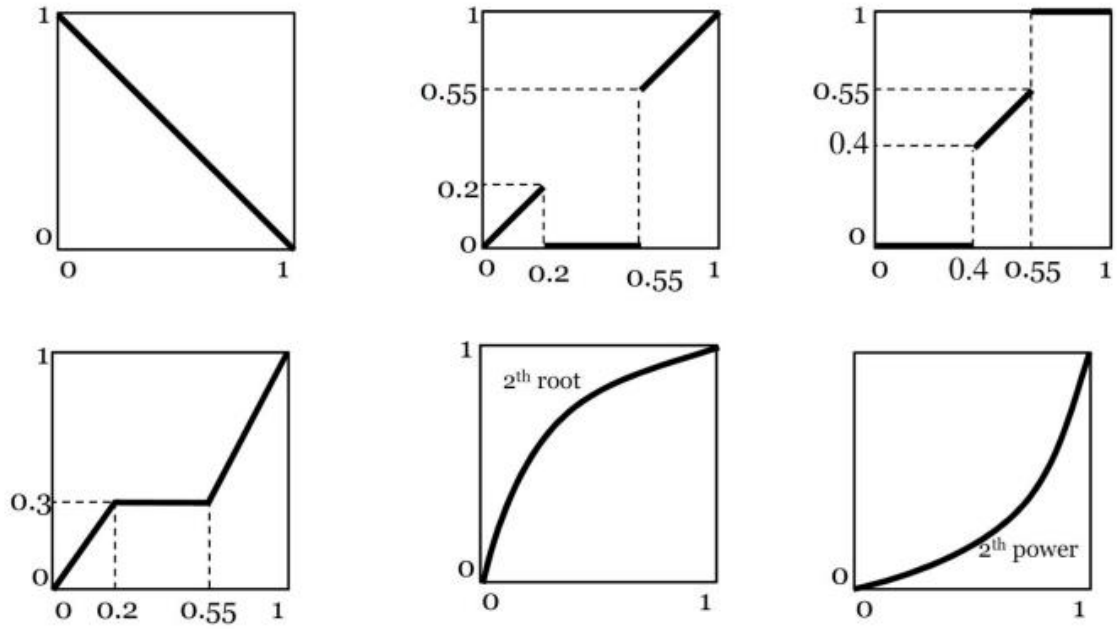
ـ بسم الله الرحمن الرحیم ـ

تمرین تحویلی شماره 1 - Multimedia

زهرا لطیفی 9923069

---

i. درباره کاربرد ها و حالت های خاص Contrast Stretching توضیح دهید. سپس با استفاده از تصویر ضمیمه شده img1.tif، خروجی تبدیلات زیر روی این تصویر را رسم کرده و تغییراتی که هرکدام در تصویر ایجاد میکند را توضیح دهید.



Contrast Stretching یا Normalization یکی از این عملیات‌های اعمالی بر روی تصاویر است که Contrast تصویر را بهبود می‌بخشد تا جزئیات موجود در تصویر با وضوح بیشتری دیده شوند. مثلا در تصاویر پزشکی به روش‌هایی مانند CT، MR، NM، اغلب برای مشاهده جزئیات پنهان در تصاویر، احتیاج داریم.

هدف، افزایش کنتراست بین تاریک‌ترین و روشن‌ترین پیکسل‌ها و در عین حال حفظ تفاوت‌های نسبی بین مقادیر میانی است. Contrast Stretching را می‌توان با استفاده از روش‌های مختلف مانند linear، logarithmic، exponential یا histogram equalization انجام داد.

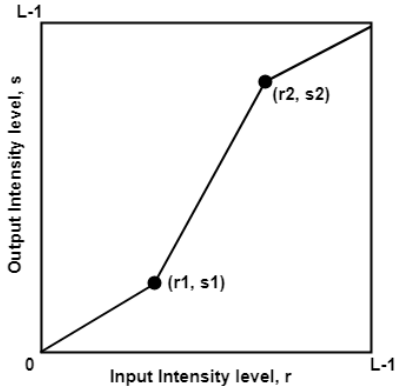
بسته به تابع تبدیل استفاده شده، روش های Contrast Enhancement را می‌توان به خطی و غیرخطی تقسیم کرد. روش خطی شامل Contrast Stretching است که از توابع خطی تکه‌ای استفاده می‌کند. در حالی که روش‌های

غیرخطی شامل Histogram Equalization, Gaussian Stretch و... است که از توابع تبدیل غیرخطی استفاده می‌کند که به طور خودکار از هیستوگرام تصویر ورودی به دست می‌آیند.

Contrast Stretching مستقیماً بر روی تصویر اعمال می‌شود و هر پیکسل موجود در تصویر را تغییر می‌دهد. (یعنی شیوه کار مبتنی بر استفاده از kernel ها نیست و مقدار هر پیکسل با استفاده از مقادیر پیکسل‌های همسایگی تعیین نمی‌شود و intensity هر پیکسل با استفاده از فرمول Normalization اصلاح می‌شود و محدوده intensity ها بزرگ‌تر می‌شود.)

همچنین اینکه Contrast Stretching مستقیماً روی تصویر اعمال می‌شود یعنی این فرایند شامل تبدیل تصویر به برخی از فرم‌های میانی و در نهایت اعمال یک تبدیل وارون برای بازگشت تصویر نیست. این عملیات، یک عملیات خطی است به این معنی که مقدار پیکسل جدید به صورت خطی بر اساس مقدار پیکسل اصلی تغییر کرده و به همین دلیل یک تصویر با کنتراست را می‌توان به تصویر اصلی تبدیل کرد. در واقع Contrast Stretching یک محدوده intensity موجود در تصویر را به محدوده intensity دیگری map می‌کند. البته محدوده intensity جدید باید با استفاده از هیستوگرام یک تصویر انتخاب شود تا مقادیر حداقل و حداکثر intensity موجود در تصویر بدون احتساب نقاط پرت به دقت انتخاب شوند.

برای مثال شکل زیر یک تابع تبدیل معمولی مورد استفاده برای Contrast Stretching را نشان می‌دهد.



اینجا با تغییر مکان نقاط (r1, s1) و (r2, s2) می توانیم شکل تابع تبدیل را کنترل کنیم. مثلا،

- وقتی r1=s1 و r2=s2 تبدیل به یک تابع خطی می شود.
- وقتی r1=r2، s1=0 و s2=L-1، تبدیل به یک تابع thresholding می‌شود.
- هنگامی که (r1, s1) = (rmin + c, 0) and (r2, s2) = (rmax – c, L-1) است، یعنی Min-Max Stretching داریم.
- هنگامی که (r1, s1) = (rmin + c, 0) و (r2, s2) = (rmax – c, L-1) است، یعنی Percentile Stretching داریم.

در Min-Max Stretching، مقادیر پایین و بالای تصویر ورودی به گونه‌ای ساخته می‌شوند که دامنه دینامیکی کامل را در بر می‌گیرد. به عبارت دیگر، مقدار پایین تصویر ورودی به 0 و مقدار بالایی به 255 map می‌شود. به همه مقادیر میانی دیگر، مقادیر intensity جدیدی مطابق فرمول‌هایی اختصاص داده می‌شود.

گاهی اوقات، هنگامی که Min-Max انجام می شود، انتهای دنباله هیستوگرام بلند می‌شود و در نتیجه کیفیت تصویر بهبود نمی‌یابد. بنابراین، بهتر است درصد مشخصی مانند 1٪، 2٪ از داده ها را از انتهای دنباله هیستوگرام تصویر ورودی clip کنید. این به عنوان Percentile Stretching شناخته می شود.

درنهایت باید توجه داشت که برای یک تصویر رنگی، باید آن را به greyscale تغییر داد و سپس Contrast Stretching را اعمال کرد یا آن را به مدل رنگی دیگری مانند HSV تغییر داد و سپس Contrast Stretching را روی V اعمال کرد.

در سوال اول مربوط به تمرین ما، اعمال نمودار اول از سمت چپ باعث می‌شود پیکسل های روشن تصویر، تیره شده و تیره‌ها روشن شوند. با مراجعه به فایل Q1_HW1_9923069 آپلود شده، image1 نتیجه را نشان می‌دهد که در آن شاهد تعویض پیکسل‌های روشن و تیره با هم هستیم.

اعمال نمودار دوم باعث می‌شود پیکسل‌هایی که intensity بین *2550.2 تا 0.55*255 دارند، کاملا سیاه شده و مابقی با تصویر اصلی یکسان باشند. برای این منظور، intensity هر پیکسل را بر 255 تقسیم کردیم تا مقادیر بین 0 و 1 قرار بگیرند و از اینجا به بعد با این مقادیر کار کردیم. نتیجه را در image2 می‌بینیم.

اعمال نمودار سوم باعث می‌شود پیکسل‌هایی که intensity بین 0 تا *2550.4 دارند، کاملا سیاه و بین 0.55*255 تا 255، کاملا سفید شده و مابقی با تصویر اصلی یکسان باشند. نتیجه را در image3 می‌بینیم.

چهارمین نمودار intensity پیکسل‌های بین *2550.2 تا 0.55*255 را روی 0.3*255 فیکس کرده و برای بازه پیش از آن، مقادیر را در 1.5 ضرب کرده و برای بازه پس از آن در 9/14 ضرب کرده و 9/5*255 از آن کم کرده است. نتیجه را در image4 می‌بینیم.

نمودار پنجم از intensity هر تصویر جذر گرفته. باید توجه داشت که چون مقادیر را بین 0 و 1 قرار دادیم، با جذر گرفتن، هرکدام بزرگتر شده و در نتیجه تصویر به طور کلی روشن‌تر خواهد شد. نتیجه در image5 قابل مشاهده است.

نمودار ششم intensity هر تصویر را به توان دو رسانده. باید توجه داشت که چون مقادیر را بین 0 و 1 قرار دادیم، با به توان رساندن، هرکدام کوچکتر شده و در نتیجه تصویر به طور کلی تیره‌تر خواهد شد. نتیجه در image6 قابل مشاهده است.

ii. تصویر img2.png را خوانده، هر کدام از کانال های رنگی R, G, B را جدا کرده و نمایش دهید. سپس هریک از کانال های رنگی را با استفاده از یک تبدیل Linear interpolation، تغییر دهید و تصویر جدید را همراه با هیستوگرام کانال های رنگی جدید آن رسم کنید. مشاهدات خود را از این تمرین بنویسید.

با مراجعه به فایل Q2_HW1_9923069 از تمرین آپلود شده، شاهدیم که نقاطی از تصویر که بیشتر آبی هستند، مانند آسمان، در تصویر Blue Channel Only نسبت به تصویر سایر کانال‌ها روشن‌تر هستند. که این یعنی درصد حضور آبی در آن نقاط بیش از سایر نقاط است. همین اتفاق درباره درخت‌ها در تصویر Green Channel Only و درباره ساحل در تصویر Red Channel Only افتاده است.

در بخش بعد گفته شده که به هر یک از کانال‌های رنگی یک interpolation خطی اعمال کنیم. interpolation تصویر زمانی اتفاق می‌افتد که اندازه تصویر خود را از یک پیکسل گرید به گرید دیگر تغییر می‌دهیم. تغییر اندازه تصویر هم زمانی لازم است که نیاز به افزایش یا کاهش تعداد کل پیکسل‌ها داریم. می‌دانیم که زوم کردن به افزایش تعداد پیکسل‌ها برمی‌گردد، به طوری که وقتی یک تصویر را زوم می‌کنیم، جزئیات بیشتری را مشاهده کنیم. در این سوال چون با افزایش سایز، تغییرات مشهودی را نمی‌دیدیم برای هر کانال از scaling دو دهم استفاده کردیم.

مشاهده می‌شود که تمام کانال ها کاهش کیفیت داشتند که در نتیجه 0.2 شدن تعداد پیکسل‌های هرکدام است.

هیستوگرام هرکدام هم مجزا و هم همزمان در یک نمودار رسم شده. این نمودار نشان می‌دهد مثلا intensity رنگ آبی از کمرنگ تا پررنگ در هر پیکسل به چه اندازه است.

حال هر سه کانال را مجدد merge کرده و تصویر را بازسازی کردیم. دیدیم که کیفیت 0.2 برابر شده.

درنهایت هیستوگرام این تصویر و تصویر اولیه را با هم رسم کردیم که نشان می‌دهد intensity ها در 0.2 ضرب شده اند.

iii. تصویر img3.jpg را خوانده و سعی کنید با انتخاب Morphological operator و Structuring element مناسب خطوط را از دایره ها جدا کنید. الگوریتم انتخابی میتواند شامل چندین مرحله morphological operation باشد. دلیل انتخاب های خود را ذکر کنید.

آیا میتوانید الگوریتمی مبتنی بر عملیات‌های Morphological ارائه دهید که خطوط یا دایره ها را از تصویر حذف کرده و هر کدام را در یک تصویر جداگانه ذخیر کنید؟

برای جدا کردن دایره ها ابتدا یک کرنل دایروی با اندازه‌ای بین ابعاد دایره ها و خط ها (بین 5 تا 15 پیکسل) و نزدیکتر به ابعاد دایره‌ها تعریف کردیم. سپس با عملیات erosion و این کرنل، نازک سازی انجام شد که در نتیجه خطوط را حذف کرده و با dilation مجدد دایره های باقی مانده را بزرگ کردیم. در نهایت با اعمال opening با حذف سفیدی های غیر دلخواه، نتیجه را بهبود دادیم و تصویر دایره های جدا شده را به نام "Circles.png" در پوشه آپلود شده ذخیره کردیم.

سپس این تصویر را از تصویر اصلی کم کردیم که در نتیجه به طور تقریبی خطوط باقی ماندند بجز نقاط مشترک که این بار از بین رفتند. اینجا باز هم با اعمال opening با حذف سفیدی‌های غیر دلخواه، نتیجه را بهبود دادیم. نتیجه به میزان کافی مطلوب نبود. پس این بار برای جدا کردن خطوط به طور جداگانه اقدام کردیم.

ابتدا دو کرنل افقی و عمودی تعریف کردیم و با اعمال آن به تصویر اصلی به صورت erosion و dilation، همه چیز را مگر خطوط کاملا افقی یا کاملا عمودی حذف کرده و نتیجه را با هم جمع کردیم تا خطوط کاملا افقی و عمودی تصویر جدا شده و نمایش داده شوند. از اینجا وارد یک حلقه for می‌شویم که در آن هربار تصویر اصلی را با scale یک حول مرکزش به اندازه 10 درجه (تا رسیدن به 90 درجه) می‌چرخانیم و بعد دوباره همان کرنل های افقی و عمودی را اعمال می‌کنیم. در اینجا خطوطی که با 10 درجه چرخش به افقی و عمودی کامل تبدیل شده اند، جدا می‌شوند. اما نباید این نتیجه را با نتایج قبلی جمع کرد. بلکه باید این تصویر را مجدد 10- درجه چرخاند تا خطوط مجدد به شکلی که در تصویر اصلی هستند در بیایند. (هم برای خطوط افقی و هم عمودی این کار را انجام دادیم.) حال می‌توان در هر مرحله نتایج با را نتایج قبلی جمع کرد تا کم کم تصویر اصلی بازسازی شود. سپس نتایج گام به گام را در فایل آپلود شده به نام Q3_HW1_9923069 نمایش دادیم.

نتیجه آخرین مرحله در برخی خطوط کمرنگ تر از تصویر اصلی بود. اینجا با یک حلقه for پیکسل هایی که intensity مثبت داشتند را سفید و مابقی را سیاه کردیم تا نتیجه مطلوب ما حاصل شد.

تصویر نهایی خطوط جدا شده را به نام "Lines.png" در پوشه آپلود شده ذخیره کردیم.

iv. بررسی smoothing filter های متفاوت برروی الگوریتم های لبه یابی. تصویر img4.jpg را خوانده، سپس Gaussian Filter, Median Filter و Bilateral Filter را روی آن اجرا کنید. حال اثر الگوریتم های لبه یابی Sobel و Canny را روی آن ها بررسی کنید. کدام فیلتر Smoothing در لبه یابی مناسب تر عمل کرده است؟ برای مقایسه معیار های زیر را در نظر بگیرید :

- **ضخامت لبه های تشخیص داده شده.**
- **عملکرد الگوریتم در تشخیص لبه های افقی و عمودی**
- **صحت لبه های تشخیص داده شده در دو تصویر. آیا لبه به درستی تشخیص داده شده است؟**

همچنین به صورت خلاصه هر یک از Filter ها را توضیح دهید.

فیلتر Median هر پیکسل را با پیکسل میانه یا «middle-valued» (برخلاف پیکسل میانگین) در یک ناحیه مستطیلی در اطراف پیکسل مرکزی جایگزین می‌کند. Blurring ساده با میانگین‌گیری ممکن است نسبت به تصاویر پر نویز حساس باشد، به ویژه تصاویری با داده‌های پرت بزرگ (مثلاً نویز عکس در عکاسی دیجیتال). اختلاف‌های زیاد حتی در تعداد کمی از نقاط می‌تواند باعث جابجایی قابل توجه در میانه شود. فیلتر Median قادر است با انتخاب نقاط میانی، نقاط پرت را نادیده بگیرد.

```
void cv::medianBlur(
 cv::InputArray src, // Input image
 cv::OutputArray dst, // Result image
 cv::Size ksize // Kernel size
);
```

فیلتر Gaussian شامل کانوالو هر نقطه در آرایه ورودی با یک کرنل گاوسی (نرمال شده) و سپس جمع کردن برای تولید آرایه خروجی است:

```
void cv::GaussianBlur(
 cv::InputArray src, // Input image
 cv::OutputArray dst, // Result image
 cv::Size ksize, // Kernel size
 double sigmaX, // Gaussian half-width in x-direction
 double sigmaY = 0.0, // Gaussian half-width in y-direction
 int borderType = cv::BORDER_DEFAULT // Border extrapolation to
use
);
```

برای Blurring گاوسی، پارامتر ksize عرض و ارتفاع پنجره فیلتر را نشان می‌دهد. پارامتر بعدی مقدار سیگما (نصف عرض در نصف حداکثر) کرنل گاوسی را در بعد x نشان می‌دهد. پارامتر چهارم به طور مشابه مقدار سیگما را در بعد y نشان می‌دهد. اگر فقط مقدار x را مشخص کنیم و مقدار y را 0 (مقدار پیش فرض آن) قرار دهیم، آنگاه مقادیر y و x برابر در نظر گرفته می‌شوند.

```
void cv::bilateralFilter(
 cv::InputArray src, // Input image
 cv::OutputArray dst, // Result image
 int d, // Pixel neighborhood size (max distance)
 double sigmaColor, // Width param for color weight function
 double sigmaSpace, // Width param for spatial weight function
 int borderType = cv::BORDER_DEFAULT // Border extrapolation to
use
);
```

فیلتر Bilateral یکی از عملیات‌هایی است که از کلاس بزرگ‌تری از عملگرهای آنالیز تصویر به نام edge-preserving smoothing شناخته می‌شود. فیلتر Bilateral زمانی که در مقابل smoothing گاوسی قرار گیرد به راحتی قابل درک است. یک هدف رایج برای Gaussian smoothing این است که پیکسل ها در یک تصویر واقعی باید به آرامی در فضا تغییر کنند و به این شکل با همسایه‌های خود مرتبط شوند، در حالی که می توان انتظار داشت نویز تصادفی از یک پیکسل به پیکسل دیگر بسیار متغیر باشد. (یعنی نویز از نظر فضایی همبستگی ندارد). از این نظر است که Gaussian smoothing ضمن حفظ سیگنال، نویز را کاهش می‌دهد.

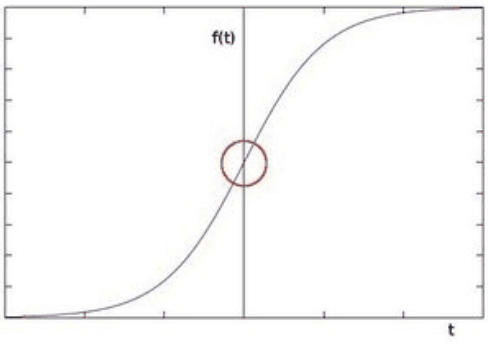
متأسفانه، این روش لبه‌های نزدیک را خراب می‌کند، جایی که ما انتظار داریم پیکسل‌ها با همسایه‌های خود در سراسر لبه همبستگی نداشته باشند. در نتیجه Gaussian smoothing لبه ها را محو می کند. با هزینه‌ای که متأسفانه زمان پردازش بسیار بیشتر است، فیلتر Bilateral ابزاری برای smoothing یک تصویر بدون صاف کردن لبه‌های آن فراهم می‌کند.

مانند Gaussian smoothing ، فیلتر Bilateral میانگین وزنی هر پیکسل و اجزای مجاور آن را می‌سازد. وزن دهی دارای دو جزء است که اولین آن همان وزن دهی است که توسط Gaussian smoothing استفاده می شود. مؤلفه دوم نیز یک وزن دهی گاوسی است، اما بر اساس فاصله مکانی از پیکسل مرکزی نیست، بلکه بر اساس تفاوت در intensity است. می‌توان فیلتر Bilateral را smoothing گاوسی در نظر گرفت که پیکسل‌های مشابه را بیشتر از پیکسل‌های کمتر مشابه وزن می‌کند و لبه‌های با کنتراست بالا را تیز نگه می‌دارد. اثر این فیلتر معمولاً تبدیل یک تصویر به چیزی است که به نظر می رسد یک نقاشی آبرنگ از همان صحنه است.

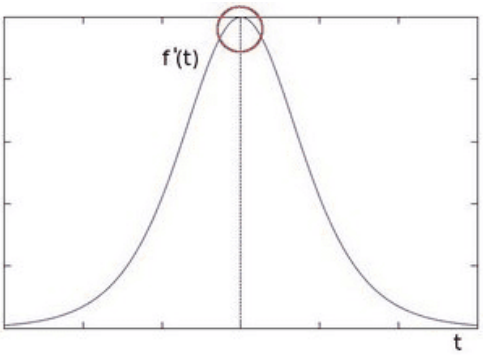
فیلتر Bilateral سه پارامتر (غیر از مبدا و مقصد) دارد. اولین مورد قطر d همسایگی پیکسل است که در هنگام فیلتر کردن در نظر گرفته می‌شود. دوم عرض کرنل گاوسی مورد استفاده در حوزه رنگی به نام sigmaColor است. سوم عرض کرنل گاوسی در حوزه فضایی به نام sigmaSpace است. هر چه پارامتر دوم بزرگتر باشد، دامنه intensity که در smoothing گنجانده می‌شود گسترده‌تر خواهد بود.

با مراجعه به فایل Q4_HW1_9923069 از تمرین آپلود شده، شاهد نتیجه اعمال این سه فیلتر بر تصویر اصلی هستیم. سپس دو تابع Sobel و Canny را تعریف کردیم تا فرایند Edge-Detection را انجام دهیم. این دو تابع را بر تصویر اصلی و نتیجه اعمال هر سه فیلتر، اعمال کردیم و نتایج را مقایسه کردیم.

Sobel Edge Detection یکی از پرکاربردترین الگوریتم‌ها برای تشخیص لبه است. اپراتور Sobel لبه‌هایی را که با تغییرات ناگهانی در intensity پیکسل مشخص شده اند، شناسایی می کند.



وقتی اولین مشتق تابع intensity را رسم می کنیم، افزایش شدت حتی بیشتر مشهود خواهد بود.



نمودار بالا نشان می دهد که لبه ها را می توان در مناطقی که گرادیان بالاتر از یک مقدار threshold خاص است شناسایی کرد. علاوه بر این، یک تغییر ناگهانی در مشتق، تغییر در intensity پیکسل را نیز آشکار خواهد کرد. با در نظر گرفتن این موضوع، می توانیم مشتق را با استفاده از یک کرنل 3×3 تقریب بزنیم. از یک کرنل برای تشخیص تغییرات ناگهانی در intensity پیکسل در جهت X و دیگری در جهت Y استفاده می‌کنیم. کرنل‌هایی که برای تشخیص لبه Sobel استفاده می شوند به این شکلند:

$$\text{X-Direction Kernel} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\text{Y-Direction Kernel} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

وقتی این کرنل‌ها با تصویر اصلی کانوالو می‌شوند، یک «Sobel edge image» حاصل خواهد شد. اگر فقط از کرنل عمودی استفاده کنیم، کانولوشن یک تصویر Sobel با لبه‌های افزایش یافته در جهت X ایجاد می‌کند و با استفاده از کرنل افقی، یک تصویر Sobel به دست می‌آید که لبه‌های آن در جهت Y افزایش یافته است.

دستور زیر برای اعمال تشخیص لبه Sobel با استفاده از OpenCV است:

```
cv2.Sobel(src, ddepth, dx, dy, ksize)
```

پارامتر ddepth دقت تصویر خروجی را مشخص می‌کند، در حالی که dx و dy ترتیب مشتق را در هر جهت مشخص می کنند. مثلا:

اگر dx=1 و dy=0 باشد، اولین مشتق تصویر Sobel را در جهت x محاسبه می کنیم.

اگر هم dx=1 و هم dy=1، اولین مشتق تصویر Sobel را در هر دو جهت محاسبه می‌کنیم.

نتایج اعمال این اپراتور در راستای X و Y و هردو در فایل مذکور آمده. اگر به خطوط بدن ببر در هر دو تصویر دقت کنیم، می‌بینیم که چگونه لبه‌های عمودی قوی خطوط در تصویر Sobel در جهت x بیشتر مشهود است و لبه‌های افقی در جهت Y. شکل XY هم تصویر Sobel را برای گرادیان در هر دو جهت نشان می‌دهد، که تصویر اصلی را به یک تصویر از تنها لبه‌ها تبدیل می‌کند، به طوری که یکپارچگی ساختاری آن دست نخورده باقی می‌ماند.

Canny Edge Detection یکی از محبوب‌ترین روش‌های تشخیص لبه است که امروزه مورد استفاده قرار می‌گیرد زیرا بسیار قوی و انعطاف پذیر است. خود الگوریتم یک فرآیند سه مرحله‌ای را برای استخراج لبه‌ها از یک تصویر دنبال می‌کند. اضافه کردن Blurring به تصویر، یک مرحله پیش پردازش ضروری برای کاهش نویز است. پس درواقع یک فرآیند چهار مرحله‌ای داریم شامل:

1. Noise Reduction
2. Calculating the Intensity Gradient of the Image
3. Suppression of False Edges
4. Hysteresis Thresholding

دستور زیر برای اعمال Canny Edge Detection با استفاده از OpenCV است:

```
cv2.Canny(image, threshold1, threshold2)
```

دو مقدار گرفته شده که در مرحله چهارم به کار برده می‌شوند، دو مقدار threshold هستند. مقادیر گرادیان با این دو مقدار مقایسه می‌شوند که یکی کوچکتر از دیگری است.

اگر مقدار گرادیان از مقدار آستانه بزرگتر باشد، آن پیکسل ها با لبه‌های solid مرتبط می‌شوند و در نقشه لبه نهایی گنجانده می شوند و اگر مقادیر بزرگی گرادیان کمتر از مقدار آستانه کوچکتر باشد، پیکسل ها از نقشه لبه نهایی حذف می‌شوند.

تمام پیکسل‌های دیگر، که قدر شیب آنها بین این دو آستانه قرار می‌گیرد، به‌عنوان لبه‌های «ضعیف» علامت‌گذاری می‌شوند. (یعنی گزینه احتمالی ای برای قرار گرفتن در نقشه لبه نهایی می‌شوند.)

اگر پیکسل های "ضعیف" به پیکسل های مرتبط با لبه های solid متصل شوند، در نقشه لبه نهایی نیز گنجانده می‌شوند.

ما از یک آستانه پایین 100 و یک آستانه بالایی 200 استفاده کردیم که نتایج آن در فایل مذکور موجود است. دیدیم که چون فیلتر median کمتر لبه ها را از بین برده، اعمال الگوریتم بر روی تصاویر حاصل از آن بهتر نتیجه داده است.

همینطور با توجه به نتایج، Canny Edge Detection بهترین نتیجه را ایجاد می‌کند زیرا نه تنها از تشخیص لبه Sobel بلکه از Suppression of False Edges و Hysteresis Thresholding نیز استفاده می‌کند. این انعطاف‌پذیری بیشتری را در نحوه شناسایی و اتصال لبه‌ها در مراحل نهایی الگوریتم فراهم می‌کند.

به جهت ضخیم تر بودن، Canny به وضوح بهتر عمل کرده اما اعمال Sobel تنها در راستای یکی از محورها در تشخیص لبه های افقی و عمودی بهتر بوده است.