

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

تمرین دوم درس تصویر پردازی رقمه‌ی

دکتر رحمتی

غلامرضا دار ۴۰۰۱۳۱۰۱۸

بهار ۱۴۰۱

فهرست مطالب

۳	سوال (۱)
۱۱	سوال (۲)
۲۷	سوال (۳)
۴۴	سوال (۴)
۵۲	سوال (۵)
۹۳	سوال (۶)

سوال (۱)

بخش (a) از آنجایی که بزرگترین مقدار ممکن برای تصویر ۷ می‌باشد، برای بدست آوردن متمم تصویر کافی است مقدار هر پیکسل را از ۷ کم کنیم و در آن پیکسل جای‌گذاری کنیم به این ترتیب ۷ به ۰ تبدیل می‌شود، ۶ به ۱، ۵ به ۲ و الی آخر. برای نمایش بهتر تصویر حاصل، اعداد به دست آمده به کمک `matplotlib` رسم شدند اما محاسبه اعداد به صورت دستی انجام گرفت.

```
1 def disp_matrix(A, s=10):
2     fig, ax = plt.subplots(1, 1, figsize=(s,s))
3     ax.matshow(A, cmap='gray', vmin=A.min()-3, vmax=A.max()+1)
4     ax.grid(False)
5
6     for i in range(A.shape[0]):
7         ax.axhline(y=i+0.5, color='k', linewidth=1.5)
8         ax.axvline(x=i+0.5, color='k', linewidth=1.5)
9
10    font_size = 35 if A.max() < 10 else 25
11    for (i, j), z in np.ndenumerate(A):
12        ax.text(j, i, f"{z}", ha='center', va='center', color='k', fontsize=font_size, fontweight='bold')
13
14 plt.show()
✓ 0.6s
```

ماتریس متمم:

	0	1	2	3	4	5	6	7
0	2	1	2	1	2	5	6	6
1	3	3	2	3	2	4	5	5
2	2	2	4	2	1	4	5	5
3	3	2	4	3	4	2	4	4
4	3	3	5	4	1	1	2	4
5	2	3	5	1	1	1	1	4
6	2	2	3	3	2	3	3	4
7	5	5	4	4	5	6	5	6

بخش (b) در این بخش با استفاده از رابطه زیر به جداسازی bit-plane های تصویر می پردازیم.

$$a_k = \left\lfloor \frac{I}{2^k} \right\rfloor \bmod 2$$

Bit-plane 0

	0	1	2	3	4	5	6	7
0	1	0	1	0	1	0	1	1
1	0	0	1	0	1	1	0	0
2	1	1	1	1	0	1	0	0
3	0	1	1	0	1	1	1	1
4	0	0	0	1	0	0	1	1
5	1	0	0	0	0	0	0	1
6	1	1	0	0	1	0	0	1
7	0	0	1	1	0	1	0	1

Bit-plane 1

	0	1	2	3	4	5	6	7
0	0	1	0	1	0	1	0	0
1	0	0	0	0	0	1	1	1
2	0	0	1	0	1	1	1	1
3	0	0	1	0	1	0	1	1
4	0	0	1	1	1	1	0	1
5	0	0	1	1	1	1	1	1
6	0	0	0	0	0	0	0	1
7	1	1	1	1	1	0	1	0

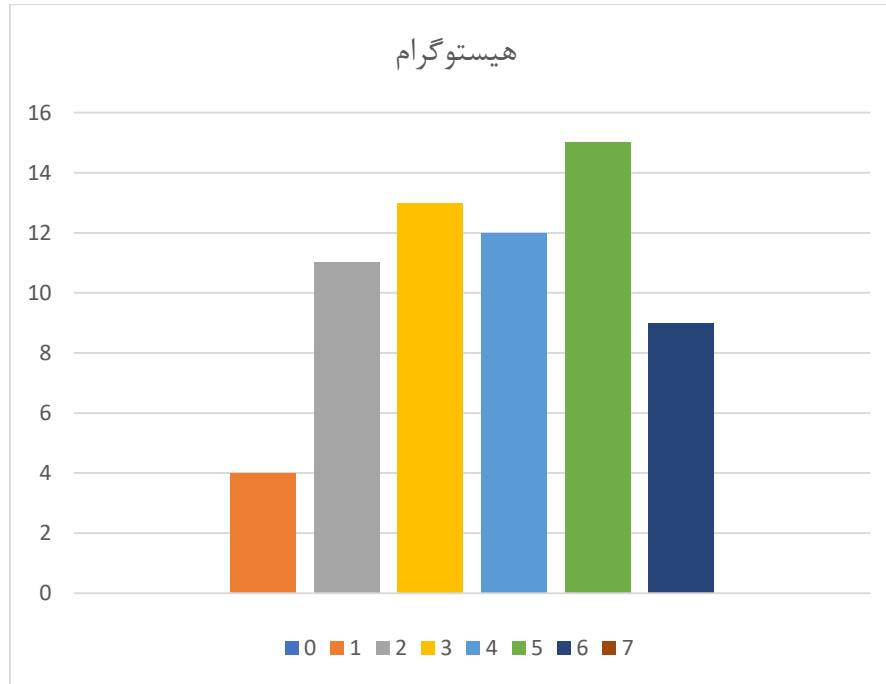
Bit-plane 2

	0	1	2	3	4	5	6	7
0	1	1	1	1	1	0	0	0
1	1	1	1	1	1	0	0	0
2	1	1	0	1	1	0	0	0
3	1	1	0	1	0	1	0	0
4	1	1	0	0	1	1	1	0
5	1	1	0	1	1	1	1	0
6	1	1	1	1	1	1	1	0
7	0	0	0	0	0	0	0	0

Bit-plane 3

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0

بخش c) در این بخش تعداد پیکسل‌های با مقادیر مختلف را می‌شماریم و در نمودار ستونی نمایش میدهیم.



بخش d) همانطور که دیده می‌شود در مقدار ۴ یک Valley داریم. تمام مقادیر بیشتر از ۴ را به ۷ (ماکسیمم مقدار) و مقادیر کمتر از ۴ را به ۰ (مینیمم مقدار) نسبت میدهیم.

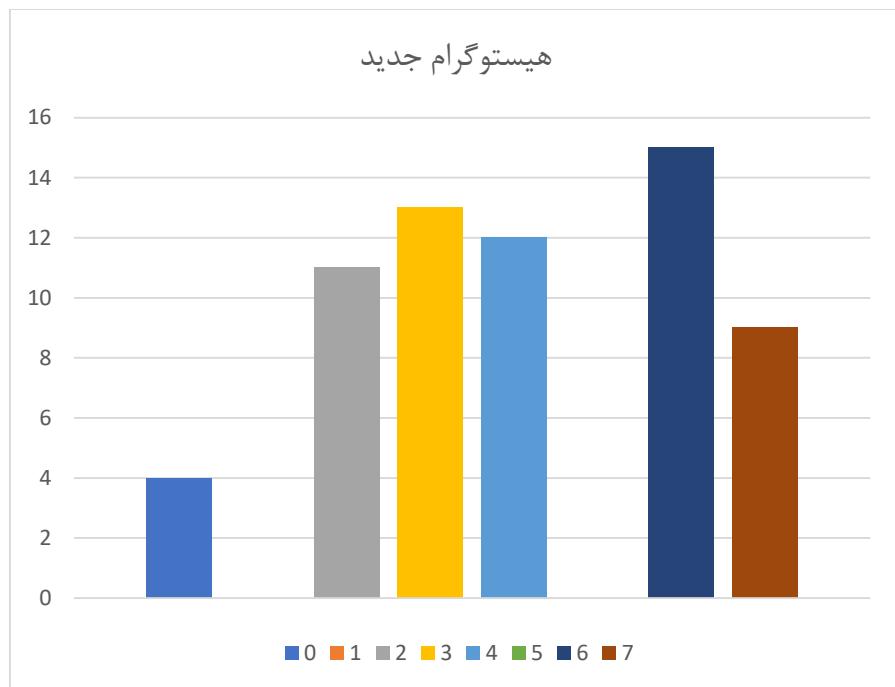
	0	1	2	3	4	5	6	7
0	7	7	7	7	7	0	0	0
1	0	0	7	0	7	0	0	0
2	7	7	0	7	7	0	0	0
3	0	7	0	0	0	7	0	0
4	0	0	0	0	7	7	7	0
5	7	0	0	7	7	7	7	0
6	7	7	0	0	7	0	0	0
7	0	0	0	0	0	0	0	0

بخش e) در این بخش تمام مقادیر را منهای مینیمم مقدار موجود میکنیم (۱) سپس تمام اعداد حاصل را بر ماسکیمم جدید (۵) تقسیم میکنیم. با این کار تمام مقادیر پیکسل ها در بازه ۰ تا ۱ و به صورت اعشاری قرار دارند. سپس با ضرب کردن تمام مقادیر در ۲۵۵ این مقادیر به بازه ۰ تا ۲۵۵ بردگ می شوند.

۰	۱	۲	۳	۴	۵	۶	۷	
۰	204	255	204	255	204	51	0	0
۱	153	153	204	153	204	102	51	51
۲	204	204	102	204	255	102	51	51
۳	153	204	102	153	102	204	102	102
۴	153	153	51	102	255	255	204	102
۵	204	153	51	255	255	255	255	102
۶	204	204	153	153	204	153	153	102
۷	51	51	102	102	51	0	51	0

بخش f) در این بخش میخواهیم عمل Histogram Equalization را انجام بدھیم. به این منظور، جدول فراوانی و فراوانی نسبی هر مقدار را مینویسیم، به آن ستون فراوانی تجمعی را اضافه میکنیم. سپس این فراوانی تجمعی را در ۷ ضرب میکنیم و مقدار Equalize شده متناظر با هر مقدار اصلی بدست می آید.

مقدار	فراوانی	فراوانی نسبی	فراوانی تجمعی	مقدار جدید	فراوانی جدید
۰	۰	۰/۰	۰/۰	۰	۴
۱	۴	۰/۰۶۲۵	۰/۰۶۲۵	۰	
۲	۱۱	۰/۱۷۱۸	۰/۲۳۴۳	۲	۱۱
۳	۱۳	۰/۲۰۳۱	۰/۴۳۷۴	۳	۱۳
۴	۱۲	۰/۱۸۷۵	۰/۶۲۴۹	۴	۱۲
۵	۱۵	۰/۲۳۴۳	۰/۸۵۹۳	۶	۱۵
۶	۹	۰/۱۴۰۶	۰/۹۹۹۹	۷	۹
۷	۰	۰/۰	۰/۹۹۹۹	۷	



بخش g) در این بخش ابتدا هیستوگرام ها را Equalize میکنیم .

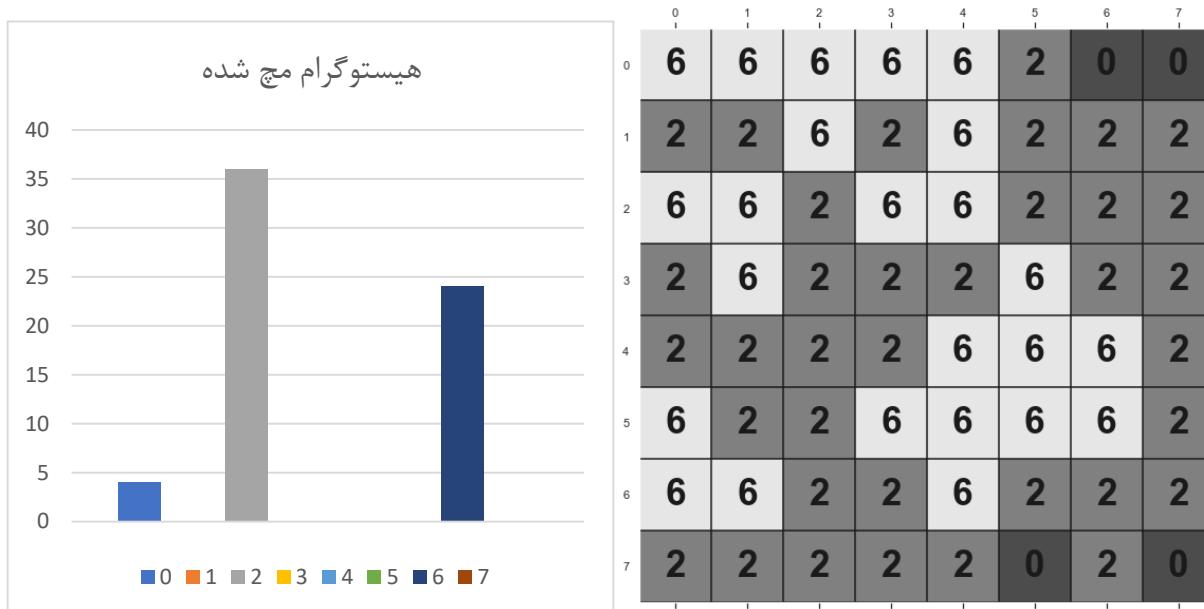
هیستوگرام مقصد

مقدار	فراوانی	فراوانی نسبی	فراوانی تجمعی	مقدار جدید
۰	۰	۰/۰	۰/۰	۰
۱	۰	۰/۰	۰/۰	۰
۲	۵۱۲	۰/۵	۰/۵	۴
۳	۰	۰/۰	۰/۵	۴
۴	۰	۰/۰	۰/۵	۴
۵	۰	۰/۰	۰/۵	۴
۶	۵۱۲	۰/۵	۱/۰	۷
۷	۰	۰/۰	۱/۰	۷

سپس از مقادیر Equalize شده به عنوان پلی بین دو هیستوگرام استفاده میکنیم. هر مقدار در هیستوگرام مبدا به مقداری در هیستوگرام مقصد نسبت داده میشود که پس از Equalize شدن این دو مقدار به یک مقدار نسبت داده شوند.

Value A	Equalized A	Equalized B	Value B
.	.	.	.
.	.	.	1
2	2	4	2
3	3	4	3
4	4	4	4
5	6	4	5
6	7	7	6
7	7	7	7

راهنمایی از <https://theailearner.com/2019/04/10/histogram-matching-specification>



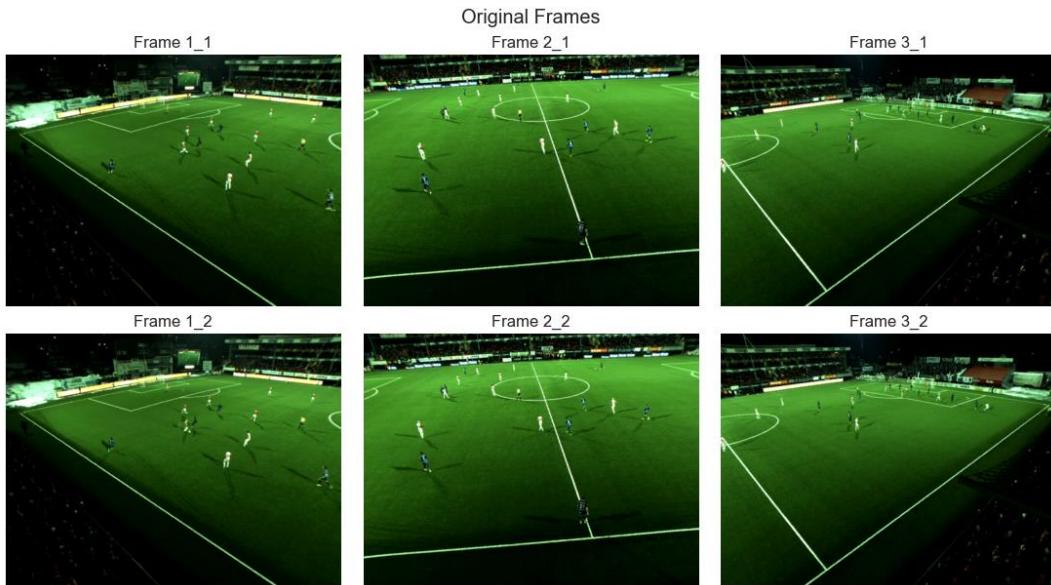
بخش **h**) برای این منظور از رابطه زیر کمک میگیریم

$$s = T(r) = (N - 1) \int_0^r p_r(w) dw$$

$$\begin{aligned} s &= T(r) = (2 - 1) \int_0^r \frac{e^w - 2}{e - 1} dw \\ &= (2 - 1) \left(\left(\frac{e^r - 2r}{e - 1} \right) - \left(\frac{e^0 - 0}{e - 1} \right) \right) \\ &= \left(\frac{e^r - 2r - 1}{e - 1} \right) \end{aligned}$$

(۲) سوال

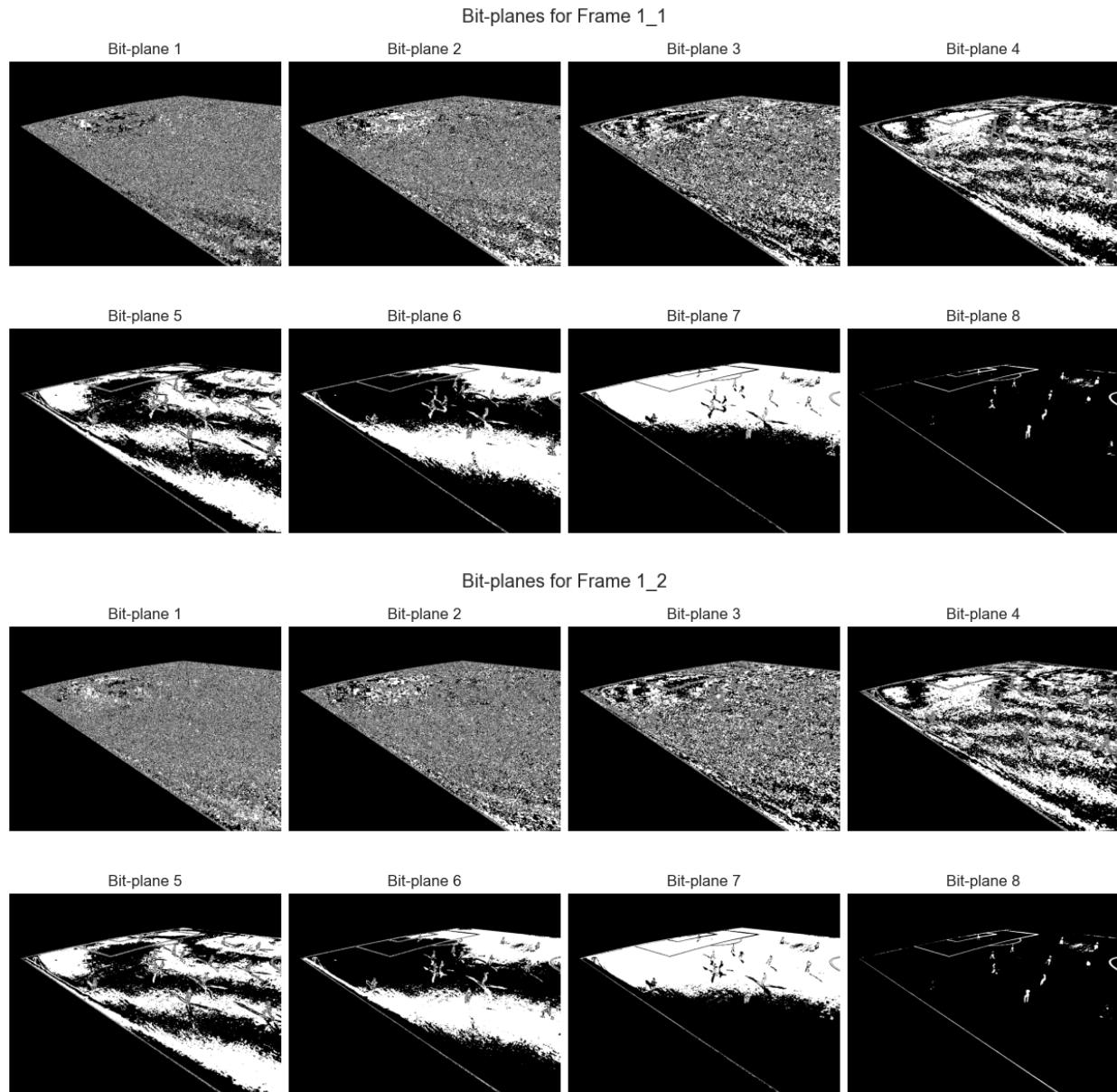
ابتدا تصاویر را فراخوانی میکنیم.



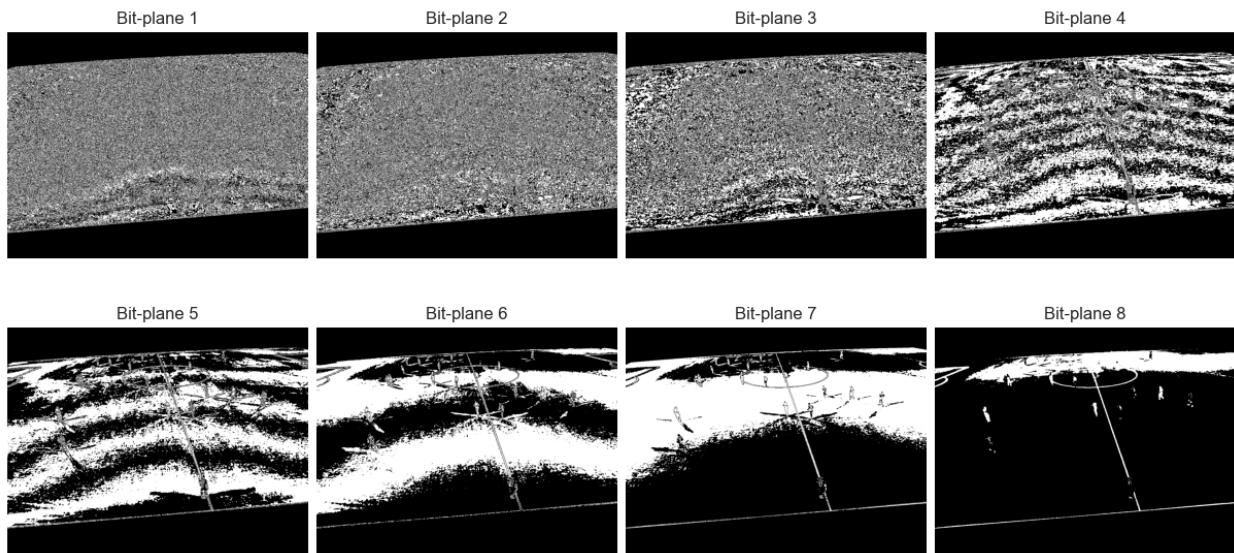
طبق آزمایش‌هایی که برای این سوال انجام دادیم به این نتیجه رسیدیم که نتایج خیلی قابل قبولی از مرحله آخر دریافت نمی‌کنیم. بنابراین تعدادی تکنیک را به کار گرفتیم. ابتدا تصاویر اصلی را کمی *Blur* کردیم. این کار باعث شد تا حدی نتایج XOR کم نویز تر شود ولی در عین حال موقعیت بازیکنان با دقیقیت قبلی بدست می‌آمد. تکنیک بعدی که به کار رفت، استفاده از یک ماسک برای جداسازی زمین فوتبال از ناحیه تماشاچیان و بقیه استادیوم بود. با توجه به اینکه دوربین‌ها در سه زاویه‌ای که تصاویر ضبط شده اند به نظر ثابت می‌رسند، به صورت دستی یک ماسک برای هر زاویه دوربین طراحی کردیم و در همین مرحله اول به تصاویر اعمال کردیم.



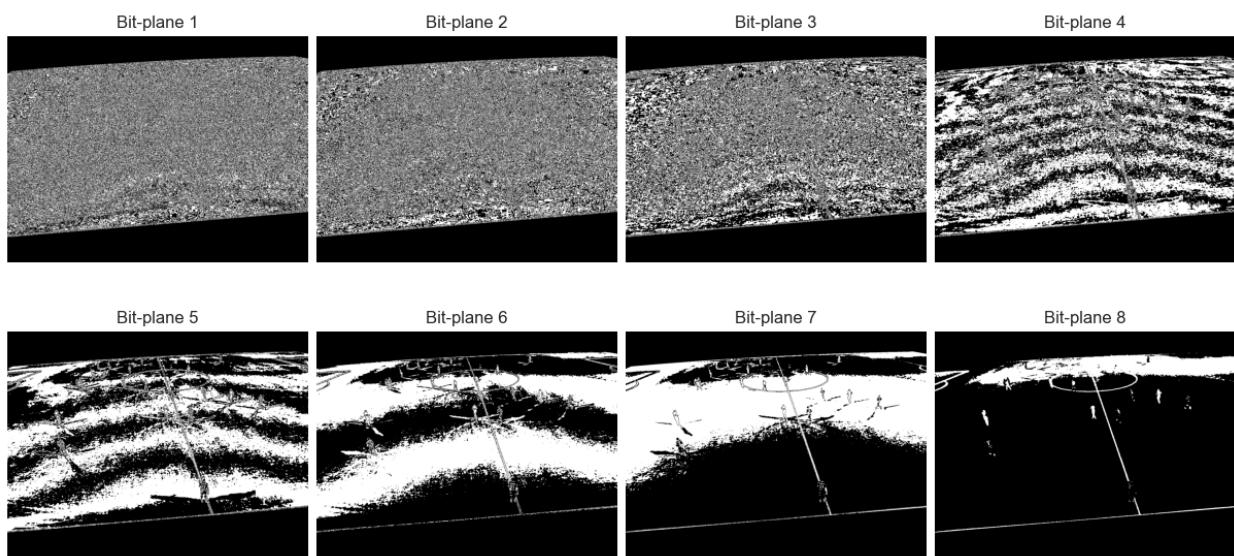
بخش a) در این بخش، با استفاده از رابطه‌ی داده شده در صورت سوال، تصاویر را به bit-plane های سازنده آنها تقسیم میکنیم.
۶ تصویری که در ادامه میبینید مربوط به استخراج bit-plane های ۶ تصویر ورودی هستند.



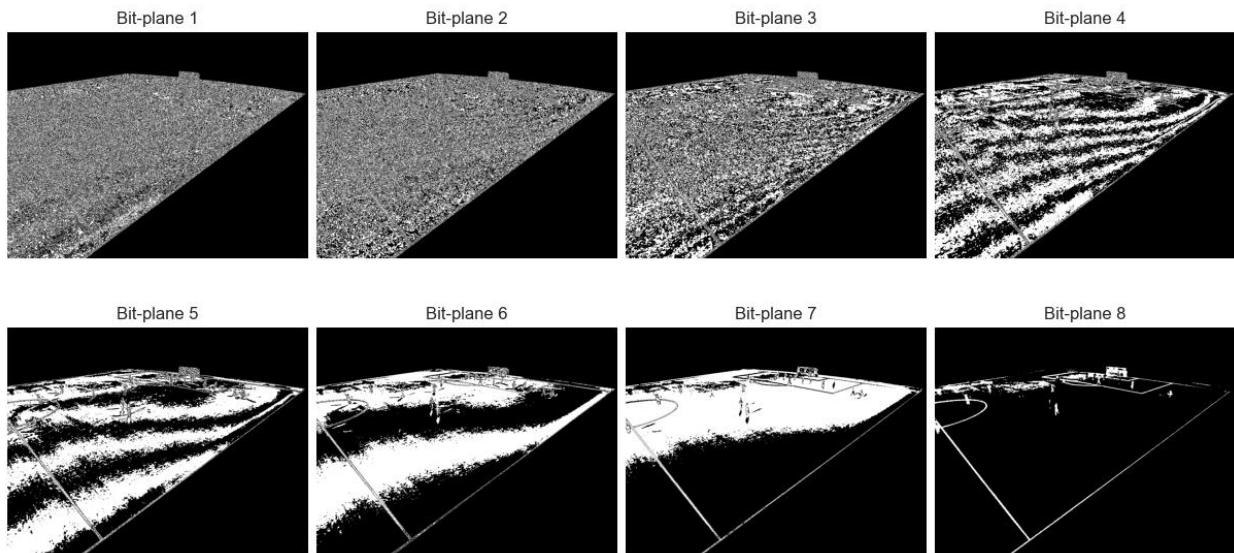
Bit-planes for Frame 2_1



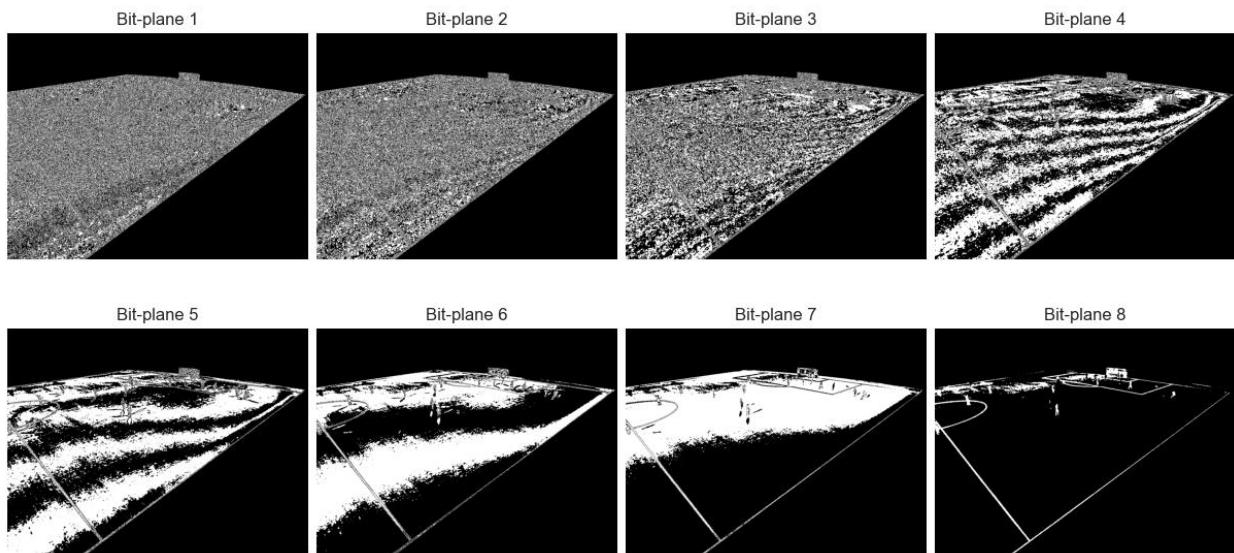
Bit-planes for Frame 2_2



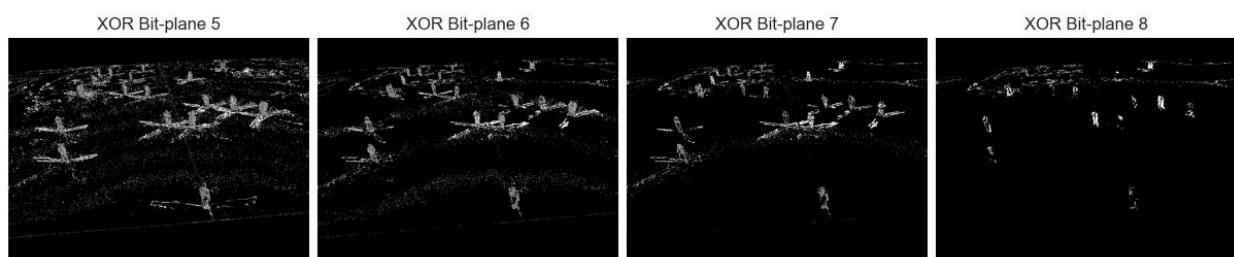
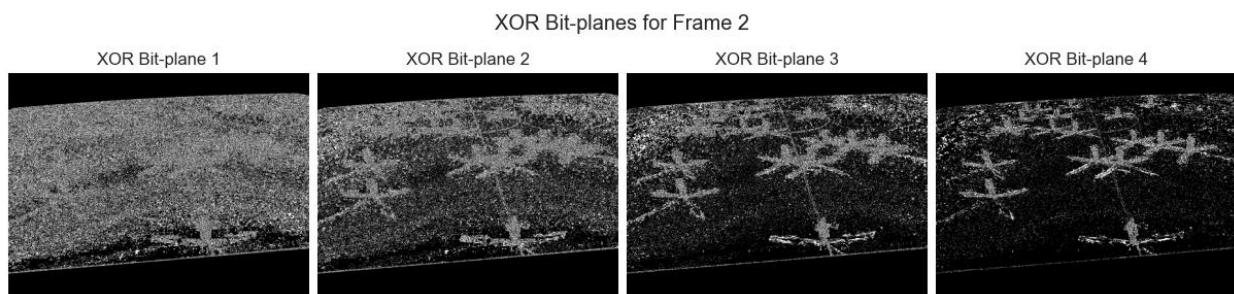
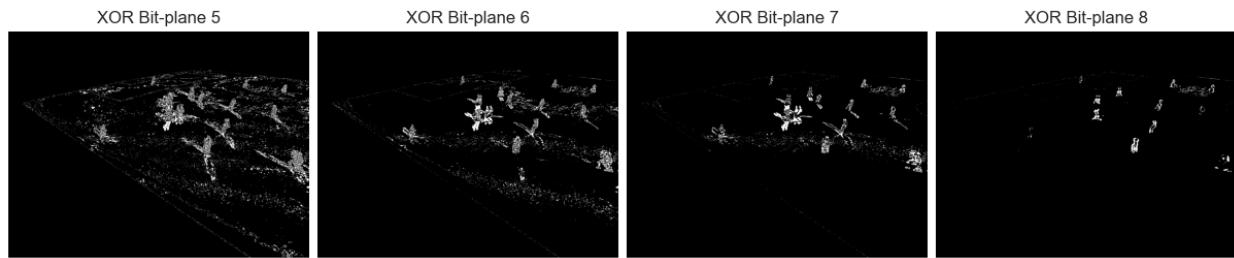
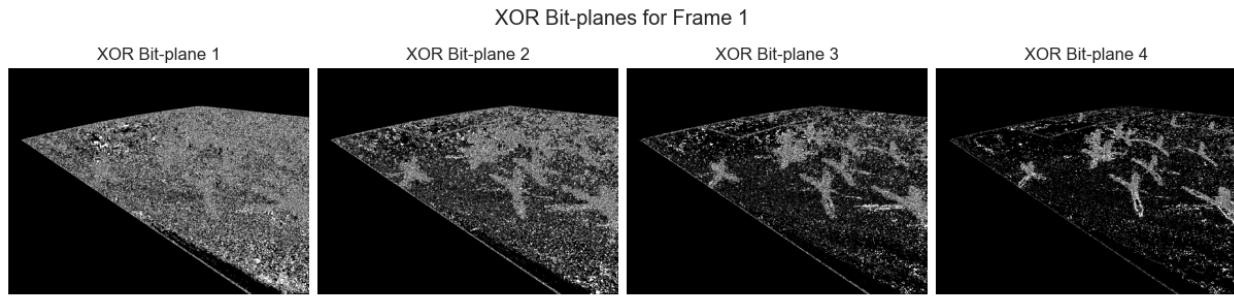
Bit-planes for Frame 3_1

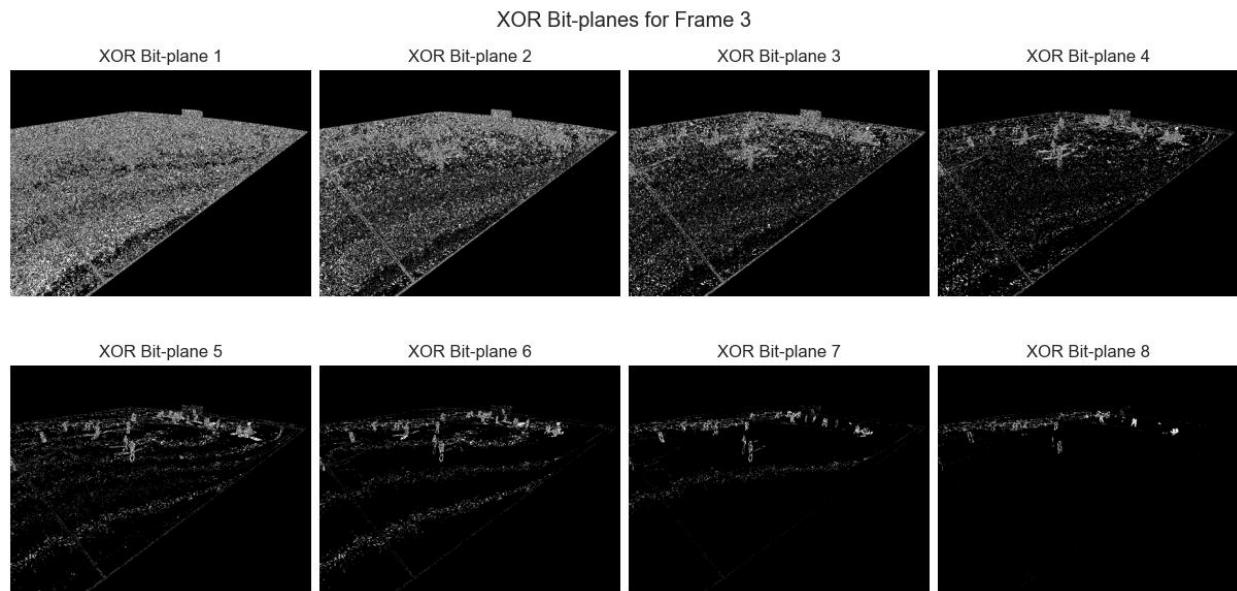


Bit-planes for Frame 3_2

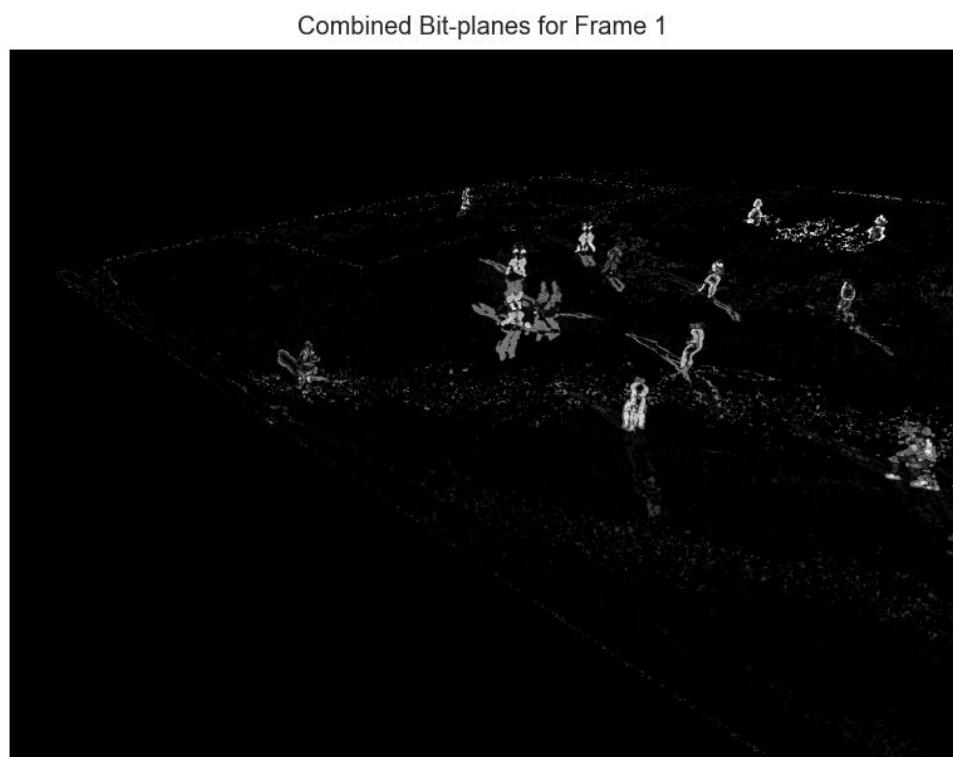


بخش b) در این بخش تصاویر فریم های مختلف از یک زاویه را به صورت bit-plane به bit-plane با هم XOR میکنیم. نتیجه این عمل را در ۳ تصویری که در ادامه آمده است مشاهده می کنید.





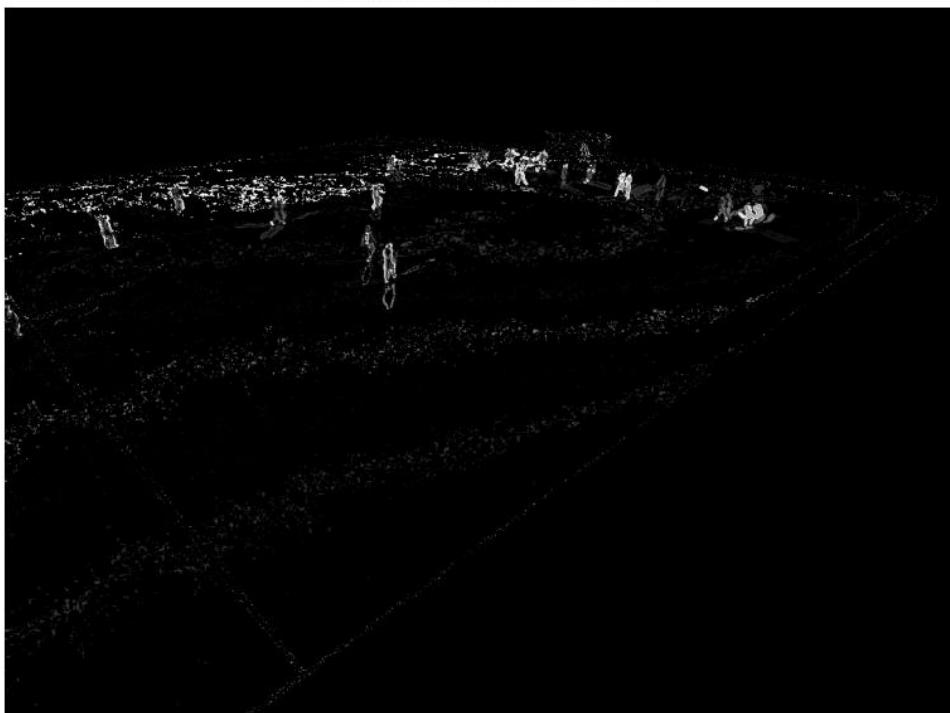
بخش ۵) در ادامه ۴ bit-plane پرارزش را با یکدیگر جمع میکنیم و نتیجه حاصل را برای هر سه زاویه در سه تصویر زیر مشاهده میکنید.



Combined Bit-planes for Frame 2

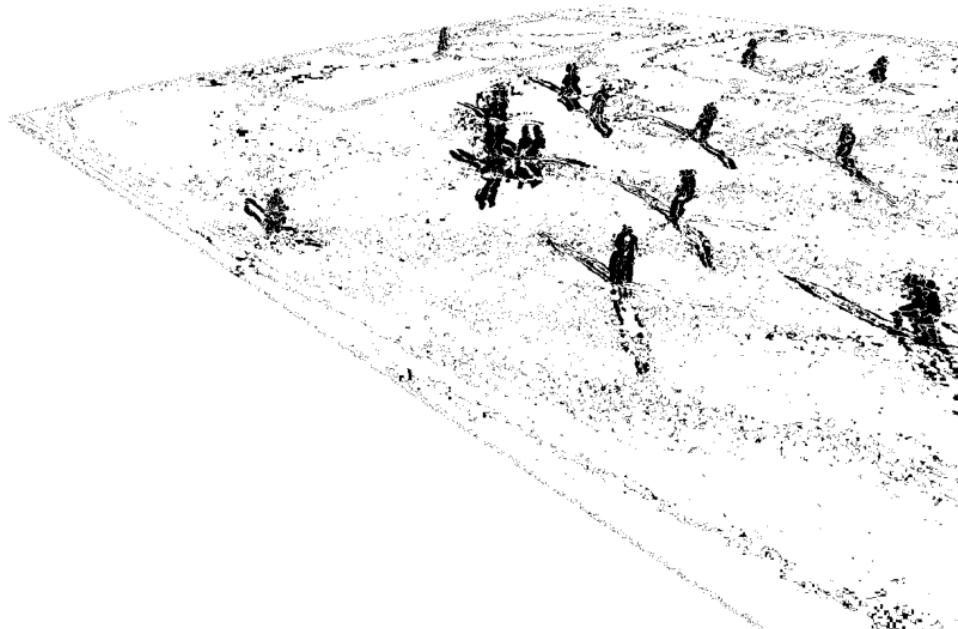


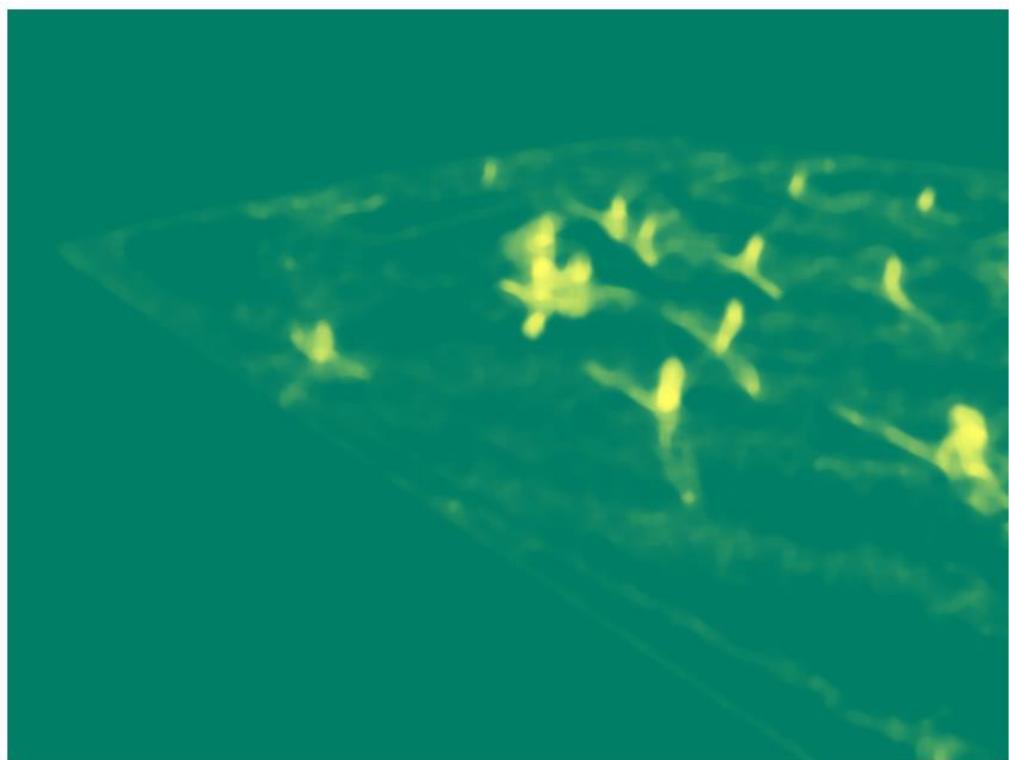
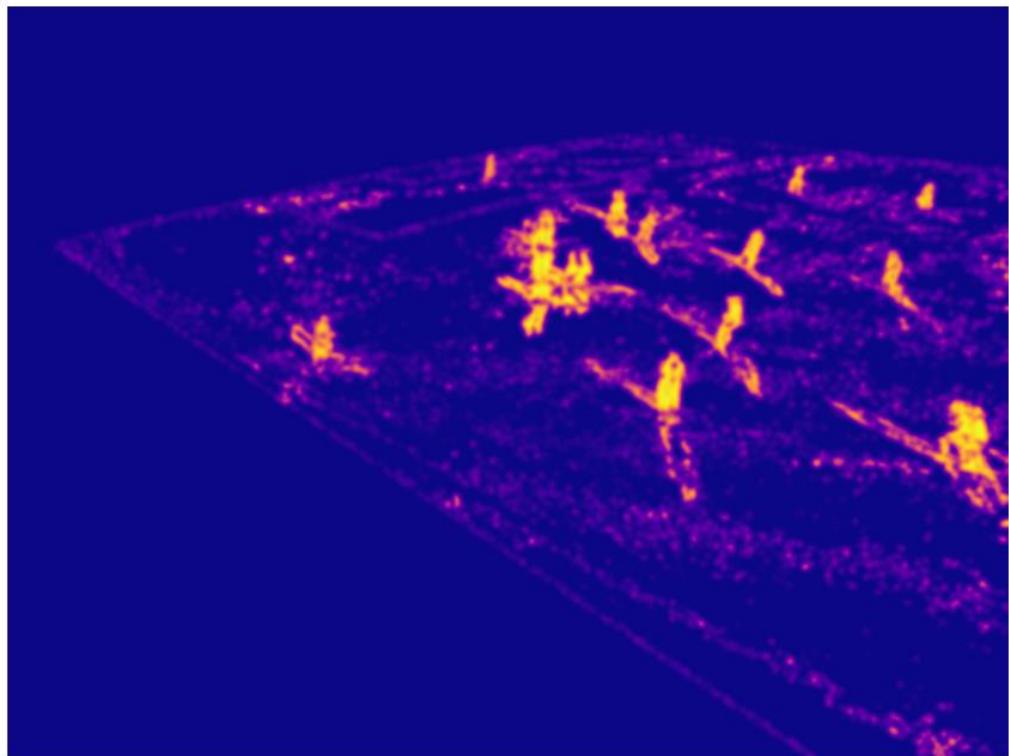
Combined Bit-planes for Frame 3

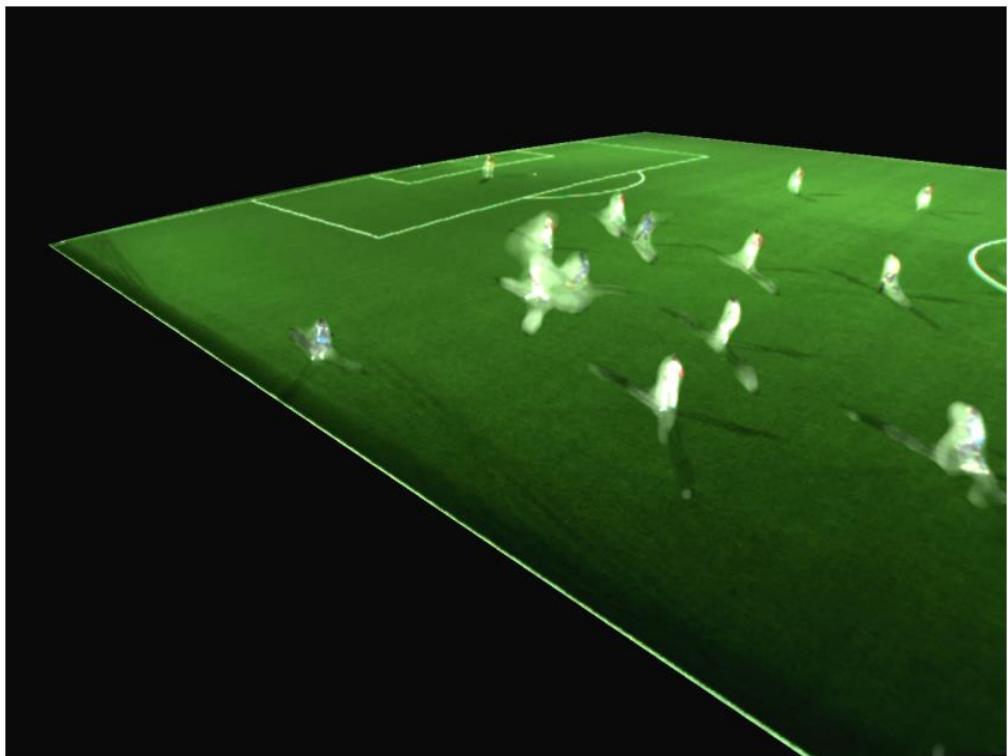


بخش d) برای این مرحله به مشکلات زیادی بخورد کردیم. از جمله مشکلات اصلی سایه بازیکنان بود که خیلی بزرگ و واضح در تصویر مشاهده می شد و از آنجایی که همراه بازیکنان حرکت می کرد، در تشخیص حرکت به عنوان نتیجه مثبت اعلام می شد. در ادامه از هر زاویه دوربین، ۵ تصویر مشاهده می کنید. تصویر اول نتیجه اعمال آستانه گیری است. تصویر دوم نتیجه اعمال Gaussian Blur است و تصویر بعدی نتیجه اعمال فیلتر Median Blur است. تصویر چهارم نتیجه اعمال فیلترهای Erode و Dilate استو در نهایت تصویر پنجم نتیجه جمع تصویر چهارم با تصویر اصلی مربوطه است. در تصویر پنجم می توان دید که موقعیت بازیکنان در زمین مشخص شده است.

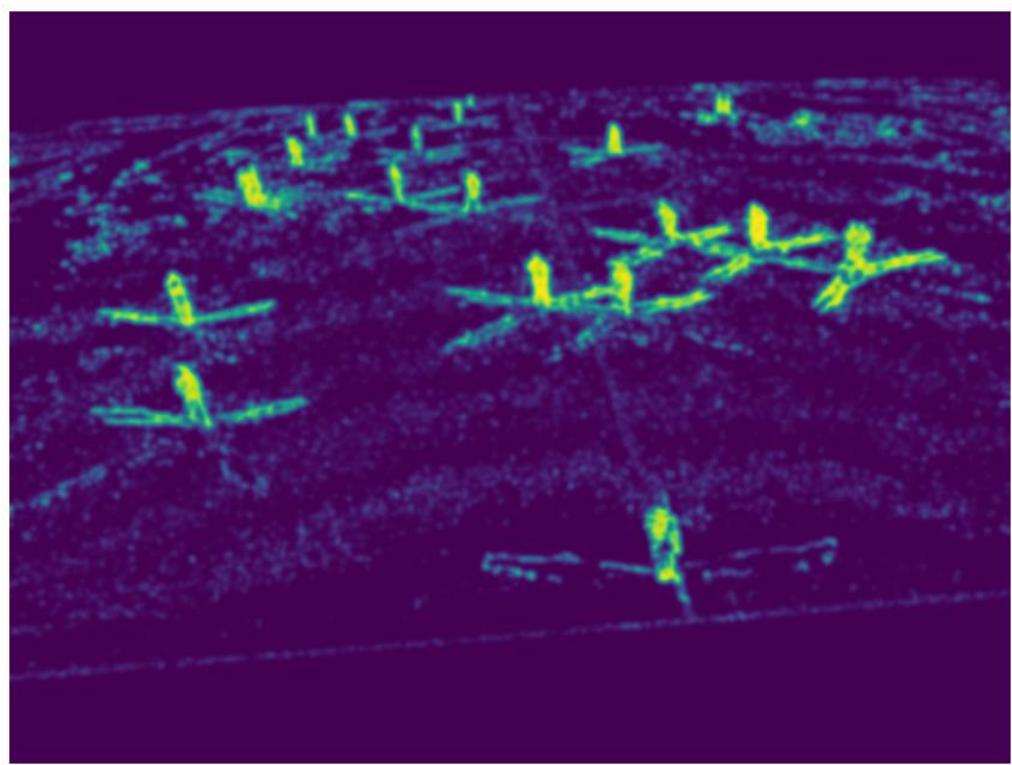
تصویر اول:

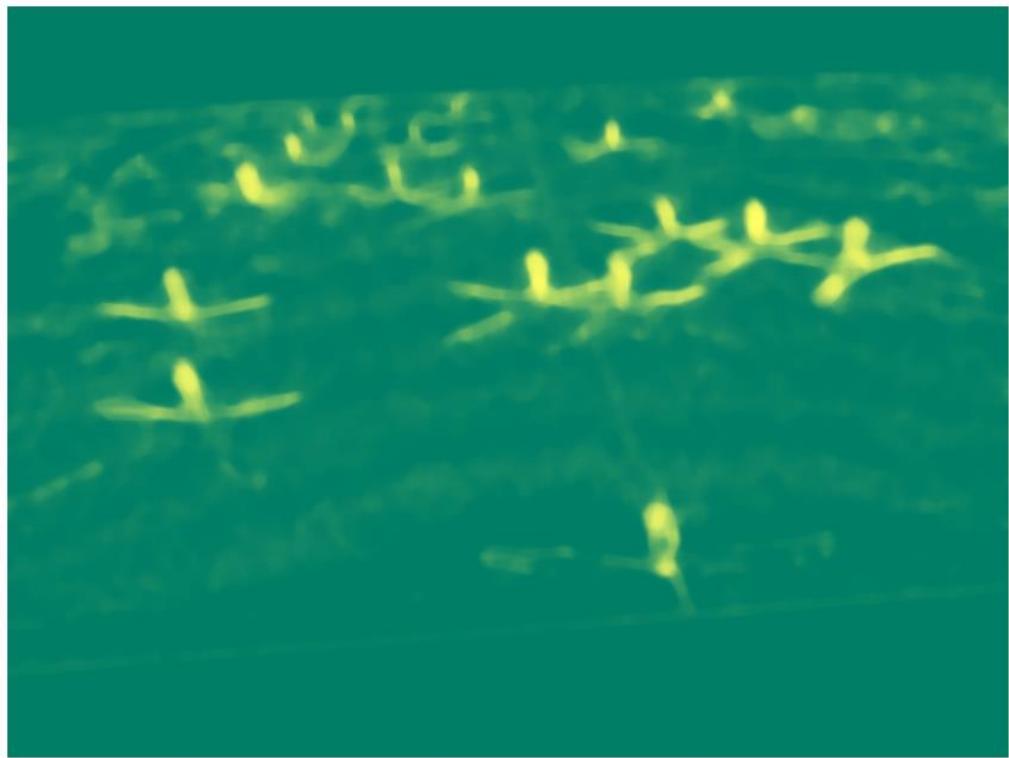


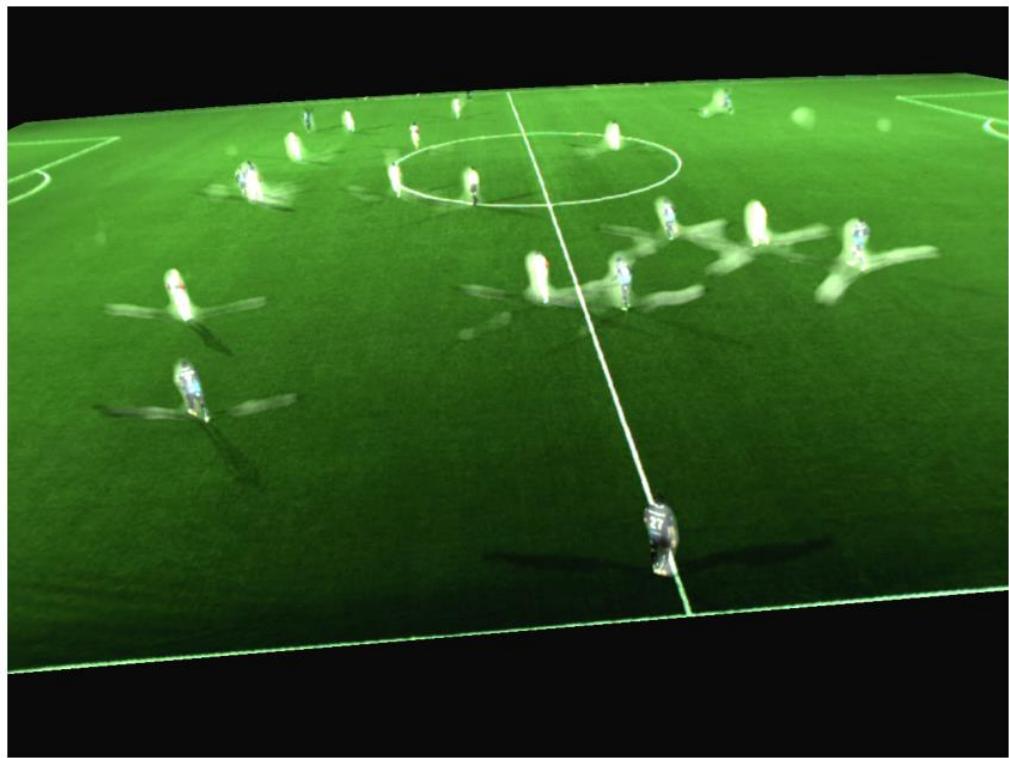




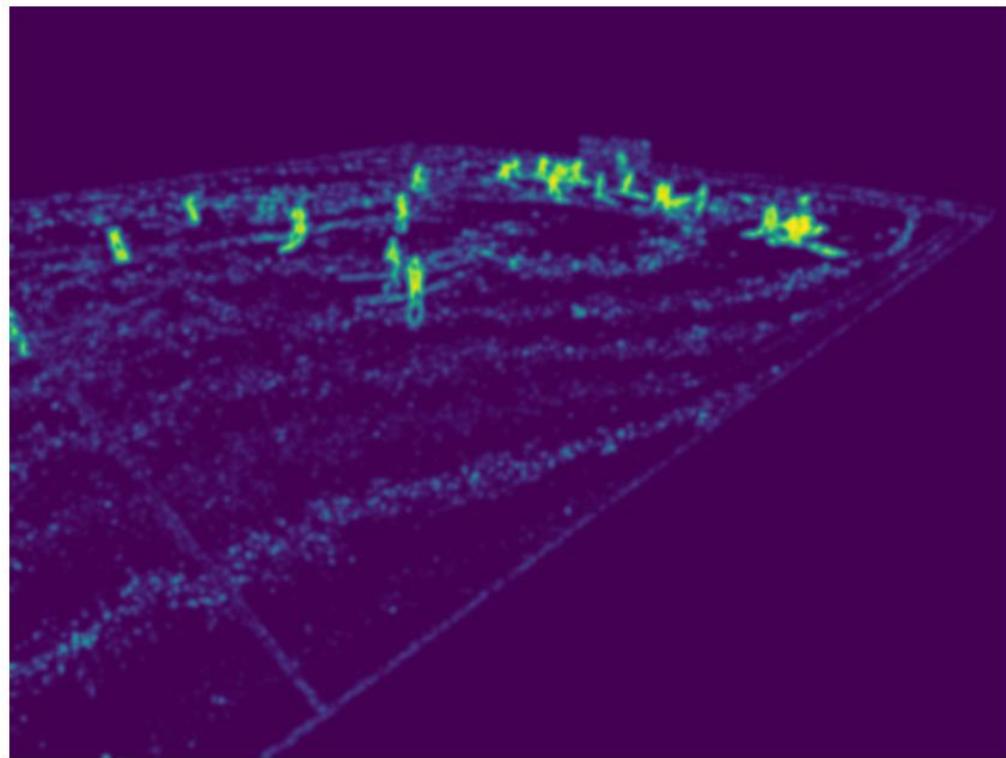
تصویر دوم:

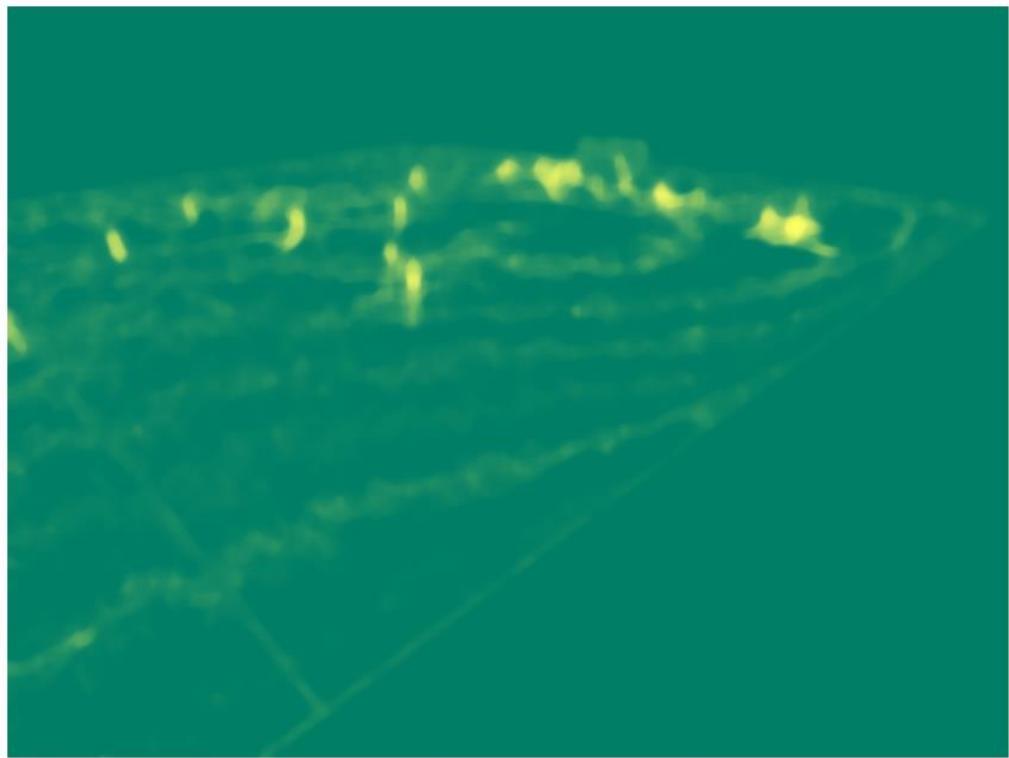


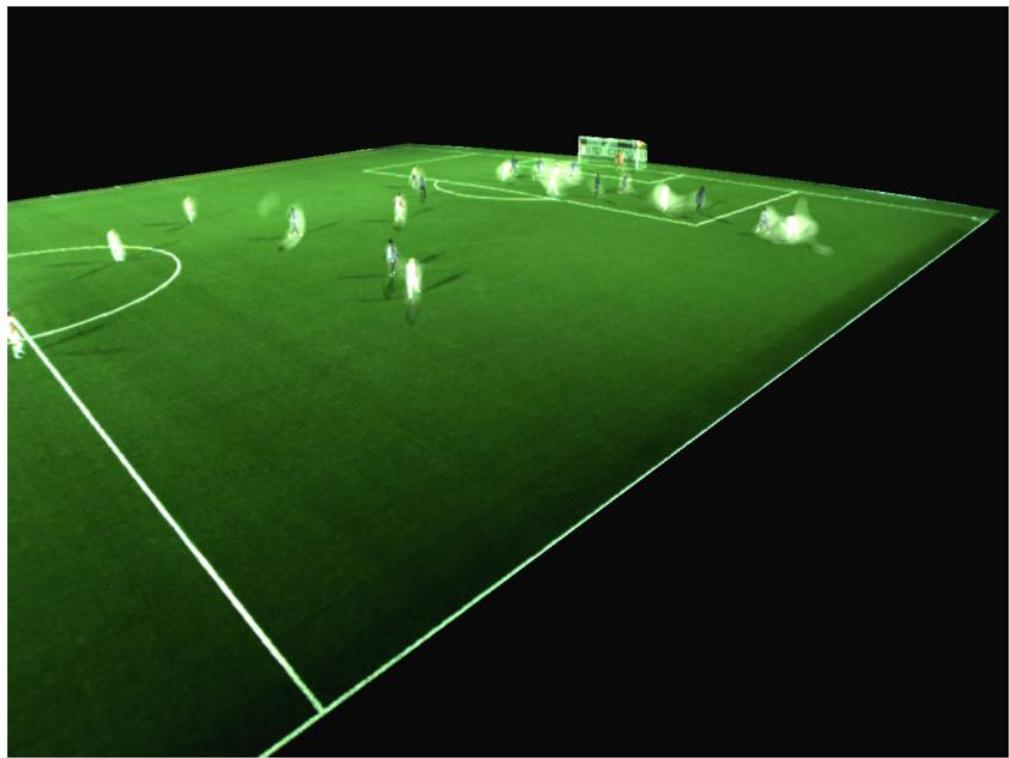




تصویر سوم:

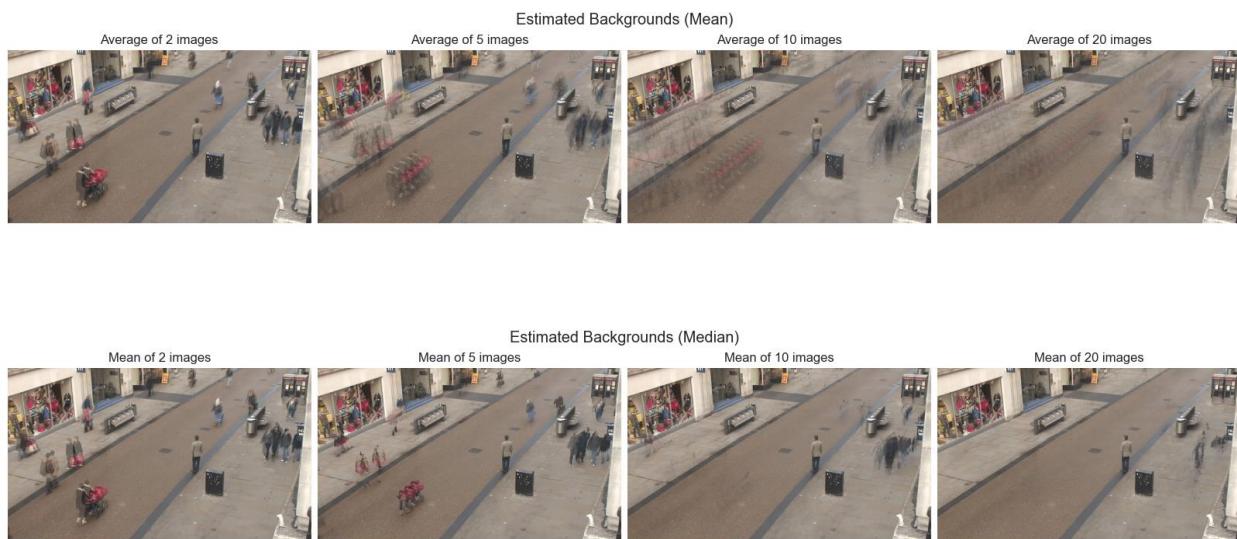




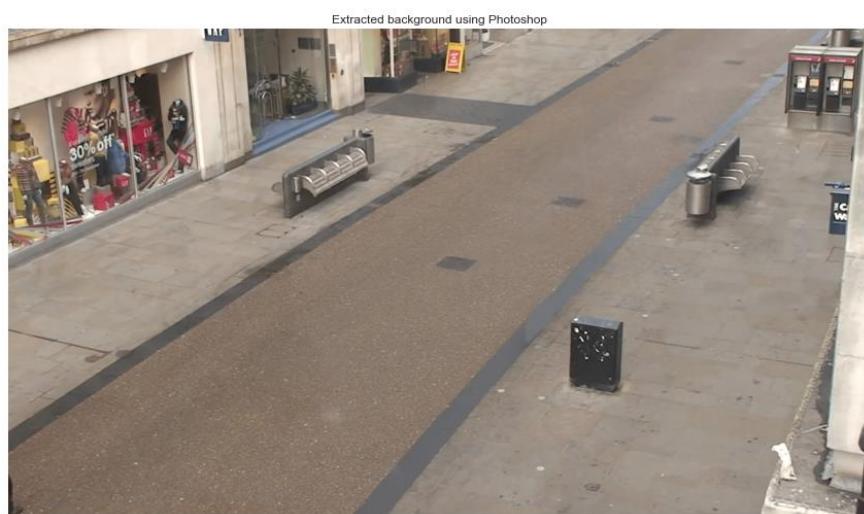


سوال (۳)

بخش a) در این بخش، تصاویر را به تعداد مشخص شده توسط سوال با هم میانگین می‌گیریم. به این دلیل که افراد در بین تصاویر مختلف حرکت می‌کنند اما پس زمینه همواره ثابت است، پس از میانگیری توقع می‌رود که اثر وجود افراد در تصویر کم‌رنگ‌تر شود. پس از انجام آزمایش‌ها متوجه شدیم که عمل میانه گیری بین تصاویر مناسب‌تر از میانگین گیری است اگر تعداد تصاویر بیشتری داشتیم احتمالاً نتیجه میانه و میانگین سیار شبیه می‌شد اما با این تعداد تصویر، میانه به دلیل اینکه احتمال وقوع پس زمینه در یک پیکسل در طول فریم‌های متوالی بیشتر از وقوع انسان بود، بهتر عمل کرد.



در ادامه آزمایش‌ها تصمیم گرفتیم با کمک نرم‌افزار فتوشاپ یک پس زمینه خالی با دقت بالاتر تولید کنیم. با توجه به اینکه دوربینی که این تصاویر را ثبت کرده ثابت است، تولید یک تصویر پس زمینه به صورت دستی برای ساعات مختلف شبانه روز می‌تواند تا حد زیادی به نتایج کمک کند. مخصوصاً اگر این کوچه یک کوچه بسیار شلوغ باشد و میانگین گیری، نتواند پس زمینه را به خوبی جدا کند.



بخش (b) در این بخش تصاویر تست را از پس زمینه‌های تخمین زده شده کم می‌کنیم و نتیجه را مشاهده می‌کنیم.

تصاویر میانگین گرفته شده با ۲ فریم

Subtraction



Subtraction



تصاویر میانگین گرفته شده با ۵ فریم

Subtraction



Subtraction



تصاویر میانگین گرفته شده با ۱۵ فریم

Subtraction



Subtraction



تصاویر میانگین گرفته شده با ۲۰ فریم

Subtraction



Subtraction



تصاویر میانه گرفته شده با ۲ فریم

Subtraction



Subtraction



تصاویر میانه گرفته شده با ۵ فریم

Subtraction



Subtraction



تصاویر میانه گرفته شده با ۱۵ فریم

Subtraction



Subtraction



تصاویر میانه گرفته شده با ۲۰ فریم

Subtraction



Subtraction



و در نهایت تفاضل دو تصویر تست را با پس زمینه تولید شده به کمک فتوشاپ بررسی میکنیم.

Subtraction



Subtraction



بخش c) طبق خواسته سوال در این بخش با Threshold گیری روی نتایج تفربیق یک ماسک سیاه و سفید از افراد بدست می‌آوریم.

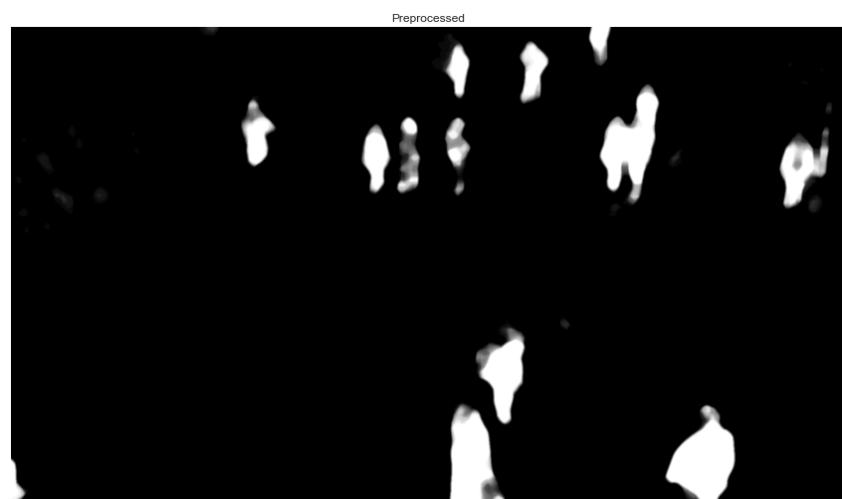


بخش d) در این مرحله با استفاده از مجموعه ای از فیلترها سعی میکنیم افراد را از پس زمینه به طور واضح جدا کنیم. از آنجایی که این مرحله نیاز به تغییرات دستی زیادی دارد و نمیتوان یک سری عملیات را بر روی تصاویر مختلف اعمال کرد و توقع نتیجه یکسان داشت، این مرحله را فقط بر روی بهترین نتیجه مرحله b انجام میدهیم. آوردن نتایج این مرحله با پس زمینه های دیگر مفهومی نخواهد داشت و طبق تست های انجام شده همه آنها نتیجه بدتری با این عملیات می دهند (در صورت نیاز برای مشاهده تصاویر دیگر به نو特 بوک مراجعه کنید).

ماسک مرحله قبل به تصویر تست ۱ اعمال شده



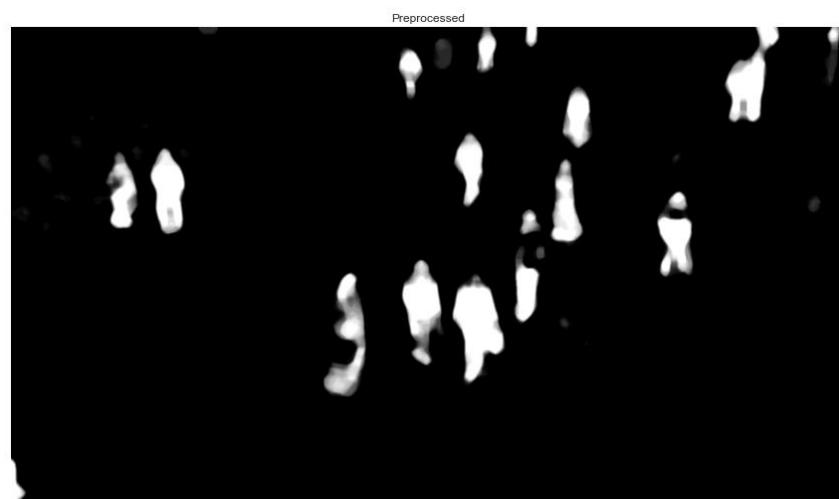
ماسک پردازش شده تصویر تست ۱ برای مرحله بعد



ماسک مرحله قبل به تصویر تست ۲ اعمال شده



ماسک پردازش شده تصویر تست ۲ برای مرحله بعد



بخش ۵) هم‌زمان با انجام دادن مراحل برای میانگین در بخش‌های قبل، نتایج مربوط به میانه نیز ارائه شد.

بخش ۴) برای تشخیص و شمارش تعداد افراد در تصویر، ابتدا روش Template matching آزمایش شد. تصویری که در زیر میبینید به عنوان نماینده یک انسان در تصویر پردازش شده است. الگوریتم Template matching این Template matching را بر روی تمام پیکسل های تصویر قرار می دهد و به هر نقطه یک امتیاز بر اساس میزان شباهت با Template نسبت میدهد.



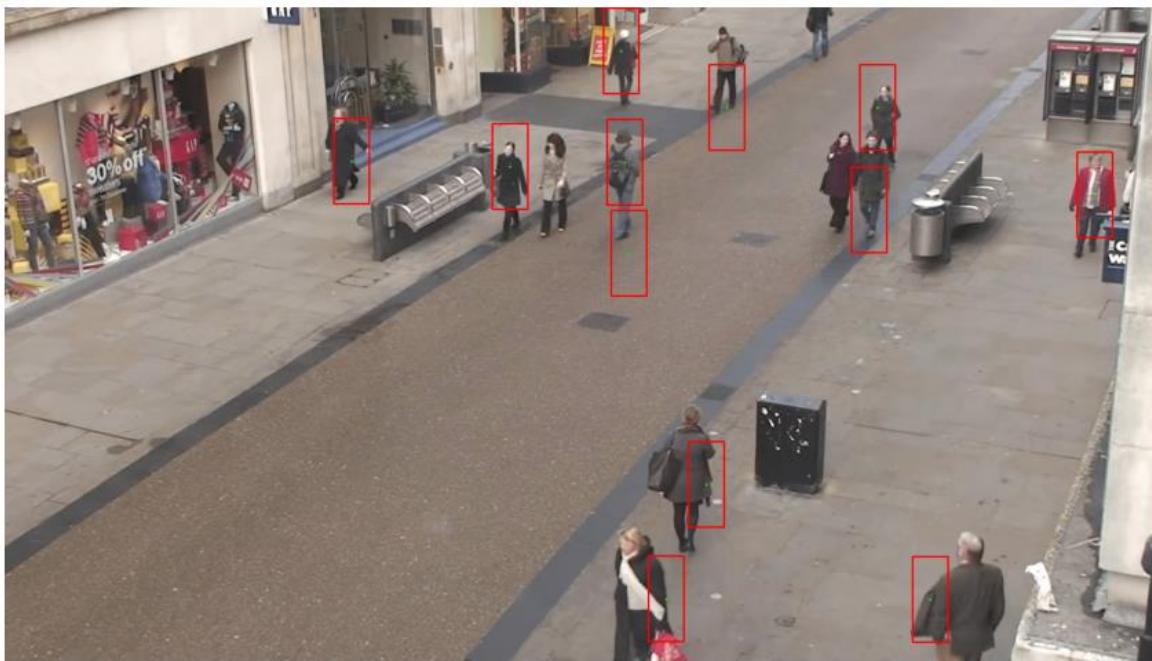
Preprocessed



اما این روش نتوانست افرادی که به صورت ناقص در تصویر وجود دارند را به خوبی تشخیص دهد. همچنین، بعضی افراد را بیش از یک بار تشخیص می داد. برای مقابله با این مشکل مینیمم فاصله ای برای انسان های تشخیص داده شده تعریف کردیم اما با این کار تعدادی از افراد که بسیار به یکدیگر نزدیک بودند تشخیص داده نمی شدند.

در تصویر زیر می‌توانید نمونه‌ای از نتیجه این روش را ببینید.

نتیجه Bounding box هایی که از یک حدی از هم دورتر بوده اند



نقاط مرکز Bounding Box ها

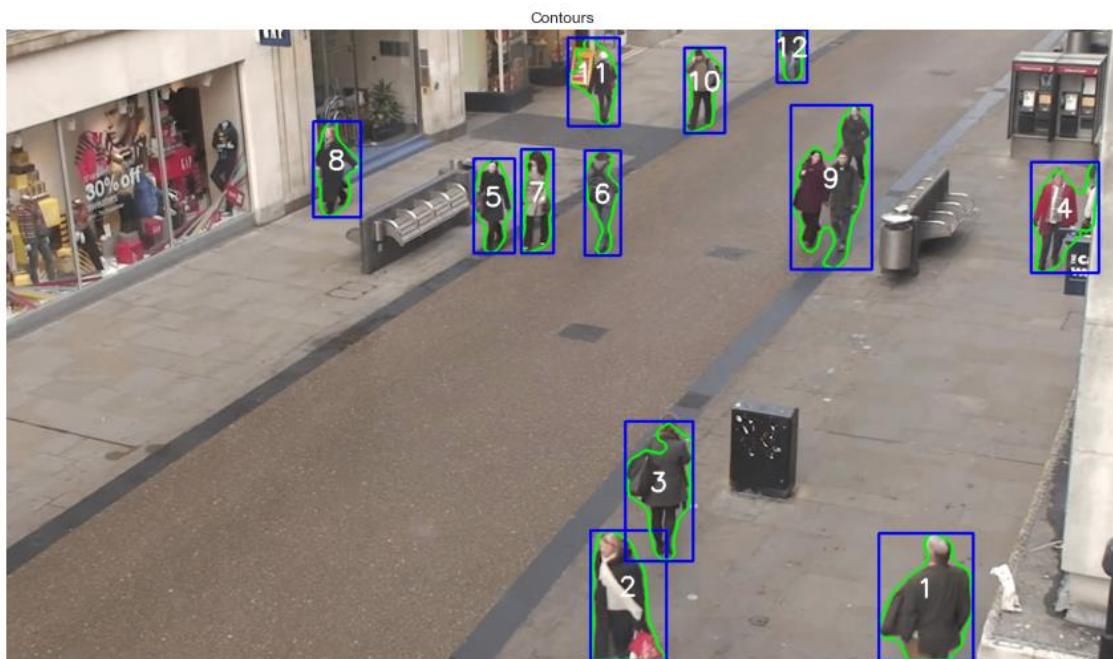


نتیجه Matching (امتیاز پیکسل ها)

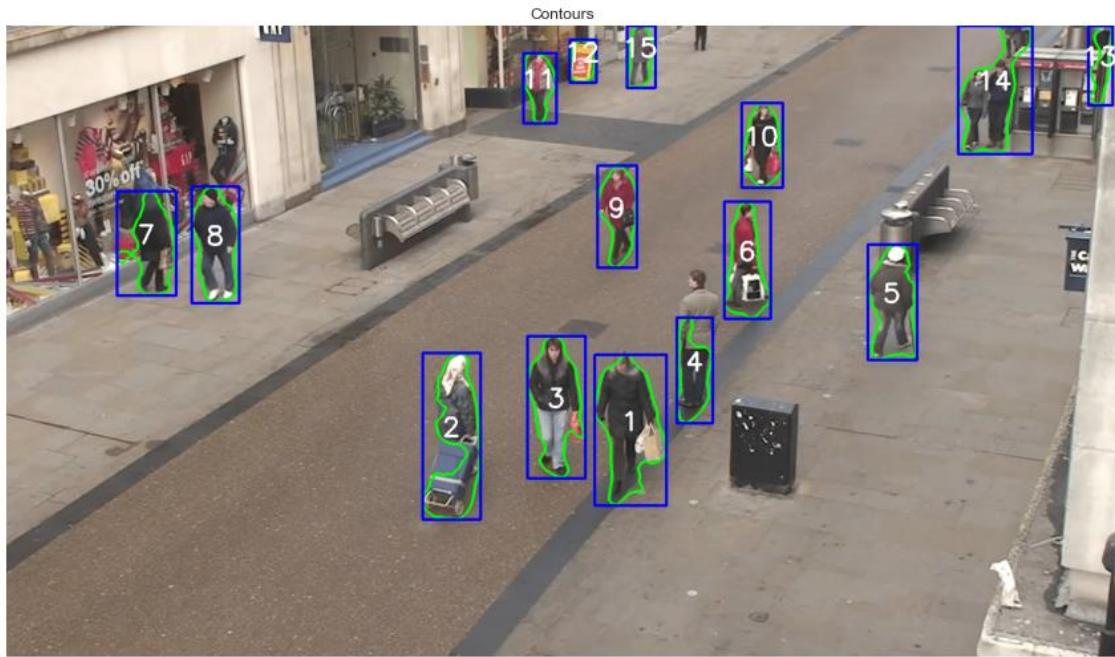


روش دومی که آزمایش شد محاسبه contour های تصویر سیاه و سفید بود. این روش نیز مشکلات جزئی ای دارد اما به طور کلی نتیجه قابل قبول تری ارائه می دهد. پس از محاسبه کنتورها، آنها یک سایز مشخصی بزرگتر بودند به عنوان انسان قبول شدند و یک Bounding Box دور محیط آنها رسم شد. تعداد این کنتورهای قبول شده، تعداد افراد موجود در تصویر است. لازم به ذکر است که افرادی که بسیار به یکدیگر نزدیک بودند به عنوان یک کنتور و در نتیجه به عنوان یک انسان تشخیص داده شدند.

نتیجه کنتور بر روی تصویر تست ۱



نتیجه کنتور بر روی تصویر تست ۲



همچنین علاوه بر تصاویر تست، الگوریتم را بر روی تصاویری که برای تولید پس زمینه استفاده شد نیز اجرا کردیم که به صورت یک ویدیو در فایل های پیوست تمرین قابل مشاهده می باشد.

سوال (۴)

بخش (a) تابع ذکر شده پیاده سازی شد و همچنین با تابع cv2.bilateralFilter اپن سیوی مقایسه شد و نتایج تقریباً یکسانی بدست آمد.

```
1  @jit
2  def bilateral_filter_ghd(image, d, sigma_d, sigma_r):
3      sigma_d = 2*sigma_d*sigma_d
4      sigma_r = 2*sigma_r*sigma_r
5
6      image = image.astype(np.float32)
7      result = np.zeros(image.shape, dtype=np.float32)
8      for i in range(image.shape[0]):
9          for j in range(image.shape[1]):
10             numerator = np.array([0, 0, 0], dtype=np.float32)
11             denominator = np.array([0, 0, 0], dtype=np.float32)
12             for k in range(i-d//2, i+d//2):
13                 for l in range(j-d//2, j+d//2):
14                     if 0 <= k < image.shape[0] and 0 <= l < image.shape[1]:
15                         w_ijkl = w(image, i, j, k, l, sigma_d, sigma_r)
16                         numerator += image[k, l] * w_ijkl
17                         denominator += w_ijkl
18             result[i, j] = numerator / denominator
19
20     return result.astype(np.uint8)

1  @jit
2  def w(image, i, j, k, l, sigma_d, sigma_r):
3      return np.exp(-((i-k)**2 + (j-l)**2)/((sigma_d)**2 - ((image[i][j] - image[k][l])**2)/(sigma_r**2)))
```

بخش (b)





بخش c) در این بخش ابتدا تصاویر Blur شده را تبدیل به Grayscale میکنیم.



سپس تمام مقادیر را بر ۴ تقسیم صحیح میکنیم.



و در نهایت به هر کدام از رنگ‌های حاصل یک رنگ از قبیل مشخص شده نسبت میدهیم. در تصویر زیر به عنوان مثال رنگ‌های آبی و قرمز و زرد و سبز به چهار سطح خاکستری تصویر قبل نسبت داده شده اند.



که پس از انتخاب رنگ‌های مناسب تر نتیجه زیر حاصل می‌شود.



بخش d) در این بخش میخواهیم پوسترهايی مانند پوستر نمونه اوباما برای هر سه شخص (اوباما، زلینسکي و پوتين) بسازیم. برای این کار مراحل توضیح داده شده را برای هر شخص تکرار میکنیم. علاوه بر مراحل قبل، در این بخش با کمک ماسک موجود، پس زمینه را نیز تغییر میدهیم و همچنانی یک متن به قسمت زیر پوستر میافزاییم.







بخش (e) مقدار Threshold ای که در نظر میگیریم مشخص میکند اندازه هر ناحیه رنگی چقدر باشد. به عنوان مثال در پوستر زلینسکی، یک هایلایت کوچک بر روی چانه شخص قرار دارد، اگر مقادیر Threshold را متفاوت در نظر میگرفتیم ممکن بود این ناحیه بزرگتر شود یا به کل از بین برود. همچنین ما در کد خود برای تغییر اندازه این نواحی یک مرحله Gamma Correction در نظر گرفته ایم که قبل از Posterization رخ میدهد و اندازه نواحی را تغییر میدهد. به تصاویر زیر دقت کنید. مقدار گاما در این تصاویر به ترتیب از راست به چپ برابر با $1/2 - 0/85 - 0/50$ میباشد.



سوال (۵)

قبل از شروع پاسخ دهی به سوال های مختلف لازم بود توابع مورد نظر را پیاده سازی کنیم و از صحت عملکرد آنها اطمینان حاصل کنیم. بنابراین ابتدا تعدادی آزمایش بر روی تصاویر ساده تر و مطمئن تر انجام شد.

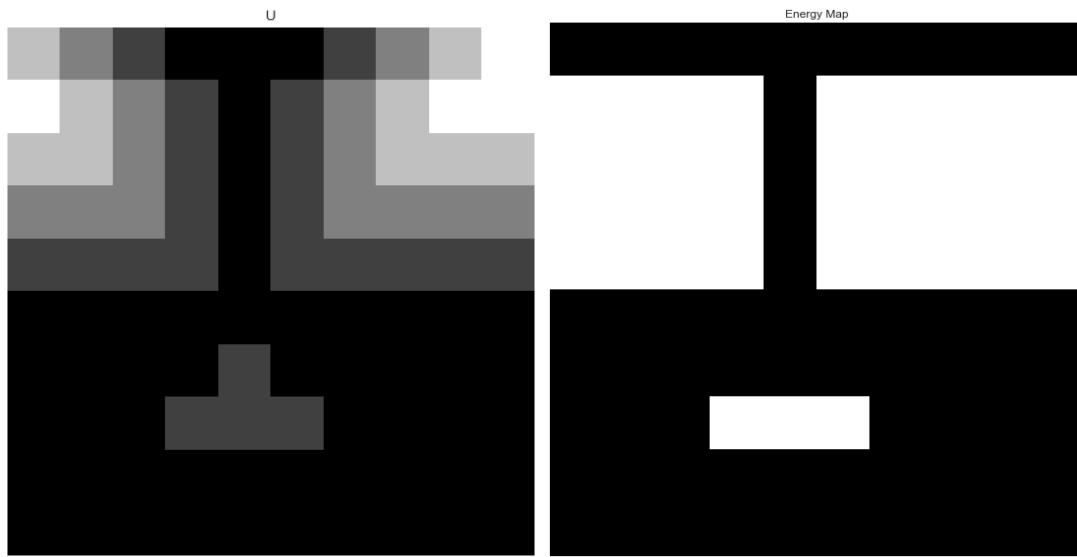
تولید **Energy Map**: برای اینکه مطمئن شویم نقشه انرژی به خوبی تولید می شود، از تصویر Broadway Tower که در ویکی پدیای **Seam Carving** نیز آمده استفاده کردیم.



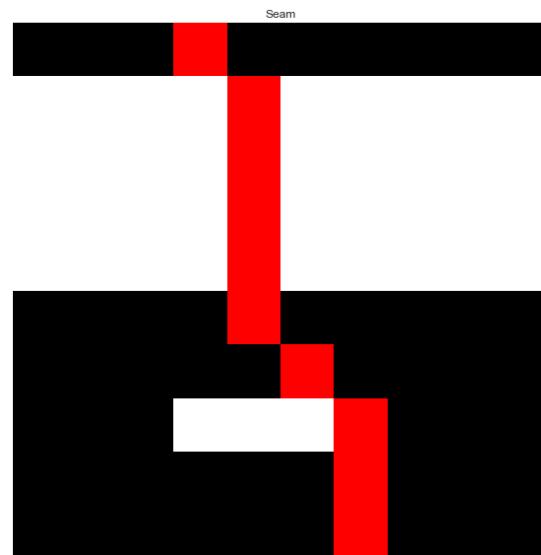
همانطور که میبینید نقشه انرژی به خوبی توانسته لبه ها و قسمت های مهم تصویر را از قسمت های بی اهمیت تر مانند چمن و آسمان جدا کند.



یافتن بهترین Seam: این مرحله بسیار حیاتی و مهم بود به همین دلیل بهترین seam را علاوه بر تصویر ذکر شده در بخش قبل، بر روی یک تصویر با رزولوشن کم که به عنوان تست طراحی شده بود نیز آزمایش کردیم.



تصویر سمت راست نقشه انرژی طراحی شده و تصویر سمت چپ ماتریس میزان هزینه از بالای تصویر تا پایین تصویر است(dp^1). همانطور که در تصویر زیر میبینید، الگوریتم توانست به راحتی کم هزینه ترین Seam را برگزیند.

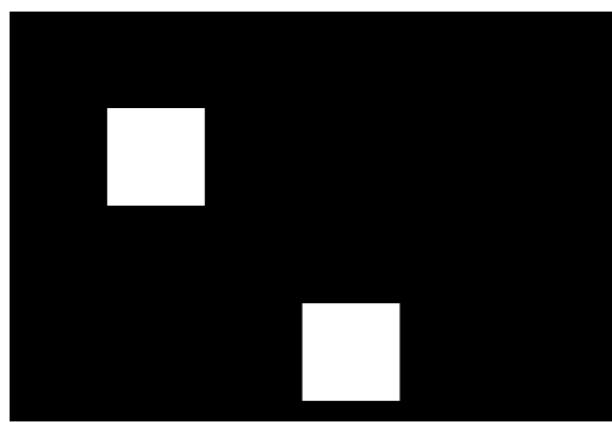
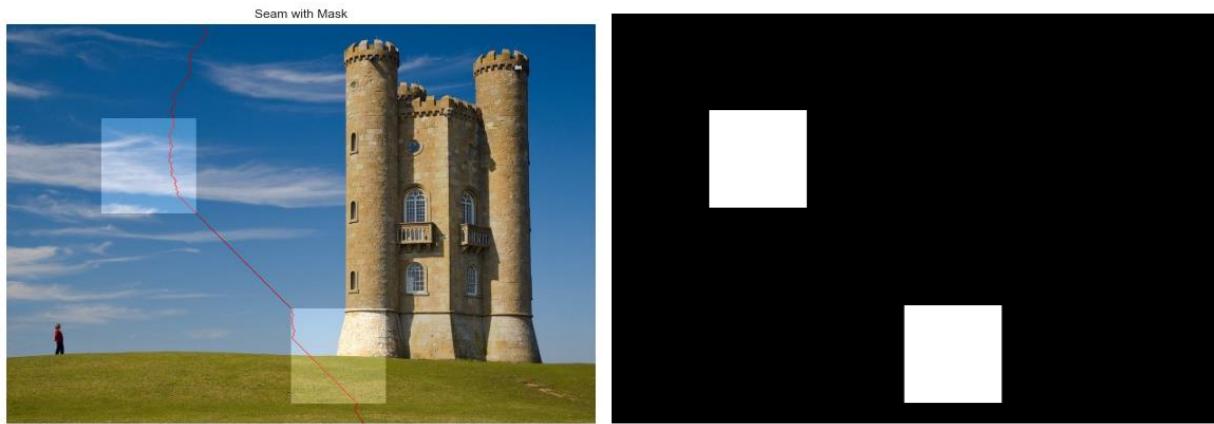


¹ Dynamic programming look up table

علاوه بر این تصویر، الگوریتم را بر روی تصویر برج نیز آزمایش کردیم و به نظر نتیجه قابل قبولی بدست آمد.



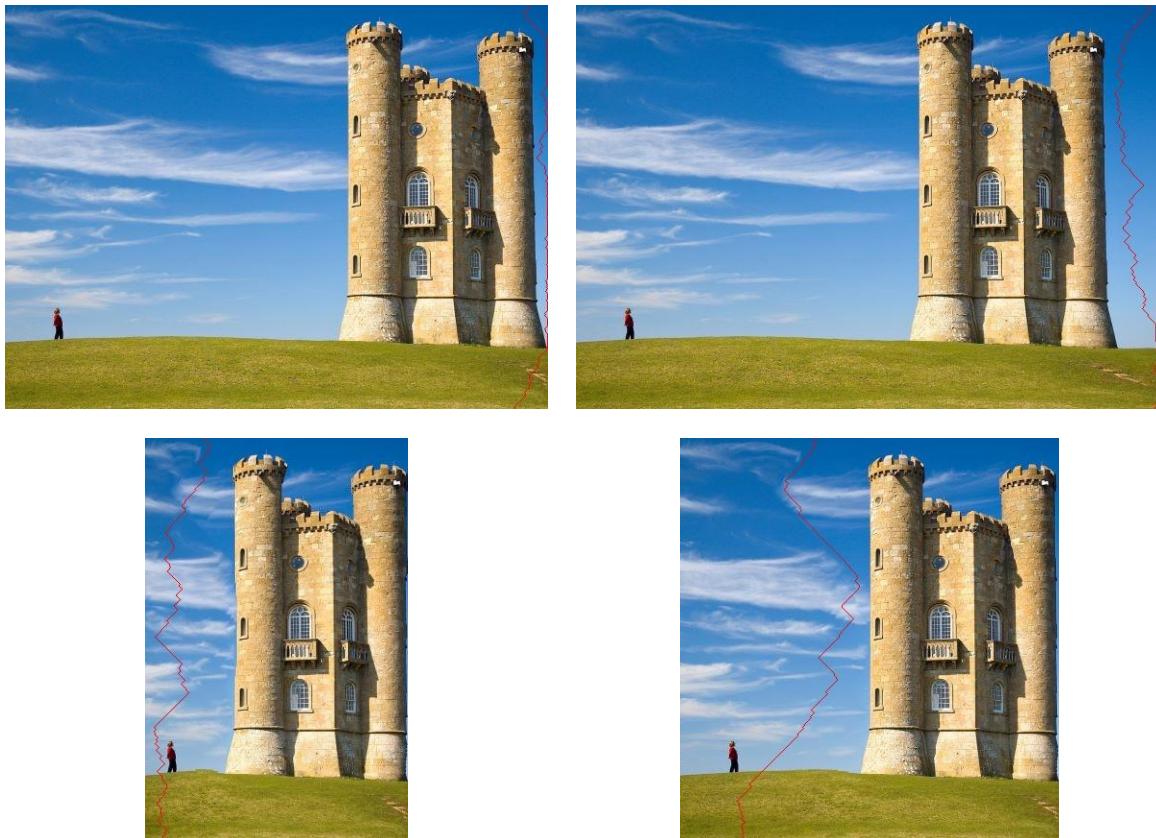
ماسک: در ادامه قابلیت ماسک را بررسی میکنیم به این شکل که برای تصویر برج، ماسک زیر را طراحی میکنیم و نقاط سفید این ماسک را نقاط با انرژی منفی ۱۰۰۰۰ در نظر میگیریم تا به هنگام انتخاب Seam با کمترین انرژی همواره Seam هایی که از ماسک میگذرند انتخاب شوند.



همانطور که مشاهده می‌شود پس از اعمال ماسک به نقشه انرژی، میبینیم که بهترین Seam آن عی می‌شود که از قسمت‌های سفید ماسک عبور کند.

کوچک کردن تصویر: مانند بخش‌های قبلی، این بخش را نیز بر روی تصویر برج آزمایش می‌کنیم. نحوه کار به این صورت است که ابتدا نقشه انرژی را محاسبه می‌کنیم، سپس با کمک برنامه نویسی پویا بهترین Seam را پیدا می‌کنیم (Seam عی که کمترین انرژی را داشته باشد) سپس آن Seam را از تصویر (و در صورت وجود از ماسک) حذف می‌کنیم و بار دیگر نقشه انرژی را محاسبه می‌کنیم. این روند را به تعداد Seam مورد نیاز تکرار می‌کنیم. (انیمیشن کامل کوچک کردن این تصویر در فایل های تمرین موجود است. تعدادی از فریم‌های میانی این انیمیشن را در ادامه می‌بینیم).

۴ فریم از انیمیشن کاهش اندازه تصویر برج در جهت افقی



بزرگ کردن تصویر: اگر بخواهیم برای بزرگ کردن تصویر مانند کوچک کردن تصویر عمل کنیم، اتفاقی که می‌افتد این است که هر بار بهتری سیم کپی می‌شود، سپس از آنجایی که آن سیم هنوز در تصویر موجود است باز هم همان سیم به عنوان بهترین سیم انتخاب می‌شود و این روند تکرار می‌شود. نتیجه نامطلوب زیر، نتیجه این تکنیک است.

Expanded Towers



همانطور که می‌بینید بهترین سیم (در چند صفحه قبل آن سیم را دیدیم) به دفعات کپی می‌شود و نتیجه غیرباورپذیری را ارائه می‌دهد.

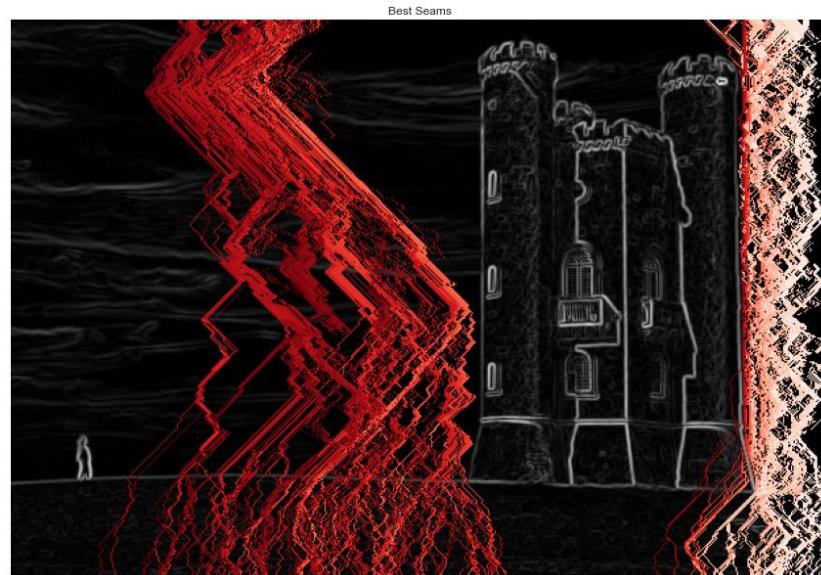
راه دیگر این است که ابتدا ۷ سیم برتر را پیدا کنیم. این کار با پیدا کردن بهترین سیم و حذف آن از یک تصویر موقت و ادامه دادن این روند انجام می‌پذیرد. پس از پیدا شدن بهترین سیم‌ها کافی است آنها را به ترتیب پیداشدن در تصویر کپی کنیم. منتهی یک نکته بسیار مهم اینجا وجود دارد. سیم اول را از تصویر با اندازه اصلی پیدا کرده ایم و بنابراین مشکلی ندارد اما سیم‌های بعدی از تصویری با سایز کوچکتر پیدا شده اند و این باعث می‌شود اندیس ستون هایی که در آن سیم ذخیره شده است اشتباہ باشند. این اتفاق همیشه رخ نمی‌دهد به طور خاص اگر سیمی که در حال حاضر در نظر داریم سمت چپ تمام سیم‌هایی باشد که تا به الان از تصویر اصلی حذف کرده‌ایم، مشکلی برای سیم فعلی رخ نمی‌دهد. به طور مشابه اگر بخشی از سیم فعلی، سمت چپ سیم‌هایی که تا الان از تصویر اصلی حذف کرده ایم باشد، برای آن بخش مشکلی پیش نمی‌آید. اما آن دسته از سیم‌هایی از سیم‌ها) که سمت راست سیم دیگری بوده‌اند، در واقع یک واحد به سمت چپ شیفت خورده اند و باید این شیفت را به نحوی برگردانیم. برگرداندن این شیفت بسیار ساده است و کافی است آن ستون هایی از سیم‌ها که سمت راست سیم قبلی افتاده اند را در هر مرحله ۲ واحد به راست شیفت دهیم (۱ واحد به دلیل اینکه در تصویری پیدا شده‌اند که ۱ ستون کوچکتر بوده و ۱ واحد به دلیل اینکه در مرحله قبل ما سایز تصویر را ۱ ستون افزایش داده ایم (در مرحله بزرگ کردن تصویر هستیم)).

با انجام این تغییرات تصویر زیر حاصل می‌شود (میبینیم که خود برج نیز مقداری خم شده که این به دلیل این است که سایز تصویر را برای بررسی کارکرد این بخش الگوریتم، بیش از حد افزایش داده ایم، با تغییرات کمتر این اتفاق نمی‌افتد).

Expanded Towers Good



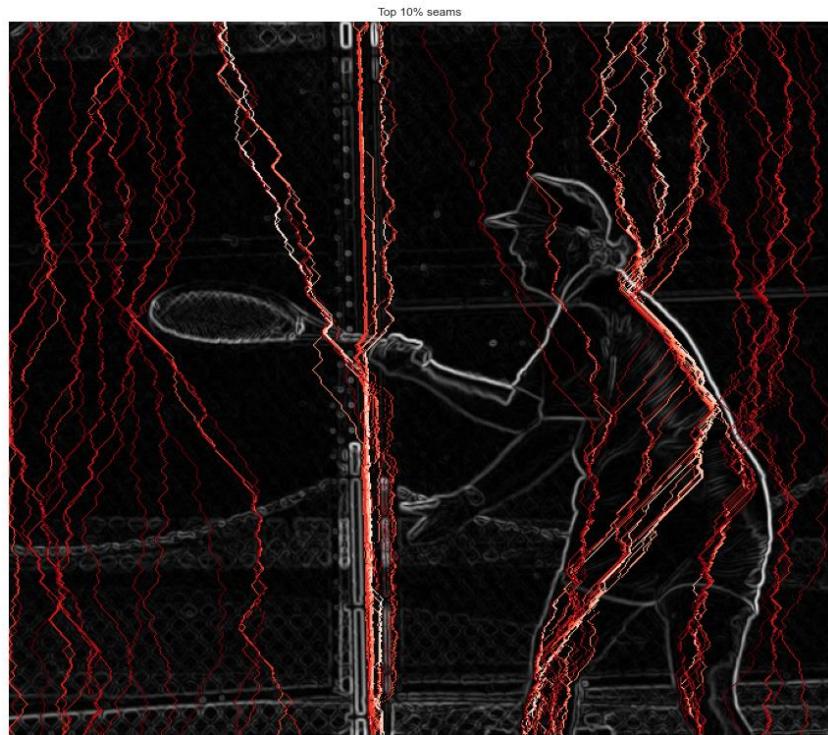
نمایش Seam های بروت: از تکیکی که در بخش قبل ذکر کردیم می‌توان برای نمایش سیم‌های بروت نیز استفاده کرد. به این شکل که ابتدا سیم بروت پیدا می‌شود، سیم از تصویر موقت حذف می‌شود و این روند ادامه پیدا می‌کند. پس از یافتن سیم‌های بروت سیم‌ها یکی یکی بر روی تصویر(یا بر روی نقشه انرژی بسته به نیاز کاربر) رسم می‌شوند. پس از رسم هر سیم، سایر سیم‌ها را به میزان مورد نظر شیفت می‌دهیم تا در مکان متناظر در تصویر با سایز اصلی قرار بگیرند. رنگ سیم‌ها نشان دهنده اولویت آنهاست. سیم‌های سفیدتر انرژی کمتری داشته اند بنابراین زودتر حذف شده اند.

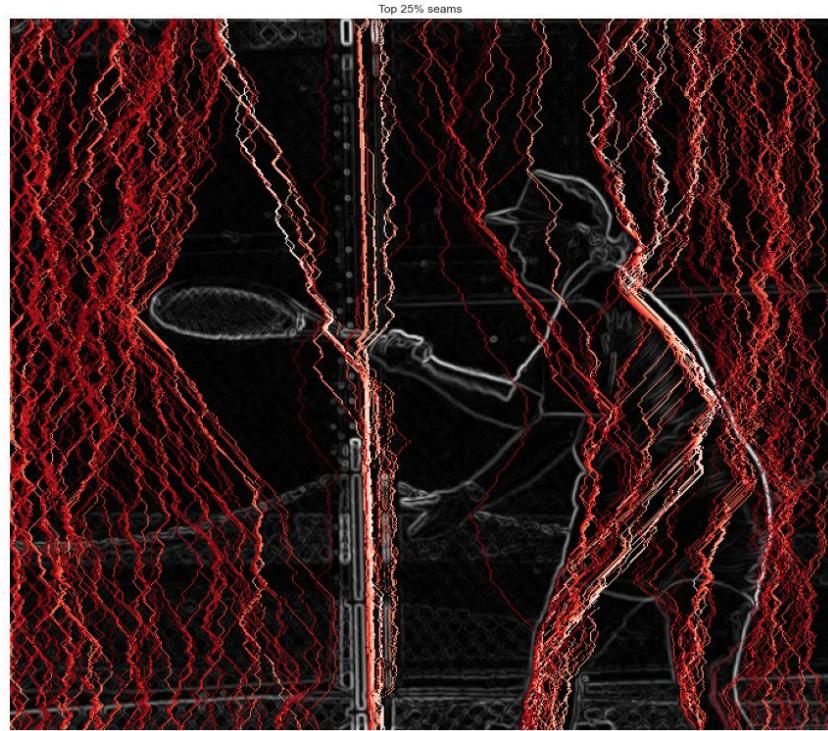


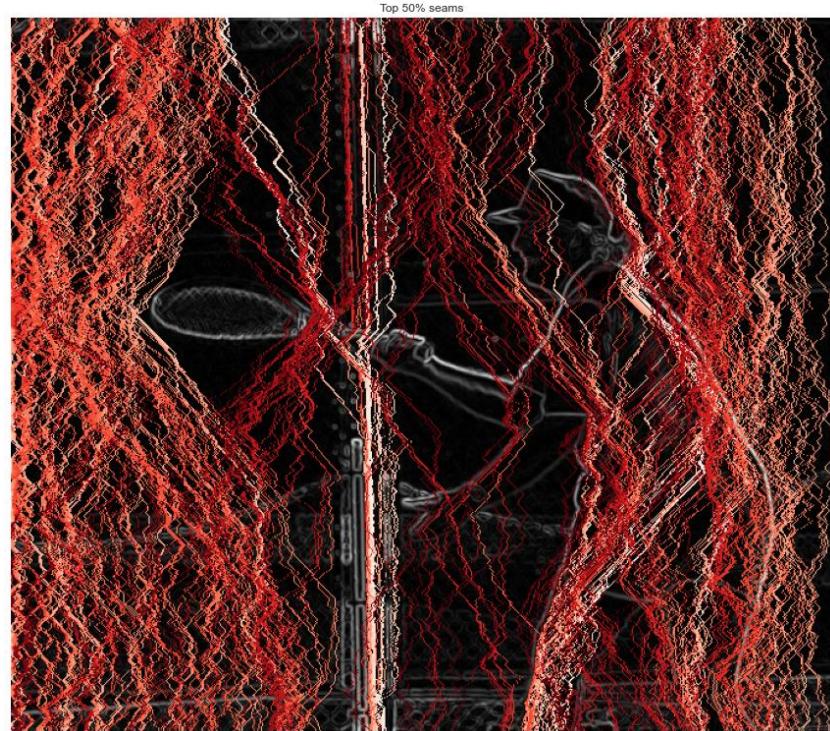
بخش a) کوچک کردن تصویر به اندازه ۱۰٪ و ۲۵٪ در جهت عمودی

توضیحات لازم در مورد الگوریتم و روند انجام کار گفته شد. در بخش‌های باقی‌مانده صرفا نتایج به نمایش گذاشته می‌شود و توضیحات تکمیلی در صورت نیاز داده می‌شود. همچنین اطلاعات مربوط به هر تصویر در بالای هر تصویر آورده شده است.







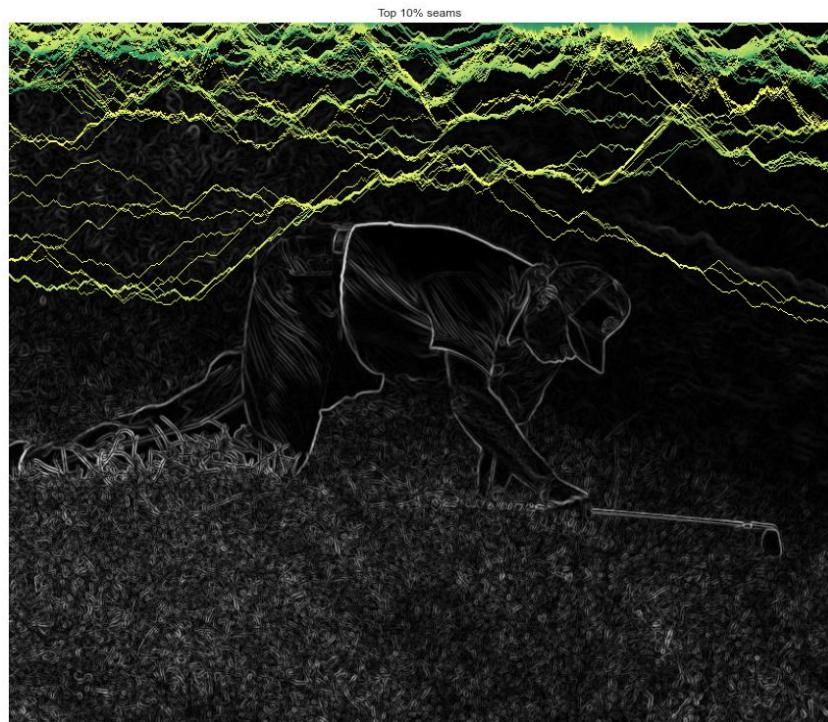


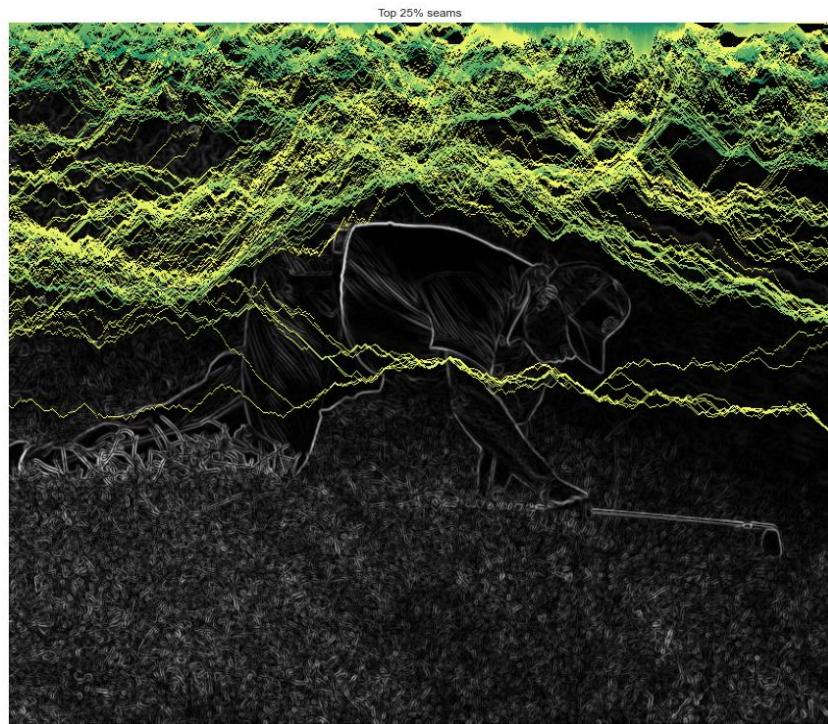
مشکلات این تصویر: اگر به تصویر نقشه انرژی مربوط به این عکس نگاه کنید، متوجه می‌شوید که فنس(Fence) سی که در پس زمینه وجود دارد دارای جزئیات بیشتری نسبت به بدن و مخصوصاً پای دونالد ترامپ است. بنابراین سیمی که از درون بدن دونالد عبور کند، با وجود اینکه باید از یک Edge با وزن بالا بگذرد، اما با این حال انرژی مجموع کمتری خواهد داشت. در ادامه برخی از تصاویر را به صورت مصنوعی تغییر می‌دهیم و عملکرد الگوریتم را بر روی آن‌ها نیز مشاهده می‌کنیم.

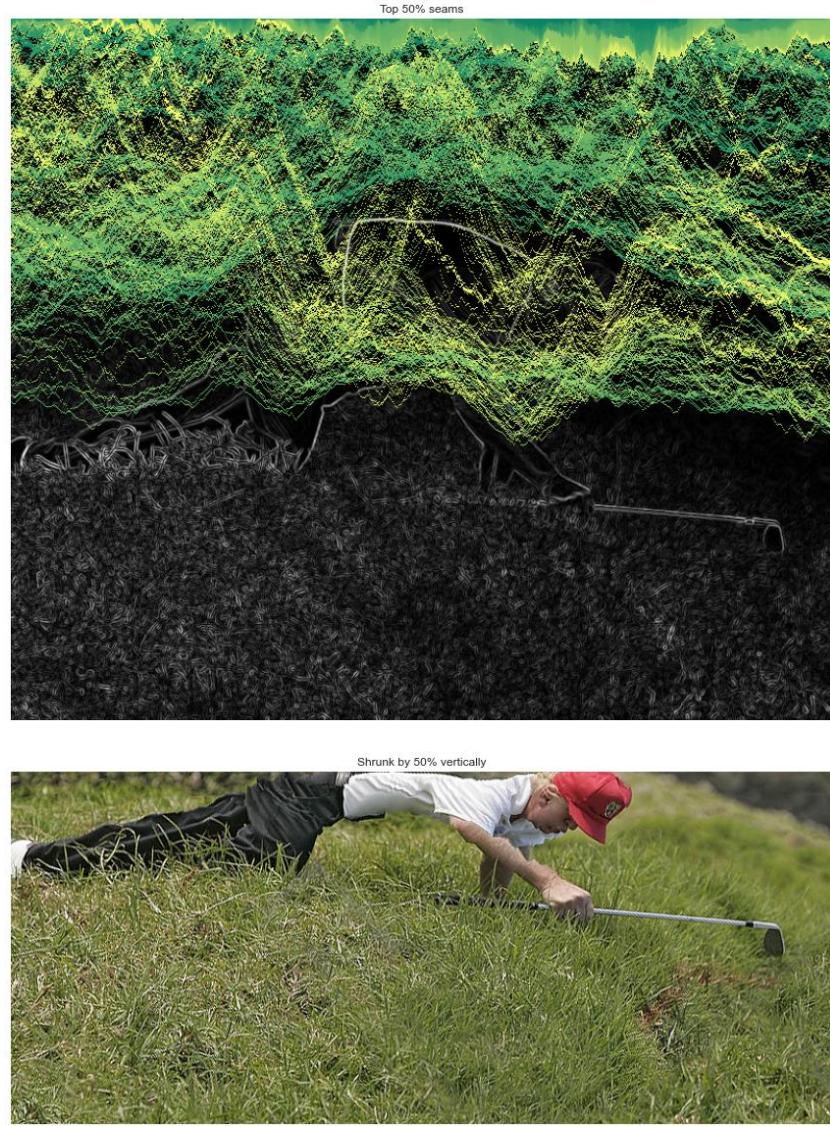
بخش b) کوچک کردن تصویر به اندازه ۱۰٪ و ۲۵٪ و ۵۰٪ در جهت عمودی



توضیحات در رابطه با جهت عمودی: اگر دقت کرده باشید برای سادگی بیشتر، در طول این گزارش تمام سیم‌ها عمودی فرض شدند(عنی تغییر سایز تصویر در جهت افقی رخ می‌دهد). در پیاده سازی نیز این از این سادگی استفاده شد و تمام الگوریتم‌ها برای جهت عمودی سیم‌ها پیاده شدند. در نهایت در توابع اصلی کاهش و افزایش سایز، اگر جهت افقی سیم‌ها ذکر شده باشد، تصاویر را ۹۰ درجه میچرخانیم، سیم‌های عمودی را استخراج میکنیم و پردازش‌های لازم را انجام می‌دهیم در انتها تصویر و تمام تصاویر بدست آمده را ۲۷۰ درجه(یا ۹۰ درجه) میچرخانیم و تصویر در جهت اصلی به دست می‌آید.







مشکلات این تصویر: همانطور که مشاهده میکنید کاهش سایز تا اندازه ۲۵ درصد بدون مشکل انجام شد. و برای انجام این کار، سیم‌هایی از بالای دونالد تراپ (جنگلی که Out of focus است) انتخاب شدند. اما برای حالت ۵۰ درصد، الگوریتم ترجیح داده خود دونالد را حذف کند و بگذارد چمن‌های پیش زمینه دست نخورده باقی بمانند. اگر بار دیگر به تصویر نقشه انرژی مربوط به این عکس نگاه کنید میبینید که چمن‌های پیش زمینه انرژی بسیار بالایی دارند.

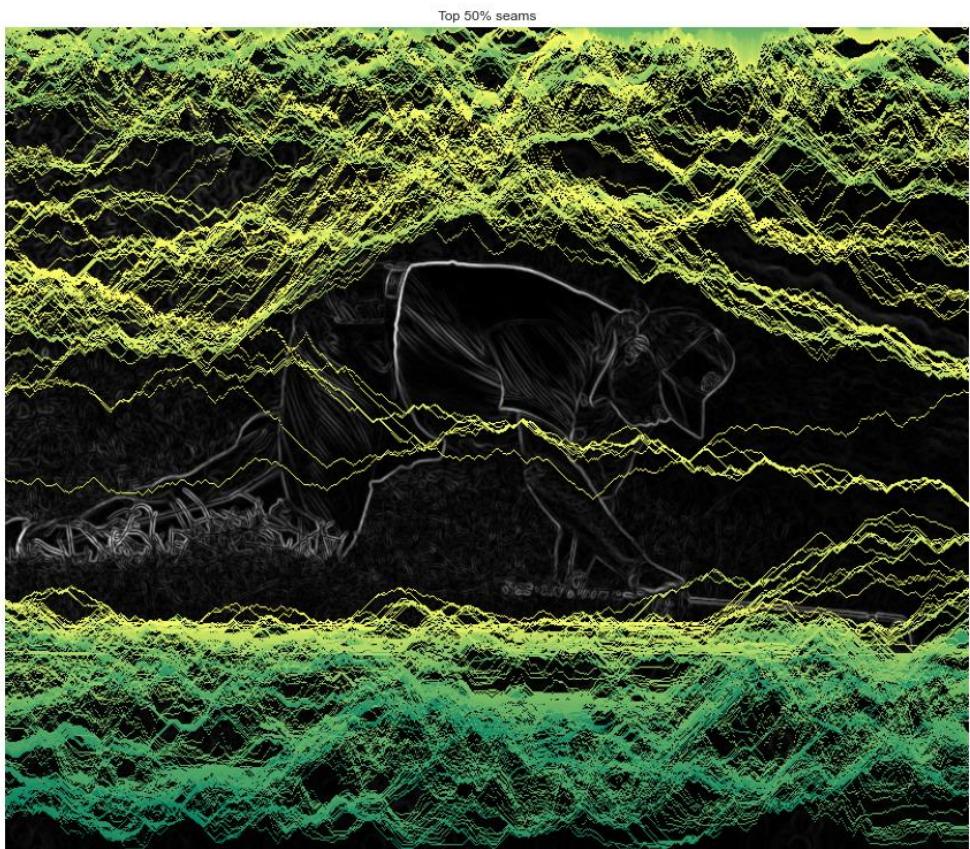
در این بخش کوتاه، میخواهیم انرژی چمن‌های پیش زمینه را کم کنیم و بار دیگر تصویر را به اندازه ۵۰ درصد کاهش اندازه دهیم. امیدواریم در این حالت چمن‌ها قبل از خود دونالد ترامپ حذف شوند. انرژی جدید را مشاهده میکنید که نسبت به انرژی قبلی در ناحیه چمن به طور واضح متفاوت است.

نقشه انرژی جدید:



نقشه انرژی قدیم:





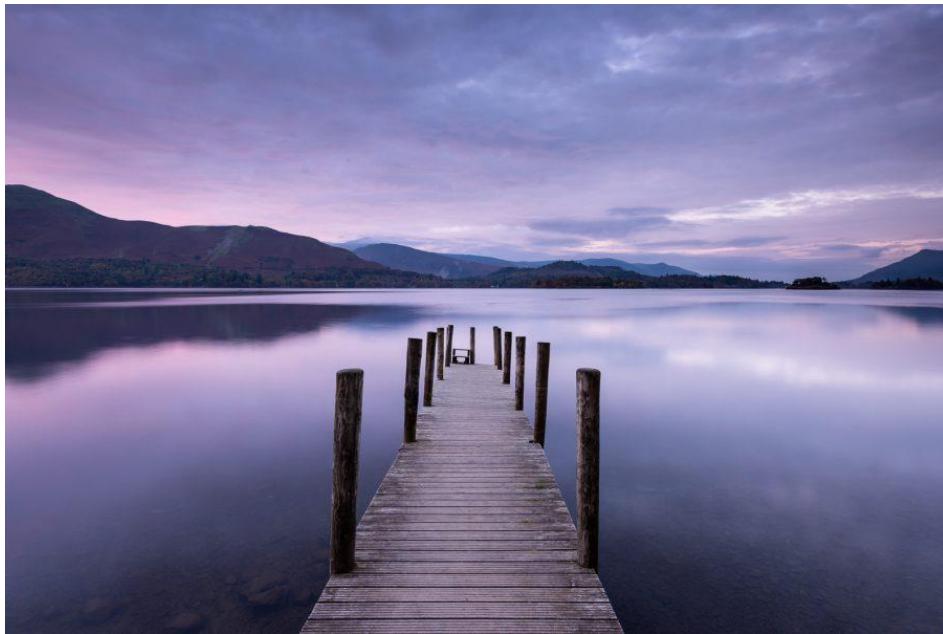
Shrunk by 50% vertically



همانطور که انتظار داشتیم سیم‌های برتر از قسمت بالا و پایین دونالد ترامپ انتخاب شدند و خود سوژه دست نخورده باقی ماند.

یک تکنیکی که در اینجا میتوانستیم استفاده کنیم استفاده از یک ماسک مثبت بود. همانطور که در حال حاضر از ماسک منفی برای حذف کردن اجسام در تصاویر استفاده میکنیم میتوانستیم یک ماسک مثبت داشته باشیم و بگوییم در پیکسل‌های سفید این ماسک انرژی مثبت ۱۰۰۰۰ است. با این کار خود به خود سیم‌ها از نواحی دیگر تصویر انتخاب می‌شدند. اما به دلیل اینکه این خواسته سوال نبود و با کمبود وقت مواجه شدیم این قابلیت را پیاده سازی نکردیم.

بخش ۵) در این بخش تصویر زیر را انتخاب کردیم. روش Seam Carving به طور خاص بر روی تصاویر منظره که نویز ساختاری زیادی دارند خوب جواب می‌دهد. زیرا ما انسان‌ها عادت کرده‌ایم کوه‌ها و سبزه‌ها و آسمان را به شکل‌های مختلفی ببینیم و این اجسام معمولاً ساختار نویزی ای دارند. به همین دلیل با تغییر کردن این نویزها به علت حذف شدن تعدادی Seam مغز ما متوجه چیز غیرعادی ای نمی‌شود. اما به عنوان مثال مغز انسان‌ها در تشخیص چهره انسان بسیار قدرتمند است و اگر Seam Carving بر روی چهره یک انسان انجام شود، حتی اگر نتوانیم متوجه بشویم که این یک تصویر فیک است، متوجه می‌شویم که این انسان کمی غیرعادی است!



Energy Map



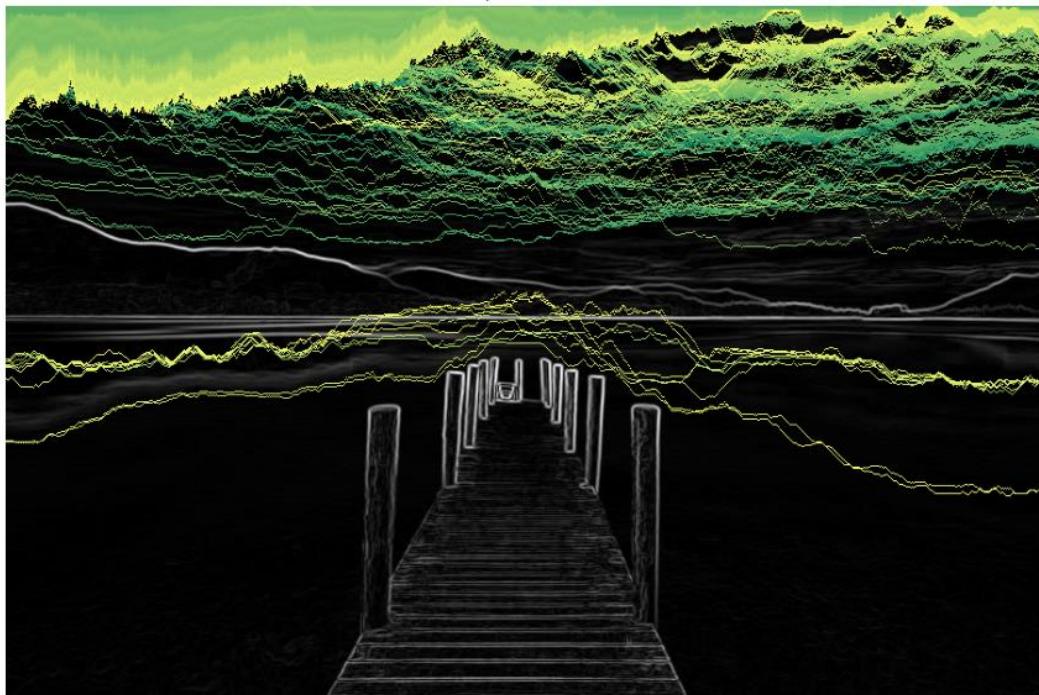
Top 30% seams



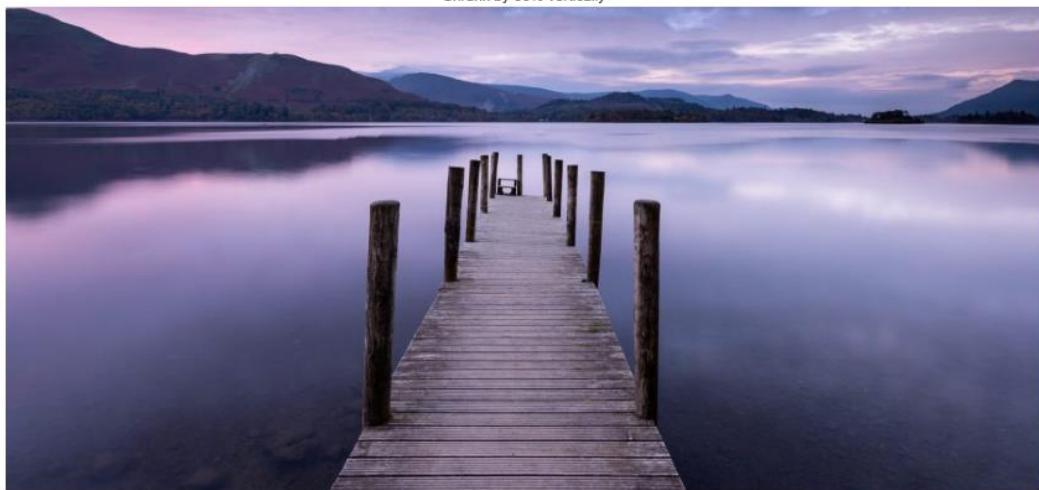
Shrunk by 30% horizontally



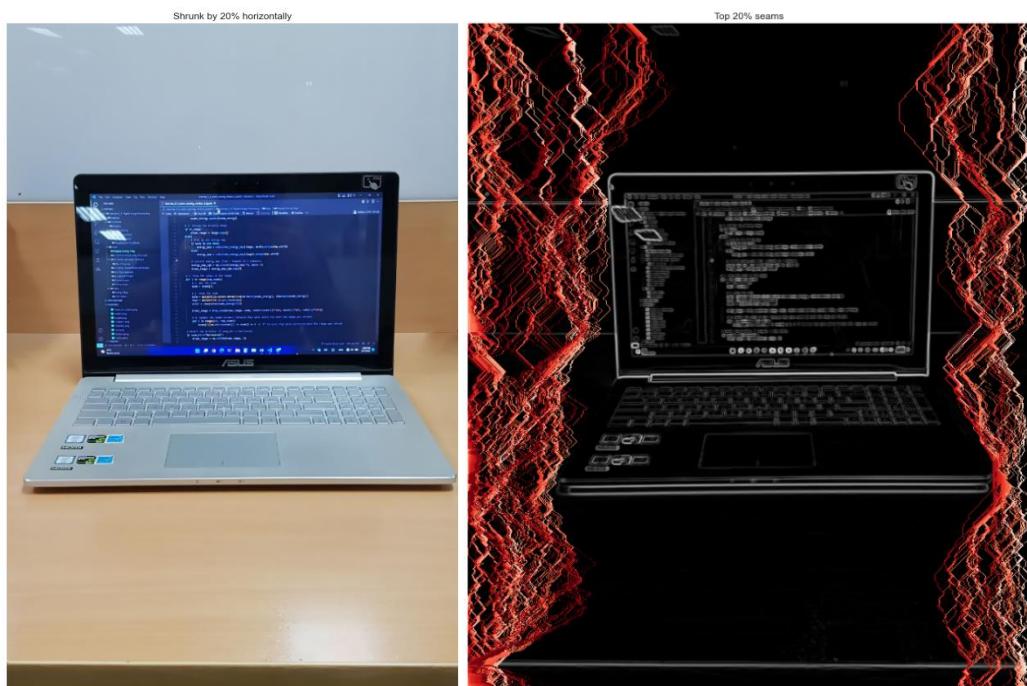
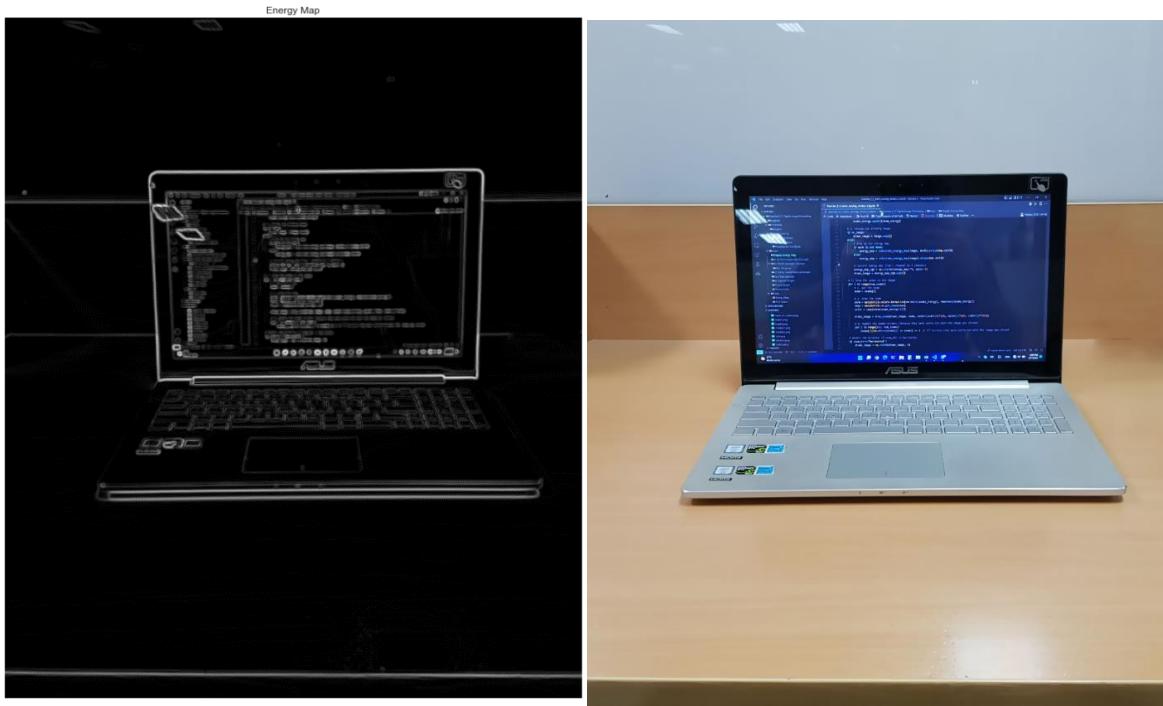
Top 30% seams



Shrunk by 30% vertically



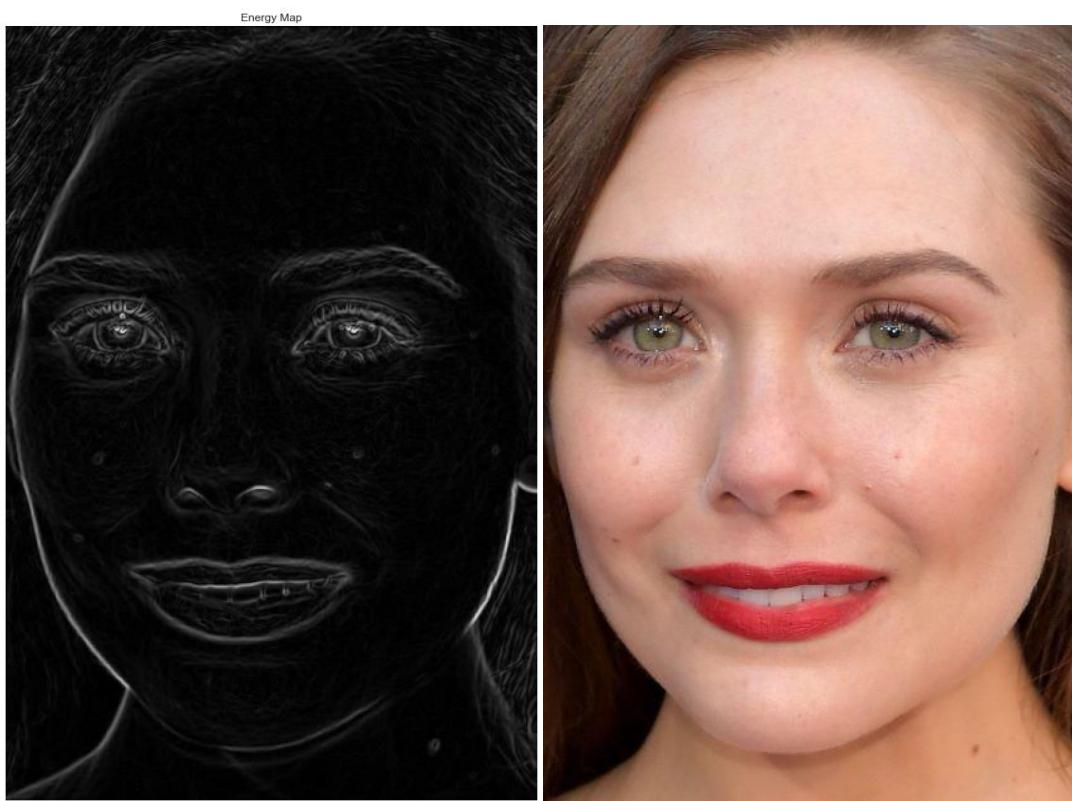
مثال دیگر:

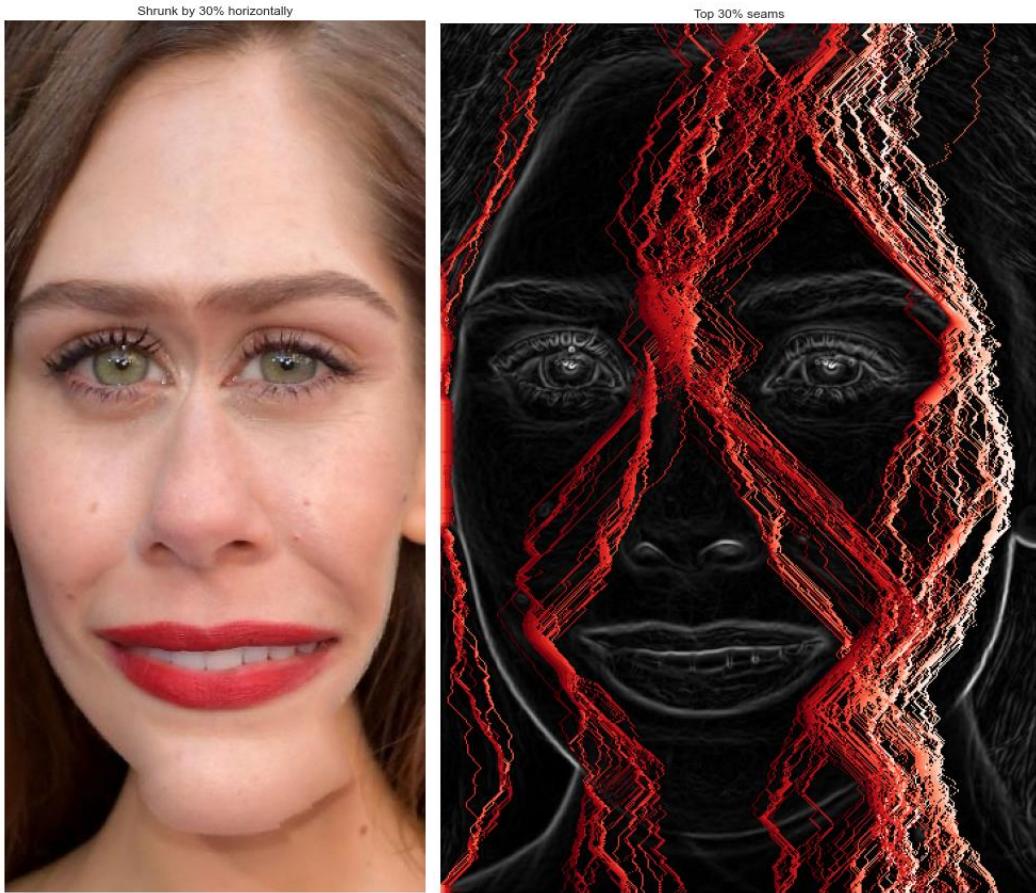




بخش d) همانطور که در بخش قبل توضیح داده شد، این الگوریتم با تصاویر چهره انسان به خوبی کار نمی‌کند و دلیل آن هم توانایی مغز ما در شناسایی کوچکترین نقص‌ها در چهره انسان‌ها است. علاوه بر این مورد، تصاویری که اشکال و خطوط هندسی و دقیقی دارند نیز در این دسته قرار می‌گیرند. به عنوان مثال اگر تصویر یک دایره را داشته باشیم، نمیتوان بدون تبدیل آن دایره به بیضی تعدادی سیم از یک جهت کاهش داد.

ابتدا تصویر چهره یک انسان را آزمایش می‌کنیم. (سعی شده تصویری استفاده شود که پس زمینه نداشته باشد زیرا حذف کردن پس زمینه عموماً بدون مشکل انجام می‌شود.)

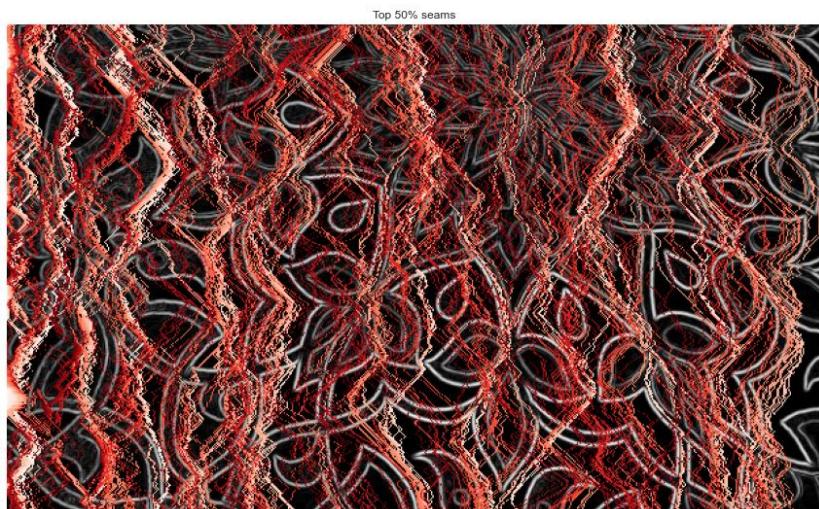




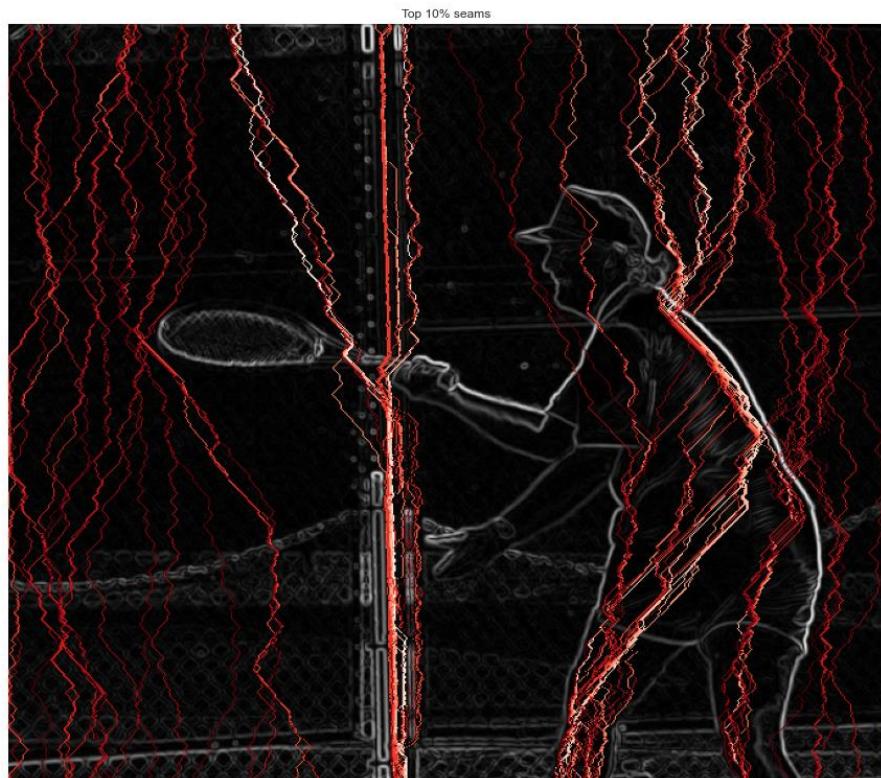
همانطور که مشاهده می‌شود، الگوریتم وظیفه اش را به درستی انجام داده و از نواحی صورت که انژی بالایی دارند دوری کرده و فقط نواحی پوست را حذف کرده. این دقیقاً همان کاری است که الگوریتم با تصویر منظره انجام داد. اما این بار با وجود اینکه اعضای اصلی صورت مانند چشم و دهان تغییری نکرده اند، ما میتوانیم مشکل دار بودن این تصویر را مشاهده کنیم. نزدیک شدن چشمها به یکدیگر یکی از ویژگی‌هایی است که این تصویر را مصنوعی جلوه داده در صورتی که اگر این اتفاق برای منظره می‌افتد و دو قله کوه به هم نزدیک‌تر می‌شوند ما متوجه نمی‌شديم و میگفتیم ممکن است اتفاق بی‌افتد دیگر!

مثال بعدی یک طراحی هندسی-اسلامی است. همانطور که میبینید این تصویر دارای منحنی‌هایی با خمیدگی مشخص است. اگر این منحنی‌ها متفاوت بودند ممکن بود از زیبایی طرح کاسته شود. موقع داریم پس از کاهش ابعاد این تصویر به طراحی ای بررسیم که یک طراح نابلد رسم کرده است. تصویر اصلی را مشاهده میکنید:





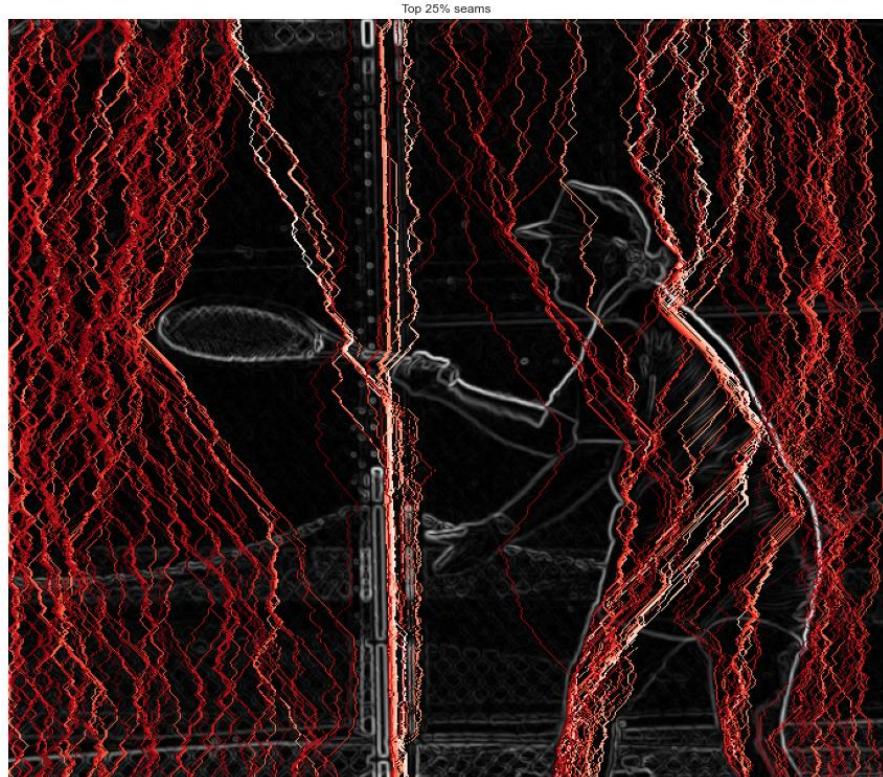
بخش e) در مقدمه نحوه بزرگ کردن تصاویر توضیح داده شد.



Top 10% seams



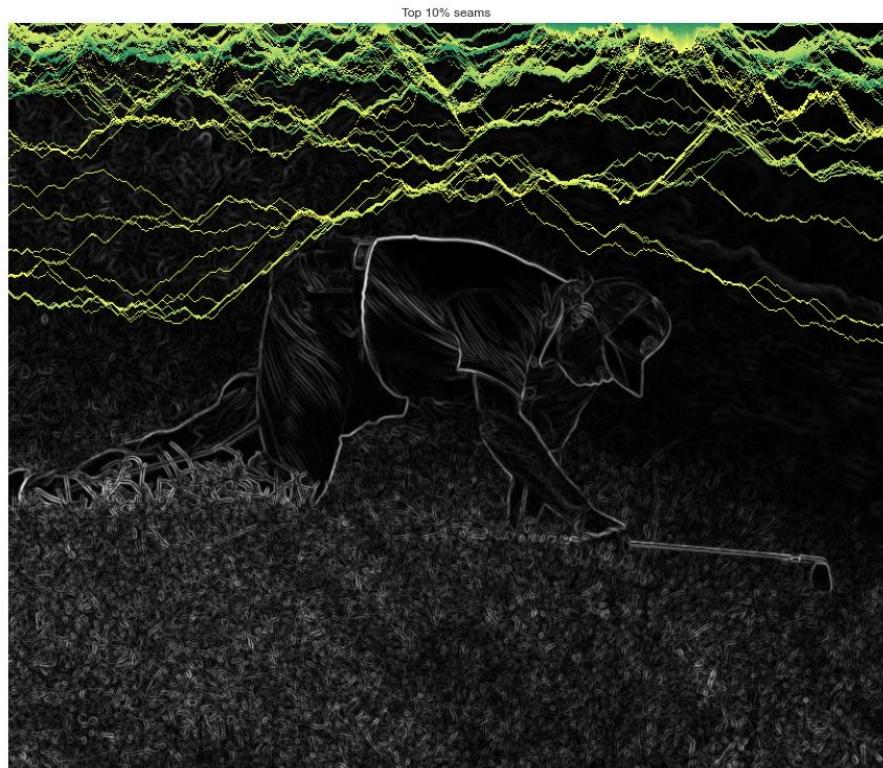
Expanded by 10% horizontally

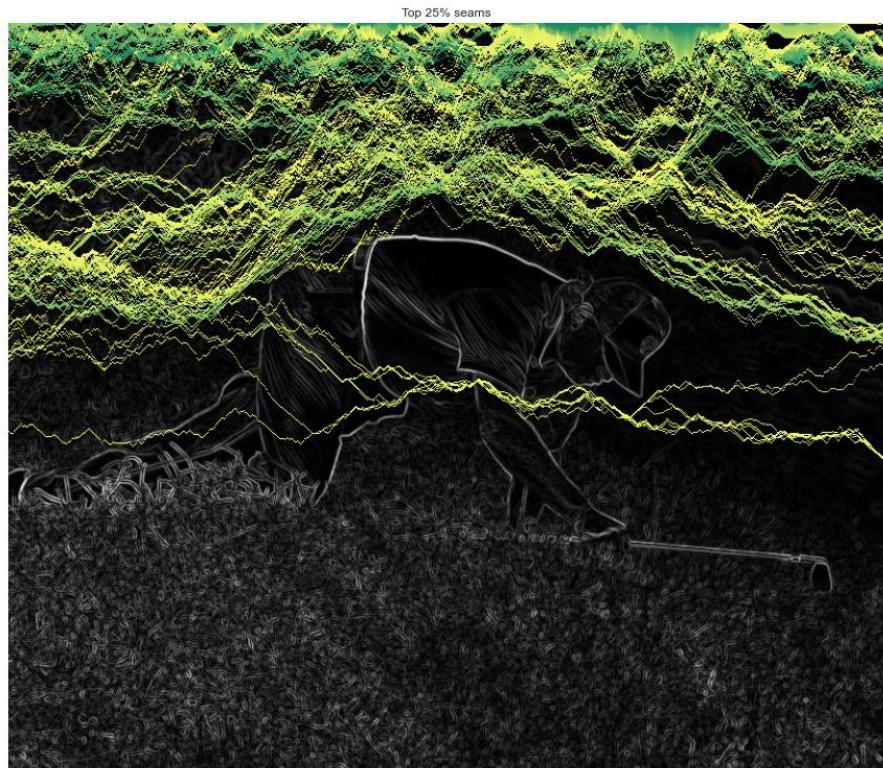


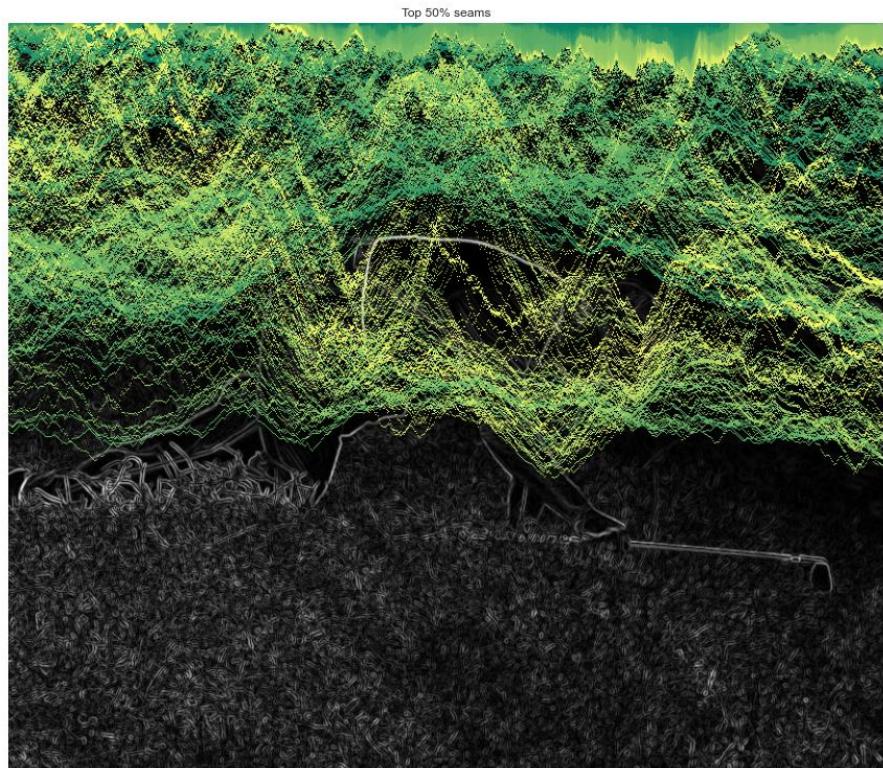
Expanded by 25% horizontally











و در نهایت تصویر مربوط به منظره را هم به صورت افقی و هم به صورت عمودی بزرگ کردیم. سیم‌های برتر همان سیم‌هایی است که در بخش d دیدیم.



Expanded by 30% vertically and horizontally



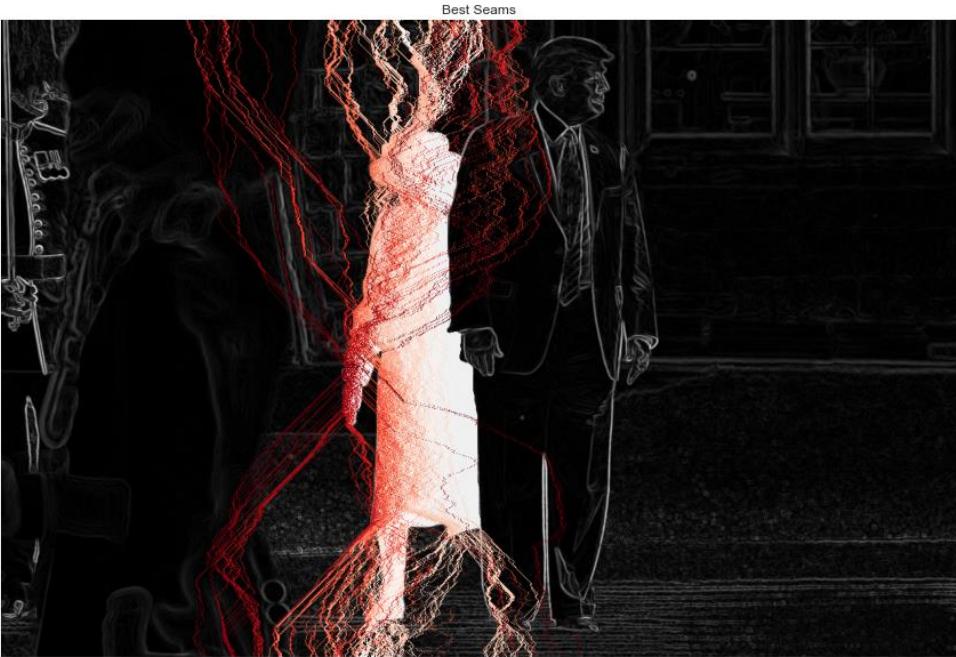
بخش ۴) این قسمت از ترکیب دو بخش کوچک کردن و بزرگ کردن تصویر تشکیل شده است. ابتدا آنقدر تصویر را کوچک میکنیم تا هیچ پیکسل سفیدی در ماسک باقی نماند باشد. سپس همانقدر تصویر را بزرگ میکنیم تا به اندازه اصلی تصویر برسیم.

ترامپ و ملکه



Energy Map





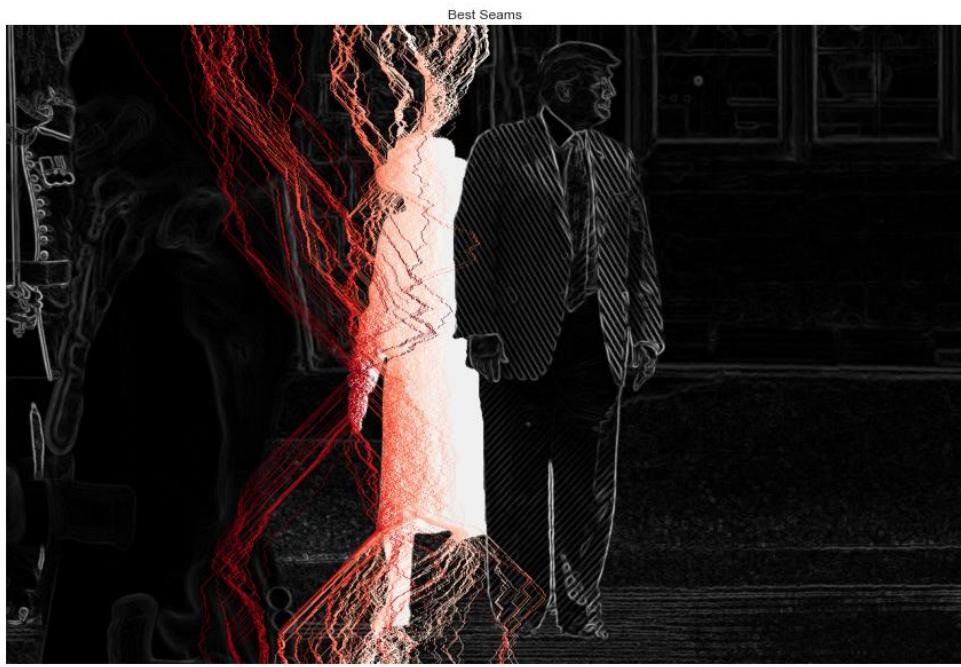


```
image.shape (850, 1276, 3)  
expanded_image.shape (850, 1276, 3)
```

همانطور که مشاهده میکنید، قسمتی از پا و دست راست ترامپ حذف شده است. دلیل آن این است که پس از حذف ملکه، دیوار سنگی کنار ملکه به ترامپ میچسبد و از آنجایی که جزییات بیشتری از لباس یکدست ترامپ دارد، انرژی بالاتری خواهد داشت و حذف کردن ترامپ اولویت پیدا خواهد کرد. برای حل این مشکل همانطور که در بخش های قبل گفتیم می توان یک ماسک مثبت داشت و به پیکسل های مربوط به ترامپ انرژی بسیار زیادی نسبت خواهد داد.

برای حل موقت این مشکل از تصویر زیر به عنوان جایگزین استفاده خواهیم کرد.

ترامپ و ملکه بپیشنهاد یافته



همانطور که می‌بینید بهترین سیم‌ها کمتر درون ناحیه مربوط به تراپ ورود می‌کنند.

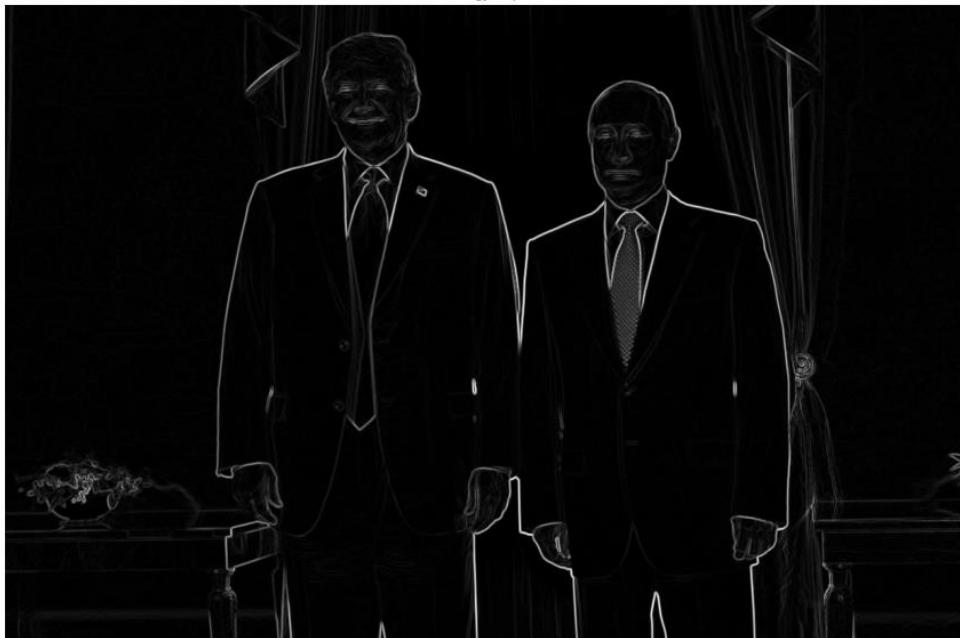


همانطور که انتظار می‌رفت هم مرحله کاهش اندازه و هم مرحله افزایش اندازه بهتر عمل کردند و به ناحیه ترامپ وارد نشدند. دلیل این اتفاق جزیيات بیشتر در ناحیه کت دونالد ترامپ است.

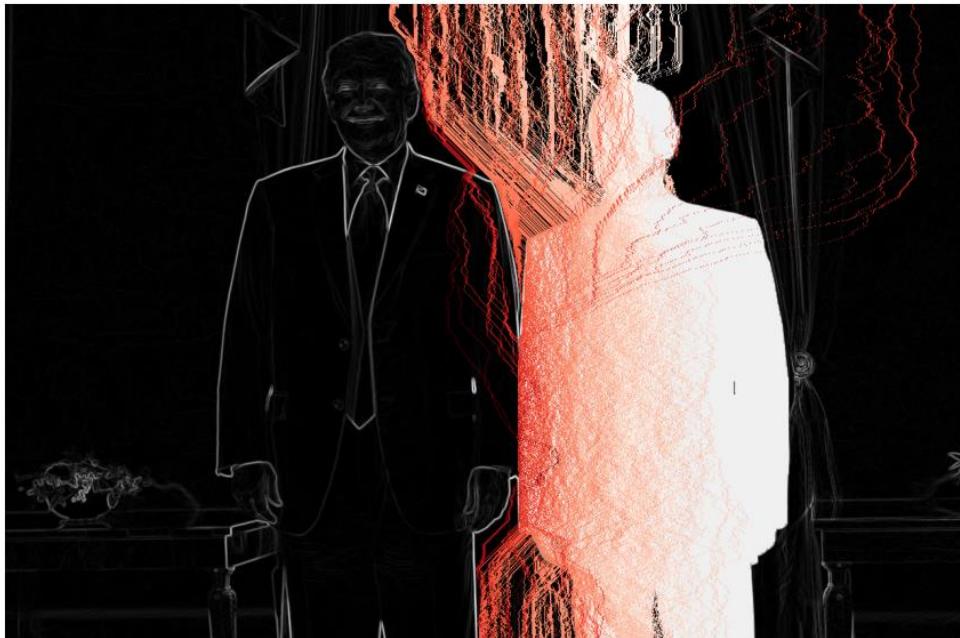
ترامپ و پوتین



Energy Map



Best Seams



Shrunk Image



Expanded Image

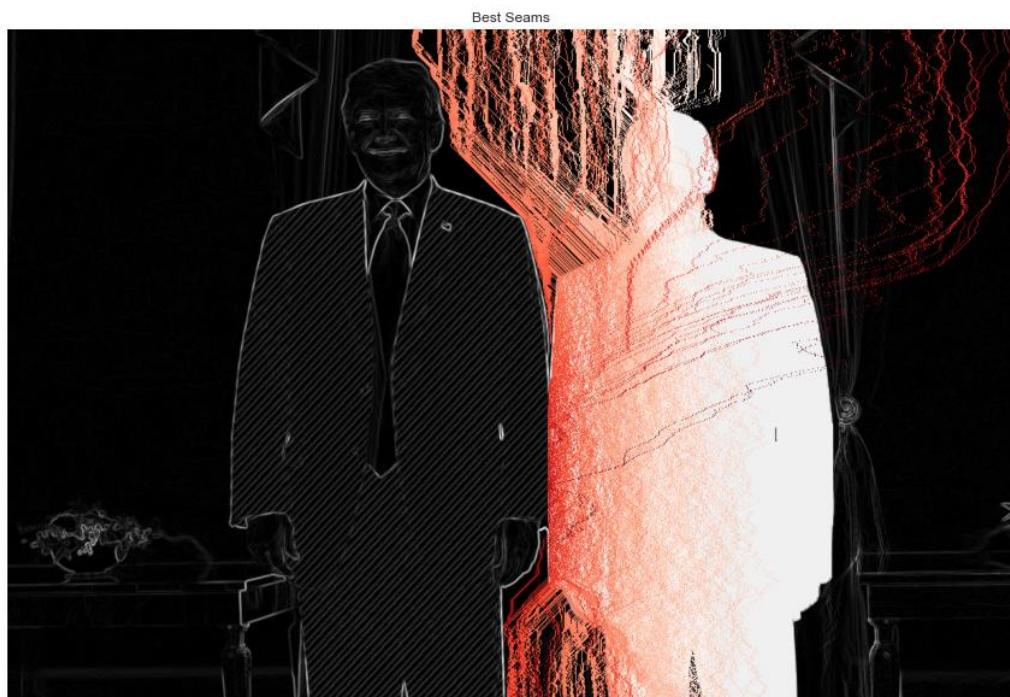


```
image.shape (820, 1232, 3)  
expanded_image.shape (820, 1232, 3)
```

در این مثال مشکل زیادی با بخش کوچک کردن نداریم اما در بخش بزرگ کردن بهترین سیم‌های سیم‌های مربوط به ترامپ انتخاب شده اند و به همین دلیل ترامپ چاق شده.

برای حل موقت این مشکل از تصویر زیر به عنوان جایگزین استفاده خواهیم کرد.

ترامپ و پوتین بهبود یافته





```
image.shape (820, 1232, 3)  
expanded_image.shape (820, 1232, 3)
```

همانطور که انتظار می‌رفت هم مرحله کاهش اندازه و هم مرحله افزایش اندازه بهتر عمل کردند و به ناحیه ترامپ وارد نشدند. دلیل این اتفاق جزیيات بيشتر در ناحيه کت دونالد ترامپ است

لازم به ذکر است که اینیمیشن مربوط به تعدادی از آزمایش های این سوال در فایل های تمرین موجود است که دیدن آنها خالی از لطف نیست.

سوال ۶

بخش a) برای پاسخ دادن به این سوال یک مثال ساده می‌زنیم. تصویر 2×2 ای را در نظر بگیرید. میخواهیم یک فیلتر Spatial 3×3 را بر روی آن اعمال کنیم.

1	2
3	4

برای اعمال فیلتر 3×3 به نقطه ۱، ابتدا تصویر 2×2 را pad میدهیم.

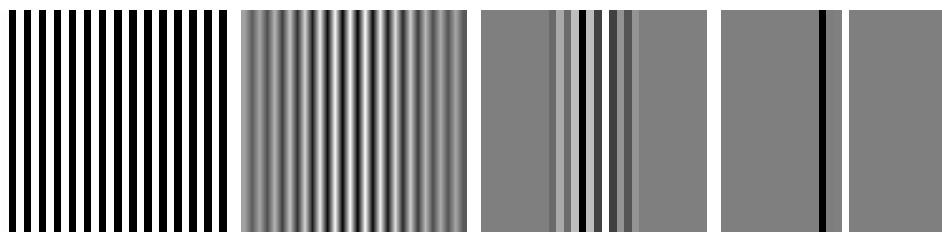
.	.	.	.
.	1	2	.
.	3	4	.
.	.	.	.

حالا وقتی فیلتر 3×3 را به مرکزیت سلول ۱ قرار میدهیم به شکل زیر می‌رسیم. سلول‌های فیلتر 3×3 با رنگ سبز رنگ نشان داده شده‌اند.

.	.	.	.
.	1	2	.
.	3	4	.
.	.	.	.

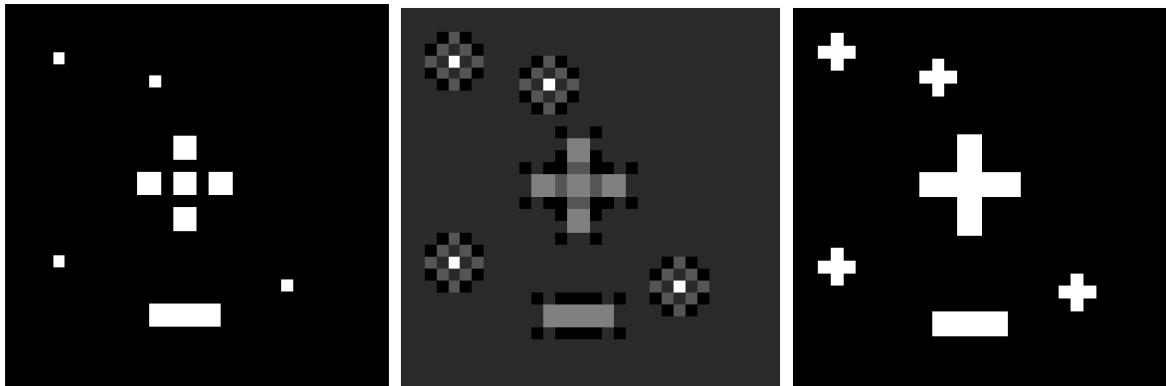
بنابراین با استفاده از تکنیک Padding هیچ مشکلی برای استفاده از فیلتر بزرگتر از تصویر رخ نمی‌دهد. اندازه Padding مورد نظر در جهت افقی برابر با $2 \times \left\lceil \frac{H_{filter}}{2} \right\rceil$ و در جهت عمودی $2 \times \left\lceil \frac{W_{filter}}{2} \right\rceil$ است که W به ترتیب عرض و ارتفاع فیلتر هستند.

بخش b) بله. طبق آزمایش‌هایی که انجام شد پس از تعدادی تکرار، به یک حالت پایدار می‌رسد. ویدیوی مربوط به این آزمایش در پوشه P6_results قرار دارد. تصویر زیر در ابتدا یک Edge بود که در نهایت به پس از ۱۰۰ مرحله به تصویر سمت چپ همگرا شد.



بخش c) خیر امکان پذیر نیست. زیرا این روش Scale invariant نیست و فقط میتواند + هایی با اندازه مشخصی را پیدا کند.

تصویر سمت راست تصویر ورودی است، تصویر وسط نتیجه اعمال فیلتر است و تصویر چپ نتیجه اعمال Non Maximum Suppression برای پیدا کردن روشن ترین نقاط محلی است. همانطور که دیده می شود + هایی که به اندازه فیلتر بودند به خوبی تشخیص داده شده اند و روشن ترین نقاط در تصویر هستند. اما بین + بزرگتر و - تفاوت زیادی وجود ندارد و امکان شناسایی و تفکیک آنها وجود ندارد.



فیلتر مورد استفاده:

$$\begin{array}{|c|c|c|} \hline -1 & 1 & -1 \\ \hline 1 & 1 & 1 \\ \hline -1 & 1 & -1 \\ \hline \end{array}$$

بخش d) فیلتر کردن تصویر با دو فیلتر 1 بعدی بهینه تر است. به جدول زیر توجه کنید.

اندازه تصویر $m \times n$ است.

تعداد جمع	تعداد ضرب	نوع فیلتر
$(m \times n)((k \times l) - 1)$	$(m \times n)(k \times l)$	$k \times l$ 2 بعدی
$(m \times n)(k - 1)$	$(m \times n)(k)$	k 1 بعدی
$(m \times n)(l - 1)$	$(m \times n)(l)$	l 1 بعدی
$(m \times n)(k + l - 2)$	$(m \times n)(k + l)$	دو فیلتر 1 بعدی بالا

واضح است که اعمال دو فیلتر 1 بعدی بهینه تر است.

بخش e) در این نوع تصاویر که در بین آنها زاویه دوربین ثابت است و پس زمینه نیز حرکت نمی کند، میانگین گیری می تواند باعث کاهش نویز شود زیرا نویز در یک پیکسل خاص، برخلاف محتوی واقعی صحنه، در بین تصاویری که به صورت پی در پی گرفته شده اند ثابت نیست و تغییر می کند. دلیل وقوع این نویزها اختلالات الکتریکی و ... است و باسته به زاویه دوربین نیستند. از آن طرف محتوی اصلی صحنه تنها با تغییر زاویه دوربین تغییر میکند (در این مثال فرض کرده ایم وضعیت نوری تغییر نمیکند و پس زمینه ثابت است). بنابراین میانگین گیری بین تصاویر مختلف باعث می شود نویزهای موجود در پیکسل ها یکدیگر را خنثی کنند و قدرتشان در عمل میانگین گیری کمتر از قدرت پیکسل هایی با مقادیر واقعی متعلق به صحنه شود.