



دانشگاه صنعتی امیرکبیر
(پای تکنیک تهران)

پاسخ تمرین دوم تصویر پردازی رقمی

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
۴	مسئله ۲ قسمت (a)
۵	مسئله ۲ قسمت (b)
۶	مسئله ۲ قسمت c و (d)
۸	مسئله ۳ قسمت (a)
۹	مسئله ۳ قسمت (b)
۱۰	مسئله ۳ قسمت (c)
۱۲	مسئله ۳ قسمت (d)
۱۳	مسئله ۳ قسمت (e)
۱۷	مسئله ۳ قسمت (f)
۲۱	مسئله ۴ قسمت (a)
۲۱	مسئله ۴ قسمت (b)
۲۳	مسئله ۴ قسمت c و (d)
۲۴	مسئله ۴ قسمت (e)
۲۵	مسئله ۵ قسمت (a)
۳۰	مسئله ۵ قسمت (b)
۳۴	مسئله ۵ قسمت c و (d)
۳۷	مسئله ۵ قسمت (e)
۵۰	مسئله ۵ قسمت (f)
۵۳	مسئله ۶ قسمت (a)
۵۳	مسئله ۶ قسمت (b)
۵۳	مسئله ۶ قسمت (c)
۵۴	مسئله ۶ قسمت (d)
۵۴	مسئله ۶ قسمت (e)

مسئلہ اول

(1) ←

Subject:
Year: Month: Date:

- ② To obtain negative of image we only need to subtract image from full value of possible where in this case

is $7 = 2^3 - 1$

2	1	2	2	5	6	6
3	3	2	3	2	4	5
2	2	4	2	1	4	5
3	2	4	3	4	2	4
3	3	5	4	1	1	2
2	3	5	1	1	1	4
2	2	3	3	2	3	4
5	5	4	5	6	5	6

- ③ we use following formula to obtain bit-plane:

$$a_k = [I] \bmod 2 \rightarrow \text{indicated in Problem 2}$$

bit-plane 1	bit-plane 2	bit-plane 0
0 1 0 1 0 1 0 0	1 1 1 1 1 0 0 0	1 0 1 0 1 0 1 1
0 0 0 0 1 1 1 1	1 1 1 1 1 0 0 0	0 0 1 0 1 1 0 0
0 0 1 0 1 1 1 1	1 1 0 1 1 0 0 0	1 1 1 1 0 1 0 0
0 0 1 0 1 0 1 1	1 1 0 1 0 1 0 0	0 1 1 0 1 1 1 1
0 0 1 1 1 1 0 1	1 1 0 0 1 1 1 0	0 0 0 1 0 0 1 1
0 0 1 1 1 1 1 1	1 1 0 1 1 1 1 0	1 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1	1 1 1 1 1 1 1 0	1 1 0 0 1 0 0 1
1 1 1 1 1 0 1 0	0 0 0 0 0 0 0 0	0 0 1 1 0 1 0 1
0 0 0 0 0 0 0 0	0 0 1 1 1 1 1 0	1 0 0 0 0 0 0 0
1 0 1 1 1 1 1 1	1 1 0 0 0 0 0 0	1 1 1 1 1 1 1 1

$0 \rightarrow 0, 001 \rightarrow 1, 010 \rightarrow 2, 011 \rightarrow 3, 100 \rightarrow 4$

$101 \rightarrow 5, 110 \rightarrow 6, 111 \rightarrow 7$

Subject:
Year: Month: Date:

- ④ we use the following formula to map input image pixels

Value to output image pixels values:

$$\text{output} = (\text{input} - \text{minInput}) * \frac{\text{maxOutput} - \text{minOutput}}{\text{maxInput} - \text{minInput}} + \text{minOutput}$$

new obtained image:

204	255	204	255	204	51	0	0
153	153	204	153	204	102	51	51
204	204	102	204	255	102	51	51
153	204	102	153	204	102	204	102
153	153	51	202	255	255	204	102
204	153	51	255	255	255	255	102
204	204	153	153	204	153	153	102
51	51	102	102	51	0	51	0

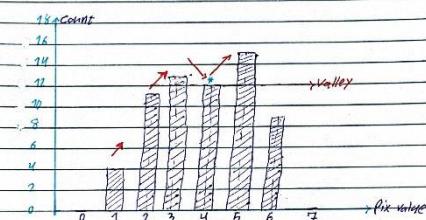
- ⑤ First we make frequency table and its cumulative, then multiply it by 7 (max value of range) and make it discrete to obtain new values

Value	Count	Cumulative Count	Cumulative Relative	New Value
0	0	0	0	x7 0
1	4	4	0.06 x7 0	
2	11	15	0.23 x7 1	
3	13	28	0.43 x7 3	
4	12	40	0.62 x7 4	
5	15	55	0.85 x7 6	
6	9	64	1 x7 7	
7	0	64	1 x7 7	

Subject:
Year: Month: Date:

- ⑥ we count the seen values and then chart the histogram

$0 \rightarrow 0, 1 \rightarrow 4, 2 \rightarrow 11, 3 \rightarrow 13, 4 \rightarrow 12, 5 \rightarrow 15, 6 \rightarrow 9, 7 \rightarrow 0$



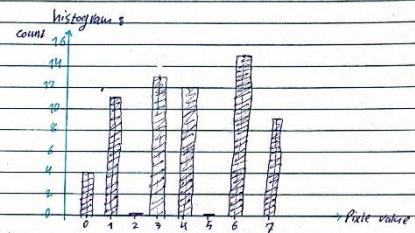
- ⑦ we see that we have a Valley in value of 5 in above

histogram chart, so we set threshold in this point and make all smaller and equal values to 0 and one to 7

7	7	7	7	7	0	0
0	0	7	0	7	0	0
7	7	0	7	7	0	0
0	7	0	0	7	0	0
0	0	0	0	7	7	0
7	0	0	7	7	0	0
7	7	0	0	0	0	0
0	0	0	0	0	0	0

Subject:
Year: Month: Date:

- ⑧ continue... now we map pixels values and chart



- ⑨ First, Same to f we obtain equalized histogram of given

histogram, then map two equalized histograms to each other

Value	Count	Cumulative Count	Cumulative Relative	New Value
0	0	0	0	x7 0
1	0	0	0	x7 0
2	512	512	0.05 x7 4	
3	0	0	0.5 x7 4	
4	0	0	0.5 x7 4	
5	0	0	0.5 x7 4	
6	512	1024	1 x7 7	
7	0	1024	1 x7 7	

(4) for obtain desired transformation, we use following integral that mentioned during class and DIP book:

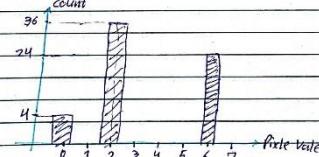
$$\begin{aligned} S = T(r) &= (L-1) \int_0^r P(w) dw = (2-1) \int_0^r \frac{e^{-2w}}{e-1} dw \\ &= 1 \left[\frac{e^{-2w}}{e-1} \right]_0^r = \frac{e^{-2r} - e^0}{e-1} = \frac{e^{-2r} - 1}{e-1} \\ &= \frac{e^{-2r} - 1}{e-1} = \boxed{\frac{e^{-2r}-1}{e-1}} \end{aligned}$$

(5) continue now we chart a table to map our input image value to desired based on equalized form of them.

input value	equalized input	equalized output	output value
0	0	②	② 0
1	0	0	1
2	1	④	④ 2
3	3	4	3
4	4	4	4
5	6	x because greater than 4	5
6	7	⑦	⑦ 6
7	7	7	7

matched histogram has 3 bar

matched Histogram



we see that if ignore the 0 value as the error,

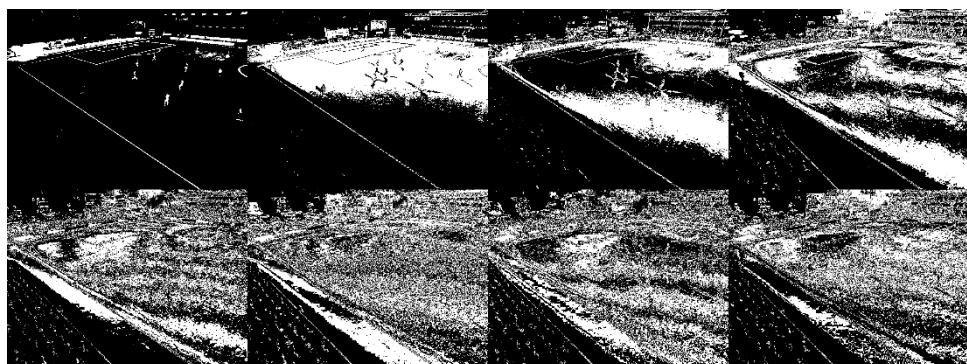
the obtained histogram very matched to given histogram, based on their shape.

مسئله دوم

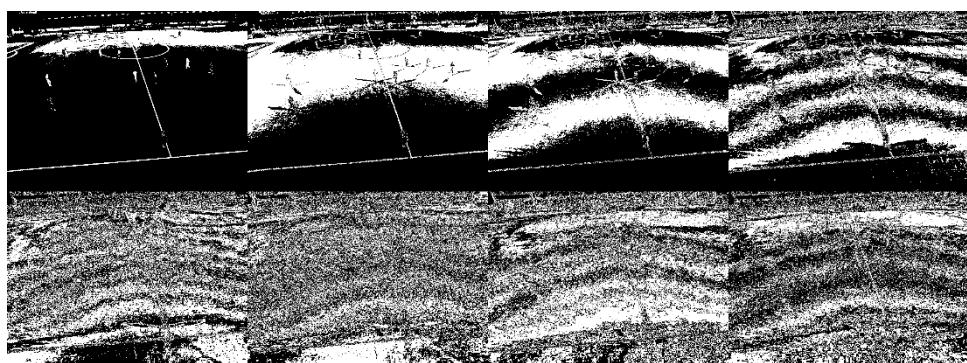
مسئله ۲ قسمت (a)

تابع مد نظر `bitplane_slice` در فرمت خواسته شده پیاده سازی شده است. این تابع یک عکس را ورودی میگیرد و سپس آن را به فرمت `gray-scale` تبدیل کرده و یک ماتریس خالی به شکل $(h, w, 8)$ به عنوان خروجی این تابع ایجاد می کند(h و w ابعاد تصویر); سپس بوسیله تابع کمکی پیاده سازی شده `get_bin`) مقدار هر یک از پیکسل ها را به معادل باینری آن تبدیل کرده و هر کدام از اعداد $(0, 1)$ حاصل را در یکی از هشت درآیهای متناظر با آن ذخیره می کند. همچنین برای نمایش اسلایس های هشتگانه در پاسخ به این سوال، یک تابع کمکی به نام `save_slices` پیاده سازی شده است که هشت بیت اسلاید را در یک عکس کنار هم ادغام کرده و قابل نمایش می کند؛ در تصاویر خروجی این تابع ارزش بیت ها بصورت چپ به راست و بالا به پایین کاهش پیدا می کند. در زیر خروجی به ازای فریم های اول از سه سکانس متفاوت آورده شده است (همچنین فریم دوم سکانس ها نیز در فolder خروجی ضمیمه شده است)؛

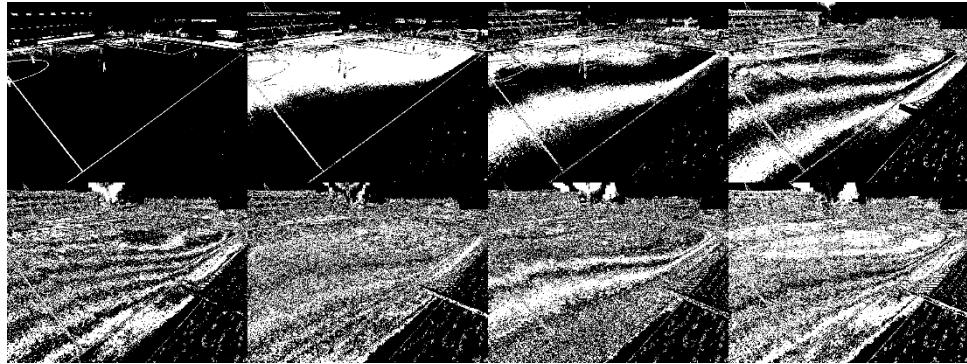
خروجی `bitplane slicing` برای سکانس اول و فریم اول:



خروجی `bitplane slicing` برای سکانس دوم و فریم اول:



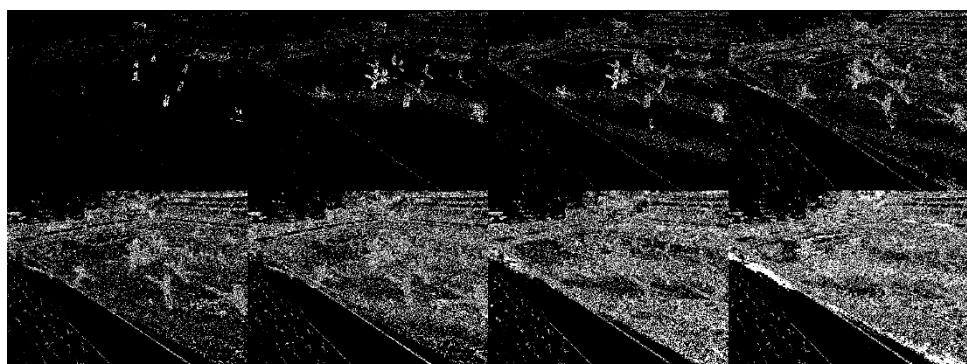
خروجی bitplane slicing برای سکانس سوم و فریم اول:



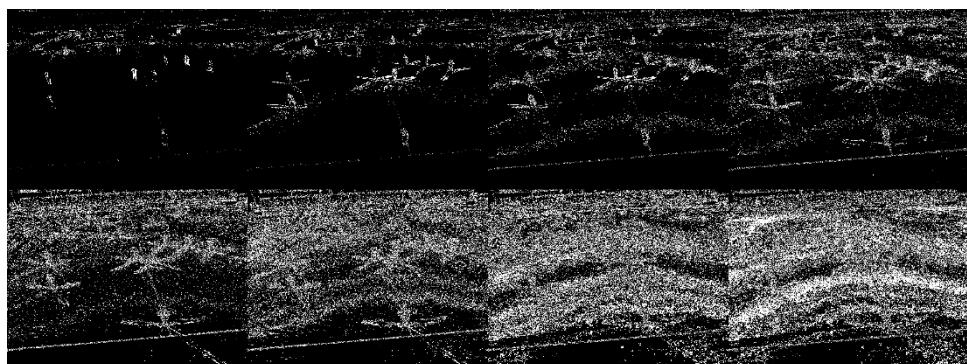
مسئله ۲ قسمت (b)

برای پاسخ‌گویی به این بخش یک تابع به نام xor_slices پیاده سازی شده است که در آن با استفاده از عملگر XOR کتابخانه numpy اسلایس های دو فریم مربوط به یک سکانس با هم xor می شود و خروجی برگردانده می شود.

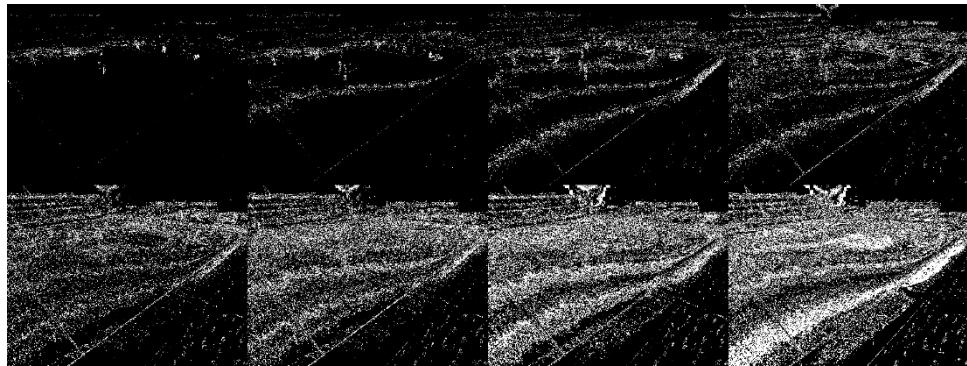
خروجی XOR فریم اول و فریم دوم مربوط به سکانس اول:



خروجی XOR فریم اول و فریم دوم مربوط به سکانس دوم:



خروجی XOR فریم اول و فریم دوم مربوط به سکانس سوم:



مسئله ۲ قسمت c و d

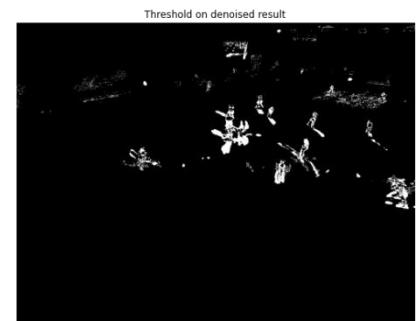
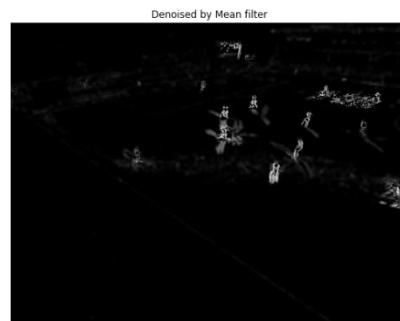
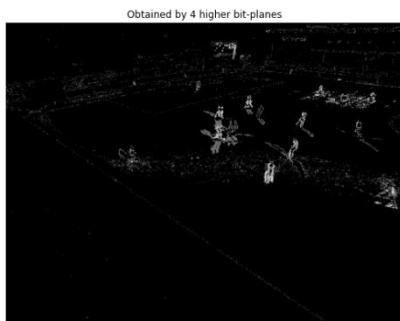
برای پاسخ گویی به این دو بخش از این سوال، دو تابع با نام های enhancement و get_moving_objs پیاده سازی شده است. در تابع enhancement رابطه ای ذکر شده در متن سوال باهم ترکیب شده و عکس خروجی gray-bit plane با ارزش (۷-۴) طبق رابطه ای ذکر شده در متن سوال باشند. در تابع get_moving_objs که را حاصل می کند که در آن نواحی متحرک در تصویر قابل نمایش است (تصویر سمت راست در خروجی های صفحه بعد).

اما همانطور که در متن سوال گفته شده است، این خروجی دارای نویز بوده و چندان مطلوب مورد نظر نمی باشد؛ از این تابع enhancement پیاده سازی شده است. در این تابع دو تکنیک برای حذف نویز و بهبود نواحی متحرک استفاده شده است. اولین مورد حذف نویز با اعمال فیلتر میانگین گیری می باشد که در کلاس درس آموخته ایم که این فیلتر ضمن ایجاد تاری، باعث حذف نویز می باشد. از جایی که استخراج کلیت ناحیه ای متحرک مطلوب مورد نظر ما می باشد و تار شدن اهمیت چندانی ندارد، این فیلتر را می توان استفاده کرد.

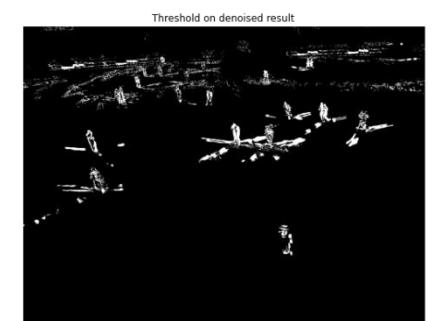
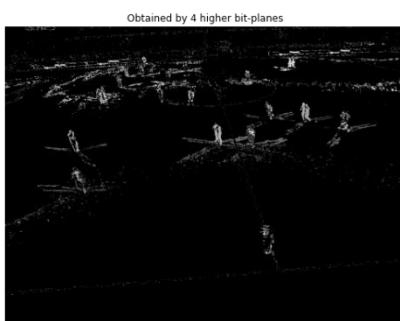
در خروجی های صفحه بعد تصویر وسطی مربوط به اعمال فیلتر میانگین گیری می باشد؛ همانطور که ملاحظه می شود نتیجه ای آن حذف برخی از نویز ها و همچنین کم رنگ شدن برخی دیگر از نویز ها نظیر خطوط حاشیه ای زمین فوتبال در سکانس اول می باشد. در مرحله قبل مشاهده شد که مقدار زیادی از نویزها در خروجی حتی اگر حذف نشده اند، دچار تاری و کمرنگی شده اند؛ از این رو به عنوان گام دوم بهبود نتیجه، با قرار دادن یک threshold روی خروجی مرحله قبل و حذف پیکسل هایی که سطوح روشنایی بسیار اندکی دارند به تصویری با تفکیک پذیری بهتر نواحی متحرک دست پیدا می کنیم.

در خروجی ها صفحه بعد تصویر سمت راست مربوط به اعمال thresholding می باشد. همچنین پارامتر های مورد نیاز برای فیلتر و threshold بصورت سعی و خطا و آزمایش حالات مختلف تعیین شده است.

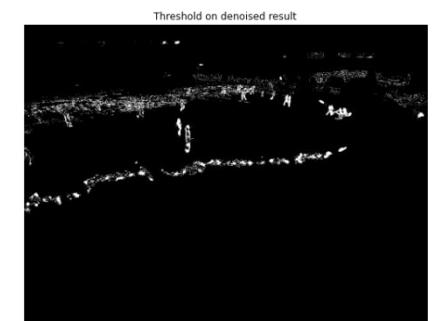
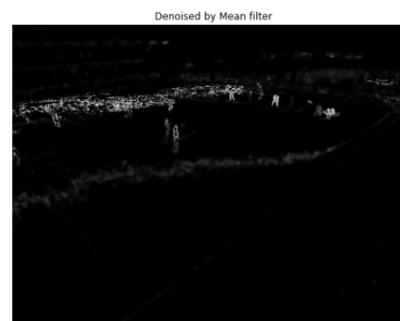
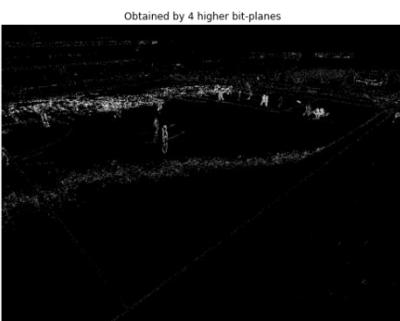
نواحی متحرک در سکانس اول:



نواحی متحرک در سکانس دوم:



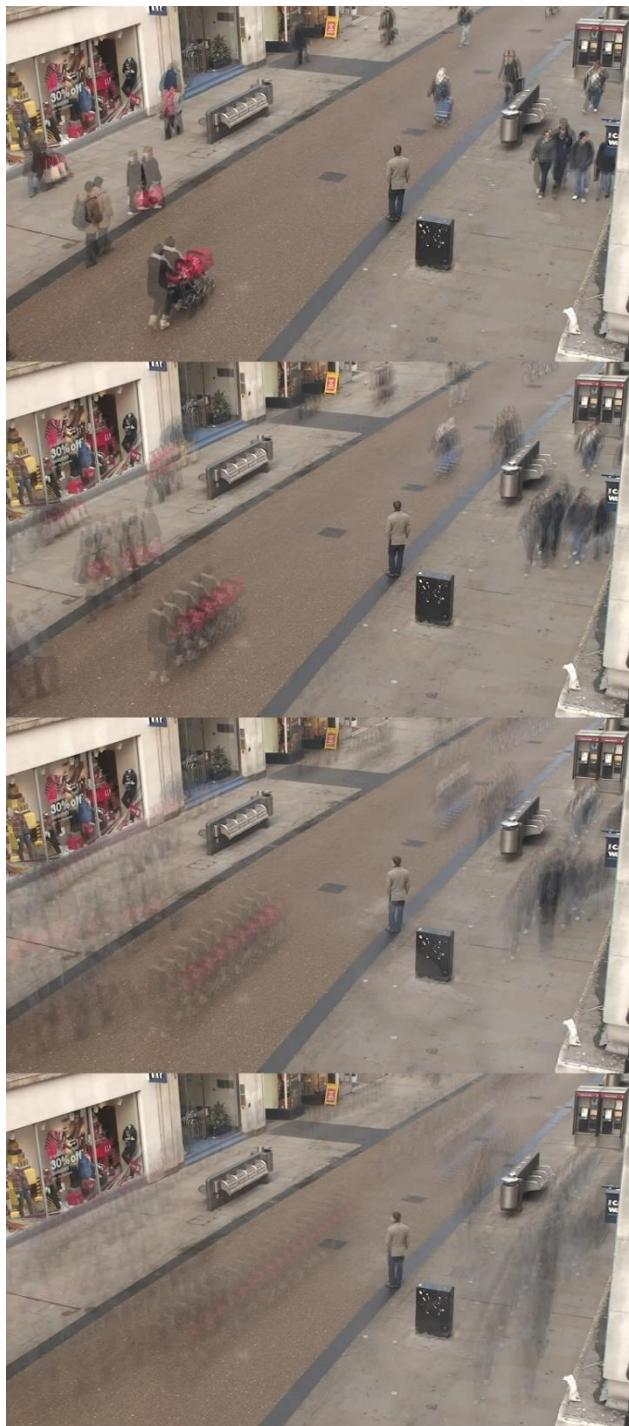
نواحی متحرک در سکانس سوم:



مسئله سوم

مسئله ۳ قسمت (a)

برای پاسخ‌گویی به حل این قسمت از سوال، یک تابع `estimate_background` از پایه و بصورت دستی پیاده سازی شده است که دو پارامتر n به عنوان تعداد فریم انتخابی و $med(ian)$ به عنوان تعیین نوع عملگر را ورودی گرفته و تصویر پس زمینه تخمین زده شده را بر می‌گرداند. اگر `med` معادل با `True` باشد، از میانه و در غیراینصورت از میانگین برای تخمین استفاده می‌کند. تصاویر مقابل حاصل پیش‌بینی پس زمینه به ازای n های ۲، ۵، ۱۰ و ۲۰ با استفاده از تخمین میانگین می‌باشد (از بالا به پایین):

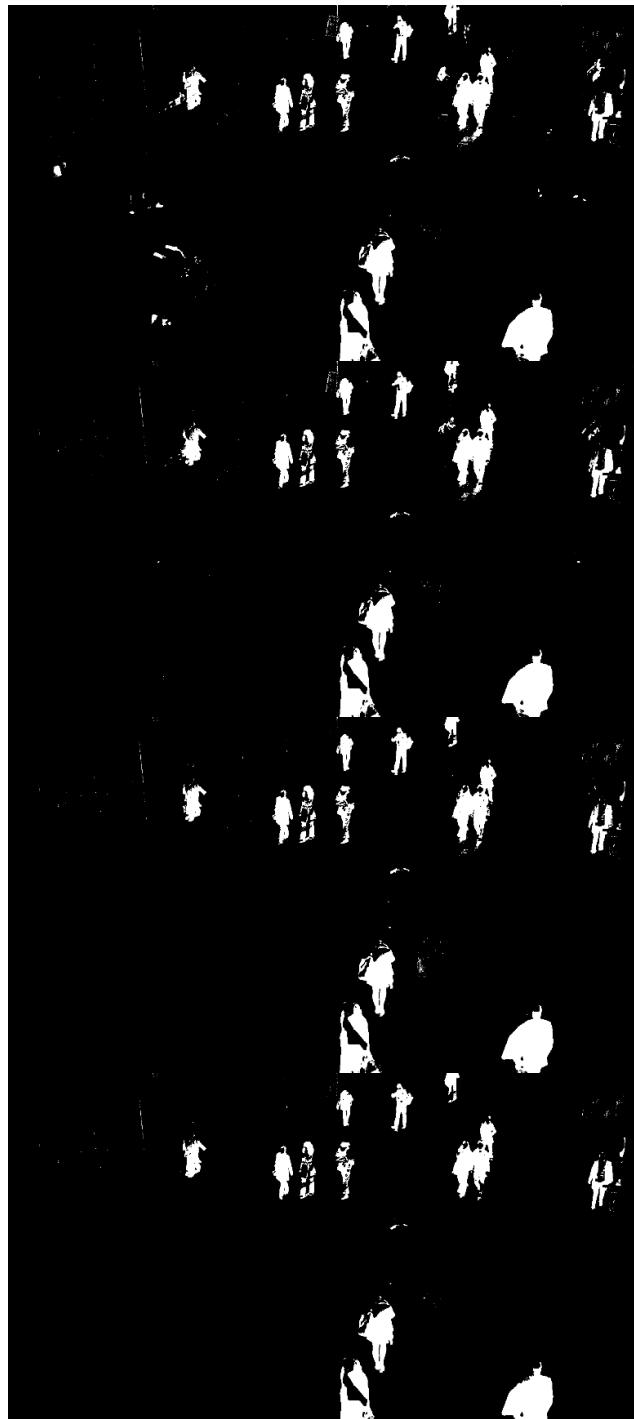


مسئله ۳ قسمت b

حال به ازای هر یک تخمین های بدست آمده، هر کدام از تصویر های آزمون (Test) را از آن کسر می کنیم تا تفاوت آن ها نواحی متغیر در پس زمینه را به ما خروجی بدهد که مطلوب همان عابرین پیاده هستند. در گزارش های زیر، خروجی های سمت راست مربوط به تصویر اول آزمون و خروجی های سمت چپ مربوط به تصویر دوم آزمون می باشد:



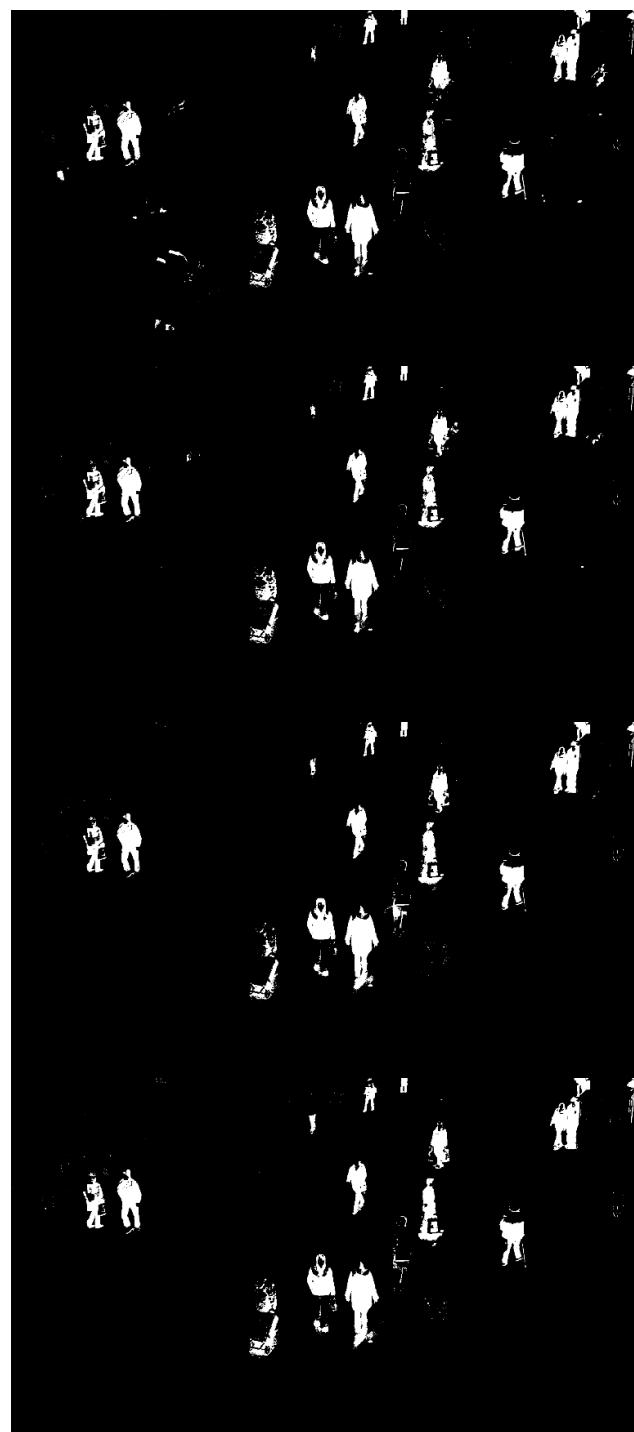
برای بدست آوردن ماسک عابرین پیاده موجود در تصاویر آزمون، بایستی یک مرز (آستانه) سطح روشنایی برای اختصاص دادن پیکسل ها به عنوان عابرپیاده یا پس زمینه تعیین کنیم. هر چقدر این مرز مناسب تعیین شود، همانقدر کیفیت و دقت خروجی ماسک نیز بالا خواهد بود. در گام اول برای بدست آوردن این مرز بصورت سعی و خطأ تلاش کردم که در آن بازه‌ی ۲۰ تا ۳۵ بازه‌ی معقولی برای انتخاب مرز بود.



در این بازه هر چقدر به سمت اعداد کوچکتر پیش رفتیم، نویز های موجود در سایر نواحی به غیر از عابرین پیاده بیشتر شد؛ همچنین هر چقدر به سمت اعداد بزرگتر پیش رفتیم، نویز های کمتری را شاهد بودیم اما در مقابل قسمت هایی از خود عابرین پیاده را نیز از دست می‌دادیم لذا باید در یک trade-off تصمیم بگیریم.

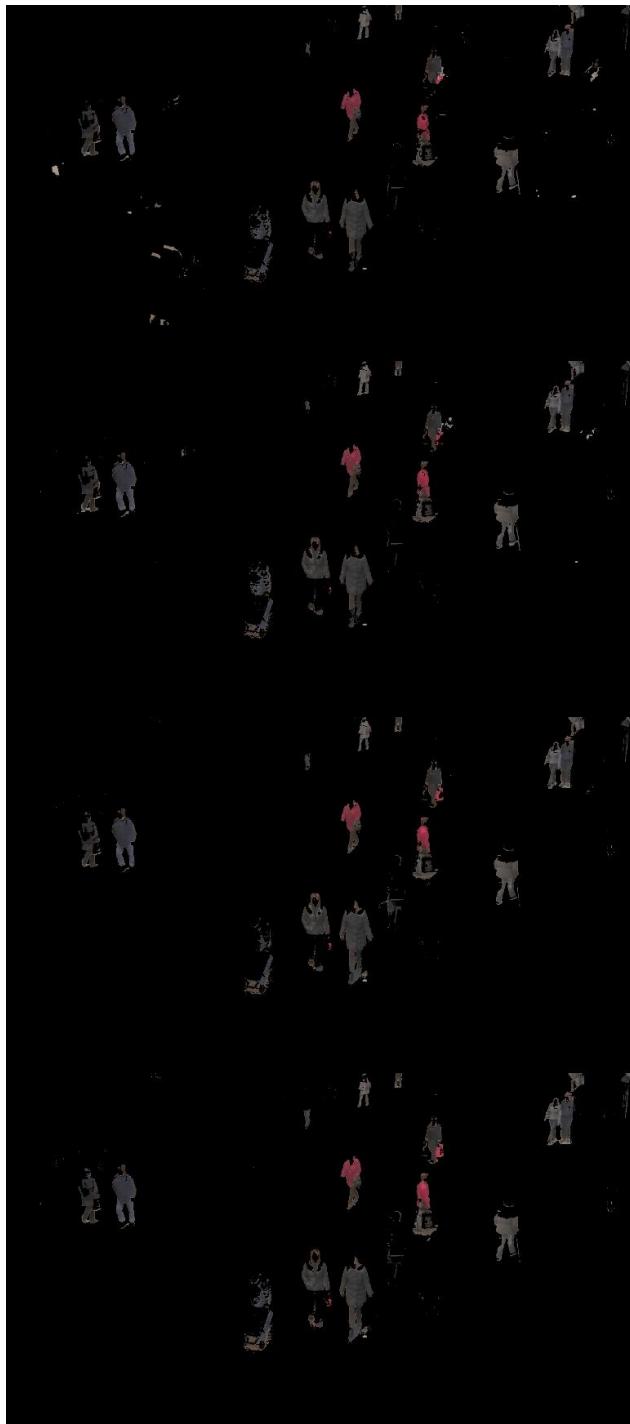
بنده برای حل این چالش از اعداد خروجی هیستوگرام سطح روشنایی به ازای حاصل تفریق هر تصویر پس زمینه‌ی تخمینی با تصویر آزمون استفاده کرده ام. این کار علاوه بر حل تقریبی چالش فوق و انتخاب یک مرز مناسب، باعث می‌شود مرز انتخابی به ازای هر تصویر پس زمینه تخمینی مختص همان باشد و یک مرز سخت گیرانه برای همه‌ی تصاویر تخمینی اعمال نشود. سطح سومین ستون در این نمودار را به عنوان مرز آستانه در نظر گرفتم که در تصاویر مختلف معمولاً بین ۳۰ الی ۳۴ حاصل شده است. تصویر مقابل ماسک های بدست آمده به ازای تصویر آزمون اول به ازای تصاویر پس زمینه تخمینی را نشان می‌دهد.

تصویر زیر نیز ماسک های بدست آمده به ازای تصویر آزمون دوم را به ازای تصاویر پس زمینه تخمینی نشان می دهد.



مسئله ۳ قسمت d

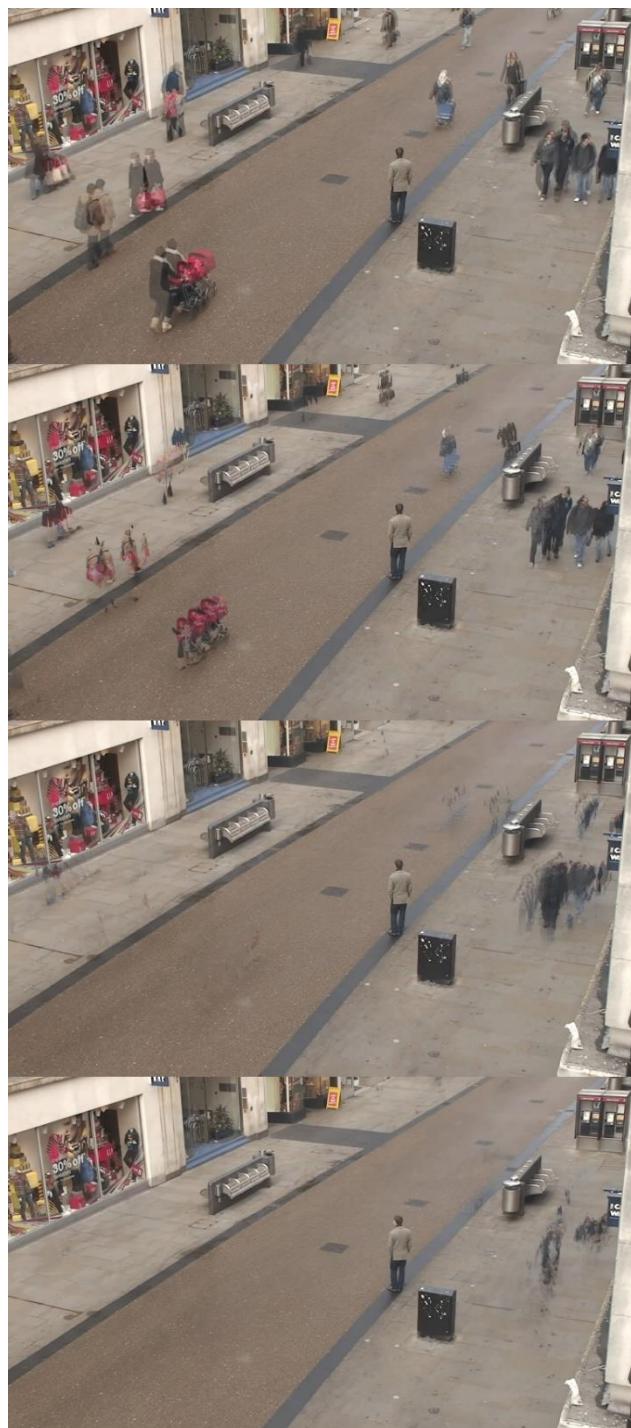
برای پاسخ گویی به این پرسش، صرفا کافی است ماسک بدست آمده در مرحله قبل را با خود تصاویر آزمون AND بیتی کیم تا بدین صورت نواحی مشکی رنگ همان مشکی فیلتر شوند(AND بیتی اعداد صفر با هر عددی صفر است) و نواحی سفید رنگ نیز که ماسک عابرین پیاده‌ی ما هستند، از تصویر آزمون انتخاب و قابل نمایش شوند(AND بیتی اعداد یک با هر عددی همان خود عدد است). در پیاده سازی این بخش از این مربوط bitwise_and مربوط به کتابخانه open-cv استفاده شده است. در گزارش زیر، خروجی های سمت راست مربوط به تصویر آزمون اول و خروجی های سمت چپ مربوط به تصویر آزمون دوم می باشد.



مسئله ۳ قسمت e)

برای پاسخ گویی به این پرسش، صرفا گام های بالا و با همان توضیحات فوق را به ازای عوض کردن پارامتر med در تابع estimate_background تکرار می کنیم. همانطور که مشهود است تخمین تصویر پس زمینه با استفاده از میانه (این حالت) خصوصا در برخی نواحی مانند اواسط خیابان که حالات خالی در چندین فریم موجود بوده است بسیار بهتر و با دقت تر از تخمین توسط میانگین بوده است. در گزارش های زیر، خروجی های سمت راست مربوط به تصویر آزمون اول و خروجی های سمت چپ مربوط به تصویر آزمون دوم می باشد و ترتیب گزارش ها نیز از a-d و از بالا به پایین می باشد.

خروجی به ازای خواسته‌ی قسمت a:



خروجی به ازای خواسته‌ی قسمت b:



خروجی به ازای خواسته‌ی قسمت c:



خروجی به ازای خواسته‌ی قسمت d:



شمارش عابرین پیاده در تصویر یکی از مسائل مهم و چالشی حوزه‌ی بینائی می‌باشد که تاکنون توسط راه حل‌های گوناگونی حل شده است که تقریباً ابتدایی ترین آن صرفاً مقایسه‌ی چند فریم متوالی از صحنه‌ی مورد نظر می‌باشد که در این سوال نیز روند انجام شده مشابه با آن می‌باشد لذا در در همین ابتدایی پاسخ گویی میتوان انتظار داشت که دقت شمارش عابرین پیاده با ماسک استخراجی به هیچ عنوان قابل قبول حاصل نخواهد شد. (محدود بودن فریم‌ها و تخمین ضعیف پس زمینه و...) دو چالش برای شمارش عابرین پیاده در وضعیت موجود وجود دارد: ۱) وجود نویز در ماسک‌های استخراجی و ۲) تقطیع اندام برخی از عابرین پیاده بخارط وجود پوشش‌های بسیار متفاوت و گاه شبیه به پس زمینه.

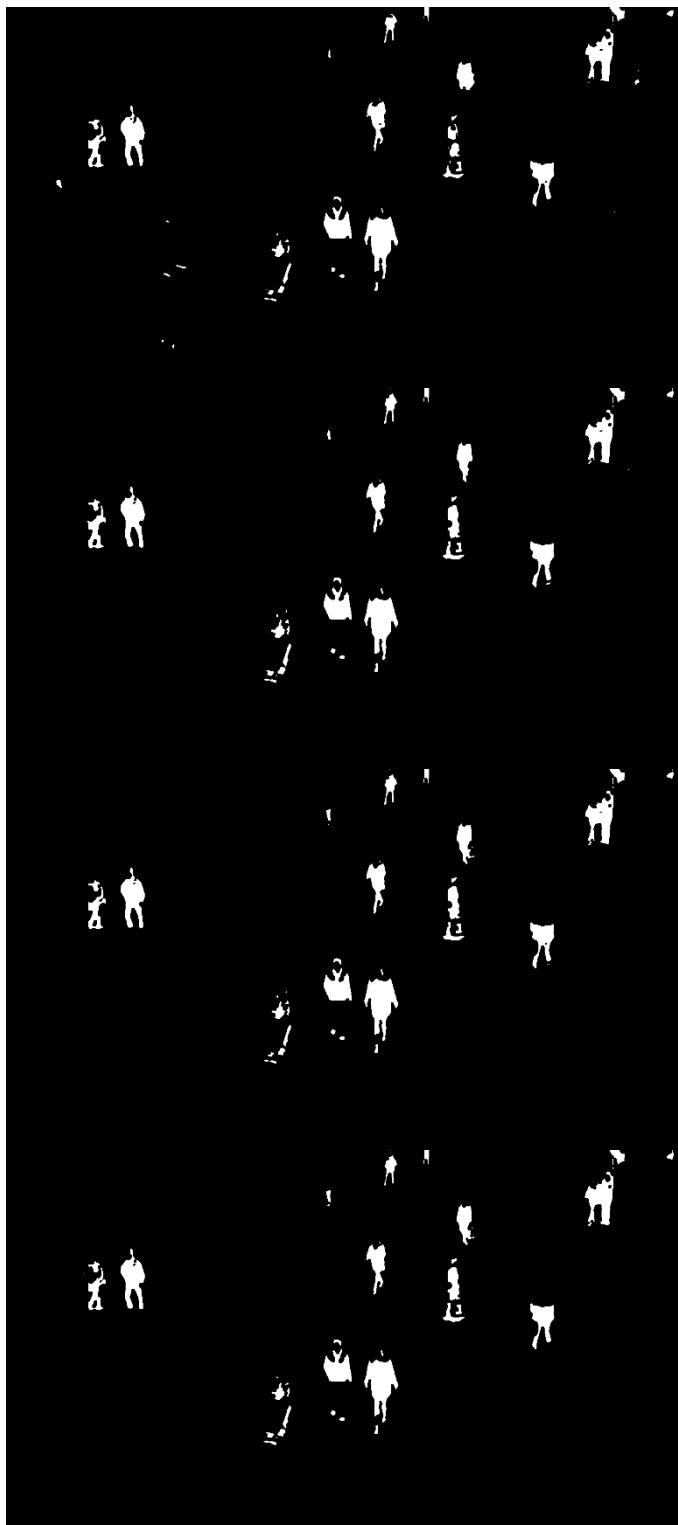
برای حل این مسئله ابتدا توسط سه فیلتر مختلف Averaging Gaussian و Median ماسک‌های عابرین (چهار ماسک به ازای چهار تصویر تخمینی از پس زمینه) را هموار می‌کنیم و سپس با یک threshold نواحی تار شده که نویز‌ها بوده اند را حذف می‌کنیم تا بدین گونه بتوانیم چالش اول را مدیریت کنیم. سپس در گام دوم و با استفاده ازتابع connectedComponents از کتابخانه open-cv شروع به شمارش قطعه ماسک‌های موجود در تصویر می‌کنیم (در متن سوال خواسته نشده است که تابع حتماً دستی پیاده سازی شود و صرفاً ایده پیاده سازی دستی به عنوان یک نظر ارائه شده است لذا از این تابع آماده استفاده کردم. در صورت لزوم پیاده سازی دستی میتوان از الگوییم‌های تشخیص اتصال در گراف استفاده کرد که در اینجا هدف نمی‌باشد). این تابع اجزای همبند در تصویر را شمارش می‌کند که همان مطلوب ماست؛ همچنین برای تعیین همسایگی اتصال نوع ۴ در نظر گرفته شده است (در اسلاید‌های درس انواع اتصال‌ها را آموخته ایم). از قطعه ماسک‌های شمارش شده‌ای که کمتر از ۳۰۰ پیکسل هستند صرف نظر می‌کنیم چرا که این قطعات یا نویز هستند یا صرفاً بخشی از یک عابر پیاده زیرا که ۳۰۰ پیکسل نمیتواند نماینده یک عابر پیاده باشد. بدین‌گونه چالش دوم نیز تا حدی مدیریت می‌شود و تعداد افراد حاضر در صحنه حاصل می‌شود. (این مرز ۳۰۰ پیکسلی را با سعی و خطا بدست آورده‌ام) همانطور که در توضیحات ابتدایی مطرح شد نمیتوان انتظار بسیار دقیقی داشت و تخمین انجام شده برای شمارش افراد حاضر در صحنه ساده لوحانه است زیرا ممکن اندام یا اجزای گستته‌ی یک عابر پیاده دو بار شمارش شود (جای بحث در خصوص آستانه ۳۰۰ پیکسلی وجود دارد و نمیتوان به سعی و خطا اکتفا کرد) و یا یک عابر پیاده که پوشش بسیار چالشی دارد هیچ گاه شمارش نشود (هر کدام از اندام‌های تقطیع شده اش زیر ۳۰۰ پیکسل باشد)؛ حتی ممکن است افرادی که کنار هم هستند و ماسک‌هایشان باهم همپوشانی دارند به عنوان یک عابر شمارش شود. (میتوان برای این مورد نیز آستانه تعیین کرد و در صورت بیشتر بودن از یک سطح آن قطعه را دو فرد شمارش کرد ولی نیازمند سعی و خطای بسیار است زیرا دوری و نزدیک افراد به دوربین می‌تواند این مرز را دچار چالش کند) با وجود همه‌ی توضیحات فوق، گزارش‌های حاصل از شمارش عابرین پیاده به ازای هر یک از فیلتر‌های هموار ساز مذکور در صفحه بعد آورده شده است. هر سطر از چهار سطر گزارش شده به ازای هر تصویر آزمون مربوط به یک تصویر پس زمینه‌ی تخمینی می‌باشد که از بالا به پایین به ترتیب به ازای ۲، ۵، ۱۰ و ۲۰ فریم می‌باشد:

```
Proc test #1:  
-> Count of pedestrians extracted(Averaging filter): 18  
-> Count of pedestrians extracted(Averaging filter): 16  
-> Count of pedestrians extracted(Averaging filter): 16  
-> Count of pedestrians extracted(Averaging filter): 15  
Proc test #2:  
-> Count of pedestrians extracted(Averaging filter): 18  
-> Count of pedestrians extracted(Averaging filter): 18  
-> Count of pedestrians extracted(Averaging filter): 18  
-> Count of pedestrians extracted(Averaging filter): 20
```

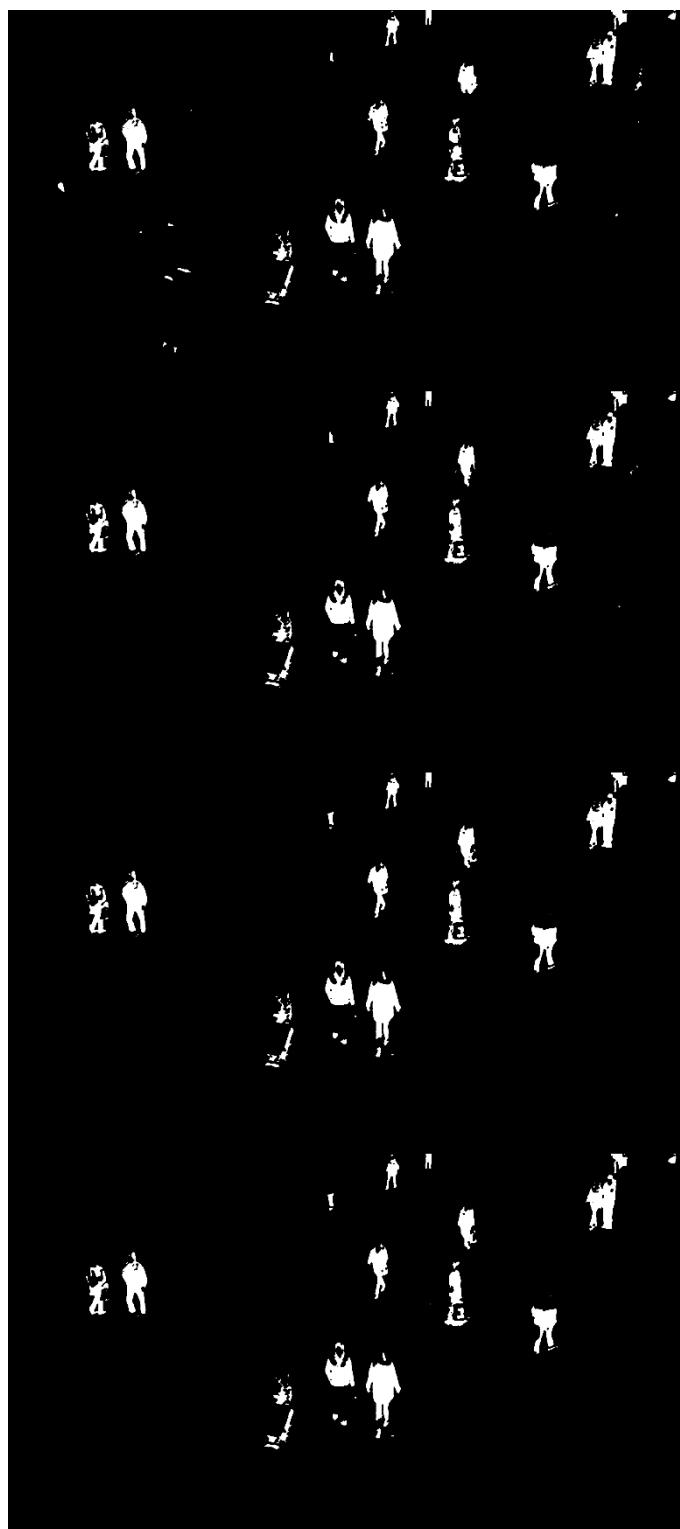
```
Proc test #1:  
-> Count of pedestrians extracted(Gaussian filter): 20  
-> Count of pedestrians extracted(Gaussian filter): 16  
-> Count of pedestrians extracted(Gaussian filter): 16  
-> Count of pedestrians extracted(Gaussian filter): 16  
Proc test #2:  
-> Count of pedestrians extracted(Gaussian filter): 21  
-> Count of pedestrians extracted(Gaussian filter): 20  
-> Count of pedestrians extracted(Gaussian filter): 19  
-> Count of pedestrians extracted(Gaussian filter): 21
```

```
Proc test #1:  
-> Count of pedestrians extracted(Median filter): 22  
-> Count of pedestrians extracted(Median filter): 16  
-> Count of pedestrians extracted(Median filter): 18  
-> Count of pedestrians extracted(Median filter): 17  
Proc test #2:  
-> Count of pedestrians extracted(Median filter): 24  
-> Count of pedestrians extracted(Median filter): 21  
-> Count of pedestrians extracted(Median filter): 21  
-> Count of pedestrians extracted(Median filter): 23
```

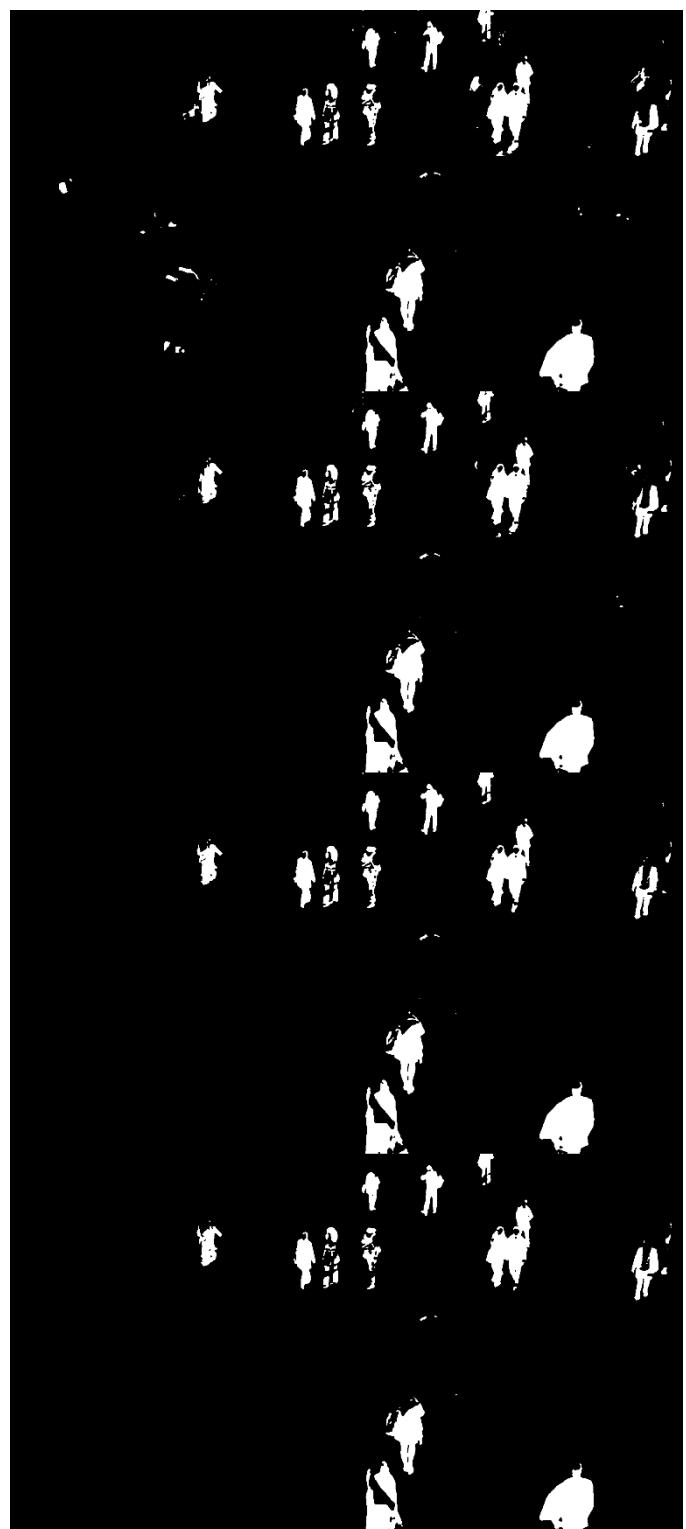
خروجی اعمال فیلتر Averaging به ازای ماسک تصویر آزمون اول(سمت راست) و تصویر آزمون دوم(سمت چپ):



خروجی اعمال فیلتر Gaussian به ازای ماسک تصویر آزمون اول(سمت راست) و تصویر آزمون دوم(سمت چپ):



خروجی اعمال فیلتر Median به ازای ماسک تصویر آزمون اول(سمت راست) و تصویر آزمون دوم(سمت چپ):



مسئله چهارم

مسئله ۴ قسمت (a)

توابع مورد نیاز برای پیاده سازی فیلتر دو سویه‌ی مدل نظر بصورت دستی و از پایه پیاده سازی شده و کد آن در سورس پیوستی قابل مشاهده است. اولین تابع پیاده سازی شده `bilateral_weight` می باشد که همان تابع `w` زیر می باشد:

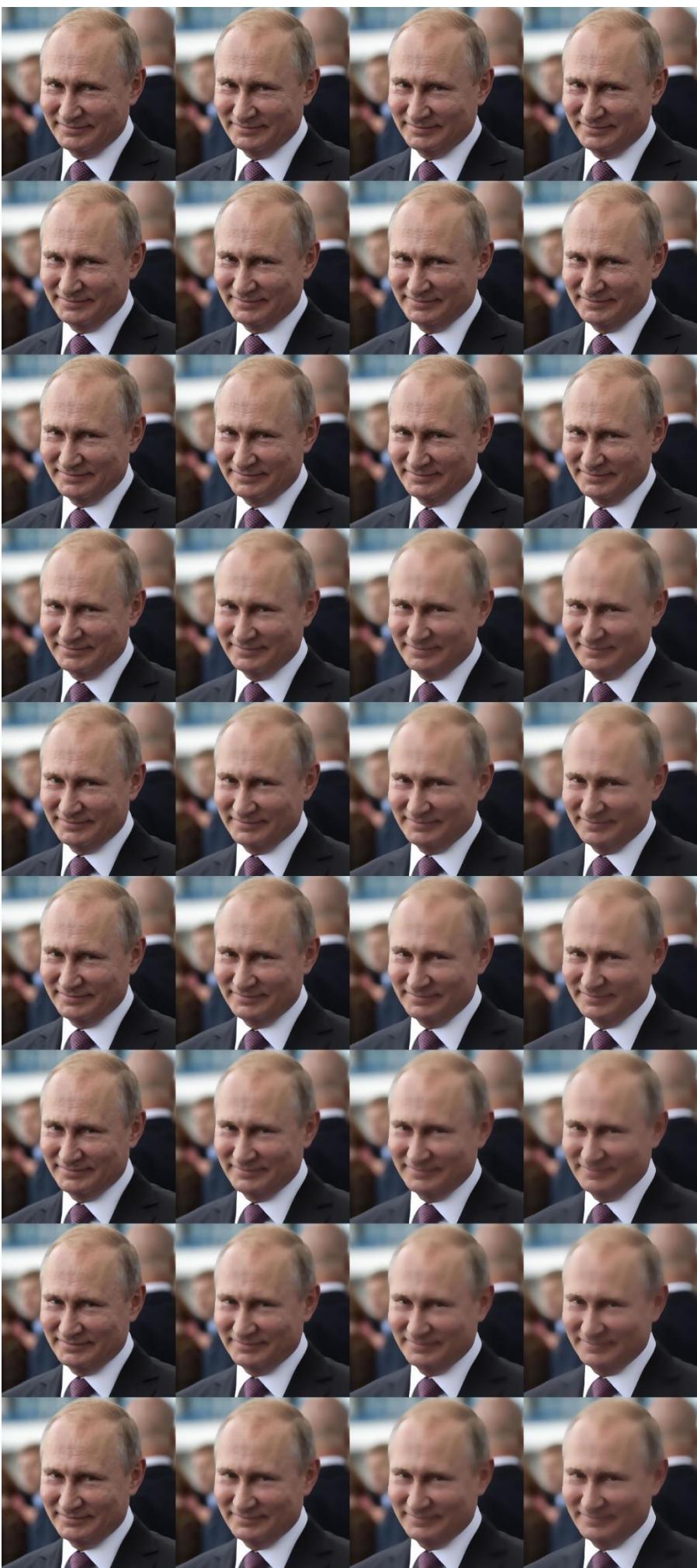
$$\omega(i, j, k, l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2} - \frac{\|I(i, j) - I(k, l)\|^2}{2\sigma_r^2}\right)$$

این تابع علاوه بر موارد یاد شده در فرمول بالا، خود تصویر(I) و پارامترهای هموارساز را نیز به عنوان پارامترهای ورودی دریافت می کند. تابع دوم پیاده سازی شده `bilateral_filter` می باشد که پارامترهای تصویر(I)، شعاع انتخاب همسایگی(r) و پارامترهای هموارساز را به عنوان ورودی دریافت می کند. این تابع ابتدا یک تصویر خالی هم سایز با تصویر ورودی را به عنوان تصویر فیلتر شده برای خروجی تابع ایجاد می کند؛ سپس شروع به پیمایش پیکسل های تصویر ورودی و همسایه‌های آن کرده و رابطه‌ی زیر را محاسبه و مقدار بدست آمده را در تصویر فیلتر شده و در موقعیت متناظر با پیکسل پردازشی قرار قرار می دهد:

$$I_F(i, j) = \frac{\sum_{k,l} I(k, l) \omega(i, j, k, l)}{\sum_{k,l} \omega(i, j, k, l)}$$

مسئله ۴ قسمت (b)

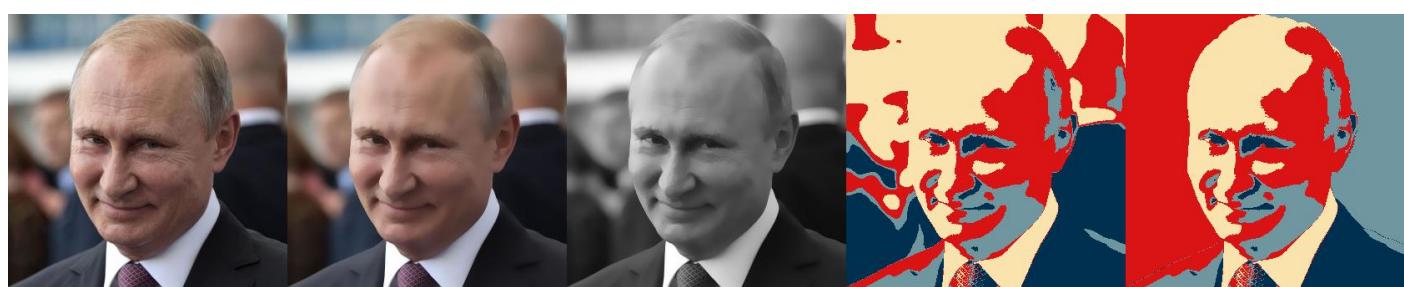
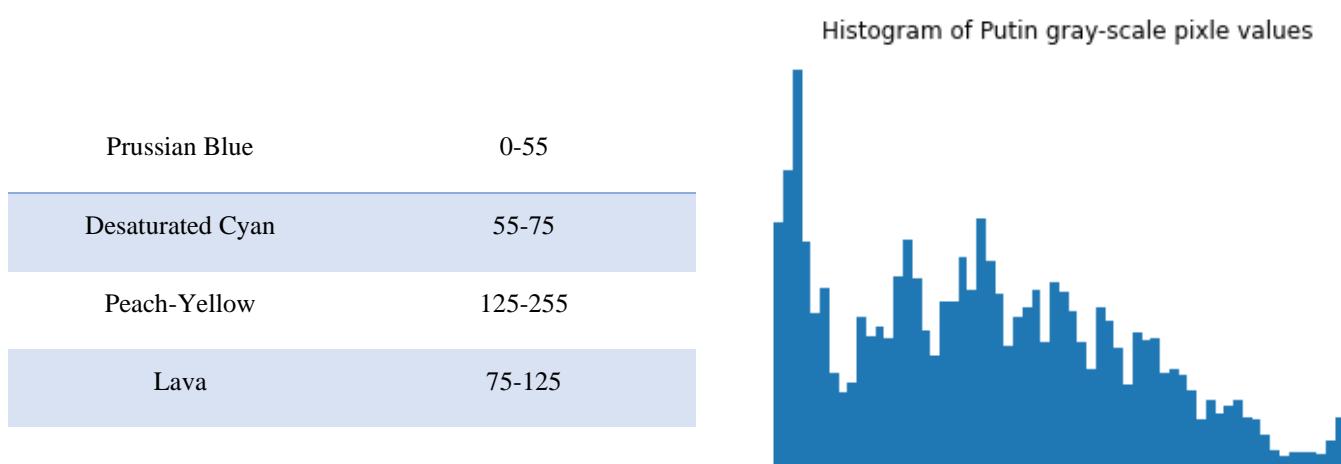
با توجه به اینکه ابعاد تصاویر ورودی بسیار بزرگ بوده و ضمناً زمان مورد نیاز برای اعمال فیلتر برای آن زیاد می باشد(تابع پیاده سازی شده با فرمول گفته شده بهینگی کامل ندارد چرا که پیمایش پیکسل به پیکسل می باشد؛ لذا تصاویر ورودی را به 50 درصد ابعاد خود تغییر سایز می دهیم تا زمان مورد نیاز برای پردازش و خروجی گرفتن در لپ تاپ شخصی کاهش یابد. به جای اعمال چهار بار ارزیابی برای بررسی پارامترهای موردنیاز در فیلتر دوس سویه، 36 بار فیلتر مذکور روی تصویر `putin` اعمال شده است تا بتوان بهترین پارامترها را استخراج نمود. پارامترهای اعمالی بدین گونه است که بتوان تغییرات پارامترهای هموارساز(sigma_r و sigmad) را توامان با تغییرات شعاع همسایگی پیکسلها آزمایش نمود. تغییرات پارامتر همسایگی به ازای مقادیر 2 و 4 و 6 و 8 بررسی شده است؛ همچنین تغییرات هر یک از پارامترهای هموارساز به ازای مقادیر 10 و 45 و 80 بررسی شده است که در مجموع 36 آزمایش را تشکیل داده است. صفحه بعد خروجی این آزمایشها را نشان می دهد. در هر سطر از چپ به راست شعاع همسایگی بیشتر شده است و در هر سطر نسب به دیگری یکی از پارامترهای هموارساز تغییر کرده است(sigma_r ابتدا سه بار تغییر میکند و سپس sigmad یکبار تغییر می کند و این منوال ادامه پیدا می کند)



مسئله ۴ قسمت c و d

از بین آزمایش های انجام شده برای انتخاب پارامتر، در گزارش صفحه قبل و در سطر پنجم و سومین تصویر از چپ برای اعمال رنگ آمیزی انتخاب شد؛ دلیل این انتخاب آن است که جزئیات چهره نظیر چشم در اثر تار شدن دچار تغییر نشده و در عین حال تاری باعث حذف قسمتی از چروک چهره شده است و این دو در حد نسبی خوبی قرار دارد.(اگر تاری و در نتیجه حذف چروک زیاد باشد، باعث خواهد شد در زمان رنگ آمیزی تصویر حاصل از چهره اصلی فاصله داشته و همچنین تا حدودی پیدا نمودن آستانه رنگ آمیزی مناسب سخت باشد.) پارامتر های این تصویر برای شاعع همسایگی برابر با ۶ و برای پارامتر های هموارساز نیز به ترتیب برابر با ۴۵ و ۱۰ می باشد.

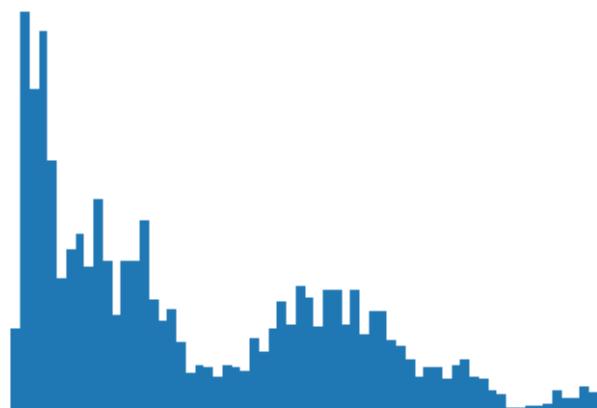
برای انتخاب آستانه مناسب برای رنگ آمیزی از نمودار هیستوگرام هر کدام از تصاویر استفاده می کنیم و برای بازه هایی که جهت تغییرات فراوانی(با نگاه چشمی) در حال عوض شدن می باشد، رنگ نیز عوض می شود تا کل بازه‌ی ۲۵۶ سطح روشنایی به ۴ بازه افزار شود. برای تصویر پوتین بازه های انتخابی بر اساس هیستوگرام جدول زیر می باشد؛ همچنین در ادامه تصویر نهایی رنگ آمیزی شده(با/بدون اعمال ماسک) قابل مشاهده است:



برای تصویر زلنسکی نیز بازه های انتخابی بر اساس هیستوگرام جدول زیر می باشد؛ همچنین در ادامه تصویر نهایی رنگ آمیزی شده(با/بدون اعمال ماسک) قابل مشاهده است:

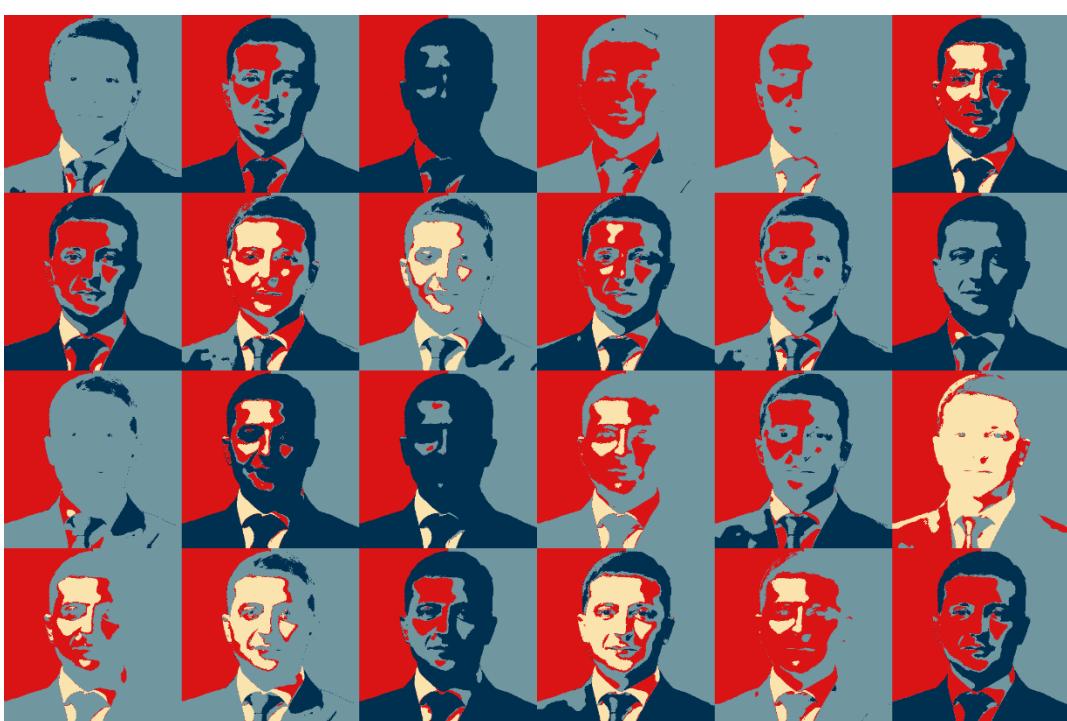
Histogram of Zelensky gray-scale pixle values

Prussian Blue	0-70
Desaturated Cyan	70-105
Peach-Yellow	155-255
Lava	105-155



مسئله ۴ قسمت (e)

قطعاً تعیین آستانه‌ی تغییر رنگ آمیزی در سطح خاکستری تصویر بسیار حائز اهمیت است چرا که این مرز‌ها نواحی از تصویر هستند که معمولاً لبی معنی دار یا تغییرات معنایی در آن وجود دارد و در صورتی که برای یافتن این مرز‌ها تلاش و استدلالی نشود، معنای تصویر ممکن است در اثر رنگ آمیزی نامناسب از بین رود. برای نشان دادن این نکته، تصویر زلنسکی با اعمال همان فیلتر دو سویه‌ی قسمت قبل را در نظر گرفته و ۲۴ بار آن را بصورت تصادفی رنگ آمیزی می‌کنیم که نتیجه به صورت زیر حاصل می‌شود؛ همانطور که مشهود است در چندین مورد جزئیات چهره در اثر تعیین آستانه رنگ آمیزی نامطلوب از بین رفته است و عملاً امکان بازشناصی به هیچ طریقی میسر نیست:



مسئله ۵ قسمت (a)

توابع خواسته شده به صورت کامل و از پایه و طور بهینه پیاده سازی شد؛ علاوه بر موارد خواسته شده، دو تابع `get_cost_matrix` و `Handler` برای سهولت و بهبود روند خواسته شده کد نویسی شده است. تابع `get_cost_matrix` تابعی هست که کارکرد آن بصورت بازگشته بوده و ماتریس M زیر را بصورت برنامه نویسی پویا محاسبه و بر میگردداند:

$$M(i, j) = e(i, j) + \min(M(i-1), M(j-1), M(i-1, j), M(i-1, j+1))$$

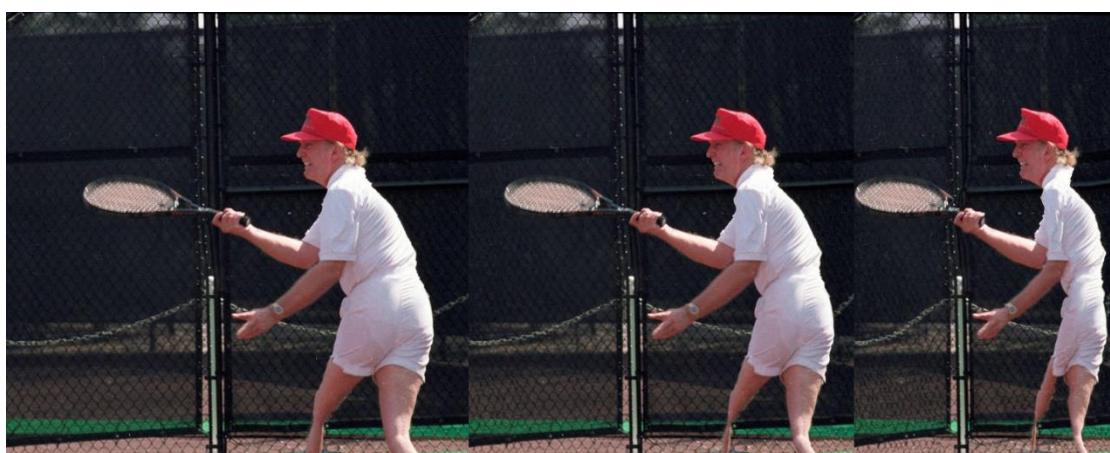
تابع `Handler` نیز تصویر ورودی، درصد کاهش و جهت کاهش را دریافت کرده و با توابع پیاده سازی شده زیر آن را کاهش سایز می دهد و سپس خروجی ها را مورد نظر را گزارش گرفته و نمایش می دهد. توضیحات زیر خلاصه جزئیات پیاده سازی توابع خواسته شده در متن سوال می باشد:

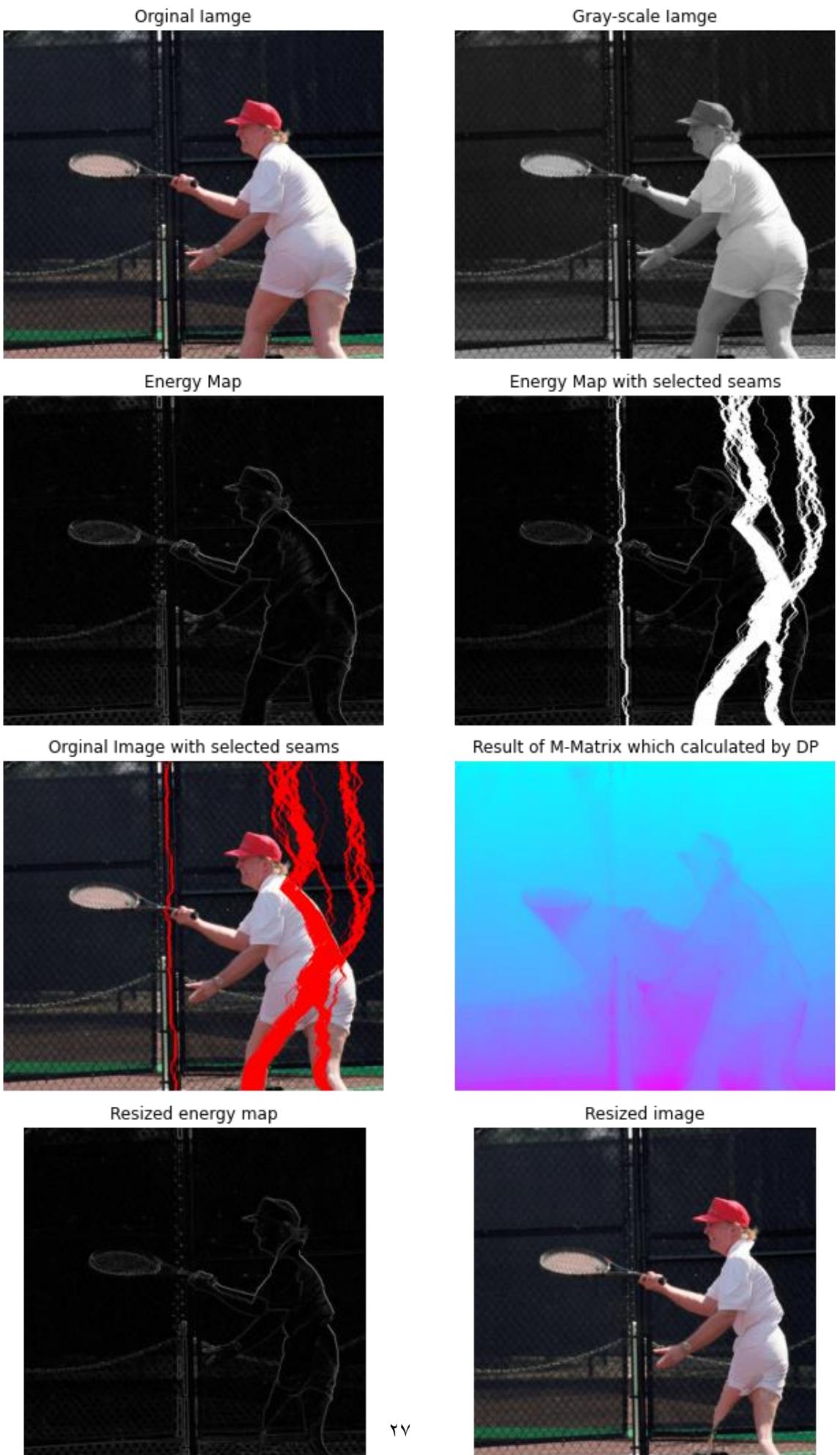
- ❖ در این تابع ابتدا تصویر به سطح خاکستری تبدیل شده و سپس گرادیان آن توسط تابع `gradient` از کتابخانه `numpy` محاسبه و نتیجه بازگردانده می شود.
- ❖ در این تابع ابتدا جهت کاهش بررسی می شود، اگر افقی بود، آن را 90 درجه دوران داده و روند را ادامه می دهد. (تغییر سایز در افق برابر است با تغییر سایز در عمود به شرط دوران 90 درجه ای) سپس توسط تابع `get_cost_matrix` ماتریس M را محاسبه می کند. سپس آخرین سطر ماتریس M را در نظر گرفته و اندیس معادل با مرتب شده مقدار آن را استخراج می کند. این اندیس ها نشان می دهد که نقطه شروع Seam در سطر آخر چه موقعیتی می باشد. سپس در یک حلقه به تعداد `Seam_num`، از سطر آخر پیکسل ها را انتخاب کرده و سپس شروع به پیمایش به بالا می کند و در هر مرحله یکی از سه پیکسل مجاور بالایی را که کمترین انرژی داشته باشند را برای حضور در Seam جهت حذف/اضافه پیکسل انتخاب می کند؛ در زمان انتخاب پیکسل از همسایه های بالایی بررسی می شود که قبل از آن پیکسل های `seam` قبلی حذف نشده باشد. اگر چنین بود اولین همسایه های بالایی در جهت مفروض را جایگزین و مقدار آن را مقایسه می کند. چگونگی کنترل این که چه پیکسل هایی قبل انتخاب شده اند توسط یک `mask` انجام می شود که مقدار صفر نمایشگر این است که پیکسل تا کنون حذف نشده است و در صورت غیر صفر، مقدارش نشان می دهد توسط چندمین `seam` انتخاب شده است. پس اتمام حلقه مذکور، این

به فرمت خواسته شده $\text{seam_num} * \text{width}(\text{height})$ تبدیل و بازگردانده می شود. در هر مرحله از این تابع تغییرات انتخاب پیکسل ها و رگه ها جهت گزارش بصری، در نقشه انرژی و خود تصویر نیز اعمال می شود.

:remove_seams این تابع نیز مانند تابع فوق ابتدا جهت را بررسی کرده و طبق توضیحات مذکور در صورت خواسته‌ی تغییرات در راستای افق، آن را ۹۰ درجه دوران می دهد تا پردازش بصورت عمودی ادامه یافته و در نهایت مجدد دوران معکوس یابد؛ این تابع ابتدا یک mask هم اندازه تصویر ورودی ساخته و آن را با داده های seam_list تکمیل می کند. سپس یک تصویر خالی در ابعاد جدید ساخته و پیکسل های مورد نظر از تصویر اصلی را توسط این ماسک فیلتر کرده و در تصویر جدید ذخیره کرده و آن را بر می گرداند. همچنین در انتهای بررسی می کند تا در صورت افقی بودن، آن مجدد معکوس دوران دهد.

نتایج صفحه ۲۷ حاصل اعمال کاهش سایز ۱۰ درصدی برای تصویر بازی تنیس ترامپ در راستای انتخاب عمودی رگه ها می باشد. همانطور که قابل مشاهده است کاهش ابعاد صورت گرفته به جهت معنایی چندان مطلوب نبوده و ناظر می تواند به جهت بصری و معنایی تشخیص دهد که پیکسل هایی از تصویر اصلی حذف شده اند یا دچار ناهنجاری و نامتعارفی می باشد چرا که می توان تشخیص داد پایی یک انسان عادی در زمان بازی تنیس نمی تواند به آن صورتی که در تصویر است قرار بگیرد. دلیل انتخاب Seam هایی در راستای پای ترامپ آن است که تغییرات محسوسی در آن مسیر وجود نداشته و سطح انرژی آن پایین می باشد(جزئیات نزدیک به هم وجود ندارد) در حالی که در سایز طرفین تقریباً وجود فنس یا ماقبی المان های تصویر باعث وجود انرژی شده است(جزئیات به هم نزدیک است). این نکته را می توان در خروجی مربوط به ماتریس M صفحه بعد ملاحظه است. نتایج صفحه ۲۸ و ۲۹ نیز حاصل اعمال کاهش سایز ۲۵ و ۵۰ درصدی برای تصویر بازی تنیس ترامپ در راستای عمودی می باشد که این دو نیز با وجود کاهش سایز در راستای خواسته شده، مشکل مذکور را دارند. دلیل این امر آن است که همه‌ی تعداد seam های مورد نظر همزمان از تصویر انتخاب و حذف می شود، در صورتی که اگر به ازای هر Seam تصویر را کاهش سایز داده و مجدد گرادیان و ماتریس M را محاسبه کنیم و سپس seam بعدی را انتخاب و حذف کنیم، نتیجه بهتری حاصل خواهد شد. سه تصویر خروجی زیر حاصل کاهش به ۱۰ و ۲۵ و ۵۰ درصدی می باشد با این تفاوت که ابتدا یک seam انتخاب می شود و سپس از تصویر حذف شده و سپس بر اساس تصویر جدید محاسبات مجدد انجام و seam بعدی انتخاب می شود؛ می بینیم که نتیجه‌ی حاصل اندکی بهتر از بدست آوردن و حذف همزمان seam ها می باشد.





Orginal lамge



Gray-scale lамge



Energy Map



Energy Map with selected seams



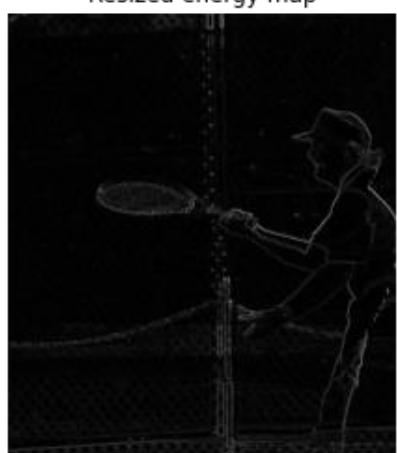
Orginal Image with selected seams



Result of M-Matrix which calculated by DP

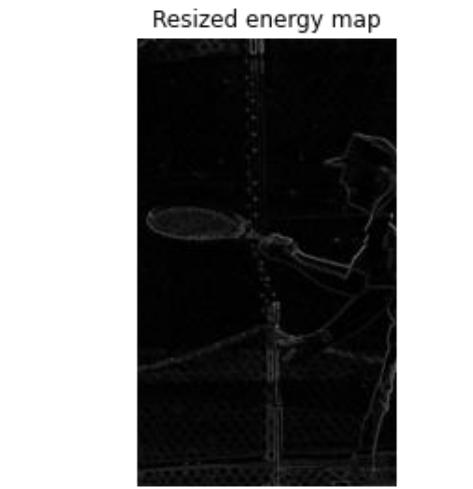
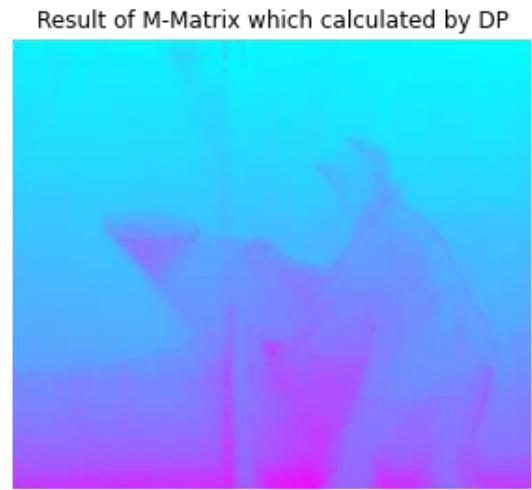
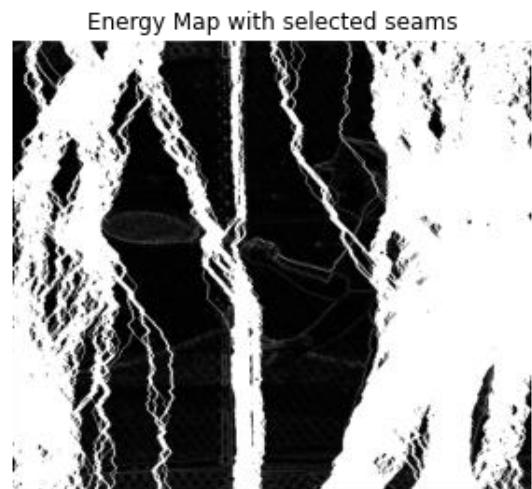


Resized energy map



Resized image





مسئله ۵ قسمت (b)

همانند قسمت a و با این تفاوت که مقدار پارامتر horizontal seam_dir برابر با seam_dir می باشد اقدام می کنیم. خروجی های مربوط به کاهش ۱۰ درصدی در صفحه ۳۱ قابل ملاحظه است. برخلاف قسمت a و همانطور که قابل مشاهده است کاهش ابعاد صورت گرفته به جهت معنایی مطلوب بوده و ناظر نمی تواند به جهت بصری و معنایی تشخیص دهد که پیکسل هایی از تصویر اصلی اصلا حذف شده است چرا که همه موارد طبیعی است. خروجی های مربوط به کاهش ۲۵ درصدی نیز در صفحه ۳۲ قرار داشته و کاهش آن نیز بسیار چشم گیر نبوده و شی اصلی که بدن ترامپ می باشد، حالت طبیعی یک فرد را نمایش می دهد و همانند قسمت a چشم انسان متوجه مورد خاصی نمی باشد.



گزارش های مربوط به کاهش ۵۰ درصدی در صفحه ۳۳ قرار گرفته است. در این کاهش و برخلاف دو مورد قبلی، عمل انجام شده به جهت معنایی مطلوب نبوده و ناظر می تواند به راحتی و به جهت بصری و معنایی تشخیص دهد که پیکسل های زیادی از تصویر و شی اصلی حذف شده است چرا که فرم عادی بدن دراز کشیده بهم ریخته است. دلیل این مشکل مشابه با موردی است که در قسمت a بصورت مفصل توضیح داده شده است. در صورتی که به جای محاسبه و حذف همزمان تعداد زیادی seam یا از تصویر اصلی، یک seam را حذف کرده و سپس مجدد seam بعدی را بر اساس تصویر جدید محاسبه کنیم، نتیجه‌ی نهایی بهتر خواهد بود. در مقابل کاهش سایز داده شده یک به یک قابل مشاهده است.

Orginal limage



Gray-scale limage



Energy Map



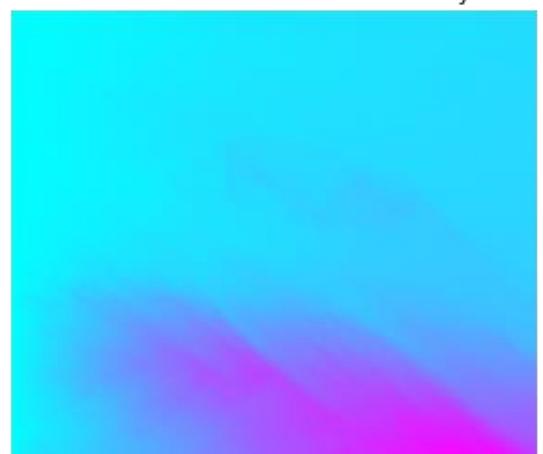
Energy Map with selected seams



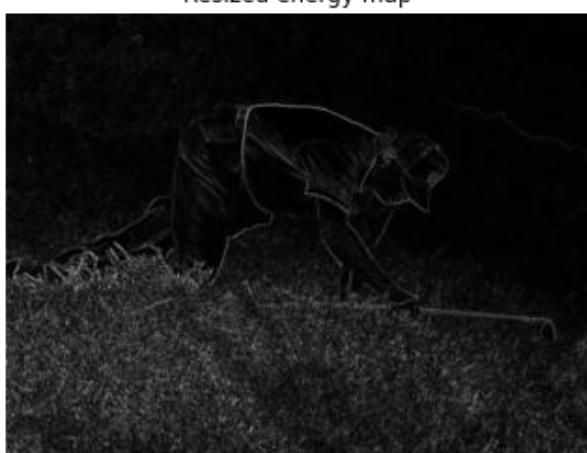
Orginal Image with selected seams



Result of M-Matrix which calculated by DP



Resized energy map



Resized image



Orginal Iamge



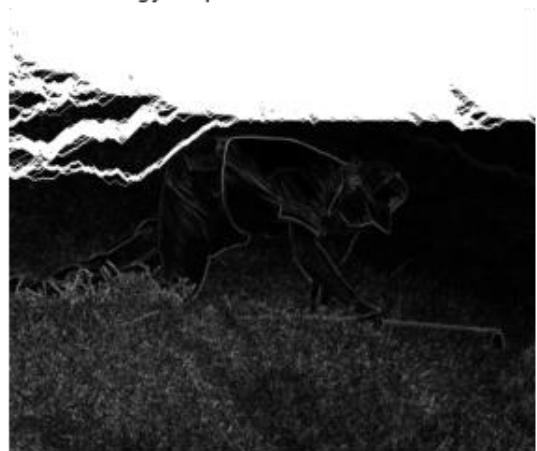
Gray-scale Iamge



Energy Map



Energy Map with selected seams



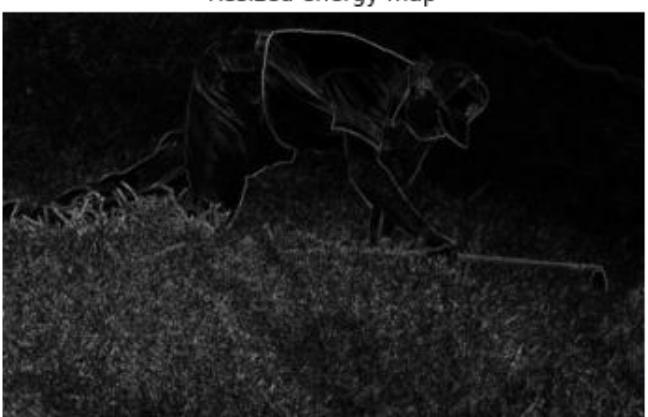
Orginal Image with selected seams



Result of M-Matrix which calculated by DP



Resized energy map



Resized image



Orginal Iamge



Gray-scale Iamge



Energy Map



Energy Map with selected seams



Orginal Image with selected seams



Result of M-Matrix which calculated by DP



Resized energy map



Resized image

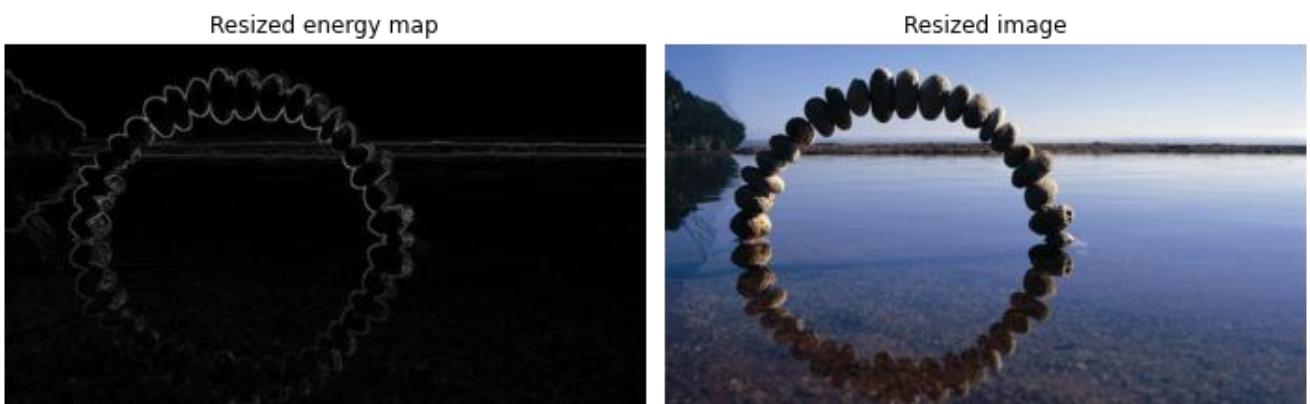
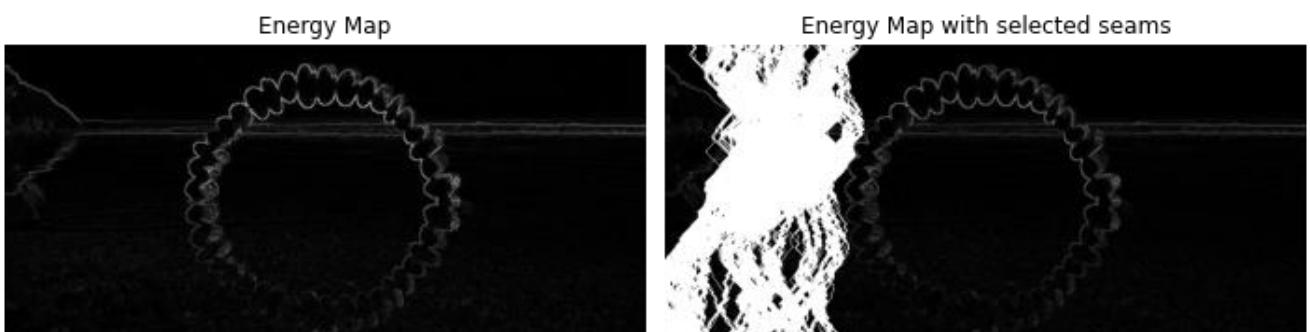


مسئله ۵ قسمت c و d)

طبق کارکرد الگوریتم، همواره سعی می شود رگه هایی از تصویر حذف شود که دارای کمترین تغییرات بر اساس نقشه انرژی باشد(اینجا نقشه انرژی ما همان گرادیان می باشد که حاصلش همان نمایان شده لبه ها می باشد). از این رو، این الگوریتم می تواند در مقابله با تصاویری که در کل لبهی کمی داشته باشند خوب عمل نماید. معمولاً تصاویری لبه های کم دارند که در راستای کاهش سایز کم تغییر بوده و دارای جزئیات ریز نباشند و به عبارت کلی تر تصویر یکدست تر و صاف بوده و جزئیات معنا دار از هم دور باشد؛ دلیل عملکرد بهتر در اینطور شرایط آن است که چشم انسان در اشیا و مناظر کلی متوجه تغییرات جزئی نمی شود زیرا مصدق آن می تواند در حقیقت وجود داشته باشد(مثلاً یک رودخانه می تواند عرض کوتاه یا بلندی داشته باشد و حذف پیکسل های یک رودخانه با عرض بلند انتظار غیر قابل باوری نیست).

لذا این الگوریتم معمولاً بیشتر در تصاویر مناظر landscape که تغییرات کم باشد از خود عملکرد بهتری نشان می دهد. یک نمونه عکس مناسب از منظره طبیعی و در حین جست و جوی در سرچ گوگل با کلمه کلیدی nature از این [لینک](#) با توضیحات فوق را یافته و الگویتم را با کاهش ۲۰ درصدی در راستای انتخاب رگه عمود بروی آن اعمال کردم. نتیجه در صفحه ۳۵ قابل مشاهده است. همانطور که مشهود است، با وجود حذف ۲۰ درصد پیکسل ها، با نگاه کردن نمی توان متوجه تغییرات شد چرا که به جهت منطقی همه موارد بصورت طبیعی در حقیقت وجود دارد. لذا عملکرد الگوریتم عالی بوده است.

با عنایت به توضیحات فوق می توان این انتظار را داشت که نقطه‌ی شکست این الگوریتم جایی است که جزئیات یک تصویر نزدیک به هم بوده و لبه های زیادی در تصویر موجود باشد لذا با حذف پیکسل ها، لبه ها و جزئیات زیادی را از دست داده و کیفیت کاهش سایز دچار افت می شود. این تیپ تصاویر می تواند عکس مدار های الکتریکی، چهره های افراد، عکس های متنی و... باشد. از این رو برای نمایش شکست این الگوریتم تصویر چهره جنیفر لارنس(بازیگر) را از اینترنت دانلود کرده و الگوریتم ۱۰ درصدی را روی آن اعمال کردم(حتی ۱۰ درصد کمتر از حالت فوق) که نتیجه آن در صفحه ۳۶ قابل ملاحظه است، همانطور که مشهود است در تصویر حاصل امکان تشخیص اینکه پیکسل هایی حذف شده است کاملاً وجود دارد زیرا چهره فرد نمی تواند این گونه باشد.



Orginal limage



Gray-scale limage



Energy Map



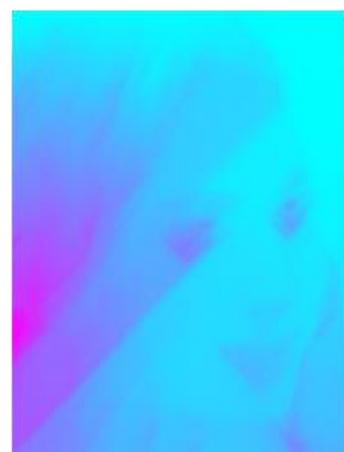
Energy Map with selected seams



Orginal Image with selected seams



Result of M-Matrix which calculated by DP



Resized energy map



Resized image



برای افزایش سایز مورد نظر، طبق راهنمایی انجام شده یک تابع با نام `add_seams` پیاده سازی شده است که بدنی اصلی آن شبیه به تابع پیاده سازی شده `remove_seams` می باشد. این تابع طبق لیست ورودی `seam` یک ماسک هم اندازه تصویر می سازد و سپس یک تصویر خالی با ابعاد بزرگتر به عنوان خروجی ایجاد میکند؛ سپس شروع به پیمایش تصویر ورودی بر اساس ماسک می کند. در صورتی که به پیکسل های ماسکی نرسیده باشد، پیکسل متناظر را در تصویر خروجی کپی می کند؛ در صورتی که به پیکسل ماسکی برسد، مقدار پیکسل متناظر را در یک پیکسل بعدی نیز کپی می کند و به ازای انجام هر بار از این عملیات، اندیس کنترل در تصویر خروجی را یک واحد افزایش می دهد تا ترتیب اضافه نمودن پیکسل ها با مشکل مواجه نشود.

به ازای تمامی تصویر ورودی، فرآیند افزایش سایز انجام شده و در صفحات آتی گزارش ها و خروجی ها حاصل قبل ملاحظه است. ترتیب گزارش ها به ترتیب افزایش درصد از ۱۰ به ۲۵ و ۵۰ درصد می باشد.

Orginal lамge



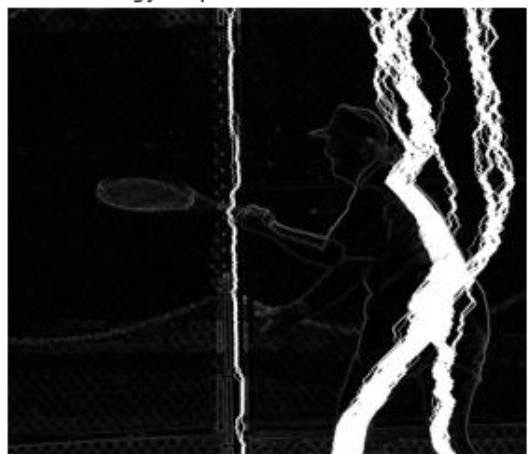
Gray-scale lамge



Energy Map



Energy Map with selected seams



Orginal Image with selected seams



Result of M-Matrix which calculated by DP



Resized energy map



Resized image



Orginal Iamge



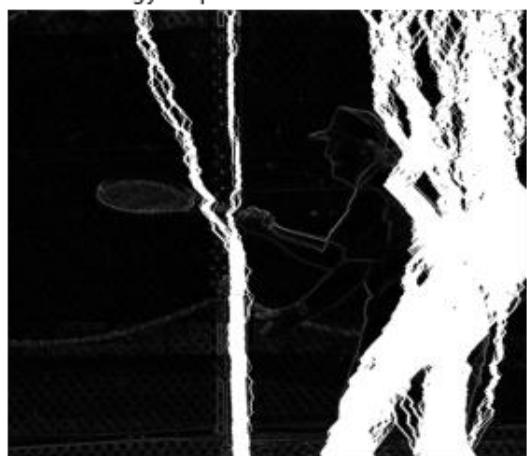
Gray-scale Iamge



Energy Map



Energy Map with selected seams



Orginal Image with selected seams



Result of M-Matrix which calculated by DP

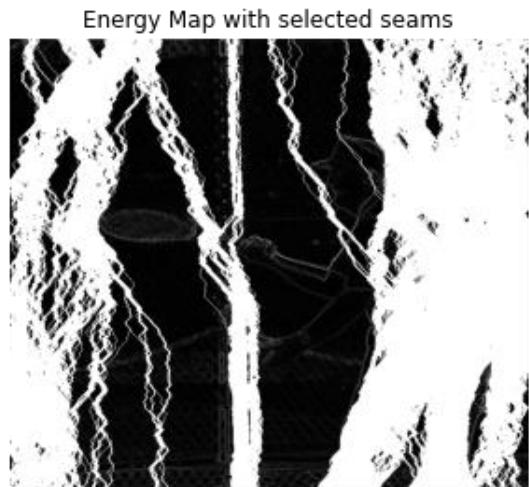


Resized energy map



Resized image





Orginal iamge



Gray-scale iamge



Energy Map



Energy Map with selected seams



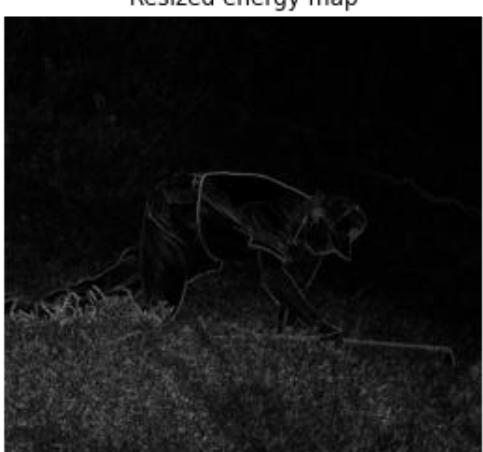
Orginal Image with selected seams



Result of M-Matrix which calculated by DP



Resized energy map



Resized image



Orginal iamge



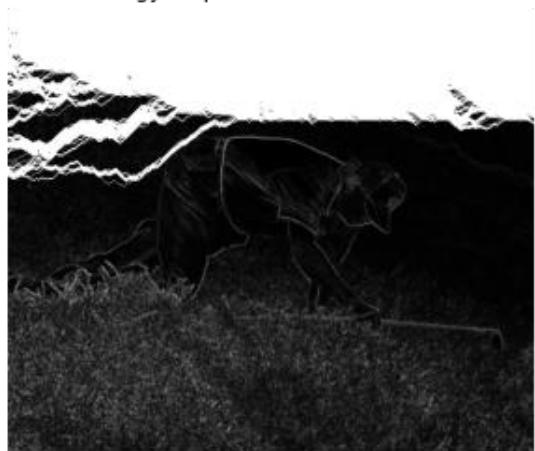
Gray-scale iamge



Energy Map



Energy Map with selected seams



Orginal Image with selected seams



Result of M-Matrix which calculated by DP



Resized energy map



Resized image



Orginal iamge



Gray-scale iamge



Energy Map



Energy Map with selected seams



Orginal Image with selected seams



Result of M-Matrix which calculated by DP



Resized energy map



Resized image



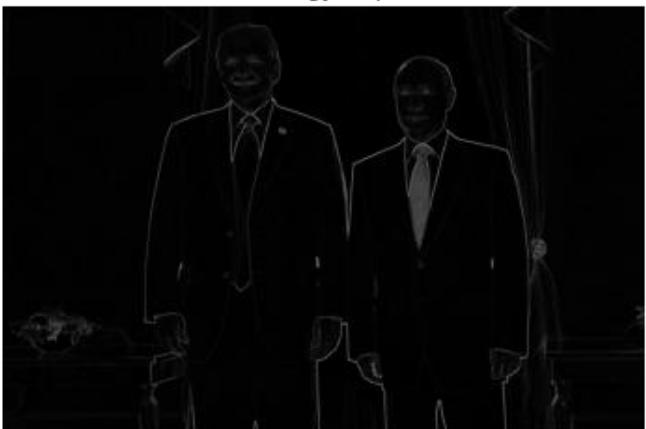
Orginal Iamge



Gray-scale Iamge



Energy Map



Energy Map with selected seams



Orginal Image with selected seams



Result of M-Matrix which calculated by DP



Resized energy map



Resized image



Orginal lамge



Gray-scale lамge



Energy Map



Energy Map with selected seams



Orginal Image with selected seams



Result of M-Matrix which calculated by DP



Resized energy map



Resized image



Orginal lамge



Gray-scale lамge



Energy Map



Energy Map with selected seams



Orginal Image with selected seams



Result of M-Matrix which calculated by DP



Resized energy map



Resized image



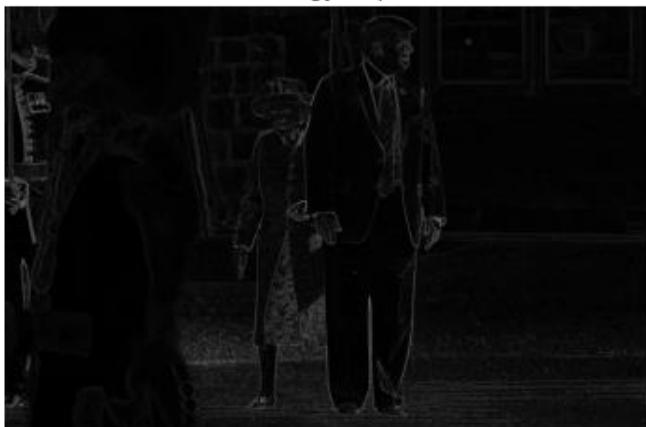
Orginal Iamge



Gray-scale Iamge



Energy Map



Energy Map with selected seams



Orginal Image with selected seams



Result of M-Matrix which calculated by DP



Resized energy map



Resized image



Orginal Iamge



Gray-scale Iamge



Energy Map



Energy Map with selected seams



Orginal Image with selected seams



Result of M-Matrix which calculated by DP



Resized energy map



Resized image



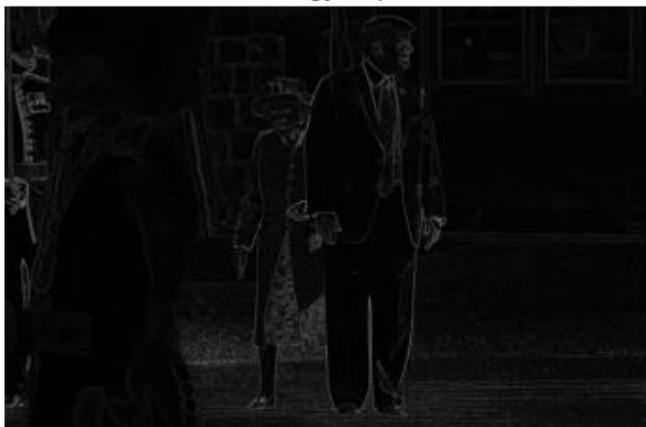
Orginal Iamge



Gray-scale Iamge



Energy Map



Energy Map with selected seams



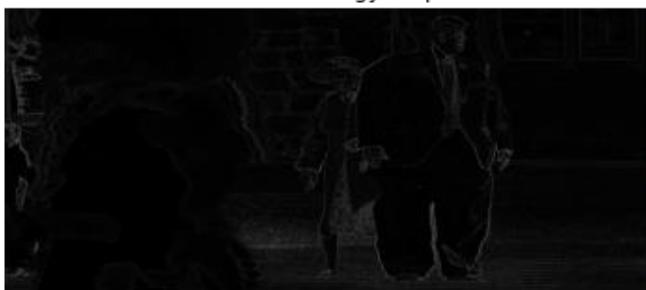
Orginal Image with selected seams



Result of M-Matrix which calculated by DP



Resized energy map



Resized image



مسئله ۵ قسمت (f)

برای حذف ملکه و پوتین از عکس ترامپ بایستی تغییراتی ایجاد کنیم تا انتخاب seam از مسیر هایی باشد که ماسک آنها حضور دارند. برای این منظور ابتدا نقشه انرژی را محاسبه می کنیم و سپس ضریبی بزرگی از معکوس ماسک داده شده(۵ برابر) را به نقشه انرژی اضافه می کنیم. در این صورت نواحی ماسک شده دارای انرژی کمتری می شوند و الگوریتم seam هایی را انتخاب می کند که از آن عبور کنند. برای حذف ملکه ۱۴ و برای حذف پوتین ۲۸.۲ درصد سایز تصویر را کاهش سایز می دهیم(این اعداد با سعی و خطا بدست آمده است) و پس از آن در صورت وجود رگه هایی از ماسک، یک به یک رگه ها را حذف می کنیم تا تصویر از حضور افراد مذکور کاملا خالی شود. از جایی که مطلوب ما آن است که صرفا افراد حذف شوند و به شکل صحنه تا حد امکان آسیبی وارد نشود، انتخاب و حذف رگه بصورت عمود اعمال شده است. گزارش و نتیجه حاصل از حذف ملکه در صفحه ۵۱ و پوتین در صفحه ۵۲ قرار داده شده است.

کیفیت حذف ماسک ها و نتیجه‌ی خروجی به ازای تصویر ترامپ و پوتین بهتر از تصویر ترامپ و ملکه حاصل شده است چرا که قسمت مرز مشترک ترامپ با پوتین در ناحیه آرنج و دست تخریب کمتری را متحمل شده است و وقتی ناظر به آن می نگرد، احساس نامعمول بودن کمتری به آن دست می دهد.

Orginal Iamge



Gray-scale Iamge



Energy Map



Energy Map with selected seams



Orginal Image with selected seams



Result of M-Matrix which calculated by DP



Resized energy map



Resized image



Orginal Iamge



Gray-scale Iamge



Energy Map



Energy Map with selected seams



Orginal Image with selected seams



Result of M-Matrix which calculated by DP



Resized energy map



Resized image



مسئله ششم

مسئله ۶ قسمت (a)

بله امکان کانولوشن یک فیلتر با سایز بزرگتر از سایز تصویر ورودی وجود دارد. برای این کار صرفاً کافی است zero padding را در تصویر ورودی اعمال کنیم بدین گونه که در راستا(ها)ی که سایز تصویر ما کوچکتر از سایز فیلتر مورد نظر است، سطر/ستون های تمام صفر اضافه کرده و سپس کانولوشن را انجام دهیم. تصویر زیر اضافه نمودن دو سطر و دو ستون تمام صفر به یک تصویر 5×5 را نمایش می دهد که سایز آن را به 7×7 تبدیل می کند.

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0					0	0	
0	0					0	0	
0	0					0	0	
0	0					0	0	
0	0					0	0	
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

مسئله ۶ قسمت (b)

بله به حالت یکتایی همگرا می شود زیرا خروجی لپلاسین بر فیلتر گوسی ضمن حذف نویز و تار کردن مرز ها، لبه های تصویر را استخراج می کند و پس از اعمال اولین فیلتر، لبه های تصویر اورجینال نمایان می شود؛ حال در تصویر حاصل، لبه های استخراجی خود نیز لبه های جدیدی با عرض کمتر هستند و در صورت اعمال مجدد فیلتر گوسی این لبه های با عرض کم مجدد تار تر شده و با اعمال لپلاسین همین لبه ها دوباره و با عرض کمتر استخراج می شود و این گونه به یک حالتی که همه لبه ها از بین رفته اند هم گرا می شویم؛ به جهت ریاضیاتی نیز می توان نشان داد که مشتق رفته از بین می رود و این دلیل همگرایی به یک حالت پایدار می باشد.

مسئله ۶ قسمت (c)

پیدا کردن الگو در تصویر با استفاده از فیلتر ها در همهی حالات امکان پذیر نیست(یافتن + ها در همهی سایز های ممکن) زیرا سایز فیلتر ثابت می باشد و در قبال الگو های مشابه با ابعاد بزرگتر یا کوچکتر یا ... درست عمل نمی کند زیرا انعطاف پذیری در بسط و پیدایش الگو را ندارد و صرفا سعی در سنجش شباهت دقیق را دارد و این باعث می شود الگو های شناسایی شده بیشتر یا کمتر از الگو های واقعی موجود در تصویر باشد؛ قابل ذکر است با اعمال فیلتر برای تشخیص الگو(مانند آنچه که در بند زیر مثال زده می شود) صرفا می توان این اطمینان

را پیدا کرد که الگوهای دقیقاً یکسان با ابعاد دقیقاً یکی با فیلتر حتماً یافت شده‌اند که البته این مورد دستاورده‌ی در استخراج الگوی سراسری محسوب نمی‌شود.

فیلتر زیر را در نظر بگیریم؛ اگر این فیلتر را بروی یک تصویر دودوئی اعمال کنیم، می‌توانیم پیکسل‌هایی که مقدار ۵ را در خروجی دارند به عنوان مرکز علامت + که ابعاد آن 3×3 می‌باشد گزارش کنیم:

0.2	1	0.2
1	1	1
0.2	1	0.2

مسئله ۶ قسمت (d)

این موضوع در کلاس درس کاملاً بحث و بررسی شده است و توضیحات ادامه حاصل بحث کلاسی می‌باشد: قطعاً اعمال دو فیلتر تک بعدی بسیار بهتر از اعمال یک فیلتر دو بعدی می‌باشد چرا تعداد محاسبات کمتری را می‌طلبد. اگر تصویر ما ابعاد با عرض h و طول w باشد؛ در صورت استفاده از دو فیلتر تک بعدی، ابتدا اولین فیلتر یک بعدی در یکی از راستاهای طول و یا عرض (دلخواه) محاسبه می‌شود که به تعداد $h * w$ بار عملیات نیاز دارد. سپس دومین فیلتر یک بعدی (در راستای دیگر) نیز روی آن بصورت مشابه اعمال می‌شود. اما در صورت استفاده از یک فیلتر دو بعدی، برخی از پیکسل‌ها چندین بار (بسته به سایز فیلتر) در عملیات‌های محاسباتی حضور پیدا می‌کند که باعث افزونگی حجم پردازشی می‌باشد.

مسئله ۶ قسمت (e)

میانگین گیری ساده در تصاویر باعث کاهش نویز و خصوصاً نویز های تصادفی می‌شود؛ زیرا زمانی که نویز‌ها تصادفی و در برخی نواحی و در برخی پیکسل‌ها وجود داشته باشد، احتمال اینکه در همسایگی‌های آن پیکسل مجدد نویز وجود داشته باشد بسیار کم می‌باشد. (البته به تابع توزیع احتمال نویز تصادفی بستگی دارد) از این رو وقتی مقدار یک پیکسل با مقدار میانگین پنجره متناظرش جایگذین می‌شود، در این صورت حتی اگر نویزی در آن پیکسل وجود داشته باشد، تاثیر آن نویز در آن مقدار جدید کاسته می‌شود زیرا میانگین گیری ساده در پنجره بدون وزن بوده و مقدار نویزی با سایر مقادیر درست (true) میانگین گرفته می‌شود و از جایی که تعداد مقادیر درست بیشتر است، مقدار حاصل نیز کم تاثیرتر از نویز و نزدیک به مقدار صحیح اولیه بدست می‌آید.