

فایل helper.py

همراه تمامی کدها، یک فایل helper.py قرار دارد که به منظور خوانایی کدهاست. تابع هایی که در فایل اصلی کدها هستند (Qi.py) در فایل helper پیاده سازی میشوند تا کد اصلی کوتاه تر و خواناتر باشد. در توضیحات سوال ها هم کدهای اصلی هم تابع های پیاده سازی شده را تا جای ممکن توضیح میدهم.

پس از خواندن عکس، آن را crop میکنم. زیرا محاسبات بسیار طول میکشید و تصمیم گرفتم فقط روی قطعه ای که لازم هست کدم رو اجرا کنم. اگر روی عکس اصلی هم اجرا کنیم همین نتیجه رو خواهیم گرفت (فقط توجه بشه که مختصات پرنده ها باید تغییر بکنه)

پرنده ها خیلی شبیه پس زمینه هستند. در نتیجه اول یه سری تغییرات میدم که عکس بهتر بشه. مثلاً با یه فیلتر شارپش میکنم. بعدش الگوریتم Felzenswalb-Huttenlocher رو از یه لایبرری روش اجرا میکنم. سپس از تابع color.label2rgb استفاده میکنم و عکس label میسازم. در واقع عکسی که هر کلاستر فقط یک رنگ داره (میانگین رنگ پیکسل های داخل یه کلاستر). این عکس segmented_image نام گذاری شده است.

یک خروجی از over-segment اولیه میگیرم تا بتونیم با ریزالت بعد از مرز کردن سوپر پیکسل ها مقایسه کنیم. (عکس over-segmentation.jpg)

سپس نقطه های وسط پرنده ها رو به صورت دستی درآوردم و اون هارو روی عکس با پلات کردن نشون دادم. از این نقطه ها برای مرز سوپر پیکسل ها اجرا کردم. میشد تمام نقطه های عکس رو هم مرز کرد، اما فک کنم اجراش چندروز طول بکشه! تابع merge_super_pixels رو در فایل helper.py پیاده سازی کردم. بعد از مرز کردن پیکسل ها، یک بار با سگمنت های جدید عکس im09 را خروجی کردم و یکبار هم عکس رو کلاستربندی کردم (با رنگ) و با نام im_09_label_image ذخیره کردم. هسته اصلی کد پیاده سازی مرز کردن پیکسل ها بود. این تابع بدین

صورت عمل میکنه :

یک for روی هر کدوم از پرنده ها داره)نقاطی که دستی انتخاب کردیم. میشد دستی انتخاب نکرد و روی همه نقاط پیاده سازی کنیم، اما همونطور که گفتم خیلییی طول میکشید (به ازای هر پرنده)منظور از پرنده پیکسلی است که انتخاب کرده ایم(رنگ کلاسترش رو میگیریم. بعد میایم و پیکسل هایی که تو همسایگی این پیکسل هستن رو نگاه میکنیم.)برای پیدا کردن پیکسل های همسایه یه پیکسل تابع `get_around_pixeld` رو پیاده سازی کردم که کار خاصی نمیکنه :)) اگه این پیکسل ها توی کلاستری متفاوتی نسبت به پیکسل اولیه بودند، میایم چک میکنیم که این دو کلاستر چقدر به هم شبیه هستند. اگه از یه مقدار `threshold` شبیه تر بودند، کلاستری که شبیه به کلاستر پرنده است رو حذف میکنیم و تمام نقاطش رو به کلاستر پرنده میدیم) با `np.vectorize` تمام نقاطی که کلاسترشون شبیه کلاستر پرنده بود رو، کلاسترشون رو کلاستر پرنده قرار میدم)

چون Felzenswalb-Huttenlocher عکس رو به سوپرپیکسل های خیلی خیلی زیادی تقسیم میکنه، الگوریتم تبدیل کردن اون به عکس لیبیل که دوبار در طی کد من استفاده شده است(یکبار در ابتدا برای استفاده در تابع `مرج` و یکبار هم پس از `مرج` برای خروجی گرفتن عکس لیبیل)، خیلی طول میکشه تا کد اجرا بشه و من نتونستم قبل از ساعت ۱۲ خروجی بگیرم و بعد از ۴ ساعت، همچنان خروجی نداده است. اما باتوجه به ریزالت ۵ و اینکه کد `مرج` کردن سوپرپیکسل ها برای این دو کد دقیقا یکی است، انتظار میره که خروجی این کد هم مثل ۵ باشه و سوپر پیکسل ها رو `مرج` کنه، ولی به علت نداشتن زمان نتونستم خروجی بگیرم ولی چون دوتا کد عینا یکسان هست، خروجی این هم باید مثل ۵ باشه تقریبا.