گزارش تمرین سری اول دید کامپیوتری

نام: محمدجواد طاهری

شماره دانشجویی: 98205849

آدرس فایل ریپو: https://github.com/mjtaheri11/hw1.1.git

سوال 1:

در این سوال عکس را در ابتدا خوانده و سپس شماره دانشجویی را در گوشه تصویر درج میکنیم. صرفا از دستور cv2.putText() استفاده شده است. در ادامه نیز همانطور که صورت سوال خواسته است با استفاده از کلید e تصویر ها را میبندیم. اینجا سعی شد که هر دو تصویر پشت سر هم ظاهر شوند. عکس gray scale با نام خاص خود و عکس رنگی نیز با نام مخصوص به خود ذخیره میگردد.

سوال1.2:

در این سوال نیز توپ را سعی میکنیم در تصویر پیدا کنیم و با استفاده از دستورrecball() مربع به دور آن کشیده و در ادامه نیز توپ را درمکان دیگری درج میکنیم

سوال2:

در این سوال ابتدا باید trackbar را میسازیم. سپس پس از محاسبه ابعاد تصویر باید ماتریس انتقال را محاسبه کنیم.در این جا چون زاویه ما با تغییر UI باید تغییر کند، مقدار آرگومان مربوط به زاویه را متغیر در نظر میگیریم. سپس تصویر چرخش یافته را نیز با استفاده از تابع cv2.wrapAffine() به دست می آوریم. در ادامه با استفاده از دستور cv2.hstack() و cv2.line() دو تصویر را ابتدا در یک تصویر میبریم و خط گفته شده در صورت سوال را به آن اعمال میکنیم. شاید در این قسمت چالش اصلی تعیین تابع مربوط به تغییرات نقطه پایانی باشد که تا حدی که قابل قبول بود سعی شد مطلوب به دست آورده شود. سپس نیز با استفاده از دستور imshow() تصویر را میکشیم.

سوال5:

در این سوال تصویر را از دو فیلتر مذکور میگذرانیم. اما در درجه اول باید بدانیم که این فیلتر ها چگونه عملکردی دارند. ایده اصلی فرسایش دقیقاً مانند فرسایش خاک است ، مرزهای شیء پیش زمینه را از بین می برد (همیشه سعی میکنیم پیش زمینه را به رنگ سفید نگه داریم). هسته از طریق تصویر می چرخد (همانگونه که در کانولوشن دو بعدی است). یک پیکسل در تصویر اصلی (یا 1 یا 0) فقط 1 در نظر گرفته خواهد شد اگر تمام پیکسل های زیر هسته 1 باشند ، در غیر این

صورت فرسایش یافته است (به صفر رسیده است). بنابراین اتفاقی که می افتد این است که تمام پیکسل های نزدیک به مرز بسته به اندازه هسته حذف می شوند. بنابراین ضخامت یا اندازه جسم پیش زمینه کاهش می یابد و یا به سادگی ناحیه سفید در تصویر کاهش می یابد. برای از بین بردن نویز های کوچک سفید و جدا کردن دو شیء متصل و غیره مفید است.

فیلتر delition دقیقاً مخالف فرسایش است. در اینجا یک عنصر پیکسل 1 است اگر حداقل یک پیکسل در زیر هسته 1 باشد. بنابراین باعث افزایش ناحیه سفید در تصویر می شود و یا اندازه جسم پیش زمینه افزایش می یابد. این جا شاید نویز های سفی از بین برود اما متعاقبا شیء ها نیز خراب میشوند.

Opening فقط نام دیگری از فرسایش است که به دنبال آن delition انجام میگردد.

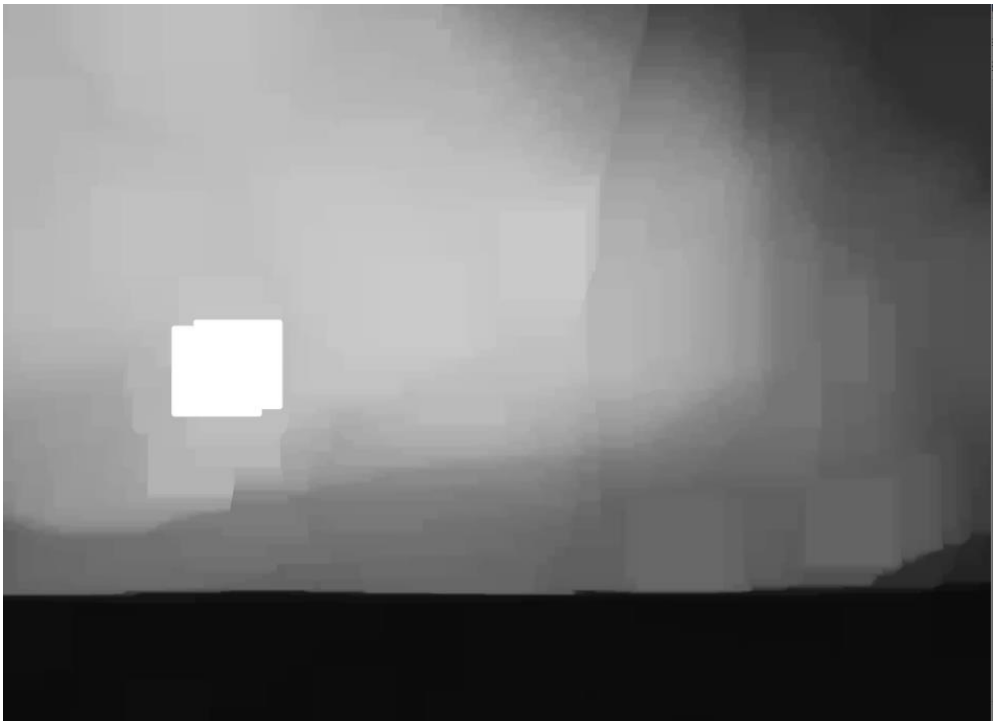
Closing معکوس opening است ، delition و بعد از آنerosion. در بستن سوراخ های کوچک در داخل اشیاء پیش زمینه یا نقاط سیاه کوچک روی جسم مفید است.

شکل های زیر با استفاده از کرنل با ماتریس 5*5 و تکرار 20 به دست آمده است.

عکس ورودی:

عکس با فیلتر erosion :

عکس به فیلتر delition :

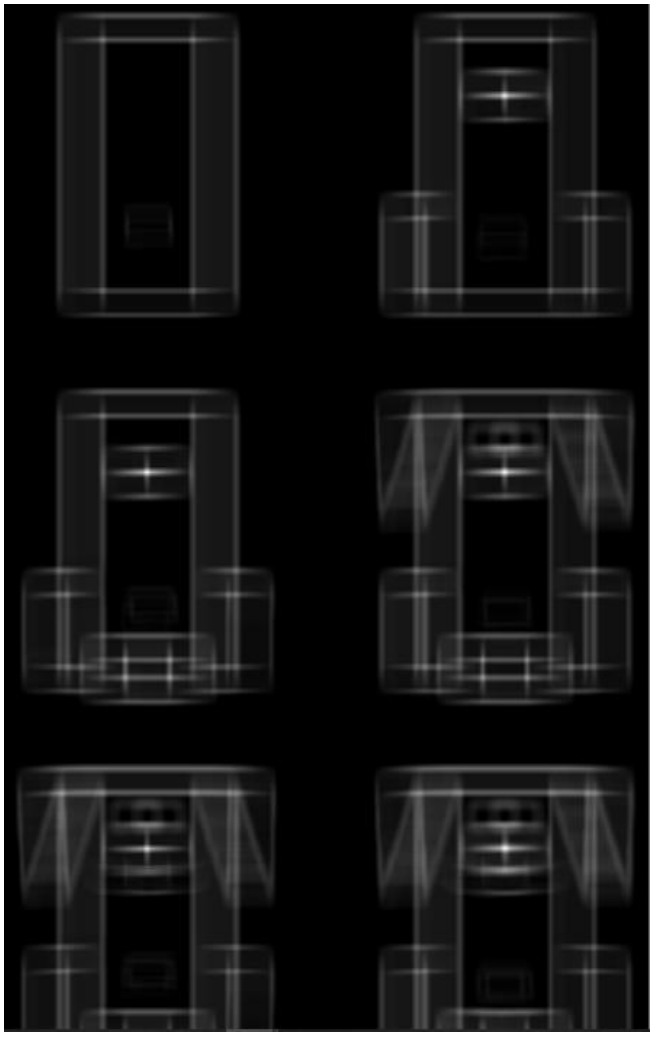


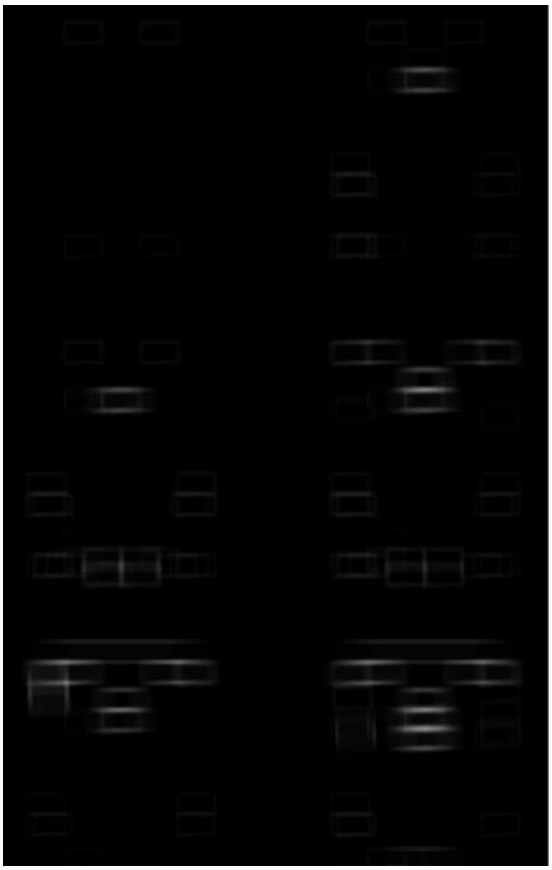
عکس با opening :

عکس با closing :

سوال 7:

در این سوال سعی شده با استفاده از کرنل مناسب فیلتر بر روی تصویر زده شود و بتوان مربع های مورد نظر را استخراج کرد. در اینجا با توجه به اینکه تصویر سگ انتهایی رنگی است بهتر است تمام تصویر را به حالت گری برده و در آن جا با استفاده از آستانه مناسب تصویر را به حالت های صرفا سیاه و سفید درآورد. ابتدا برای اولین بار که آستانه انتخاب میکنیم نقاطی را تصویر برای ما مشخص میکند. تصویر اول مشخص میکند که در ناحیه هایی که کرنل منطبق شده است تقریبا پیک های قابل توجهی رخ داده است. پس از اینکار ما باید دوباره برای تشخیص بهتر یک بار دیگه آستانه را بر روی تصویر فیلتر شده انجام میدهیم تا به صورت دقیق تری به محل نقاط مد نظر برسیم. تصاویر زیر مشخص کننده این امر میباشند.





سوال 1 قسمت ب:

در این سوال ابتدا ابتدا تصویر را میخوانیم و سپس در یک while True تصویر را از وبکم میخوانیم و در هنگامی که کلید e زده میشود عمل ذخیره سازی از بین میرود. البته متاسفانه گمان میکنم اندکی سوال غیر واضح بود زیرا نمیشود که وقتی کلید e بدون زدن کلید s قادر به سیو میباشد دیگر دلیلی ندارد که با زدن دکمه s بگوییم ضبط کند. بنابراین کدی که آورده شده به همین صورت تیپیکال میباشد.

سوال 2 قسمت ب:

فیلتر median برای نویز های مانند نمک و فلفل بسیار عالی است. الگوریتم فیلتر ، با استفاده از یک ماتریس کوچک (مانند 3*3)، کل تصویر را اسکن می کند و با استفاده از یک میانه از تمام مقادیر داخل ماتریس ، مقدار پیکسل مرکز را دوباره محاسبه می کند.

بیشتر اوقات می توانیم فرض کنیم هر پیکسل همان قطعه پس زمینه را می بیند زیرا دوربین در حال حرکت نیست. گاهی اوقات ، یک ماشین یا یک جسم متحرک دیگر در جلو قرار می گیرد و زمینه را مبهم می کند. برای یک دنباله ویدیویی ، می توانیم به طور تصادفی چند فریم را نمونه برداری کنیم (مثلاً 25 فریم). تا زمانی که پیکسل بیش از 50٪ از زمان توسط یک ماشین یا جسم متحرک دیگر پوشانده نشود ، میانه پیکسل بالای این 25 فریم تخمین خوبی از پس زمینه در آن پیکسل خواهد داد.

برای تعیین این اختلاف پیکسل نیز به صورت زیر عمل میکنیم.

1- قاب متوسط را به مقیاس خاکستری تبدیل میکنیم.
2- حلقه روی همه فریم های موجود در ویدیو. قاب فعلی را استخراج کرده و آن را به مقیاس خاکستری تبدیل میکنیم
3- اختلاف مطلق بین قاب فعلی و قاب متوسط را محاسبه میکنیم.
4- اختلاف مطلق بین قاب فعلی و قاب متوسط را محاسبه میکنیم.

به این صورت ما کد را به دست میزنیم.