پروژه درس دید کامپیوتری – بهار ۱۳۹۹

استاد : دکتر هدی محمدزاده

دانشجو : مهرسا پوریا

دانشکده مهندسی برق - دانشگاه صنعتی شریف

برای انجام خلاصه سازی مقالات مختلفی جست و جو شد اما هیچ یک به دقت مراحل کار خود را به صورت دقیق شرح نداده بودند، برای مثال در یکی از مقالات ذکر شده بود که از آنالیز Connect component سه بعدی برای یافتن تیوب‌های مکانی زمانی استفاده میشود، اما بعد از دانلود کتابخانه‌ای برای این کار (cc3d) متوجه شدم که زمان اجرای آن بسیار زیاد است و مناسب نبود. در نهایت روش زیر برای خلاصه در نظر گرفته شد که مرحله به مرحله توضیح داده شده است. هر قسمت نتیجه تعداد مختلفی تست است که این تلاش ها نیز با عنوان **نکته** شرح داده شده اند.

۱. **تخمین پس‌زمینه:** با فرضی که در صورت پروژه آمده‌است میدانیم که پس‌زمینه ثابت است برای پیدا کردن اشیا متحرک میتوان از این فرض استفاده کرد و کار راحت تر میشود، بنابراین قدم اول یافتن پس‌زمینه است. برای اینکار مشابه تمرین اول از یک فیلتر میانه روی ۲۰۰ فریم اول (بعد زمان) هرکدام از ویدیوها استفاده می‌کنیم که تصاویرزیر به ترتیب از راست به چپ پس‌زمینه استخراج شده برای ویدیو اول و دوم داده شده‌است.
نکته: در ابتدا من میخواستم اشیا متحرک را با تفاضل فریم‌ها به دست آورم اما اینکار خطای زیادی داشتر، سپس با خواندن مقالات به این نتیجه رسیدم تفاضل از پس‌زمینه بهتر است و به این مرحله تخمین پس‌زمینه حتما نیاز است.

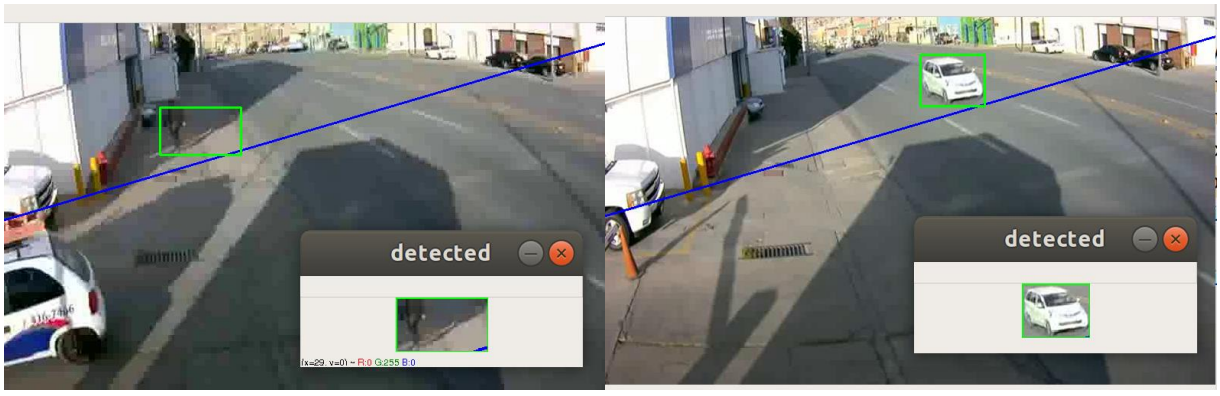
۲. **پیدا کردن اشیا متحرک:** در این مرحله مراحل زیر طی میشود:

- در هر کدام از ویدیوها با توجه به زاویه دوربین خطی در نظر گرفته میشود، در دو ویدیو داده شده خطهای در نظر گرفته شده به صورت زیر هستند:

برای یافتن خط مناسب آزمایشهای زیادی در هر ویدیو انجام شد و خطهای بالا به صورتی که در ادامه شرح داده میشوند مورد استفاده قرار گرفته و کمترین خطا را دارند، برای هر دوربین این مرحله باید Tune شود.

- سپس هر فریم (Frame) را از پس‌زمینه یافته‌شده در بخش قبل را منها میکنیم، به این امید که صرفا مکانهایی که دارای جسم متحرک هستند مقدار غیر صفر به خود بگیرند، نتیجه به تصویر gray تبدیل شده و آن تصویر gray با یافتن ترشولد مناسب که برای هر ویدیو با آزمایش محاسبه شد، به تصویر binary تبدیل میشود، در صورت لزوم تبدیل‌های مورفولوژی مانند dilate کردن یا بلوری کردن نیز انجام میشود که تصویر binary مناسبی به دست آید. پارامترهای موجود در کد برای ویدیوها با آزمایش بسیار در حالت بهینه خود هستند، برای باینری کردن میشود از روشهای اتوماتیتکی مانند otsu هم استفاده کرد اما پارامترهای فعلی کد خروجی robust تر و صحیح تری داشتند.
- سپس در تصویر باینری حاصل مرحله قبل contour ها پیدا میشوند، برای هر contour، bounding box مربوط به آن کانتور نیز محاسبه میشود، در صورتی که مساحت bounding box کمتر از حد خاصی باشد آن حذف میشود زیرا با احتمال بالا مربوط به نویزهای پس‌زمینه است.
- اگر در فریم فعلی بعد از انجام مراحل بالا bounding box ای یافت شد که فاصله آن تا خط تشخیص در بازه خاصی که برای هر ویدیو tune شده باشد، آن bounding box به عنوان شی متحرک کشف شده نگه داشته میشود.

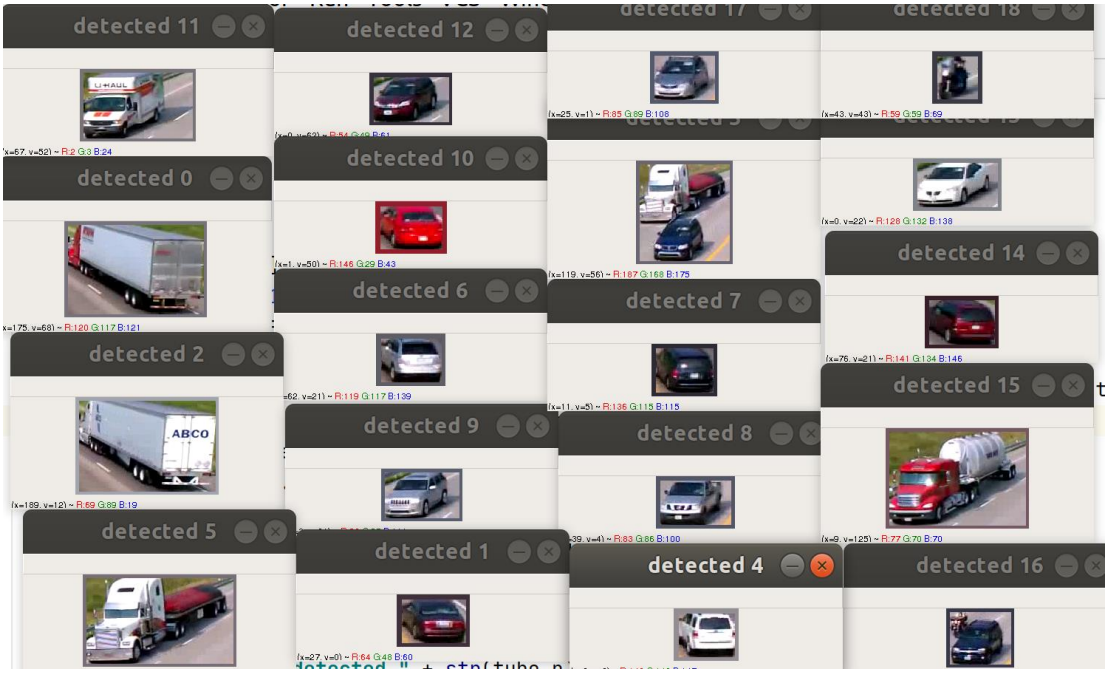
برای فهم بهتر روش گفته شده به شکلهای زیر که تشخیص چند شی متحرک است دقت کنید:



نکته: با گذاشتن شرط محدود روی بازه قابل قبول، و همچنین شروط زمانی و مکانی به این صورت که در مختصاتی نزدیک به هم در زمانی نزدیک بهم دو شی اگر تشخیص داده شد اولی را قبول نکن، میتوان خطای تشخیص دوگانه یک شی را در این روش به خوبی کنترل کرد و همچنین تعداد miss های آن نزدیک به صفر است. در ابتدا من میخواستم که هرگاه کانتوری تشخیص داده شد با چک فریم‌های قبلی بفهم که آیا شی تازه است یاخیر، اما مشکلی که هست این است که این کار برای ابجکتهایی که ورود آنها دور از دوربین هستند خطای زیادی به وجود می آورد و tracker در تعقیب آنها ناموفق میشود، اما در تشخیص به کمک این خط تشخیص این مشکلات پیش نمی آید و از لحاظ زمانی نیز سریعتر است. هرچند به ظاهر این روش براساس bounding box است، اما طریقه یافتن آنها توسط contour است و bounding box پیدا شده با کانتور هر جسم تناظر یک به یک دارد و کار را در قسمت tracker نیز ساده میکند.

در پایان مرحله ۲ ما مختصات و زمان (شماره فریم) مربوط به هر object متحرک در هنگامی خط تشخیص را قطع میکند داریم. (خوشبختانه خط ها به گونه ای تعیین شده اند که تمامی حرکات مهم ویدیو ها را پوشش دهند.) در مرحله بعد از این اطلاعات تلاش میکنیم tube زمانی مکانی هر شی متمایز را بسازیم.

برای مثال در پایان مرحله ۲ بعضی آبجکت های تشخیص داده شده در ویدیو ۱ به صورت زیر هستند.



(اگر دقت شود در تصویر بالا در اطراف هر شی مستطیلی رنگی وجود دارد که موید مرحله تشخیص رنگ که در ادامه نیز شرح میدهیم است.)

۳. **یافتن tube به کمک tracking:** ما اشیا متمایز را در مرحله بالا در یک نقطه از زمان (میتوان گفت نزدیک ترین حالت به خط تشخیص) یافته ایم. اشیا در طول زمان حضور خود در ویدیو دو باز زمانی دارند، یکی قبل از آن نقطه زمانی و یکی بعدش، با همین استدلال ما باید هم در ادامه مسیر از نظر زمانی (بعدی) آنها را track کنیم و هم در فریم های قبلی (متناظر track کردن در معکوس ویدیو). این دقیقا کاری است که من انجام داده ام، توسط tracker از نوع medianFlow هم در ادامه و هم در گذشته اشیا یافته شده را جست و جو کردم، هر مرحله track کردن یک مختصات برای مستطیل حاوی شی میدهد که آن را list پایتون ذخیره میکنم و با استفاده از خواص ساده این ساختمان داده، به ترتیبی نتایج را در کنار هم قرار میدهم که tube **منظم** زمانی مکانی از هر شی ساخته شود. شرط توقف track کردن نیز ثابت ماندن تقریبی پنجره حاوی شی در جای خودش است که متناظر ورود و یا خروجی شی از ویدیو است. بدیهی است که زمان ورود شی به ویدیو متناظر شماره اخرین فریم سرچ در ویدیو معکوس است که این زمان نیز تحت عنوان start_frame ذخیره میشود، مختصات سازنده tube نیز در لیستی به نام tube ذخیره شده اند. همچنین مختصات اولین حضور شی در ویدیو نیز در start_pos ذخیره شده است. این سه اطلاعات tube زمانی مکانی، زمان و مکان ورود هر

شی متمایز را تعیین میکنند که در لیستی به نام masks به همراه اطلاعات قسمت امتیازی که در ادامه توضیح داده میشوند ذخیره میشوند و مرحله بعدی کنار هم قرار دادن این اطلاعات به منظور ساخت ویدیو خلاصه شده است، برای ویدیو اول ۴۴ tube که متناظر ۴۴ شی است و برای دومی ۲۷ تیوب مختلف استخراج میشود.

نکته: opencv حدود ۷ نوع tracker مختلف قابل استفاده دارد که همه ی آنها تست شد، بهترین نتایج با medianFlow کسب شد، یک نکته مثبت این tracker این است که تغییرسایز شی را نیز لحاظ میکند که این نکته بسیار مهم است زیرا در ویدیوهای داده شده با توجه به پرسپکتیو اشیا با حرکت خود تغییر سایز میدهند. و tracker هایی مانند kcf این نکته را لحاظ نمیکنند و بنابراین تداخل زیادی بین تیوبهای اشیا مختلف به وجود می آید. با medianFlow این مشکل تا حد خوبی حل میشود. (مراحل تا این مرحله نسبتا خوب اجرا میشوند و به نظرم اگر بدی در کیفیت خروجی باشد دلیل عمده آن مرحله بعدی ۴ است.)

۴. **کنار هم قرار دادن tube ها و ساخت ویدیو نهایی:** یک متغیر به نام tubes تعریف شده است که تمامی frame های برابر پس‌زمینه ویدیو است، اولین tube از آغاز tubes داخل مکانهای مربوط به خودش قرار میگیرد. برای tube های بعدی فقط هنگامی اجازه ورود به tube داده میشود که ۱۶ درصد طول فریم های جلو مربوط به حضور آن کاملا خالی باشد. اینکار کمی با پشت سرهم گذاشتن فریم ها به صورت بدون تداخل تفاوت دارد و ویدیو dense تری ایجاد میکند. (البته عبور اشیا با سرعتای متفاوت از هم در آن اجتناب ناپذیر است.) هنگامی که دیگر تغییری برای اعمال باقی نماند (فقط پس زمینه نشان داده شود) انتهای ویدیو خلاصه شده است.

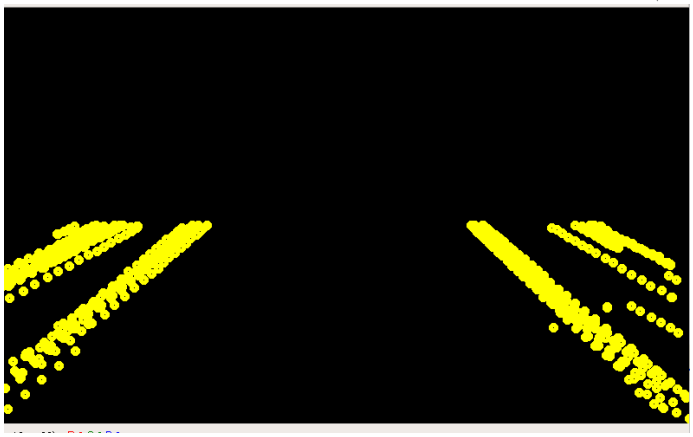
۵. **نوشتن زمانهای متناظر:** ما همانظور که گفته شد شماره اولین فریمی که شی در آن ظاهر میشود را استخراج کرده ایم، که برای رسید به ثانیه ورود شی در ویدیو اصلی کافی است آن را بر fps یا frame per second ویدیو تقسیم کنیم. هر افزایش فریم نیز به صورت خطی 1/fps ثانیه اضافه میکند که باتوجه به این مطالب زمان حضور به راحتی محاسبه میشود و روی شی با رنگ سبز نمایش داده میشود. Fps نیز برای ویدیو ها برابر 30 است.

ویدیوهای مراحل گفته شده با نام **output1.avi** و **output2.avi** ذخیره شده اند، لطفا از vlc player برای نمایش آنها استفاده کنید، در صورت اجرای کد نیز خروجی برای ویدیو اول پس از اجرای الگوریتم نمایش داده میشود و برای نمایش خروجی دومی video_num را به ۲ تغییر دهید.

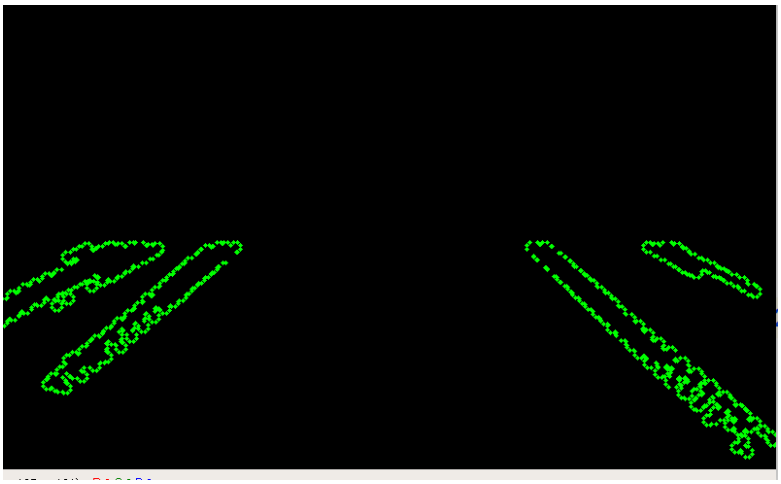
قسمت‌های امتیازی: قسمتهای زیر همگی پیاده شده اند و در خطهای ابتدایی کد میتوانید تعیین کنید در خروجی میخواهید خروجی آنها را ببینید یا خیر، در کامنت کد این قسمتها مشخص شده‌اند.

- **مسیر حرکت و Gradient Line:**

برای این قسمت همه centriod های همه ی مسیرهای یافته شده را رسم میکنیم، در هر دو تصویر چون خطوط با توجه به پرسپکتیو در انتها به هم میرسند قسمت های عقب تصویر را حذف میکنیم و به شکلی مثلا مانند زیر برای ویدیو یک میرسیم.

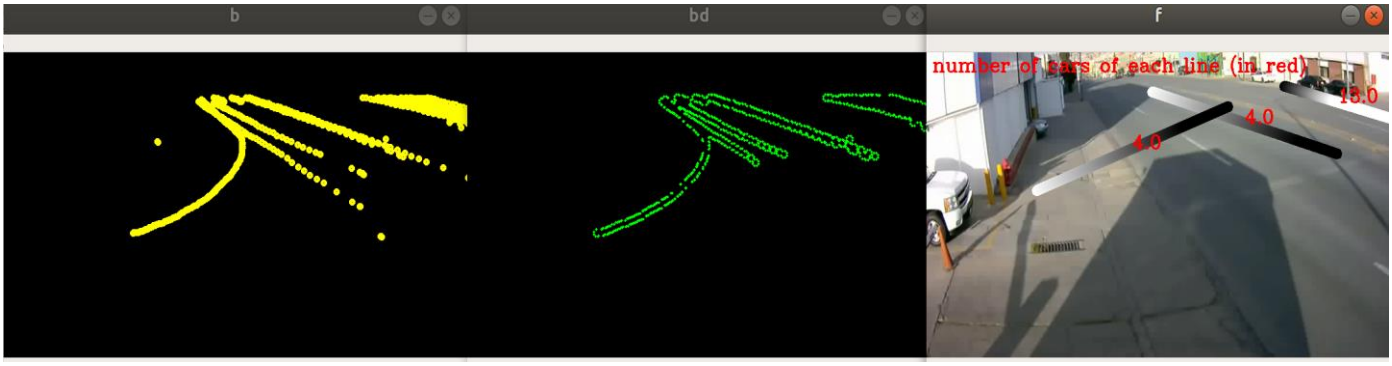


حال تصویر را باینری میکنیم و در صورت لزوم dilate میکنیم و سپس contour ها را پیدا میکنیم. در این حالت مثلا برای ویدیو اول ۴ کانتور پیدا میشود. (مشابه شکل زیر)

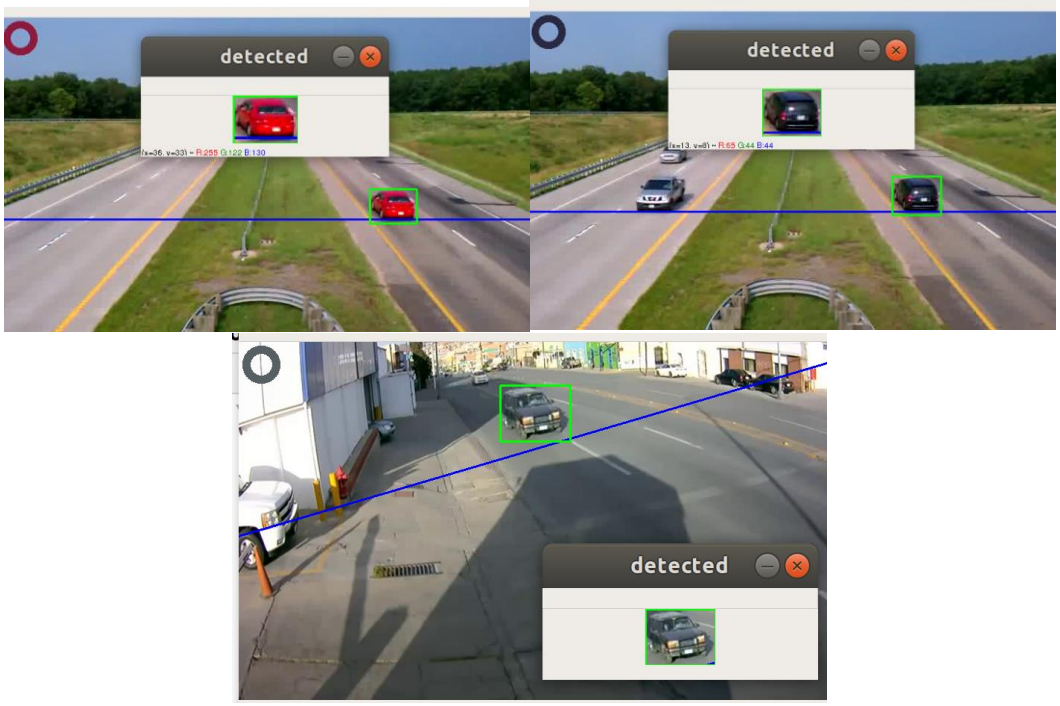


برای هر کانتور خط وصل کننده دو ظلع کوچکتر مستطیل محاط آن را پیدا میکنیم و به عنوان خط مسیر در نظر میگیریم برای مثال p1 و p2 دو سر این خط هستند، برای هر مسیر centroid چک میکنیم که آیا با ترتیب زمانی معلوم جابه جایی هر مسیر حرکت در جهت p1 به p2 بوده است یا نه! اینکار به سادگی با محاسبه فاصله نقاط یک مسیر با p1 و بررسی صعودی یا نزولی بودن آن امکانپذیر است که دقیقا همینکار در کد انجام شده است. سپس ما جهت حرکت و مسیر خطی آن را میدانیم و فقط مرحله رسم باقی میماند که توسط استفاده از نقاط و تغییر خطی رنگ عملا آن را نقاشی میکنیم. برای تصویر اول خروجی به شکل زیر میشود، اعداد قرمز در ادامه توضیح داده میشوند و فقط به خط ها توجه کنید:

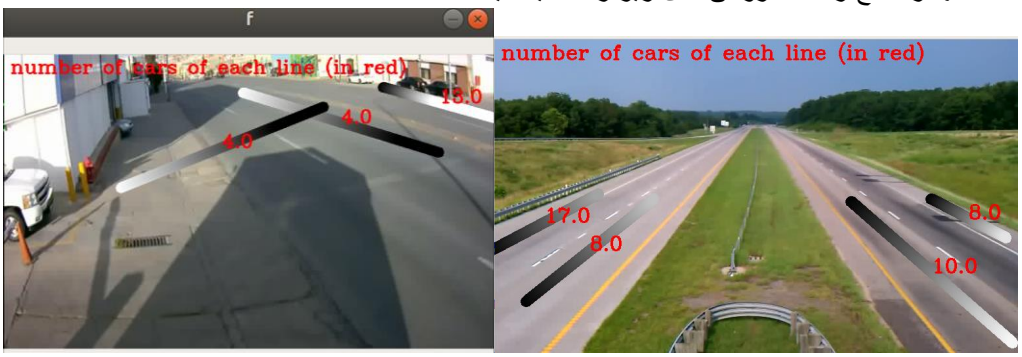
برای ویدیو دوم نیز به صورت زیر است:



- **تشخیص رنگ :** این مرحله با توجه به مراحل قبل ساده است، در پنجره تشخیص هر شی، ابتدا تصویر را از پس زمینه منها میکنیم تا اثر رنگهای پس زمینه را به گونه ای حذف کنیم، سپس آن تفاضل را باینری میکنیم و میانگین هر کانال رنگی پنجره تشخیص را در جاهایی که تصویر باینری شده صفر نیست (متناظر پیش زمینه) حساب میکنیم و عملا rgb رنگ مربوط به شی را محاسبه میکنیم. در حالت های زیر برای تست در گوشه بالای چپ پنجره اصلی دایره ای با رنگ تشخیص داده شده رسم شده است، در هنگام اجرا نیز اگر color_box را true کنیم اطراف هر شی در خروجی پنجره همگرنگ آن رسم میشود.



- **شمارش تعداد هم مسیرها:** این کار با توجه به اینکه یافتن مسیر حرکت انجام شده کار سختی نیست، در هر مسیر حرکت کافی است تعیین کنیم چه تعداد مسیر centroid ای در contour مربوط به یک جهت میافتند و تعداد آنها را جمع زد که خروجی های زیر را تشکیل میدهد:



- **تخیمن سرعت:** برای هر شی بردار حاوی جابه جایی پنجره تشخیص حاوی آن شی محاسبه میشود (جمع جابه جایی دو سر پنجره) و سپس از این بردار میانگین گرفته میشود. (یعنی جمع جابه جایی تقسیم بر تعداد فریم حضور) این معیار کاملا با مفهوم سرعت متوسط همخوانی دارد و از آن استفاده میشود اگر در کد مقدار show_speed، True شود این مقدار زیر مقدار زمان حضور در ویدیو خروجی نشان داده میشود. عدد مربوط به زمان حضور با t و عدد مربوط به سرعت با s نشان داده میشود.

مثلا در خروجی زیر: