بسم الله الرحمن الرحیم

درس بینایی ماشین

دکتر صفابخش

گزارش تمرین سری هفتم

موعد تحویل: ۱۳۹۸,۱۱,۱۶

دانشجو: حمیدرضا فهیمی

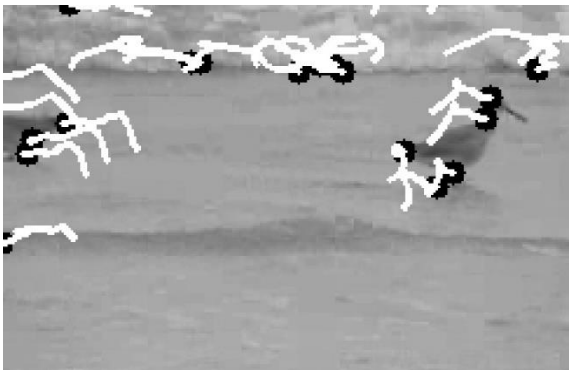
توضیح:

با توجه به این که در این تمرین با ویدیو کار می کنیم، خروجی الگوریتم ها به صورت فریم های نمونه برای هر یک از سوالات آورده شده است. کلیه برنامه های نوشته شده نیز در ضمیمه موجود است.

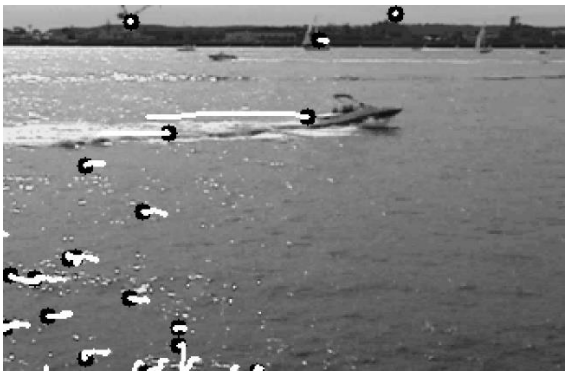
۱ – ۱:

برای به دست آوردن شار نوری با الگوریتم shi-thomasi از تابع calcOpticalFlowPyrLK استفاده می شود. این الگوریتم از یافت نقاط کلیدی و گوشه و تطبیق بین فریم های متوالی استفاده می کند. بنابراین نقطه هایی که به عنوان ورودی به تابع calcOpticalFlowPyrLK داده می شود، خروجی تابع goodFeaturesToTrack هستند. نتیجه اعمال این الگوریتم برای هر ویدیو به صورت زیر است.

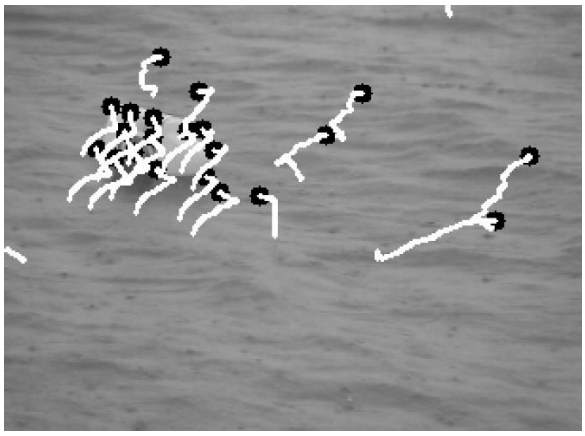
birds



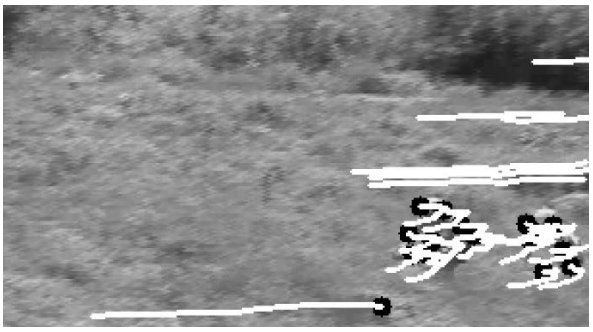
boats



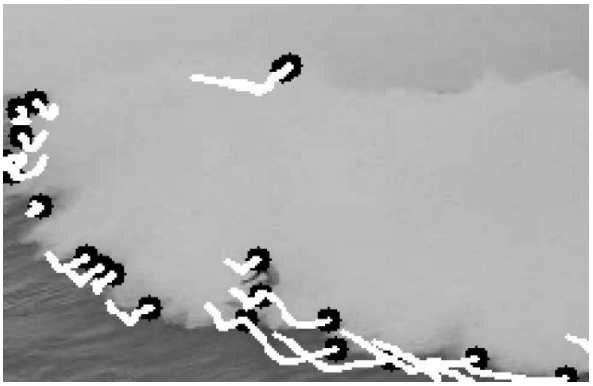
bottle



cyclists



surf



و اما شار نوری حاصله به وسیله الگوریتم gunnar farneback همچنین dense optical flow نامیده می شود. تابع محاسبه گر calcOpticalFlowFarneback می باشد. خروجی این تابع در برنامه نوشته شده به تابع draw_flow فرستاده شده و شار نوری به نحو زیر ترسیم می شود:

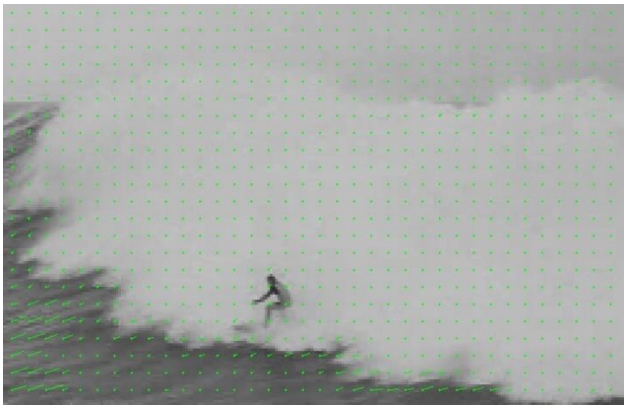
birds

boats

bottle

cyclists

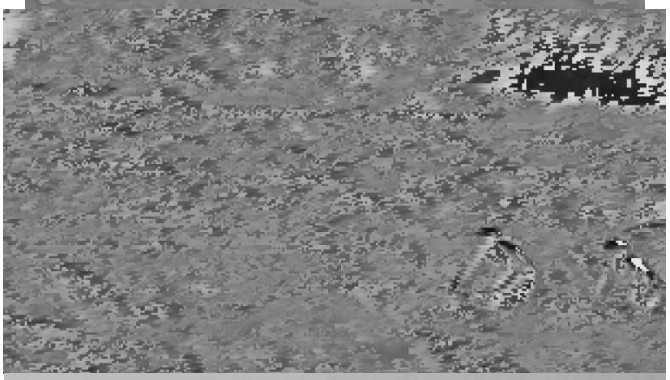
surf



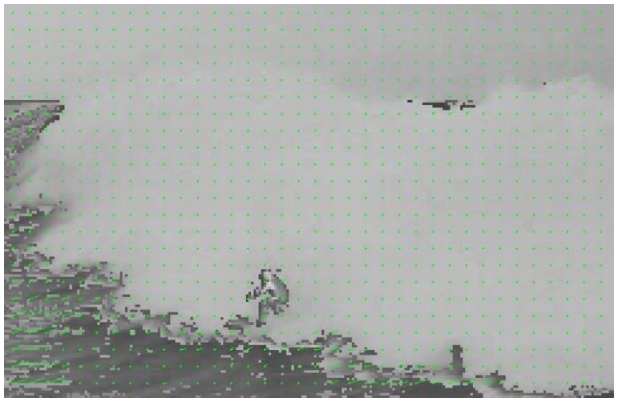
مقایسه: مشاهده می شود که روش اول از نقاط کلیدی استفاده کرده و آنها را دنبال می کند. اما روش دوم یک توری از نقاط روی تصویر قرار داده و تغییر جزئی از تصویر که در زیر هر نقطه قرار دارد را در بردار واقع در آن نقطه بازتاب می دهد. به این ترتیب روش اول برای تشخیص اشیاء متحرک و روش دوم برای زیر نظر داشتن کل صحنه مناسب است.

۱ – ۲:

از تابع `absdiff` برای حصول قدر مطلق تفاضل دو فریم متوالی استفاده کرده و تحرک را در هر قدم بین فریم ها به دست آوردیم. سپس حاصل را با فراخوانی دوباره `absdiff` از فریم کم کرده و مطابق خواسته ی سوال، خروجی ها به صورت زیر به دست آمد. برای `shi-thomasi`:

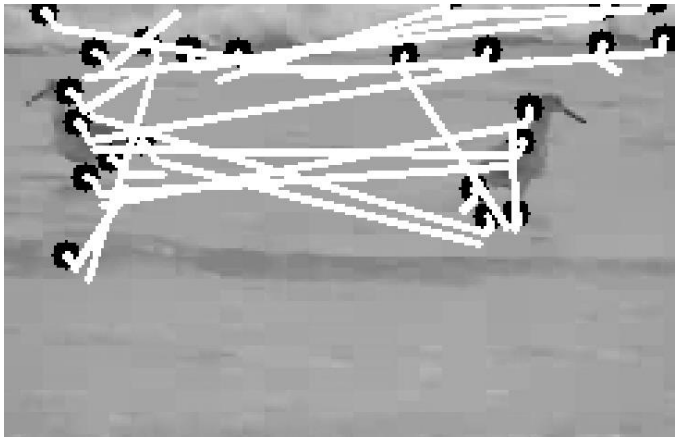


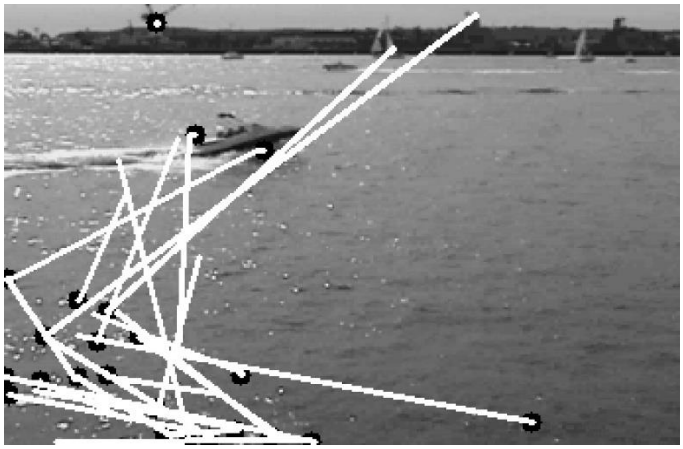
برای farneback:

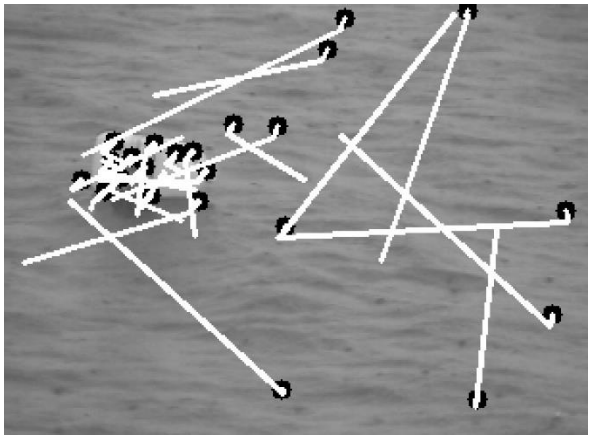


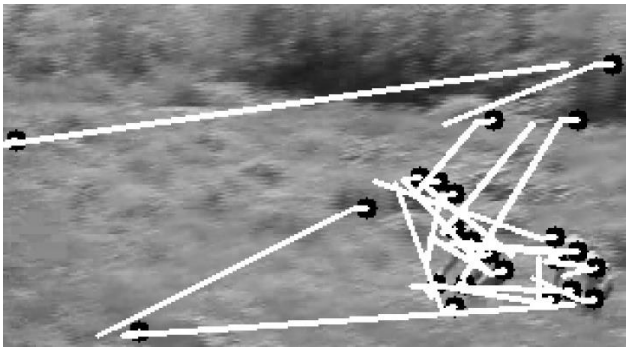
۱ – ۳:

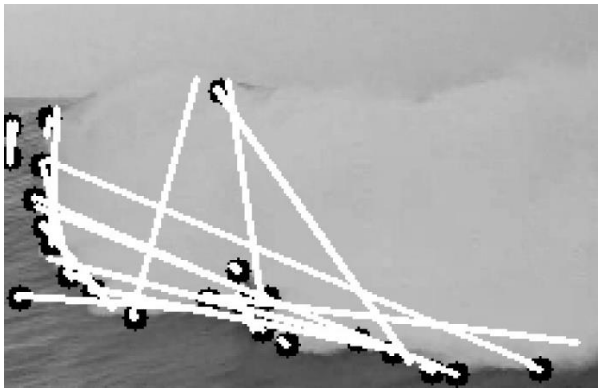
پرچم های خواسته شده ورودی های تابع calcOpticalFlowPyrLK هستند. پرچم OPTFLOW_USE_INITIAL_FLOW از تخمین اولیه استفاده می کند. خطوط بلند رسم شده در ادامه به دلیل همین تخمین اولیه بین فریم اول و دوم در ابتدای پردازش ایجاد شده اند:



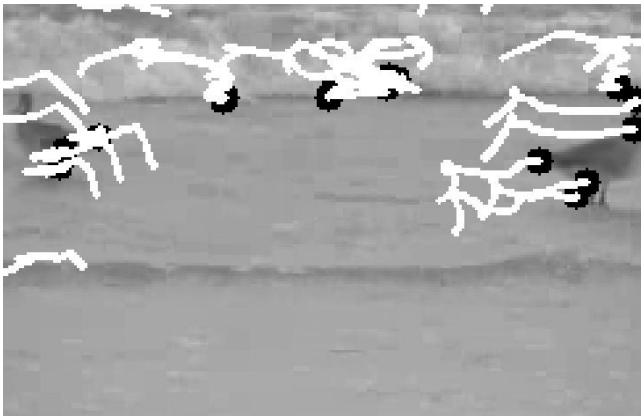


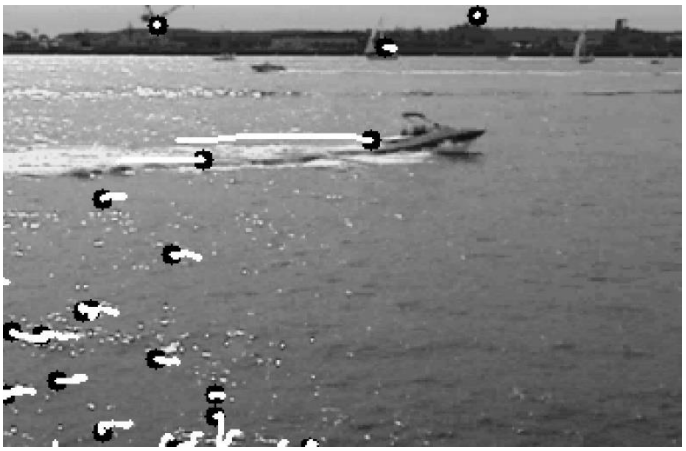


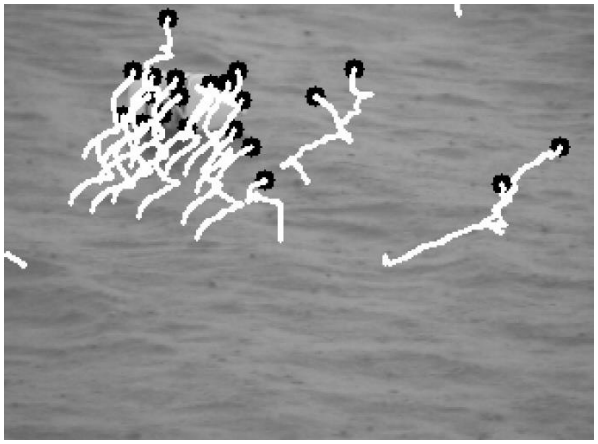


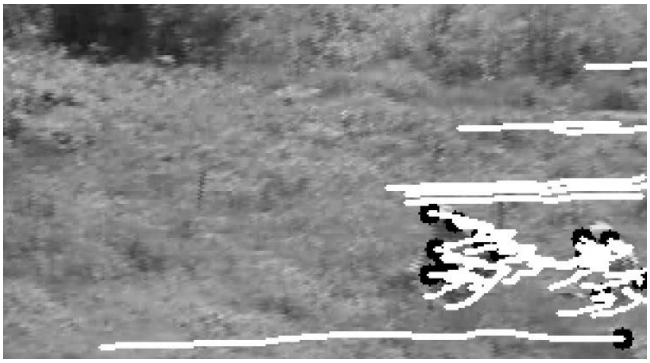


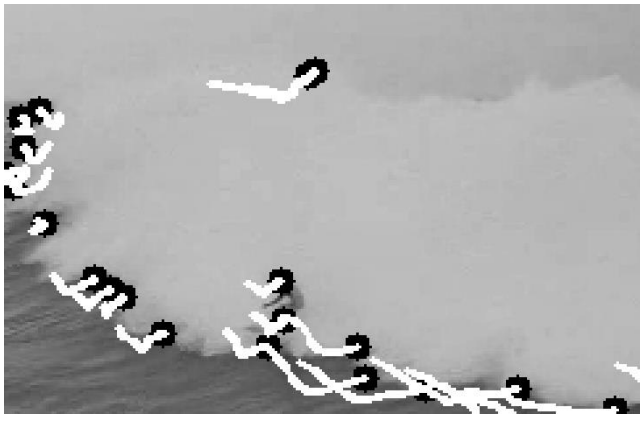
خروجی به ازای پرچم OPTFLOW_LK_GET_MIN_EIGENVALS مناسب تر است. در زیر مشاهده می شود:







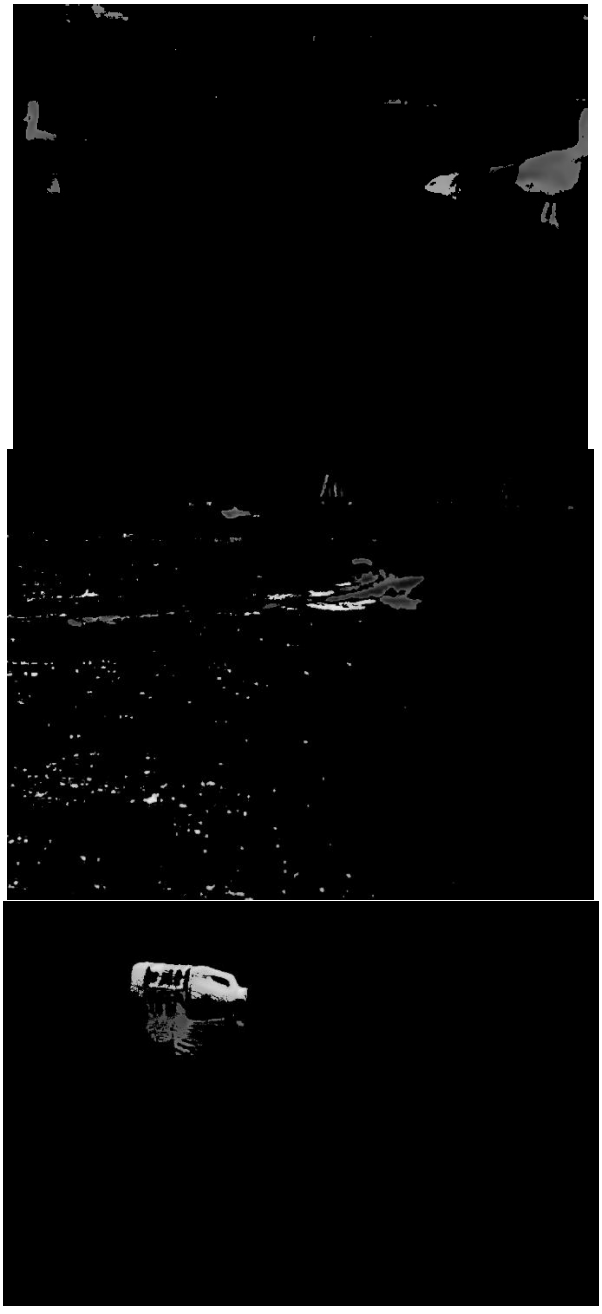




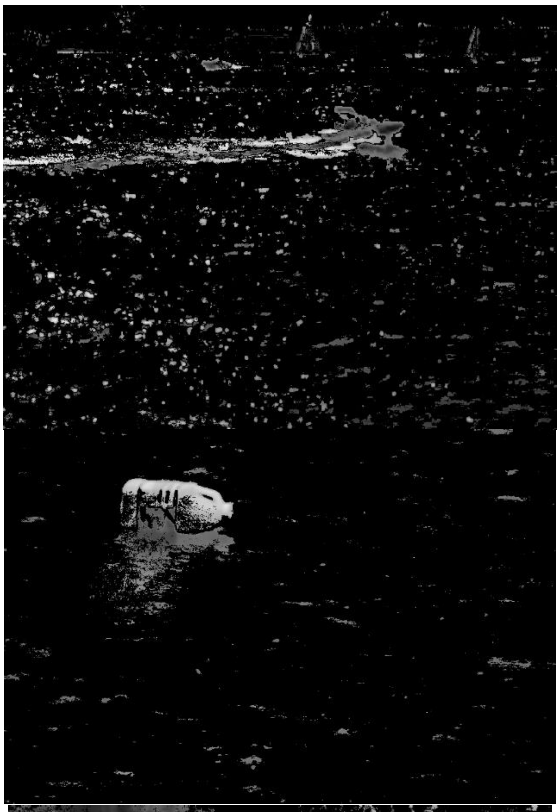
۲:

برای اعمال الگوریتم های خواسته شده از کلاس `BackgroundSubtractor` استفاده می شود. سه برنامه مجزا برای سه الگوریتم نوشته شده است. در هر برنامه، یک شیء (bg_model) از کلاس `BackgroundSubtractor` ساخته شده و توسط آن تابع apply فراخوانی می شود. به این ترتیب اشیاء متحرک در متغیر fgmask ریخته می شوند. خروجی از قرار زیر است.

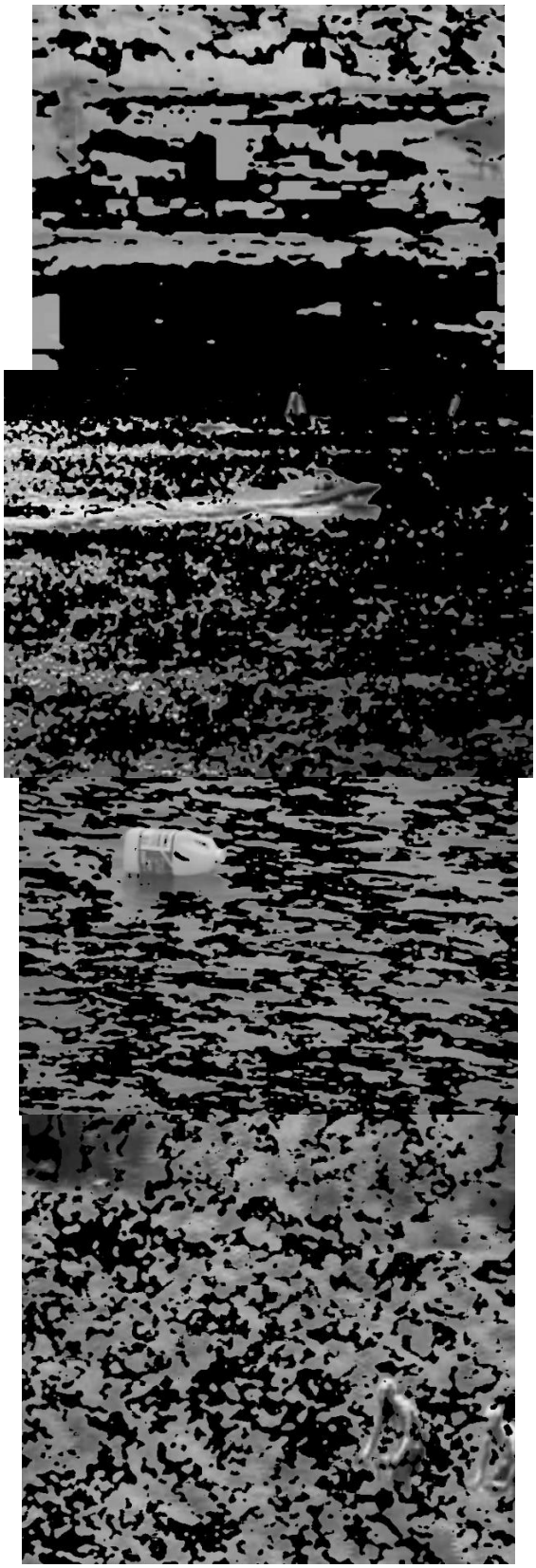
خروجی الگوریتم BackgroundSubtractorMOG:

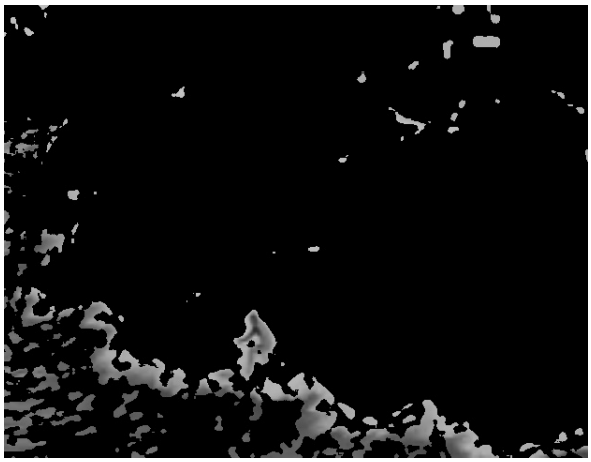


خروجی الگوریتم BackgroundSubtractorMOG2:



خروجی الگوریتم BackgroundSubtractorGMG:





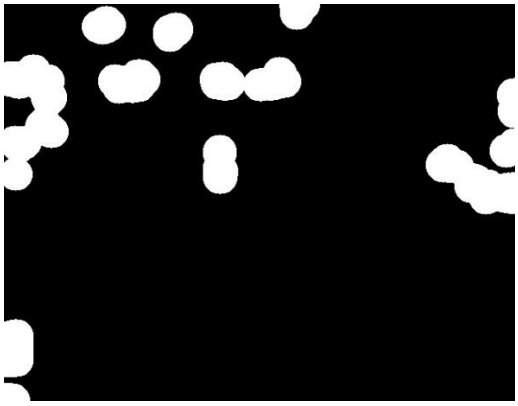
مقایسه: روش BackgroundSubtractorMOG در موارد حاضر نتیجه ی بهتری داده است. خروجی این روش دقیق و بدون اضافات است؛ در حقیقت با وجود اشیاء متحرک دیگر در صحنه، این روش تنها شیء دارای بیشترین تحرک را شناسایی می کند. روش BackgroundSubtractorMOG2 همه اشیاء متحرک را می شناسد. به طور کلی تشخیص غلطی ندارد. اما در صحنه هایی که حرکت در آنها زیاد بوده و تنها یک حرکت اصلی مدنظر است ممکن است خوب نباشد. در پیاده سازی روش BackgroundSubtractorGMG مشاهده شد که تا ۲۰ فریم اول ویدیو، هیچ خروجی وجود ندارد. از فریم ۲۱ به بعد این روش یک خروجی افزایش یابنده به طور پیوسته را می دهد و از جایی به بعد ماسک پیش زمینه کاملا سفید می شود. به نظر می رسد که برای موارد فعلی الگوریتم BackgroundSubtractorGMG مناسب نیست.

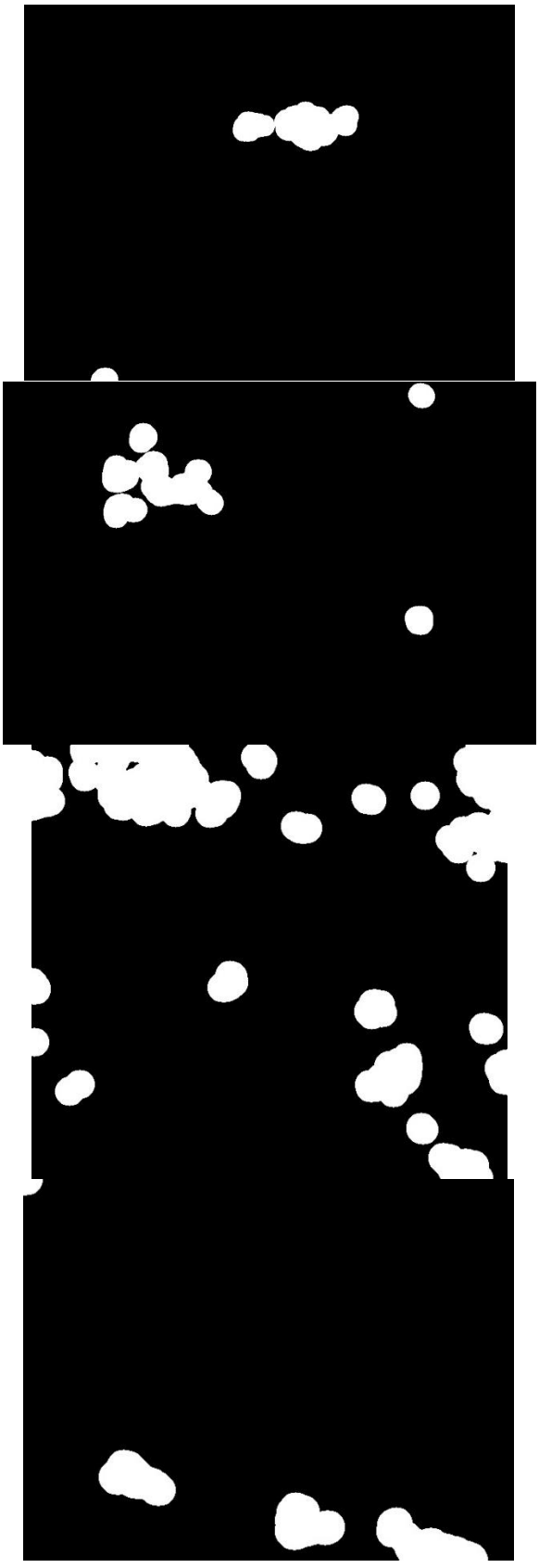
۲ – ۱:

برای بررسی تاثیر تغییر اندازه کرنل در الگوریتم BackgroundSubtractorGMG، کرنل را با تابع `getStructuringElement` مقداردهی نموده و از آن در حلقه ی پردازش ویدیو به عنوان ورودی تابع `morphologyEx` استفاده می کنیم. جهت بررسی تاثیر تغییرات اندازه، یک بار اندازه کرنل ۳ و یک بار ۲۰ در نظر گرفته شده است. همان طور که بیان شد، هر سه الگوریتم های استخراج پیش زمینه یک ماسک که شامل اشیاء متحرک است را به عنوان خروجی می دهند. در ادامه ماسک پیش زمینه مورد مقایسه قرار گرفته است.

نتایج الگوریتم به ازای اندازه کرنل ۳:

نتایج الگوریتم به ازای اندازه کرنل ۲۰:



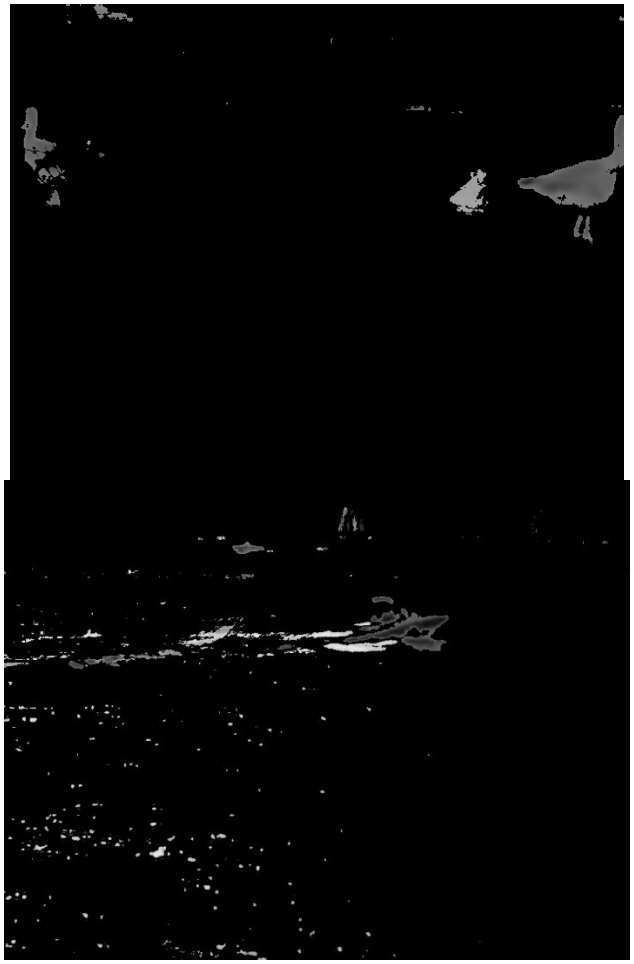


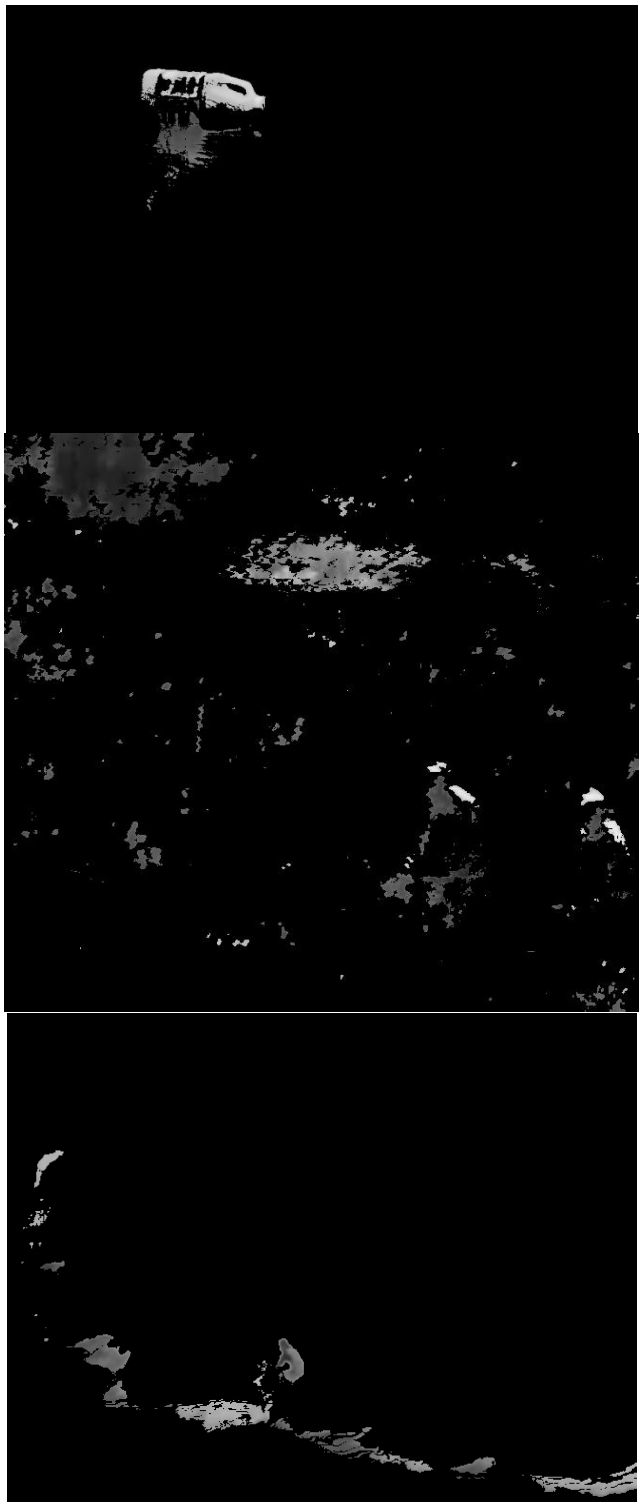
بررسی: مشاهده می شود که افزایش اندازه کرنل موجب حذف جزئیات در ماسک پیش زمینه می شود. اما از طرفی می تواند در شرایطی که تنها به تقریبی درست از ناحیه ای که در آن تحرک وجود دارد نیاز داریم مفید باشد. افزایش اندازه کرنل به علاوه موجود حذف نقاط با تحرک کمتر و نقاط بدون تحرک نسبت به اندازه کرنل ۳ شده است.

۲ – ۲:

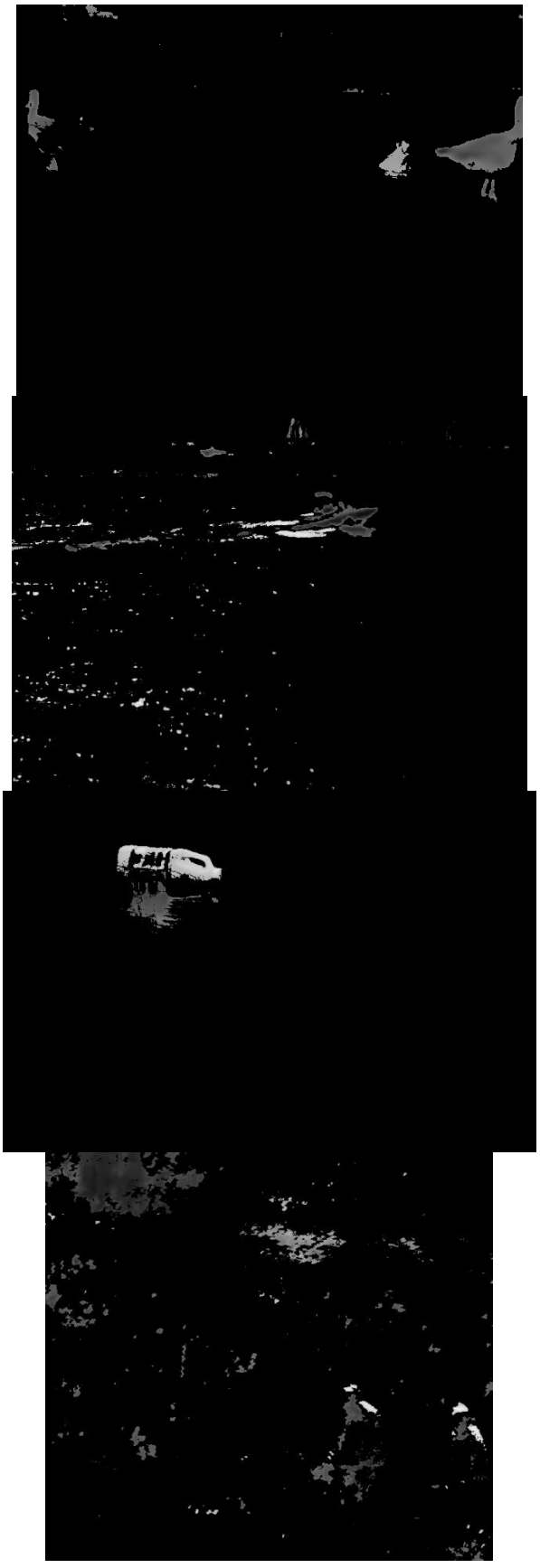
تعداد مولفه های گاوسی (nmixtures) در الگوریتم BackgroundSubtractorMOG یکی از ورودی های اولیه در هنگام ساخت شیء مدل پس زمینه از کلاس `BackgroundSubtractor` است. در اینجا سه مقدار ۲، ۳ و ۵ به این پارامتر اختصاص یافته است. در ادامه خروجی مشاهده می شود. لازم به ذکر است مقدار ۵ مقدار پیش فرض برای این الگوریتم می باشد.

نتایج الگوریتم BackgroundSubtractorMOG به ازای تعداد مولفه های گاوسی ۲:

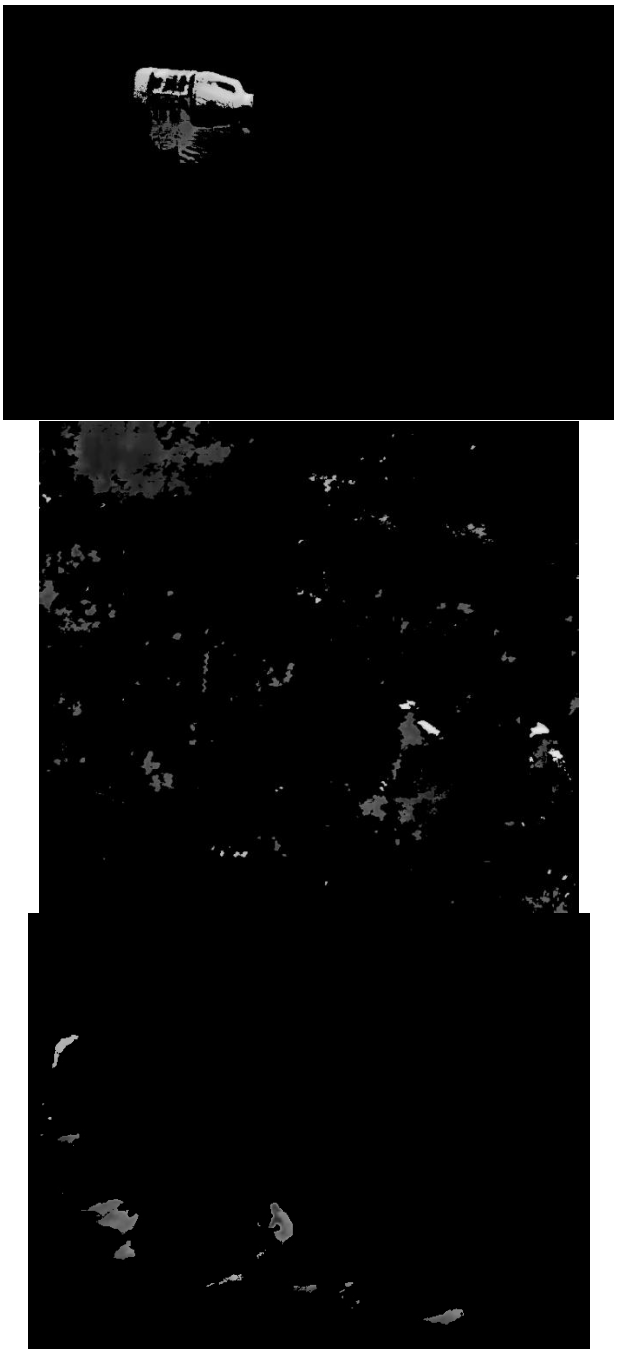




نتایج الگوریتم BackgroundSubtractorMOG به ازای تعداد مولفه های گاوسی ۳:



نتایج الگوریتم BackgroundSubtractorMOG به ازای تعداد مولفه های گاوسی ۵:



بررسی: تغییرات جزئی و به نحوی است که با یک نگاه نمی توان به سادگی نتیجه گیری نمود. پس از بررسی مشاهده شد که در اکثر موارد، افزایش n منجر به کاهش اجزاء متحرک جزئی و تاکید بیشتر بر اشیاء اصلی شده است. در واقع n را می توان یکی از فاکتورهای اصلی موثر بر تفاوت BackgroundSubtractorMOG با BackgroundSubtractorMOG2 در تشخیص تحرک منحصر به اشیاء دارای تحرک کلی تر دانست.

۲ – ۳:

روندی مشابه قسمت ۱-۲ در اینجا نیز تکرار شده است. در هر مورد پیش زمینه ی به دست آمده از الگوریتم ها از پس زمینه کم شده و خروجی ها به صورت زیر است.

پس زمینه حاصل از الگوریتم BackgroundSubtractorMOG:

پس زمینه حاصل از الگوریتم BackgroundSubtractorMOG2:

پس زمینه حاصل از الگوریتم BackgroundSubtractorGMG: