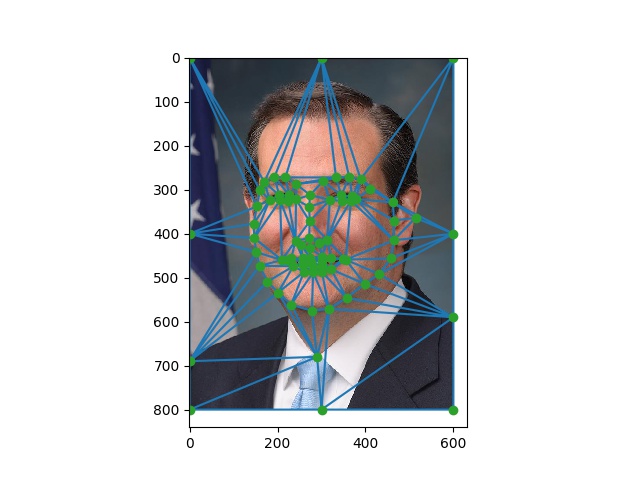
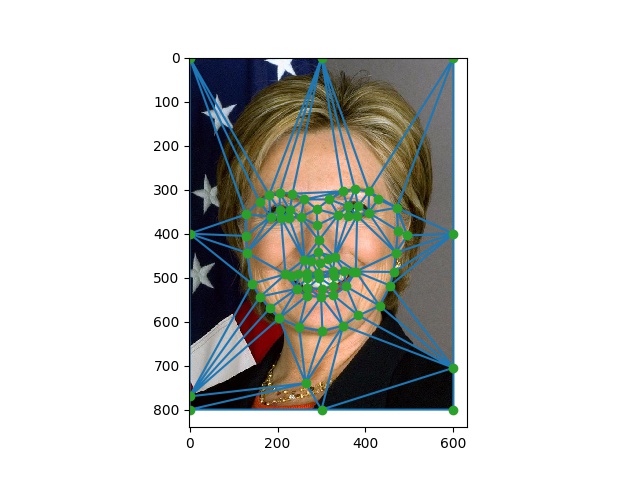
محمد جواد شریعتی سوال شماره دو ۹۶۱۰۰۴۱۴

فایل

 همراه تمامی کدها،‌ یک فایل

ابتدا نقطه ها رو روی تصویر اولی بدست آوردم و سپس متناظر همان نقاط رو در تصویر آخر هم بدست آوردم.

بعد از اون همانطور که در کلاس و صورت سوال گفته شد،‌ از لایبرری Delaunay برای مثلث بندی شکل استفاده کردم. دقت شود که روی یکی از شکل ها(تصویر اول) مثلث بندی را انجام میدهیم و سپس همان را برای هر دو تصویر استفاده میکنم.



مشاهده می شود که چطور مثلث ها در دو شکل متناظر یکدیگر هستند.

خروجی مثلث بندی، به ما میگوید که کدام نقطه ها( با استفاده از اندیس نقاط) با یکدیگر تشکیل مثلث میدهند. به صورت آرایه ای از tuple های سه تایی.

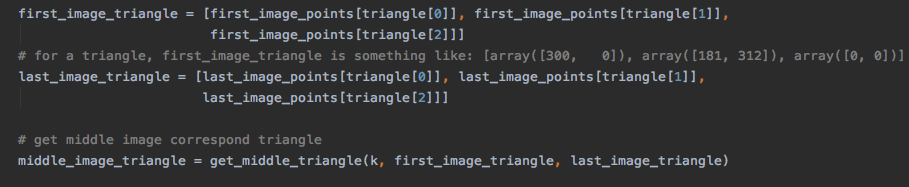
حال نیاز داریم که در بین تصویر اول و آخر تعدادی تصویر میانی بدست آوریم. بدین منظور از یک for استفاده میکنم که در هر مرحله باتوجه به مقدار k تصویر میانی را بدست می آورد و در یک آرایه میریزد. در نهایت با استفاده از این آرایه یک تصویر gif میسازم.

دقت شود که k در واقع همان نسبت k به m است که در توضیحات صورت سوال آمده است.

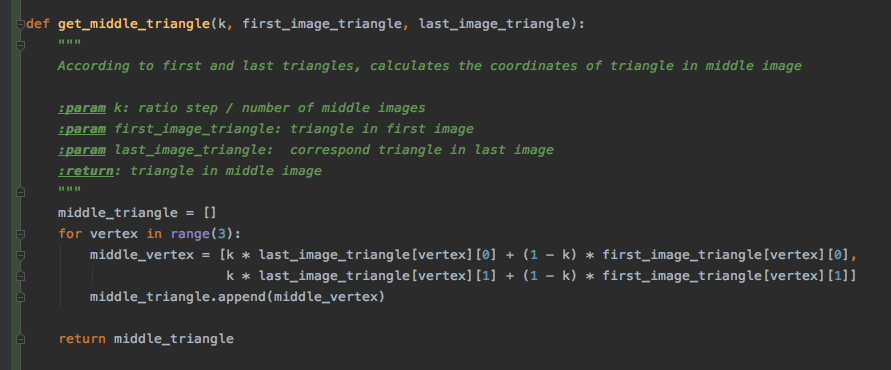
حال توضیح میدهم که پیداکردن تصویر میانی چگونه انجام می شود. این کار را تابع git\_middle\_morphing\_image انجام میدهد که در helper.py پیاده سازی شده است.

ابتدا فقط مثلث ها را وارپ کردم که در ریزالت قسمت هایی خالی (سیاه) وجود داشت. بعد از سرچ کردن در اینتنرنت، فهمیدم که باید با مستطیل ها کار بکنم و در نهایت یک مثلث رو نگه دارم. بدین صورت:

ابتدا مختصات نقاط مستطیل ها را بدست می آورم. (در قسمت قبل بدست آوردیم که هرکدام از مثلث ها از کدامین نقاط هستند(مثلا نقطه اول و بیستم و سی ام) حال میخواهیم مختصات آن ها را بدست آوریم:

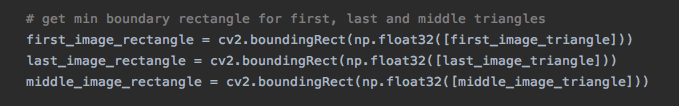


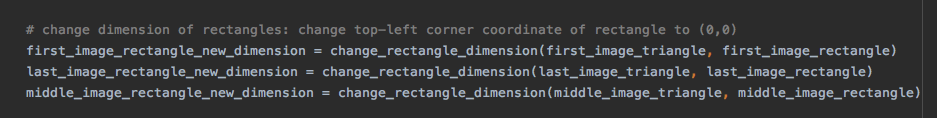
برای نقاط مثلث عکس میانی،‌ یک تابع جدا نوشته ام:



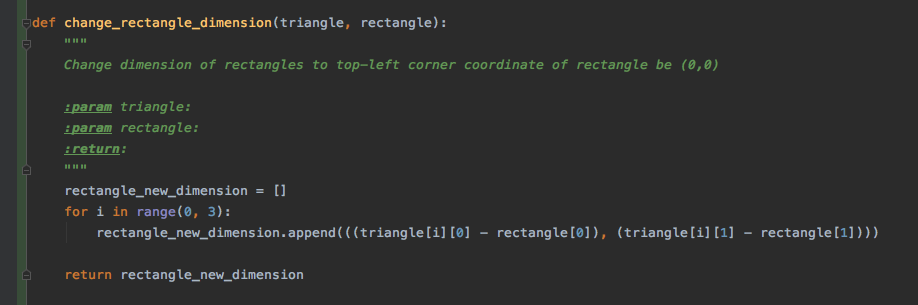
که عملکرد آن همان گونه است که در توضیحات سوال گفته شده است.

حال به ازای هر مثلث، یک مستطیل که محاط به آن است پیدا میکنم. یعنی کوچکترین مستطیلی که مثلث را کاملا در برگیرد:

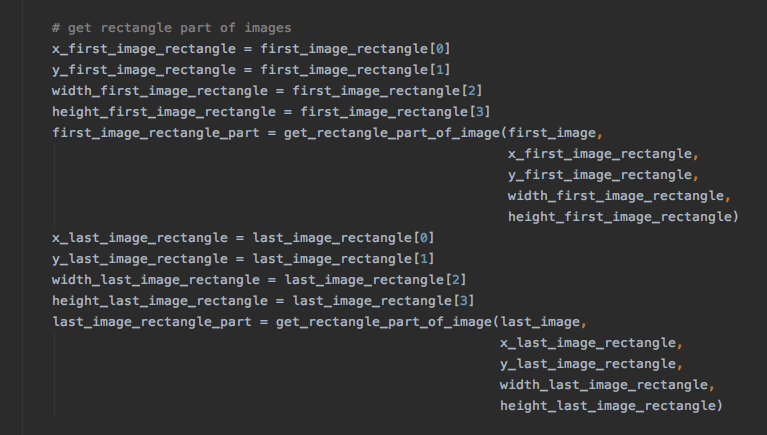


مسئله بعدی این است که مستطیل هایی که در بالا بدست آوردم مختصاتشان مختصات تصویر است. اما برای آنکه راحت تر کار کنیم(برای وارپ کردن نیاز به همان مستطیل به تنهایی داریم و مهم نیست که کجای عکس ما قرار دارد) نقاط مثلث را طوری تغییر می دهیم که نقطه بالا سمت چپ مستطیل (0,0) باشد:

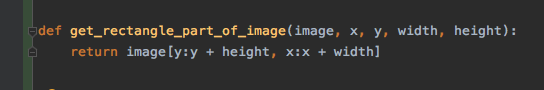
که تابع استفاده شده پایین تر پیاده سازی شده است:



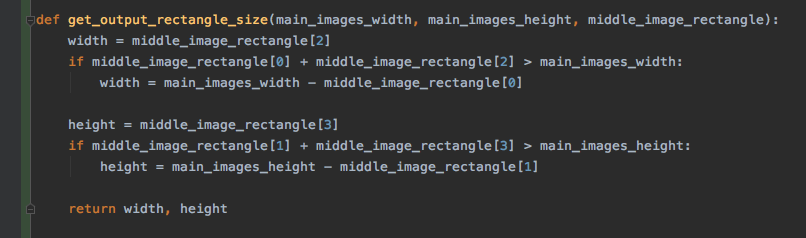
مسئله بعدی کراپ کردن تصویرها برای بدست آوردن مستطیل ها است. (قسمتی از تصویر که مستطیل ما قرار دارد را فقط نیاز داریم):



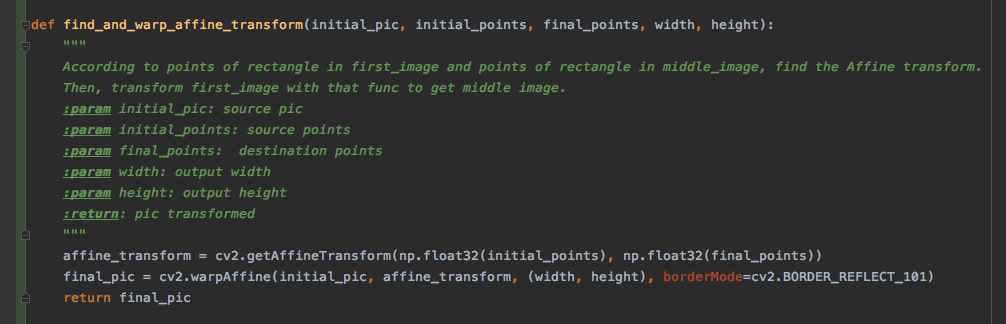
که از این تابع استفاده میکند:



حال میخواهیم تصویر را وارپ کنیم. برای سایز مستطیلی که تصویر وارپ شده به ما می دهد، در ابتدا از سایز خود مستطیل استفاده میکردم. اما در بعضی از مستطیل ها این موضوع باعث ارور میشد. بعد از بررسی فهمیدم که این موضوع وقتی اتفاق می افتد که طول مستطیل بعلاوه x نقطه بالا چپ آن بیشتر از طول عکس اصلی شود(به همین ترتیب برای عرض مستطیل و ارتفاع عکس). پس ابتدا با استفاده از تابع زیر این موضوع رو برطرف کردم:



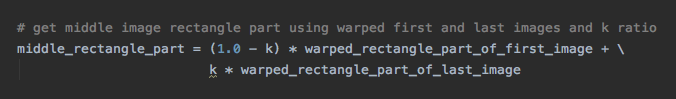
حال باید تابع Affine که بین تصویر اول و تصویر وسط و همینطور بین تصویر آخر و تصویر وسط وجود دارد را پیدا کنیم. تابع Affine درجه آزادی ۶ دارد. و ما هم ۶ مختصات داریم(۳ راس مثلث). بنابراین میتوان پیدا کرد که چه تابعی بوده که باعث شده که مثلث در شکل اول به مثلث در شکل میانی تبدیل شود و سپس این تابع را روی کل مستطیل اول اعمال کرد تا یک مستطیل در شکل میانی به ما بدهد که شامل مثلثی است که به دنبال آن هستیم:

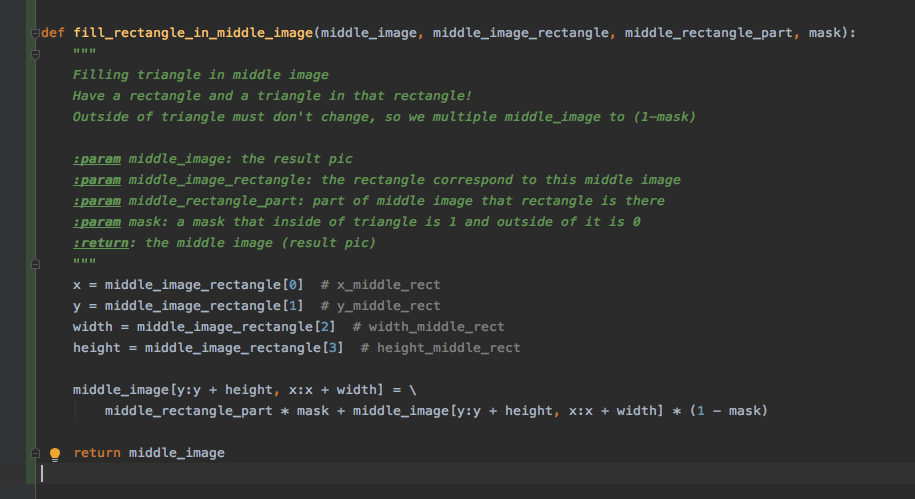


بدین ترتیب ما دو تصویر وارپ شده بدست می آوریم:

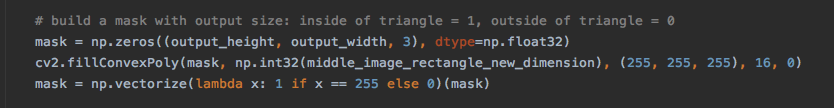


یکی تصویر میانی حاصل اعمال تابع Affine که از تصویر اول به تصویر میانی بدست آوردیم، و دیگری تصویر میانی ای که از اعمال تابع Affine که از تصویر آخر به تصویر میانی بدست آوردیم. حال با استفاده از نسبت K ، این دو تصویر را ادغام میکنم.



حال برای اینکه میخواهیم تنها مثلث را در تصویر میانی جایگذاری کنیم نه کل مستطیل را باید یک ماسک بسازیم که داخل مثلث ۱ باشد و خارج آن ۰. بدین ترتیب میتوانیم بدون تاثیرگذاشتن روی قسمت های خارج مثلث، قسمت مورد نظر را پر کنیم:

برای ساختن ماسک، بهترین روش کشیدن مثلث با رنگ سفید داخل مستطیل است. که بدین منظور کافی است از تابع fillConvexPoly استفاده کنم:



بدین ترتیب با وارپ کردن تک تک مثلث ها، یک تصویر میانی ساخته میشود.

گیفی از تشکیل این مثلث ها با نام triangle\_completion.gif در پوشه out قرار دارد که روند ساخته شدن مثلث ها را در میانی ترین تصویر (k=0.5) نشان می دهد.