بسم الله الرحمن الرحیم

دیدِ کامپیوتری

گزارش پروژه درس

دکتر هدی محمدزاده

امیرحسین رستمی ۹۶۱۰۱۶۳۵

دانشگاه صنعتی شریف

نکته مهم:

فایل کد تحویل داده شده یک فایل JupyterNoteBook است که سلول سلول شده است و گزارش نوشته شده به صورت سلول سلول نوشته شده است.

جهت ارزیابی زمان اجرا کافی است در سلول های مدنظر تایمر زده (در ابتدا و انتهایش) و سپس زمان هر سلول را محاسبه کرده(تفاضل زمان تایمر ها) و در نهایت با جمع کردن آن ها به زمان اجرای پک سلول ها برسید.

سلول اول:(SimpleShow):

هدف: نمایش فیلم original به صورت GrayScale و سریع ☺.

در این سلول به ترتیب اعمال زیر را انجام دادیم:

1- بازکردن فایل ویدیو
2- تک تک خواندن فریم های ویدیو
3- تبدیل هر تک فریم به حالت GrayScale
4- در هر حلقه یک میلی ثانیه منتظر می ماند و نیز در صورت فشرده شدن کلید "q" نیز نمایش متوقف می شود.
5- تکرار مراحل فوق تا اتمام فیلم.
6- آزادسازی منابع استفاده شده.

یک فریم از خروجی ویدیو پخش شده:

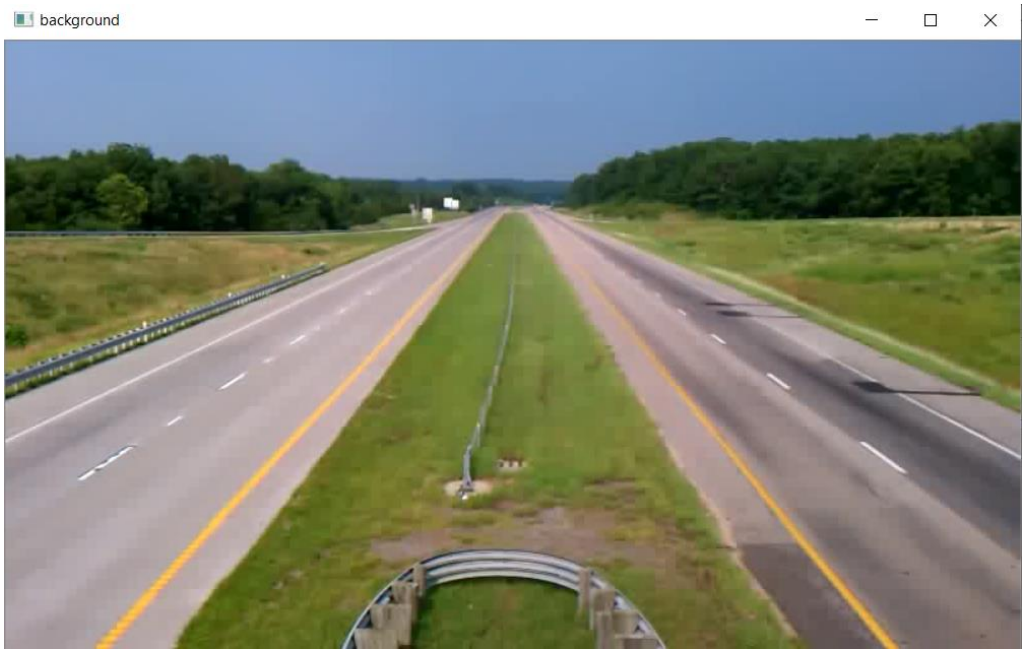
سلول دوم:(BackGroundExtration):

هدف: استخراج کردن background ویدیو.

در این سلول به ترتیب اعمال زیر را انجام دادیم:

1- بازکردن فیلم
2- انتخاب تعداد رندومی اندیس که در میان اندیس های فریم های ویدیو قرار دارند.
3- استخراج فریم های رندوم انتخاب شده
4- گذراندن لیست فریم های تصادفی بالا از فیلتر Median و
5- خروجی این فیلتر بالا برابر است با پیش زمینه تصویر و آن را نمایش می دهیم.

خروجی BackGround:



سلول سوم:(MotionDetection):

هدف:استخراج کردن هر جنبده ای! در تصویر☺.

در این سلول به ترتیب اعمال زیر را انجام دادیم:

1- ابتدا ویدیو را باز می کنیم و تعداد فریم های موجود در ویدیو را استخراج می کنیم.
2- حال در هر مرحله از حلقه تکرار که یک فریم می خوانیم اعمال زیر را انجام می دهیم:
   a. ابتدا فریم را grayscale می کنیم.
   b. از یک فیلتر گوسی تصویر grayscale قسمت قبل را رد می کنیم.
   c. اعمال فوق را برای پس زمینه نیز انجام می دهیم.
   d. تفاضل این فریم و فریمِ background را محاسبه می کنیم.
   e. خروجی تفاضل قسمت قبل را از **BinaryThreshold** عبور می دهیم.
   طبیعتا خروجی این قسمت به صورت سیاه سفید است که پیکسل های متحرک در اثر اعمال ترشهولدینگ سفیده شده است و پیکسل های پس زمینه سیاه اند.

   f. حال فیلتر dilate را روی خروجی باینری قسمت قبل اعمال می کنیم.
   #یادآوری: فیلتر dilate نقاط و حفره های سفید تصویر باینری را گشادتر می کند.
   نمونه ای از اثر فیلتر dilate:(تصویر سمت راست خروجی فیلتر dilate شده سمت چپ است).

   g. حال به کمک تابعِ cv2.findContours() کانتور های خروجی dilate شده قسمت قبل را می یابیم.
   h. حال روی کانتور های تشخیص داده شده روی قسمت قبل جاروب می زنیم:
   با تعیین حدِ ترشهولدی برای مساحت کانتور (که در کد 500 در نظرگرفتم) کانتور هارا فیلتر می کنیم(کانتور های با مساحت کمتر را صرفنظر می کنیم).

دور کانتور ها،مستطیل محاط کننده به کمک cv2.boundingRect(contour) تنظیم میکنیم و آن مستطیل را از فریم اصلی crop می کنیم و در لیست contsInOneFrame تزریق می کنیم که این لیست کلِ کانتور های قبول شده هر فریم را در خودش دارد.

i. دور مستطیل کانتور های تشخیص داده شده در تصویر نهایی مستطیل سبز رنگ رسم می کنیم.
j. حال فریمِ حاوی کانتور های مستطیل دارِ این قسمت را در لیستِ فریم هایمان تزریق می کنیم.
k. 4 نمودارِ زیر را رسم می کنیم در هر مرحله:
   - gray
   - difference(حاوی تفاضل فریم فعلی و پیش زمینه)
   - threshold(خروجی قسمت f)
   - فریمِ اصلی

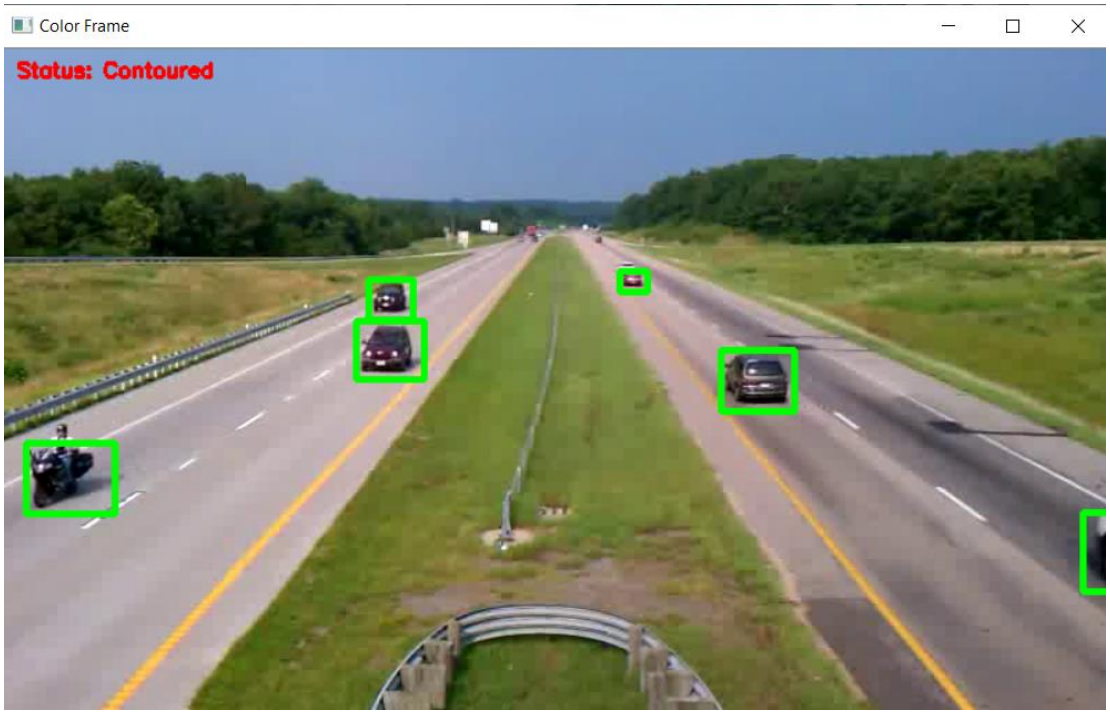
نکته:

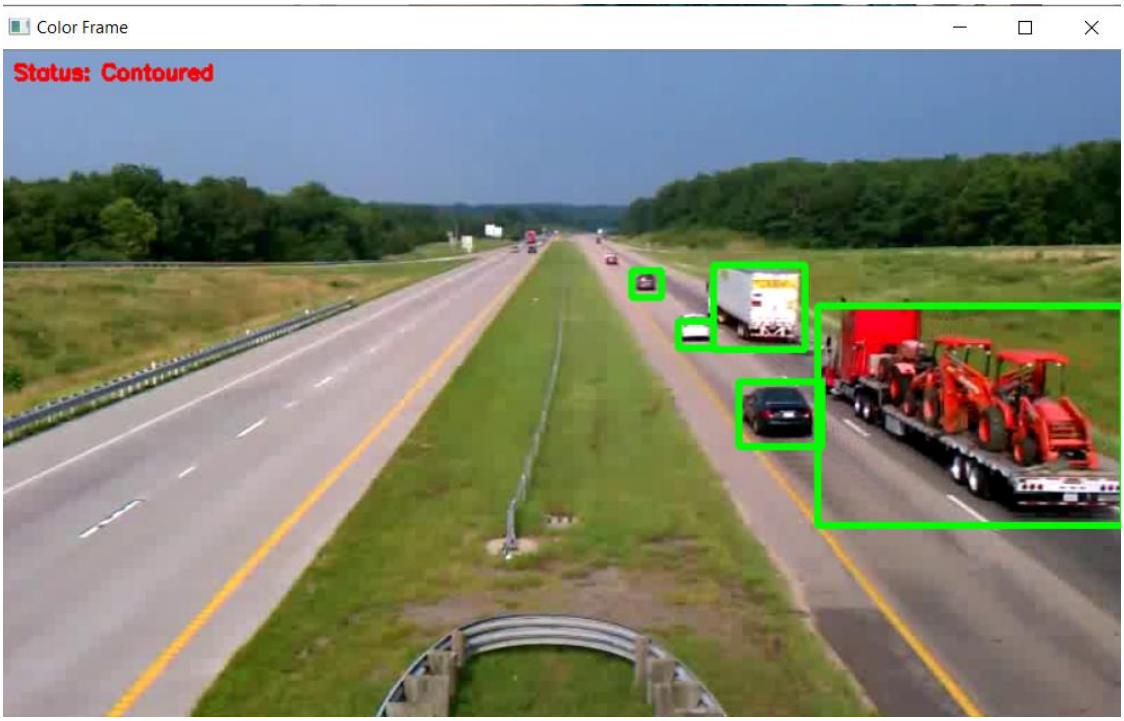
ما اطلاعات هر کانتور را در قالب tuple های با سه فیلد به شکل زیر در contsInOneFrame ذخیره می کنیم:

- **فیلد اول : کانتورِ تشخیص داده شده متحرک**
- **فیلد دوم: اطلاعات مکانی و ابعادی کانتور در فریم اصلی (x,y,w,h)**
- **زمانِ تشخیص کانتور.**

نکته مهم: زمان در فیلد بالا زمانِ تشخیص کانتور طبق ساعتِ لپتاپ است و در ادامه خواهم گفت که با چه نگاشتی این زمان را به زمانِ واقعی فیلم تبدیل خواهم کرد.

خروجی سلول سوم در تعیین اجسام متحرک:





سلول چهارم:

هدف:تعریف کردن تعدادی تابع مفید جهت استفاده در قسمت های بعد.

حال به معرفی توابع می پردازیم:

- MSE

این تابع دو تصویر(هم سایز) را می گیرد و تفاضل مربعات RGB آن ها را محاسبه کرده و در خروجی می دهد.

```
err = np.sum((imageA.astype("float") - imageB.astype("float")) ** 2)
err /= float(imageA.shape[0] * imageA.shape[1])
```

- Compare_Images

این تابع تفاوت دو تصویر(هم سایز) را می گیرد و تفاوت آن هارا به کمک دو معیار زیر محاسبه می کند و در خروجی بر می گرداند:

  - MSE: همان معیار تفاضل مربعات که در صفحه قبل گفته شد.
  - SSIM: این معیار، معیارِ معتبرتری برای ارزیابی شباهت دو تصویر است که در کد اصلی نیز با تکیه بر این معیار به بررسی شباهت پرداختیم

```
m = mse(imageA, imageB)
s = measure.compare_ssim(imageA, imageB)
```

- sameSizing

همانطورکه می دانید کانتور های مختلف در اسکیل های مختلف موجود اند و تشخیص نزدیکی آن ها زمانی میسر است که هم سایز شده باشند لذا این تابع را نوشتیم تا با interpolation دو عکس را هم سایز کند،و این هم سایزی به سمت سایز بزرگتر انجام می گیرد یعنی دو تصویر نهایی هم سایز با تصویر بزرگترِ ورودی اند.

- findCenterDistance

به کمک این تابع فاصله بین مراکز دو کانتور را محاسبه می کنیم.

توجه کنید که هر کانتور را به صورت یک Tuple سه تایی که دارای سه فیلد زیر است ذخیره کرده ایم و برای محاسبه ی فاصله مراکز دو تصویر نیاز است که ابتدا tuple هارا unpack کرده و سپس مختصات مراکز را یافته(به کمک ابعاد ذخیره شده) و نیز در ادامه به یافتن فاصله ی اقلیدسی دو مرکز می پردازیم.

سلول پنجم:

پیش از بررسی کد ابتدا منظورمان از Tube را مطرح می کنیم و سپس بیان می کنیم که چگونه این Tube هارا استخراج کردیم.

تعریف Tube: Tube ها آرایه ای از لیست هایی از Tuple هایی اند که حرکت یک جسم مشخصِ ثابت را در بازه زمانی t1 تا t2 در مسیر مکانی x1 تا x2 را در بردارد.

نکته مهم: هر درایه از Tube برابر است با یک آرایه ای از Tuple ها که مربوط به یک شیِ مشخص اند.

در مرحله قبل ما لیست frames را ساختیم که هر درایه از آن لیستی از کانتور های تشخیص داده شده در آن فریم از ویدیو است.

1ـ لیست کانتور هایِ فریم n ام را در نظر بگیرید.

2ـ لیست کانتور هایِ فریم n+1 ام را نیز درنظر بگیرید.

3ـ روی کانتور های فریم n+1ام جاروب بزن،هر کانتور از این فریم را با تک تک کانتور های فریم n ام مقایسه کن بدین طریق که:

ابتدا فاصله بین دو مرکز این کانتور هارا بیاب

- اگر این فاصله از یه ترشهولدی کمتر بیشتر یعنی که این کانتور ها انگار مالِ یک Tube نیستند و continue می زنیم.
- اگر این فاصله از ترشهولدی کمتر بود داریم:(مشکوک به هم Tube بودن)
  - ابتدا تصاویر را هم سایز کن.
  - MSE, SSIM دو تصویر را محاسبه کن. (البته ما فقط از SSIM در ادامه استفاده کردیم).
  - اگر معیار SSIM از یه ترشهولدی بیشتر بود یعنی دو تصویر مربوط به یک Tube اند اگر این اتفاق رخداد، کانتورِ فریمِ n+1 ام را به tube ِ کانتورِ فریمِ n ام تزریق می کنیم.

در غیر اینصورت ادامه بده(به مقایسه با باقی کانتور های فریمِ n ام بپرداز).

نکته:ما کانتور های n_ frame ام را به شماره یِ Tube ای که به آن مرتبط اند نشاندار کرده ایم.

- **حال اگر کانتوری در فریم n+1 ام با هیچ کدام از کانتور های فریم n ام هم Tube نبود برایش جداگانه Tube تولید می کنیم.**

مرحله 3 ام را آنقدر انجام می دهیم تا تمامی tube ها ساخته شوند و همه این tube ها به صورت درایه هایی در لیست tubes در کد موجود اند،توجه کنید که هر tube که یک درایه از لیست tubes است به مثابه ی یک لیستی از پک هایی به صورت Tuple هایی سه فیلده(که قبلا نیز مطرح شد) اند.

نکته:در اینجا یک تابع به نامِ getHardCopy تعریف کرده ام که یک deepCopy از آرایه ی Tubes می سازد و برمیگرداند(از آنجا که این آرایه از پر مصرف ترین آرایه های کد هست(و دستخوش تغییرات) نیاز است که یک deepCopy از نسخه اصلی آن داشته باشیم و آن را در mainTubes ذخیره می کنیم(برای مصرف در آینده)).

نمونه خروجی استخراج Tube ها:

```
dist: 240.65119987234635
distIgnorance
dist: 1.0
MSE: 339.5833333333333
detected
dist: 0.0
MSE: 2886.933286351891
dist: 241.11822826157297
distIgnorance
32
dist: 241.11822826157297
distIgnorance
dist: 0.0
MSE: 28.200617283950617
detected
dist: 0.0
MSE: 2376.290118577075
dist: 241.11822826157297
distIgnorance
```

نکته 1: **آن قسمت هایی که در آن ها distIgnorance آماده است یعنی کانتوری از فریم n+1 ام فاصله اش بیشتر از حدِ ترشهولد بود از کانتوری از فریم n و لذا جهت بررسی یکسانیِ tube صرف نظر شد(چون وقتی فاصله اش بین دو فریم آن قدر زیاده یعنی به احتمال بسیار زیاد این دو کانتور مربوط به دو tube مختلف است).**

نکته2:**آن قسمت هایی که checked نوشته شده است یعنی که یکی از کانتور های فریم n+1 ام مشابهت ساختاری خوبی با کانتوری از فریم n ام داشته است و لذا در یک tube با آن قرار میگیرد است(در اصل به tube ای که کانتورِ فریمِ n ام به آن متعلق است تعلق می گیرد).**

بزرگ سلول ششم:

هدف:تزریق کردن **رنگِ کانتور ها و سرعت کانتور ها به کانتور هایِ عکس.**

ابتدا به معرفی منطقِ این بخش می پردازیم و سپس نکات پیاده سازی و تابعیِ آن را ذکر می کنیم:

- ابتدا Tubes هایِ به دست آمده از مرحله قبل را فیلتر می کنیم،به این طریق که اگر Tube هایی اندازه کانتور های درونیشان از ترشهولدی کمتر باشد ایگنور می شوند.
- حال که tubes فیلتر شده است به کمک منطق های زیر به محاسبه سرعت و رنگ می پردازیم:

محاسبه سرعت در فیلم اصلی:(برای هر Tube)

- در هرکدام از Tuple هایِ کانتور، Image,Data,t داریم.
- **کانتور اول و آخر هر Tube را استخراج میکنیم.**
  - **فاصله مکانی بین مرکزِ کانتور اول و آخر را به دست می آوریم.**
  - **فاصله زمانی بین کانتورِ اول و آخر را به دست می آوریم.**

  توجه کنید که در هر کانتور اطلاعات مختصاتی و زمانی آن را ذخیره کرده ایم و محاسبات بالا به کمک این اطلاعات میسر است.
  - حال سرعت Tube را به صورت تقریبی برابر زیر درنظر میگیریم:

$$TubeSpeed = \frac{lastContourPlace - firstContourPlace}{lastContourTime - firstContourTime}$$

حال داریم که این سرعت را در کانتور های این Tube تزریق می کنیم.

نکته مهم: مقدار سرعت را در سمت چپ و وسط عکس می نویسم.

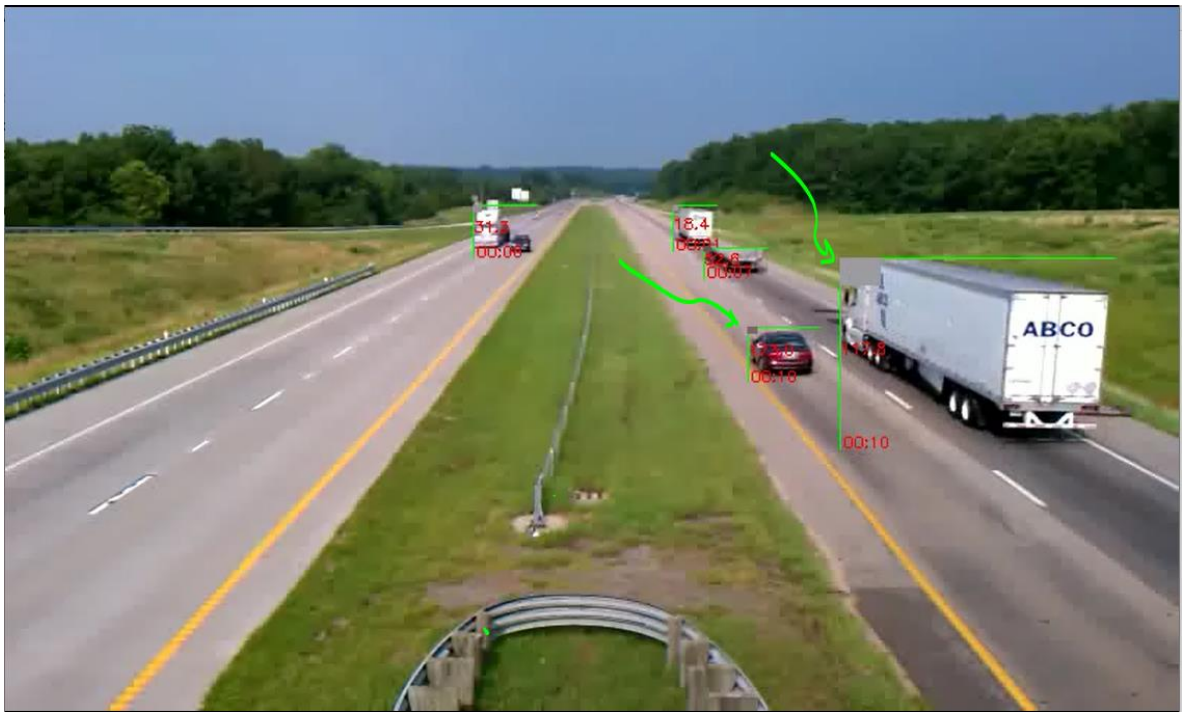
```
cv2.putText(newImage, str(round(speed,1)),
            (0,newImage.shape[0]//2),
              cv2.FONT_HERSHEY_SIMPLEX,0.35, (0, 0, 255), 2)
```

محاسبه رنگِ کانتور ها:

- می دانیم که هر Tube حاوی تعدادی کانتور است،جهتِ تعیین رنگِ Tube،کانتور **وسطِ** Tube را به عنوان نماینده در نظر میگیرم.
- ابتدا آن بخشی از background که کانتورِ نماینده روی آن قرار دارد را استخراج می کنیم.
- سپس جاروب می زنیم روی پیکسل های این دو تصویر(background و کانتورِ نماینده) و
  - ابتدا فاصله رنگی (RGB) دو پیکسل متناظر(یکی در background و یکی در کانتورِ نماینده) را محاسبه می کنیم.
  - اگر فاصله رنگی این دو پیکسل از یه حدی بیشتر بود یعنی که پیکسل کانتورِ نماینده مربوط به جسم متحرک است حال اگر چنین بود مرحله زیر را انجام می دهیم و اگر چنین نبود از این پیکسل صرف نظر می کنیم.
    - پیکسل هایی که فاصله رنگیشان از پیسکل متناظرشان در پس زمینه از یه حدی بیشتر بود را در لیستی append می کنیم.(این لیست با نامِ filteredPixels در کد موجود است).
- پس از پایان این جاروب روی پیکسل ها می رویم سراغ لیست filteredPixels که آماده شده است.
- رنگِ tube را برابرِ میانگین رنگیِ پیکسل های filteredPixels در نظر میگیریم.
- در نهایت رنگ تعیین شده در مرحله قبل را به کلِ Tube تزریق می کنم.
- **رنگ به این شکل به کانتور ها اضافه می شود که percent ای که در تابع تزریق رنگ می دهیم به معنای این است که percent درصد از کل کانتور را از بالا چپ را "رنگی" می کند به رنگ تشخیص داده شده است.این مقدار در صورت وارد نشدن پیش فرض 15 درصد را در نظر می گیرد.**

نمونه ای از خروجی ها:

همه کانتور ها رنگ در بالا سمت چپ مستطیلشان رسم شده است و دو نمونه از این کانتور ها و رنگ هایشان در شکل زیر نشان داده شده است.



نکات:

- رنگ کانتور ها در بالا سمت چپ مستطیل محاط کشیده شده است.
- سرعت جسم متحرک در وسط سمت چپ مستطیل محاط کشیده شده است.
- زمان اصلی حضور در فیلم اصلی نیز در گوشه پایین سمت چپ مستطیل محاط کشیده شده است.

بزرگ سلول هفتم:

هدف:تولیدِ ویدیویِ Brief و تزریق تایم.

ابتدا بخش تزریقِ تایم رو می نویسم:

محاسبه زمان در فیلم اصلی:

- **ابتدا مدت زمان فیلم اصلی را به دست می آوریم.(VideoTotalTime)**
- **نکته: هر کانتور زمانی که تشخیص داده شد را طبقِ زمانِ لپتاپ دارد ☺**
  زمانِ شروعِ process بخشِ تشخیصِ کانتور هارا داریم(تحت عنوان startTime)
  زمانِ پایانِ process بخشِ تشخیصِ کانتور هارا داریم(تحت عنوان endTime)
  حال چنین عمل می کنیم:
  - **ابتدا scale ِ بازه زمانی فیلم به مدت زمان پردازش را محاسبه می کنیم به طریق زیر:**

$$scale = \frac{\text{VideoTotalTime}}{endTime - startTime}$$

- حال مدت زمان کانتور مشخصِ را از startTime محاسبه می کنیم و در اسکیل اندازه گرفته شده ضرب می کنیم و بدین ترتیب زمان محاسبه شده در ویدیو اصلی تولید می شود.
- حال که زمانِ اصلی کانتور ها محاسبه شده است در این کانتور ها،زمان ها را به صورت text ذخیره می کنیم. ذخیره سازی زمان به صورت رو به رو خواهد بود: MM:SS.

```
    time = ('%02d:%02d'%(difference[0],difference[1]))
    cv2.putText(miniImage, time,
                (2, miniImage.shape[0]- 2),
                cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255), 1)
```

حال در صفحه بعد به توضیح نحوه تولید BriefVideo می پردازیم.

تولید ویدیویِ خلاصه شده:

- ابتدا متغیری به نام noOfTubesPerFrame داریم که به این معناست که در هر فریم چه تعدادی Tube قرار بدهیم.بنده به صورت دلخواه این مقدار را برابر 5 درنظر گرفته ام.
- حال در هر مرحله چنین عمل می کنیم:
  - حال از noOfTubesPerFrame تایِ اولِ FilteredTubes، این تعداد Tube برمیداریم و در آرایه frameTubePack می ریزیم و در هر مرحله چنین عمل می کنیم:
    - یک کپی از background می گیریم.
    - از تک تک Tube های موجود در frameTubePack یک عدد کانتور برمیداریم و بر روی background می نویسیم(یک فریم از فیلم خلاصه نهایی به این ترتیب تولید می شود):

      نکته: اگر یکی از Tube های frameTubePack همه اعضایش تمام شود،جایِ این Tube یک tube جدید از لیست tubes برمیدارد و جایش قرار می دهد.
    - حال فریم تولید شده را به لیست Frames تزریق می کنیم.
- در نهایت داریم که لیست Frames فیلمِ خلاصه شدهِ فیلم اولی با نرخِ noOfTubesPerFrame tube بر فریم است.

تی ای محترم می تواند عدد noOfTubesPerFrame را افزایش بدهد و ویدیویِ خلاصه شده با طول کوتاه تری تولید خواهد شد.(در صورت افزایش این عدد هر فریم از ویدیو خروجی تعداد Tube های بیشتری خواهد داشت).

- حال که فریم های ویدیو خروجی را ساخته ایم اکنون این فریم هارا ویدیو می کنیم:
  - یک فایل باز می کنیم.
  - جاروب می زنیم روی فریم های تولید شده و در هر مرحله آن فریم را ذخیره می کنیم.
  - خروجی ویدیو آماده است☺.

چند نمونه از خروجی های این ویدیو:

نکته: متغیر noOfTubesPerFrame را برابر 5 تنظیم کردیم یعنی هر فریم حداکثر 5 کانتور از tube های مختلف را دارا خواهد بود.در ادامه دو فریم از قسمت های مختلف ویدیو را مشاهده می کنید:

نکات:

- رنگ کانتور ها در بالا سمت چپ مستطیل محاط کشیده شده است.
- سرعت جسم متحرک در وسط سمت چپ مستطیل محاط کشیده شده است.
- زمان اصلی حضور در فیلم اصلی نیز در گوشه پایین سمت چپ مستطیل محاط کشیده شده است.

سلول هشتم:

هدف:تشخیص جهت های حرکت و ذخیره سازی آنها،مثلا تشخیص تعداد جهت هایِ شمالغرب به جنوبِ شرق.

مراحل انجام این کار به شرحِ زیر است:

- برای تشخیص راستا حرکت یک Tube به طریق زیر عمل می کنیم:
  - ابتدا نقاط مرکز کانتور های Tube را در می آوریم.
  - روی مرکز نقاط LinearRegression می زنیم و بهترین خط عبوری از این نقاط مرکزی را محاسبه می کنیم.
  - شیب این خط را محاسبه می کنیم و از روی این شیب با رابطه زیر به **زاویه با افقِ راستای حرکت Tube می رسیم**.

```
import math
return int(math.degrees(math.atan((-1)*slope)))
```

- حال که زاویه با افق را داریم کافیست جهت را تعیین کنیم،جهت بدین معنا که الان می دانیم که شی مثلا دراستای شمال غرب ــ جنوب شرق حرکت می کند اما نمی دانیم که آیا از شمال غرب به جنوب شرق می رود یا بالعکس،جهت تعیین این موضوع به بررسی نحوه ی تغییرات x,y مرکز کانتور ها می پردازیم و در اصل تابع giveMeDirection همین کار را می کند و در خروجی رشته یِ جهت را می دهد.

همانطور که می دانید 8 جهت کلی داریم که به شرح زیر اند:

```
NW_SE = "شمال غرب به سمت جنوب شرق"
SE_NW = "جنوب شرق به سمت شمال غرب"
WN_ES = "غرب شمال به سمت شرق جنوب"
ES_WN = "شرق جنوب به سمت غرب شمال"
WS_EN = "غرب جنوب به سمت شرق شمال"
EN_WS = "شرق شمال به سمت غرب جنوب"
SW_NE = "جنوب غرب به سمت شمال شرق"
NE_SW = "شمال شرق به سمت جنوب غرب"
```

حال به کمک if های موجود در تابع giveMeDirection که رویِ نحوه تغییرات x,y اند جهت حرکت را نیز که یکی از 8 مورد بالاست پیدا کرده و در فایلی به ازای هر tube یک مورد ذخیره می کنیم.

(خروجی این فایل در فایلِ perTubeMoveMents.txt ذخیره می شود).

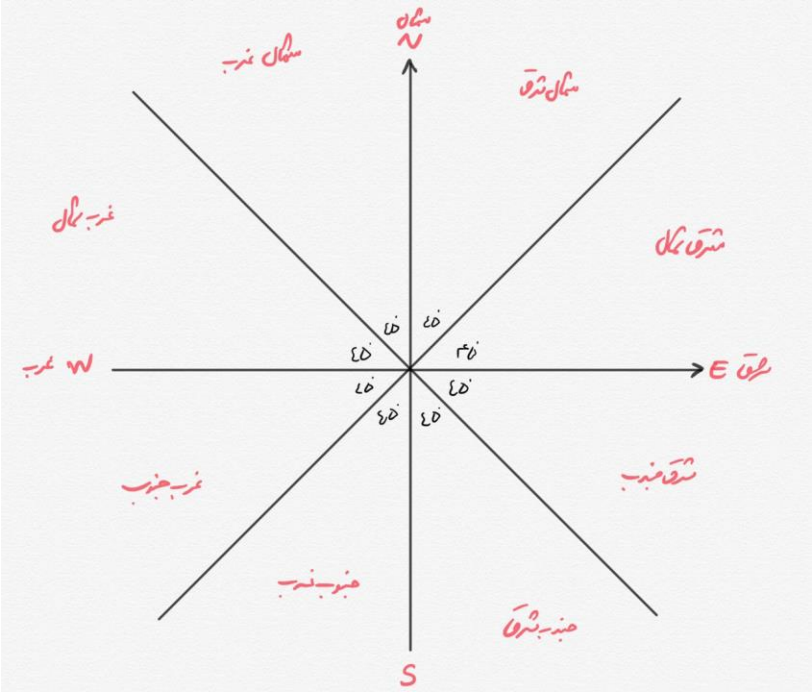
- حال باید روی 8 مورد بالا Clustering بزنیم و تعداد کل tube هایی که در مثلا جهت شمالغرب به جنوبِ شرق است را بیابیم و این کار را کرده و سپس نتیجه خالص خروجی را در فایلِ clusteredMoveMents.txt ذخیره می کنیم.

فایل perTubeMoveMents.txt شامل به ترتیب جهتِ Tube ها است(یعنی اولین جهت نوشته شده بیانگرِ جهتِ اولین Tube است و دومین جهت نوشته شده بیانگر جهتِ دومینِ Tube است و ...)

فایل clusteredMoveMents.txt شامل Clustering شده ی جهت هاست:

```
تعداد شمال غرب به سمت جنوب شرق ها برابر است با: 0
تعداد جنوب شرق به سمت شمال غرب ها برابر است با: 0
تعداد غرب شمال به سمت شرق جنوب ها برابر است با: 0
تعداد شرق جنوب به سمت غرب شمال ها برابر است با: 38
تعداد غرب جنوب به سمت شرق شمال ها برابر است با: 0
تعداد شرق شمال به سمت غرب جنوب ها برابر است با: 32
تعداد جنوب غرب به سمت شمال شرق ها برابر است با: 0
تعداد شمال شرق به سمت جنوب غرب ها برابر است با: 0
```

:☺ #



سلول نهم:

هدف:تولیدِ ویدیویِ GradientLine

مراحل انجام این کار به شرح زیر است:

ابتدا مراحل را برای یک تک Tube می گویم و در نهایت کافیست یک for روی کل tube ها بزنیم:

- ابتدا روی کانتور های هر Tube جاروب می زنیم و مرکز هر کدام را محاسبه می کنیم.
- به کمک LinearRegression بهترین خط عبوری را از مراکز کانتور ها را تعیین می کنیم.
- حال یکبار دیگر روی کانتور ها از ابتدا جاروب می کنیم و خروجی مدلِ regression را برای هر x از مرکز کانتور ها محاسبه می کنیم و تکه تکه مرکز کانتور هارا از ابتدا به هم وصل می کنیم و رنگ هر تکه به گونه ای تغییر می کند که
  - اولا رنگ به صورت تصاعد حسابی رشد می کند از سیاه(0و0و0) تا سفید(255و255و255)
  - ثانیا اگر tube ما دارای n کانتور باشد،داریم که این الگوریتم n-1 تا تکه خط تولید می کند☺ که رنگ هایشان به صورت تصاعدی رشد می کند(و بدین ترتیب Gradient تولید می کند).

موارد بالا جهت رسم GradientLine یک tube مطرح گردید.

حال داریم که با تنظیم متغیر noOfGradientLinesPerFrame تعیین می کنیم که در هر frame از ویدیویِ GradientLine چند عدد از Tube ها GradientLine اش رسم شود.

خروجی در قالب فایل GradientLine.avi ذخیره شده است و تی ای محترم می تواند به سادگی برای دیدن خروجی به آن مراجعه کند.

یک نمونه خروجی:

توجه کنید که مقدار متغیر noOfGradientLinesPerFrame را برابر 4 قرار داده ام یعنی هر فریم ویدیو خروجی دارای 4 GradientLine است.

مشابه همین خروجی ها برای ویدیو 2 نیز اماده شده است و در فایل پاسخ ضمیمه شده است و برای جلوگیری از بیش از حد شلوغ شدن گزارش دیگر آن هارا جدا در فولدری ضمیمه کرده ام.