن یک فیتلر گاوسین برروی

تصویر اعمال می کنیم این امر باعث می شود نویز ها کم شوند و تاثیر آنها تعطیف داده شود بدون اینکه خیلی به لبه ها آسیب برسد حال می آییم فیلتر لبه یاب لاپلاسین را اعمال می کنیم،طبیعتا همانند گفته های قسمت های قبل،شدت هر لبه برابر است با میزان شیب تغییرات در آن نقطه.(مطالب بالا همگی از داکیومنت openCV استخراج شده است).



مقایسه ها!:

مقایسه نتایج:

ابتدا یک مقاله ی عالی معرفی کنم

مقایسه ی قشنگ روش ها

Canny: از آنجا که فیلترینگ Canny محاسبات زیادی دارد و جزییات زیادتری را در نظر میگیرد لذا همانطور که انتظار داشتیم با ظرافت زیادی لبه ها را تشخیص می داد و هم چنین لبه ها دقیق مشخص می شدند با ضخامت کم!(که خوب است)برخلاف روش های دیگر که لبه ضخامت بیشتری داشت.

هم چنین این روش لبه هارا متصل(پیوسته) استخراج می کرد در صورتی که روش های دیگر لبه ها گاها تکه تکه بودند و ضریب پیوستگی(اتصالشان) کم بود و به علت ویژگی Double thresholding آن دستمان بیشتر باز است و لذا می توان بسته به عکس کیفیت لبه یابی را بیشتر هم کرد.

Sobel: از آنجا که Sobel دقت کمتری دارد ولی خب محاسبات کمتری هم دارد! و در حد محاسبات خودش واقعا قابل قبول عمل می کند.اما توجه کنید چون مشتق گیری اش گسسته بوده است و گاهی وقتا دقتش کاهش پیدا می کند که در آن مواقع گفتیم که کرنل ها به کرنل های scharr تبدیل باید بشوند تا دقت حفظ شود.

LoG:(لاپلاسین):از آنجا که این روش از مشتق دوم استفاده می کند دقت بالایی در لبه های ظریف دارد و همانطور که می بینید با ظرافت عالی لبه های قشنگ بال پروانه در آمده است در این روش.

روش هایی جهت حذف نویز و ارتقا لبه یابی:

1ـ فیلترینگ گاوسی:این فیلترینگ انواع نویز هایی از قبیل نویز پوآسون و ... را تا حد خوبی بهتر می کند اما حواسمان باشد که اثر آن لبه هارا مات نکند و تیزی آن ها خیلی به هم نخورد.

2ـ اپراتور های مورفولوجیکال از قبیل Dilation,Erosion,Opening and Closing که مزایای هر کدام در تمرین سری قبل به تفصیل ذکر گردید.مثلا Closing یا Opening با اعمال کرنل های مناسب اختلالات ناشی از نویز و propagation های ارور محاسبات با دقت کم را تا حدی جبران می کند و مثلا بریدگی های اتصالات یک لبه را درست می کند و در کل با آسیب خیلی کمتری به لبه ها اثرات نویز را تعدیل می کند.

سوال ششم:

ابتدا به تعریف blob می پردازیم،تعریف blob عبارت است از مجموعه ای از نقاط متصل که ویژگی مشترکی را شامل می باشند که این ویژگی در شکل سوال عبارت است محتوای gray پیکسل هاست پس blob ها در شکل مورد سوال عبارت است از مجموعه بخش های سیاه(تیره) متصل به هم است و هدف از الگوریتم blob detection عبارت است از یافتن این region ها است.

این تابع مجموعه ای از مشخصات(params) دارد که در زیر به بررسی دقیق هر کدام می پردازیم:

1_Thresholding

2_Grouping

3_Merging

4_Center & Radius CalCulation

حال به توصیف نحوه ی عملکرد و تعیین هر یک می پردازیم.

ترشهولدینگ:

عکس اصلی پاس داده شده به این تابع توسط دو باند بالا و پایین که عبارت اند از(minThreshold & maxThreshold) باینری می گردد.و میزان افزایش گام های هرقدم Thresholding عبارت است از متغیر thresholdStep لذا داریم که ترشهولد ها از minThreshold شروع می شوند و در هر مرحله به اندازه ی thresholdStep افزایش پیدا می کنند تا به ماکزیمم ترشهولد برسند.

اثر تغییر:در اثر نزدیک کردن این دو حد به یکدیگر میزان دقت کاهش و در اثر افزایش (آن تا حدی) میزان دقت آن افزایش می یابد.

فیلترکردن blob ها بر اثر Color,Size,Shape:

Color: توجه کنید که این ویژگی اندکی دچار مشکل داخلی درون خود پیاده سازی شده است و استفاده خیلی نمی شود اما به هر حال ذکر می کنم.

ابتدا باید filterByColor را برابر 1 باید بکنیم و سپس برای تشخیص blob های تیره تر باید blobColor را صفر تنظیم کنیم و به همین ترتیب برای blob های روشن تر این مقدار را افزایش می دهیم تا نهایتا برای blob های خیلی روشن مقدار 255 را برای blobColor در نظر میگیریم.

Size: می توانیم blob هارا نیز به کمک مساحتشان نیز استخراج کنیم که برای این کار باید ابتدا filterByArea را برابر 1 تنظیم کنیم و سپس با تنظیم کردن minArea و maxArea تعیین کنیم که blob های ما بین چه محدوده هایی

از مساحت پیکسلی باشند: مثلا minArea = x & maxArea = y تعیین می کند که blob های بین حداقل مساحت پیکسلی برابر x و حداکثر مساحت پیکسلی y انتخاب شوند.

:Shape

1 ـ تعیین ساختار دایروی

2 ـ تعیین تحدب

3 ـ تعیین Interia Shape

مورد اول:

به کمک این ویژگی می توانیم تعیین کنیم که ساختار blob ما چقدر به ساختار دایروی نزدیک باشد،برای مثال 6ضلعی منتظم نسبت به مربع از ساختار دایروی بیشتری بهره مند است.برای بهره مندی از این ویژگی ابتدا باید filterByCircularity را برابر 1 تنظیم کنیم و سپس به کمک تعیین دو پارامتر minCircularity and maxCircularity تعیین کنیم که حدود دایروی بودن blob مدنظر است و حال باید تعریفی ریاضی از ساختار دایروی بودن بدهیم.فیچر دایروی بودن برابر است با مقدار فیلد زیر:

$$Circularity = \frac{4\pi Area}{perimeter^2}$$

هر چه این مقدار به 1 نزدیک باشد ساختار شکل،دایروی تر است!(بدیهتا برای دایره این مقدار 1 است).

مورد دوم:

از این ویژگی جهت تعیین حدود تحدب blob ها استفاده می کنیم،توجه کنید که برای فعال شدن این ویژگی ابتدا باید filterByConvexity را برابر 1 قرار می دهیم و سپس با تعیین دو پارامتر maxConvexity و minConvexity حدود تحدب شکل را تعیین می کنیم،توجه کنید که این پارامتر ها بین حداقل مقدار 0 و حداکثر مقدار 1 باید باشند.

$$minConvexity \geq 0 \;\&\&\; 1 \geq maxConvexity$$

مورد سوم:

در بیضی نسبت قطر کوچک به قطر بزرگ (و در حالت کلی در اشکال مختلف نسبت کوچکترین قطر شکل به بزرگترین قطر شکل) توصیفی از میزان درازای(elongated) یک شکل در یک راستا و کوچک بودنش در راستای دیگر است و هرچه این مقدار به 1 بیشتر نزدیک باشد یعنی پخشی شکل حول مرکزش یکنواخت تر است(دایره تر است) و هر چه این مقدار به صفر نزدیک تر باشد(بدیهی است که این مقدار بین 0و1 قرار دارد) یعنی این شکل در راستای قطر بزرگ کشیده تر است و معیاری درازی! شکل است.برای فعال کردن این ویژگی ابتدا باید

filterByInertia را برابر 1 تنظیم می کنیم و سپس حدود درازی blob مطلوب را نیز می دهیم که عبارت اند از پارامتر های زیر:

$$minInteriaRatio \ \& \ maxInteriaRatio \rightarrow it\ is\ obvious\ that: \min \geq 0 \ \&\& \ 1 \geq max$$

حال برای تعیین blob های شکل ضمیمه شده در تمرین،آنقدر با این ویژگی ها کار می کنیم تا در نهایت به State زیر که params مناسبی را جهت تشخیص همه ی این blob ها می دهد و همانطور که ملاحظه می کنید این شکل نیز به طرز دقیقی طراحی شده است به طوری که:

سطر اول تصاویر آن،صحت عملکرد AreaFiltering را تعیین می کند.

سطر دوم تصاویر صحت عملکرد Thresholding را تعیین می کند.

سطر سوم تصاویر صحت عملکرد Circularity را تعیین می کند.

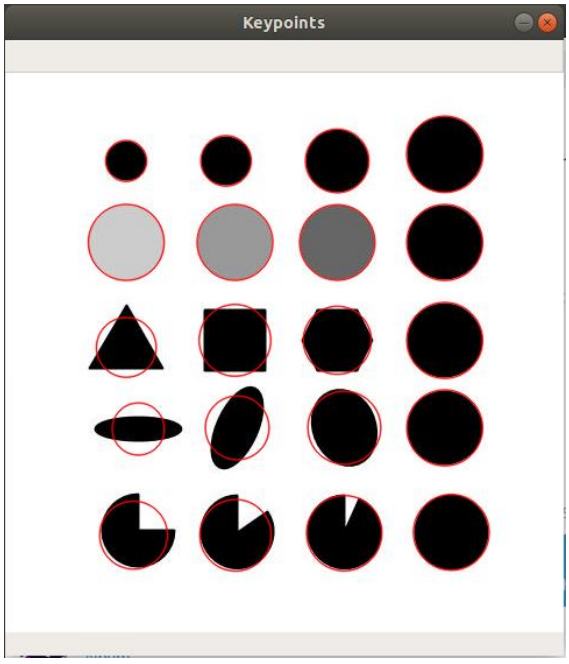
سطر چهارم تصاویر صحت عملکرد Interia را تعیین می کند.

سطر پنجم تصاویر صحت عملکرد Convexity را تعیین می کند.

حال با بررسی و تغییر params با تعیین پارامتر ها به شرح زیر تمامی این blob هارا تعیین می کنیم.

خروجی و تنظیمات:

```
params = cv2.SimpleBlobDetector_Params()

params.minThreshold = 100;
params.maxThreshold = 200;

params.filterByArea = True
params.minArea = 100

params.filterByCircularity = True
params.minCircularity = 0.1

params.filterByConvexity = True
params.minConvexity = 0.9

params.filterByInertia = True
params.minInertiaRatio = 0.01
```



بخش دوم

سوال سوم:

نکته ی مهم:در این بخش بنده ویدیو های هر روش را در فولدر مخصوص داخل فایل پاسخ فاز دو قرار داده ام و از آنجا که آوردن frame های مختلف آن گزارش را واقعا حجیم می کرد،لذا بنده صرفا مغز نتایج را کاملا آورده ام و مصحح بزرگوار (جهت ملاحظه ی frame ها و ویدیو های ذخیره شده و بررسی صحت پیاده سازی ها) به مشاهده ی ویدیو های موجود در لینک مراجعه کنید.

https://drive.google.com/drive/folders/1P5FxwvrV1y5NrvliD_TLNltyKVyeqimk?usp=sharing

ویدیوی اصلی در فایل تمرین وجود دارد که در صورت نیاز آن را می توانید مشاهده کنید و با ران کردن section های مربوطه در کد، خروجی های با فیلترینگ گاوسی اولیه و بدون فیلترینگ گاوسی اولیه را می توانید ملاحظه کنید.

نکته 1:

بنده ویدیویی از خود! ضبط کردم چون صورت سوال خواسته بود ویدیوی از webcam ذخیره شود.

نکته 2:

در فولدری که پاسخ تمرین،ویدیوی من موجود است ولی بخش های نویزی در گوگل درایو اند(چون حجمشان در کنار ویدیوی اصلی از حد مجاز آپلود سی دبلیو بیشتر می گردید)

روش های Sobel , Canny در سوالات قبل به تفصل معرفی گردیدند حال کافی است روش prewitt را هم معرفی کنم و سپس به دادن پاسخ سوالات پرسیده شده بپردازم.

روش prewitt:این روش تا حد زیادی شبیه روش Sobel است اما با این تفاوت که برخلاف کرنل های Sobel در این حالت کرنل های ما وزن کاملا برابری را به پیکسل های رنگی نسبت می دهند.

کرنل های این روش عبارت اند از:

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Prewitt Mask

طبیعتا این روش با توجه به سادگی زیادش،ازش دقت زیادی نیز انتظار نمی رود.اما خب طبیعتا سرعت بالایی دارد.

مقایسه روش های مختلف:

بخش خوبی از مقایسه ی روش ها در پاسخ سوال 3 بخش الف آمده است اما مجدد آن هارا ذکر می کنیم.

مقایسه نتایج:

Canny: از آنجاکه فیلترینگ Canny محاسبات زیادی دارد و جزییات زیادتری را در نظر میگیرد لذا همانطورکه انتظار داشتیم با ظرافت زیادی لبه ها را تشخیص می داد و هم چنین لبه ها دقیق مشخص می شدند با ضخامت کم!(که خوب است)برخلاف روش های دیگر که لبه ضخامت بیشتری داشت.

هم چنین این روش لبه هارا متصل(پیوسته) استخراج می کرد در صورتی که روش های دیگر لبه ها گاها تکه تکه بودند و ضریب پیوستگی(اتصالشان) کم بود و به علت ویژگی Double thresholding آن دستمان بیشتر باز است و لذا می توان بسته به عکس کیفیت لبه یابی را بیشتر هم کرد.

Sobel: از آنجاکه Sobel دقت کمتری دارد ولی خب محاسبات کمتری هم دارد! و در حد محاسبات خودش واقعا قابل قبول عمل می کند.اما توجه کنید چون مشتق گیری اش گسسته بوده است و گاهی وقتا دقتش کاهش پیدا می کند که در آن مواقع گفتیم که کرنل ها به کرنل های scharr تبدیل باید بشوند تا دقت حفظ شود.

Prewitt:

با توجه به مقاله ای(مقایسه ی قشنگ روش ها) که من راجع به تفاوت این روش ها خواندم(در زیر آورده شده است)به این نتیجه رسیدم که ما انواع سبک طبقه بندی برای Edge ها داریم(که کامل در مقاله معرفی شده است ولی برای جلوگیری از اطناب گزارش بنده لینک مقاله را صرفا آورده ام)و خب طبیعتا هدف از Edge detection تشخیص همه ی این انواع لبه هاست اما الگوریتم prewitt به علت اینکه وزن برابری در مشتق گیری می دهد لذا احتمالا فقط مشتق های تند را دیتکت می کند یعنی لبه های تند را تشخیص می دهد ولی لبه های نرم را اغلب با دقت خیلی کمتری تشخیص می دهد و خروجی ها هم صحت این استنتاج را تایید کردند و این الگوریتم اغلب لبه های تند تصویر را (که حاکی از رسیدن به مرز های یک segment عکس یا لبه های اندامک های یک عکس(بخش های متصلی که نسبت به هم تفاوت رنگی نسبتا شارپ دارند ولی همگی برای یک segment اند) را استخراج می کند و الباقی انواع لبه هارا می خورد!

نکته ی مهم:روش هایی که در ذات خود اعمال فیلترینگ گاوسی را داشتند(مانند Canny) خروجی شان تغییر نکرد(حداقل تغییر محسوسی نکرد) اما روش هایی که در ذات خود فیلترینگ گاوسی نداشتند همانند Sobel , Prewitt بهبود پیدا کردند و دقتشان در تعیین لبه بهتر شد.

روش هایی جهت حذف نویز و ارتقا لبه یابی:

1 ـ فیلترینگ گاوسی:این فیلترینگ انواع نویز هایی از قبیل نویز پوآسون و ... را تا حد خوبی بهتر می کند اما حواسمان باشد که اثر آن لبه هارا مات نکند و تیزی آن ها خیلی به هم نخورد.

2 ـ اپراتور های مورفولوجیکال از قبیل Dilation,Erosion,Opening and Closing که مزایای هر کدام در تمرین سری قبل به تفصیل ذکر گردید.مثلا Closing یا Opening با اعمال کرنل های مناسب اختلالات ناشی از نویز و propagation های

ارور محاسبات با دقت کم را تا حدی جبران می کند و مثلا بریدگی های اتصالات یک لبه را درست می کند و در کل با آسیب خیلی کمتری به لبه ها اثرات نویز را تعدیل می کند.