

به نام خدا

دانشگاه صنعتی شریف

دانشکده مهندسی برق

دکتر عمال الدین فاطمی زاده - پردازش تصاویر دیجیتال

نیم سال دوم ۱۴۰۰-۱۴۰۱



## تمرین عملی سری ششم

**لطفاً به نکات زیر توجه بفرمایید: (رعایت نکردن این موارد باعث کاهش نمره می‌شود.)**

۱. نتایج و پاسخ‌های خود را در یک فایل با فرمت zip به نام HW6-Name-StudentNumber در سایت Quera قرار دهید. همچنین فایل پایتون یا متلب خود را به همان نام در قسمت مخصوص به خود آپلود کنید.
  ۲. کسب نمره کامل در هر سوال مستلزم تحويل کدها (۴۰ نمره) و توضیحات (۳۰ نمره) و نتایج (۳۰ نمره) می‌باشد.
  ۳. کدهای شما تمامًا باید توسط خودتان نوشته شده باشند. هرگونه استفاده از کد دیگران، اعم از دوستان و اینترنت، به هر شکل ممکن، تقلب محاسب می‌شود و نمره تمام تمرینات جاری و تمام تمرینات قبلی صفر خواهد شد. با اجرای این کدها باید همان نتایجی که فرستاده اید قابل بازیابی باشند. برنامه شما باید به گونه‌ای باشد که بدون نیاز به هیچ تغییری قابل اجرا باشد، در غیر این صورت هیچ نمره‌ای تعلق نخواهد گرفت.
  ۴. برای تمام سوالات، باید جزئیات روشنی که استفاده کردید را توضیح دهید و نتایجی که گرفته‌اید را ارائه دهید. این توضیحات می‌تواند در یک فایل pdf و یا در یک فایل ipynb باشد. در توضیحات، باید اشاره کامل به کارهایی که انجام داده‌اید بنمایید به طوری که یک شخص آگاه از موارد درس بتواند به آسانی متوجه کاری که شما انجام داده‌اید شود.
  ۵. در طول ترم ارسال با تاخیر پاسخ همه‌ی تمارین تا سقف شش روز و در مجموع بیست و یک روز وجود دارد. پس از گذشت این مدت، پاسخ‌های ارسال شده پذیرفته نخواهند بود. همچنین، به ازای هر روز تأخیر غیر مجاز بیست درصد از نمره تمرین به صورت ساعتی کسر خواهد شد.
  ۶. اگر از Jupyter notebook استفاده می‌کنید، می‌توانید خروجی‌ها را پاک کنید تا حجم فایل تحويلی زیاد نشود.
  ۷. مهلت تحويل: **۷**
  ۸. نام طراح هر سوال در زیر آن نوشته شده است و شما می‌توانید سوالات خود را از طریق ایمیل یا تلگرام از طراح سوال پرسید.
- ارسان فیروزی: @Arsalanfiroozi - Arsalan.firoozi@gmail.com
- سید سعید رضوی: @RazooIs - Saeedrazavi890@gmail.com
- امیرحسین جوادی: @Amirhosein\_javadi - Javadiamirhosein.2000@gmail.com
- امیررضا حاتمی‌پور: @Arhp78 - arhp78@gmail.com

## ۱ لاین دیتکشن

طراح : امیرحسین جوادی

در این تمرین روش هاف (Hough) برای پیدا کردن خطوط در یک تصویر را پیاده سازی می نمایید. دقت نمایید که فقط از توابع آماده‌ای که در ادامه گفته می شود می توانید استفاده کنید و بقیه موارد را خود شما باید پیاده سازی کنید.

۱. دو تصویر im01.jpg و im02.jpg را در نظر بگیرید. ابتدا لبه‌های این دو تصویر را با روش دلخواه خود به دست آورده و تصاویر آن ها را با نامهای res01.jpg و res02.jpg به ترتیب برای تصاویر اول و دوم ذخیره کنید. برای به دست آوردن لبه‌ها می توانید از توابع لبیابی آماده استفاده نمایید

۱.



(ب) im02

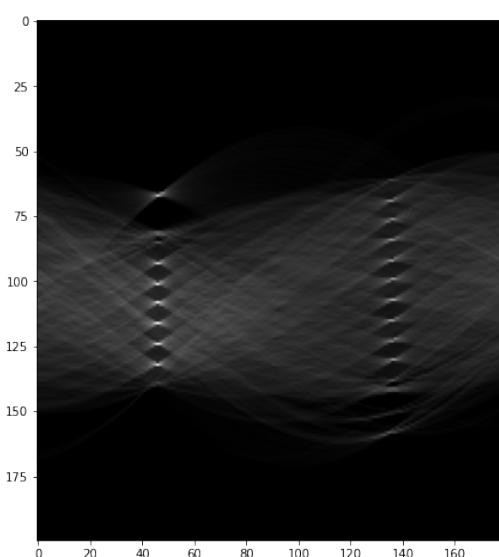


(الف) im01

شکل ۱

۲. خطوط را در دستگاه قطبی در نظر بگیرید. زاویه خطوط با محور افقی و فاصله آن‌ها تا مبدأ را به دلخواه خود گسسته نمایید و یک ماتریس انباشتی (accumulator) برای آن‌ها در نظر بگیرید. تمام درایه‌های این ماتریس را در ابتدا مساوی صفر قرار دهید. برای هر پیکسل لبه، برای تمام خطوطی که از آن پیکسل عبور می‌کنند، درایه متناظر در ماتریس انباشتی را یک واحد افزایش دهید. در انتها، این ماتریس انباشتی را به صورت یک تصویر نمایش داده و با نام res04-hough-space.jpg و res03-hough-space.jpg به ترتیب برای تصویر اول و دوم ذخیره نمایید. این قسمت را خود شما باید پیاده سازی نمایید و نمی توانید از توابع آماده استفاده نمایید.

۲.



شکل ۲ : accumulator matrix

۳. خطوط را از ماتریس‌های انباشتی به دست آورده و روی تصاویر اصلی کشیده و با نام‌های res05-lines.jpg و res06-lines.jpg ذخیره نمایید. خطوط را از ماتریس‌های انباشتی طوری به دست بیاورید که تمام خطوط صفحه شطرنجی پیدا شوند. در این صورت تعدادی خطوط دیگر هم در هر تصویر پیدا خواهند شد. با استفاده از دانش پردازش تصویری که دارید، سعی کنید بقیه خطوط را حذف کرده و فقط خطوط محدوده شطرنجی باقی بمانند. خطوط باقی‌مانده را در تصاویر اصلی رسم کرده و با نام‌های res07-chess.jpg و res08-chess.jpg ذخیره نمایید.

۲.

۴. از محل طلاقی خطوط محدوده شطرنجی گوشه‌های مربع‌های محدوده شطرنجی را به دست آورده و آن‌ها را روی تصاویر اصلی نشان داده و با نام‌های res09-corners.jpg و res10-corners.jpg ذخیره نمایید.

۳.

## ۲ گردالی دیتکشن

طراح: امیرحسین جوادی

الگوریتم بالا را برای تشخیص دایره استفاده کنید. دقت نمایید که فقط از توابع آماده‌ای که در ادامه گفته می‌شود می‌توانید استفاده کنید و بقیه موارد را خود شما باید پیاده سازی کنید. روش Hough transform در تمام مسائلی که به فرم  $g(v, c) = 0$  که  $v$  بردار مختصات و  $c$  بردار ضرایب باشد، قابل استفاده است. به عنوان مثال، نقاط روی دایره به فرم زیر هستند.

$$(x - c_1)^2 + (y - c_2)^2 = c_3^2$$

تفاوت با حالت تشخیص خط این است که در این مسئله سه پارامتر  $\{c_1, c_2, c_3\}$  برای مشخص کردن داریم.

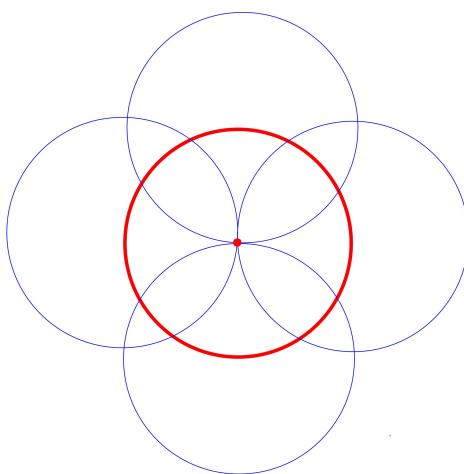
۱. تصویر im03.jpg را در نظر بگیرید. ابتدا لبه‌های این تصویر را با روش دلخواه خود به دست آورده و تصاویر آن را با نام res11.jpg ذخیره کنید. برای به دست آوردن لبه‌ها می‌توانید از توابع لبیابی آماده استفاده نمایید. در نهایت با گذاشتن استانه مناسب باید به یک تصویر باینری برسید که لبه را از غیر لبه مشخص کند.

۴.



شکل ۳: im03

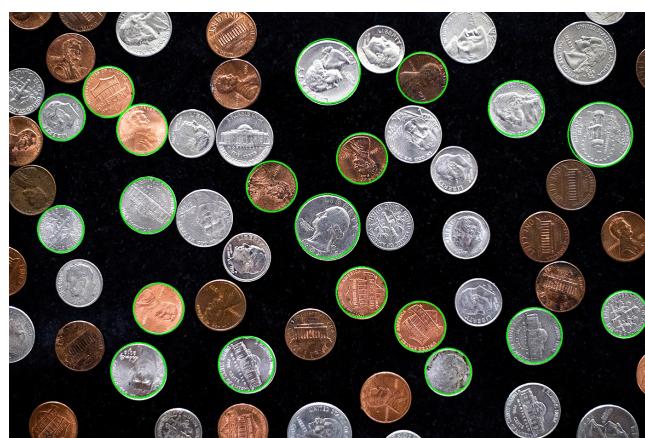
۲. اگر پیکسلی مانند  $(x, y)$  روی دایره‌ای به مرکز  $(c_1, c_2)$  و شعاع  $c_3$  باشد، مکان هندسی نقاط ممکن برای مرکز این دایره برابر است با دایره‌ای به مرکز  $(x, y)$  و شعاع  $c_3$ . اگر برای هر یک از نقاط روی دایره، مکان هندسی ذکر شده را در نظر بگیریم، مشخص است که اشتراک این مکان هندسی‌ها همان مرکز مورد نظر ماست. از این نکته برای پیدا کردن دایره روی تصویر استفاده می‌کنیم. به شکل ۴ توجه کنید. هدف پیدا کردن مرکز دایره ( نقطه‌ی قرمز ) است. همان طور که مشخص است اگر از هر نقطه روی محیط دایره، دایره‌ای به شعاع مورد نظرمان بزنیم از مرکز می‌گذریم. پس با اشتراک گیری بین این دایره‌ها می‌توانیم به مرکز برسیم.



شکل ۴

موردی که وجود دارد این است که ما شعاع را هم به عنوان مجھول داریم. اگر شعاع را داشتیم میتوانستیم به مرکز هر لبه دایره‌های به شعاع مورد نظر بزنیم و در آخر نقاطی که دایره‌های زیادی از آنها گذشته است را به عنوان مراکز احتمالی دایره‌های عکس‌مان گزارش کنیم. حال که شعاع را نداریم باید برای شعاع‌های مختلف این مورد را بررسی کنیم. به همین دلیل است که در این مسئله فضای جست و جو به جا دو بعدی، سه بعدی است و علاوه بر مختصات مرکز، شعاع هم باید جست و جو شود.

ماتریس انباشتی سه بعدی به اندازه‌ی  $[h, w, c]$  در نظر بگیرید که  $[h, w]$  اندازه‌ی تصویر ورودی است و  $c$  تعداد شعاع‌هایی است که می‌خواهید بررسی کنید. شعاع‌های مورد نظرتان را می‌توانید به صورت دستی و مخصوص به این مسئله انتخاب کنید. مثلاً برای شعاع‌هایی به اندازه‌ی  $\{50, 55, 60, 65, 70\}$  پیکسل، دایره‌های زیر حداقل در نهایت مشخص می‌شوند.



شکل ۵

بعد از انتخاب شعاع‌های مورد علاقه‌تان برای بررسی این مسئله، ماتریس انباشتی متناظر را بسازید. برای هر لایه شعاع متناظر لایه را مشخص کنید. سپس برای هر لبه، دایره‌ای به شعاع مشخص شده بزنید و به درایه‌ی متناظر هر پیکسل روی این دایره در ماتریس انباشتی یک واحد اضافه کنید. در نهایت این مرحله شما باید به ماتریس انباشتی موردنظرتان بررسید.

۱۵۰

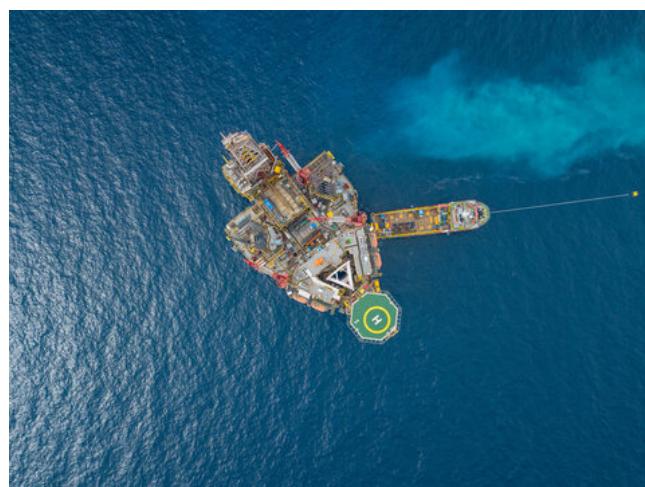
۳. سپس هر کدام از درایه‌های ماتریس انباشتہتان که از مقدار threshold بیشتر باشد را میتوانید به عنوان مرکز یک دایره در نظر بگیرید. دایره‌ها را از ماتریس انباشتی به دست آورده و روی تصاویر اصلی کشیده و با نام res12-circles.jpg ذخیره نمایید. همچنین تعداد دایره‌هایی که پیدا کردید را در برنامه چاپ کنید و در گزارشتان بیاورید. نمره نهایی تابع تعداد دایره و دقت تشخیص شما (تشخیص حداکثر یک دایره برای هر سکه) است.

**ب)**

### Active Contour ۳

طراح : ارسلان فیروزی

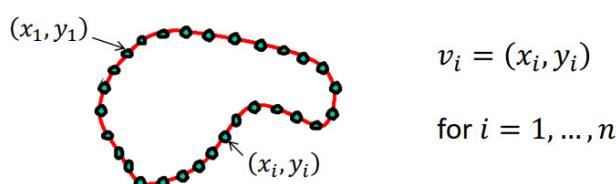
در این سوال قصد داریم با استفاده از روش Active Contours سکوی نفتی در عکس im04.jpg را تقسیم بندی کنیم.



شکل ۶ : im04

یک روش تقسیم‌بندی است که برای جدا کردن پیکسل‌های مورد نظر از یک تصویر برای پردازش و تحلیل بیشتر استفاده می‌کند.

کانتورها مرزهایی هستند که ناحیه مورد نظر را در یک تصویر مشخص می‌کنند. هر کانتور از اتصال چندین گره تشکیل شده است. (شکل ۷) در هر مرحله از این الگوریتم سعی می‌کنیم این نقاط را به نحوی تغییر دهیم که در نهایت بر روی حاشیه جسم قرار گیرند.



شکل ۷ : Contour

این تغییر نقاط به نحوی انجام می‌شود که تابع انرژی زیر کمینه شود.

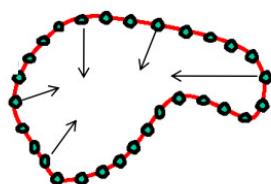
$$E_{total} = E_{internal} + \gamma E_{external}$$

انرژی درونی نماینده شکل کلی کانتور است. انتظار داریم که کانتور ما هموار باشد و یا از شکل هندسی خاصی تبعیت کند.

منظور از هموار بودن کانتور این است که نقاط به گونه‌ای نباشند که یکی از نقاط در فاصله دور از توده اصلی نقاط قرار گیرد و یا نقاط از هم فاصله‌های نامتقارن بگیرند. انرژی درونی را با رابطه زیر پیاده سازی کنید:

$$E_{internal} = \sum_{i=1}^n ((x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 - \alpha d)^2$$

در رابطه بالا،  $d$  میانگین فواصل نقاط در ایتریشن فعلی است. اگر این ترم وجود نداشت کانتور همواره به سمت کوچک شدن پیش میرفت و در نهایت در یک نقطه همگرا می‌شد. این مورد در شکل ۸ نشان داده شده است.

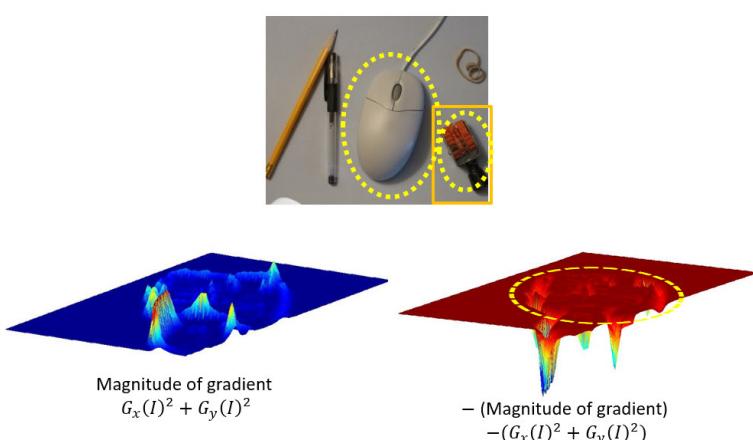


شکل ۸

انرژی خارجی مشخص کننده‌ی میزان مطابقت کانتور با داده‌های تصویر است و باعث می‌شود که کانتور بر روی حاشیه جسم فیت شود. برای این منظور از جمع مقادیر منفی اندازه گرادیان در نقاط کانتور صورت می‌پذیرد. (شکل ۹)

$$E_{external} = - \sum_{i=1}^n (G_x(x_i, y_i)^2 + G_y(x_i, y_i)^2)$$

در گرادیان تصویر، حاشیه جسم مقادیر بزرگی دارد و در صورتی که منفی آن مقادیر را در تابع هزینه در نظر بگیریم، مقدار کمینه معادل با قرار گرفتن این نقاط بر روی حاشیه جسم به صورت هموار است.



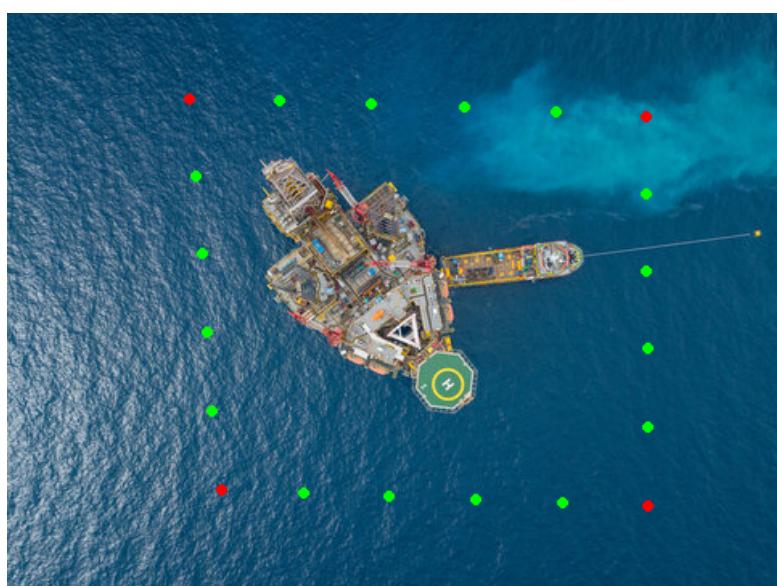
شکل ۹

با مشخص شدن تابع هزینه، باید همه نقاط را به گونه‌ای تغییر دهیم که در راستای بیشترین کاهش تابع هزینه حرکت کنیم. نتیجه مطلوب با استفاده از Dynamic Programming قابل حصول است. در هر iteration، هر کدام از گره‌های کانتور به همسایگی ( $n \times n$ ) خود (که معمولاً  $(3 \times 3)$ ،  $(5 \times 5)$  و یا  $(7 \times 7)$  است) بروند و  $E_{total}$  را مینیمم سازند. از آن جایی که هر گره در یک همسایگی خود میتواند حرکت کند، خوشبین داریم که با اجرای الگوریتم کانتور به جسم مورد نظر ما نزدیک و نزدیک‌تر شود و به مینیمم تابع هزینه برسد و در مینیمم‌های موضعی گیر نیوفتد.

روشی است که برای حل این مسئله با هزینه کم محاسباتی. برای یادگیری این الگوریتم از Dynamic Programming لینک می‌توانید استفاده کنید. شرط پایان الگوریتم را رد شدن از حداقل تعداد ایتریشن و یا حرکت نکردن بیش از 90 درصد گره‌های کانتور در نظر بگیرید. در صورتی که نیاز به اضافه کردن ترم جدیدی درتابع هزینه دارد، می‌توانید این کار را انجام دهید. در نهایت از شما انتظار می‌رود فیلمی از حرکت گره‌های کانتور از ایتریشن اول تا ایتریشن آخر ارائه دهید. الگوریتم خود را کاملا در گزارش توضیح دهید و پارامترهای آزاد مسئله‌ای که انتخاب کردید را در گزارش ذکر کند.

در زیر پیشنهادهایی جهت پیاده سازی راحت‌تر شما ارائه می‌شود:

- 20 نقطه اولیه را به صورت دستی در کد مشخص کنید. راه دیگر استفاده از تابع cv2.setMouseCallback در پایتون است. با کلیک روی تصویر، چهار نقطه گوش را بگیرید و سپس 20 نقطه را به صورت متقارن با استفاده از 4 نقطه گوش گرفته شده از کاربر، توزیع کنید و به عنوان نقطه شروع کانتور در نظر بگیرید. شکل ۱۰ را مشاهده کنید. نقاط قرمز گره‌های انتخاب شده اولیه است و نقاط سبز به صورت یکنواخت بین این نقاط توزیع شده است.



شکل ۱۰

- در پایان هر ایتریشن عکس همراه با نقاط کانتور روی آن را در پوشه‌ای به ترتیب ذخیره کنید و در نهایت با استفاده از پکیج ffmpeg با دستوری مشابه دستور زیر آن‌ها را به یک فیلم تبدیل کنید (دقت کنید که این پکیج باید بر روی سیستم عامل نصب شود).

```
os.system('ffmpeg -start_number 0 -i Images/%d.png -r 30 -vcodec mpeg4 Contour.MP4');
```

- پیش پردازش تصویر می‌تواند موثر واقع شود.

نتیجه نهایی را به شکل res13-contour.mp4 ذخیره نمایید.

## SLIC Supapixel ۴

طراح: امیررضا حاتمی‌پور

الگوریتم‌های superpixel، گروهی از پیکسل‌ها را که از نظر ساختاری شبیه هم می‌باشند در یک دسته/کلاس قرار می‌دهد. بخارط بار محاسباتی پایینی که این الگوریتم‌ها دارا می‌باشد، در کاربردهایی مانند تشخیص عمق، تشخیص شی و ... استفاده

می‌شوند. در این سوال به پیاده سازی یک الگوریتم ساده به نام Simple Linear Iterative Clustering (SLIC) می‌پردازیم. (لینک مقاله مربوطه)

این الگوریتم تصویر را به  $K$  قسمت تقسیم می‌کند. هر پیکسل در فضای پنج بعدی ویژگی‌های رنگ و مختصات به صورت  $(l, a, b, x, y)$  مشخص شود. می‌توانید بجای فضای رنگ Lab از فضاهای دیگر رنگی مانند rgb یا HSV و ... استفاده کنید. همچنین تعریف‌های مختلفی برای فاصله مثل نرم یک، نرم دو و ... را می‌توانید استفاده کنید. مقدار  $K$  را مقادیر 64، 256، 1024، 2048 قرار دهید.

۱. تصویر im05.png را در نظر بگیرید.  $K$  مرکز به صورت یکنواخت در تصویر قرار دهید. (مثلاً تصویر را به  $K$  بخش مستطیلی مساوی تقسیم کنید و مرکز هر دسته را وسط مستطیل قرار دهید). سپس در یک همسایگی (مثلاً  $5 \times 5$ ) در هر مرکز، پیکسلی را که کمترین گرادیان را دارد بعنوان مرکز جدید انتخاب می‌کنیم (برای آن که مراکز انتخاب شده در مرزهای تصویر نباشند).

۲. هر پیکسل را با مراکزی که به فاصله حداقل  $S$  از آن هستند مقایسه کنید و آن را به بهترین مرکز وصل کنید. مقدار  $S$  را می‌توانید برابر  $\sqrt{N/K}$  بگیرید که  $N$  تعداد پیکسل‌های تصویر است. میزان بهینه بودن هر مرکز را از تابع هزینه زیر متوجه شوید.

$$d_{lab} = \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2}$$

$$d_{xy} = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2}$$

$$D = d_{lab} + \alpha d_{xy}$$

که در این تعریف  $i$  اندیس پیکسل  $i$  است و  $k$  اندیس مرکزی مورد بررسی است. هر پیکسل به دسته‌ای تعلق دارد که فاصله کمتری داشته باشد. پس در پایان این مرحله باید هر پیکسل به یک مرکز مربوط شده باشد. تصویری به اندازه‌ی تصویر اصلی تشکیل دهید که مقدار هر پیکسل برابر با شماره دسته آن پیکسل باشد. چهار نتیجه به دست آمده برای مقادیر مختلف  $K$  را به ترتیب res17-labels.jpg، res16-labels.jpg، res15-labels.jpg، res14-labels.jpg بنامید و ذخیره کنید.

۳. مرز قطعه‌ها را روی تصویر بکشید و چهار نتیجه به دست آمده برای مقادیر مختلف  $K$  را به ترتیب res18-clusters.jpg، res19-clusters.jpg، res20-clusters.jpg، res21-clusters.jpg بنامید و ذخیره کنید. در صورت نیاز، با استفاده از فیلتر median روی لیل‌ها، قطعات نهایی را هموار و پیوسته کنید.

۴. تصویری به اندازه تصویر اصلی در تشکیل دهید که مقدار هر پیکسل برابر میانگین مقدار پیکسل‌های داخل قطعه مخصوص‌اش است. چهار نتیجه به دست آمده برای مقادیر مختلف  $K$  را به ترتیب res22-Superpixel.jpg، res23-Superpixel.jpg، res24-Superpixel.jpg، res25-Superpixel.jpg بنامید و ذخیره کنید.

## Texture transfer ۵

طراح: سیدسعید رضوی

در این سوال می‌خواهیم pattern، (طرح) یک تصویر را بر روی یک تصویر دیگر اعمال کنیم. برای مثال به دو شکل پایین و نتیجه حاصل نگاه کنید:

برای اینکار ابتدا لازم است تا تعریف و توضیح مختصه نسبت به texture synthesis، داشته باشیم : فرض کنید یک طرح کوچک در اختیار داریم و میخواهیم با استفاده از آن یک تصویر بزرگتر بسازیم (نمونه این کار را میتوانید در تصاویر پایین ببینید )

شکل ۱۲

برای ساختن تصویر بزرگتر روش های متفاوتی در پیش رو داریم:

۱. اگر بدون هیچ زحمتی هر بار بخشی از طرح اولیه را به صورت بلاک هایی در کنار هم قرار دهیم به تصویر زیر میرسیم:

شکل ۱۳

۲. راه دیگری که در پیش داریم این است که اولین بلاک (block)، انتخاب شده به صورت رندوم و کاملاً دلخواه باشد اما بقیه بلاک ها را متناسب با ناحیه اورلپ انتخاب کنیم . به این صورت ناحیه ای را به عنوان اورلپ در نظر بگیریم و آن ناحیه اورلپ را در عکس (texture)، خود سرچ کنیم و نواحی با کمترین هزینه یا بیشترین شباهت (هزینه را میتوان به گونه های مختلفی تعریف کرد و یکی از بهترین تعاریف ، L2 norm، میباشد که برابر است با توان ۲ اختلاف پیکسل ها در ناحیه اورلپ ) را در نظر گرفته ، یکی از آنها را را انتخاب کنیم و به بلاک دوم بچسبانیم و ادامه تصویر را بسازیم . اگر این کار را بکنیم داریم :

شکل ۱۴

همانطور که مشاهده میکنید کیفیت ساخت تصویر بسیار بهتر شده اما باز هم ناپیوستگی ها در تصویر حاصل مشخص است . برای رفع این مشکل از روش سوم استفاده میکنیم :

۳. می توانیم از روش cut minimum boundary ، که در تمرین سری ۵ نیز با آن آشنا شدید استفاده کنید . اگر از این روش استفاده کنیم نتیجه زیر حاصل میشود :

شکل ۱۵

اگر بخواهیم طرح کلی از فرایند texture synthesis، داشته باشیم به شکل زیر میرسیم :

شکل ۱۶

حال توضیح مختصه در رابطه با texture transfer میرسیم :  
ورودی ما یکی پtern (texture) و دیگری تصویر هدف (که میخواهیم پtern مدنظر روی آن اعمال کنیم ) میباشد . برای این کار علاوه بر اینکه هدف مینیم کردن هزینه در ناحیه اورلپ می باشد باید بلاک انتخاب شده در هر مرحله (به جز مرحله اول که انتخاب بلاک رندوم است ) علاوه بر برآورده کردن شرط کمترین هزینه بین ناحیه اورلپ بلاک جدید و قبلی آن در پtern ورودی ، باید کمترین هزینه را بین بلاک جدید و متناظر آن در همان مختصات در تصویر

هدف برآورده شود. به عبارتی هدف پیدا کردن بلوکی است که شرط زیر را برآورده کند :

$\text{Minimize} : [\text{cost}_f\text{unction}(\text{block}_i \text{and} \text{block}_{i-1} \text{in} \text{textureimage}) + (1 - )\text{cost}_f\text{unction}(\text{block}_i \text{in} \text{textureimage})]$   
برای مطالعه بیشتر مقاله را از ( این لینک ) مشاهده کنید .

حال که با texture transfer آشنا شدید همین الگوریتم را برای دو تصویر که در فایل زیپ با نام های im06.jpg و res26-texture.jpg ذخیره نمایید . قرار دارند انجام دهید و نتیجه نهایی را با نام im07.jpg انجام دهید .