به نام خدا

اصول پردازش تصویر

تمرین سری چهارم

استاد درس

دکتر مصطفی کمالی تبریزی

نیمسال اول ۱۴۰۱-۱۴۰۰

نکات تکمیلی :

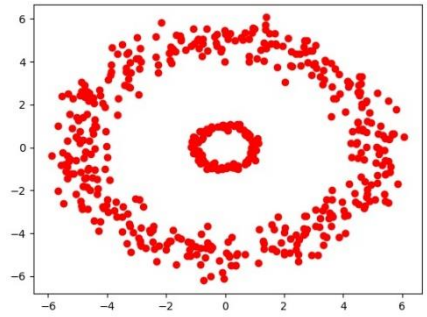
۱. برای ران کردن کد ها ، نیاز به لایبرری های Numpy ، OpenCV , Scipy , Skimage و Matplotlib داریم.
۲. عکس های ورودی را در پوشه resources قرار دهید.
۳. بعد از اجرای هر کد ، نتایج آن در پوشه Result قرار میگیرد. نتایج اصلی در همین پوشه و در صورتی که نتایج دیگری نیز ایجاد شده باشد، در زیرپوشه های دیگری به تفکیک هر تمرین قرار گرفته است.
۴. در لینک زیر نیز میتوانید کلیه نتایج حاصل را مشاهده کنید :

https://mega.nz/folder/Lkw1nK4b#82sQqo_5RgMSluLj1C-mGg

۵. نتایج res01 تا res16 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.

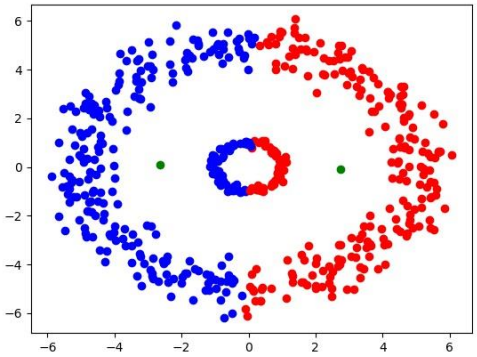
سوال اول)

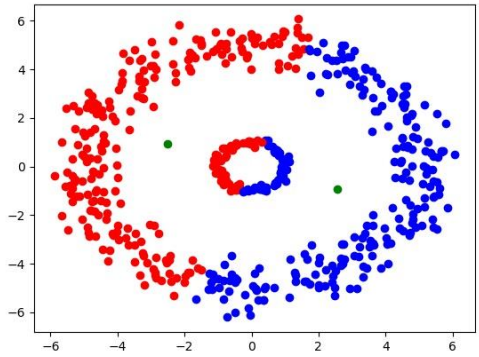
ورودی ما به صورت زیر است :



از آنجا که نقاط به طور تقریبی، متوازن هستند، با هر انجام الگوریتم k_means ، نتایج متفاوتی به دست می آید.(با توجه با اینکه نقاط ابتدایی تصادفی انتخاب میشوند.)

دوبار اجرای این الگوریتم، منجر به نتایج زیر میشود :



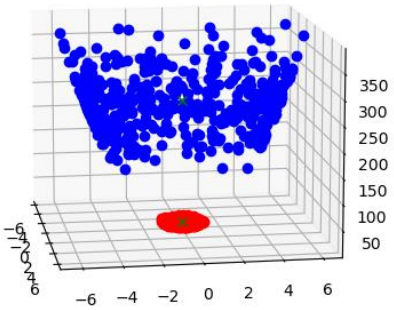


که تقریبا نقاط را به دو دسته مساوی تقسیم کرده است.(نقاط سبز رنگ، مرکز های کلاسترها هستند.)

حال اگر برای هر نقطه، علاوه بر مولفه های x,y ، یک مولفه ی دیگر به صورت ضریبی از فاصله تا مرکز در نظر بگیریم، یعنی یک مولفه ی

$$z = 10 * (x^2 + y^2)$$

در نظر بگیریم و عملیات k_means را در سه بعد انجام دهیم، به نتیجه ای شبیه نتیجه زیر میرسیم :



حداکثر تعداد پیمایش ها نیز، ۱۰ پیمایش درنظر گررفته شده که قابل تغییر است.

سوال دوم)

این سوال نکته خاصی ندارد و همان الگوریتم mean-shift پیاده سازی شده است. به ازای هر پیکسل در عکس اصلی، یک همسایگی در فضای rgb برای آن در نظر میگیریم. سپس نقاط موجود در این همسایگی را پیدا میکنیم و میانگین rgb این نقاط را بدست آورده و ان پیکسل را به این نقطه جدید منتقل میکنیم. این حرکت را انچنان تکرار میکنیم تا به یک همگرایی نسبی با TOL = 1e-7 برسیم. انگاه متوقف میشویم و عکس را ذخیره میکنیم.(البته برای گرفتن نتیجه بهتر، مقادیر xy را نیز با ضریب ۰.۱ در محاسبات وارد میکنیم.)

عکس اصلی

عکس نهایی

سوال سوم)

برای این تمرین، از مقاله زیر که مقاله اصلی SLIC است، استفاده شده است :

https://www.iro.umontreal.ca/~mignotte/IFT6150/Articles/SLIC_Superpixels.pdf

فرض کنیم که عکس ما دارای N پیکسل باشد و بخواهیم در نهایت نیز به طور حدودی K سگمنت داشته باشیم. در این صورت، به طول حدودی در هر سگمنت چیزی در حدود $S^2 = \frac{N}{K}$ پیکسل وجود دارد. به طور خیلی خیلی تقریبی فرض کنید که هر سگمنت مربعی به طول S باشد. در این صورت، برای مراکز اولیه کلاسترها، فاصله هر کلاسترسنتر تا کلاستر سنتر مجاور، چیزی به اندازه ی S خواهد بود.

با دانستن این موضوع ، به طور حدودی K کلاستر سنتر به صورت مرتب از عکس انتخاب میکنیم.(به دلیل تقریب هایی که در فروض اولیه مثلا بدست آوردن عدد S داریم، ممکن است در نهایت تعداد کل کلاسترها دقیقا K نباشد، اما این مقدار به K نزدیک خواهد بود.) سپس هر کلاستر سنتر را در یک همسایگی ۵*۵ حرکت میدهیم و در نقطه ای که کمترین گرادیان را دارد قرار میدهیم.(این کار را انجام میدهیم که این مرکز در نقطه مناسبی باشد، مثلا روی مرز یک شی نباشد و ...)

بعد از انتخاب K کلاسترسنتر، حال نوبت به لیبل گذاری برای پیکسل های عکس است. از آنجا که در نظر گرفتیم هر کلاستر به طور تقریبی S^2 پیکسل دارد، آنگاه میتوان در نظر گرفت که این پیکسل ها در یک همسایگی 2S * 2S حول کلاسترسنتر باشد. بنابراین یک همسایگی حداکثر به اندازه 2S*2S از حول هر کلاستر سنتر انتخاب میکنیم، سپس با استفاده از فرمول های گفته شده، فاصله آن پیکسل ها تا آن کلاستر سنتر را بدست میآوریم.

$$d_{lab} = \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2}$$
$$d_{xy} = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2}$$
$$D_s = d_{lab} + \frac{m}{S} d_{xy} \ ,$$

چنانچه در مقاله بیان شده، ضریب آلفا را به صورت $\alpha = m/S$ تعیین میکنیم که S را بالاتر تعیین کردیم و m را کاربر میتواند تعیین کند.(در برنامه ما، m=20 انتخاب شده است.) دلیل تعریف این ضریب به این صورت این است که در عکس هایی با سایز بزرگ ، فاصله اقلیدسی نقاط را با تقسیم کردن بر S نرمال میکنیم و آن را با یک ضریب مثل m به فاصله رنگی نقاط اضافه میکنیم.

بنابراین مقادیر آلفا عبارتند از :

۱. برای K= 64 : 0.183
۲. برای K= 256 : 0.370
۳. برای K= 1024 : 0.740
۴. برای K= 2048 : 1.052

با بدست آمدن فاصله پیکسل ها تا یک کلاستر سنتر، هر پیکسل را به کلاستر سنتری وصل میکنیم که فاصله آن پیکسل تا آن کلاستر سنتر، کمینه باشد.(به طور معمول، هر پیکسل در همسایگی حداکثر ۴ کلاستر سنتر خواهد بود.)

در نهایت، بعد از اینکه این عملیات را برای تمام کلاسترسنتر ها انجام دادیم و تمام پیکسل های عکس لیبل گذاری شدند، آنگاه کلاستر سنتر ها را آپدیت میکنیم و همین حرکت را به اندازه ی max_iter انجام میدهیم.(در برنامه ما برای اینکه در زمان کوتاه به نتیجه برسیم max_iter=3 تنظیم شده که میتوانید آن را تغییر دهید.)

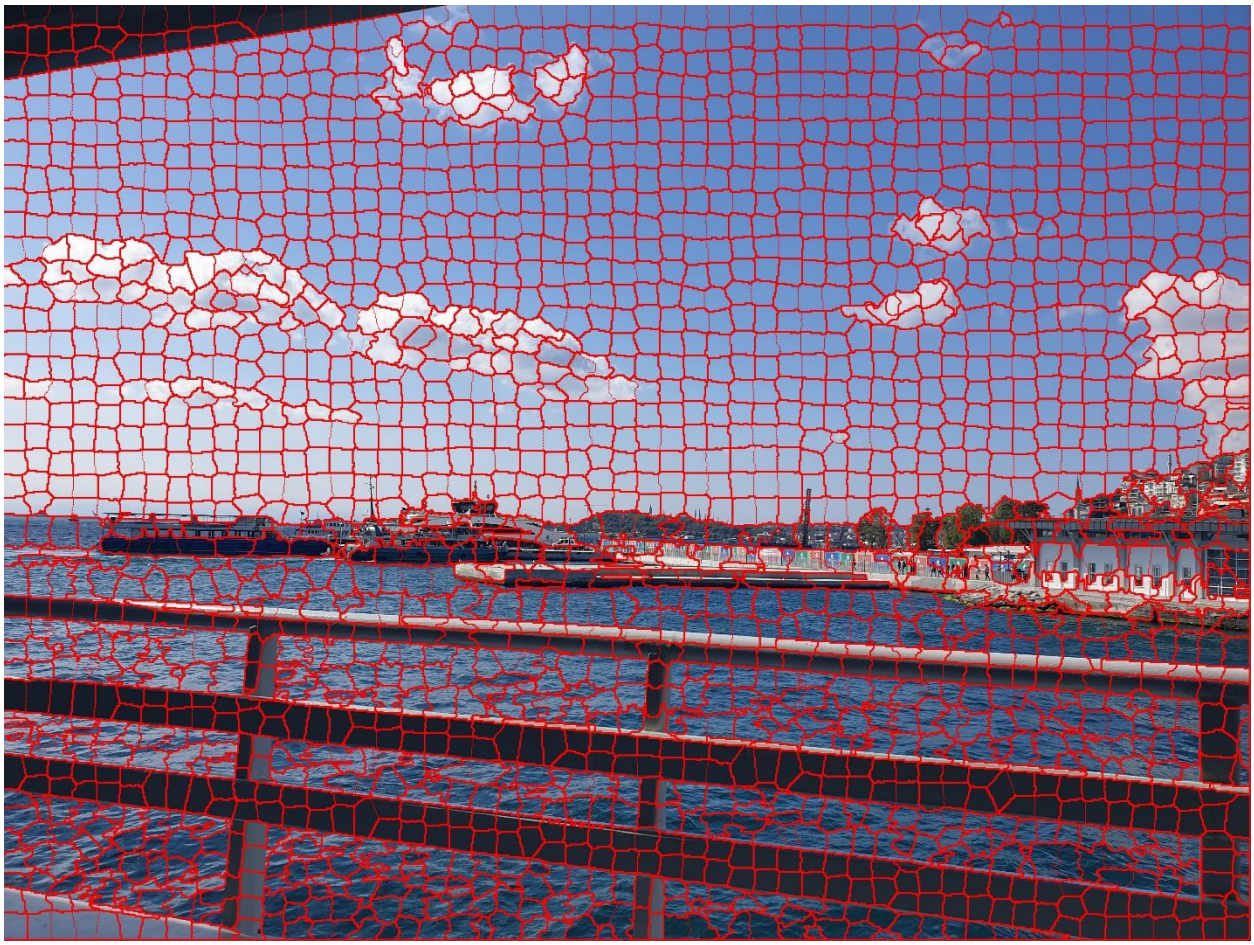
بعد از انجام این کار مثلا به ازای K=64 به نتیجه ای شبیه به عکس زیر میرسیم :

چناچه مشاهده میشود بعضی از کلاستر ها(مخصوصا در آب) به چند مولفه همبندی تقسیم شده اند که یک مولفه بزرگ و تعدادی مولفه کوچکتر را شامل میشود.

برای رفع این مشکل، عملیات enforce_connectivity را انجام میدهیم. این کار در دو مرحله انجام میدهیم. در اولین مرحله با یکبار پیمایش روی عکس، تعدادی از پیکسل هایی که نامطلوب هستند را پیدا میکنیم و آنها را درست میکنیم. با اینکار ، به تعداد مولفه های همبند کل عکس اضافه خواهد شد.(اگر به شکل روبرو نگاه کنید، میبینید که مولفه هبندی صورتی و آبی، مولفه های بزرگی هستند، اما قسمت هایی شبیه دم دارند که در یکدیگر فرو رفته اند و ما با حذف تعداد از پیکسل ها، پیوستگی این دم ها به مولفه اصلی را قطع میکنیم و با اینکار تعداد مولفه همبندی با سایز کوچک، به عکس اضافه میشود که در مرحله بعدی به راحتی قابل حذف است.)

حال با انجام یک عملیات مثل BFS ، به ازای هر مولفه ی همبندی، سایز ان مولفه را بدست میآوریم. اگر سایز آن مولفه از یک مقدار بیشتر بود، متوجه میشویم که این، مولفه همبندی مناسبی است و آن را نگه میداریم. در صورتی که سایز آن کم باشد، یک مولفه همبندی دوم در همسایگی آن (معمولا همسایه بالا، چرا که در مراحل قبلی اصلاح شده است.) را انتخاب میکنیم و تمام نقاط در این مولفه همبندی کوچک را به آن مولفه همبندی دوم منتقل میکنیم.(مولفه همبندی دوم چون در مراحل قبل پردازش شده ، سایز مناسبی دارد و با اضافه شدن تعدادی پیکسل به آن، سایز آن بیشتر میشود که مشکلی ایجاد نمیکند.) اگر این عملیات را به ازای تمام مولفه های همبندی انجام دهیم، مولفه همبندی های کوچک و نامطلوب حذف میشوند و فقط کلاستر های اصلی باقی خواهند ماند :

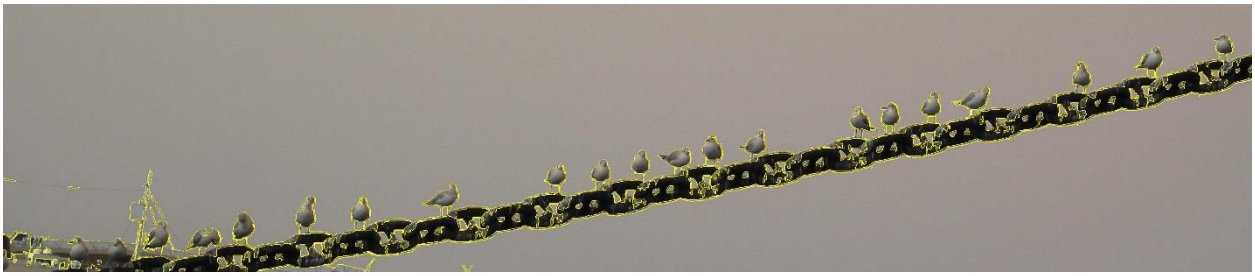
در ابتدای الگوریتم، عکس را به ¼ سایز اصلی کوچک میکنیم، سپس پردازش خود را انجام میدیم. و در نهایت، نتیجه بدست امده را بزرگ کرده و روی عکس اصلی اعمال میکنیم :



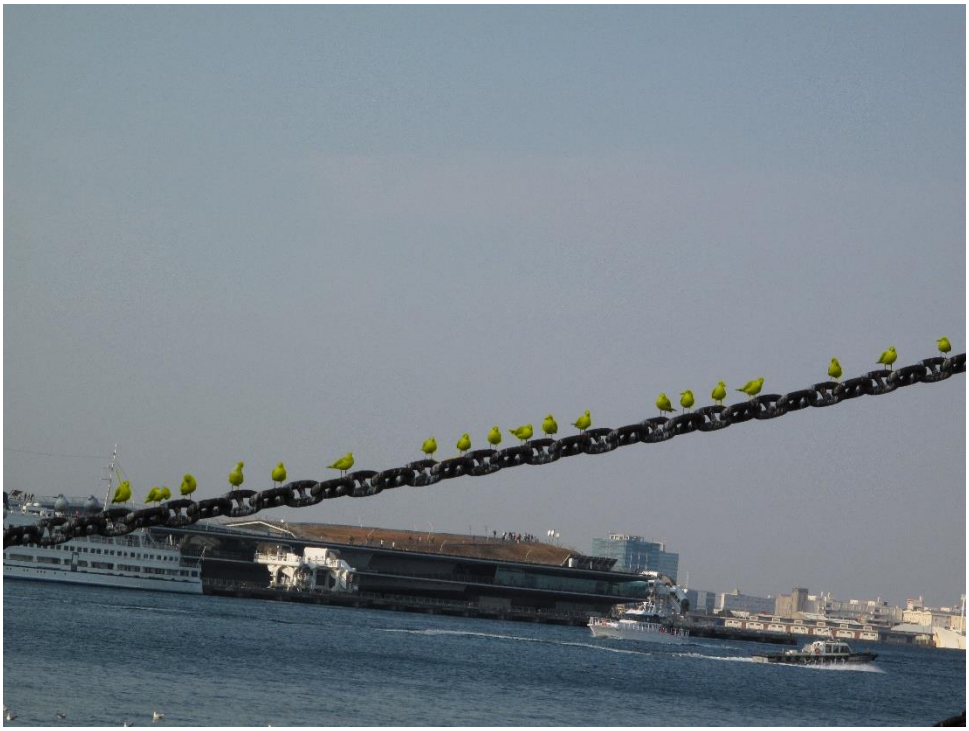
سوال چهارم)

ابتدا برای آنکه حجم محاسبات کمتر باشد، قسمتی از عکس را که پرندگان در آن هستند را کراپ کنیم :

آنگاه با استفاده از روش felzenwalb **که در کلاس تدریس شد، عکس مورد نظر را سگمنت میکنیم :**



قبل از اجرای برنامه نیز، تعدادی نمونه گیری از پرنده ها انجام دادیم و به صورت هاردکد در کد قرار دادیم. در نتیجه به کمک این نمونه ها میتوانیم سگمنت هایی که حاوی پرنده هستند را شناسایی کنیم :

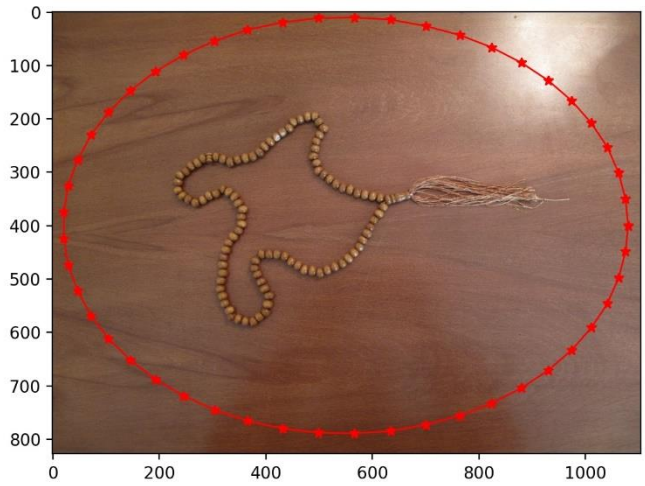


و در نهایت این قسمت را مجددا به عکس اصلی برمیگردانیم و سیو میکنیم.

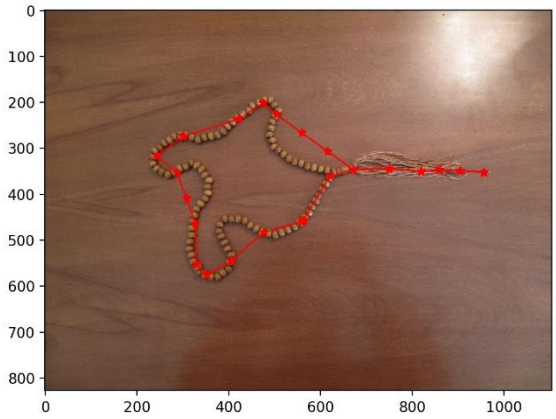
سوال پنجم)

میخواهیم یک اکتیو کانتور به دور تسبیح روبرو رسم کنیم.

ابتدا ، یک کانتور با ۵۰ نقطه به صورت بیضی در نظر میگیریم :

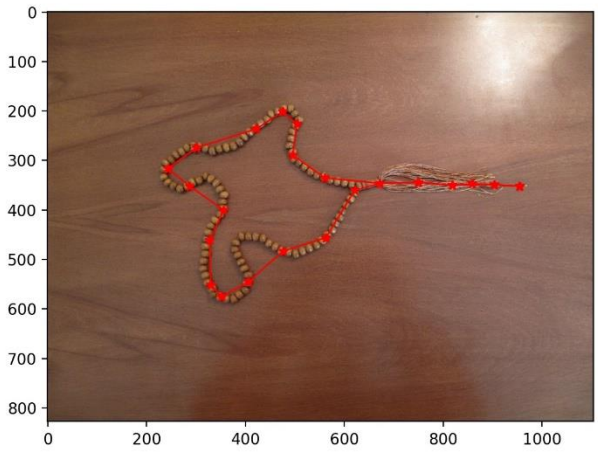


با استفاده از روش اکتیو کانتور که در کلاس توضیح داده شده است ، این عمل را انجام میدهیم. علاوه بر دو تابع انرژی دیفالت(اینترنال و اکسترنال) ، دو تابع انرژی دیگر نیز تعریف میکنیم. تابع سوم ، مجموع فاصله نقاط از مرکز عکس و تابع چهارم ، مجموع فاصله نقاط از مرکز تسبیح است(که این نقطه را در ابتدای اجرای برنامه، کاربر از طریق کنسول ورودی میدهد. برای عکس اصلی از مقادیر پیشفرض x=400 و y=300 استفاده کنید) .

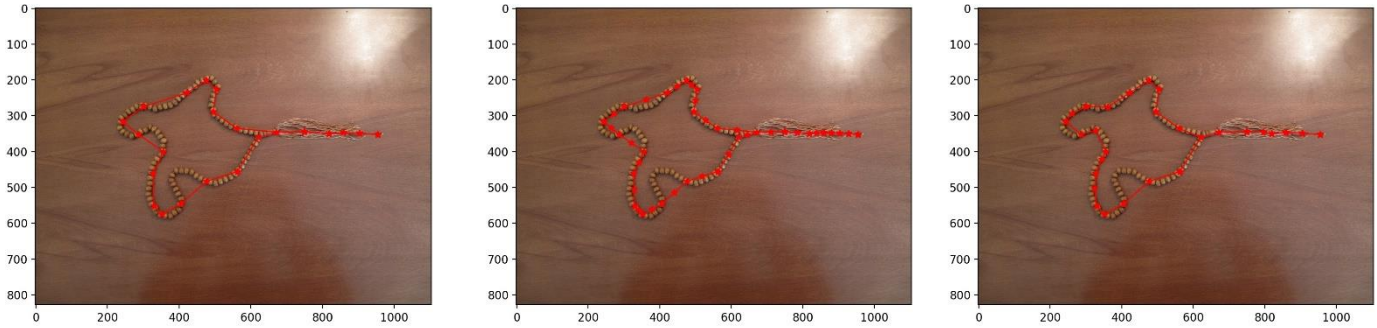


ابتدا توابع سوم و چهارم که گفته شد، غیر فعال هستند. با شروع با ۵۰ نقطه و دو تابع اصلی(اکسترنال و اینترنال) تا جای ممکن کانتور را به عکس نزدیک میکنیم. هنگامی فرا میرسد که در دو پیمایش متوالی، نقاط حرکت نمیکنند. (مثلا شکل روبرو)

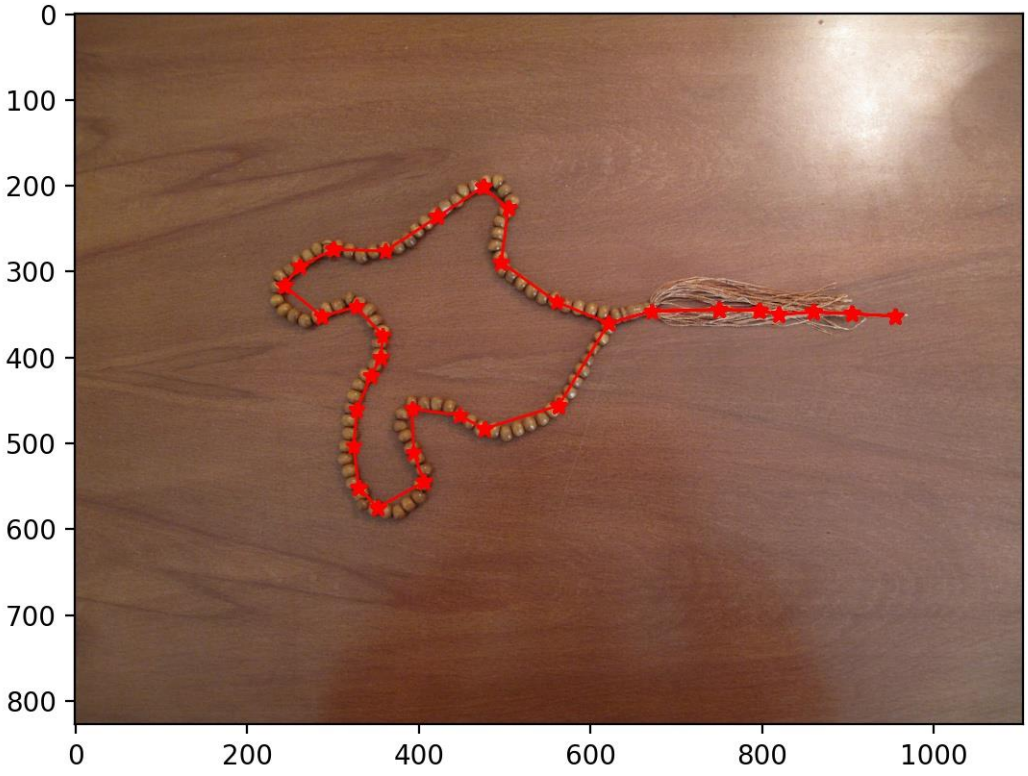
در این هنگام ، ضرایب بتا و لامبدا مروبط به توابع سوم و چهارم را بیشتر میکنیم تا این دو تابع نیز به ما کمک کنند و کانتور بیشتر به عکس بچسبد.(عکس زیر)



هنگامی که به استیت بالا برسیم، برای دومین بار، نقاط مرحله iام و مرحله i+1ام یکسان و بدون تغییر مکانی هستند. در این حالت، تعداد نقاط کانتور را دوبرابر میکنیم(یعنی در وسط هر دو نقطه از کانتور اصلی، یک نقطه دیگر اضافه میکنیم.) آنگاه ضریب گاما (مربوط به تابع اکسترنال) را نیز بیشتر میکنیم تا نقاط اضافه شده به سمت مرز های تسبیح بیشتر کشیده شوند. (به ترتیب از چپ به راست: ۱- عکس اصلی، ۲- بعد از دو برابر شدن نقاط ۳- شکل کانتور نهایی بعد از دوبرابر شدن)



مجددا همین فرایند دوبرابر کردن نقاط رو دو بار دیگر در دومرحله دیگر تکرار میکنیم(مراحلی که در طی آن نقاط دو مرحله پشت سر هم یکسان باشند.) انجام میدهیم تا کل نقاط کانتور ما مجموعا به ۴۰۰ نقطه برسد.



و در نهایت از کل مراحلی که انجام دادیم. یک فایل گیف درست میکنیم و خروجی میدهیم.