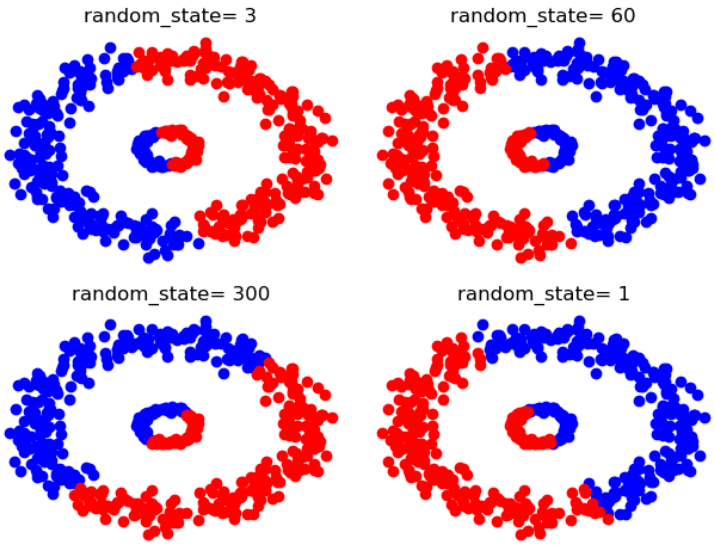
گزارش تمرین سری سوم

اصول پردازش تصویر

رضا اکبریان بافقی - ۹۵۱۰۰۰۶۱

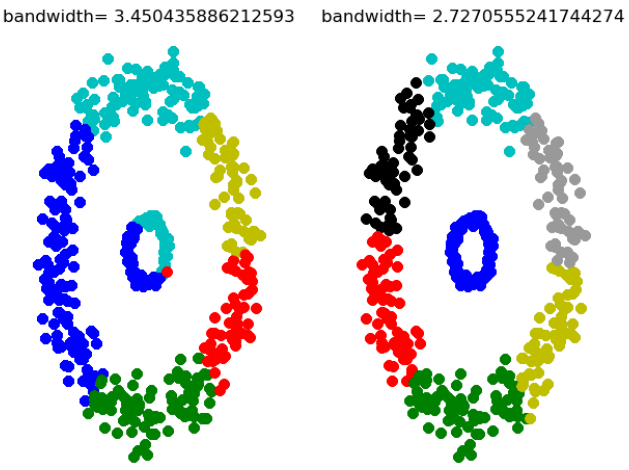
۱

در واقع اگر بخواهیم با k-means عمل کلاسترینگ را انجام بدهیم اگه فقط مختصات نقاط، پارامترهای ما باشند. نمی‌توانیم کلاسترینگ خوبی با این شکل از نقاط داشته باشیم. نتایج مختلفی که با random_state های مختلف گرفته شده‌است را در شکل یک می‌توانید ببینید.



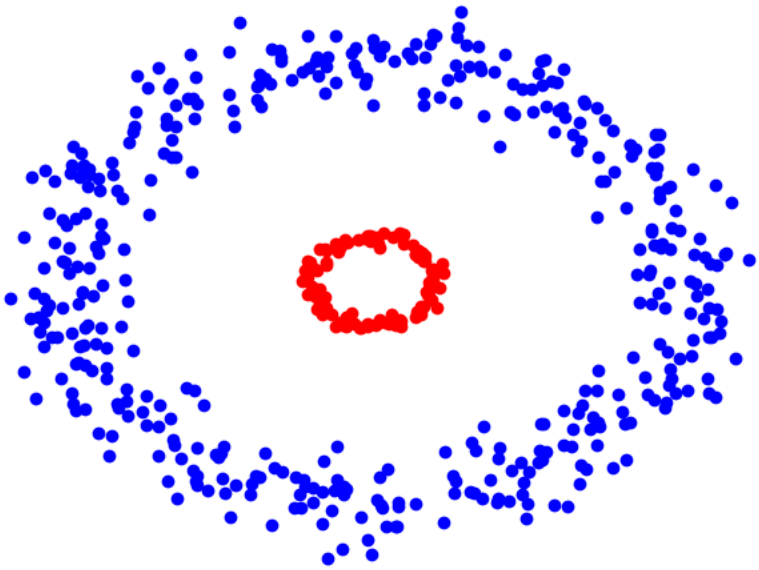
شکل 1 k-means روی نقاط با random_state های مختلف

برای مین‌شیفت در واقع هرچه مقدار bandwidth را عدد کوچکتری قرار دهیم، تعداد دسته‌ها بیشتر خواهد شد. و زمانی که تعداد دسته‌ها بیشتر شود آنگاه می‌تواند دایره وسط را پیدا کند. در شکل دو، با دو مقدار متفاوت bandwidth نمودار رسم شده‌است.



شکل ۲ مین‌شیفت با bandwidth های متفاوت

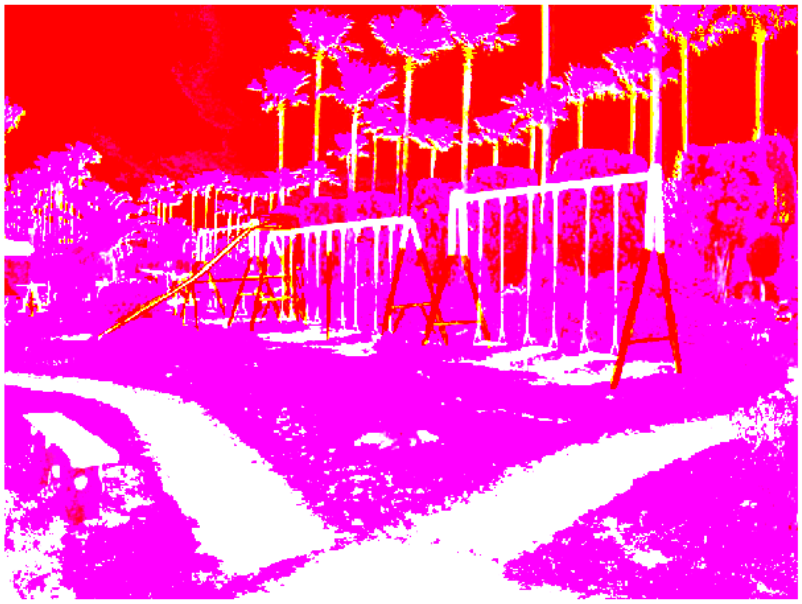
حال اگر ما یک پارامتر سومی اضافه کنیم که در آن فاصله از مرکز قرار بگیرد، آنگاه اگر k-means را انجام بدهیم، می‌بینیم که نتیجه همانطور که مطلوب ما است خواهد بود.



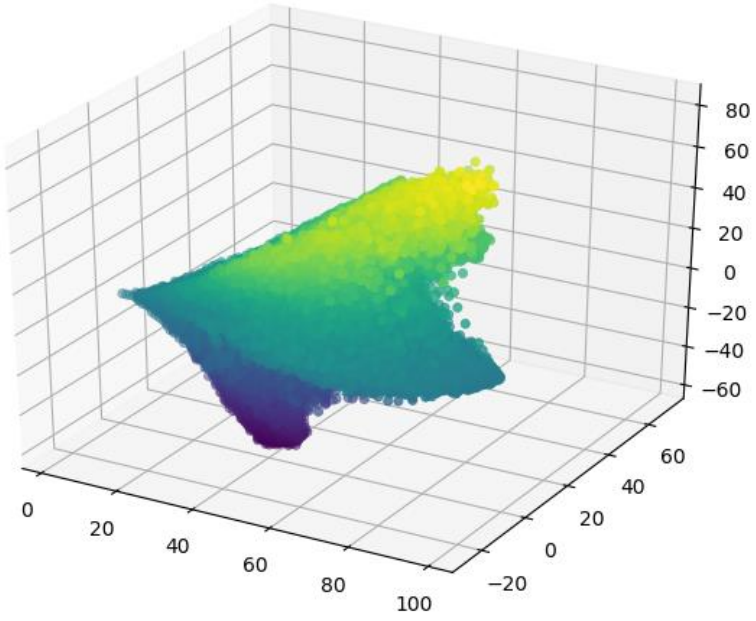
شکل ۳ k-means با استفاده از فاصله از مرکز

۲

ابتدا تصویر را به فضای Luv می‌بریم. برای سریع‌تر شدن مین‌شیفت، ابعاد تصویر را ۰,۲ برابر می‌کنیم. روی تصویر یک گاوس با سیگمای ۳ می‌زنم تا قسمت‌های زائد حذف شوند. شکل نمودار نقاط در فضای Luv را نشان می‌دهد.



شکل 4 تصویر در فضای Luv



شکل 5 فضای Luv تصویر

سپس bandwidth را از طریق تابع زیر محاسبه کرده

```
cl.estimate_bandwidth(image, quantile=0.08, n_samples=1000)
```

و پس از آن مین‌شیف را انجام داده و تصویر را به آن فیت می‌کنم.

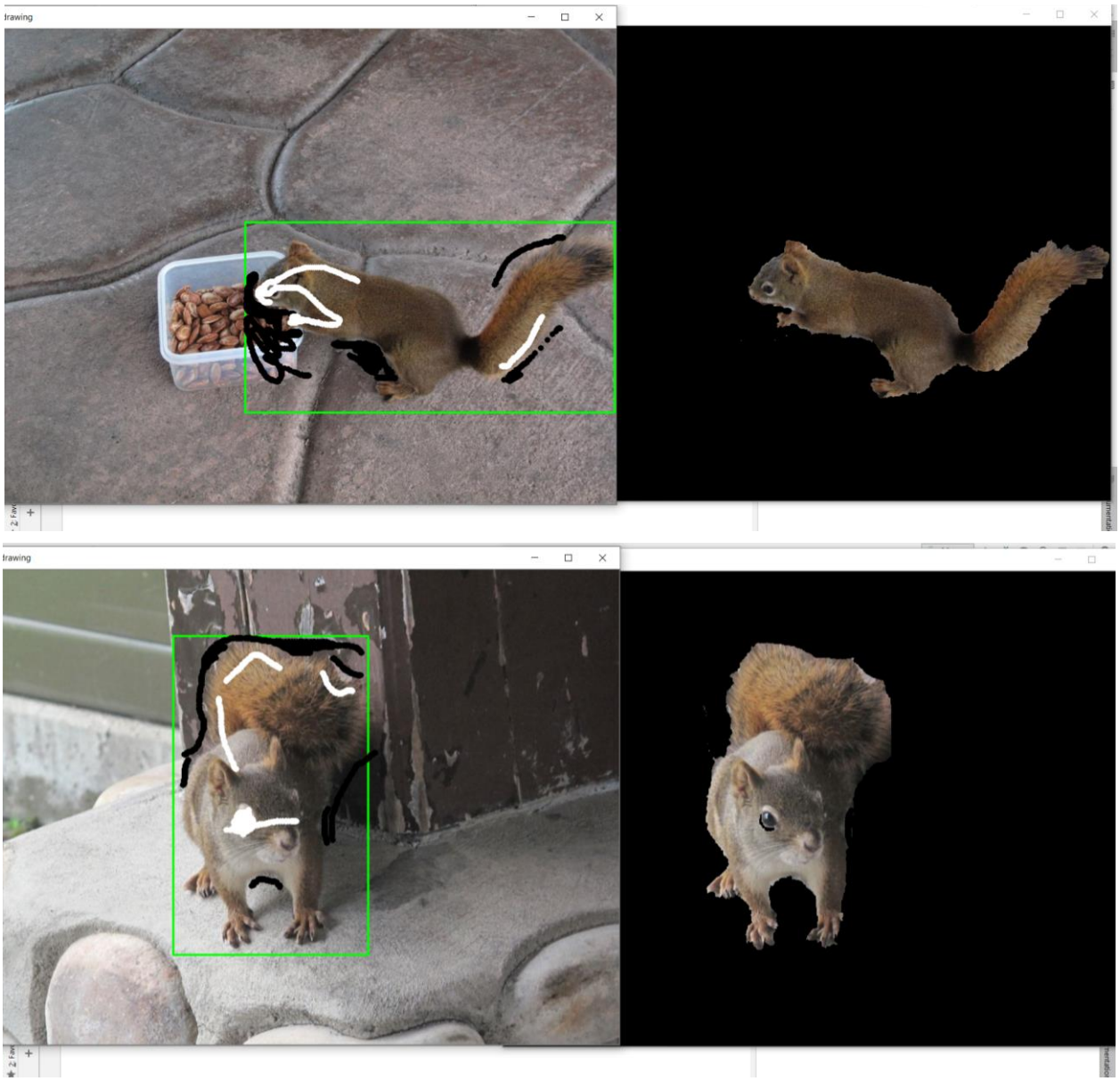
در نهایت برای تشخیص رنگ هر بخش، ابتدا مرکز آن بخش را پیدا کرده و سپس رنگ آن را از تصویر اولیه برمی‌دارم و در نهایت آن را به فضای rgb می‌برم.

۳

در این سوال به روش grabcut سگمنتیشن را انجام می‌دهم. بدین صورت که ابتدا پنجره شامل عکس باز می‌شود و از ما مستطیل ورودی که شکل شامل آن است را می‌گیرد. سپس با فشردن دکمه r، نتیجه فانکشن در پنجره جدید باز می‌شود.

حال با فشردن دکمه b می‌توانید پس‌زمینه و با فشردن دکمه f آبجکت را با کشیدن خط مشخص کنید. حال با زدن دکمه e در هر ایتریشن از تغییرات روی تصویر اعمال می‌شود.

به عنوان نمونه تصویر زیر از طریق همین کار ساخته‌شده‌است.



شکل ۶ دو تصویر، به همراه نقاط اولیه مشخص‌کننده آن‌ها.

۴

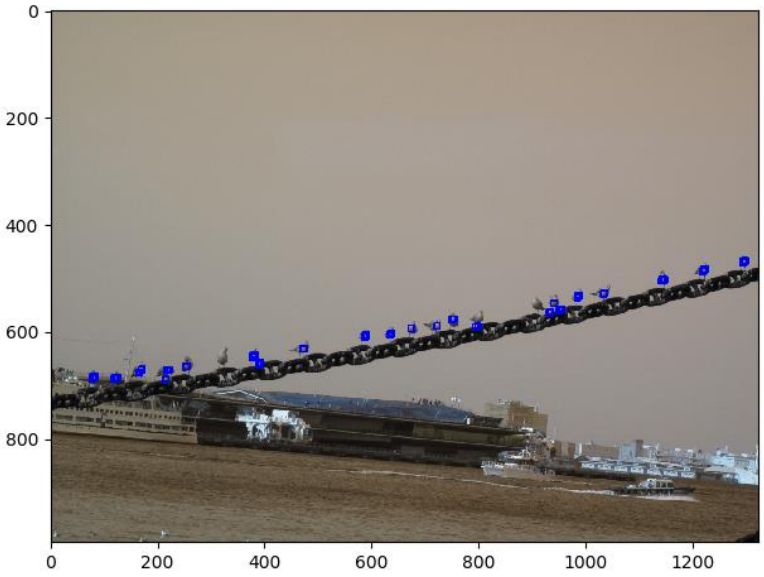
ابتدا ابعاد تصویر را ۰٫۳ برابر می‌کنم. با تابع زیر تصویر سگمنت بندی می‌کنم.

```
segments_fz = felzenszwalb(img, scale=200, sigma=0.7, min_size=100)
```

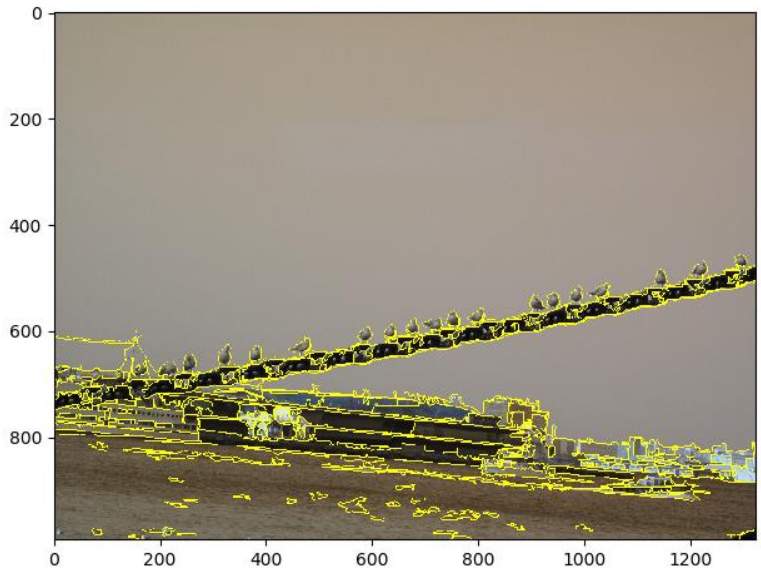
سپس با استفاده از matchTemplate، تصویر بدن کبوتر را در تصویر پیدا می‌کنم. سپس در segments_fz می‌بینیم که آنجایی که matchTemplate گفته بدن کبوتر دیده است، مربوط به سگمنت شماره چند است. آنها را در لیستی ذخیره می‌کنیم. سپس سگمنت‌های مربوطه را در یک تصویر نمایش می‌دهیم. من ۳ نقطه را دستی وارد کردم. چون آن‌ها توسط matchTemplate پیدا نشده بودند یا اشتباه پیدا شده بودند.

برای matchTemplate، ترشهولد ۰٫۹ را در نظر می‌گیریم.

شکل ۷ تمپلیتی که در تصویر به دنبال آن می‌گردیم



شکل ۸ تصویر حاصل از matchTemplate، مربع‌های آبی پرنده‌های پیداشده هستند.



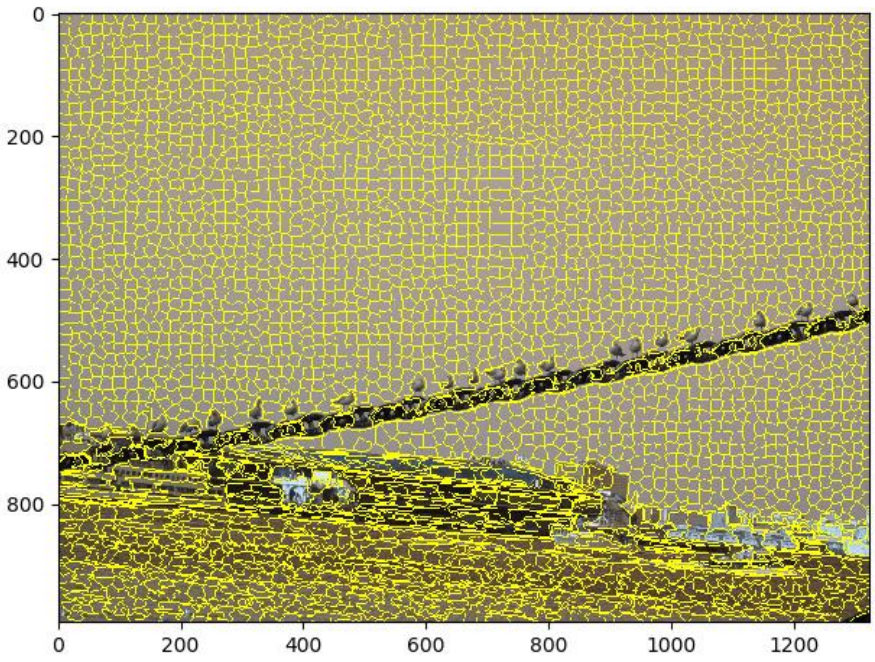
شکل ۹ حاصل سگمنتیشن با استفاده ازfelzenszwalb

۵

مانند سوال قبل ابتدا ابعاد تصویر را ۰٫۳ برابر می‌کنم. با تابع زیر تصویر سگمنت بندی می‌کنم.

```
segments_fz = slic(img,sigma=2.3, n_segments=4000,compactness=1.3,convert2lab=True)
```

و مانند قبل از matchTemplate برای پیدا کردن مکان پرنده‌ها استفاده می‌کنم.



شکل ۱۰ تصویر پس از سگمنتیشن SLIC

۶

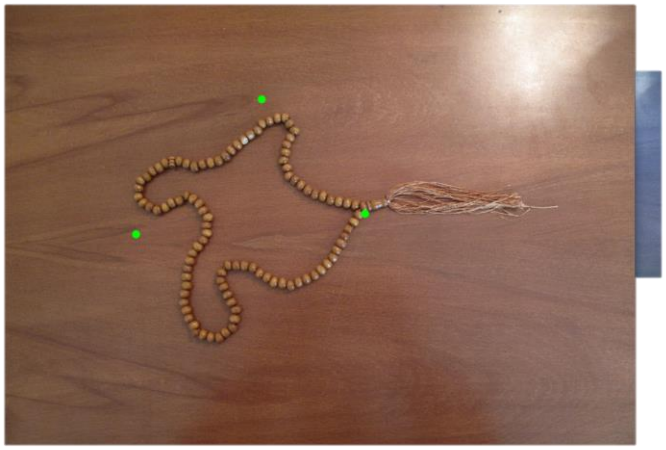
ابتدا از کاربر ۴ نقطه در اطراف تسبیح می‌گیرم. این کار با کلیک چپ روی مختصات نقطه مورد نظر انجام می‌شود. تصویر را با تابع گاوسی با سیگمای ۰٫۳ میگذرانم سپس از تصویر گرادین در راستای افقی و عمودی گرفته و در g بزرگی گرادیان افقی و عمودی را قرار می‌دهم. تابع انرژی نقاط ما به‌صورت زیر برای نقطه p می‌باشد:

```
e_ext = g[p[1]][p[0]]
e_int = np.abs((np.sqrt((p1[0] - p[0]) **2 + (p1[1] - p[1]) **2) - d))
```

که d در هر ایتریشن محاسبه می‌شود. سپس با ضرایب زیر باهم جمع می‌شوند:

```
1400*e_ext +  0.7*e_int
```

در هر ایتریشن اگر فاصله بین نقاط بیش‌تر از دو برابر d قبلی بود، آن‌ها را به‌هم نزدیک می‌کنم. در هر ایتریشن ضریب ۰٫۸۷ را در d ضرب می‌کنم تا d کوچک‌تر شود. هر ۳ ایتریشن در پوشه iteration قرار می‌گیرد. پنجره‌ای که برای جست‌جو در نظر گرفته‌ام ۲۵ در ۲۵ است و هر بار به اندازه نصف مختصات نقطه‌ای که در آنجا بهینه شده است به طرف آن حرکت می‌کنم



شکل ۱۱ انتخاب ۴ نقطه روی شکل