

به نام خدا



درس مبانی بینایی کامپیوتر

---

### تمرین سری سوم

---

مدرس درس:  
جناب آقای دکتر محمدی

تهیه شده توسط:  
الناز رضایی ۹۸۴۱۱۳۸۷

تاریخ ارسال: ۱۴۰۱/۰۷/۲۷

### سوال ۱:

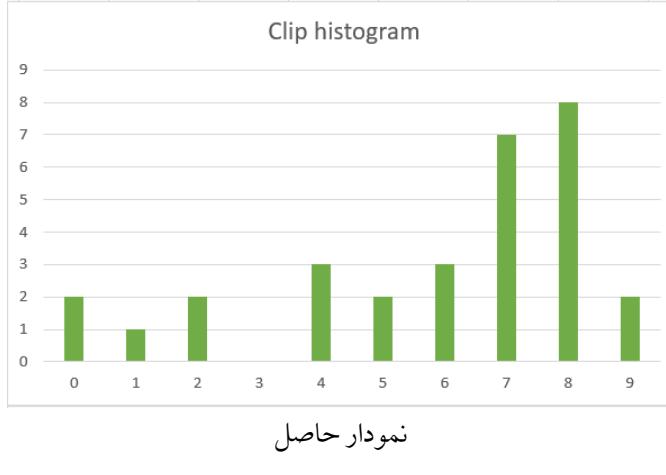
شکل زیر یک تصویر ۱۰ سطحی است (محدوده مقادیر ممکن روشنایی از صفر تا ۹ هستند). . ابتدا هیستوگرام این تصویر را به دست آورید و با استفاده از برش هیستوگرام ۱۰ درصد از مولفه های بالا و پایین را قطع کنید و هیستوگرام به دست آمده را رسم کنید. سپس با استفاده از متعادل سازی هیستوگرام برش خورده، تصویر را بهبود دهید و هیستوگرام نهایی را رسم کنید (حل تشریحی کامل را انجام دهید؛ مثلا مراحل متعادل سازی هیستوگرام را هم بنویسید و فقط جواب آخر نباشد). (۳۰ نمره)

۷	۷	۸	۸	۸	۸
۲	۱	۴	۴	۴	۸
۷	۰	۵	۵	۲	۸
۸	۰	۶	۹	۹	۷
۸	۷	۶	۶	۷	۷

### پاسخ ۱:

• هیستوگرام تصویر:

k	0	1	2	3	4	5	6	7	8	9
$h_r(k) = n_k$	2	1	2	0	3	2	3	7	8	2



• هیستوگرام با برش ۱۰ درصد:

$$0.10 * 30 = 3$$

پس نتیجه می‌گیریم ۳ پیکسل از ابتدا و انتها باید حذف کنیم. بنابراین مقادیر هیستوگرام جدید مطابق جدول زیر می‌باشد:

k	0	1	2	3	4	5	6	7	8	9
$h_r(k) = n_k$	0	0	2	0	3	2	3	7	7	0

همچنانی نمودار آن نیز به شکل زیر درمی‌آید:



حال فرمول برش هیستوگرام که به شرح زیر می‌باشد را بر روی مقادیر سطوح روشنایی تصویر، پیاده‌سازی می‌کنیم:

$$g(x, y) = \text{clip}[f(x, y)] = \left( \frac{f(x, y) - f_{10}}{f_{90} - f_{10}} \right) (MAX - MIN) + MIN$$

$$f_{10} = f(0.1 * 24) = f(2.4) = f(2) = 2$$

$$f_{90} = f(0.9 * 24) = f(21.6) = f(22) = 8$$

$$g(0) = \left( \frac{0-2}{8-2} \right) (9 - 0) + 0 = -3 \Rightarrow 0$$

$$g(1) = \left( \frac{1-2}{8-2} \right) (9 - 0) + 0 = -\frac{3}{2} \Rightarrow 0$$

$$g(2) = \left( \frac{2-2}{8-2} \right) (9 - 0) + 0 = 0 \Rightarrow 0$$

$$g(3) = \left( \frac{3-2}{8-2} \right) (9 - 0) + 0 = 1.5 \Rightarrow 2$$

$$g(4) = \left( \frac{4-2}{8-2} \right) (9 - 0) + 0 = 3 \Rightarrow 3$$

$$g(5) = \left( \frac{5-2}{8-2} \right) (9 - 0) + 0 = 4.5 \Rightarrow 5$$

$$g(6) = \left( \frac{6-2}{8-2} \right) (9 - 0) + 0 = 6 \Rightarrow 6$$

$$g(7) = \left( \frac{7-2}{8-2} \right) (9 - 0) + 0 = 7.5 \Rightarrow 8$$

$$g(8) = \left( \frac{8-2}{8-2} \right) (9 - 0) + 0 = 9 \Rightarrow 9$$

$$g(9) = \left( \frac{9-2}{8-2} \right) (9 - 0) + 0 = 10.5 \Rightarrow 9$$

حال مقادیر سطح روشنایی خانه‌های تصویر را با مقادیر به دست آمده جایگزین می‌کنیم.

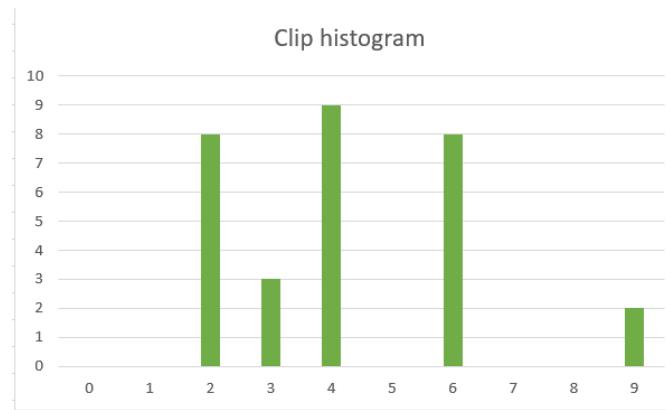
تصویر پس از برش و کشش به شکل زیر درمی‌آید:

8	8	9	9	9	9
0	0	3	3	3	9
8	0	5	5	0	9
9	0	6	9	9	8
9	8	6	6	8	8

تصویر حاصل

سپس مقادیر جدید هیستوگرام را به دست آورده و هیستوگرام جدید را رسم می‌کنیم:

k	0	1	2	3	4	5	6	7	8	9
$h_r(k) = n_k$	5	0	0	3	0	2	3	0	7	10



نمودار حاصل

• متعادل سازی:

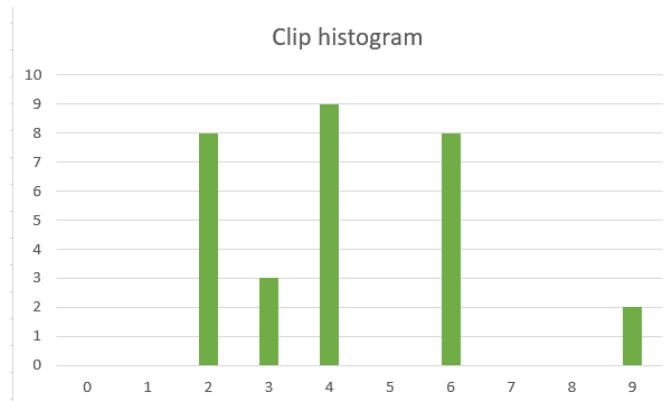
k	0	1	2	3	4	5	6	7	8	9
$h_r(k) = n_k$	5	0	0	3	0	2	3	0	7	10
$\sum_{j=0}^k n_j$	5	5	5	8	8	10	13	13	20	30
$\sum_{j=0}^k \frac{n_j}{n}$	$\frac{5}{30}$	$\frac{5}{30}$	$\frac{5}{30}$	$\frac{8}{30}$	$\frac{8}{30}$	$\frac{10}{30}$	$\frac{13}{30}$	$\frac{13}{30}$	$\frac{20}{30}$	$\frac{30}{30}$
$(L - 1) \sum_{j=0}^k \frac{n_j}{n}$	1.5	1.5	1.5	2.4	2.4	3	3.9	3.9	6	9
Round	2	2	2	2	2	3	4	4	6	9

• هستوگرام و تصویر جدید:

4	4	6	6	6	6
2	2	2	2	2	6
4	2	3	3	2	6
6	2	3	9	9	4
6	4	4	4	4	4

تصویر حاصل

k	0	1	2	3	4	5	6	7	8	9
$h_r(k) = n_k$	0	0	8	3	9	0	8	0	0	2



نمودار حاصل

## سوال ۲:

در این سوال باید موارد زیر را در نوتبوک Q2 پیاده سازی کنید (در این سوال تصاویر را به صورت سیاه و سفید (تک کانال) بخوانید.): (۴۰ نمره)

- (الف) ابتدا تابع متعادل سازی هیستوگرام را به کمک Python و Numpy پیاده سازی کنید و تابع خود را روی تصویر River اعمال کنید. سپس از تابع آماده متعادل سازی هیستوگرام که در کتابخانه OpenCV موجود است، استفاده کنید و آن را مجددا روی تصویر River اعمال کنید. نتایج را به همراه تصویر حاصل شده، گزارش کنید و با هم مقایسه کنید.
- (ب) از تابع CLAHE که در کتابخانه OpenCV موجود است، استفاده کنید و آن را روی تصویر River اعمال کنید. نتیجه را به همراه تصویر حاصل شده، گزارش کنید و با نتایج قسمت اول یعنی متعادل سازی هیستوگرام مقایسه کنید.
- (پ) قسمتهای ۱ و ۲ را برای تصویر City نیز انجام دهید.
- (ت) اگر با تصاویر رنگی سروکار داشتیم و میخواستیم کیفیت آنها را ارتقا دهیم، آیا خواندن تصویر به صورت RGB و مثال اعمال تابع متعادل سازی هیستوگرام روی هر کانال به صورت جداگانه میتوانست کیفیت تصویر را ارتقا دهد؟ اگر نه راه حل مناسب تری برای آن شرح دهید.

## پاسخ ۲:

- ۲- (الف) ابتدا با توجه به فرمول زیر، کد خود را برای نرمال سازی تصویر مینویسیم.

$$(L - 1) \sum_{j=0}^n \frac{n_j}{n}$$

کد مربوط به این بخش، به شرح زیر میباشد:

```
path = r'E:\University\Term7\FCV\Homeworks\HW3\Images'
img = cv2.imread(os.path.join(path, 'River.jpg'), 0)
orginalImg = cv2.imread(os.path.join(path, 'River.jpg'), 0)
### YOUR CODE #####
# START
equ = hist_eq(img)
# END

res = np.hstack((orginalImg, equ)) #stacking images side-by-side
cv2.imwrite(os.path.join(path, 'normalizedRiver.jpg'), equ)
plt.figure(figsize=(16, 16))
plt.imshow(res, cmap='gray')
```

کد برای نمایش خروجی تابع نرمال سازی تصویر

```

from numpy import histogram

def hist_equ(image):
    ...
    input:
    image (ndarray): input image
    output:
    output_image (ndarray): enhanced image
    ...

#####
# Your code
# Start
hist = np.zeros((256,), dtype=int)
w = image.shape[0]
h = image.shape[1]
L = 256
numN = w * h
#histogram
for i in range(0,w):
    for j in range(0,h):
        hist[image[i,j]]+=1
#normalized histogram
for i in range(1,L+1):
    if i != L:
        hist[i] += hist[i-1]
    hist[i-1] = (L-1)*(hist[i-1]/numN)
#make image normalized
for i in range(0,w):
    for j in range(0,h):
        image[i,j] = hist[image[i,j]]
# End

return image

```

کد تابع برای نرمالسازی تصویر



نتیجه تابع برای نرمالسازی تصویر

سپس بار دیگر با استفاده ازتابع آماده متعادل‌سازی هیستوگرام، تصویر خود را نرمال می‌کنیم.  
کد زده شده برای این بخش نیز مطابق تصویر زیر می‌باشد.

```
img = cv2.imread(os.path.join(path, 'River.jpg'), 0)

### YOUR CODE #####
# START
equ = cv2.equalizeHist(img)
# END

res = np.hstack((img, equ)) #stacking images side-by-side
cv2.imwrite(os.path.join(path, 'cvNormalizedRiver.jpg'),equ)
plt.figure(figsize=(16, 16))
plt.imshow(res, cmap='gray')
```

کد تابع آماده open cv متعادل‌سازی هیستوگرام



نتیجه کد تابع آماده open cv متعادل‌سازی هیستوگرام

از مقایسه دو تصویر خروجی، نتیجه می‌گیریم کتراست در هر دو تصویر افزایش یافته است و این به دلیل استفاده از نرمال‌سازی می‌باشد. همچنین با توجه به اینکه دو تصویر تفاوت چندانی با هم ندارند، می‌توانیم نتیجه بگیریم که به جای پیاده‌سازی هیستوگرام توسعه خودمان، می‌توانیم از تابع آماده آن استفاده کنیم.

۲- ب) گاهی اوقات، استفاده از تابع متعادل‌سازی هیستوگرام، به علت روشنایی زیاد، باعث از دست دادن برخی از جزئیات تصویر می‌شود. چرا که هیستوگرام به یک محدوده خاص، محدود نشده است. برای حل این مشکل، از روش adaptive histogram equalization استفاده می‌شود که در آن، تصویر را به blocks کوچک‌تری که اندازه آن به صورت default در openCV ۸\*۸ می‌باشد، تقسیم می‌کنند. سپس برای هر یک از این بخش‌های کوچک‌تر، عملیات متعادل‌سازی هیستوگرام انجام می‌شود. در این صورت، اگر نویزی وجود داشته باشد، تقویت، می‌شود. بنابراین برای جلوگیری از این اتفاق، محدودیت کتراست اعمال می‌شود.

به این صورت که اگر هر یک از این هیستوگرام‌ها بالاتر از حد کنترast مشخص شده باشند، بریده شده و به صورت یکنواخت در خانه‌های هیستوگرام‌های دیگر انداده می‌شوند. تابع CLAHE در openCV این کار را انجام می‌دهد. ورودی‌های این تابع optional دارند و می‌توانند نباشند. ورودی اولی که در کد زیر به این تابع داده شده است، محدودیت کنترast و ورودی دوم، اندازه خانه‌هایی است که در آن می‌خواهیم هیستوگرام را محاسبه کنیم.

```
def CLAHE(image, gridSize, clip_limit):
    ...
    inputs:
        image (ndarray): input image
        gridSize (tuple): window size
        clip_limit (int): threshold for contrast limiting
    output:
        output_image (ndarray): improved image
    ...

    #####
    # Your code
    # Start
    clahe = cv2.createCLAHE(clipLimit=clip_limit, tileGridSize=gridSize)
    output_image = clahe.apply(image)
    # End

    return output_image
```

```
✓ 0.4s

img = cv2.imread(os.path.join(path, 'River.jpg'), 0)

### YOUR CODE ###
# START
clh = CLAHE(img, (8,8), 2.0)
# END

res = np.hstack((img, clh)) #stacking images side-by-side
cv2.imwrite(os.path.join(path, 'claheNormalizedRiver.jpg'), equ)
plt.figure(figsize=(16, 16))
plt.imshow(res, cmap='gray')
```

پیاده‌سازی تابع CLAHE



نتیجه این بخش

همانطور که طبق توضیحات هم مشخص بود، این تابع از بخش اول عملکرد بهتری دارد. در تصویر خروجی این بخش نیز، شاخه‌ها و برگ‌های درختان با جزئیات بیشتری دیده می‌شوند و کنتراس است تصویر افزایش یافته است، در صورتی که در تصویرهای قبلی، جزئیات باوضوح مشخص نبود.

نکته: برای توضیحات شفاهی این بخش، از لینک زیر استفاده شد:

[https://docs.opencv.org/4.x/d5/daf/tutorial\\_py\\_histogram\\_equalization.html](https://docs.opencv.org/4.x/d5/daf/tutorial_py_histogram_equalization.html)

(۲-پ)

```
#histogram equalization implementation function
img = cv2.imread(os.path.join(path, 'City.jpg'), 0)
originalImg = cv2.imread(os.path.join(path, 'City.jpg'), 0)
equ = hist_equ(img)
res = np.hstack((originalImg, equ)) #stacking images side-by-side
plt.figure(figsize=(16, 16))
plt.imshow(res, cmap='gray')

#histogram equalization OpenCV library|
img = cv2.imread(os.path.join(path, 'City.jpg'), 0)
equ = cv2.equalizeHist(img)
res = np.hstack((img, equ)) #stacking images side-by-side
plt.figure(figsize=(16, 16))
plt.imshow(res, cmap='gray')

#CLAHE OpenCV library
img = cv2.imread(os.path.join(path, 'City.jpg'), 0)
clh = CLAHE(img, (8,8), 2.0)
res = np.hstack((img, clh)) #stacking images side-by-side
plt.figure(figsize=(16, 16))
plt.imshow(res, cmap='gray')
```

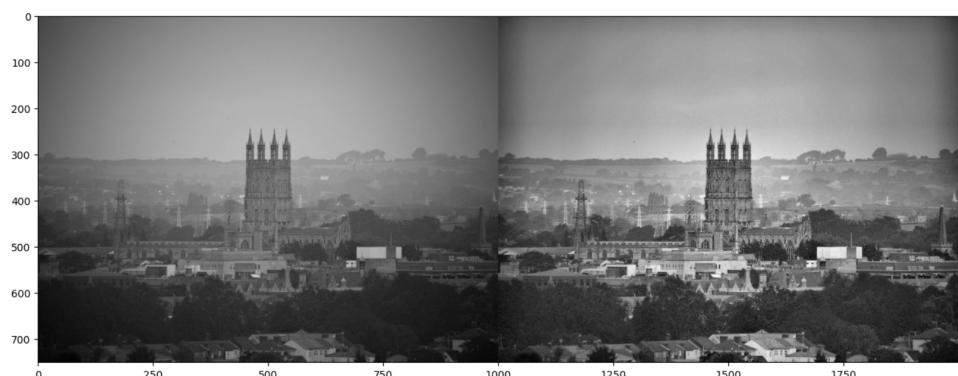
کدهای مربوط به این بخش



نتیجه متعادل سازی با استفاده از پیاده سازی تابع



نتیجه متعادل سازی با استفاده از تابع آماده OpenCV



نتیجه متعادل سازی با استفاده از تابع CLAHE

همانطور که در این بخش نیز مشخص است، با استفاده از تابع CLAHE، تصویر شارپتر و با کنتراست بالاتری داریم و جزئیات تصویر در آن، نسبت به بخش‌های قبل، واضح‌تر می‌باشد.

۲-ت) خیر، نمی‌توان این کار را انجام داد؛ چرا که متعادل‌سازی هیستوگرام یک فرآیند غیر خطی می‌باشد و تقسیم کanal‌ها و انجام عملیات متعادل‌سازی به صورت مجزا روی هر یک از کanal‌ها، کار نادرستی است. در واقع متعادل‌سازی شامل مقادیر شدت روشنایی تصویر می‌باشد، نه اجزای رنگ. بنابراین عملیات متعادل‌سازی را نمی‌توانیم مستقیماً روی کanal‌ها اعمال کنیم، بلکه به گونه‌ای باید اعمال شود که مقادیر شدت روشنایی، بدون بر هم زدن تعادل رنگ‌های تصویر نیز برقرار باشند. پس تنها کافیست تصویر را به یکی از فضاهایی که شدت روشنایی را از اجزای رنگ جدا می‌کند ببریم. برخی از گزینه‌های پیشنهادی به شرح زیر می‌باشند:

HSV/HLS، YUV، YCbCr ، ...

منبع:

<https://prateekvjoshi.com/2013/11/22/histogram-equalization-of-rgb-images/>

### سوال ۳:

در این سوال باید موارد زیر را در نوتبوک Q3 (پیاده سازی کنید) در این سوال تصاویر را به صورت رنگی (سه کanal) بخوانید: (۰ نمره)

- الف) از تابع آماده تطبیق هیستوگرام که در کتابخانه skimage موجود است، استفاده کنید و به کمک آن هیستوگرام تصویر ورودی Hades را بر هیستوگرام تصویر ورودی A Plague Tale تطبیق دهید. نتیجه را همراه تصویر حاصل شده، گزارش کنید.
- ب) تابع تطبیق هیستوگرام را به کمک Numpy و Python پیاده سازی کنید و به کمک آن هیستوگرام تصویر ورودی Hades را بر هیستوگرام تصویر ورودی A Plague Tale تطبیق دهید. نتیجه را به همراه تصویر حاصل شده، گزارش کنید و با نتایج قسمت اول یعنی کد آماده مقایسه کنید.
- پ) مجدداً قسمتهای ۱ و ۲ را اجرا کنید و این بار هیستوگرام تصویر ورودی A Plague Tale را بر هیستوگرام تصویر ورودی Hades تطبیق دهید

### پاسخ ۳:

۳-الف) برای انجام این بخش، از تابع match\_histograms استفاده می‌کنیم. وروری‌های این تابع به ترتیب تصویر مبدا و تصویر مرجع و همینطور یک boolean که نشان‌دهنده یک یا چند کanalه بودن است، می‌باشد. هدف ما از این کار، گرفتن یک تصویر ورودی و آپدیت کردن

شدت پیکسل های آن است، به گونه ای که توزیع هیستوگرام تصویر ورودی و مرجع، مطابقت داشته باشند. در شکل های زیر، کد مربوط به این بخش و نتیجه حاصله را مشاهده می نمایید.

```

path = r'E:\University\Term7\FCV\Homeworks\HW3\Images'
source = plt.imread(os.path.join(path,'A Plague Tale.jpg'))
reference = plt.imread(os.path.join(path,'Hades.jpg'))

### YOUR CODE ####
# START
matched = match_histograms(source, reference, multichannel=True)
# END

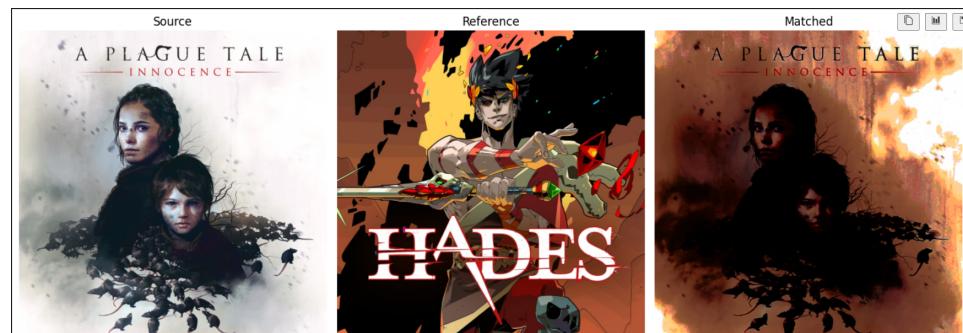
fig, (ax1, ax2, ax3) = plt.subplots(nrows=1, ncols=3, figsize=(14, 14),
                                    sharex=True, sharey=True)
for aa in (ax1, ax2, ax3):
    aa.set_axis_off()

ax1.imshow(source)
ax1.set_title('Source')
ax2.imshow(reference)
ax2.set_title('Reference')
ax3.imshow(matched)
ax3.set_title('Matched')

plt.tight_layout()
plt.show()

```

کد تطبیق هیستوگرام



نتیجه حاصل شده

همانطور که در شکل بالا مشخص است، شدت روشنایی پیکسل های تصویر source بر مبنای توزیع هیستوگرام تصویر reference آپدیت شده اند.

- ۳-ب) در این بخش نیز با استفاده از numpy تابع تطبیق هیستوگرام را پیاده سازی می کنیم.

## پیاده‌سازی کد تطبیق هیستوگرام

```
path = r'E:\University\Term7\FCV\Homeworks\HW3\Images'
source = plt.imread(os.path.join(path, 'A Plague Tale.jpg'))
reference = plt.imread(os.path.join(path, 'Hades.jpg'))

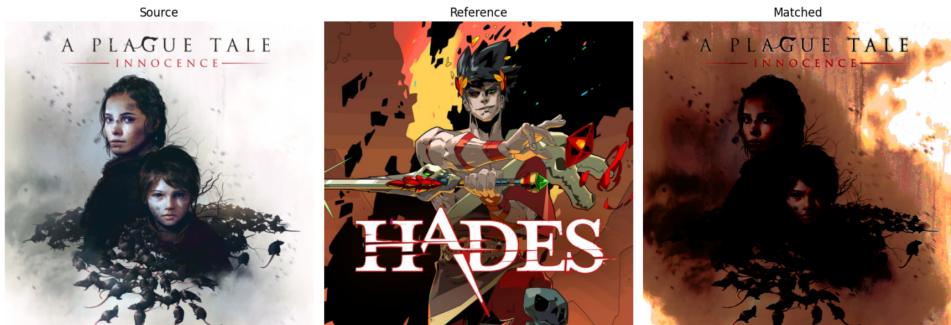
### YOUR CODE ####
# START
matched = hist_matching(source, reference)
# END

fig, (ax1, ax2, ax3) = plt.subplots(nrows=1, ncols=3, figsize=(14, 14),
                                    sharex=True, sharey=True)
for aa in (ax1, ax2, ax3):
    aa.set_axis_off()

ax1.imshow(source)
ax1.set_title('Source')
ax2.imshow(reference)
ax2.set_title('Reference')
ax3.imshow(matched)
ax3.set_title('Matched')

plt.tight_layout()
plt.show()
```

## پیاده‌سازی خروجی کد تطبیق هیستوگرام



نتیجه حاصل شده

همانطور که مشاهده می‌شود، این تصویر نیز مانند تصویر حاصل شده از بخش قبل، شدت روشنایی پیکسل‌های تصویر ورودی، براساس توزیع هیستوگرام تصویر مرجع، تغییر می‌دهد.  
منع:

<https://vzaguskin.github.io/histmatching1/>

- ۳-پ) با تغییر دادن تصویر منع و مرجع، همانطور که در شکل مشاهده می‌شود این بار شدت روشنایی تصویر A Plague Tale را بر اساس تصویر Hades تغییر می‌دهد و تصویر به سمت رنگ‌های خنثی‌تر می‌رود. همچنین خروجی به ازای هر دو تابع تقریباً مشابه می‌باشد.

```

# match_histograms OpenCV
path = r'E:\University\Term7\FCV\Homeworks\HW3\Images'
source = plt.imread(os.path.join(path, 'A Plague Tale.jpg'))
reference = plt.imread(os.path.join(path, 'Hades.jpg'))
matched = match_histograms(reference, source, multichannel=True)
fig, (ax1, ax2, ax3) = plt.subplots(nrows=1, ncols=3, figsize=(14, 14),
                                    sharex=True, sharey=True)
for aa in (ax1, ax2, ax3):
    aa.set_axis_off()
ax1.imshow(source)
ax1.set_title('Source')
ax2.imshow(reference)
ax2.set_title('Reference')
ax3.imshow(matched)
ax3.set_title('Matched')
plt.tight_layout()
plt.show()

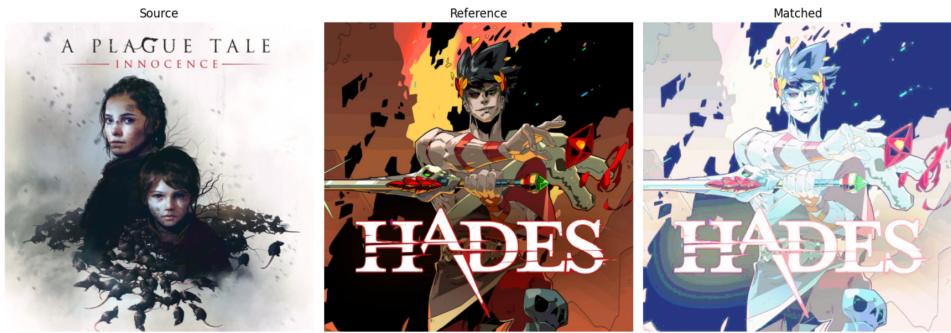
# histogram matching implementation
path = r'E:\University\Term7\FCV\Homeworks\HW3\Images'
source = plt.imread(os.path.join(path, 'A Plague Tale.jpg'))
reference = plt.imread(os.path.join(path, 'Hades.jpg'))
matched = hist_matching(reference, source)
fig, (ax1, ax2, ax3) = plt.subplots(nrows=1, ncols=3, figsize=(14, 14),
                                    sharex=True, sharey=True)
for aa in (ax1, ax2, ax3):
    aa.set_axis_off()
ax1.imshow(source)
ax1.set_title('Source')
ax2.imshow(reference)
ax2.set_title('Reference')
ax3.imshow(matched)
ax3.set_title(['Matched'])
plt.tight_layout()
plt.show()

```

کدهای مربوط به این بخش



نتیجه حاصل شده با استفاده از تابع آماده



نتیجه حاصل شده با استفاده از پیاده‌سازی تابع تطبیق هیستوگرام