به نام خدا

اصول پردازش تصویر

تمرین سری پنجم

استاد درس

دکتر مصطفی کمالی تبریزی

نیمسال اول ۱۴۰۱-۱۴۰۰

نکات تکمیلی :

۱. برای ران کردن کد ها ، نیاز به لایبرری های Numpy ،OpenCV , Scipy و imageio داریم.

Python = 3.8

imageio~=2.9.0

numpy~=1.21.2

scipy~=1.7.1

opencv == 4.0.1

۲. عکس های ورودی را در پوشه resources قرار دهید. (به طور پیشفرض قرار گرفته اند.)

۳. بعد از اجرای هر کد ، نتایج آن در پوشه Result قرار میگیرد. نتایج اصلی در همین پوشه و در صورتی که نتایج دیگری نیز ایجاد شده باشد، در زیرپوشه های دیگری به تفکیک هر تمرین قرار گرفته است.

۴. در لینک زیر نیز میتوانید کلیه نتایج حاصل را مشاهده کنید :

https://mega.nz/folder/H4pzmYpR#oauhlAOyS9V3ZkXCp8IpFQ

۵. نتایج res01 تا res10 در پوشه اصلی لینک بالا قرار گرفته اند. در صورتی که برای تمرینی، نتایج و خروجی های دیگری نیز درست شده باشد، آنها را به تفکیک هر تمرین در زیر پوشه ها قرار داده ام.

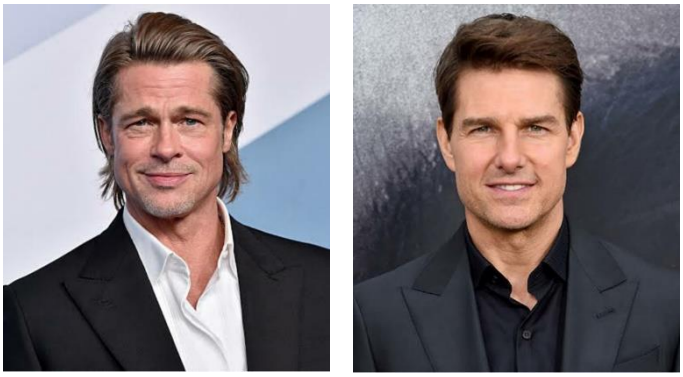
۶. در اکثر تمرین ها، برای انکه نتیجه محاسبات دقیقتر باشد، فرمت عکس ورودی (که uint8 هست) را تبدیل به float میکنم.

مشورت ها:

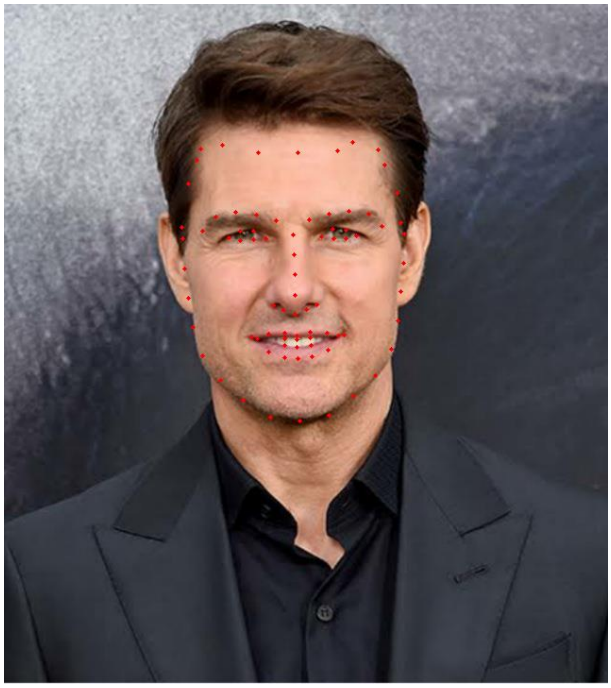
علی شفیعی – مشورت درباره استفاده از ماتریس dia_matrix برای ساخت ماتریس ضرایب سوال دوم

سوال اول)

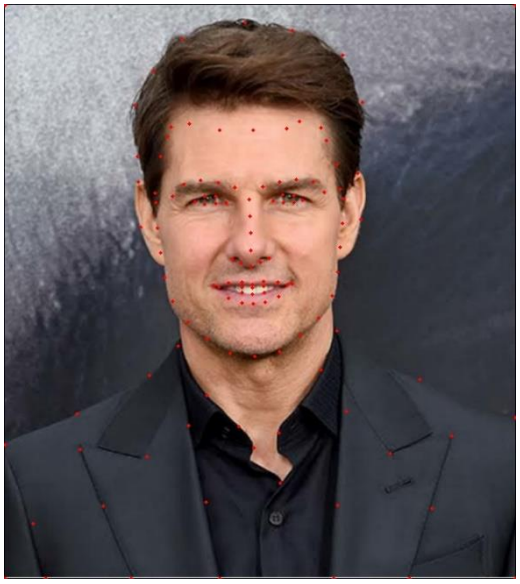
دو عکس زیر را در نظر بگیرید :



از قبل ، با استفاده از لایبرری dlib (در واقع تعمیمی از آن) و یادگیری ماشین، ۸۱ نقطه از صورت هر کدام را بدست می آوریم:

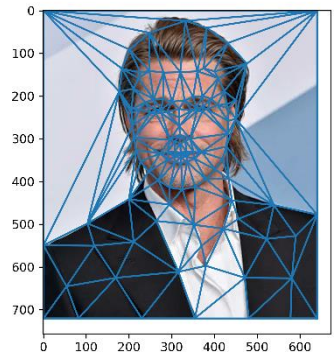


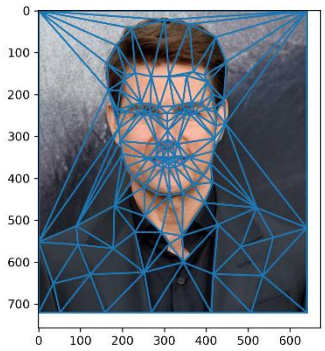
علاوه بر این نقاط، تعدادی نقطه دیگر نیز به صورت دستی انتخاب میکنیم. این نقاط شامل شانه، گردن، گوش، سر، نقاط متناظر روی کت و همچنین چهار نقطه گوشه هر تصویر است :



کل این نقاط را دو فایل txtذخیره میکنیم و برای ران شدن برنامه اصلی از این پس از همین نقاط استفاده میکنیم.

بعد از پیدا کردن نقطه ها. با استفاده از تابع Delaunay از کتابخانه Scipy ، دو عکس را مثلث بندی میکنیم :





حال برای فریم های میانی همچین کاری میکنیم :

از عکس اول یک مثلث با راس های p1,p2,p3 و از عکس دوم، مثلث متناظر با راس های متناظر q1,q2,q3 را در نظر بگیرید.

با استفاده از رابطه ی زیر، مکان راس های مثلث میانگین این دو را در فریم میانی بدست آورید :

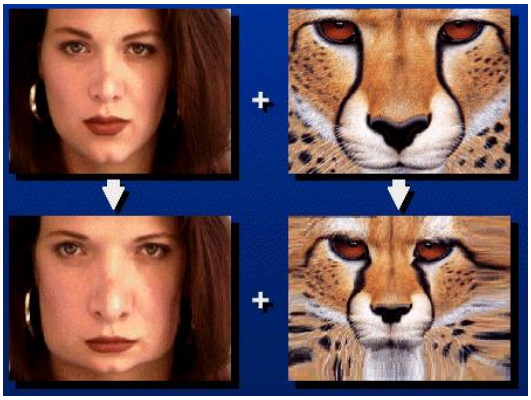
$$x_t = (1 - \alpha) * x_i + \alpha * x_j$$

$$y_t = (1 - \alpha) * y_i + \alpha * y_j$$

حال که مکان راس های مثلث میانیT به دست آمده، به طور جداگانه یکبار مثلث P را به مثلث T وارپ میکنیم(با استفاده از توابع آماده و affain transform) و یکبار نیز مثلث Q را به مثلث P وارپ میکنیم. به این صورت دو مثلث مثل P’ و Q’ بدست می آوریم که شکل آنها مشابه است.(از درونیابی inter_cubic نیز استفاده میکنیم تا نتایج دقیق تری داشته باشیم.)

برای وارپ کردن مثلثی نیز، ابتدا یک BoundingRectدور هر مثلث میکشیم و این قسمت ها را کراپ میکنیم(تا حجم محاسباتمان در ادامه کمتر باشد.) بعد با استفاده از پوش محدب ، یک ماسک مثلثی شکل میسازیم . آنگاه کل مستطیل اول(از عکس اول یا دوم) را به مستطیل دوم(مستطیل فریم میانی) وارپ میکنیم، سپس ماسک مثلثی را اعمال میکنیم تا فقط همان ناحیه مثلثی شکل باقی بماند. و آن را در جای مناسب جایگذاری میکنیم.

حالا اگر به ازای جفت مثلث متناظر از عکس اول و دوم این کار را انجام دهیم. دو عکس جدید بدست می آید، به صورتی که مثلث های متناظر آنها دقیقا شکل یکسانی دارند.(یعنی فرایند شکل زیر را انجام دادیم.)



حال که مثلث های متناظر دو عکس کاملا بر هم منطبق هستند، یک میانگین گیری وزن دار از پیکسل های آن انجام میدهیم تا فریم میانی ساخته شود :

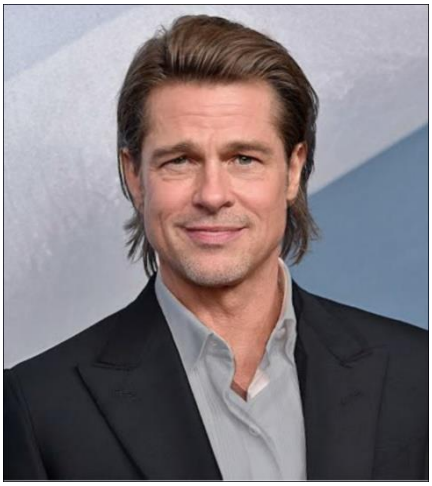
$$M = (1 - \alpha) * I_1' + \alpha * I_2'$$

که I_1' , I_2' همان عکس های تغییر یافته و مثلث-منطبق‌شده‌ی عکس اول و دوم هستند.

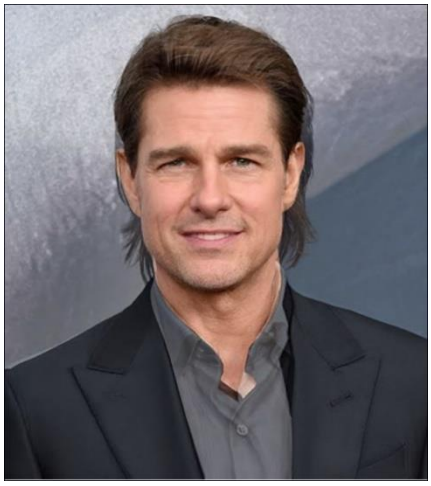
اگر تعریف کنیم $rate = \frac{1}{45}$ و آنگاه $\alpha = i * rate \quad ; \quad i = 0,1,2,\dots,45$

و فرایند گفته شده را به ازای ۴۵ آلفای مختلف انجام دهیم، ۴۵ فریم ساخته خواهد شد. آنها را در یک لیست ذخیره میکنیم، سپس وارون لیست را به خود لیست اضافه میکنیم و آن را به صورت یک ویدیو ذخیره میکنیم.

تصاویر ۱۵ام دنباله :

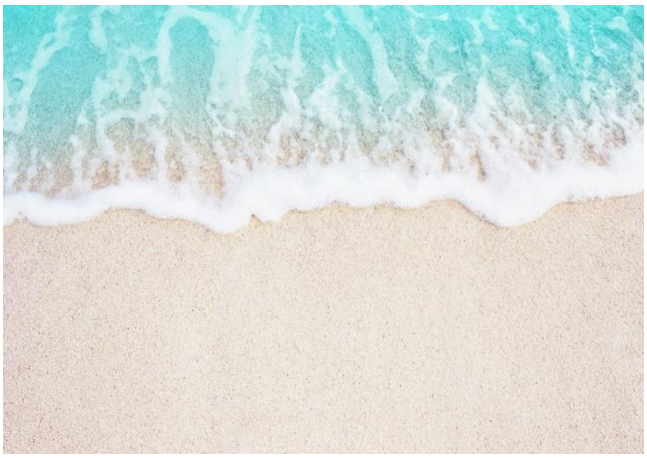


تصویر ۳۰ام دنباله :



سوال دو)

عکس های روبرو را در نظر بگیرید :



و میخواهیم تصویر آدم برفی را در ساحل قرار دهیم. روش کار دقیقا همان روش پواسون بلندینگ توضیح داده شده در کلاس است. ابتدا یک ماسک درست میکنیم.(خارج از برنمه در فوتوشاپ) سپس ماتریس b را میسازیم. این ماتریس، یک ماتریس هم اندازه عکس سورس(آدم برفی) است. برای ساختن آن :

۱. **اگر درایه i , j ماسک، برابر صفر باشد، انگاه مقدار متناظر آن را از عکس مقصد (ساحل) می آوریم.**
۲. **در غیر این صورت، مقدار متناظر را، برابر با مقدار متاظر لاپلاسین عکس سورس(آدم برفی) در نظر میگیریم.**

بعد از ساختن ماتریس b (که مثلا سایز آن h*w است) آن را یک بعدی میکنیم(1 * (h*w)) .

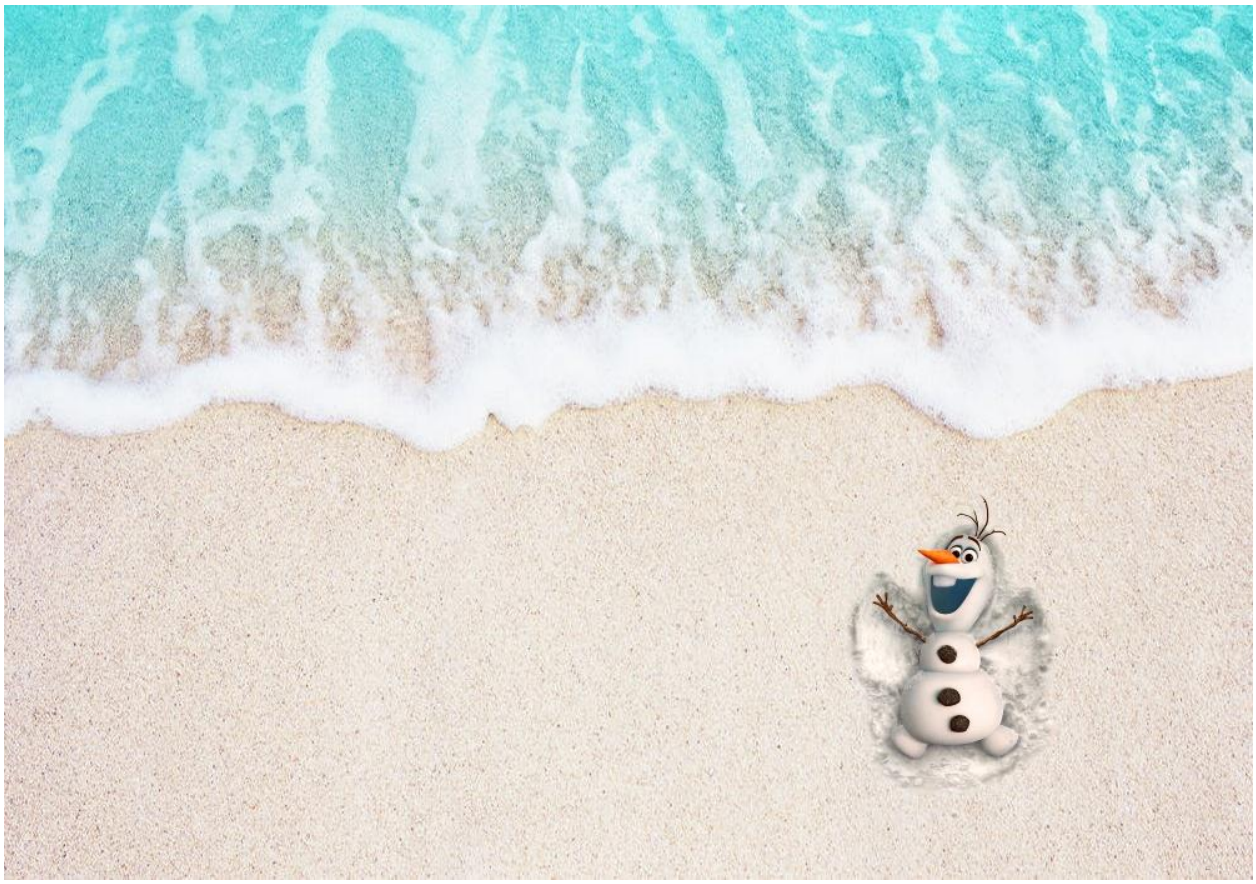
سپس ماتریس ضرایب A را میسازیم. این ماتریس، یک ماتریس ۵ قطری تنک (SPARSE) است. این قطرها عبارتند از : قطر اصلی، قطر بالای اصلی، قطر زیر اصلی، قطر width-فاصله بالای قطر اصلی و قطر width-فاصله پایین قطر اصلی.

۵ لیست متناظر با ۵ قطر در نظر بگیرید. این قطر ها را اینگونه میسازیم که با پیمایش ماسک (فرض کنید ماسک را نیز به صورت یک بردار یک بعدی در میاوریم) اگر درایه i ماتریس ماسک برابر ۰ باشد(یعنی بخواهیم یک مقدار از عکس تارگت انتخاب کنیم.) یک ۱ به لیست قطر اصلی و به بقیه لیست ها ۰ اضافه میکنیم و اگر درایه i صفر نباشد(یعنی بخواهیم از گرادیان یا به طور مشابه از لاپلاسین عکس سورس استفاده کنیم) 4- در قطر اصلی و به بقیه لیست ها ۱ اضافه میکنیم.

با اتمام این مرحله قطر های ماتریس را ساخته ایم.

حال چون میدانیم که ماتریس اسپارس داریم، از کتابخانه scipy و داده ساختار dia_matrix این کتابخانه استفاده میکنیم که یک ماتریس اسپارس را به صورت قطری نگهداری میکند. به دلیل مدل ساخته شدن این ماتریس، ابتدا قطر های خود را در جهت های راست و یا چپ به میزان مورد نیاز شیفت میدهیم، سپس این قطرها را به همراه یک آرایه offset به سازنده dia_matrix میدهیم تا قطر های ما را در نقاط موجود در offset که مشخص کردیم قرار دهد.

بعد از ساختن ماتریس ضرایب (که یک ماتریس (h*w) * (h*w) است) ، دستگاه Ax = b را حل میکنیم(با استفاده از تابع spsolve کتابخانه scipyکه مخصوص حل دستگاه معادلات اسپارس است) سپس نتیجه ی x را که یک ماتریس یک بعدی است تغییر شکل میدهیم تا دو بعدی شود و این ماتریس را در نقطه مورد نظر از عکس اصلی قرار میدهیم. نتیجه به صورت زیر است :



نتیجه اصلی برنامه من، عکس بالا است.

اما نتایج دیگری که روی عکس های دیگر انجام شده نیز در ادامه قرار داده میشود.(این نتایج فرعی هستند و نتیجه اصلی همان عکس بالاست.)

ریسورس این عکس ها نیز در فایل قرار گرفته است اما در حالت کامنت قرار دارند که با خراج کردن از حالت کامنت و ران کردن، این نتیجه ها به دست خواهد آمد.

سوال سوم)

ابتدا برای عکس های ورودی، برای هر کدام یک استک گاوس و یک استک لاپلاسین میسازیم. برای اینکار از یک فیلتر گاوس با کرنل ۴۵*۴۵ و سیگما های مختلف در لول های مختلف استک استفاده میکنیم.

ابتدا عکس I را با سیگما ۲ بلور میکنیم تا G1 بدست بیاید. سپس L1 = I – G1 است را محاسبه میکنیم.

سپس G1 را با سیگما ۴ بلور میکنیم تا G2 بدست بیاید. آنگاه L2 = G1 – G2 را محاسبه میکنیم.

به همین طریق G3 (سیگما ۸) و L3 ، G4 (سیگما ۱۶) و L4 و در نهایت G5(سیگما ۳۲) و L5 را محاسبه میکنیم.

با اینکار ۴ استک با ۵ لول بدست می آوریم.

حال میخواهیم عکس ها را ادغام کنیم. در طبقات بالای استک(یعنی فرکانس های پایین) ادغام قوی تری انحام میدهیم. و هر چه به طبقات پایین هرم (یعنی فرکانس های بالا که جزییات را نشان میدهید) نزدیک میشویم، ادغام ها را یواش تر انجام میدهیم.

در اولین گام ادغام، ابتدا ماسکی که داریم را با یک فیلتر گاوس با سایز (45*45) و سیگما ۳۲ هموار میکنیم. آنگاه یک نتیجه مانند زیر میسازیم :

$$R_1 = mask * G_5^1 + (1 - mask) * G_5^2$$

یعنی در اولین گام، بالاترین طبقه استک های گاوس را با یکدیگر ترکیب میکنیم.

سپس در مراحل بعد، فقط از استک های لاپلاسین استفاده میکنیم. این استک ها حاوی جزییان تصویر مثل ناهمواری ها و مرز ها و ... هستند.در هر مرحله، ابتدا سیگما را نصف میکنیم(تا شدت ادغام را کمتر کنیم) سپس از روی ماسک اولیه یک ماسک جدید هموار شده با کرنل ۴۵*۴۵ و سیگمای جدید(نصف شده) میسازیم(مثلا M' ، یعنی در مرحله ماسک خود را از ماسک مرحله قبل نمیسازیم، بلکه ماسک را از همان ماسک اولیه میسازیم.) و آنگاه

$$R_i = R_{i-1} + \left[M' * L_i^1 + (1 - M') * L_i^2\right]$$

را بدست می آوریم. این حرکت را تا پایین لول هرم ادامه میدهیم و در هر مرحله جزییات بیشتری به عکس اضافه میکنیم تا در نهایت عکس نهایی ساخته شود.

عکس های ورودی :

اگر به طور عادی کیوی را ببریم و در ناحیه مورد نظر بگذاریم :

اگر از الگوریتم گفته شده استفاده کنیم :

چنانچه مشخص است ، مرز کیوی و موز در حالت دوم بسیار *smooth* تر است.

نتایج دیگری که روی عکس های دیگر انجام شده نیز در ادامه قرار داده میشود.(این نتایج فرعی هستند و نتیجه اصلی همان عکس بالاست.)

ریسورس این عکس ها نیز در فایل قرار گرفته است اما در حالت کامنت قرار دارند که با خراج کردن از حالت کامنت و ران کردن، این نتایج به دست خواهند آمد.