



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

تمرین سری **هفت**

درس بینایی ماشین

فرهاد دلیرانی

۹۶۱۳۱۱۲۵

dalirani@aut.ac.ir

dalirani.1373@gmail.com

فهرست

ابزارهای استفاده شده ۱

تمرین ۱-۱-۱ ۲

تمرین ۱-۱-۲ ۶

تمرین ۱-۱-۳ ۹

تمرین ۱-۱-۴ ۱۱

تمرین ۱-۲ ۱۳

ابزارهای استفاده شده

زبان برنامه نویسی: پایتون ۳,۶

محیط توسعه: PyCharm

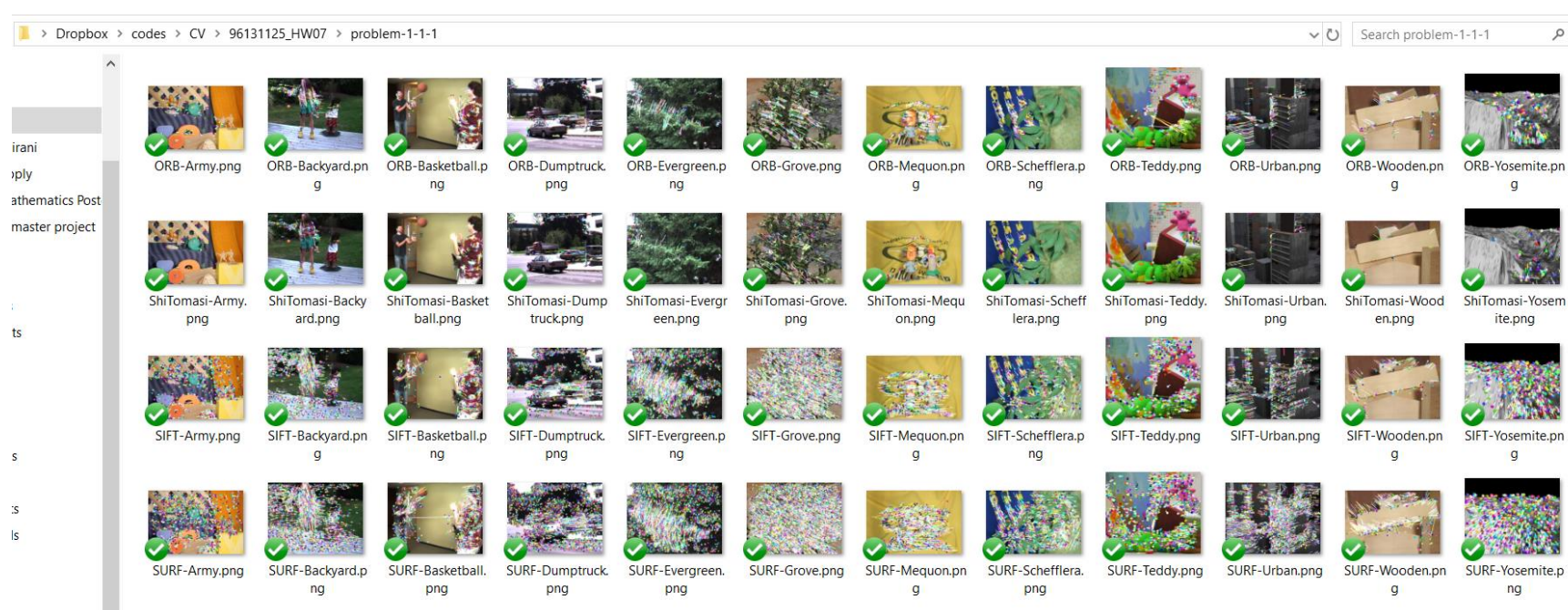
سیستم عامل: ویندوز ۱۰

تمرین ۱-۱-۱

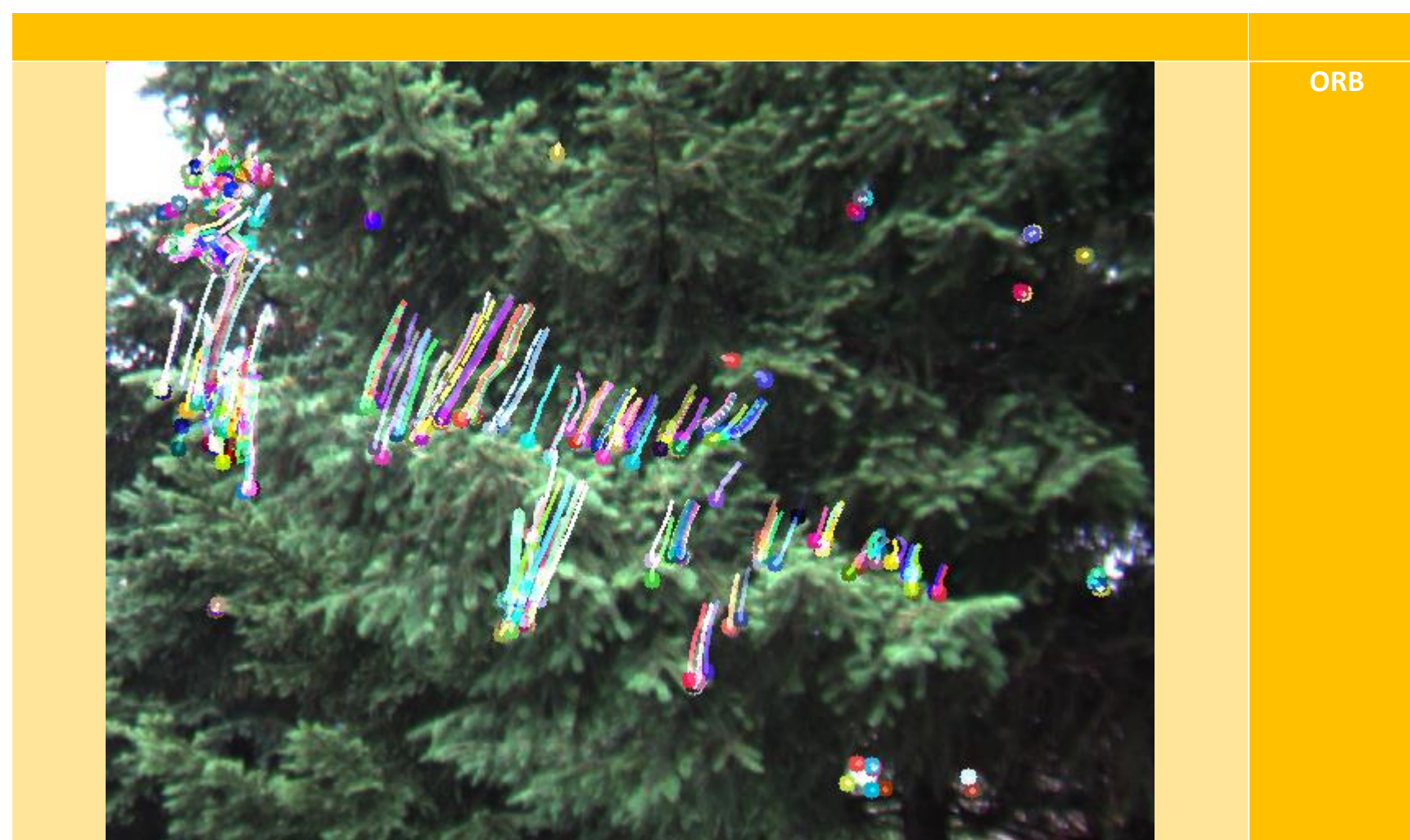
کدهای این تمرین در فایل `problem-1-1-1.py` قرار دارد، در این سوال خواسته شده است که روش‌های مختلفی را برای ویژگی‌های استخراج شده از تصویر به کار ببریم و سپس روش `Lukas Kanade` را استفاده کنیم. برای این کار از چهار ویژگی زیر استفاده کرده‌ایم:

- SIFT •
- SURF •
- ORB •
- Shi-Tomasi •

خروجی‌های این قسمت در پوشه‌ی 1-1-1 problem قرار دارد. در زیر محتوای پوشه رو مشاهده می‌کنید:

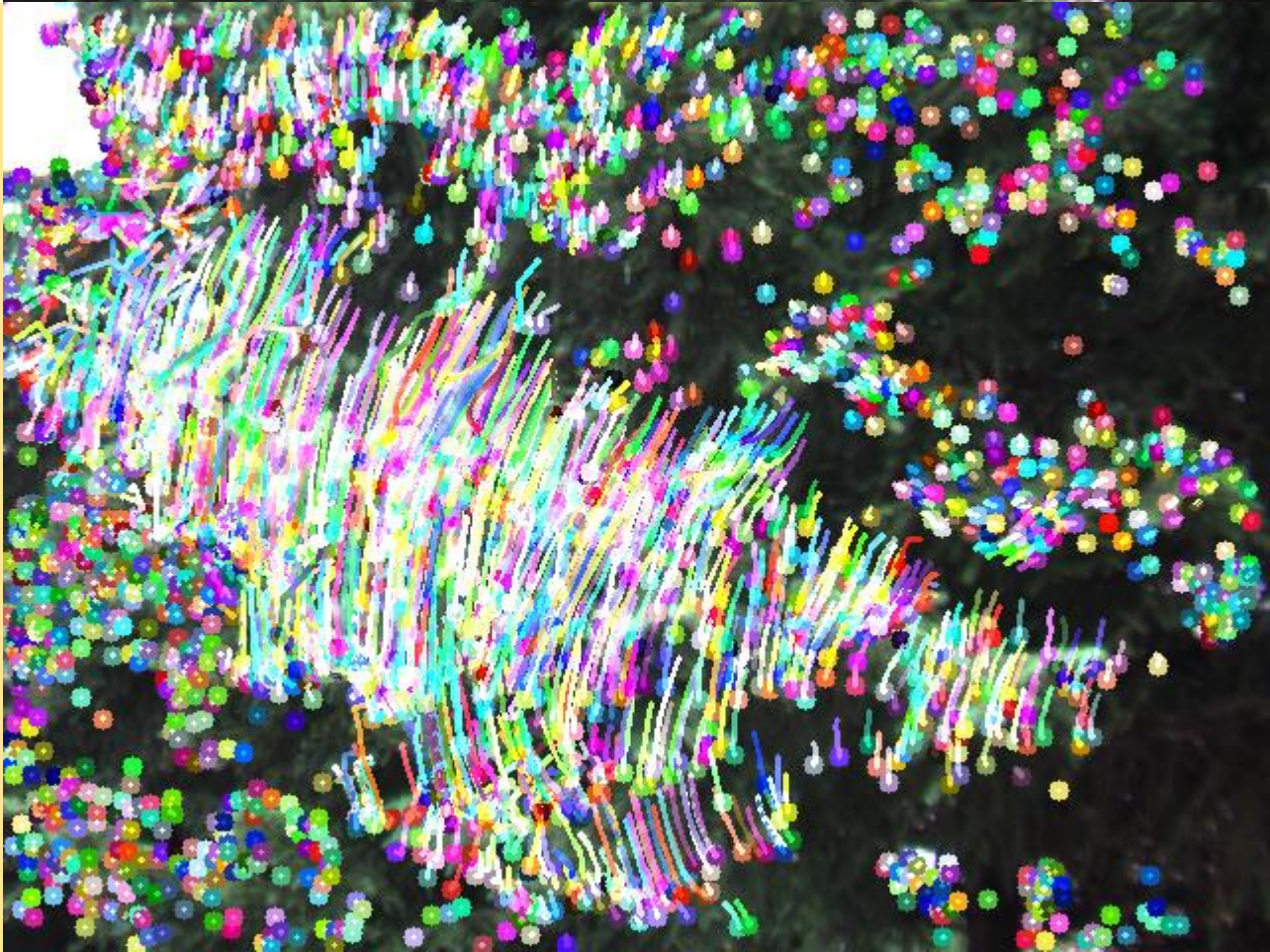


در ادامه چند مورد از خروجی‌ها را مشاهده می‌کنیم، برای دیدن تمام خروجی‌ها باید به پوشه **problem-1-1-1** مراجعه کرد.

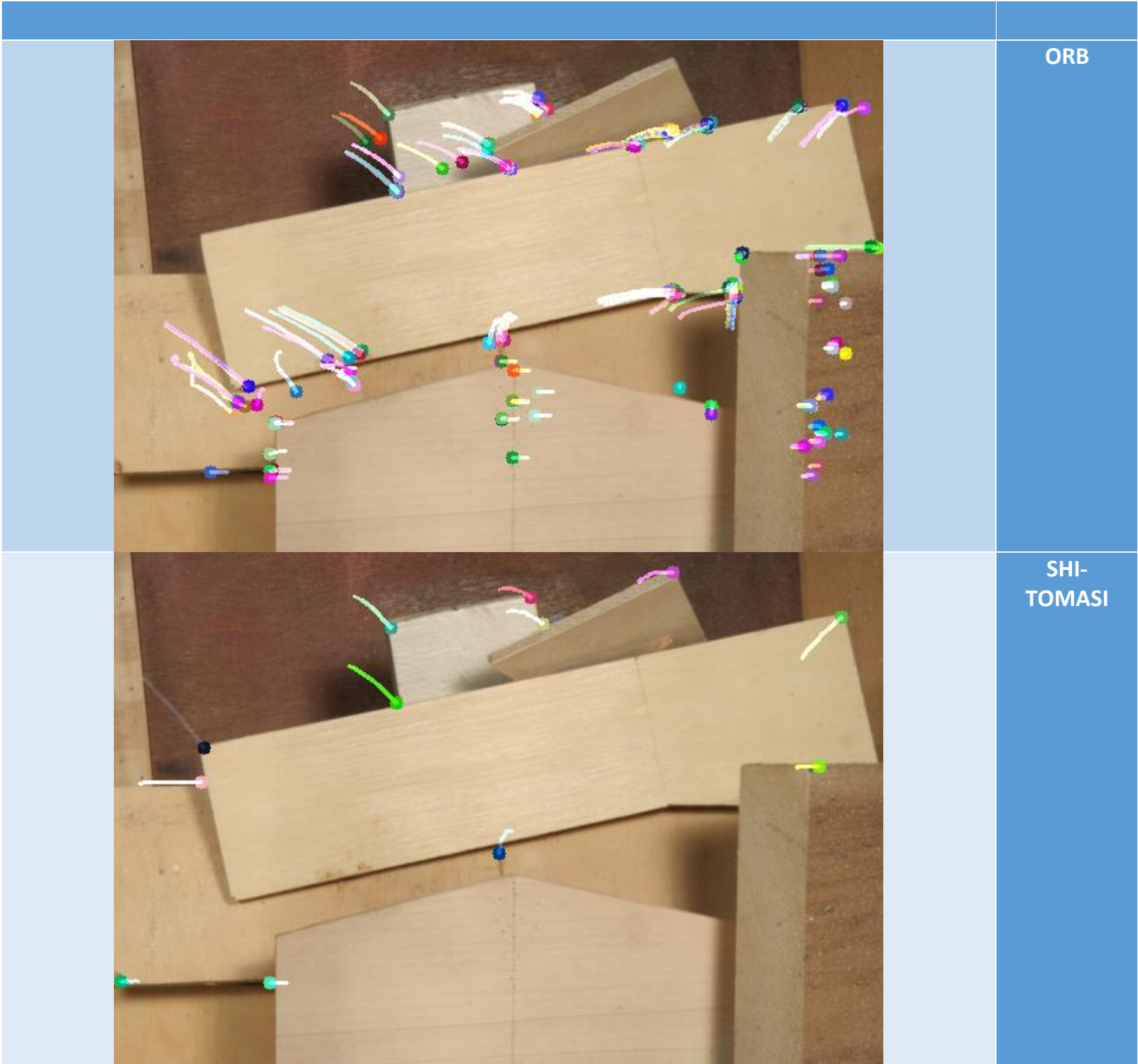
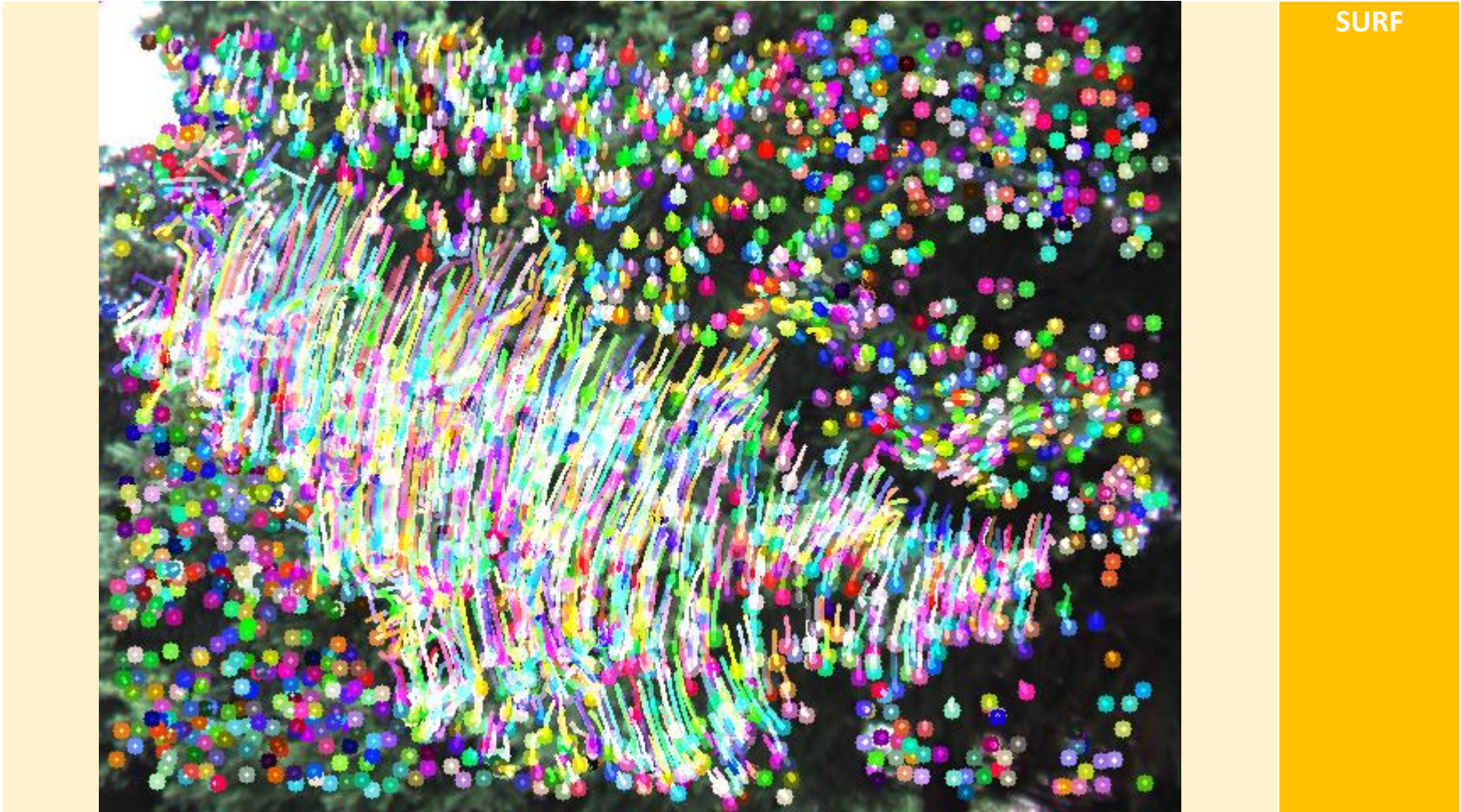






SHI-
TOMASI



SIFT



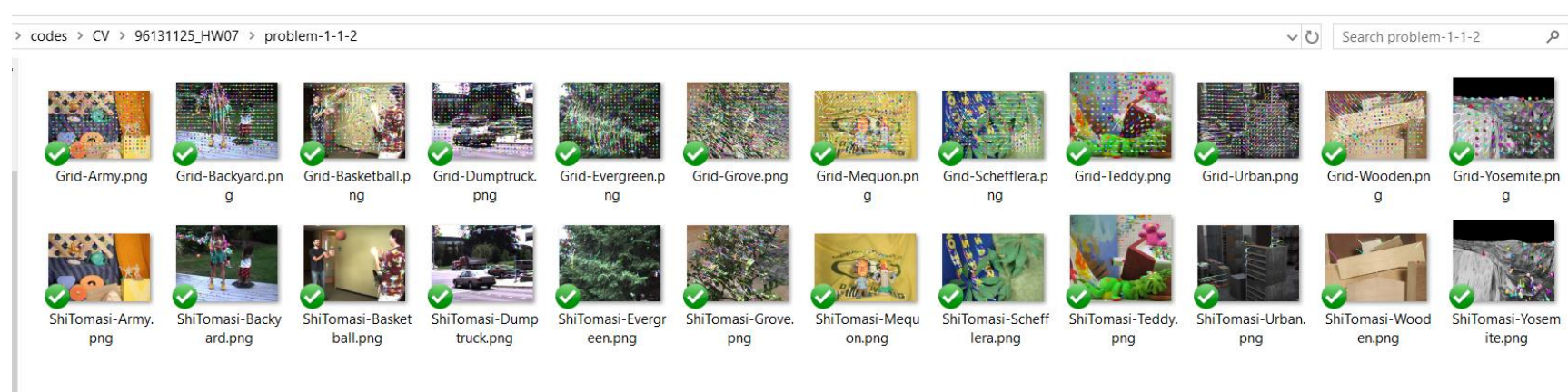
		SIFT
		SURF

نتیجه‌گیری: عملکرد ردگیری ارتباط مستقیمی با نوع ویژگی استفاده شده دارد. در ادامه چند مورد را بیان می‌کنیم.

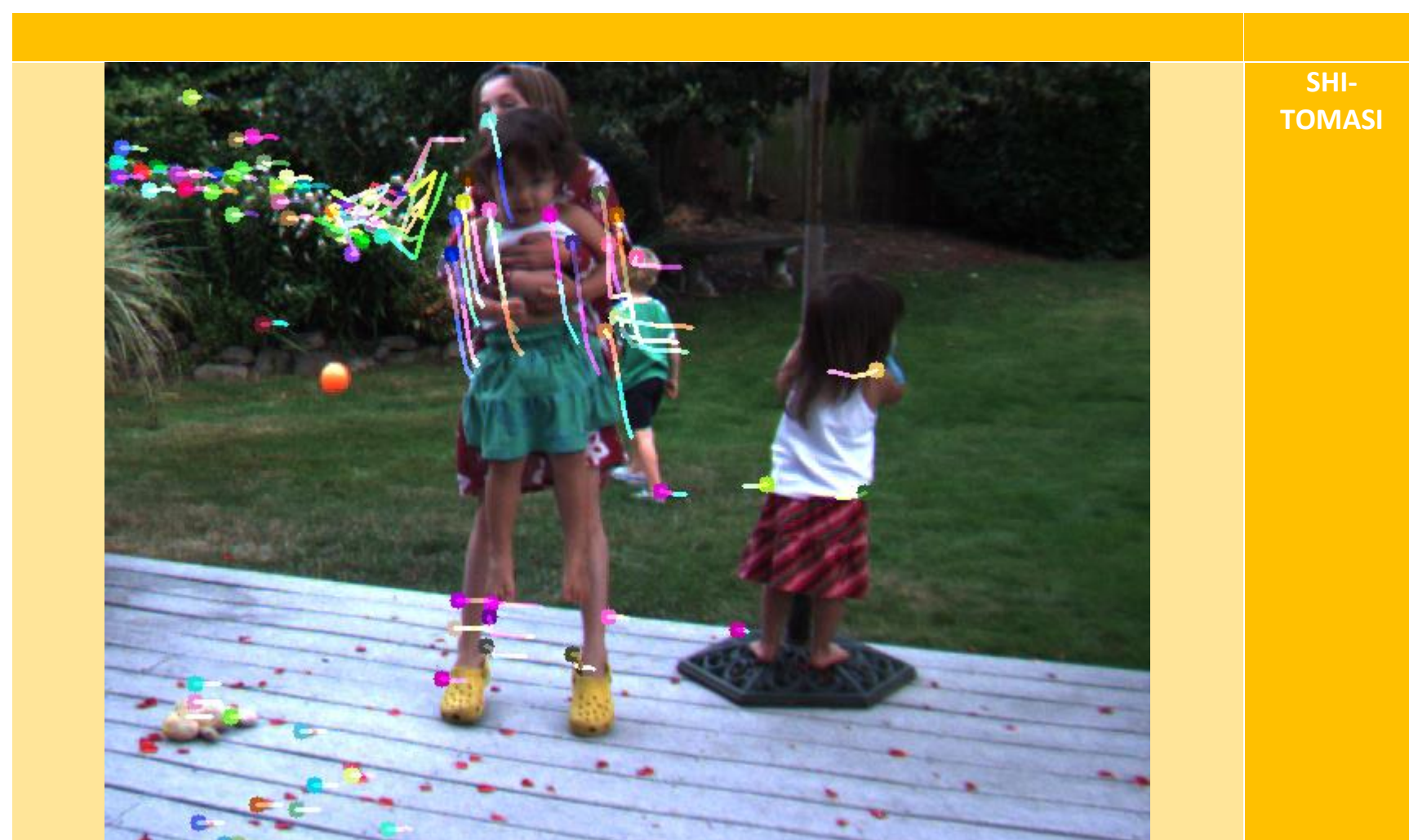
- دو نوع ویژگی داریم: ۱- گوشه ۲- حباب. گوشه‌ها محل تلاقی ۲ یا بیش از ۲ لبه هستند (مانند سورف). حباب نقاطی هستند که لبه و گوشه نیستند ولی از همسایگی خود متمایزاند. هر کدام برای محیط‌هایی خوب هستند. گوشه‌ها برای محیط‌هایی که بافت کم است مناسب است. در نتیجه از چه ویژگی استفاده کنیم تاثیر مستقیم در کیفیت ویژگی‌های استخراج شده در محیط دارد.
- تعداد ویژگی‌های استخراج شده. با توجه به اینکه چه ویژگی انتخاب می‌شود تعداد متفاوتی ویژگی استخراج می‌شود.
- هر چه روش استخراج ویژگی نادقیق‌تر باشد، در فرآیند استخراج ویژگی احتمال انتخاب نقاطی که گوشه و یا حباب نیستند بیشتر می‌شود. نقاطی هم که گوشه و حباب نیستند به راحتی در حین ردگیری گم می‌شوند و یا اشتباه ردگیری می‌شوند.

تمرین ۱-۱-۲

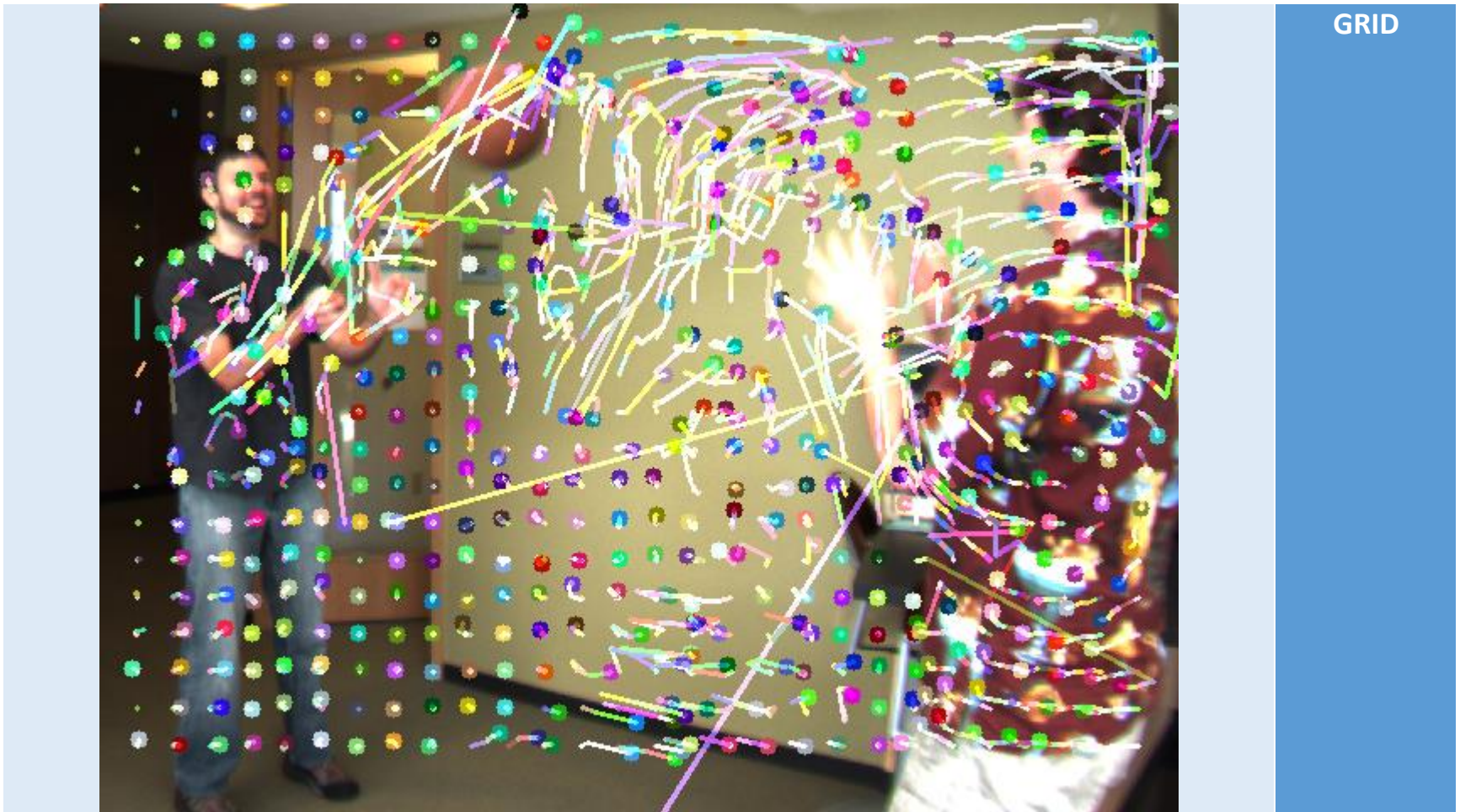
در این قسمت از سوال خواسته شده است که از دورش متفاوت برای انتخاب ویژگی بهره ببریم. یک بار از روش گوشه‌ی اجسام و بار دیگر از رئوس توری استفاده کنیم. کدهای این قسمت در فایل `problem-1-1-2.py` قرار دارد. خروجی‌های این قسمت در پوشه‌ی `problem-1-1-2` قرار دارد. در زیر محتوای این پوشه را مشاهده می‌کنید:



برای گوشه اجسام باید از ویژگی استفاده شود که بر مبنای لبه باشد به همین دلیل می‌توان از ویژگی‌های مانند SURF و یا Shi-Tomasi استفاده کرد. در این سوال از Shi-Tomasi استفاده شده است. برای روش جدول نیز از توری استفاده شده است که فاصله بین نقاط روی توری ۲۲ پیکسل است. در زیر چند مورد از خروجی‌های کد را مشاهده می‌فرمایید، برای دیدن تمام خروجی‌ها به پوشه `problem-1-1-2` مراجعه شود.







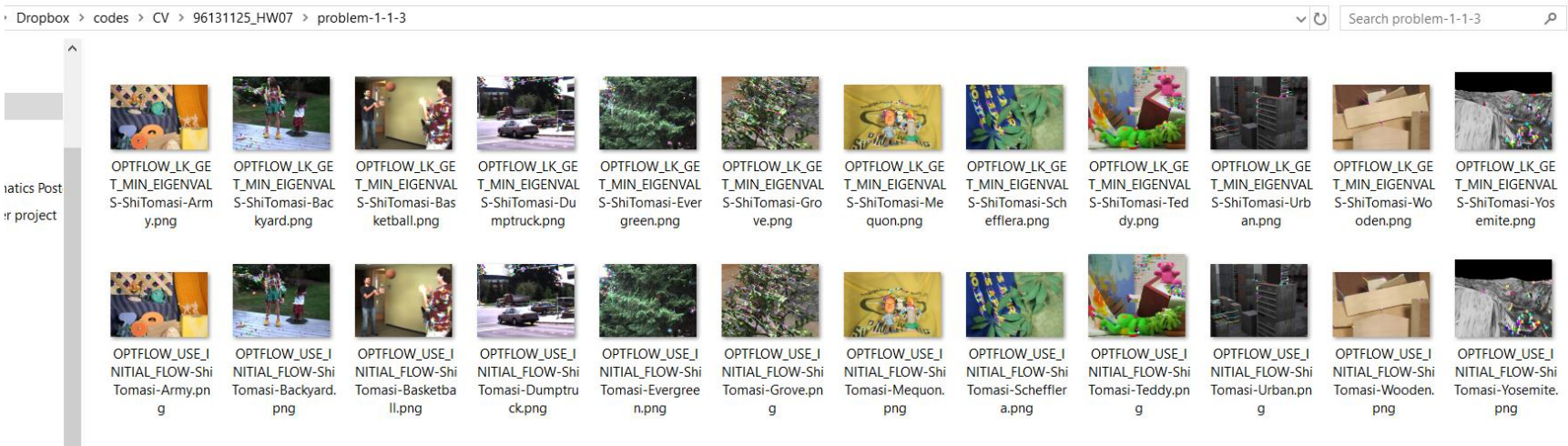
نتیجه‌گیری: روش Lucas Kanade روشی Sparse است. یعنی تعدادی نقطه انتخاب می‌شود و فقط حرکت برای آن نقاط محاسبه می‌شود. این روش دقیقاً در مقابل روش Dense قرار دارد که حرکت برای تمام نقاط محاسبه می‌شود. با استفاده از توری روش sparse optical flow را به dense optical flow تبدیل می‌کنیم. در روش توری، در بسیار از نقاط حرکتی وجود ندارد و نقاط ثابت مانده‌اند. خیلی از این نقاط، نقاطی هستند که با همسایه‌های اطراف خود تفاوت چندانی ندارند و در حین ردگیری گم می‌شوند و یا به اشتباه ردگیری می‌شوند. البته خوبی‌هایی هم در آن دیده می‌شود به عنوان مثال مسیر و حرکت توپ، تشخیص داده شده است در حالی که در روش مبتنی بر ویژگی گوشه چنین چیزی رخ نمی‌دهد.

تمرین ۳-۱-۱

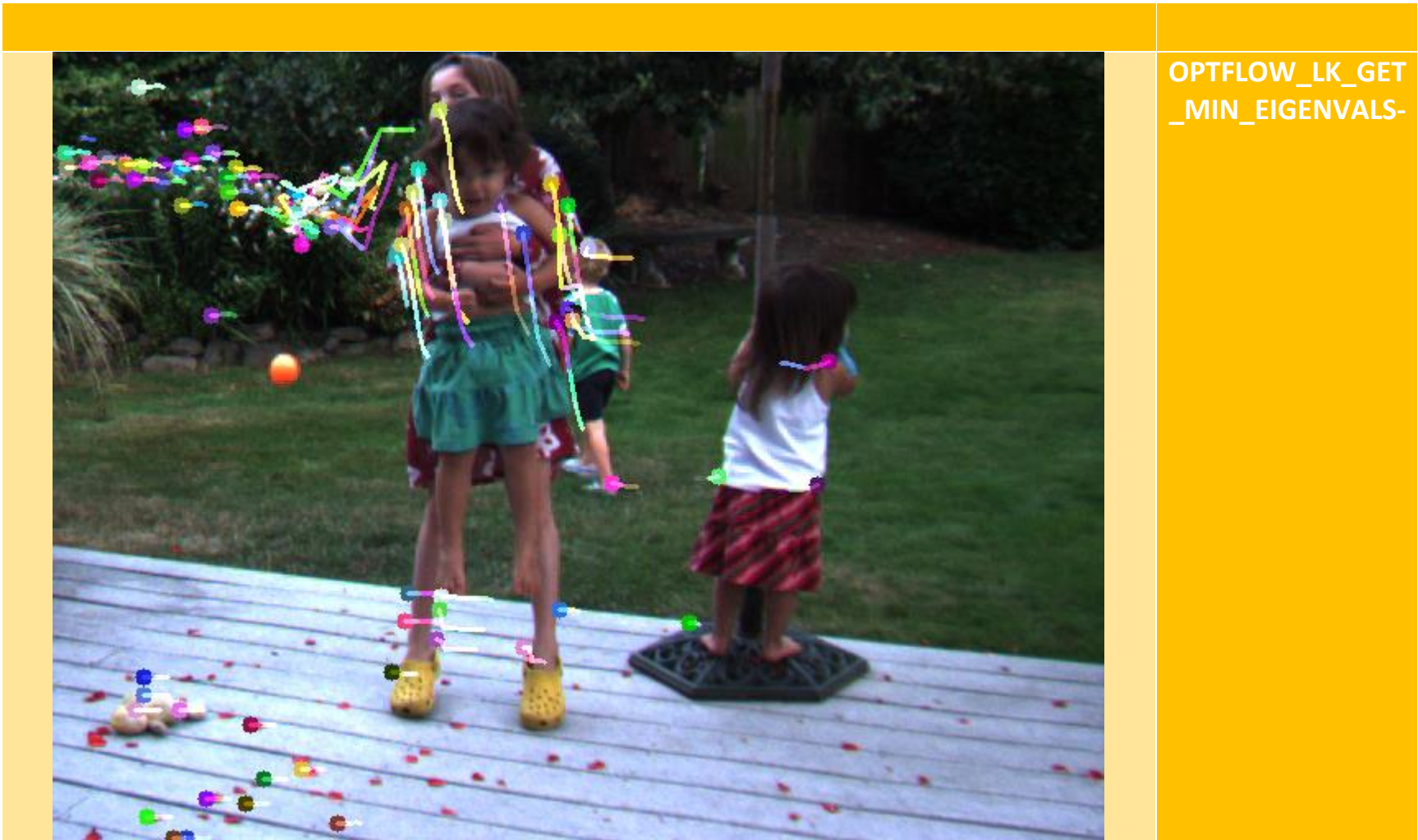
کدهای این قسمت در problem-1-1-3.py قرار دارد. در این قسمت خواسته شده است از دو flag زیر استفاده شود:

OPTICAL_FLOW_LK_GET_MIN_EIGENVALS و OPTICAL_FLOW_USE_INITIAL_FLOW

خروجی‌های این قسمت در پوشه‌ی problem-1-1-3 قرار دارد، در زیر محتوای این پوشه‌ها را مشاهده می‌کنید.



در زیر نمونه‌ای از خروجی کد را مشاهده می‌کنید. برای دیدن تمام خروجی‌ها به پوشه‌ی problem-1-1-3 مراجعه فرمایید.



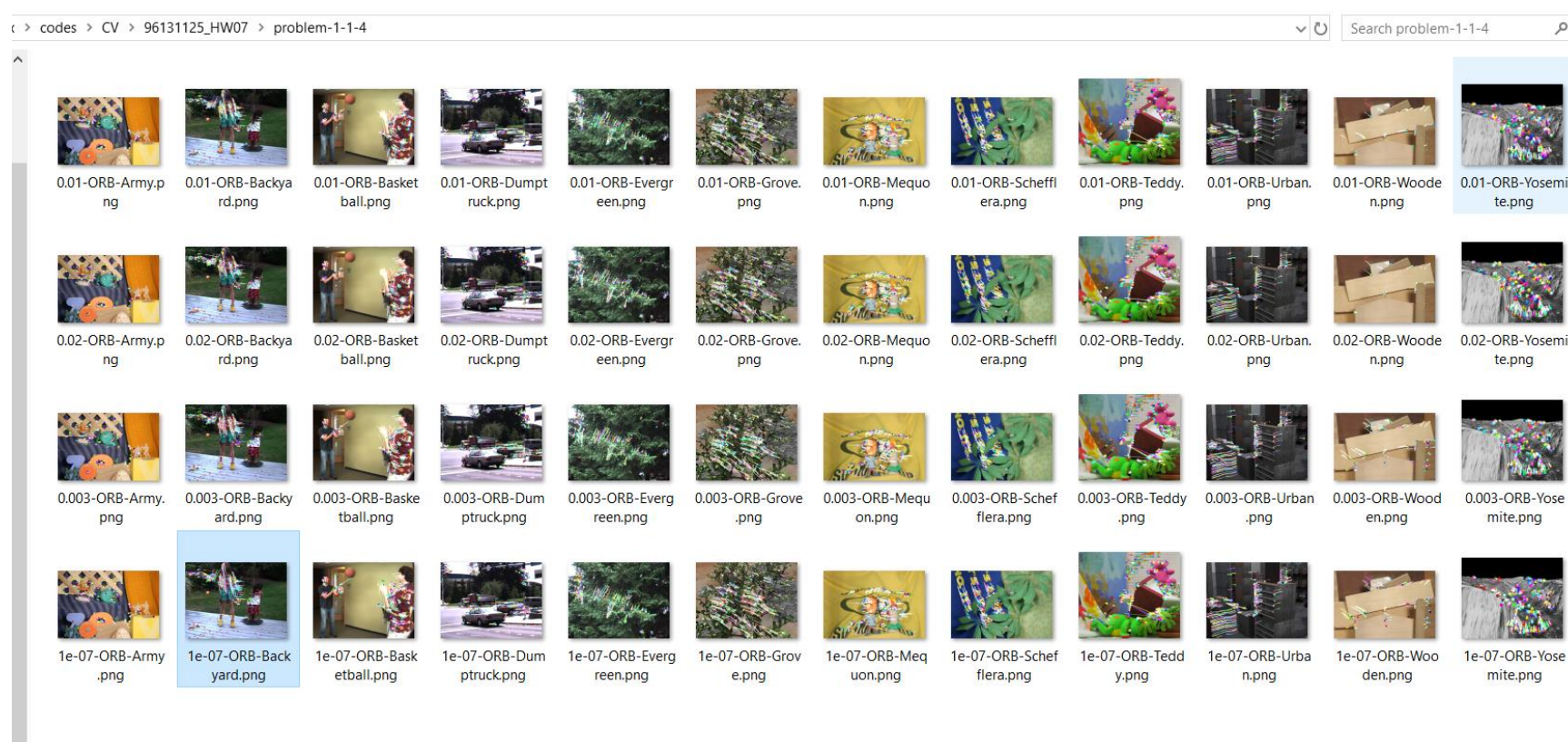


در هنگام استفاده از فلگ `cv.OPTFLOW_USE_INITIAL_FLOW` می‌توان آرگمانی با اسم `nextPts` به تابع داد. این آرگمان یک `prior information` در مورد محل قرارگیری نقاط در فریم بعدی است. این اطلاعات پیشین می‌تواند توسط یک الگوریتم دیگر محیا شود یا از همین روش لوکاس به صورت برعکس استفاده کنیم. یعنی در ابتدا نقاط فریم حال حاضر را به الگوریتم بدهیم و محل نقاط در فریم بعد را به دست بیاوریم و در ادامه نقاط ردگیری شده را به تابع بدهیم و بخواهیم در فریم قبل محل نقاط را پیدا کند. در نهایت `status` فاز `backward` و `forward` را باهم ترکیب کنیم، اینگونه نقاطی که ردگیری شده‌اند ولی از روی نقاط ردگیری شده نمی‌توان به نقاط اولیه‌شان رسید حذف می‌شوند.

در صورتی که فلگ `OPTFLOW_LK_GET_MIN_EIGENVALS` استفاده شود، کمترین مقدار ویژه یک ماتریس 2×2 که مربوط به محاسبات شارنوری است، محاسبه می‌شود و در صورتی که کمتر از `minEigThreshold` باشد، از رهگیری نقطه صرف نظر می‌شود. اگر از این فلگ استفاده نشده باشد از نرم یک نقطه حرکت داده شده با همسایگی‌اش قبل از حرکت استفاده می‌شود.

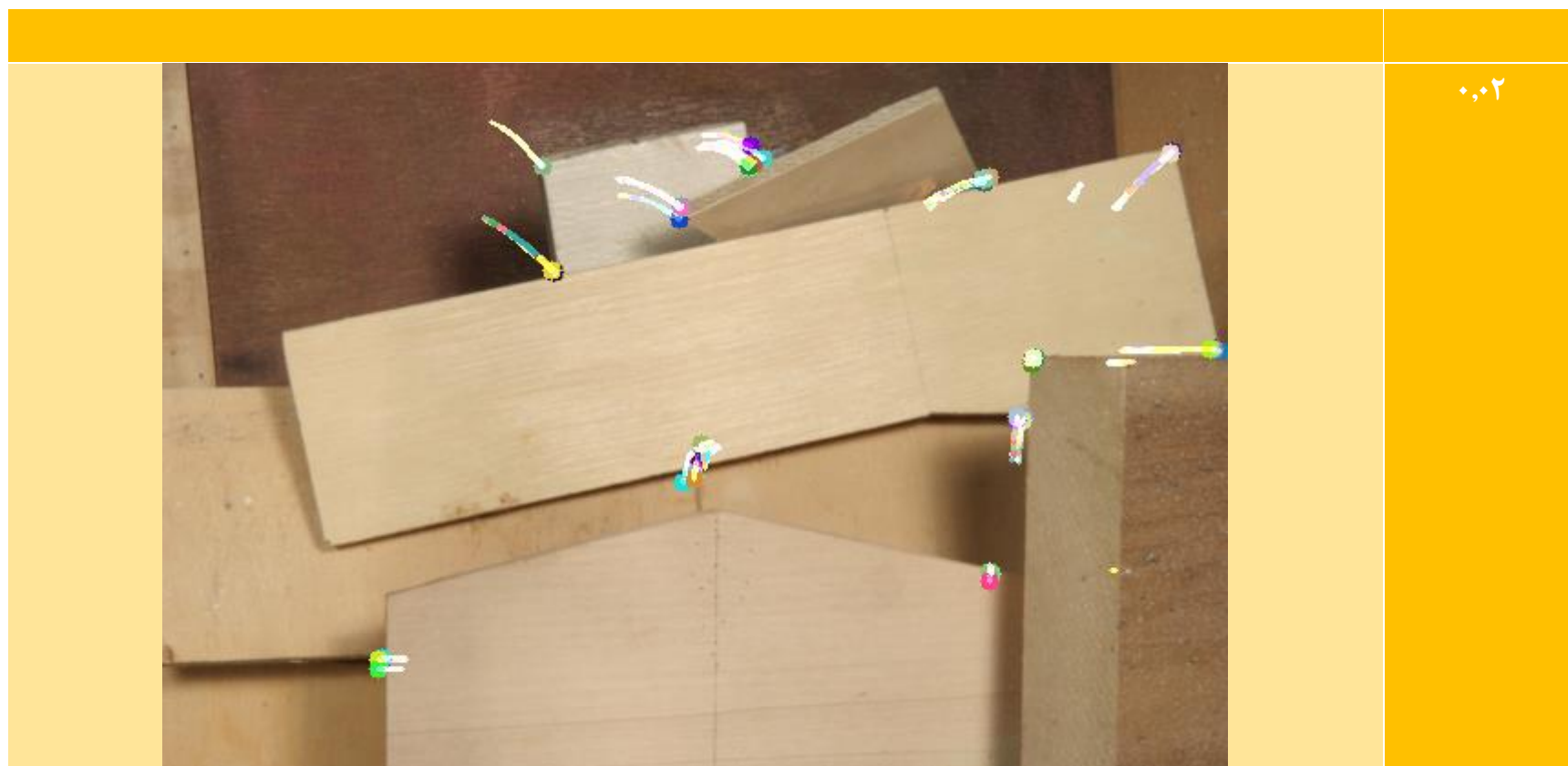
تمرین ۴-۱-۱


در این قسمت خواسته شده است تاثیر مقدار `minEigThreshold` بررسی شود. کدهای این تمرین در `problem-1-1-4.py` قرار دارد. همچنین خروجی‌های این سوال در پوشه `problem-1-1-4` قرار دارد. در زیر محتوای پوشه‌ی خروجی‌های این سوال را مشاهده می‌کنید:



در این سوال ۴ مقدار مختلف $\{0.0000001, 0.0003, 0.01, 0.02\}$ برای `minEigThreshold` در نظر گرفته شده است.

در زیر یک نمونه خروجی را مشاهده می‌کنید، برای دیدن تمام خروجی‌های این سوال به پوشه `problem-1-1-4` مراجعه شود:



		۰,۰۱
		۰,۰۰۳
		۰,۰۰۰۰۰۱

نتیجه‌گیری: همین‌طور که در تصاویر بالا و سایر تصاویر در پوشه‌ی **problem-1-1-4** دیده می‌شود، هر چه میزان **minEigThreshold** زیاد شده است، نقاط کمتری در ردگیری استفاده شده است و ردگیری نقاط بیشتری زودتر تمام شده است. همان‌طور که در بخش قبل توضیح داده شد، کوچک‌ترین مقدار ویژه یک ماتریس ۲×۲ حساب می‌شود که این مقدار ویژه در صورتی که کمتر از **minEigThreshold** باشد، از ردگیری نقطه صرف نظر می‌شود. به همین دلیل هر چه مقدار این آرگمان بیشتر می‌شود، نقاط بیشتر رد می‌شود و نقاطی که در ردگیری استفاده می‌شوند، کاهش می‌یابد.

تمرین ۱-۲

در این تمرین خواسته شده است که نقشه عمق صحنه از روی دو تصویر استریو ساخته شود، کدهای این بخش در فایل `problem-1-2.py` قرار دارد. خروجی‌های این کد در پوشه‌ی `problem-1-2` قرار دارد. در زیر خروجی‌های این کد را مشاهده می‌کنید:

