در این بخش چند تابع پیاده سازی شده که ابتدا توضیحشان میدهیم.

تابع `minCut` :

ورودی ها: 1) دو نوار از دو پچ کنار هم که میخواهیم روی آن برش بزنیم

خروجی ها: 1) مختصات افقی نقاط روی این برش با حفظ ترتیب

عملکرد: در این تابع تنها برش را برای دو پچی که میخواهیم در راستای افقی ادغام کنیم میابیم. یعنی یک پچ را سمت چپ و پچ دیگر را سمت راست فرض میکنیم که قرار است نواری از قسمت سمت راست پچ چپی و نواری از قسمت سمت چپ پچ راستی بر روی هم قرار گیرد و برشی روی آن زده شود. این تابع در واقع این دو نوار از این دو پچ را بعنوان ورودی میگیرد که قرار است پس از ادغام تبدیل به نوار مشترک شود. میدانیم برای آنکه دو پچ در این نوار با هم همپوشانی داشته باشند، باید برشی روی نوار مشترک زده شود که هر طرف این برش متعلق به یکی از این دو پچ باشد و این برش باید بهگونه ای باشد که در مرز آن، دو نوار بسیار بهم شبیه باشند تا برش ما از دید ناظر محسوس نباشد. برای این کار مجموع مجذور تفضلات پیکسل‌های این دو نوار را محاسبه میکنیم که همان `ssd` است. سپس بهترین برش، برشی است که از پیکسل های با مقدار کمتر تصویر `ssd` عبور کرده باشد. همچنین درجه‌ی آزادی‌ای میتوانیم برای برش تعیین کنیم که در اینجا `df=2` در نظر گرفته شده. این یعنی برای برش از بالا تا پایین نوار `ssd`، از هر سطر به سطر دیگر، از `2-` تا `2+` واحد میتوانیم به چپ یا راست حرکت کنیم و به پایین بیایم. پس برای شروع کمینه‌ی انرژی کل را بینهایت قرار میدهیم و در متغیر `minTotalE` ذخیره میکنیم و همچنین آرایه‌ای خالی که قرار است شامل مختصات افقی نقاط روی برش باشد را در لیست `bestCut` نگه میداریم. لیست `bestCut` تنها شامل مختصات افقی نقاط روی برش است یعنی تنها یک عدد در هر درایه از این لیست قرار دارد چون نقاط روی برش به ترتیب از بالا تا پایین در این لیست قرار میگیرند و در هر سطر از تصویر `ssd` تنها یک نقطه انتخاب میشود پس نیازی به نگه داشتن شماره‌ی سطر پیکسل ها نداریم زیرا به وضوح به ترتیب از عدد `0` تا ارتفاع `ssd` منهای یک خواهد بود. اگر بعنوان مثال درایه‌ی `i` ام `bestCut` برابر با `j` باشد به این معنی خواهد بود که نقطه‌ی `[i,j]` از تصویر `ssd` در برش حضور دارد. سپس میدانیم برای شروع برش دادن باید از یکی از پیکسل های سطر اول تصویر `ssd` شروع کنیم. پس با یک `for` روی تعداد حالت های نقطه‌ی شروع کار را آغاز میکنیم. فرض کنید نقطه‌ی `j` ام از سطر اول بعنوان نقطه‌ی شروع انتخاب شده باشد. انرژی مسیر برشی که با ادامه دادن از این نقطه خواهیم داشت را در متغیر `totalE` نگه میداریم که مقدار اولیه‌ی آن برابر با مقدار انرژی نقطه‌ی شروعمان است. همچنین مختصات افقی نقاطی که با

فرض شروع از نقطه‌ی j، روی این برش قرار میگیرند را در متغیر pixelsList نگه میداریم و مختصات افقی نقطه‌ای که در آن حضور داریم را در متغیر jCur نگه میداریم که در شروع برابر با j یا همان نقطه‌ی شروعمان است و آن را به pixelsList اضافه میکنیم. سپس با یک for دیگر روی سطرهای تصویر ssd حرکت میکنیم. فرض کنید در حال انتخاب نقطه‌ی برش روی سطر i ام هستیم. ابتدا پنجره‌ای به اندازه‌ی بازه‌ی [-df,+df] و به مرکز jCur از سطر i+1 میسازیم که در اینجا df=2 فرض شد. این پنجره به طول 2*df+1 را در متغیر window ذخیره میکنیم که نشان دهنده‌ی نقاط ممکن در سطر بعدی است که میتوانیم به آن ها برویم. دقت کنید که در لبه‌های نوار، نباید بگذاریم این پنجره خارج از ssd بیفتد یعنی مینیمم مختصات افقی این پنجره باید برابر با 0 و ماکزیمم مختصات افقی این پنجره باید برابر با پهنای ssd باشد تا out of range نشود. حال از بین عناصر داخل window آن که مقدار یا انرژی کمتری دارد را انتخاب میکنیم و به آن نقطه حرکت میکنیم. انرژی آن را به totalE اضافه کرده و مختصات افقی فعلی یا همان jCur را به آن نقطه انتقال میدهیم و مختصات افقی نقطه‌ی حاصل را به لیست pixelsList اضافه میکنیم. پس از رسیدن به سطر آخر و اجرای کامل این حلقه، انرژی کل مسیر را با فرض شروع از نقطه‌ی j داریم. آن را با minTotalE مقایسه میکنیم. اگر کمتر بود minTotalE و bestCut را بروزرسانی میکنیم. پس از اجرای کامل حلقه‌ها و بررسی تمام نقاط شروع لیست bestCut را بعنوان خروجی بازگشت میدهیم.

تابع merge :

ورودی ها:
1) دو پچ با ابعاد یکسان که میخواهیم آن دو را ادغام کنیم
2) پهنای نوار مشترک

خروجی ها:
1) پچ حاصل پس از ادغام

عملکرد: در این تابع تنها دو پچ را در راستای افقی ادغام میکنیم. یعنی یک پچ را سمت چپ و پچ دیگر را سمت راست قرار میدهیم بطوری که این دو پچ در نواری به ضخامت پهنای نوار مشترک یا همان bandWidth همپوشانی داشته باشند و سپس باید روی نوار مشترک برشی را بیابیم که سمت چپ این برش متعلق به پچ سمت چپ و سمت راست این برش متعلق به پچ سمت راست باشد. پس نواری به ضخامت پهنای نوار مشترک یا همان bandWidth از سمت راست پچ چپی برش میدهیم و در متغیر imgCut1 قرار میدهیم و سپس نواری به ضخامت bandWidth از سمت چپ پچ راستی برش میدهیم و در متغیر imgCut2 قرار میدهیم. میدانیم این دو نوار باید یکدیگر را بپوشانند و روی هم

بیفتند و توسط برشی تعیین شود که هر کدام از این دو نوار، کدام قسمت از نوار مشترک را پر کنند. بهترین برش برشی است که دو طرف آن شبیه باشند تا به نظر نیاید که برشی زده شده است. برای یافتن بهترین برش، دو نوار `imgCut1` و `imgCut2` را به تابع `minCut` که بالاتر تعریف شد میدهیم و لیستی از مختصات افقی نقاط روی خط برش بدست میاوریم که طبق توضیحات داخل بخش تابع `minCut`، دارای ترتیب اند. اگر ارتفاع و پهنای پچ‌های داده شده به ترتیب `hei` و `wid` باشد، آنگاه پچ ادغام شده‌ای با ارتفاع و پهنای `hei` و `2*wid-bandWidth` باید بعنوان خروجی بازگشت داده شود. پس در ابتدا تصویری سیاه بنام `imgMerged` با این ابعاد میسازیم. سپس باید با یک `for` بر روی تعداد سطرهای این تصویر حرکت کنیم و هر سطر را پر کنیم. برای این کار میدانیم در هر سطر یک نقطه وجود دارد که بر روی خط برش قرار دارد. پس سمت چپ آن نقطه را با تصویر چپی و سمت راست آن نقطه را با تصویر راستی پر میکنیم. همچنین در خود آن نقطه از مرز، میانگین دو تصویر چپی و راستی را قرار میدهیم. پس از اجرای کامل این `for` تمام سطرها مقدار دهی میشوند. اکنون برای آنکه مرز این برش واضح نباشد یک همسایگی `5×5` طرفین هر نقطه از خط برش را با کرنل `(3,3)` `blur` میکنیم. اکنون پچ ادغام شده همان `imgMerged` است که بعنوان خروجی بازگشت میدهیم.

تابع `initiate` :

ورودی ها:
1) تصویر اصلی
2) ارتفاع و پهنای پچ

خروجی ها:
1) پچی بصورت رندوم

عملکرد: این تابع پچی به ابعاد داده شده را به صورت رندوم از عکس اصلی داده شده برش میدهد و بعنوان خروجی بازگشت میدهد. برای این کار کافیست مختصات نقطه‌ی گوشه بالا سمت چپ این پچ به صورت رندوم انتخاب شود. فقط این نقطه نباید به گونه ای انتخاب شود که برش دادن پچی به ابعاد داده شده، از آن نقطه ناممکن شود و out of range شود. پس از انتخاب نقطه‌ی بالا سمت چپ، از همان نقطه به ابعاد داده شده میبریم و پچ حاصل را بعنوان خروجی بازگشت میدهیم.

تابع `findBestPatch` :

ورودی ها:
1) تصویر اصلی
2) عکس پچ سمت چپ
3) پهنای نوار مشترک

خروجی ها:
1) بهترین عکس پچ سمت راست

عملکرد: در این تابع پچ مناسب را تنها در حالت ادغام افقی میابیم یعنی فرض شده که پچی در سمت چپ وجود دارد و میخواهیم بهترین پچ ممکن را برای سمت راست این پچ پیدا کنیم که پس از ادغام خوب بنظر آیند. برای این کار نواری به پهنای داده شده از قسمت سمت راست پچ چپی را برش میدهیم و کافیست نواری مشابه آن را در تصویر اصلی بیابیم و سپس پچی از تصویر اصلی را برش دهیم که نوار یافت شده، در قسمت سمت چپ این پچ قرار گرفته باشد که این پچ، پچ مطلوب است و میتواند سمت راست پچ داده شده قرار گیرد. پس به اندازه‌ی پهنای نوار مشترک از سمت راست پچ چپی برش داده و به روش `cv2.TM_CCOEFF_NORMED` و با تابع آماده‌ی `cv2.matchTemplate` نقاطی از تصویر اصلی را که شبیه به نوار هستند را میابیم و خروجی آن را در `imgTmp` نگه میداریم که ماتریسی دو بعدی با درایه‌های بین صفر و یک است. به اندازه‌ی پهنای خالص پچ داده شده از سمت راست `imgTmp` برش میدهیم. پهنای خالص یعنی پهنای پچ منهای پهنای نوار مشترک. دلیل این کار این است که ما پس از پیدا کردن نوار مشابه، قرار است به اندازه‌ی ابعاد پچ داده شده از تصویر اصلی پچی را برش بزنیم که نوار یافت شده، نوار سمت چپ این پچ باشد. پس برای اینکه این برش out of range نشود، در واقع با برش دادن `imgTmp` انتخاب ناحیه‌ای که نمیتوان از آن پچی با ابعاد گفته شده را برش داد را غیرممکن میکنیم. حال از بین نقاط باقی مانده از `imgTmp` آنهایی که مقدارشان با ماکزیمم مقدار ماتریس `imgTmp` کمتر از `0.1` اختلاف دارد را کاندید میکنیم و از بین آنها رندوم یکی را انتخاب میکنیم و از آن نقطه به اندازه‌ی پهنا و ارتفاع پچ اولیه برش میدهیم که این پچ حاصل بهترین پچ است که میتواند سمت راست پچ داده شده در ورودی قرار گیرد. زیرا نوارهای کناریشان شبیه به یکدیگر است. پس این پچ را بعنوان خروجی بازگشت میدهیم.

تابع `findBestPatch2` :

ورودی ها:
1) تصویر اصلی
2) عکس پچ سمت چپ و بالا
3) پهنای نوار مشترک

خروجی ها:
1) بهترین عکس پچ میانی

عملکرد: در این تابع میخواهیم پچی را بیابیم که نوار سمت بالای آن شبیه به نوار سمت پایین پچ بالایی باشد و همچنین نوار سمت چپ آن شبیه نوار سمت راست پچ چپی باشد. مشابه کارهایی که در تابع قبلی کردیم را انجام میدهیم با این تفاوت که ابتدا نوار سمت راست پچ چپی را به تابع آماده‌ی `cv2.matchTemplate` میدهیم و خروجی را در متغیر `imgTmpL` ذخیره کرده و سپس نوار سمت پایین پچ بالایی را که فاقد مربع کوچک مشترک قسمت ⌞ شکل میباشد را به تابع آماده‌ی `cv2.matchTemplate` میدهیم و خروجی را در متغیر `imgTmpU` ذخیره میکنیم. سپس دو تصویر حاصل را به دلیل ذکر شده در بخش قبل برش میدهیم تا out of range رخ ندهد. در واقع با برش دادن این دو تصویر، انتخاب ناحیه‌ای که نمیتوان از آن، پچی با ابعاد گفته شده را برش داد را غیرممکن میکنیم. اکنون میخواهیم از `imgTmpL` و `imgTmpU` میانگین بگیریم ولی اندیس‌ها و ابعاد این دو عکس تفاوت دارند. دلیلش هم این است که یک نوار شامل مربع کوچک مشترک میباشد ولی دیگری نه. پس تصویر `imgTmpU` باید از چپ به اندازه‌ی `bandWidth` برش داده شود. حال میانگین دو تصویر حاصل را محاسبه میکنیم و در متغیر `imgTmp` نگه میداریم. مشابه قبل از بین نقاط باقی مانده از `imgTmp` آنهایی که مقدارشان با ماکزیمم مقدار ماتریس `imgTmp` کمتر از `0.1` اختلاف دارد را کاندید میکنیم و از بین آنها رندوم یکی را انتخاب میکنیم و از آن نقطه به اندازه‌ی پهنا و ارتفاع پچ اولیه برش میدهیم که این پچ حاصل بهترین پچ است که میتواند از بین دو پچ داده شده در ورودی قرار گیرد. زیرا نوارهای کناریشان شبیه به یکدیگر است. آن را بعنوان خروجی بازگشت میدهیم.

تابع `synthesis` :

ورودی ها:
1) تصویر اصلی
2) ابعاد خالص پچ ها
3) ابعاد تصویر نهایی
4) پهنای نوار مشترک

خروجی ها:
1) تصویر نهایی

عملکرد: هدف این است که پچ‌هایی از تصویر اصلی برش دهیم و کنار هم قرار دهیم تا تصویری به اندازه‌ی تصویر نهایی که از ما خواسته شده حاصل شود. در ابتدا تعداد پچ‌هایی را که باید کنار هم قرار گیرند را با تقسیم ابعاد تصویر نهایی بر ابعاد پچ، بدست میاوریم و در متغیر های `heiCnt` و `widCnt` قرار میدهیم که اولی تعداد پچ های لازم در راستای عمودی و دومی تعداد پچ های لازم در راستای افقی است ولی چون در این سوال ابعاد تصویر نهایی و ابعاد پچ هردو مربعی اند، مقدار این دو متغیر یکی خواهد بود. در این تابع ابعاد خالص پچ ها به ما داده شده. خالص یعنی بدون در نظر گرفتن نوار مشترک. ولی ما به

اندازه‌ی پهنای نوار مشترک یا bandWidth به ابعاد پچ اضافه میکنیم که این باعث میشود پس از کنار هم قرار دادن پچ ها، همپوشانی ای بین آنها رخ دهد که این همپوشانی را با تابع minCut که بالاتر تعریف شد، باید به بهترین شکل برش دهیم تا پچ ها ادغام شده به نظر آیند. برای شروع تصویر سیاهی را به ابعاد تصویر نهایی بعلاوه‌ی پهنای نوار مشترک میسازیم و در متغیر imgSyn ذخیره میکنیم و پس از پر کردن آن با پچ ها نوار سمت راست و پایین آن را جدا میکنیم و حاصل قرار است بعنوان خروجی یا همان عکس نهایی بازگشت داده شود. در ذهن خود یک grid بندی ای برای imgSyn متصور شوید که ابعاد هر خانه‌اش، برابر با ابعاد خالص پچ ها باشد و در کل به تعداد widCnt×heiCnt خانه در این گرید بندی وجود دارد. هر پچ باید به طور خالص در این گرید قرار گیرد. ولی از آنجا که نواری به آن اضافه شده، در واقع به اندازه‌ی پهنای این نوار از سمت راست و پایین اضافاتی را به گرید های همسایه‌ی خود وارد میکند. خانه‌ی [0,0] این گرید بندی را با پچی رندوم پر میکنیم. این پچ را با تابع initiate که بالاتر تعریف شد بدست میاوریم. اکنون میخواهیم سطر صفرم این گرید بندی را پر کنیم. با یک for از 1 تا تعداد خانه‌های سطر گرید بندی انجام میدهیم. فرض کنید در حال پر کردن خانه‌ی i ام گرید بندیمان باشیم. آنگاه خانه‌ی چپی اش یا همان i-1 ام را به همراه نوارهای اضافی اش انتخاب کرده و در متغیر ptchL نگه میداریم. سپس بهترین پچی که میتواند سمت راست این قرار گیرد را با استفاده از تابع findBestPatch که بالاتر تعریف شد بهترین پچی که میتواند در خانه‌ی i ام قرار گیرد تا نوار سمت چپش شبیه نوار سمت راست پچ خانه‌ی i-1 ام شود را پیدا میکنیم و در متغیر ptchR نگه میداریم. سپس این دو پچ را با تابع merge که بالاتر تعریف شد، با یکدیگر ادغام میکنیم و نتیجه‌ی حاصل را در خانه‌ی i-1 ام و i ام قرار میدهیم .چون تابع merge دو پچ مربعی را ادغام میکند و به یک مستطیل بزرگ تبدیل میکند و بعنوان خروجی بازگشت میدهد پس دو خانه از گرید بندیمان را باید به خروجی این تابع اختصاص دهیم که البته به اندازه‌ی bandWidth از سمت راست و پایین اضافاتی را به گرید های همسایه‌ی خود وارد میکند. دلیل اینکه تصویر imgSyn را علاوه بر ابعاد اصلی اش بعلاوه‌ی bandWidth کردیم، اینجا مشخص میشود. زیرا هنگام پر کردن خانه‌ی اخر سطر اول، اضافاتی باید به سمت راست و پایین گرید های همسایه وارد شود و برای آنکه out of range رخ ندهد این کار انجام شد. حال پس از پر شدن کامل سطر اول، ستون اول را میخواهیم پر کنیم که به صورت مشابه با پر کردن سطر اول انجام میشود. فقط با این تفاوت که توابع استفاده شده مانند findBestPatch و merge برای حالت افقی پیاده سازی شده اند. یعنی برای دو پچ که سمت راست و چپ هم قرار میگیرند تعریف شده اند. برای آنکه بتوانیم از همان ها استفاده کنیم هنگام یافتن پچ مناسب برای خانه‌ی j ام گرید بندی، پچ خانه‌ی j-1 و همچنین عکس اصلی را 90 درجه در جهت موافق دایره مثلثاتی میچرخانیم و استفاده از تابع

findBestPatch بهترین پچ را میابیم و نتایج مانند ادغام افقی دو پچ اند. ولی پس از ادغام یک مستطیل افقی داریم که برای آنکه به حالت اول بازگردد خروجی تابع merge را 90 درجه در جهت مخالف دایره مثلثاتی میچرخانیم و درون imgSyn در جای مناسبش قرار میدهیم. اکنون سطر صفرم و ستون صفرم پر شده اند و خانه‌های دیگر اشتراک ⌞ دارند. تمام خانه‌های باقی مانده را باید با این روش پر کنیم. پس با دو for تو در تو از 1 تا انتهای هر بعد این کار را انجام مدهیم. فرض کنید در حال پر کردن خانه‌ای در سطر j ام و ستون i ام گرید بندیمان باشیم. خانه‌ی چپی که در سطر j ام و ستون i-1 ام قرار دارد را به همراه نوارهای اضافه اش در متغیر ptchL ذخیره میکنیم و همچنین خانه‌ی بالایی که در سطر j-1 ام و ستون i ام قرار دارد را به همراه نوارهای اضافه اش در متغیر ptchU ذخیره میکنیم. با استفاده از تابع findBestPatch2 بهترین پچی را که هم به پچ چپی و هم به پچ بالایی شبیه است را انتخاب میابیم و در متغیر ptchM ذخیره میکنیم. سپس این پچ را ابتدا با پچ چپی با تابع merge که بالاتر تعریف شد ادغام میکنیم و خروجی را در سطر j ام و ستون i-1 تا i قرار میدهیم زیرا گفتیم که همواره باید دو خانه به خروجی تابع merge اختصاص داد شود که البته دوباره اضافات آن هم به گرید های همسایه وارد میشوند. اکنون میخواهیم پچ میانی یا همان ptchM را با پچ بالایی ادغام کنیم. ولی پچ میانی پس از ادغام با پچ راستی دستخوش تغییراتی شده و ممکن است بخش هایی از پچ چپی در آن تنیده شده باشد. برای آنکه این تغییرات اثر داده شود دوباره محدوده‌ی خانه‌ی سطر j ام و ستون i ام را بهمراه نوارهای اضافی‌اش بعنوان پچ وسط انتخاب میکنیم و در متغیر ptchM ذخیره میکنیم. اکنون با همان ترفند چرخش 90 درجه که بالاتر ذکر شد آن را با پچ بالایی با تابع merge که بالاتر تعریف شد ادغام میکنیم و خروجی را در سطر j تا j-1 و ستون i ام قرار میدهیم که البته دوباره اضافات آن به گرید های همسایه وارد میشوند. همواره این اضافات را باید نگه داریم زیرا اطلاعات درون آنها در ادغام دو پچ مورد نیاز است. اکنون تمام خانه های گریدبندی پر شده اند. فقط کافیست به اندازه‌ی پهنای نوار مشترک یا همان bandwidth از طرف راست و پایین تصویر imgSyn بریده شود. تا گرید بندی خالص گردد و تصویری با ابعاد هدف حاصل شود. اکنون imgSyn را بعنوان خروجی بازگشت میدهیم.

حال پس از توضیحات مربوط به توابع پیاده سازی شده به توضیحات روند استفاده از آنها و رسیدن به خروجی مطلوب میپردازیم.

میدانیم که تصویر نهایی باید 2500×2500 باشد. پس میتوانیم از پچ‌هایی به ابعاد خالص 125×125 استفاده کنیم که در هر سطر و ستون 20 عدد پچ جا میشود. حال پهنای نوار مشترک را 35 در نظر گرفته و با تابع synthesis که بالاتر تعریف شد تصویر نهایی را بدست میاوریم. لازم به ذکر است

از آنجا که اولین پچ رندوم انتخاب میشود، به تبع آن با هربار اجرا نتیجه متفاوت خواهد شد ولی دقت آن تغییری نخواهد کرد.