توضیح سوال 2

توضیحات اولیه:

1) هر بافت در حدود 1 دقیقه طول کشید تا ساخته شود. این ساخت بافت با اندازه بیشتر بود، ولی چون یک دور دیگه آن را برای هر عکس در یک عکس بزرگتر هم ذخیره کردم تا کنار بافت اصلی قرار بگیرد، زمان اجرا به حدود 2 دقیقه برای هر عکس رسید.
2) من کلی خروجی دیگر به عنوان تصویر حین ساختن بافت میگرفتم که مطمئن شم دارم درست عمل میکنم. میتوانید آن‌ها را در فایل‌های در کنار نتایج مشاهده کنید.
3) برای همه بافتهای انتخاب شده یک سایز بوک و یک سایز برای قسمت مشترک بلوک‌های مجاور در نظر گرفتم. در واقع کد من بسته به این سایزها برای بافتهای با سایز بزرگ خوب کار میکند و از بین بافتهای داده شده نیز روی بزرگ‌ها امتحانش کردم. برای اینکه بتونم از بافت‌های کوچکتر هم بافت بزرگ خوبی بسازم کافیه این تنظیماتو بسته به سایز بافت اولیه و نیز سایز پترن‌های تکراریش عوض کنم.
4) تمام تصاویر را برای نتایج بهتر به فضای LAB بردم و آنجا عملیاتها را روشون انجام دادم و بعد برای چاپ هر کدام به فضای BGRبرگرداندم. البته نتایج در این فضا خیلی تغییر محسوسی نسبت به حالت عادی نداشتند (خصوصا در سوال 3) این شد که دیگه این سوالو در فضای LAB و سوال 3 رو با استفاده از همین سوال ولی در فضای عادی هم تست کردم.

توضیحات پیاده سازی:

همانطور که در کلاس درس گفته شد عمل میکنیم. ابتدا یک بلوک رندوم از بافت داده شده انتخاب میکنیم و در خانه اول بافتی که میخواهیم بسازیم میگذاریم. لازم به ذکر است گاهی مقدار کمی تفاوت در کیفیت عکس نهایی مشاهده میشود که در کل به دلیل استفاده از توابع رندوم است. اگر عکسی که تولید شد به نظرتون قابل

بهبود بود، دوباره برنامه را اجرا کنید. سپس باید سطر اول را پر کنیم که این کار در تابع fill_first_row انجام شده است. در سطر اول، اشتراک بلوکها با هم دیگر تنها از سمت چپ و به صورت یک لایه عمودی است. در هر مرحله یک بلوک برای سمت راست هر بلوک پیشین میبابیم و آن را با یک میزان اشتراکی (یعنی با یک مقدار همپوشانی با بلوک قبل) به قبلی میچسبانیم. این بلوک را طوری انتخاب میکنیم که بیشترین شباهت بین دو بلوک در قسمت اشتراک وجودداشته باشد. برای این کار، قسمت مشترک را با استفاده از یک template matching در کل بافت جستجو میکنیم و سپس از بین شبیه ترین ها یکی را برمیداریم. لازم به ذکر است که همواره شبیه ترین را برنمیداریم چون میخواهیم بافت به طور طبیعی به نظر برسد و مصنوعی جلوه نکند. همچنین سعی میکنیم در نواحی دور از هم و حتی الامکان فاقد اشتراک به دنبال patch مدنظرمان (قسمت اشتراک 2 بلوک که در سر اول میشود بخش سمت راست هر بلوک) بگردیم تا طبیعی تر جلوه کند. پس در نهایت با تابع آماده بخش مدنظرمان را میابیم. از تابع آماده ای برای این کار استفاده شده است که بتواند maskرا هم قبول کند. برای همین از cv2.TM_SQDIFF_NORMED یا cv2.TM_SQDIFF استفاده میکنیم. لازم به ذکر است که هرچی مقدار در اثر خروجی این روشها کمتر باشد، شباهت بیشتر بوده است. همچنین mask به این صورت عمل میکند که تنها بیتهایی از تصویری که میخواهیم آن را جستجو کنیم را مدنظر قرار میدهد که 0 نیستند. برای درستی آزمایی روش خود من ماسکها راهم چاپ کرده ام که جاهایی که مدنظر قرار میگیرند در ماسکها سفید و معادل 255 قرار داده شده اند و جاهایی که قرار است در نظر گرفته نشوند 0 اند. همچنین منطقا ماسک و چیزی که دنبالش میگردیم همواره باید یک سایز داشته باشند. حال باید به طوری این دو ناحیه را بهم بچسبانیم که بیشترین اشتراک را داشته باشند. یعنی باید مرزی که میخواهیم عکس یافته شده از template matching و عکس بلوک اول را درهم فرو کنیم را بیابیم. برای این کار از برنامه نویسی پویا استفاده میکنیم و مسیر با وزن کمینه را میابیم. لازم به ذکر است که منظور از مسیر، مسیر روی آرایه‌ی اختلاف دو تصویر است. در اینجا یافتن اختلاف دو تصویر با استفاده از یک تابع ساده انجام شده است و در واقع مجذور اختلافات هر دو بخش در نظر گرفته شده است. به نظرم نتایجم قابل قبول بود

اما اگر خوب درنمیامدند میتوانستیم برای نتایج بهتر از توابع مثلا گاوسی استفاده کنیم و اینگونه عمل کنیم که هرچی از مرکز شکل دورتر میشویم، اختلافات کم‌اثرتر باشند و در وزن کمتری ضرب شوند و در کل نیز تابع نمایش دهنده اختلافات از یک منحنی گاوسی تبعیت کند. در روش برنامه نویسی پویا برای یافتن مسیر با وزن کمینه یا در حالت ایده آل مسیر با وزن 0، (یعنی وقتی دقیقا دو تو بلوک منطبق میشن) باید در هر مرحله ببینیم به کدوم نود بریم کمترین وزنو در نهایت داریم. یعنی در هر لایه برای هر نود از بین چند همسایه مجاور آن نودی را انتخاب میکنیم که کمترین وزن را به مسیر اضافه کند. به همین ترتیب به عقب برمیگردیم تا درنهایت در لایه آخر بین همه نودها نودی را انتخاب میکنیم که کمترین مقدار را دارد و این کمترین مقدار در واقع کمترین وزن کل مسیر بوده است چون در هر مرحله وزن آن نود به وزن کل مسیر افزوده ایم و بعد به لایه بعدی رفته ایم. سپس از نود نتیجه شروع کرده و جلو میرویم (مسیر را نگه داشته ایم) و روی مرز حرکت میکنیم. برای مثال نواحی روی مرز نیز جزو حالت اول در نظر گرفته میشوند. در واقع خروجی این تابع min_cut این است که یک آرایه برمیگرداند که مسیر را مشخص میکند و هر جا که مرز است را 0 میدهد. این آرایه را یک بار به اندازه سایز کل بلوک و یک بار به اندازه ای که هست برگرداندم (زیرا برگرداندن کل بلوک منطقی تر و راه دست تر بود چون اینجوری همه چا همیشه مسیر در یک بخش 120* 120 قرار داشت مثلا، اما جاهایی هم لازم بود خود مسیر را به طور عمودی یا افقی داشته باشم دقیقا بنابراین خودش را هم برگرداندم). بعد از خروجی این تابع، به این صورت نودها را انتخاب میکنیم که برای سمت چپ مرز پیکسلهای بلوک اول و برای سمت راست مرز پیکسلهای بلوک دوم (بلوکی که با مچ کردن یافتیم) را انتخاب میکنیم و از این مرز دو بلوک را تو هم میکنیم. همین کار را برای کل سطر اول میکنیم.

سپس برای کل سطرهای دیگر، یک تابع داریم که در همان اول بلوک اول هر سطر را انتخاب و جاگذاری کرده است. این کار با استفاده از یافتن شباهت بلوک بالایی و بلوک پایینی انجام میشود. پس همه چیز مانند گذشته است و تنها تفاوت این است که بلوکهای اشتراک به جای عمودی افقی اند. پس آن ها را ترانهاده میکنیم تا

بتوانیم از کدهای قسمت قبل مانند کد یافتن مسیر بهینه استفاده کنیم. به طور مشابه ستون اول نیز پر شد. این کارها در تابع fill_first_block_of_other_rows انجام شده است.

حال به پر کردن بلوکهای مرکزی در تابع میپردازیم. اشتراک این بلوکها با بلوکهای چپ و بالای خود است پس یک ناحیه L برعکس شکل را برای مچ کردن به تابع میدهیم و این برای برای یافتن مرز، هم مرز عمودی هم افقی را در نظر میگیریم. برای این کار خروجی دو تا مرز را (یعنی مرز عمودی و ترانهاده مرز افقی) بیت به بیت اند میکنیم تا کل مرز را داشته باشیم. حال در تمام قسمتهای مشکی ماسک بدست آمده، از تصویر قبلی استفاده میکنیم (یعنی از patch) و در تمام قسمتهای بدست آمده از ماسک که سفیدند از تصویر یافته شده با مچ استفده میکنیم. این کارها در تابع اصلیم یعنی تابع fill_other_parts_of_other_rows انجام شده است. در نهایت در تابع make_textures، 3 مرحله گفته شده در بالا به ترتیب انجام شدند. در قسمت انتهایی این تابع برای هر بوک، مختصات بلوک بالا و چپ آن را میدهیم تا برای آن بلوک جدید، بر حسب بلوکهای بالا و راستش ناحیه مشابه انتخاب شود و در نهایت به تصویر اصلی اضافه گردد.

تصویر ساخته شده برای هر 4 ورودی همواره بزگتر از 2500 در 2500 بوده که در انتها از آن 2500 تاش را جدا کردم. در main برنامه هم برای هر 4 تصویر گفته شده تابع make_textures را صدا زدم.