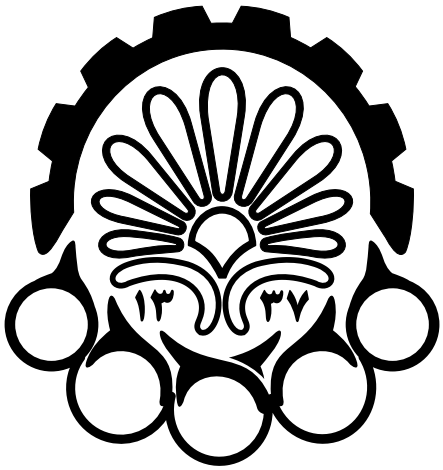


دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

تمر ین هشتم درس بینایی ماشین

دکتر صفابخش

غلامرضا دار ۴۰۰۱۳۱۰۱۸

بهار ۱۴۰۱

فهرست مطالب

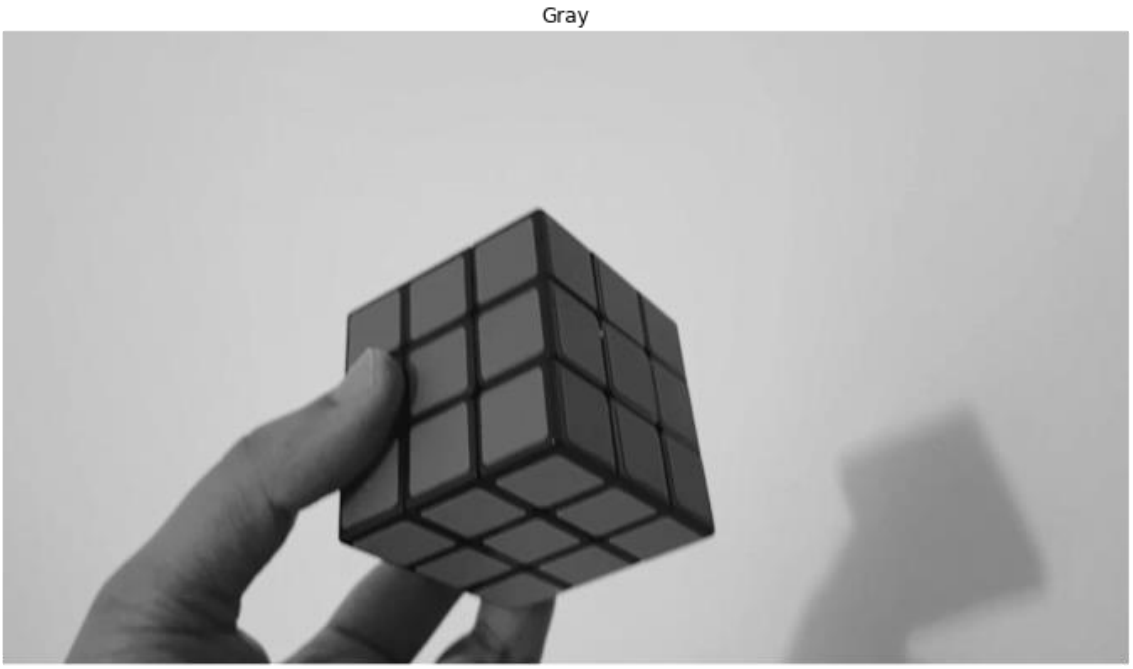
۱) استخراج نقاط به کمک Shi-Tomas ۳
۲) الگوریتم Lucas-Kanade ۶
۳) الگوریتم Gunner-Farneback ۷
۴) مقایسه روشها ۹

۱) استخراج نقاط به کمک Shi-Tomas

در این بخش به بررسی پارامترهای الگوریتم Shi-Tomas برای تشخیص گوشه‌ها می‌پردازیم.

پارامتر	توضیح
Image	تصویر ورودی سطح خاکستری برای تشخیص نقاط
maxCorners	بیشینه تعداد نقاط. اگر تعداد نقاط مطلوب بیش از این مقدار شود تنها این مقدار از بهترین نقاط برگردانده می‌شوند.
qualityLevel	این پارامتر کنترل می‌کند نقاط مطلوب باید حداقل چه کیفیتی داشته باشند. بسته به اینکه برای تشخیص نقاط از الگوریتم Harris یا MinEigenVal استفاده شود محاسبه کیفیت نقاط متفاوت خواهد بود.
minDistance	حداقل فاصله اقلیدسی نقاط مطلوب مجاور. از تراکم نقاط در یک ناحیه جلوگیری میکند.
mask	ماسکی که مشخص می‌کند از نقاط باید از کدام بخش‌های تصویر انتخاب شوند.
blockSize	اندازه بلاک برای محاسبه Derivative Covariation Matrix در الگوریتم cornerMinEigenVal
useHarrisDetector	اگر برابر True باشد از الگوریتم Harris و در غیر این صورت از الگوریتم cornerMinEigenVal استفاده می‌کند.

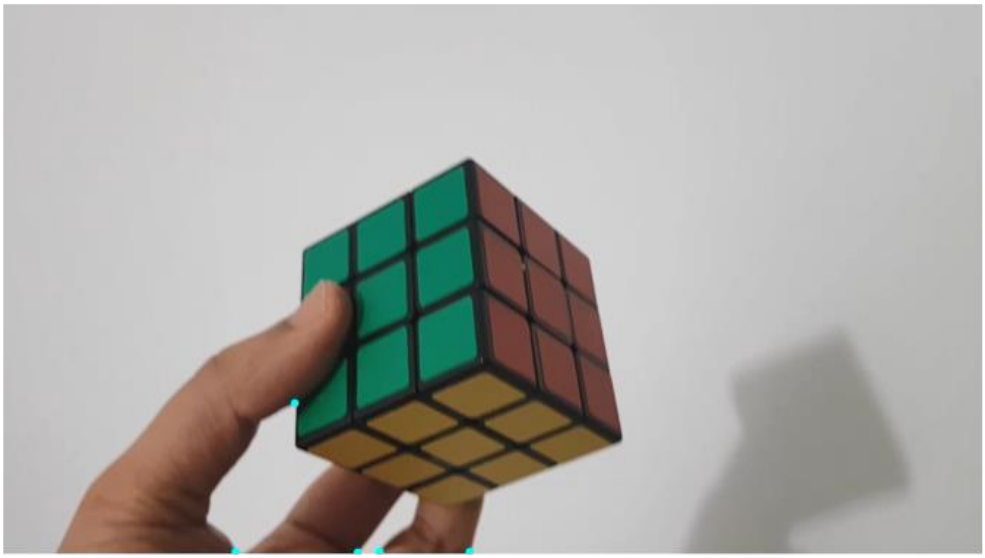
در ادامه با تغییر این پارامترها سعی می‌کنیم بهترین نقاط ممکن را برای تصویر زیر پیدا کنیم. لازم به ذکر است که تصویر ورودی این تابع سطح خاکستری است.



تنظیمات پیشفرض(نقاط تشخیص داده‌شده با رنگ آبی روشن نشان داده شده اند)

```
maxCorners = 10,
qualityLevel = 0.3,
minDistance = 0,
blockSize = 7
```

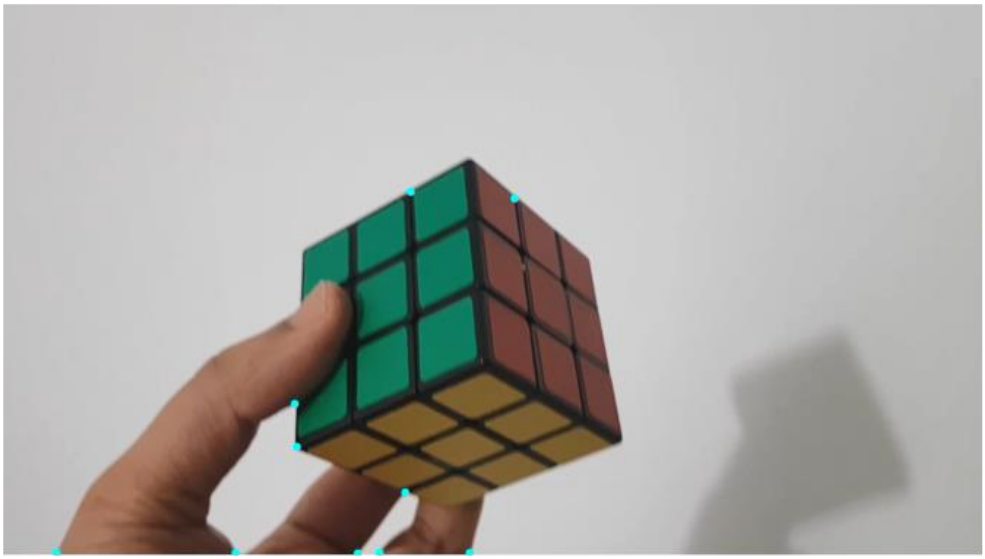
Features



با این تنظیمات تعداد نقاط مطلوب بسیار کم است. برای رفع این مشکل می‌توان qualityLevel را کاهش داد تا نقاطی با کیفیت کمتر نیز پذیرفته شوند.

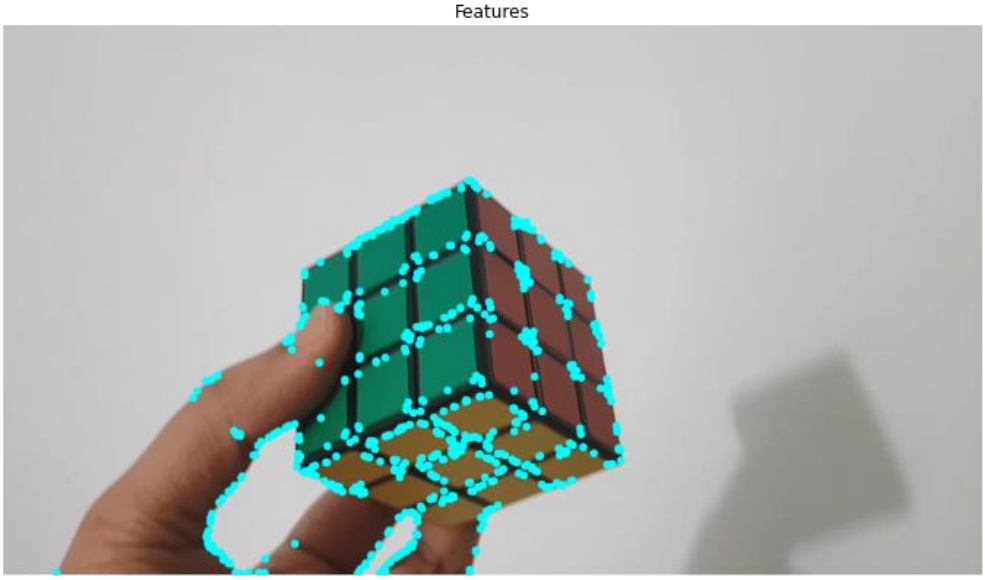
```
qualityLevel = 0.01
```

Features



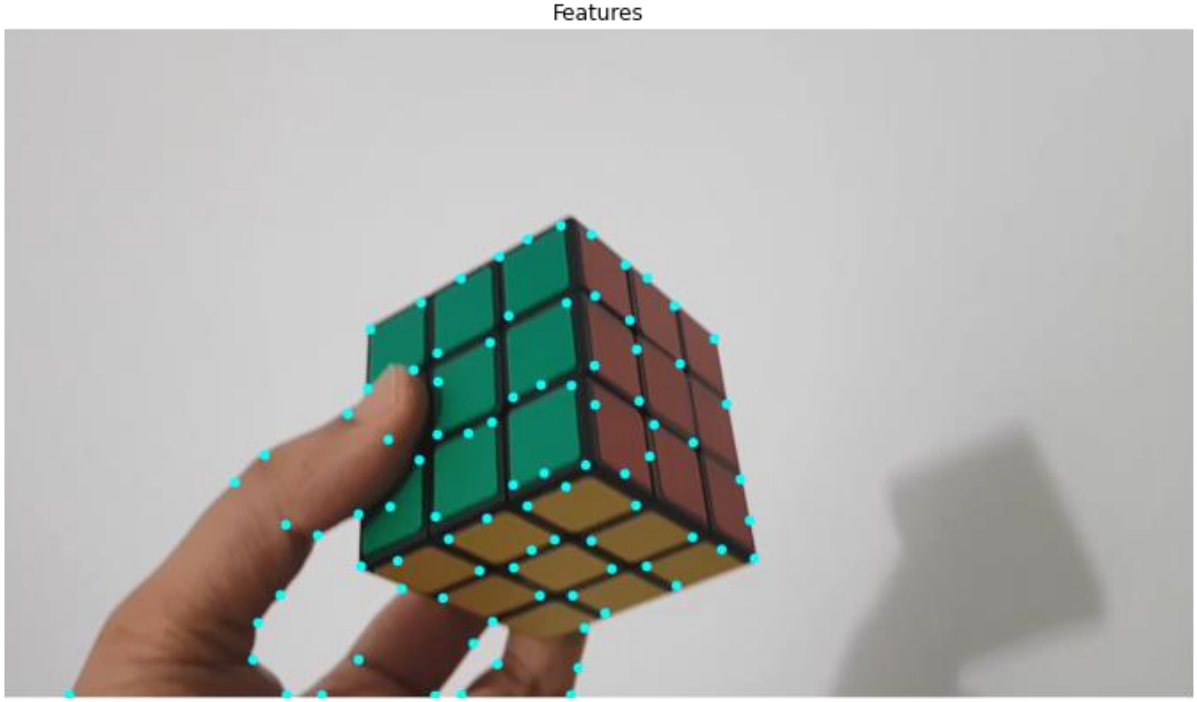
حالا نقاط غیرمطمئن‌تری نیز پذیرفته می‌شوند و تعدادی نقطه بر روی سوژه اصلی تصویر نیز داریم. اما تعداد کل نقاط پذیرفته شده کم است بنابراین پارامتر maxCorners را تغییر می‌دهیم.

```
maxCorners = 1000
```



حالا تعداد خیلی زیادی نقطه نزدیک به هم داریم. با افزایش پارامتر minDistance می‌توان این نقاط بسیار نزدیک را حذف کرد.

```
minDistance = 40
```



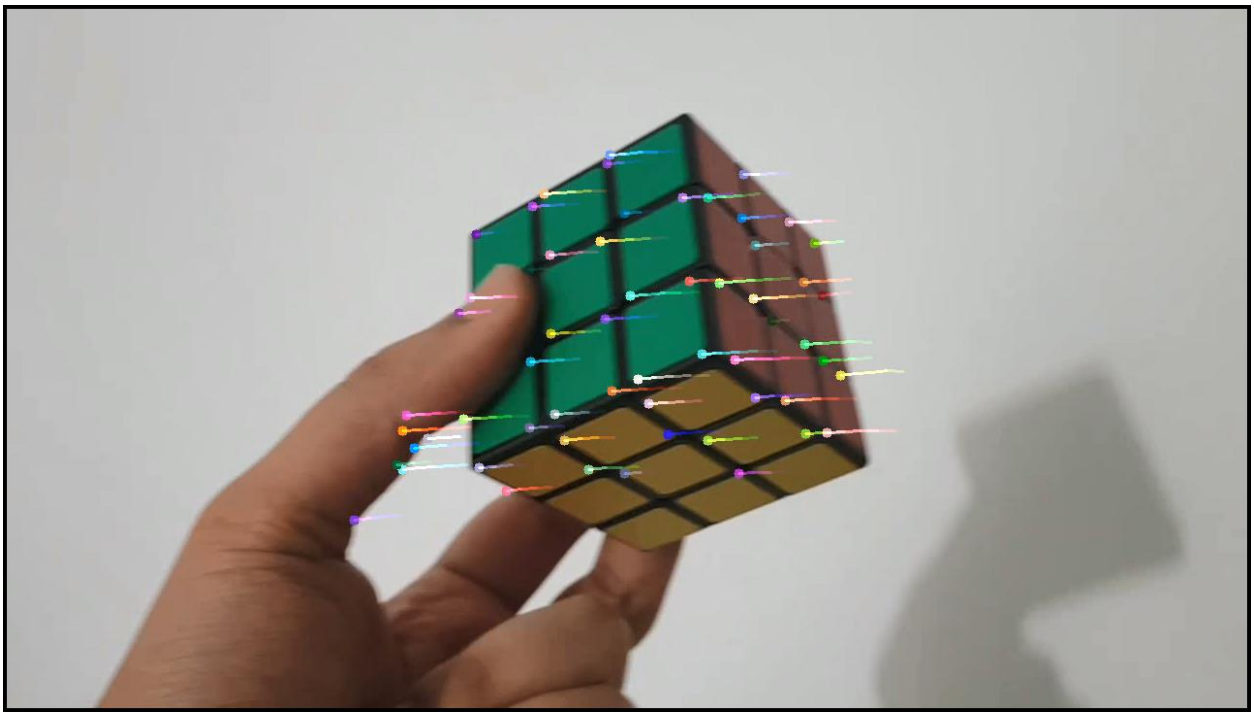
نتیجه حاصل برای ردیابی سوژه مناسب است و در بخش بعد از پارامترهای مشابه استفاده خواهیم کرد.

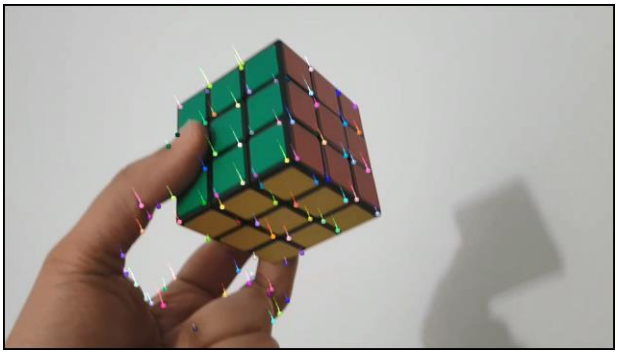
۲) الگوریتم Lucas-Kanade

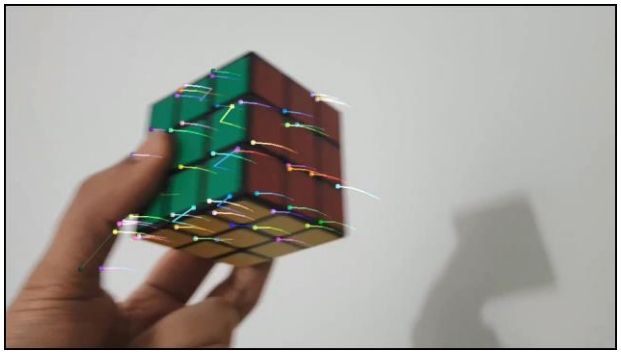
در این بخش از الگوریتم Lucas-Kanade برای محاسبه شارنوری Sparse استفاده می‌کنیم. ورودی این الگوریتم نقاط تعیین شده در مرحله قبل برای فریم اول ویدیو است. در طی تکرارهای متوالی، این الگوریتم تصویر را بررسی می‌کند و موقعیت جدید نقاط را تعیین می‌کند. همچنین برخی نقاط بین فریم‌ها گم می‌شوند. در هر مرحله با کمک خروجی st این الگوریتم، نقاط گم شده را نادیده می‌گیریم.

برای دیدن ویدیوی نتیجه این بخش به پوشه results مراجعه کنید. در ادامه تعدادی فریم از این ویدیو را مشاهده می‌کنید.

results/ output_LK_rubik_2.mp4





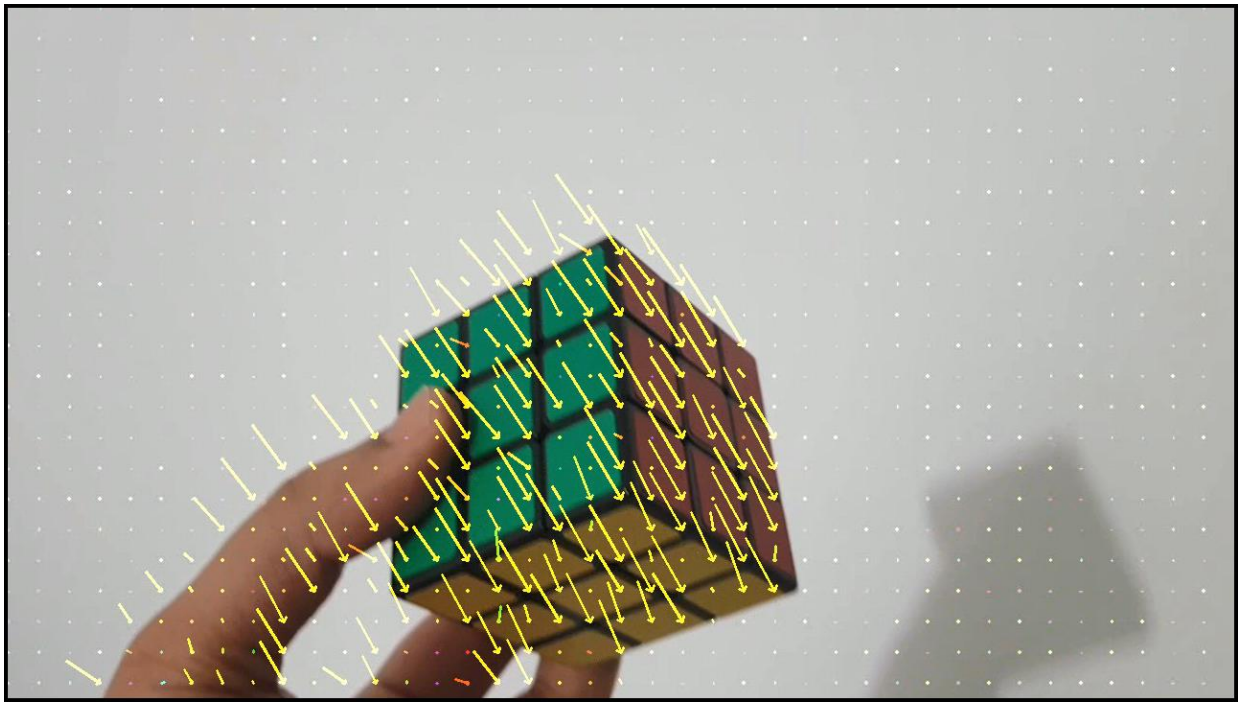


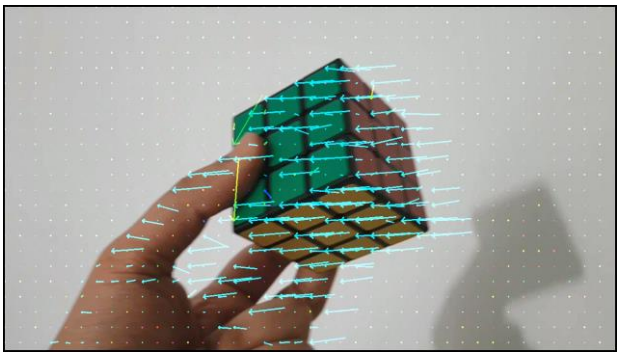
۳) الگوریتم Gunner-Farneback

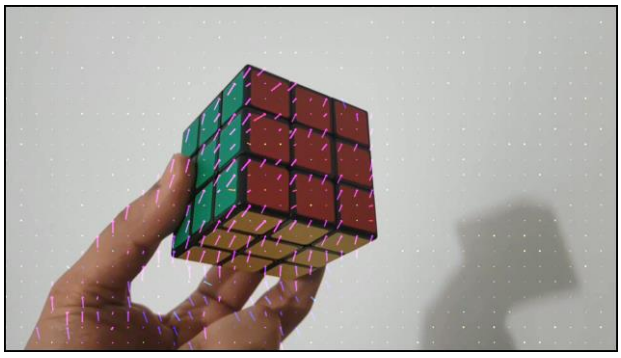
در این بخش، مشابه بخش قبل بر روی ویدیوی نمونه شار نوری را محاسبه می‌کنیم. اما این بار از الگوریتم gunner-farneback برای محاسبه شار نوری استفاده می‌کنیم. این الگوریتم به صورت Dense برای هر پیکسل تصویر یک بردار دوبعدی که نشان‌دهنده جهت و سرعت حرکت اطلاعات زیر آن پیکسل است تولید می‌کند.

برای دیدن ویدیوی نتیجه این بخش به پوشه *results* مراجعه کنید. در ادامه تعدادی فریم از این ویدیو را مشاهده می‌کنید.

results/ output_Farneback_vf_rubik_2.mp4

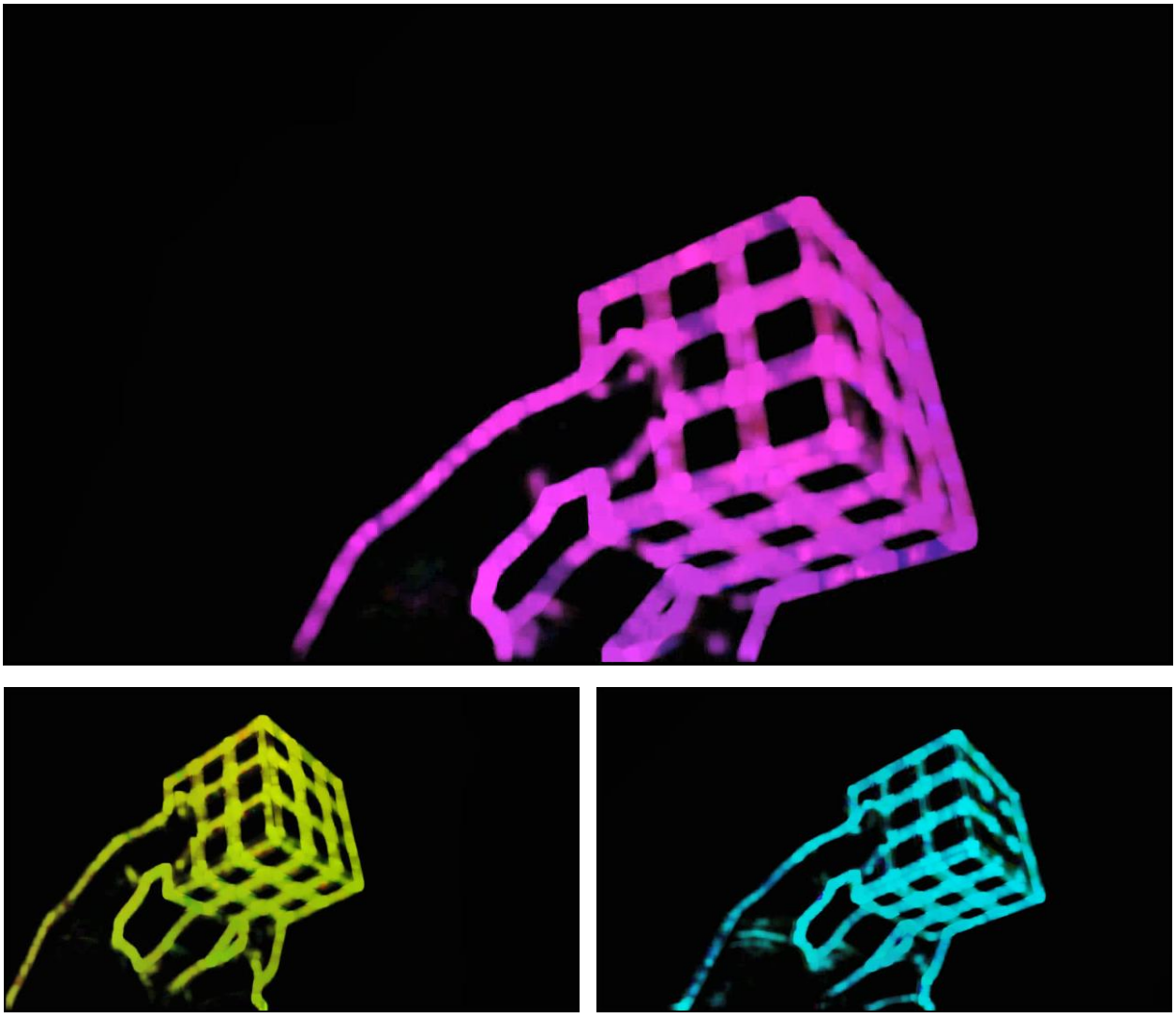






نوعی خروجی دیگر: جهت حرکت هر پیکسل Hue و میزان حرکت Value را در فضای رنگی HSV کنترل می‌کنند. مقدار S برای تمام پیکسل‌ها برابر 255 قرار داده شده است.

results/ output_Farneback_rubik.mp4



۴) مقایسه روش‌ها

همان‌طور که می‌دانید، می‌توان از الگوریتم‌های Feature matching مانند SIFT برای تطبیق‌دادن دو تصویر استفاده کنیم. اگر این دو تصویر، فریم‌های متوالی یک ویدیو باشند و این کار را به طور متوالی برای فریم‌های دیگر نیز انجام دهیم، درواقع به کمک تطبیق ویژگی عمل Tracking را انجام داده ایم. منتهی این روش‌ها برای این کار ساخته نشده اند و نسبت به روش‌های موجود برای محاسبه شار نوری از سرعت و دقت کمتری برخوردارند. یکی از برتری‌های الگوریتم‌هایی مانند Lucas-Kanade و Gunner-Farenback که در این تمرین با آن‌ها آشنا شدیم، استفاده از زمان به عنوان ورودی به مسئله است.

نحوه کار این الگوریتم‌ها به طور کلی به این صورت است که ۱) فرض می‌کنند **میزان روشنایی پیکسل‌ها در بین فریم‌های متوالی سازگار است** (تغییر ناگهانی ندارد) و ۲) فرض می‌کنند **پیکسل‌های همسایه، حرکت مشابه** دارند.

تفاوتی که بین دو الگوریتم بررسی‌شده در این تمرین وجود دارد این است که الگوریتم Lucas-Kanade به صورت Sparse عمل می‌کند و نیاز دارد قبل از شروع، تعدادی نقطه کاندید برای ردیابی به آن داده شوند. در این تمرین دیدیم که با استفاده از الگوریتم Shi-Tomas می‌توان این نقاط را برای این الگوریتم مهیا کرد. این الگوریتم در طول ویدیو، سعی می‌کند مسیر حرکت این نقاط کاندید را ردیابی کند. اما الگوریتم Gunner-Farenback به صورت Dense مسیر حرکت هر پیکسل را در طول ویدیو بررسی می‌کند.

هر کدام از این الگوریتم‌ها می‌توانند کاربرد خاص خود را داشته باشند. به عنوان مثال اگر یک سوژه مشخص در ویدیو داشته باشیم و بخواهیم مسیر حرکت آن را به طور خاص و دقیق بررسی کنیم، الگوریتم Lucas-Kanade گزینه بهتری است. اما اگر بخواهیم به طور کلی در یک ویدیو حرکت تمام متحرک‌ها را بررسی کنیم و به عنوان مثال در یک کارخانه بخواهیم میزان پایدار بودن یا لرزش قطعات را بررسی کنیم آن‌گاه الگوریتمی مشابه Gunner-Farenback می‌تواند مناسب تر باشد.