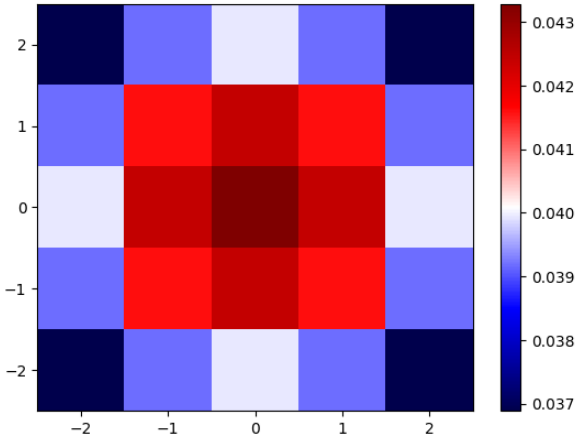
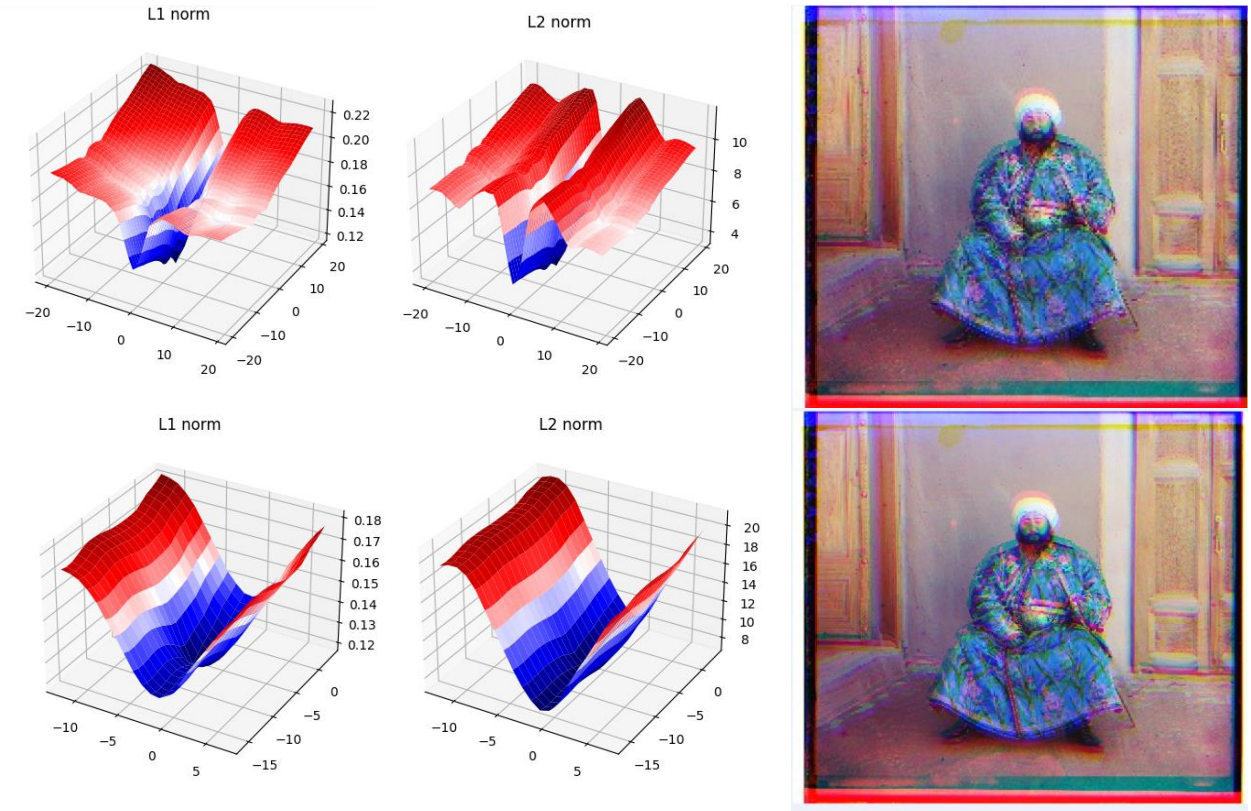
در این سوال الگوریتم پیاده شده را روی یک عکس خاص (امیر بخارا) توضیح می‌دهیم و در انتها تمامی نتایج را می‌آوریم.
ابتدا عکس را از حالت uint16 به float64 تغییر می‌دهیم تا بتوانیم محاسبات اعشاری را به راحتی انجام دهیم. سپس عکس داده شده را که به فرمت tif. است به سه قسمت مساوی تقسیم می‌کنیم که به ترتیب از بالا به پایین نشان دهنده ی کانال آبی و سبز و قرمز است. حال اگر این ۳ عکس را روی هم قرار دهیم نتیجه بصورت زیر است:

همانطور که مشاهده می‌شود کیفیت تصویر اصلا مطلوب نیست و کانال های رنگی روی هم به خوبی قرار نگرفته اند. ابتدا هر سه کانال رنگی را به بازه ی صفر و یک می‌بریم زیرا محاسبه ی اعداد کوچک به نسبت سریعتر است. از آنجایی که کیفیت عکس بالاست، پردازش روی آن بسیار زمانگیر خواهد بود برای همین از image pyramid استفاده می‌کنیم. به این صورت که تا لایه ی 4 پیش می‌رویم و در هر لایه ابعاد عکس را نصف می‌کنیم. بنابراین ابعاد لایه ی fine، 16 برابر ابعاد لایه ی coarse است. برای آنکه در downsampling، aliasing کمتری داشته باشیم، قبل از آنکه رزولوشن عکس را کم کنیم آن را در یک فیلتر گاوسی کانوالو می‌کنیم تا اطلاعات هر پیکسل، به پیکسل های مجاور هم برسد و در حین downsampling اطلاعات کمتر از دست برود. برای این کار از یک فیلتر گاوسی با ابعاد 5×5 با انحراف معیار 5 بصورت زیر استفاده می‌کنیم:



همچنین قبل از آنکه رزولوشن عکس را کم کنیم، از مرز های شکل به اندازه ی 10% ابعاد تصویر را دور می‌ریزیم. این کار هم موجب سریعتر شدن برنامه می‌شود و همچنین مقداری از حاشیه های تصویر درنظر گرفته نمی‌شود و همین امر دقت برنامه را افزایش می‌دهد این دور ریختن تنها حین پردازش انجام می‌شود و هیچ دخالتی در عکس با رزولوشن اصلی نخواهد داشت. در واقع تمامی روش های بریدن و قرار گرفتن سه کانال روی یکدیگر، تنها روی عکس اصلی انجام خواهد شد.

ما کانال سبز (کانال وسط) را به عنوان مبدا در نظر می‌گیریم و دو آبی و قرمز را روی این کانال حرکت می‌دهیم و سطح مشترک بین کانال سبز و یکی از این کانال ها را درنظرمی‌گیریم و فاصله ی بین دو تصویر را با معیار نرم L1 محاسبه می‌کنیم. میزان حرکت دو تصویر را روی هم در لایه ی coarse بازه ی $[-10,10]$ در نظر می‌گیریم و برای لایه های بعدی بازه ی $[-6,6]$ درنظر می‌گیریم. توجه کنید که هرچقدر که سطح مشترک بین دو تصویر کمتر باشد، تعداد نمونه های داخل سطح مشترک نیز کمتر خواهد بود و همین باعث می‌شود که نرم L1 دو تصویر کمتر شود و تابع خطا از حالت محدب بودن خارج شود. برای رفع این مشکل، نرم L1 را بر مساحت سطح مشترک دو مستطیل تقسیم می‌کنیم تا بتوانیم تمام مقادیر را با یکدیگر مقایسه کنیم و به دقت خوبی برسیم. همین امر در رابطه با نرم L2 نیز درست است. وقتی در یک لایه نقطه ی بهینه را برای هر دو کانال آبی و قرمز پیدا کردیم، برای رفتن به لایه ی بعدی این نقاط را در 2 ضرب می‌کنیم (زیرا ابعاد لایه زیرین دو برابر لایه ی بالایی است.) و سپس حول نقطه ی بدست آمده دوباره بهینه سازی را انجام می‌دهیم و نقطه ی بعدی را پیدا می‌کنیم. این روند تا وقتی که به لایه ی fine برسیم ادامه پیدا می‌کند. در زیر عکس با رزولوشن $\frac{1}{16}$ و $\frac{1}{8}$ نرم های L1, L2 ی آن روی کانال آبی رنگ نمایش داده شده است:



همانطور که مشاهده می‌شود تابع خطا در تمامی حالات دارای تحدب است و می‌توان از وجود داشتن مینیمم مطلق مطمئن بود.

پس از اجرای الگوریتم بالا، 4 عدد که میزان شیفت کانال های آبی و قرمز را روی کانال سبز را نشان می‌دهند برگردانده می‌شود. حال باید سطح اشتراک سه کانال رنگی (که هیچ مقداری از مرز های آن را حذف نکرده ایم) را پس از شیفت بدست آوریم. پس از بدست آوردن سطح اشتراک سه کانال در هرلایه تصاویر زیر بدست آمدند (توجه کنید که تصاویر scale شده اند):

همانطور که مشاهده می‌شود کیفیت عکس بسیار مطلوب است و هر سه کانال به خوبی روی یکدیگر قرار گرفته اند. میزان شیفت برای کانال های آبی و قرمز بصورت زیر است:

b: $(-24, -49)$
r: $(17,57)$

برای دو تصویر دیگر نتایج بصورت زیر است:


Аь 132


Аь 132

میزان شیفت برای کانال های آبی و قرمز بصورت زیر است:

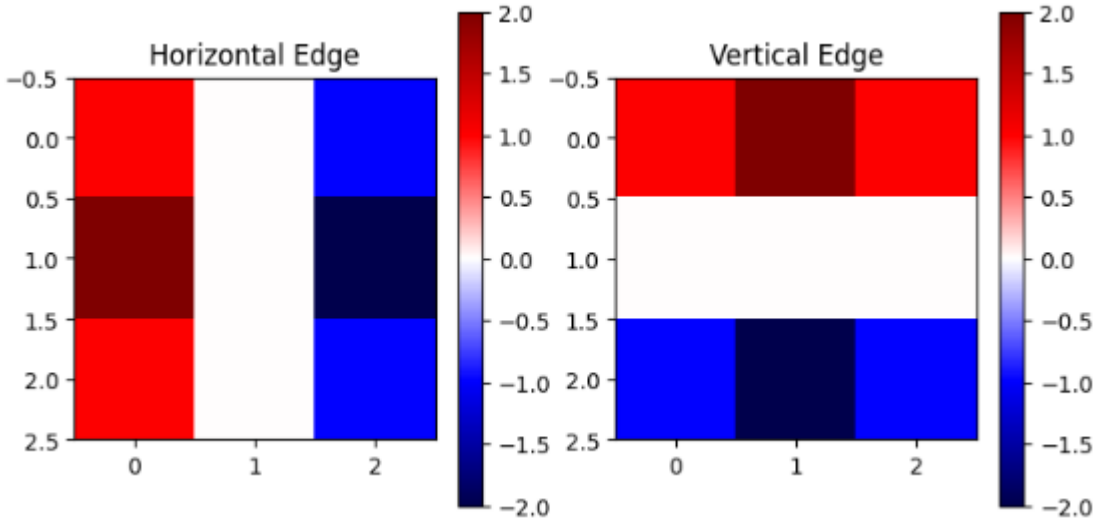
b: $(-6, -43)$
r: $(27, 44)$

میزان شیفت برای کانال های آبی و قرمز بصورت زیر است:

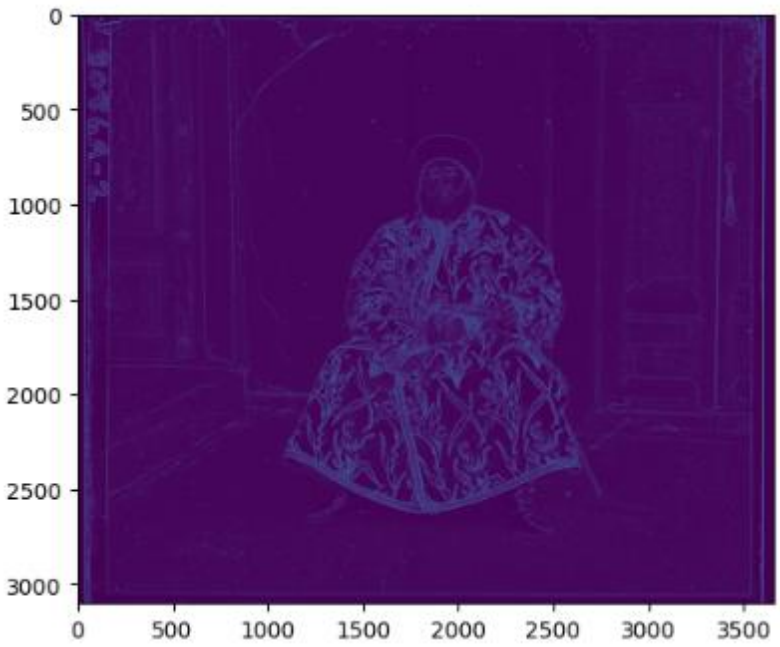
b: $(-10, -56)$
r: $(10, 68)$

در حاشیه های عکس رنگ های نامتعارفی بوجود آمده اند که مطلوب نیستند. این رنگ ها حاصل از روی هم قرار گرفتن بخش های Negative کانال های رنگی است که موجب ایجاد چنین رنگی شده است. خوشبختانه این حاشیه ها دارای الگوری عمودی و افقی هستند و برای حذف آنها از Edge Detection استفاده خواهیم کرد.

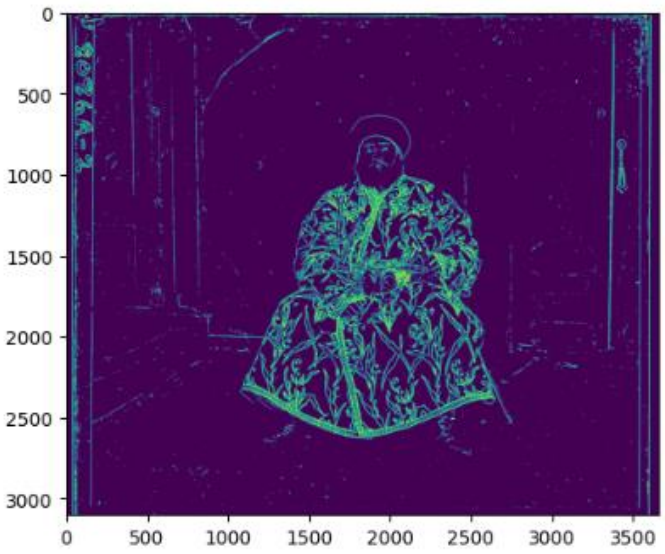
برای این کار از فیلتر Sobel استفاده می‌کنیم به این صورت که در دو راستای افقی و عمودی فیلترهای زیر را اعمال می‌کنیم و سپس اندازه ی اقلیدسی آن را به عنوان محل هایی که Edge تشخیص داده شده اند درنظر می‌گیریم.



حال از رابطه ی $\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$ استفاده می‌کنیم. عکس زیر برای یکی از کانال های رنگی حاصل می‌شود:



اما همانطور که مشاهده می‌شود Edge ها خیلی واضح نیستند و شاید نتوان تنها با پیدا کردن اندازه ی گرادیان، لبه های تصویر را تشخیص داد. برای رفع این مشکل روی درایه های تصویر بالا یک threshold اعمال می‌کنیم بطوریکه برای هر کانال رنگی اگر مقدار پیکسلی بیشتر از 0.1 بود مقدار آن را 1، در غیر این صورت مقدار آن را 0 درنظر بگیرد (درایه های تصویر بین صفر و یک هستند). تصویر حاصل برای یکی از کانال های رنگی نمایش داده شده است:



در تصویر بالا هرجا لبه ای تشخیص داده شده باشد مقدار یک، در غیر این صورت مقدار صفر دارد و همانطور که مشاهده می‌شود لبه های تصویر به خوبی تشخیص داده شده اند. حال برای بریدن تصویر به این صورت عمل می‌کنیم که 300 سطر یا ستون از گوشه های تصویر جلو آمده در این بازه دنبال خط های افقی یا عمودی با درایه ی یک (که نشان دهنده ی لبه ها هستند) می‌گردیم. اگر تعداد درایه های یک خط افقی یا عمودی از $\frac{1}{3}$ ابعاد راستای متناظر آن خط بیشتر باشد، خط را به عنوان مرز در نظر می‌گیریم و میان تمامی این خطوط بدست آمده، آن خطی که داخل‌تر است را به عنوان مرز تصویر جدید در نظر خواهیم گرفت. پس از اعمال الگوریتم بالا تصویر زیر حاصل شد:

همانطور که مشاهده می‌شود لبه های تصویر به خوبی جدا شده اند و تصویر از کیفیت خوبی برخوردار است. در نتایج بدست آمده از چند عکس دیگر را نمایش می‌دهیم.


Аб 132


Аб 132