گزارش تمرین سری اول

اصول پردازش تصویر

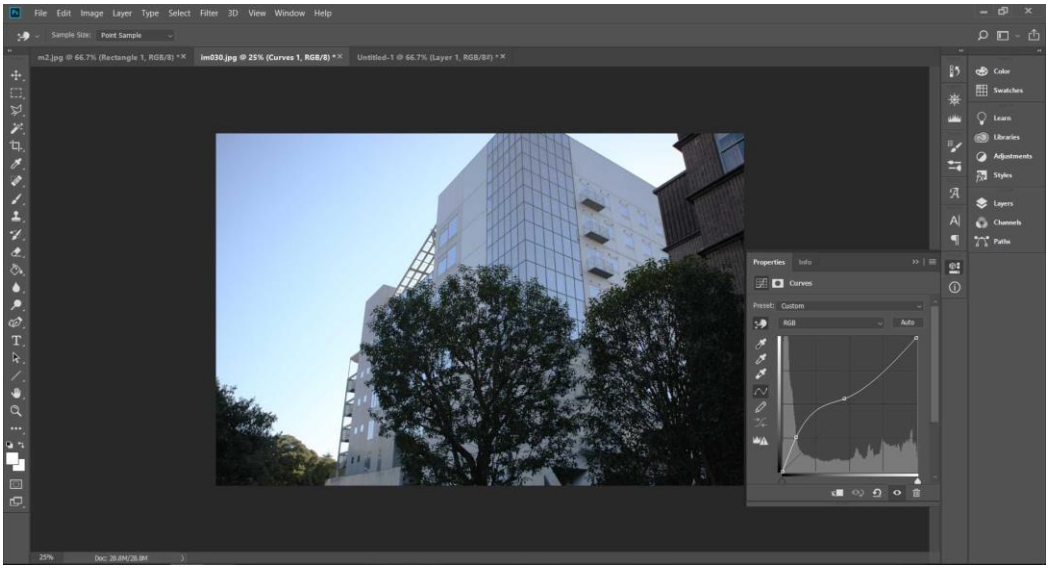
رضا اکبریان بافقی - ۹۵۱۰۰۰۶۱

۱

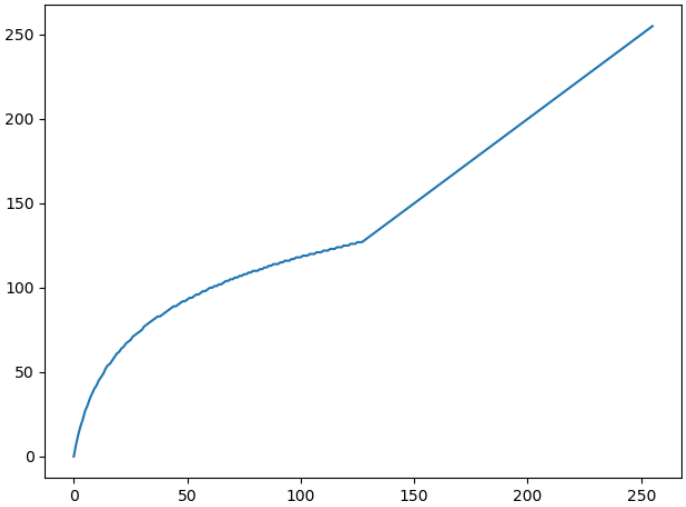
ابتدا در فتوشاپ با تغییر نمودار روشنایی، حالتی که بهترین عملکرد را روی تصویر داشت پیدا کردم (شکل ۱) سپس سعی کردم تابعی شبیه به آن را بسازم (شکل ۲) و در تصویر اعمال کنم. که نتیجه در شکل ۳ قابل مشاهده است.

در این سوال ابتدا از نقطه ۰ تا ۱۲۷ را Log Transformation با ضریب ۰,۲ و از نقطه ۱۲۸ تا ۲۵۵ را همان تابع همانی استفاده کردم.

می‌شد برای اینکه رنگ‌ها طبیعی‌تر باشند، تابع را جوری تنظیم کرد که دارای شکستگی نباشد ولی این تابع دارای شکستگی است. این مشکل را نتوانستم رفع کنم ولی با همین تابع هم دیوارها روشن‌تر شدند.



شکل ۱ – اسکرین شات از فتوشاپ و نموداری که تصویر را بهتر نشان می‌داد.



شکل ۲ – نموداری که روی تصویر اعمال کردم.


شکل ۳ – تصویر اولیه در سمت چپ و تصویر نهایی در سمت راست قرار دارند.

۲

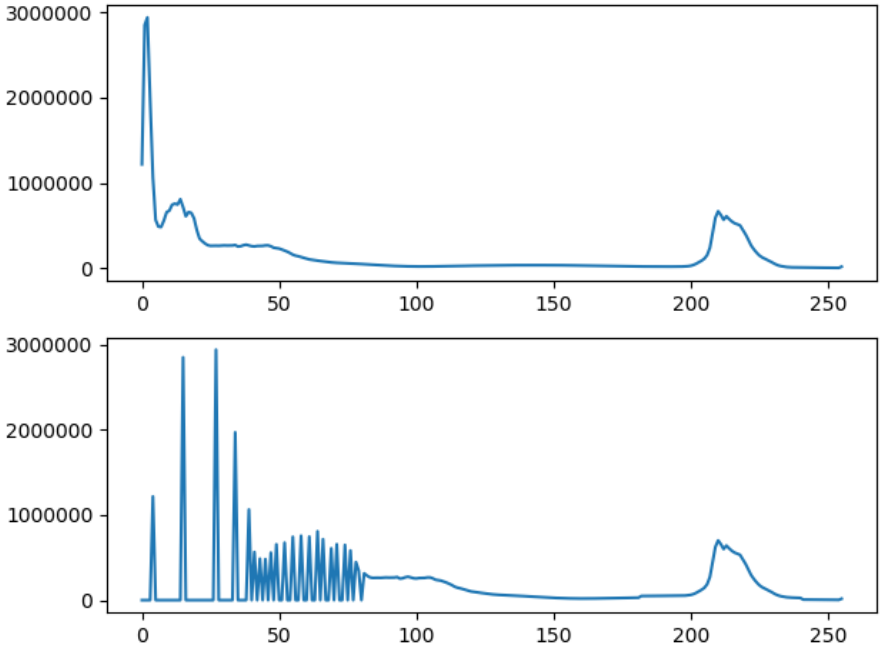
از طریق فرمولی که در اسلایدها توضیح داده شده برای scratch کردن هیستوگرام استفاده شده است به طوری که نمودار هیستوگرام به صورت زیر تغییر پیدا کرده است. (شکل ۴)

در قسمت‌های مختلف کد کامنت گذاشته‌ام که نشان می‌دهد هر قسمت چه کاری را انجام می‌دهد ولی اگر به صورت کلی بخواهم بگویم ابتدا s را مشخص می‌کنم که نشان می‌دهد در هر روشنایی چقدر پیکسل قرار است باشد. (s= ۲۹۱۳۷۳,۲) سپس تابع تجمعی را بر اساس این s تشکیل می‌دهم. سپس تابع تجمعی تصویر اولیه را به دست می‌آورم. حال طبق الگوریتمی که در اسلایدها آمده یک نگارشی به دست می‌آورم که نشان می‌دهد چگونه روشنایی‌های مختلف از ۰ تا ۲۵۵ به روشنایی‌های جدید map می‌شوند.

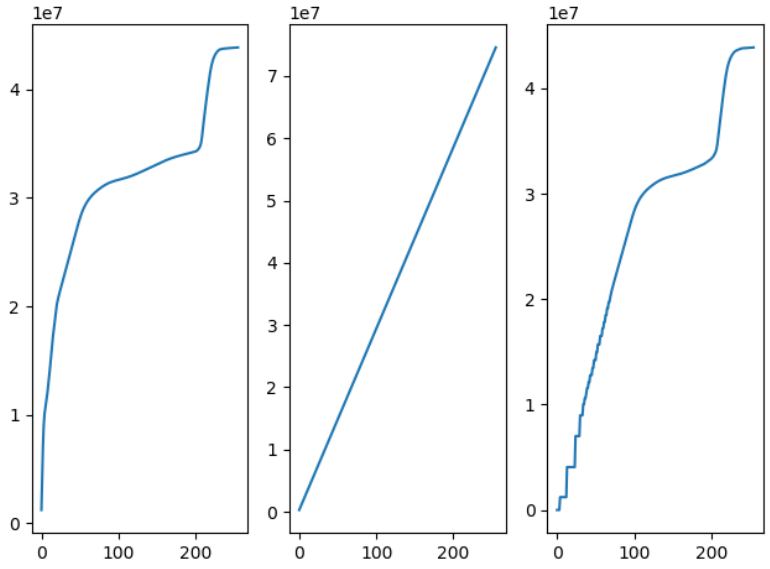
نمودارهای تابع تجعمی که در نظر گرفتیم و تابع تجمعی که در نهایت تشکیل میشود هم در آورده شده‌است.(شکل ۵) در این scratch کردن به دلیل اینکه هدف این بود که تصویر روشن‌تر شود، نقاط با روشنایی بیشتر از ۲۴۰ را به همان صورتی که از قبل بودند، گذاشتم و آن‌ها را تغییر ندادم چون رنگ‌ها در آن حالت به هم می‌ریخت.

به دلیل اینکه می‌خواستم تصویر طبیعی‌تر به نظر برسد هیستوگرام را بدین صورت تغییر دادم با تغییر ضریبی که در s ضرب می‌شود روشنایی تغییر می‌کند.

نتیجه نهایی را می‌توانید در شکل ۶ می‌توانید مشاهده کنید.



شکل ۴ - تصویر بالا هیستوگرام اولیه و تصویر پایین هیستوگرام نهایی را نشان می‌دهد



شکل ۵ - تصویر سمت چپ تابع تجمعی اولیه، تصویر وسط تابع تجمعی در نظر گرفته شده و تصویر سمت راست تابع تجعی نهایی

شکل ۶ – تصویر اولیه در سمت چپ و تصویر نهایی در سمت راست قرار دارند.

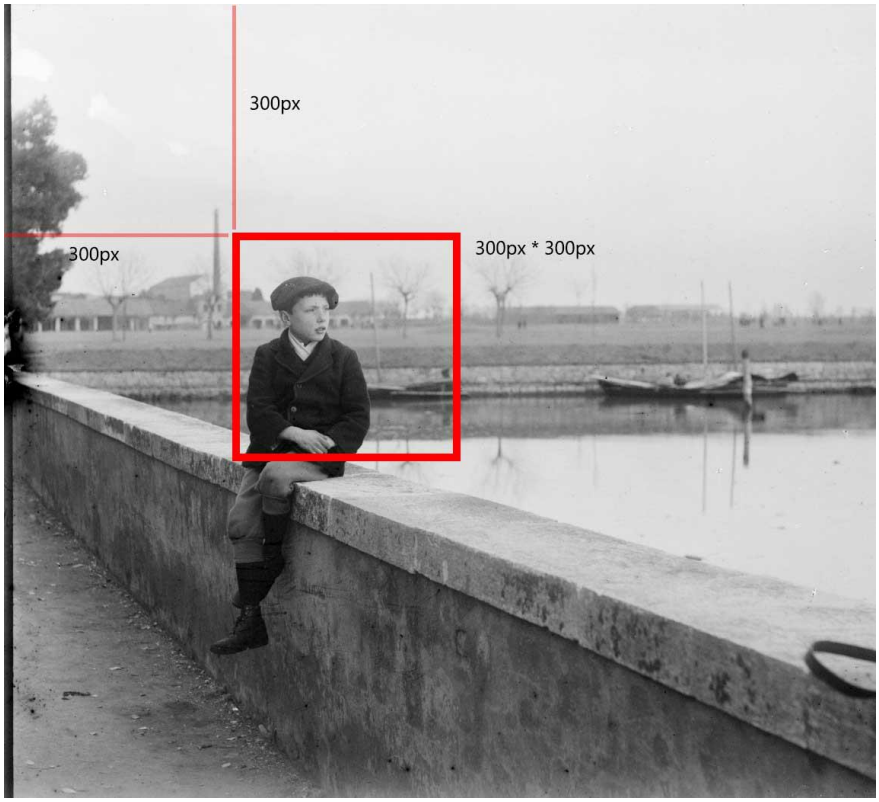
۳

ابتدا سایز عکس را یک سوم می‌کنم بعد از آن حاشیه سفید و تا جایی که ممکن است حاشیه سیاه عکس را می‌برم. سپس ارتفاع عکس را به سه قسمت تقسیم می‌کنم و سه ماتریس که حاوی سه عکس متوالی می‌باشند را تشکیل می‌دهم. (m1,m2,m3)

سپس عکس m1 را پایه قرار داده و به دنبال تطابق عکس‌های دیگر در m1 می‌گردیم. یک مربع کوچک از ماتریس دوم را در نظر می‌گیریم و در جاهای مختلف از ماتریس اول قرار می‌دهیم تا بیشترین تطابق را پیدا کنیم. تابعی که این کار برای ما می‌کند (findMatch) دو ماتریس و سه عدد را می‌گیرد. عدد اول بازه‌ای که این مربع قرار است حرکت کند را مشخص می‌کند ([-a,a]) این حرکت در چهار جهت می‌باشد و در مربعی ۹ برابر مربع اولیه را به دنبال تطابق می‌گردد، عدد دوم در واقع x و y نقطه گوشه بالا راست مربع را می‌گیرد و عدد سوم طول ضلع مربع می‌باشد. در اینجا من مقدار این سه عدد را ۳۰۰ گذاشته‌ام. که در شکل ۷ آمده‌است.

خروجی تابع قبلی به ما می‌گویید که تصویر ثانویه چقدر باید در محور x و y جابجا شود تا روی تصویر اولیه بیافتد. حال تابع دیگری داریم که این جابجایی را انجام می‌دهد (fitPicture) خروجی این تابع ماتریسی هست که آن را من در چنلی که باید قرار بگیرد، می‌گذارم.

در نهایت به صورت دستی حاشیه های اضافی عکس را از عکس حذف می‌کنیم که می‌توانید در شکل ۸ آن را مشاهده کنید.



شکل ۷ - جایگاه اولیه مربع در تصویر دوم

شکل ۷ - تصویر نهایی بعد از حذف حاشیه‌ها