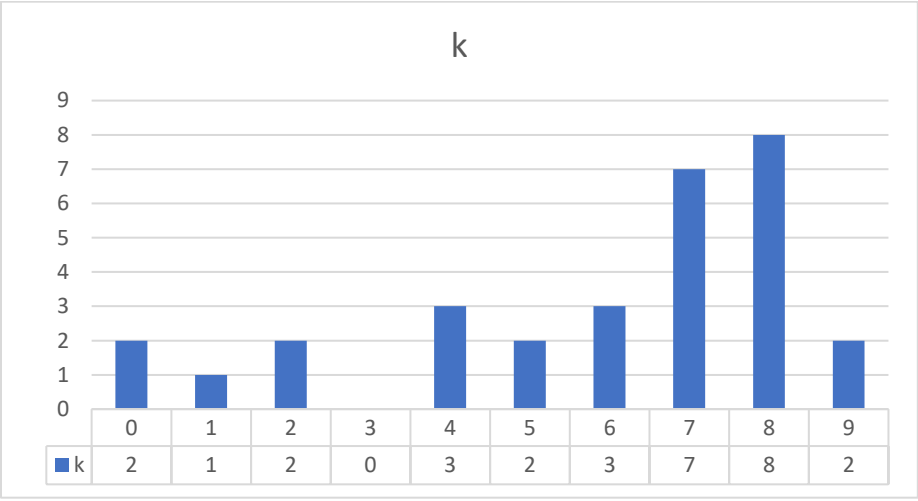
**Computer Vision**
**Dr. Mohammadi**
**Fall 2022**
**Hoorieh Sabzevari - 98412004**
**HW3**

۱.

| | | | | | |
|---|---|---|---|---|---|
| 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 |

نمودار هیستوگرام تصویر:




پیکسل‌های تصویر ما پس از مرتب‌سازی به صورت زیر قرار می‌گیرند:

[0, 0, 1, 2, 2, 4, 4, 4, …, 8, 8, 9, 9]

چون ۳۰ پیکسل داریم، پس ۱۰٪ آن معادل ۳ پیکسل خواهد بود و باید ۳ پیکسل از ابتدا و انتهای دنباله‌ی بالا حذف کنیم. لذا دنباله به صورت زیر خواهد بود:

[2, 2, 4, 4, 4, …, 8]

فرمول کشش هیستوگرام به صورت زیر است که $f_{10}=2$ و $f_{90}=8$ بدست آمد.

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

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

$g(1) = -1.5 \Rightarrow 0,\ g(2) = 0,\ g(3) = 1.5,\ g(4) = 3,\ g(5) = 4.5,\ g(6) = 6$

$g(7) = 7.5\ ,\ g(8) = 9 = g(9)$

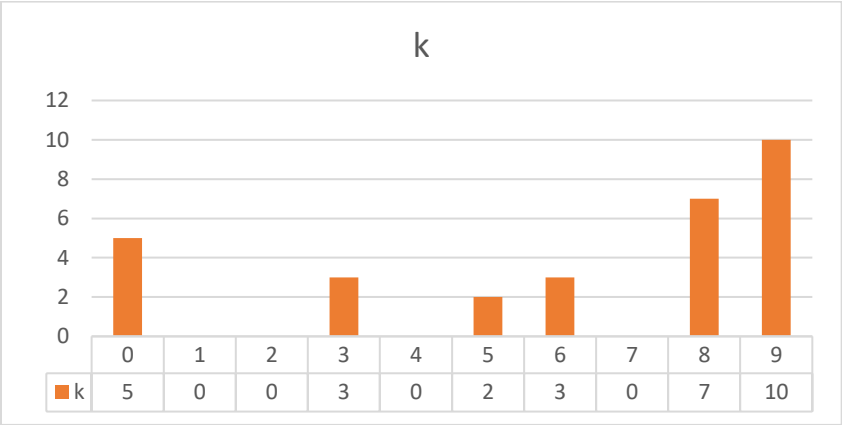
تصویر پس از برش و کشش به صورت زیر خواهد بود:

| | | | | | |
|---|---|---|---|---|---|
| ۷.۵ | ۷.۵ | ۹ | ۹ | ۹ | ۹ |
| ۰ | ۰ | ۳ | ۳ | ۳ | ۹ |
| ۷.۵ | ۰ | ۴.۵ | ۴.۵ | ۰ | ۹ |
| ۹ | ۰ | ۶ | ۹ | ۹ | ۷.۵ |
| ۹ | ۷.۵ | ۶ | ۶ | ۷.۵ | ۷.۵ |

سپس مقادیر را رند می‌کنیم:

| | | | | | |
|---|---|---|---|---|---|
| ۸ | ۸ | ۹ | ۹ | ۹ | ۹ |
| ۰ | ۰ | ۳ | ۳ | ۳ | ۹ |
| ۸ | ۰ | ۵ | ۵ | ۰ | ۹ |
| ۹ | ۰ | ۶ | ۹ | ۹ | ۸ |
| ۹ | ۸ | ۶ | ۶ | ۸ | ۸ |

هیستوگرام جدید به صورت زیر خواهد بود:




حال برای عملیات متعادل‌سازی ابتدا توزیع تجمعی را پیدا می‌کنیم:

| $k$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| $n_k$ | 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 |

برای تبدیل به احتمال توزیع تجمعی را تقسیم بر تعداد پیکسل‌ها می‌کنیم تا نرمال شود:

| $k$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| $n_k$ | 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 |
| $\sum_{j=0}^{k} \frac{n_j}{n}$ | $\frac{5}{30}$ | $\frac{5}{30}$ | $\frac{5}{30}$ | $\frac{8}{30}$ | $\frac{8}{30}$ | $\frac{10}{30}$ | $\frac{13}{30}$ | $\frac{13}{30}$ | $\frac{20}{30}$ | 1 |

حال باید عبارت را در ضریب $L-1$ ضرب کنیم. در اینجا $L$ میزان روشنایی ما یعنی ۱۰ است.

| $k$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| $n_k$ | 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 |

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

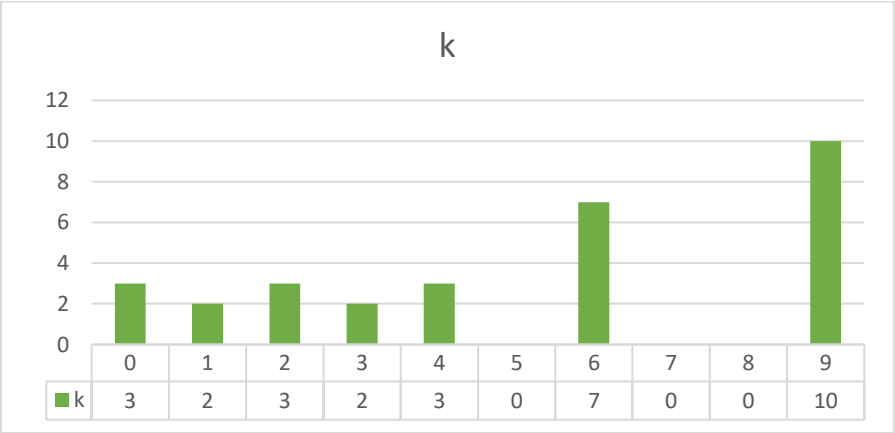
سپس مقادیر سطر آخر را رند می‌کنیم.

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

تصویر جدید به صورت زیر خواهد بود:

| | | | | | |
|---|---|---|---|---|---|
| ۴ | ۴ | ۶ | ۶ | ۶ | ۶ |
| ۲ | ۲ | ۲ | ۲ | ۲ | ۶ |
| ۴ | ۲ | ۳ | ۳ | ۲ | ۶ |
| ۶ | ۲ | ۳ | ۹ | ۹ | ۴ |
| ۶ | ۴ | ۴ | ۴ | ۴ | ۴ |

هیستوگرام نهایی:



۲.

الف) ابتدا مراحل متعادل‌سازی هیستوگرام را در کد پیاده‌سازی می‌کنیم. سپس از تابع آماده‌ی هیستوگرام در کتابخانه‌ی opencv استفاده می‌کنیم و خروجی را با بخش قبل مقایسه می‌کنیم.

کد خود:

```
def hist_equ(image):
    '''
    input:
    image (ndarray): input image
    output:
    output_image (ndarray): enhanced image
    '''

    #############
    # Your code
    # Start
    w ,h = image.shape # width and height
    image_size = w*h # image size
    histogram = np.zeros(256) # histogram initializing
    cdf=[]
    normalized =[]
    L = 256

    for r in range(w):

        for c in range(h):
            histogram[image[r][c]] += 1

    cdf.append(histogram[0])
    index = 0
    for i in range(L):
        cdf.append(cdf[index] + histogram[i])
        index += 1

    l = len(cdf)
    for i in range(l):
        normalized.append(cdf[i]*(L-1)/image_size)

    for r in range(w):
      for c in range(h):
        image[r][c]= normalized[image[r][c]]
```

```
    return image
```

خروجی کد خود:


کد تابع opencv :

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

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

res = np.hstack((img, equ)) #stacking images side-by-side

plt.figure(figsize=(16, 16))
plt.imshow(res, cmap='gray')
```

خروجی تابع opencv :

همانطور که مشخص است در هر دو روش کیفیت تصویر بهبود یافته و کنتراست تصویر نیز افزایش یافته است و فرق زیادی بین خروجی تابع خودمان با تابع آماده‌ی opencv نیست.

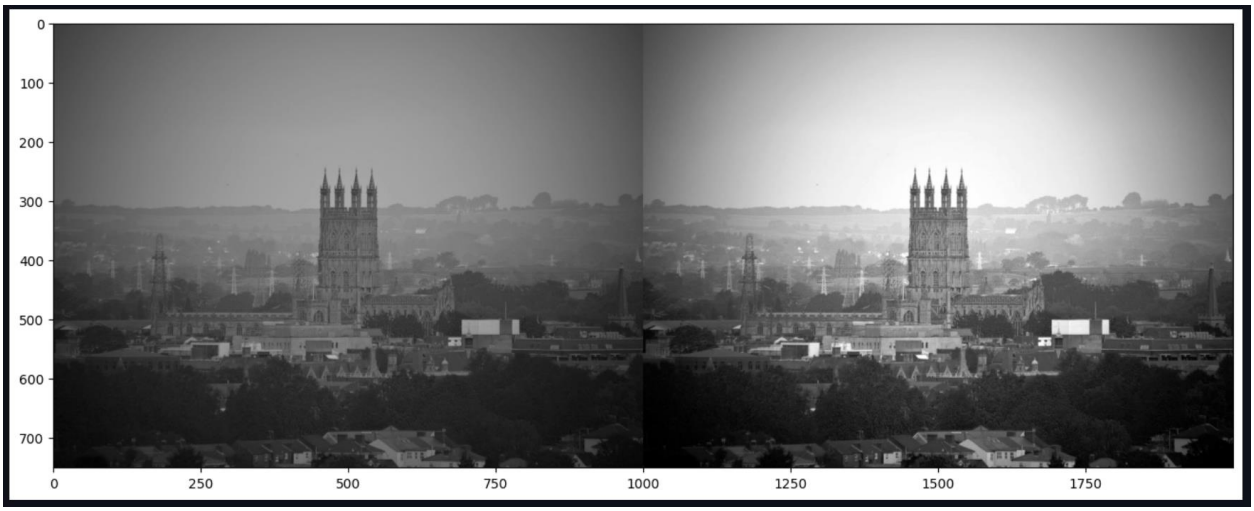
ب) در این سوال از تابع آماده‌ی opencv استفاده می‌کنیم. دو پارامتر clipLimit و tileGridSize را باید به تابع بدهیم. clipLimit آستانه‌ی محدود کردن کنتراست را تعیین می‌کند و gridSize سایز شبکه را مشخص می‌کند که چه تعداد سطر و ستون داشته باشد.

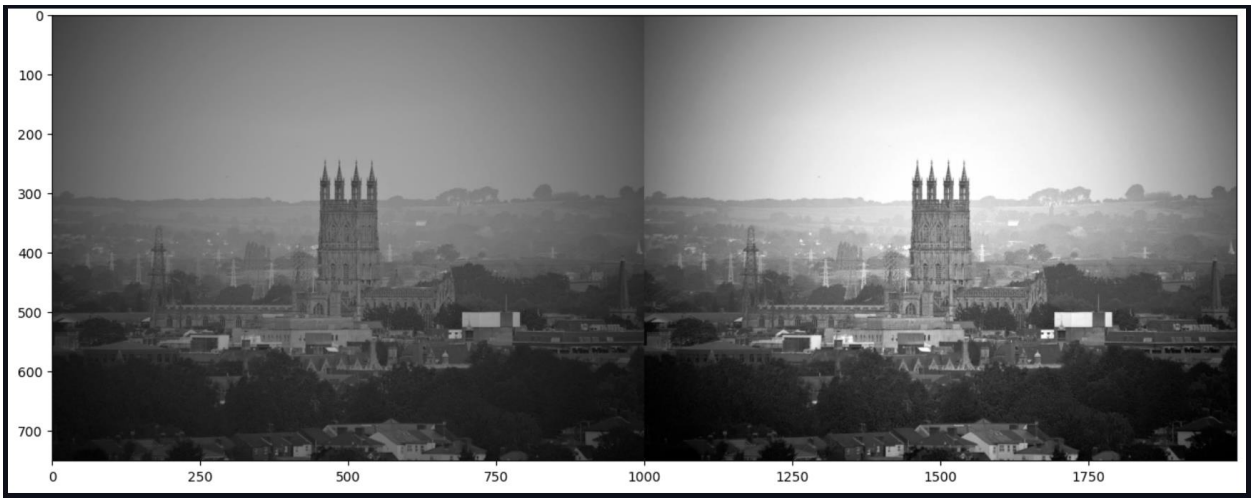
```python
def CLAHE(image, gridSize, clip_limit):
    '''
    inputs:
    image (ndarray): input image
    gridSize (tuple): window size
    clip_limit (int): threshold for contrast limiting
    output:
    output_image (ndarray): improved image
    '''

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

    # End

    return output_image
```

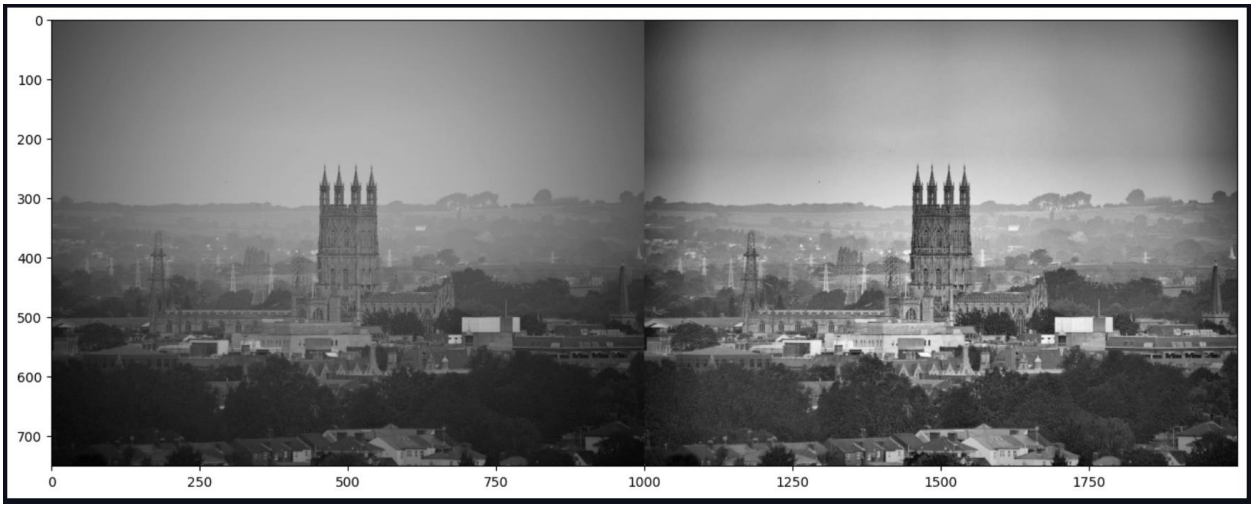
خروجی:

خروجی این تابع بسته به پارامترهای ورودی‌های آن است. هر چه سایز شبکه بزرگتر باشد تصویر شارپ‌تر و با کنتراست بالاتری است. همچنین انتخاب آستانه‌ی مناسب نیز از اهمیت زیادی برخورددار است. لذا با توجه به این دو پارامتر می‌توان تصویر بهتری نسبت به بخش قبلی داشت و برای قسمت‌های مختلف تصویر هیستوگرام اختصاصی محاسبه شده و از آن برای ارتقاء کنتراست تصویر استفاده می‌شود. برای مثال در تصویر خروجی این بخش درخت‌ها نیز با جزئیات بیشتری قابل مشاهده هستند و کنتراست بالاتری دارند، در حالی که تصویر خروجی بخش اول جزئیات درخت‌ها و گیاهان به این وضوح قابل‌مشاهده نیست.

پ) خروجی‌ها بترتیب:






همانطور که می‌بینیم، با روش CLAHE تصویر با کنتراست بیشتر و جزئیات بهتری خواهیم داشت. البته سایه‌ی شبکات روی تصویر کمی مشخص است.

ت) متعادل‌سازی هیستوگرام برای بهبود کنتراست هر تصویر است یعنی قسمت تاریک‌تر را تیره‌تر و قسمت روشن را روشن‌تر می‌کند. در یک تصویر خاکستری هر پیکسل با مقدار روشنایی نشان داده می‌شود. اما برای یک تصویر رنگی کار نمی‌کند. زیرا هر کانال از G, R, B نشان-دهنده‌ی شدت رنگ مربوطه است نه روشنایی تصویر، به همین علت اجرای این روش برای هر کدام از کانال‌ها نیز راه مناسبی نیست. روش بهتر این است که روشنایی تصویر را از رنگ جدا کنیم و سپس متعادل‌سازی هیستوگرام را اعمال کنیم. می‌توان از تبدیل فضای رنگی RGB به فضاهای رنگی دیگری مانند YCbCr استفاده کنیم و پس از اجرای HE در کانال Y که برای روشنایی است مجددا به فضای رنگی RGB تبدیل نماییم.(منبع)

(بخشی از سوال با آقای امیرحسین سماوات حل شده است.)

۳. تطبیق هیستوگرام را می توان به بهترین شکل به عنوان یک تغییر در نظر گرفت. هدف ما گرفتن یک تصویر ورودی(منبع) و آپدیت شدت پیکسل‌های آن است به طوری که توزیع هیستوگرام تصویر ورودی با توزیع تصویر مرجع مطابقت داشته باشد. در حالی که محتوای واقعی تصویر ورودی تغییر نمی‌کند، توزیع پیکسل‌ها تغییر می کند، در نتیجه روشنایی و کنتراست تصویر ورودی بر اساس توزیع تصویر مرجع تنظیم می‌شود.

کد:

```
# source = https://pyimagesearch.com/2021/02/08/histogram-matching-with-
opencv-scikit-image-and-python/
source = plt.imread('A Plague Tale.jpg')
reference = plt.imread('Hades.jpg')

### YOUR CODE ###
# START
multi = True if source.shape[-1] > 1 else False
matched = match_histograms(source, reference, multichannel = multi)
# 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()
```

خروجی کد:


در اینجا می‌بینیم که هیستوگرام تصویر ورودی بر هیستوگرام تصویر مرجع منطبق شده و کنتراست و روشنایی آن نیز طبق تصویر مرجع تنظیم شده است.

ب) در این قسمت با استفاده از کتابخانه‌ی numpy ابتدا هیستوگرام و توزیع تجمعی هر دو تصویر را محاسبه می‌کنیم. سپس مقادیر را نرمالایز می‌کنیم. حال باید پیکسل‌ها را طبق توزیع تجمعی جابجا کنیم. یعنی مثلا می‌بینیم روشنایی صفر در تصویر اول چه توزیع تجمعی‌ای دارد، سپس در توزیع تجمعی تصویر دوم می‌بینیم که کدام میزان روشنایی همان توزیع تجمعی را دارد. در انتها در تصویر اول هر چه از میزان روشنایی صفر داریم را با میزان روشنایی‌ای که از تصویر دوم پیدا کرده‌ایم جابجا می‌کنیم.

کد:

```
#https://vzaguskin.github.io/histmatching1/
def hist_matching(imsrc, imtint):
  '''
  inputs:
    src_image (ndarray): source image
    ref_image (ndarray): reference image
  output:
    output_image (ndarray): transformed image
  '''

  ### YOUR CODE ###
```

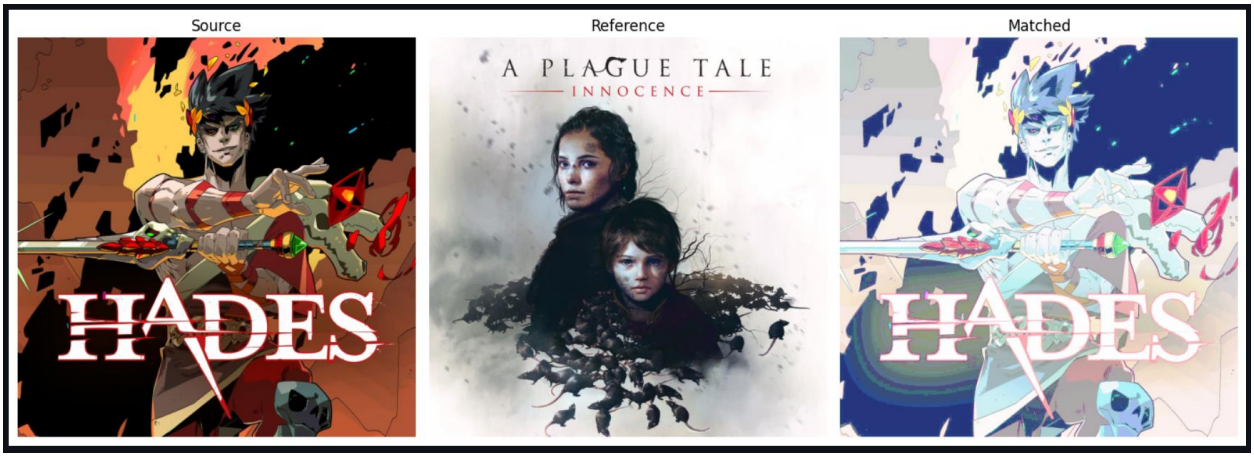
```python
# START
L = 255

output_image = imsrc.copy()

for d in range(imsrc.shape[2]):
    src_hist, bins = np.histogram(imsrc[:,:,d].flatten(), L, normed=True)
    ref_hist, bins = np.histogram(imtint[:,:,d].flatten(), L, normed=True)

    cdf_src = src_hist.cumsum() #source_cdf
    cdf_src = ((L-1) * cdf_src / cdf_src[-1]).astype(np.uint8) #normalize

    cdf_ref = ref_hist.cumsum() #ref_cdf
    cdf_ref = ((L-1) * cdf_ref / cdf_ref[-1]).astype(np.uint8) #normalize

    img2 = np.interp(imsrc[:,:,d].flatten(), bins[:-1], cdf_src)
    img3 = np.interp(img2, cdf_ref, bins[:-1])
    output_image[:,:,d] = img3.reshape((imsrc.shape[0], imsrc.shape[1]))

# END

return output_image
```

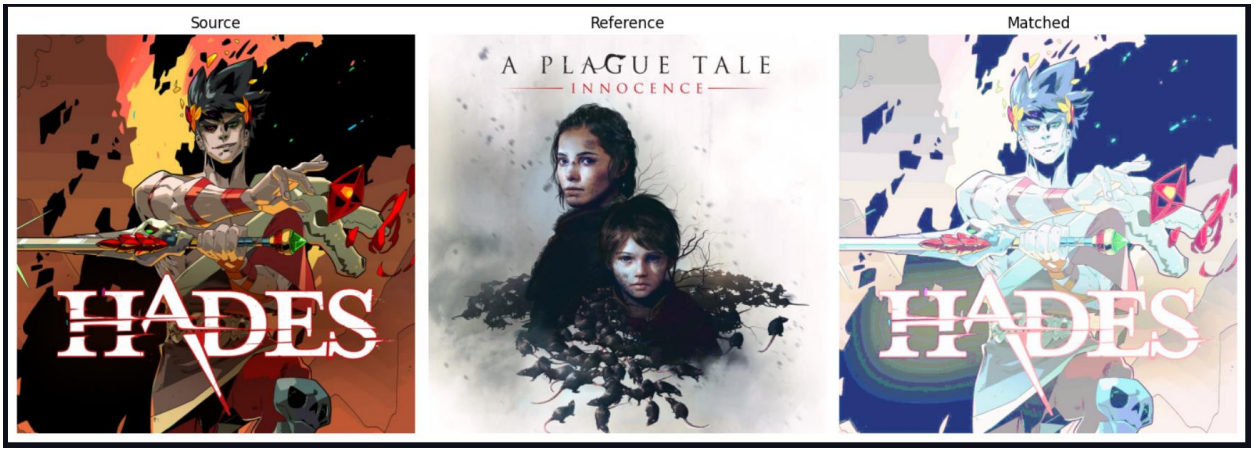
خروجی:


همانطور که واضح است خروجی مانند خروجی قسمت قبل شده و روشنایی تصویر منبع نیز طبق تصویر مرجع تنظیم شده است.

پ) در این قسمت فقط جای تصاویر مرجع و منبع را جابجا می‌کنیم و دوباره تابع آماده و تابعی که خودمان نوشتیم را فراخوانی می‌کنیم.

خروجی‌ها به ترتیب:




همانطور که واضح است میزان روشنایی تصویر منبع با میزان روشنایی تصویر مرجع تنظیم شده است و به رنگ‌های سرد تغییر یافته است. همچنین توابع نیز تفاوت محسوسی ندارند.