



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

پاسخ تمرین دوم بینایی کامپیوتو
استاد درس جناب آقای دکتر صفابخش
نیمسال اول سال تحصیلی ۱۴۰۱-۱۴۰۲

محسن عبادپور

شماره دانشجویی:

۴۰۰۱۳۱۰۸۰

ایمیل: m.ebadpour@aut.ac.ir

فهرست پاسخ ها

۲	مسئله اول
۴	مسئله دوم
۱۰	مسئله سوم
۱۲	مسئله چهارم
۱۳	مسئله پنجم

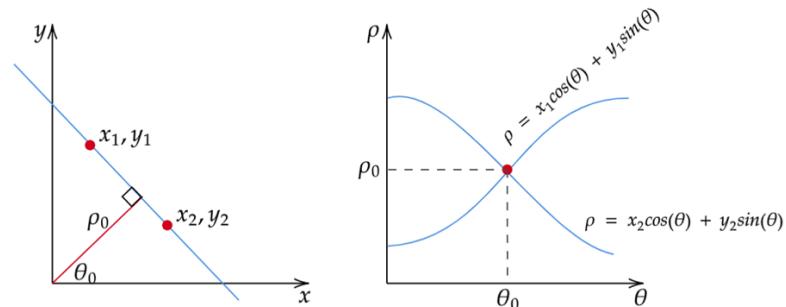
مسئله اول

مسئله اول

برای پاسخ گویی به این سوال و بررسی تابع HoughLinesP از مرجع رسمی opencv به این آدرس استفاده شده است؛ این تابع دارای شش پارامتر ورودی می باشد که به صورت زیر عمل می نماید:

Image(1): این پارامتر یک تصویر ورودی تک کاناله می باشد که میخواهیم فرآیند شناسایی خط روی این تصویر صورت گیرد و همان طور که در کلاس درس بررسی و در متن سوال نیز اشاره شده است بایستی تصویر مذکور همان لبه های استخراجی از تصویر اصلی در نظر گرفته شود.

Rho(2): این پارامتر نشان دهنده گام و اندازه‌ی تغییرات فاصله خط از نقطه‌ی مبدأ می‌باشد؛ مثلا در شکل زیر، محور عمودی مربوط به تغییرات مذکور می‌باشد. این محور پیوسته بوده و آزمایش تمامی حالات ممکن هزینه و زمان بر می‌باشد لذا بایستی این را گسسته کرده و در گام‌هایی محدود آن را آزمایش نمود که چه خط‌های معادلی وجود دارد. حال اگر مقدار یک برای آن در نظر بگیریم، یعنی فاصله‌ی بین مرکز تا خطوط مختلف یک پیکسل یک پیکسل تغییر یابد و در یک زاویه‌ی ثابت اگر بخواهیم خط بعدی (دورتر) را بدست آوریم بایستی فاصله کنونی را یک پیکسل افزایش دهیم.



theta(3): این پارامتر نیز نشان دهنده گام و اندازه‌ی تغییرات زاویه‌ی خط نسبت به نقطه‌ی مبدأ می‌باشد؛ مثلا در شکل بالا، محور افقی مربوط به تغییرات مذکور می‌باشد. این محور پیوسته بوده و آزمایش تمامی حالات ممکن هزینه و زمان بر می‌باشد لذا بایستی این را گسسته کرده و در گام‌هایی محدود آن را آزمایش نمود. حال با فرض ثابت بودن فاصله‌ی

خط از نقطه‌ی مبدا بخواهیم مقدار $\frac{\pi}{180}$ را برای این پارامتر در نظر بگیریم بدان معناست که اگر بخواهیم زاویه خط را یک واحد تغییر داده و آزمایش کنیم، آن یک واحد برابر با یک درجه مثلثاتی می‌باشد.

: این پارامتر یک حد آستانه برای این است که حداقل بایستی چند منحنی در فضای Hough از یک نقطه‌ی 4 معین عبور کنند تا آن نقطه به عنوان مختصات یک خط در فضای مکانی انتخاب شود.(به عبارتی دیگر بایستی حداقل چند منحنی به مختصات رای بدهد)

: این پارامتر یک حد آستانه برای کمینه‌ی طول خط‌های شناسایی شده می‌باشد و هر چه این پارامتر 5 بزرگتر باشد، خط‌های شناسایی شده بایستی دارای طول‌های بلندتر باشد و هر چه این حد آستانه پایین باشد، خط‌های شناسایی شده میتوانند طول‌های کوتاه‌تری داشته باشند.

: بخشی از یک خط مفروض در یک تصویر ورودی میتواند بریده یا تقطیع شده باشد و بین پیکسل‌های 6 یک خط فاصله بیافتد؛ این پارامتر تنظیم می‌کند که حداقل فاصله‌ای که بین نقاط یک خط می‌تواند به وجود آید چقدر است. هرچه قدر این پارامتر بزرگتر باشد یعنی اجازه می‌دهیم فاصله‌ی بیشتری بین قطعه‌های یک خط به وجود آید و بالعکس.

نسخه‌ی بھبود و بهینه یافته تبدیل Hough Transform یا HoughLinesP می‌باشد؛ در تبدیل Probabilistic Hough Transform یا HoughLines همانی ترکیب‌های ممکن در فضای Hough آزمایش می‌شود که زمان بر بوده و دارای بار محاسباتی و پردازشی می‌باشد. در نسخه Probabilistic Hough Transform نیاز نیست که همه ترکیب‌های ممکن بررسی شوند و برای این منظور کافی است تعدادی نقاط تصادفی از مجموعه نقاط فضا انتخاب و بررسی روی آن انجام گردد و بدیهی است که هر چقدر توزیع تصادفی بودن بین نقاط همگین باشد، نتیجه‌ی بهتری نیز حاصل خواهد شد. نکته‌ی بعدی این است که در حالت استفاده از Probabilistic Hough Transform پایین تر انتخاب شود چرا که در کل منحنی‌های کمی وجود خواهد داشت لذا این حد آستانه نیز نسبت به حالت پایه بایستی کمتر انتخاب شود.

به جهت پیاده سازی نیز تفاوت بعدی که وجود دارد این است که پارامتر بازگشتی در HoughLines مقادیر فضای (زاویه و فاصله) می‌باشد اما درتابع HoughLinesP مقادیر بازگشتی مختصات دو نقطه‌ی خط شناسایی شده می‌باشد.

مسئله دوم

مسئله دوم

برای حل این قسمت ابتدا تصویر را بارگذاری کرده و سپس با حفظ نسبت تصویر را $1/3$ ابعادش را کاهش داده ایم؛ سپس تصویر را به سطح خاکستری تبدیل می کنیم که نتایج بصورت زیر حاصل می شود:



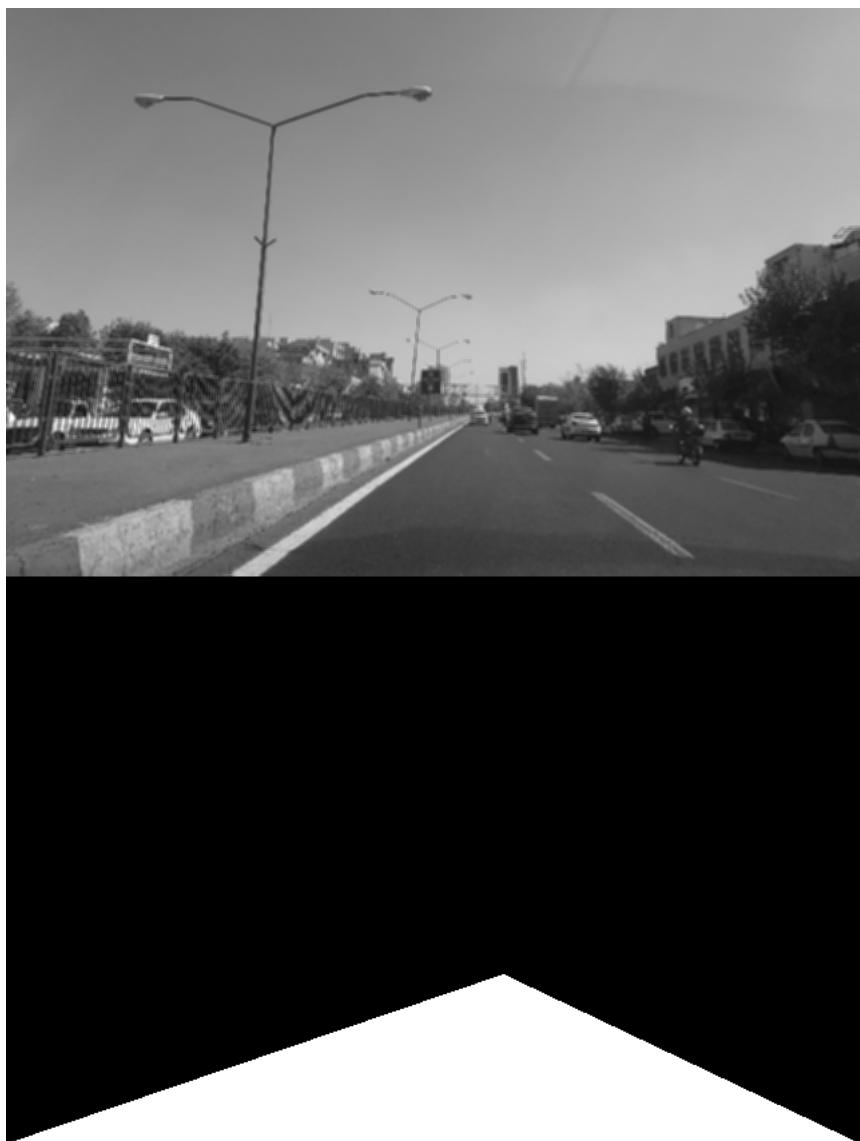
حال بایستی فیلتر مناسبی برای هموارسازی انتخاب نموده و پس از اعمال آن، فرآیند لبه یابی توسط Canny را انجام می دهیم. انتخاب پارامتر های مناسب برای نوع و پارامتر های فیلتر هموارساز و پارامتر های لبه یاب حائز اهمیت بوده و بایستی با دقت انجام

شود؛ از این رو، با استفاده از مکانیزم interactive notebook های jupyter موجود در time انتخاب شده است؛ تصویر زیر مربوط به انتخاب پارامتر های انتخابی می‌باشد:



فیلتر های هموارساز مورد بررسی عبارت اند از میانگین‌گیر، میانه، گوسی و Bilateral که به ازای اندازه کرنل های متفاوت مورد آزمایش قرار گرفته است؛ بر حسب آزمایش های صورت گرفته فیلتر هموار ساز میانگین‌گیر با اندازه کرنل 3×3 بهترین نتیجه را به همراه داشته است؛ همچین مقدار آستانه های Canny نیز در بازه‌ی ۱-۵۰۰ قابل تغییر و بررسی بود که به ازای فیلتر هموارساز انتخابی، به ترتیب ۱۱۰ و ۲۶۰ انتخاب شد تا لبه های استخراجی مناسبی حاصل شود که در تصویر فوق قابل مشاهده است.

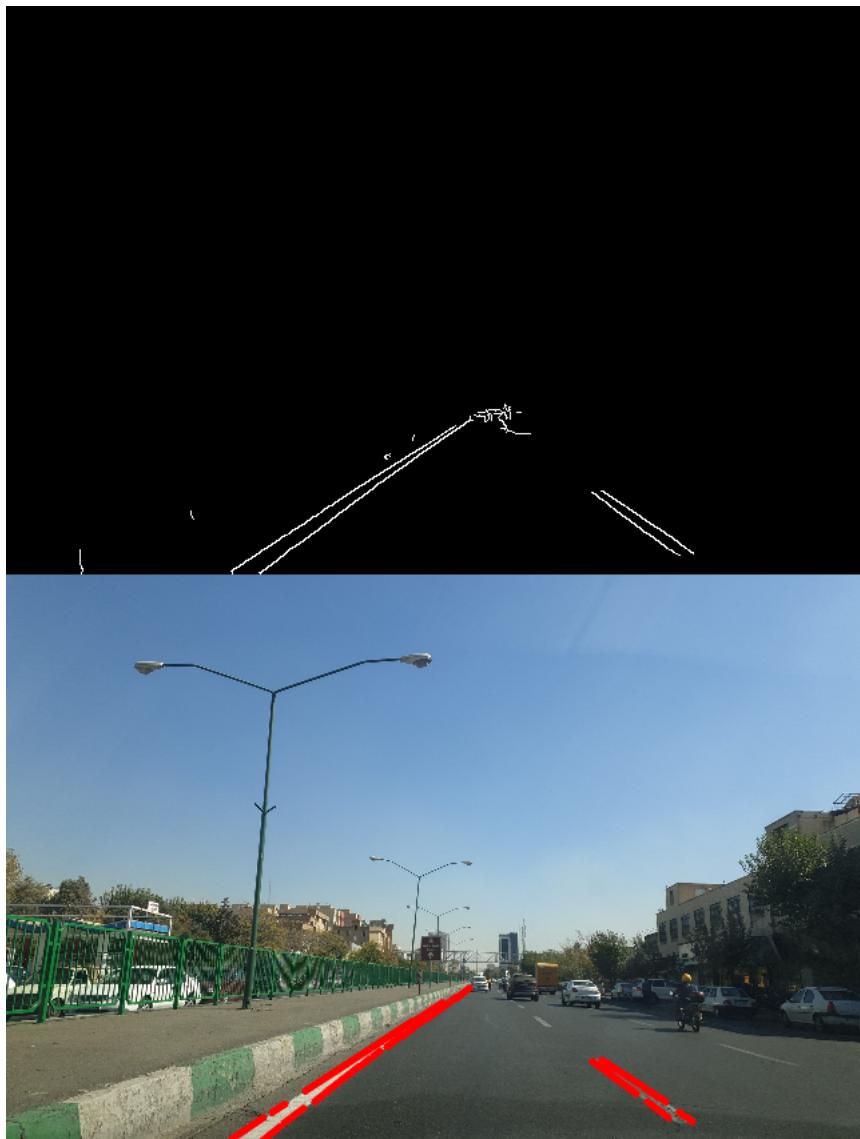
حال تصویر سطح خاکستری را با فیلتر هموارساز انتخابی(میانگین گیر با کرنل ۳*۳) هموار می کنیم(تصویر بالا در خروجی زیر). در مرحله‌ی بعد، برای ایجاد ماسک برای فضای دید ثابت دوربین از سه نقطه استفاده کرده و با استفاده از تابع fillPoly اقدام به تولید ماسک می کنیم که بصورت زیر نتایج قابل مشاهده است. نقطه وسط مثلث دقیقاً تصویر قرار نگرفته است چرا که بیشتر از میدان دید دوربین بوده و در صورت انتخاب وسط تصویر، فضای بیشتری مد نظر قرار میگیرد لذا در راستای عمود، پایین تر از وسط قرار گرفته است. در راستای افق نیز دوربین مایل به سمت راست بوده لذا نقطه وسط در راستای افق دقیقاً وسط تصویر نبوده و در سمت راست‌تر انتخاب شده است. برای ایجاد ماسک یک تابع با عنوان GetMask پیاده سازی شده است تا بتوان با پارامتر های تابع نقاط ماسک را جا به جا نمود.



حال تصویر را سطح خاکستری شده که روی آن هموارسازی نیز صورت گرفته است را با Canny (با پارامتر های ذکر شده در توضیحات فوق) اقدام به لبه یابی کرده و سپس با اعمال ماسک ناحیه مدنظر را جدا می کنیم که بصورت زیر حاصل می شود:



بایستی با استفاده از الگوریتم Hough اقدام به تعیین خطوط لبه بکنیم؛ برای بالا رفتن دقت شناسایی خط، تصویر ماسک شده‌ی thresholding به تصویر دودوئی تبدیل می‌کنیم. حال با استفاده از تابع HoughLinesP که در سوال اول مورد بررسی قرار گرفت اقدام به شناسایی خطوط می‌کنیم. پارامتر ها همانند قسمت قبل با سعی و خطا انتخاب شدند؛ برای پارامتر threshold و maxLineGap و minLineLength به ترتیب مقادیر 30 و 10 و 5 در نظر گرفته شده است. خروجی های دودوئی شده و خطوط شناسایی شده در تصویر بعد قابل ملاحظه شده است:



در این مرحله بایستی خطوط شناسایی شده را پیوسته نموده و دو خط برای طرفین بدست آورد. برای این منظور شبکه خطاها شناسایی شده را محاسبه می‌کنیم و سپس با K-Means آنها را در دو دسته خوشبندی می‌کنیم. از جایی که علامت شبکه خطوط متفاوت است(سمت راستی ها منفی و سمت چپی ها مثبت) هر کدام از خوشبندی ها حاوی خط های شناسایی شده‌ی یک طرف بوده و میانگین آن خوشبندی میانگین خط آن طرف می‌باشد که توسط یکی از نقاط مربوطه، خط را در صفحه رسم و با ماسک مرحله‌ی قبل، قسمت مربوط به میدان دید دوربین را جدا و خطوط را ارائه میدهیم. برای این منظور تابع GetDetectedLines پیاده سازی شده است تا بتوان در سوال بعد قدرت مانور بیشتری داشت. نتایج نهایی بصورت زیر حاصل می‌شود:



مسئله سوم

مسئله سوم

همانطور که در توضیحات سوال قبل ملاحظه شد برای قسمت های مختلف توابعی پیاده سازی شده است تا بتوان در این مسئله از آنها استفاده نمود. برای هر دو ویدیو نتایج قابل قبولی حاصل شده است که ویدیوها در پوششی outputs قرار گرفته است. برای هر دو ویدیو مقادیر تعیین ماسک تغییر پیدا کرده است چرا که جهت دوربین هر یک از خودرو ها نیز اندکی متفاوت است.

پارامترهای مورد استفاده برای شناسایی خطوط در ویدیوی اول برای HoughLinesP به ترتیب $40, 50$ و 10 برای threshold و maxLineGap و minLineLength در نظر گرفته شده است. در این ویدیو برای افزایش عرض خطوط در لبه های استخراجی دودوئی شده از عملیات های dilation (۲ تکرار) و erosion (۱ تکرار) به ترتیب استفاده میکنیم و سپس اقدام به شناسایی خطوط با Hough می کنیم؛ همچنین برای جلوگیری از نویز های احتمالی در شناسایی خطوط یا عدم شناسایی خط از میانگین وزن دار خطوط شناسایی شده در هر فریم و شبیب پنج فریم قبلی استفاده شده است. این کار باعث می شود جهش نیز رخ ندهد چرا که در واقعیت نیز مسیر یکهو و در عرض یک فریم مأورایی تغییر پیدا نمی کند. یک فریم پردازشی از ویدیوی اول در زیر قابل ملاحظه است:



برای ویدیوی دوم نیز همانند ویدیوی اول اقدام شده است و پارامتر های انتخابی نیز مشابه در نظر گرفته شده است(به غیر از مختصات مثلث در ماسک). اما وجود مه در ویدیوی دوم فرآیند تشخیص خط را با چالش مواجه کرده است. برای بهبود فرآیند HSV تشخیص خطوط، ابتدا و در همان تصویر رنگی ماسک را اعمال نموده و سپس اقدام به هموارسازی هیستوگرام در فضای HSV می کنیم تا تاثیر مه کمتر شده و تفکیک بین خطوط و مه تا حدی ایجاد شود اما با این وجود عرض لبه های استخراجی کم است و برای رفع این مشکل مجددا از *dilation* و *erosion* استفاده می کنیم با این تفاوت که ۳ بار *dilation* و سپس صرفاً یکبار *erosion* می کنیم تا عرض بیشتری از خطوط باقی بماند. همچنین در این قسمت نیز از میانگین وزن دار فریم های قبلی نیز استفاده می کنیم که در اینجا به جای ۵ فریم قبلی از ۱۵ فریم قبلی استفاده می کنیم چرا که عبور مه کمی طول می کشد. یک فریم پردازشی از ویدیوی دوم در زیر قابل ملاحظه است:



تصویر سمت چپ بالا مربوط به هموارسازی هیستوگرام، تصویر سمت راست بالا مربوط به شناسایی لبه در قسمت ماسک شده تصور، تصویر سمت چپ پایین مربوط به خطوط شناسایی شده و تصویر سمت راست پایین مربوط به خط تخمینی می باشد.

مسئله چهارم

مسئله چهارم

برای پاسخ گویی به این سوال از مرجع رسمی در [این لینک](#) استفاده می کنیم؛ این تابع در مجموع دارای ۱۱ پارامتر ورودی می باشد که در زیر مورد بررسی قرار می دهیم:

(۱) image: این پارامتر ورودی تصویر می باشد که می خواهیم فرآیند تشخیص روی آن انجام شود.

(۲) Snake: این پارامتر مختصات اولیه کانتور های فعال را ورودی می گیرد.

(۳) alpha: این پارامتر سرعت حرکت کانتورهای فعال به سمت لبه را مشخص می کند.

(۴) beta: مقدار نرم بودن کانتورهای پیدا شده را مشخص می کند و تعیین می کند که smooth باشد یا بتواند تکه تکه باشد.

(۵) w_line: این پارامتر به مقدار پیکسل ها وزن دهی می کند به گونه ای که با دادن مقادیر پایین تر و منفی کانتور های فعال به سمت نواحی تاریک جذب می شود.

(۶) w_edge: این پارامتر به لبه های موجود در شکل وزن دهی می کند تا کانتور های فعال به لبه ها تمایل بیشتری داشته باشند.

(۷) gamma: این پارامتر قدم های زمانی حرکت کانتورهای فعال را مشخص می کند.

(۸) max_px_move: این پارامتر مشخص می کند که کانتور های فعال در هر مرحله از تکرار حداکثر چه تعداد پیکسل مجاز به حرکت و جا به جایی هستند.

(۹) max_num_iter: این پارامتر تعیین کننده تعداد تکرار برای بررسی و جا به جایی کانتور های فعال می باشد.

(۱۰) convergence: این پارامتر ضریب همگرایی به شرایط پایداری را تعیین می کند که آیا با توجه به وضعیت کنونی نسبت به وضعیت قبلی کانتورها همچنان نیاز به ادامه می باشد یا خیر.

(۱۱) boundary_condition: این پارامتر وضعیت تعامل با کانتورهای انتهایی snake را مشخص می کند، مثلا در حالت free الگوریتم میتواند نقاط انتهایی و میانی را جا به جا کند اما در حالت fixed فقط نقاط میانی را میتواند جابه جا کند.

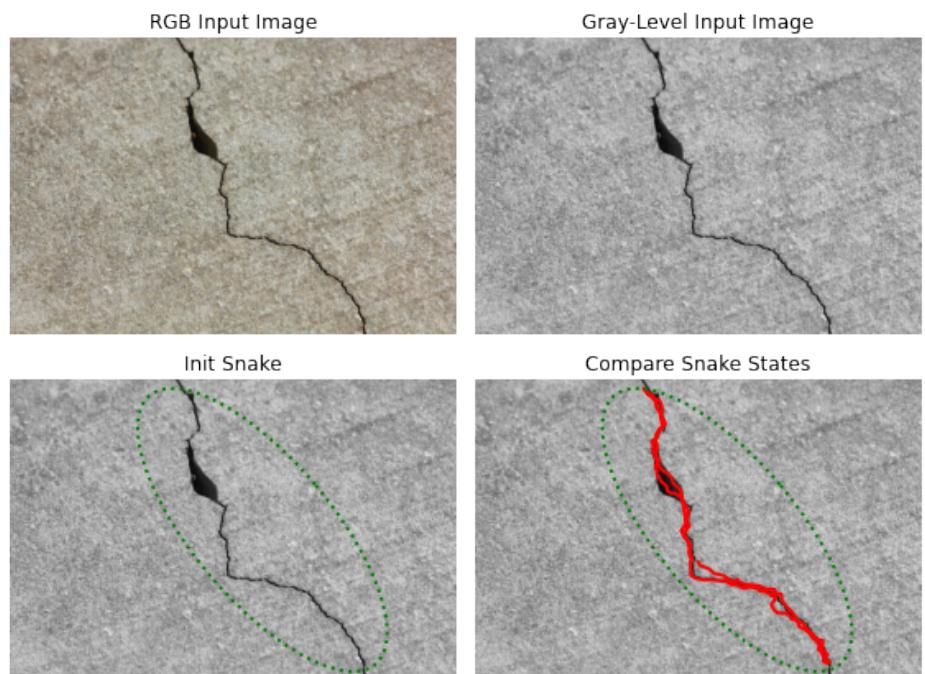
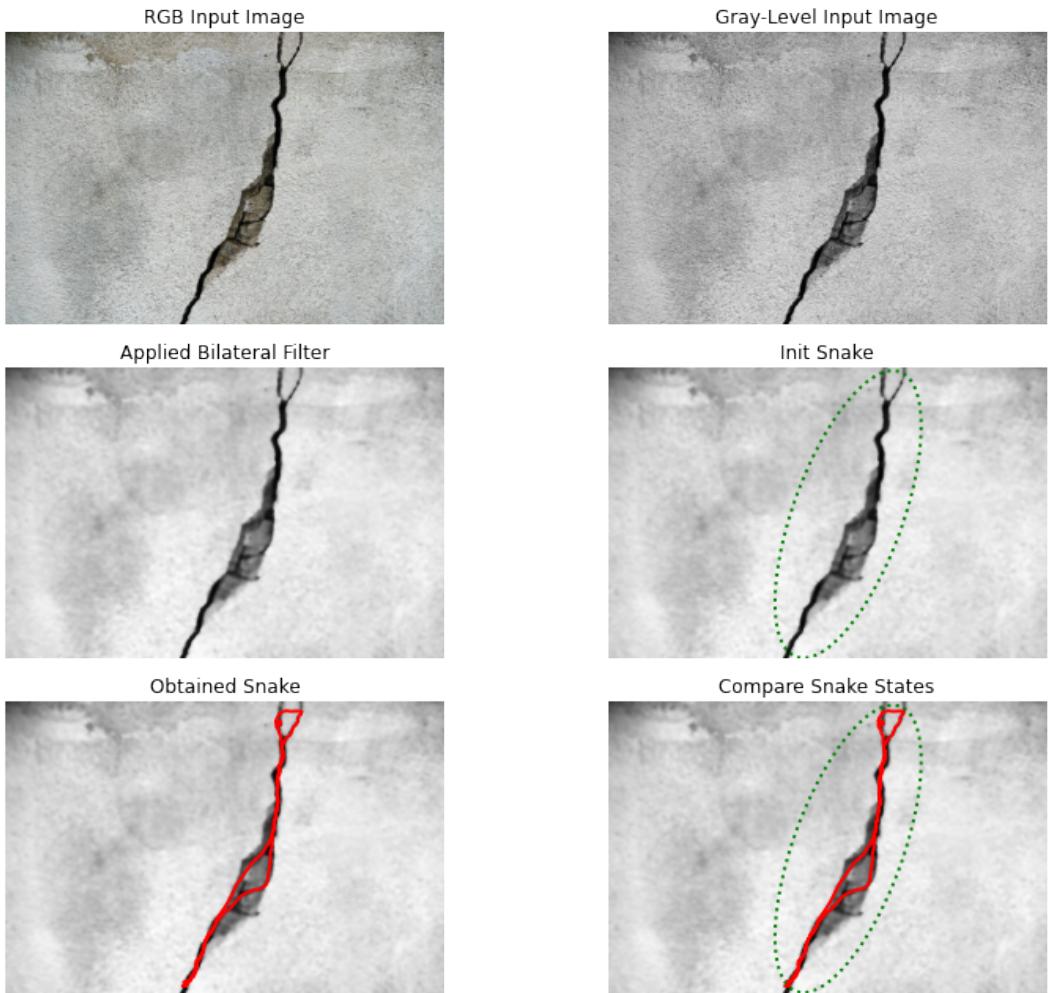
مسئله پنجم

مسئله پنجم

ابتدا تصویر img3.jpg را بارگذاری کرده و سپس آن را به سطح خاکستری تبدیل می‌کنیم. حال برای از بین بردن نویز های ریز حاصل از سطح سنگی و بهبود عملکرد active contour model از فیلتر Bilateral برای هموارسازی تصویر استفاده میکنیم تا کانتور های فعال با چالش کمتری مواجه شوند سپس توسط دوتابع سینوس و کسینوس از کتابخانه numpy یک بیضی مناسب snake ایجاد و با چرخش مناسب(توسط ماتریس دوران) آن را حول ترک موجود در تصویر قرار می دهیم که این بیضی محل اولیه را تعیین می کند. سپس با استفاده از تابع ذکر شده در متن سوال(active_contour) اقدام به تقطیع ترک موجود در تصویر می کنیم. برای تعیین پارامتر ها بیش از ۴ ساعت سعی و خطا انجام شده است که در نهایت برای alpha (سرعت حرکت کانتور) و gamma (گام زمانی حرکت کانتور) مقدار ۰.۰۱، برای beta (نرم بودن بدنی snake) مقدار ۰.۱، برای w_line (جذب کانتور به نواحی تاریک) مقدار ۲، برای w_edge (تأثیر دادن لبه های تصویر) مقدار ۲، برای max_px_move (حداکثر تعداد حرکت کانتور در یک تکرار با واحد پیکسل) مقدار ۳ و برای max_num_iter (حداکثر انجام تکرار برای بروزرسانی وضعیت کانتور ها در صورت عدم همگرایی) مقدار ۵۰۰۰ در نظر گرفته شده است. ضمناً تعداد نقاط snake نیز ۲۲۵ تعیین شده است.

برای تصویر img2.jpg نیز فرآیند مشابهی صورت گرفته است که پارامتر های مورد نیاز دیگری بدست آمده است. برای این تصویر چون عرض ترک کم است، فرآیند هموارسازی سطح خاکستری تصویر انجام نشده است. برای تعیین پارامتر های این بخش نیز با وجود تجربه‌ی تصویر قبلی بیش از ۲ ساعت سعی و خطا انجام شده است که در نهایت برای alpha (سرعت حرکت کانتور) و gamma (گام زمانی حرکت کانتور) مقدار ۰.۰۲، برای beta (نرم بودن بدنی snake) مقدار ۰.۱، برای w_line (جذب کانتور به نواحی تاریک) مقدار ۲، برای w_edge (تأثیر دادن لبه های تصویر) مقدار ۲، برای max_px_move (حداکثر تعداد حرکت کانتور در یک تکرار با واحد پیکسل) مقدار ۴، برای max_num_iter (حداکثر انجام تکرار برای بروزرسانی وضعیت کانتور ها در صورت عدم همگرایی) مقدار ۵۰۰۰ و برای convergence (همگرایی) نیز مقدار ۰.۰۰۵ در نظر گرفته شده است. همچنین تعداد نقاط snake نیز ۱۸۰ تعیین شده است.

در صفحه‌ی بعد پردازش های اعمالی و نتیجه‌ی حاصل آورده شده است.



طبق توضیحات و بررسی های انجام شده در کلاس درس، دیدیم که مدل کانتور فعال یک باند(Snake) تغییرپذیر حساس و مبتنی بر شدت گرادیان می باشد؛ در این روش ابتدا حول و نزدیک مرز شی مد نظر باندی قرار می گیرد(که میتواند از مرز داخل هم باشد) و ذرات این باند به سمتی از همسایگی خود که شدت گرادیان در آن سو بیشتر است حرکت میکنند تا در آن جا گیرند چرا که بصورت منطقی در مرز اشیا که لبه نیز هستند حداکثر گرادیان وجود دارد.

زمانی که مرز های شی به هم نزدیک نیستند هر کدام از ذرات Snake به سمت مرز حرکت کرده و آنجا سکون پیدا میکنند(شدت گرادیان بالاست) و در نهایت شی تقطیع شده از زمینه را بازنمایی میکنند؛ اما در خصوص اشیا یا نواحی ای مانند ترک ها که مرز های نزدیک به همی دارند این روش نمیتواند به جواب مناسبی برسد چرا که وقتی ذرات از پیرامون به سمت مرز(گرادیان با شدت بالا) حرکت کرده و به آنجا می رسند، در همسایگی خود مجدد ناحیه با شدت گرادیان بالا را ملاحظه میکنند که همان سوی دیگر مرز نازک می باشد لذا از مرز دقیق اولی عبور کرده و به سمت مرز دوم بصورت نوسانی متمایل می شوند چرا که در همسایگی خود همچنان گرادیان با شدت بالا را مشاهده کرده و حرکت را ادامه می دهند و بدین گونه مرز دقیق اول یافت شده از دست می رود و ضعف مدل کانتور فعال نمایان می شود. به طور خلاصه می توان بیان داشت که ضعف اصلی مدل کانتور فعال در مقابله با ساختار های نازک یا کشیده این است که شدت گرادیان های بالا در مرز ها به هم نزدیک بوده و سوی حرکت ذرات snake به خط می افتد.