به نام خدا

گزارش پروژه هوش مصنوعی

استاد : دکتر ساجدی

نام و نام خانوادگی : محمدرضا معتبر

شماره دانشجویی : ۶۱۰۳۹۸۱۹۷

خلاصه راه حل:

در پروژه از الگوریتم PSO استفاده شده است به این صورت که موقعیت هر particle یک n_clusters (تعداد cluster هایی که می خواهیم داشته باشیم) تایی مرتب است که هر عضو آن یک رنگ RGB است و نشان دهنده مرکز clusterها و رنگی است که اعضای آن cluster باید در عکس نهایی به آن تغییر پیدا کنند.

هر پیکسل در clusterی قرار میگیرد که کمترین فاصله اقلیدسی را با مرکز آن داشته باشد پس برای یک particle چگونگی عضویت پیکسل ها های تصویر در clusterهای آن پارتیکل مشخص است و هزینه یک particle برابرمجموع فاصله اقلیدسی رنگ هر پیکسل با رنگ نماینده clusterی است که به آن تعلق دارد.

سرعت هر particle نیز برابر یک n_clustersتایی مرتب است که هر عضو آن تغییراتی که در عضو متناظرش در موقعیت آن particle باید ایجاد شود نگه داری می شود.

شرح کلاس particle:

در هر particle , x موقعیت کنونی آن را مشخص میکند و با توجه به modeهای مختلف به طرق مختلفی مقدار دهی اولیه می شود و v نشان دهنده سرعت particle است که در ابتدا مقادیری بسار کوچک دارند (هم مثبت و هم منفی) و n_clusters تعداد مرکز clusterهایی را نشان می دهد که این particle نگه داری می کند و cost برابر هزینه این particle و pbest برابر بهترین جوابی است که این particle از ابتدا تا به حال در خود داشته است و pbest_cost هزینه آن بهترین جواب است.

تابع upd موقعیت کنونی particle را با توجه به سرعتی که دارد تغییر می دهد و cost , pbest, pbest_cost را بهروز رسانی میکند.

تابع normalize نیز رنگ های particle را در محدوده درست نگه می دارد و نمیگذارد یک رنگ بی معنا شود.

شرح توابع:

- تابع dist فاصله اقلیدسی دو بردار را محاسبه می کند در این سوال بردار همان رنگ RGB است.

- تابع fined_nearest_center در بین رنگ های ورودی که مرکز clusterها هستند مرکزی را پیدا می کند که فاصله آن تا یک رنگ مشخص (color ورودی تابع) کمینه باشد و اندیس آن رنگ و فاصله دو رنگ را بر میگرداند.
- تابع segment با ورودی گرفتن یک عکس و مراکز clusterها عکس حاصل از تغییر پیکسل ها به مرکز clusterها را محاسبه می کند.
- در تابع find_imp_colors با ورودی گرفتن یک تصویر رنگ های اساسی آن را محاسبه می کند به این صورت که برای هر سه رنگ قرمز و سبز و آبی قله های موجود در هیستوگرام آن ها را محاسبه می کند (X یک قله است اگر تعداد پیکسل ها با رنگ X از تعداد پیکسل ها با هر رنگ در شعاعی مشخص از X بیشتر باشد) حال همه ترکیب های سه تایی از این قله ها رنگ های اساسی در نظر گرفته می شوند.
- تابع best_colors از بین رنگ هایی که در ورودی میگیرد n_clustersتا از آن ها که اساسی تر هستند را محاسبه می کند به این صورت که به ازای هر پیکسل نزدیک ترین رنگ در رنگ های ورودی به رنگ خود پیکسل را پیدا می کنید و در گروه آن رنگ قرار می دهیم و از بین رنگ های ورودی n_clusters تایی که گروهشان بیشترین عضو را دارند انتخاب می کنیم به روش دومی هم می توانستیم رنگ های خوب را انتخاب کنیم به این صورت که رنگ هایی رو انتخاب کنیم که مجموع فاصله شان از پیکسل های تصویر کمینه باشد ولی طبق نتایج بدست آمده روش اول کمی بهتر عمل می کند
- تابع cal_cost نیز با استفاده از تابع fined_nearest_center هزینه یک particle برای یک تصویر را محاسبه می کند.
- تابع find_best_centers با استفاده از الگوریتم PSO بهترین مرکز clusterها محاسبه شده است و برای مقدار دهی موقعت اولیه particleها از دو تابه find_imp_colors و best_colors کمک گرفته شده است چرا که وقتی رنگ ها کرم و بنفش و ... هستند منطقی است که مرکز کلاستر ها نیز باید حدودا همین رنگ ها باشند و تغییر زیاد در رنگ نهایی وجود ندارد مثلا کرم به قرمز تبدیل نمیشود چرا که هزینه زیادی دارد. در الگوریتم ۲۰ particle و ۲۰ generation در نظر گرفته شده است و تصاویر به ۵ cluster قطعه بندی شده اند(n_clusters = 5).
- تابع compress نیز یک عکس را فشرده میکند به این صورت که پیکسل های موجود در مربع های size در size را با یک پیکسل که رنگش برابر میانگین رنگ آن size * size پیکسل است جایگزین میکند این کار حجم تصویر را بسیار کاهش میدهد و پیمایش روی آن به مراتب سریع تر است و الگوریتم بسیار سریعتر میشود چرا که در الگوریتم بار ها و بار ها روی عکس پیمایش انجام می دهیم از جمله برای محاسبه هزینه. درست است که این کار باعث افزایش قابل توجه سرعت اجرا می شود ولی در نتیجه نهایی تاثیر منفی دارد ولی از آنجا که تاثیر منفی آن کم است ولی افزایش سرعت آن بسیار زیاد است ارزش دارد که تصویر فشرده شود.
- تابع decompress را منبسط می کند به این صورت که بک پیکسل را به size * size تا پیکسل با رنگ همان پیکسل تبدیل می کند که کاربرد خاصی ندارد.
- تابع improve_with_mean رنگ های مرکز clusterها و تصویر را ورودی می گیرد و با کمک تابع find_nearest_center عضویت پیکسل ها در cluster ها را بدست می آورد و رنگ نماینده هر cluster را به میانگین رنگ اعضای آن cluster تغییر می دهد طبق آزمایشات انجام شده این کار تاثیر بسیار خوبی در نتیجه نهایی دارد.

- تابع improve_with_hill اطراف یک جواب (مرکز clusterها) را با روش تپه نوردی کمی جست و جو می کند تا اگر جواب بهتری در نزدیکی جواب وجود داشت آن را به عنوان جواب در نظر بگیریم.
- تابع cal_psnr نیز یک سه تایی محاسبه می کند که هر عضو آن PSNR برای یک از رنگ های قرمز و سبز و آبی است. و PSNR نهایی برابر میانگین این سه PSNRاست.

در نهایت با پیمایش روی تصاویر با شماره start تا end بهترین جواب برای تصویر ها بدست می آید و در فیل با همان اسم و jpg تصویر اصلی کنار تصویر segment شده وجود دارد و در فایل با همان اسم و txt هزینه و ۲۵۵ برابر هزینه و PSNR و مرکز clusterها قرار دارد.

توجه کنید در کد زده شده درجه رنگ ها به جای ۰ تا ۲۵۵ بین ۰ تا ۱ هستند یعنی رنگ هر خانه بر ۲۵۵ تقسیم شده است پس هزینه محاسبه شده با این فرض است و اگر رنگ ها را به درجه اصلیشان برگردانیم هزینه ۲۵۵ برابر می شود.

کد Cal_mean نیز میانگین هزینه و PSNR را برای تمام تصاویر محاسبه می کند.

قطعه بندی هر تصویر به طور میانگین ۵۵ ثانیه با مقدار دهی های داده شده طول میکشد.

و میانگین هزینه ها (در محدوده رنگی ۰ تا ۲۵۵) برابر ۵۹۴۵۷۹.۳۰۵۲ و میانگین PSNR برابر ۱۷.۴۳۳ طبق ارزیابی های تعریف شده است.

چند نمونه قطعه بندی: