



دانشگاه صنعتی امیرکبیر
(بی‌تکنیک تهران)

پاسخ تمرین چهارم تصویر پردازی رقمه

استاد درس جناب آقای دکتر رحمتی

نیمسال دوم سال تحصیلی ۱۴۰۰-۱۴۰۱

محسن عبادپور

شماره دانشجویی: ۴۰۰۱۳۱۰۸۰

ایمیل: m.ebadpour@aut.ac.ir

فهرست پاسخ ها

۲	مسئله ۱
۶	مسئله ۲ قسمت (a)
۶	مسئله ۲ قسمت (b)
۷	مسئله ۲ قسمت (c)
۷	مسئله ۲ قسمت (d)
۸	مسئله ۲ قسمت (e)
۹	مسئله ۳ (سه بخش a، b و c باهم)
۱۲	مسئله ۴ قسمت (a)
۱۲	مسئله ۴ قسمت (b)
۱۳	مسئله ۴ قسمت (c)
۱۴	مسئله ۴ قسمت (d)
۱۴	مسئله ۴ قسمت (e)
۱۵	مسئله ۵ قسمت (a)
۱۵	مسئله ۵ قسمت (b)
۱۵	مسئله ۵ قسمت (c)
۱۵	مسئله ۵ قسمت (d)
۱۶	مسئله ۵ قسمت (e)

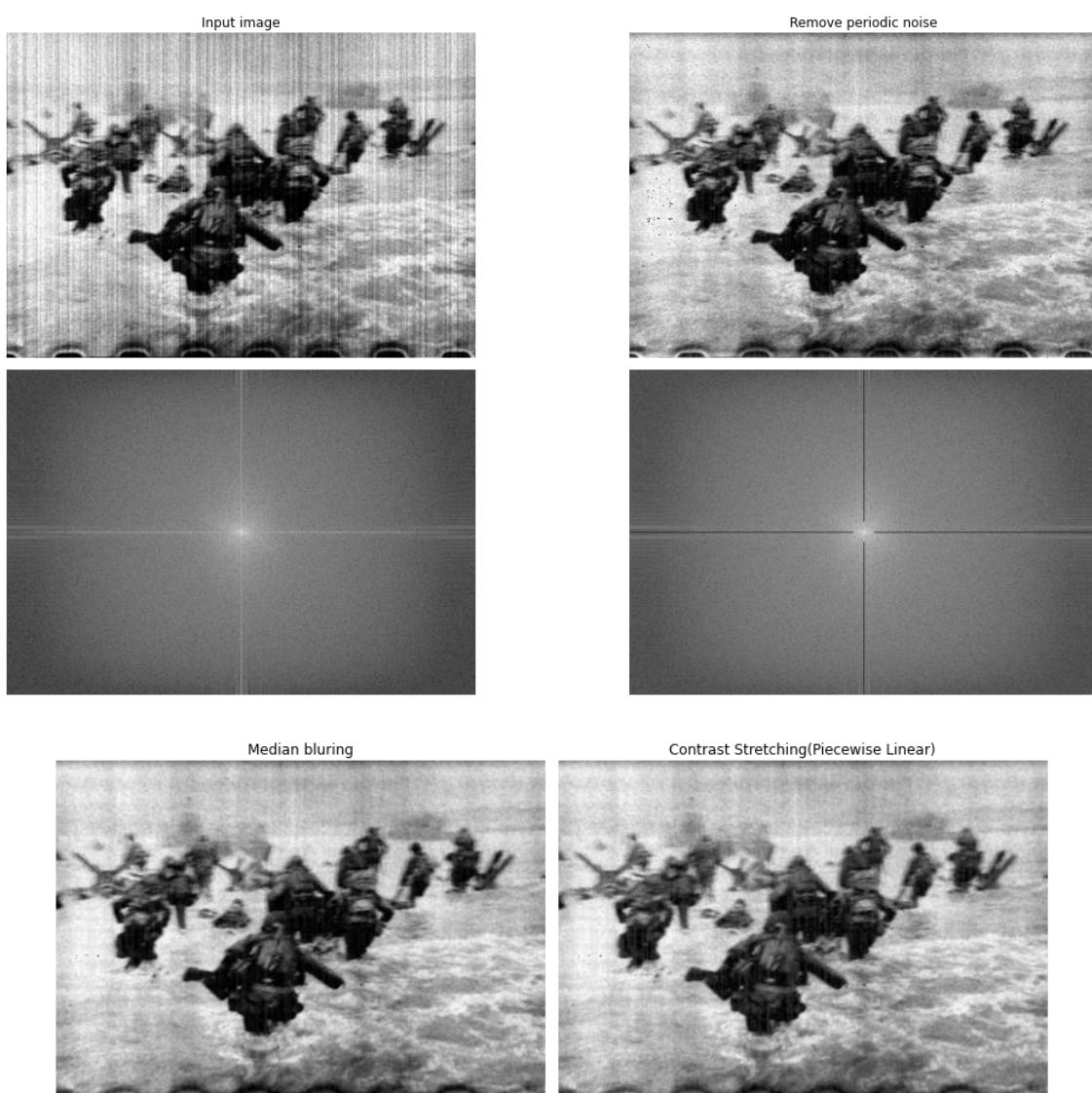
مسئله اول

مسئله ۱

برای هر کدام از تصاویر تخریب شده لیست بهبود ها و تغییرات اعمالی را بررسی کرده و نتایج را مرحله به مرحله تحلیل می کنیم.

عکس اول:

تصویر دارای نویز پریودیک می باشد؛ پس ابتدا برای حذف آن تصویر را به دامنه فوریه برد و اقدام به حذف نویز کرده و تصویر را بازسازی می کنیم؛ تصویر حاصل دارای تعدادی ریزی نویز می باشد که برای حذف آن از فیلتر Median استفاده می کنیم. تصویر حاصل در قسمت های تیره نظیر لباس/کوله سرباز دارای کنتراست بالا می باشد؛ با استفاده از Piecewise Linear فرآیند بهبود کنتراست را انجام میدهیم که در آن کش دادن ناحیه‌ی تاریکی(نژدیک صفر) اتفاق می افتد و کمی نتیجه بهتر حاصل می شود:





عکس دوم:

این تصویر تار شده است؛ پس ابتدا برای بهبود آن از فیلتر لاپلاسین استفاده نموده و تصویر را بازسازی می‌کنیم. تصویر حاصل در قسمت های تیره نظیر لباس/کوله سرباز دارای کنتراست بالا می‌باشد؛ با استفاده از Piecewise Linear فرآیند بهبود کنتراست را انجام میدهیم که در آن کش دادن ناحیه‌ی تاریکی(نزدیک صفر) اتفاق می‌افتد و کمی نتیجه بهتر حاصل می‌شود:



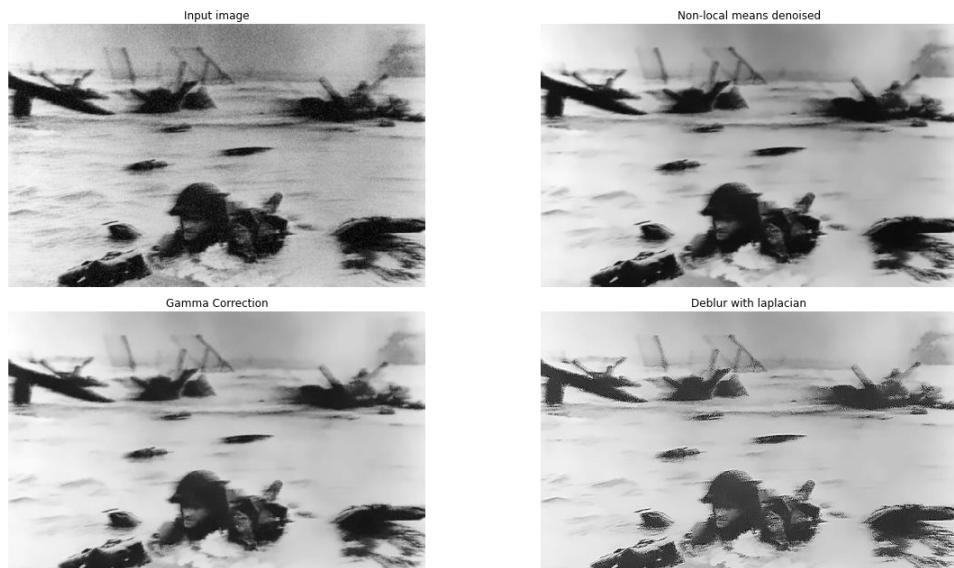
عکس سوم:

این تصویر دارای نویز های فلفلی می‌باشد که طبق بحث های صورت گرفته در کلاس برای بهبود آن می‌توان از فیلتر های median و فیلتر های میانگین گیر غیر محلی برای رفع نویز استفاده نمود که در اینجا از مورد دوم که در اینجا از مورد دوم استفاده شده و نویز ها حذف شده اند؛ در تصویر حاصل آسمان دارای نورپردازی غیریکنواخت حاصل شده است که برای رفع آن در یک حلقه تکراری و پیمایش پیکسل های آسمان، آنها را به یک مقدار ثابت تبدیل می‌کنیم:



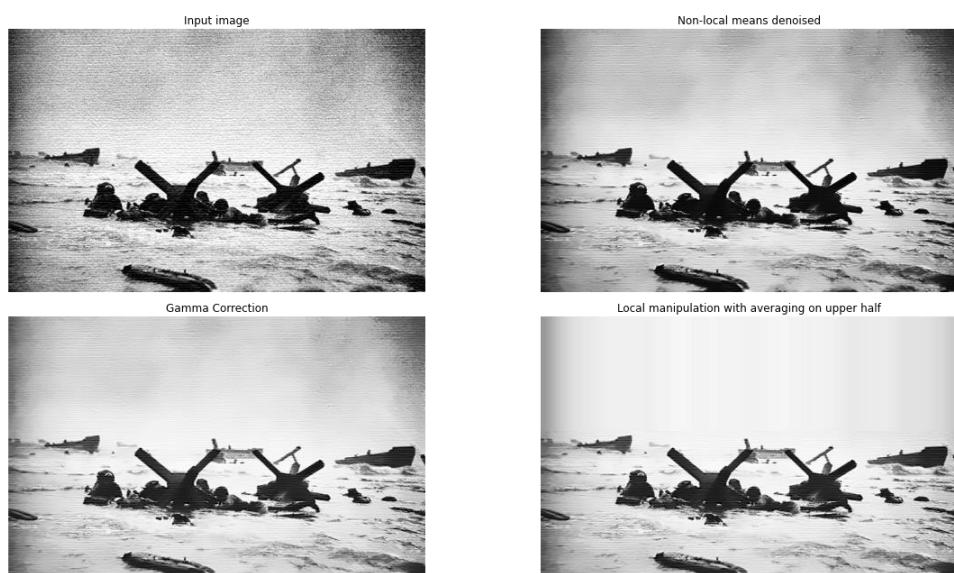
عکس چهارم:

این تصویر که بهبود آن نیز چالشی بود، ابتدا برای رفع نویز های آن از فیلتر میانگین گیر غیر محلی استفاده می کنیم؛ در تصویر حاصل جزئیات موجود در قسمت تاریک بسیار محو می باشد، برای بهبود آن از GammaCorrection استفاده می کنیم (چهره سرباز مقایسه شود). تصویری که ملاحظه می کنیم کمی گویا تار شده است، برای رفع آن نیز از فیلتر لاپلاسین استفاده می کنیم:



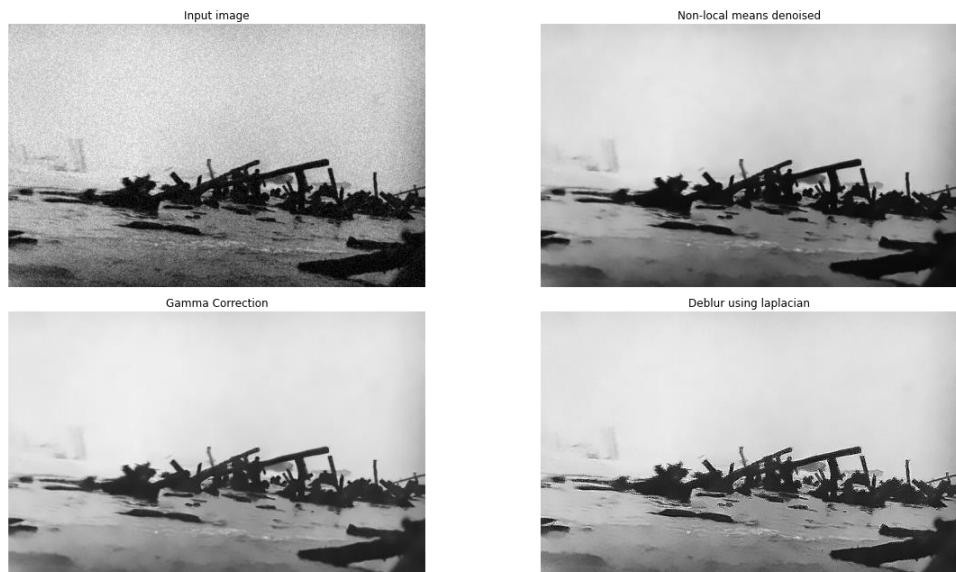
عکس پنجم:

ابتدا برای رفع نویز های آن از فیلتر میانگین گیر غیر محلی استفاده می کنیم؛ در تصویر حاصل جزئیات موجود در قسمت تاریک بسیار محو می باشد، برای بهبود آن از GammaCorrection استفاده می کنیم. در تصویر حاصل آسمان دارای نورپردازی غیریکنواخت حاصل شده است که برای رفع آن یک ردیف پیکسل را در آسمان تکرار کرده و سپس فیلتر میانگین گیر در آسمان اعمال می کنیم تا رنگ پیکسل ها پخش شده و عادی شود:



عکس ششم:

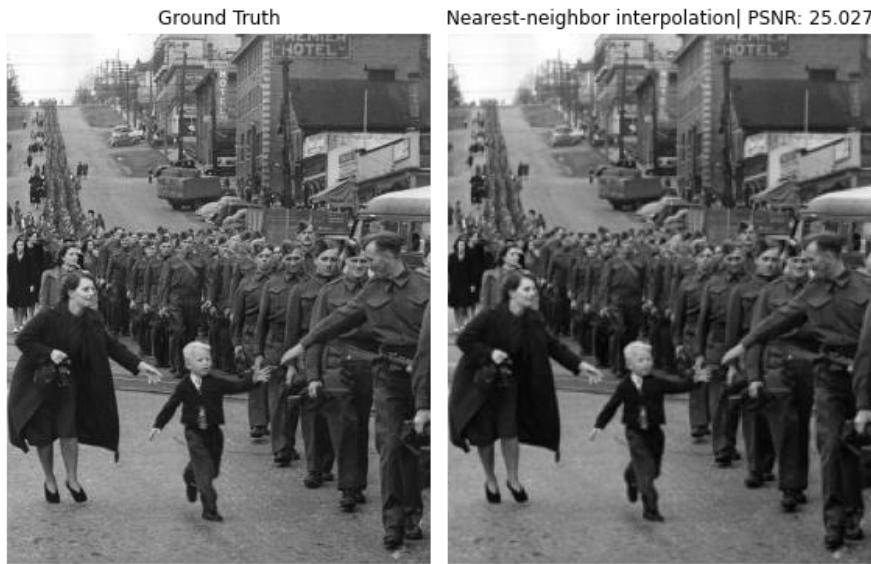
همانند قسمت قبل ابتدا برای رفع نویز های آن از فیلتر میانگین گیر غیر محلی استفاده می کنیم؛ در تصویر حاصل جزئیات موجود در قسمت در تصویر حاصل جزئیات موجود در قسمت تاریک بسیار محو می باشد، برای بهبود آن از GammaCorrection استفاده میکنیم. تصویری که ملاحظه می کنیم کمی گویا تار شده است، برای رفع آن نیز از فیلتر لاپلاسین استفاده می کنیم(مقایسه آب موج دریا):



مسئله دوم

مسئله ۲ قسمت (a)

برای پیاده سازی این قسمت یکتابع با عنوان NearestInterpolation پیاده سازی شده است که تصویر و ضریب تغییر ابعاد(که اینجا دو می باشد) را ورودی گرفته و تصویر بزرگ شده را بازمیگرداند:



مسئله ۲ قسمت (b)

برای پیاده سازی این قسمت یکتابع با عنوان BilinearInterpolation به همراه یک کمکی دیگر پیاده سازی شده است که تصویر و ضریب تغییر ابعاد(که اینجا دو می باشد) را ورودی گرفته و تصویر بزرگ شده را بازمیگرداند؛ ضمناً از تکنیک محاسبه مساحت و وزن دهی به نقاط برای بدست آوردن مقدار درون یابی شده توسط چهار نقطه استفاده شده است:



مسئله ۲ قسمت (c)

برای پیاده سازی این قسمت یک تابع با عنوان NearestValueInterpolation پیاده سازی شده است که تصویر و ضریب تغییر ابعاد(که اینجا دو می باشد) را ورودی گرفته و تصویر بزرگ شده را بازمیگرداند؛ برای محاسبه مقدار درون یابی از تابع پیاده سازی شده در قسمت قبل استفاده شده است. برای این قسمت ابتدا مقدار درون یابی را برای پیکسل متناظر انجام می دهیم و سپس چهار همسایه که دور پیکسل پردازشی تشکیل یک مربع می دهند را در نظر میگیریم؛ همسایه ای که نزدیک ترین مقدار روشنایی به مقدار درون یابی شده را دارا باشد، به عنوان مقدار پیکسل انتخاب می شود:



مسئله ۲ قسمت (d)

برای پیاده سازی این قسمت یک تابع با عنوان NonUniFormInterpolation پیاده سازی شده است؛ برای درون یابی به این روش ابتدا چهار تصویر داده شده را بارگزاری می کنیم؛ سپس در هر مرحله از تعیین سطح روشنایی در تصویر بزرگ، موقعیت پیکسل پردازشی در تصویر کوچک را بدست می آوریم، حال به ازای هر کدام از تصاویر، شیفت متناظر را در موقعیت اعمال کرده و مقدار آن را درون یابی دو خطی می کنیم.

حال چهار مقدار پیش‌بینی شده بدست می‌آید؛ این چهار مقدار را بر اساس معکوس فاصله‌ی هر کدام وزن دار کرده و مقدار پیکسل را پیش‌بینی می‌کنیم و نتیجه بصورت صفحه بعد حاصل می‌شود(وزن دار کردن باعث می‌شود که پیکسل نزدیکتر وزن بیشتر به خود گرفته و تاثیر بیشتری در تعیین سطح روشنایی داشته باشد):



مسئله ۲ قسمت (e)

اولین نکته که بایستی اشاره نمود این است که خروجی پیاده سازی قسمت های مربوط به درون یابی (b, c و d) ممکن است اندکی با حالت ایده آل (ناظیر open-cv) تفاوت داشته باشد که این تفاوت ناشی از درون یابی پیکسل های حاشیه ای تصویر می باشد؛ برای درون یابی دو خطی (رابطه اصلی) ما نیازمند چهار نقطه هستیم که این برای همه پیکسل های تصویر در دست می باشد اما برای پیکسل های حاشیه دقیقاً چهار نقطه نداریم. برای حل این چالش بnde روشنایی پیکسل های موجود در حاشیه را تکرار کرده و فرآیند درون یابی را انجام می دهم و این تخمین ممکن است اندکی در محاسبه مقادیر PSNR تاثیر داشته و مقایسه ما اشتباہ کند با این وجود و آگاه به این موضوع نتایج را بررسی می کنیم.

به جهت بصری، درون یابی NonUniFormInterpolation بهتر از سایر روش های درون یابی عمل کرده است (چشم کودک اندکی قابل تفکیک تر حاصل شده است) و این نشان می دهد داشتن چندین تصویر تخریب شده از یک صحنه ای یکسان میتواند در بازسازی هر چه بهتر آن کمک کند. به جهت مقایسه PSNR در بین سه روش استفاده کننده از درون یابی دو خطی (b, c و d) استفاده می کنند، بهترین نتیجه را کسب کرده است.

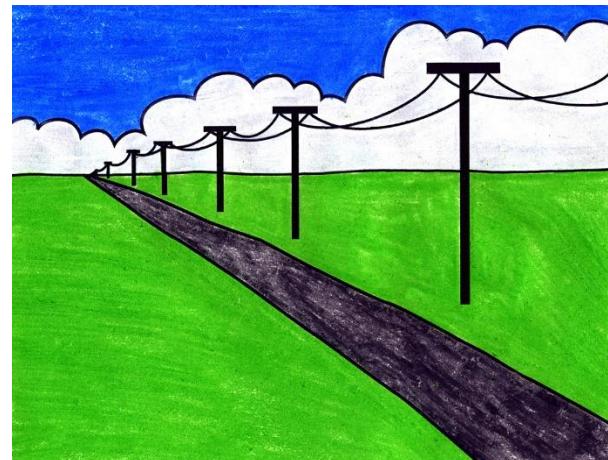
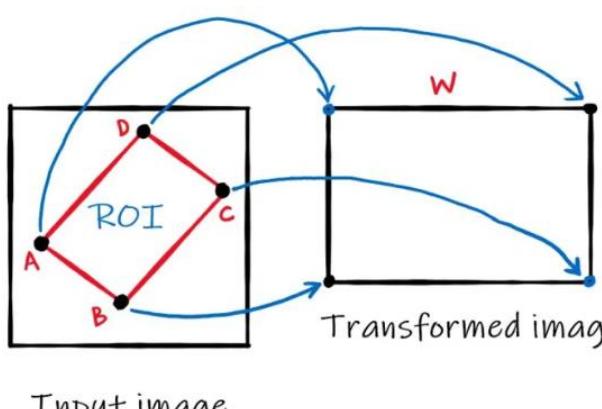
به جهت کمی و معیار PSNR، درون یابی نزدیک ترین همسایه بهترین عملکرد را کسب کرده است اما به جهت بصری و ظاهری نتیجه هی مطلوب را ارائه نکرده است (ناحیه چشم کودک نویزی شده است) و این نشان میدهد که PSNR لزوماً نشان دادن کیفیت ظاهری خوب نیست. درون یابی دو خطی نیز با وجود اینکه PSNR آن نسبت به درون یابی نزدیک ترین همسایه کمتر است اما به جهت بصری نتیجه مطلوب و Smooth تری را ارائه کرده است هر چند عاری از نویز و جزئیات می باشد.

درон یابی NearestValueInterpolation نیز با وجود اینکه کمترین شاخص PSNR را کسب کرده است اما بدترین نتیجه را ارائه نکرده است؛ همانطور که به جهت تنوری این روش که حاصل درون یابی دو خطی و نزدیک ترین همسایه است، به جهت بصری نیز ترکیب این دو روش حاصل شده است. Smooth بودن تصویر را می توان به درون یابی دو خطی و برخی نویز ها که همراه جزئیات می باشد را به درون یابی نزدیک ترین همسایه مربوط دانست.

مسئله سوم

مسئله ۳) سه بخش a. b و c باهم)

زمانی که چشم انسان به یک شی ای نگاه می‌کند که دنباله‌ی آن در امتداد دور شدن از چشم قرار دارد، اندازه‌ی آن در دید ناظر کوچکتر و جهت آن در راستای دور شدن همگرا تر می‌شود؛ به این مورد چشم انداز یا پرسپکتیو گفته می‌شود که در شکل زیر نمونه‌ی آن آورده شده است. در این شکل با وجود اینکه "جاده" یک شکل مستطیلی است ولی از جایی که در چشم ناظر راستای ادامه‌ی آن در حال دور شدن است، شکل آن از حال مستطیلی خارج شده و حاشیه جاده به هم همگرا شده و اندازه آن نیز کوچکتر حاصل شده است.



همانطور که قابل مشاهده می‌باشد، پرسپکتیو را میتوان طی یک تبدیل خطی مدل کرده و تصویر اصلی را بازسازی نمود که صرفاً کافی است ماتریس تبدیل را بدست آورد. از جایی که تصویر مدنظر و انتقالی ما یک مستطیل می‌باشد (قابل دری که صحنه‌ی قتل عام از آن دیده می‌شود)، میتوان آن را با مختصات چهار گوشه نمایش داده و تبدیل را اعمال کرد تا تصویر از رو به رو حاصل شود؛ پس برای بدست آوردن تبدیل هندسی برای هر کدام از تصاویر، بایستی چهار نقطه از صحنه را انتخاب نماییم بطوری که اگر از رو به رو به صحنه نگاه می‌کردیم، آن چهار نقطه همان چهار نقطه گوشی مستطیل میدان دید ما بود.

برای انتخاب مختصات چهار گوشی مدنظر از نرم‌افزار فتوشاپ استفاده شده است. همچنین در انتخاب نقاط نهایت دقت انجام شده است تا بتوان زاویه دید اصلی را بازسازی نمود؛ مثلاً مدنظر گرفته شده است که قسمتی از ناحیه روشن پایین در صحنه، مربوط به افتادن نور در ورودی اتاق می‌باشد و در صورتی که ما مستقیم نگاه کنیم، اصلاً در فضای رو به روی ما قرار ندارد.

پس از استخراج مختصات نقاط انتخابی برای انتقال (نقاط در تصویر گزارشی زیر مشخص شده است) همراه با ابعاد مدنظر برای خروجی به تابع getPerspectiveTransform از کتابخانه open-cv داده شده است تا ماتریس انتقال حاصل شود که برای تصویر اول و دوم بصورت زیر حاصل شده است:

```
1)
[[ -1.53714256e+00 -0.00000000e+00 4.90348476e+02]
 [ 0.00000000e+00 -1.28800252e+00 5.22929023e+02]
 [ 1.26040491e-03 -0.00000000e+00 1.00000000e+00]]
```

2)

```
[ [-1.28481139e+00 3.21202848e-02 4.13066863e+02]
 [ 6.61362832e-02 -1.03833965e+00 3.72677956e+02]
 [ 2.40495575e-04 -2.23854859e-04 1.00000000e+00] ]
```

حال که ماتریس تبدیل برای هر کدام از تصاویر بدست آمده است، با استفاده از warpPerspective از open-cv تصویر را بازسازی می‌کنیم. تصاویر بدست آمده دارای نویز هایی هستند(حاصل از درون یابی موقع تبدیل) که برای رفع آن از میانگین‌گیر غیر محلی استفاده میکنیم؛ تصویر حاصل دارای تاری می‌باشد که برای رفع آن نیز از فیلتر لاپلاسین استفاده میکنیم. تصویر نهایی اندکی در جزئیات تیره ضعیف می‌باشد که از Gamma Correction برای بهبود سطح روشنایی استفاده می‌کنیم تا نتیجه نهایی حاصل شود.

گزارش زیر مربوط به تصاویر نهایی در کنار تصاویر ورودی قابل ملاحظه بوده و مراحل انجام هر یک نیز در صفحه بعد و بصورت تسلیسل نمایش داده شده است(از انتخاب نقطه برای تبدیل تا بازسازی نهایی).

Orginal Input



Gamma Correction



Orginal Input



Gamma Correction



Orginal Input



Selected points to transform



Geometric Transformations



Denoise with NLMean



Deblur with laplacian



Gamma Correction



Orginal Input



Selected points to transform



Geometric Transformations



Denoise with NLMean



Deblur with laplacian



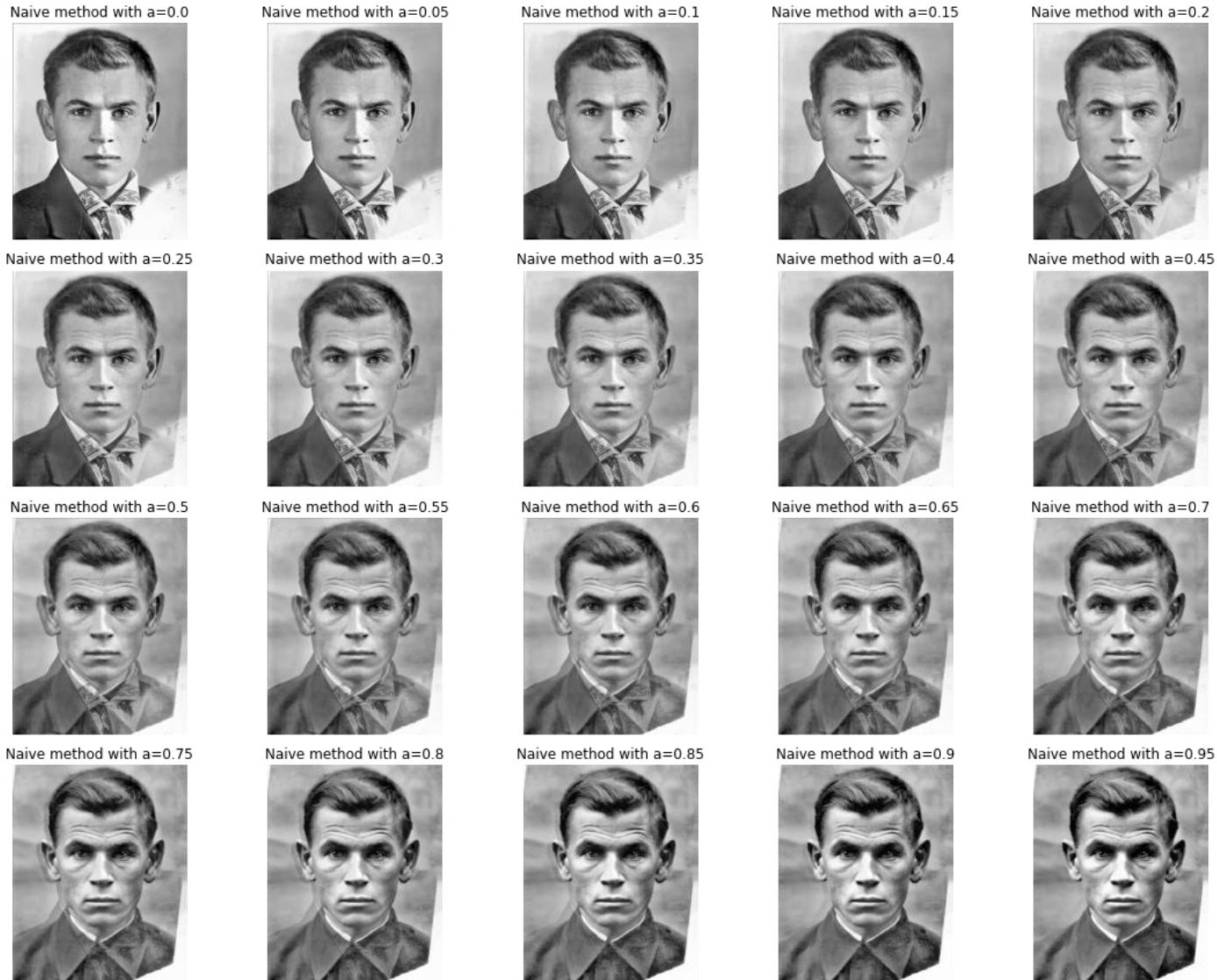
Gamma Correction



مسئله چهارم

مسئله ۴ قسمت (a)

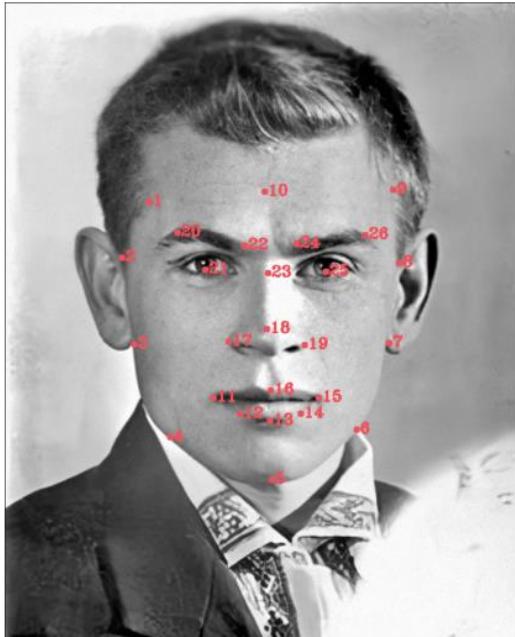
برای پاسخ به این قسمت یک تابع با عنوان NaiveMethod پیاده سازی شده است؛ نتایج حاصل بصورت زیر آماده شده است. ضمناً یک ویدیوی دو ثانیه‌ای نیز برای این بخش تولید گردیده و در پوشه نتایج قرار داده شده است:



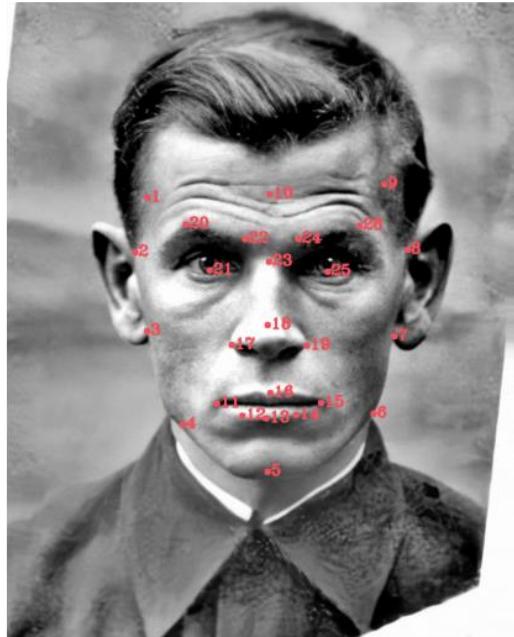
مسئله ۴ قسمت (b)

برای انجام این بخش، تصویر هر کدام از سال های ۱۹۴۱ و ۱۹۴۵ به کاربر نمایش داده شده و کاربر ۲۶ نقطه را برای هر یک انتخاب می‌کند. انتخاب این نقاط می‌تواند بیشتر یا کمتر باشد و صرفاً کافی است تا پارامتر ارسالی به تابع point_picker تعیین یابد. نقاط انتخابی طبق نمونه نقاطی که تصویر هیتلر مشخص شده بود، انتخاب گردیده است. برای صرفه جویی در وقت، نقاطی انتخابی در قالب آرایه های numpy ذخیره شده و برای بعد صرفاً بارگزاری می‌شود. نقاط انتخابی بصورت زیر حاصل شده است:

Selected keypoints for image 1941



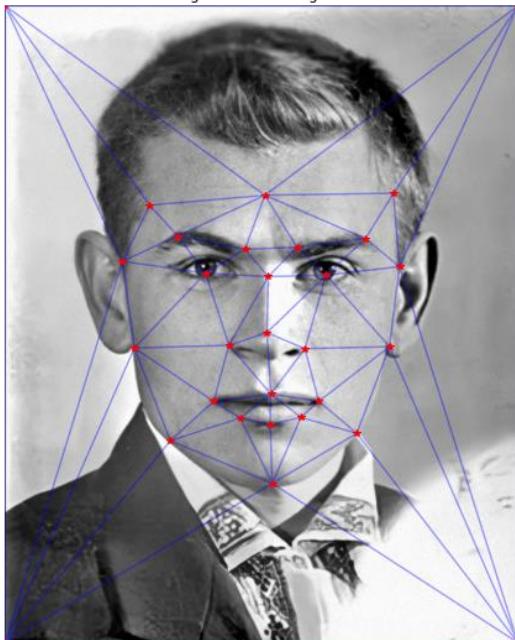
Selected keypoints for image 1945



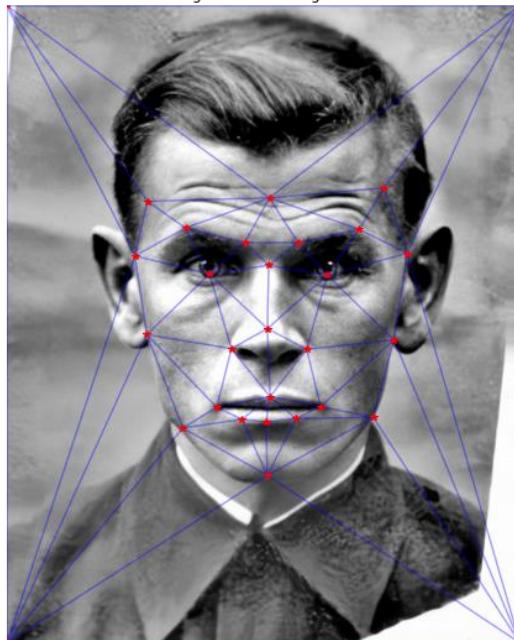
مسئله ۴ قسمت (c)

فرآیند ایجاد مثلث توسط نقاط انتخابی از کلاس Delaunay از کتابخانه `scipy` انجام شده است. این کلاس لیست نقاط مدنظر را ورودی گرفته و در فیلد `simplices` اش نشان می‌دهد که چه ترکیب‌های سه تایی از نقاط مدنظر تشکیل یک مثلث را می‌دهند؛ با استفاده از این کلاس ایجاد مثلث انجام شده و نتایج آن بصورت زیر حاصل شده است:

Triangulation for image 1941



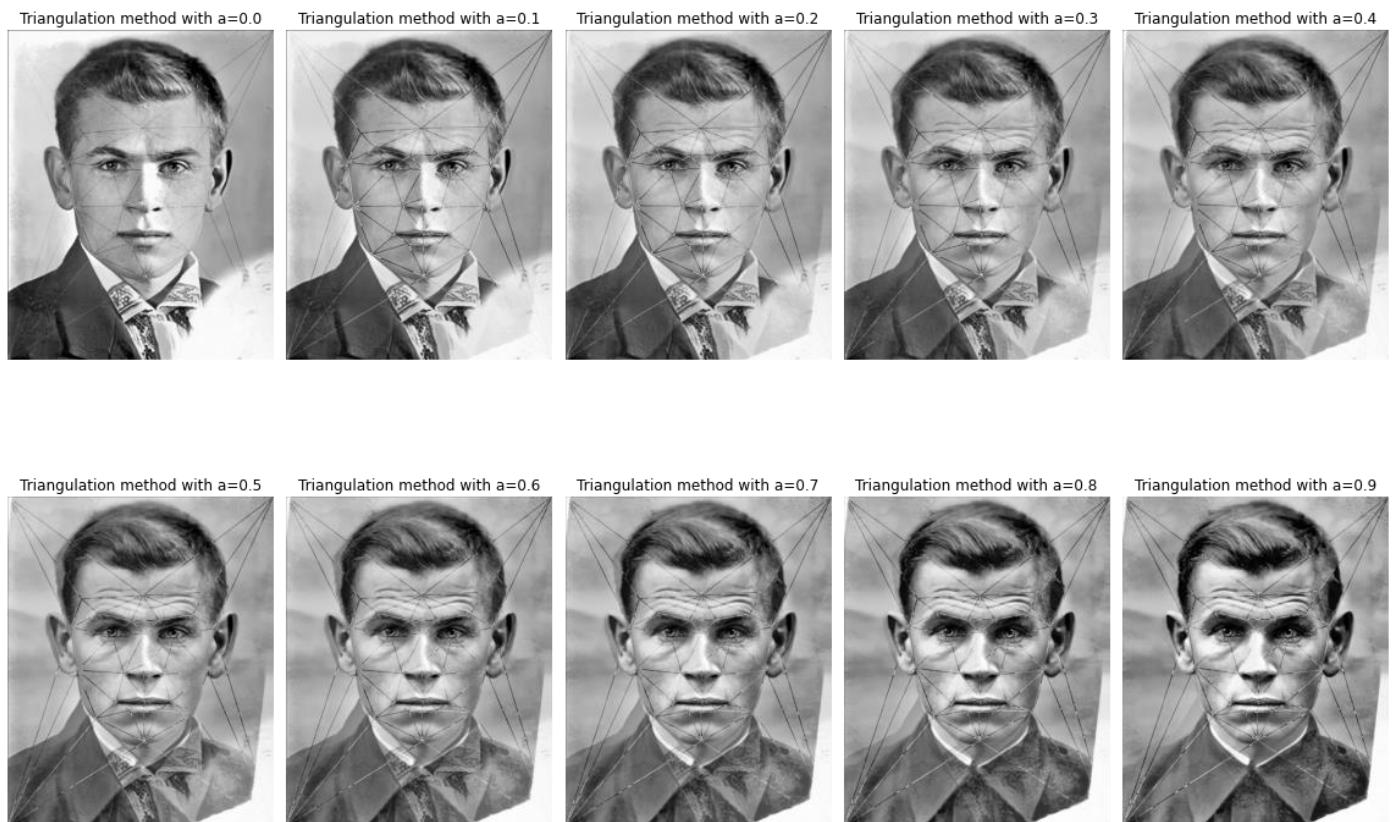
Triangulation for image 1945



مسئله ۴ قسمت (d)

برای حل این قسمت و انتقال تصویر توسط روش مثلث، دو تابع اصلی پیاده سازی شده است. تابع اولی `get_single_frame` نام دارد؛ این تابع نقاط مربوط به تصویر ورودی (M/N) را به همراه نقاط مربوط به تصویر I دریافت می کند، سپس هر یک از مثلث ها را انتقال میدهد و تصویر نهایی را بر می گرداند. (تفکیک نواحی مشابه با استفاده از تکنیک masking کنترل می شود.)

تابع دوم `get_single_image` نام دارد که یک ضریب a به همراه نقاط انتخابی دو تصویر را دریافت می کند و سپس نقاط مربوط به تصویر I را بر اساس a ورودی محاسبه می کند؛ سپس هر یک از تصاویر M/N را توسط تابع اول انتقال داده و نتایج را مانند `naive` ترکیب کرده و عکس نهایی را باز می گرداند. نتایج انتقال بصورت زیر حاصل شده است:



فرآیند انتقال بین تصاویر در روش مثلثی، بسیار روان تر و smooth تر از روش `naive` انجام می شود و به ناظر دید realistic تری منتقل کرده و از ایجاد فضای سایه ای و تار مانند ابر جلوگیری می کند لذا این روش بهتر از روش قسمت a می باشد؛ دلیل بهتر بودن آن است که جزئیات مهم و معنادار چهره نظریر چشم ها، گودی صورت و... (نقاط انتخابی) بطور خطی منتقل می شود و این باعث کاربر چندان متوجه جا به جایی تیز نشود.

مسئله ۴ قسمت (e)

برای حل این بخش از توابع پیاده سازی شده در بخش d استفاده شده و ویدیوی خواسته شده تولید گردیده است. ویدیو در پوششی خروجی ها قرار داده شده است.

مسئله پنجم

مسئله ۵ قسمت (a)

(۵)

برای هر کدام از تبدیل های خطوط استدباره $y = mx + b$ متریس تبدیل معرفاً نوشت و آنها را ب متریس T ضمیمه کنید تا بین آنها باز هم متریس در گذشتگر متریس متریس خواسته شده $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ باشند (عمل جزو).

$$T_1 = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}$$

$$T_2 = \begin{bmatrix} \cos(30) & \sin(30) & 0 \\ -\sin(30) & \cos(30) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_3 = \begin{bmatrix} 1 & 0 & 20 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T = T_1 T_2 T_3$$

مسئله ۵ قسمت (b)

سطر پایانی ماتریس انتقال affine باعث ضرب پذیری یک بردار ورودی در ماتریس تبدیل می شود چون در غیر اینصورت و بر اساس جبر خطی نمیتوان تبدیل را اعمال نمود؛ به عبارتی دیگر آن سطر کنترل می کند که نقطه ورودی در یک صفحه دو بعدی، به یک نقطه‌ی دیگر در همان صفحه نگاشت شود.(صفحه 1)

مسئله ۶ قسمت (c)

برای این بخش بایستی نوع درون یابی در upscale/downcale کردن را در نظر گرفت و سپس به این پرسش پاسخ داد. برای پاسخ به این سوال بnde این فرآیند را با چندین روش درون یابی فرآیند مذکور را بر روی چندین تصویر ورودی آزمایش کرد؛ اگر روش درون یابی مبتنی بر تخمین و محاسبه باشد، به جهت عددی تصویر همگرا نمی شود چرا که در هر مرحله از محاسبه، یک تخمین و محاسبه اعشاری صورت میگیرد و از جایی که این اعشار محدود می شود لذا بازسازی آن دقیق رخ نمی دهد و تصویر دقیقا همگرا نمی شود. در مقابل برای روش درون یابی با نزدیک ترین همسایه، تصویر همگرا می شود چرا که مقدار روشنایی پیکسل محاسبه و تخمین نشده و خطا و تقریبی نیز وجود ندارد و صرفا مقدار آن از همسایه برداشته می شود و تصویر به جهت عددی نیز همگرا می شود. لذا بصورت کلی می توان گفت که به روش درون یابی بستگی دارد و میتوان همگرا شود یا نشود.

مسئله ۶ قسمت (d)

خیر، همه تبدیل های هندسی معکوس/بازگشت پذیر نمی باشد چرا که شکل و تعداد پیکسل ها و نواحی انتقال میتواند تغییر یابد و وقتی تعداد پیکسل ها تغییر می یابد بایستی برای پیکسل های جدید در موقعیت جدید مقدار جدید درون یابی نمود و از جایی درون یابی عملی تخمینی می باشد و دقیقا نمیتوان مقدار قبلی را بازیابی نمود، بازگشت پذیری کامل نیز در همه تبدیل های هندسی وجود ندارد. یک نمونه از همین تبدیل های هندسی که امکان معکوس گرفتن دقیق وجود ندارد، تبدیل پرسپکتیو می باشد که در سوال سه مورد استفاده قرار گرفته است.

نویز های جمع شونده نویز هایی هستند که مقدار نویز اعمالی مستقل از محثوا و سطح روشنایی پیکسل های تصویر ورودی می باشد و در مقابل نویز های ضرب شونده نویزهایی هستند که بر اساس سطح روشنایی هر پیکسل، نویز اعمالی متفاوت است؛ نویز های نمک-فلفلی به صورت تصادفی به پیکسل های تصویر ورودی اعمال و افزوده می شود(۰/۲۵۵) و مقدار پیکسل های تصویر ورودی تاثیری در چگونگی نویز اعمالی ندارد لذا نویز های نمک-فلفلی یک نویز جمع شونده می باشد. نویز های پریودیک نیز نویز های جمع شونده هستند چرا که توزیع آن ربطی به مقادیر تصویر ورودی و روشنایی تک تک پیکسل ها ندارد؛ دقیقاً به همین علت است که میتوان نویز های پریودیک را در دامنه فرکانس حذف نمود. به عبارتی دیگر نویز پریودیک را می توان یک نویز جمع شونده دانست که مقدار آن در هر موقعیت نسبت به ابعاد تصویر فرق داشته و مانند نویز نمک فلفلی تصادفی نمی باشد بلکه از یک تابع (مثالاً سینوس) پیروی می کند.