# Contrast adjustment

**رضا عباس‌زاده**

**اطلاعات گزارش**

تاریخ: ۱۴۰۰/۱/۲۰

**واژگان کلیدی:**
هیستوگرام
متعادل‌سازی
متعادل‌سازی محلی
پردازش نقطه‌ای
نرمال سازی

**چکیده**

در این تمرین با نحوه محاسبه و پیاده‌سازی هیستوگرام یک تصویر آشنا می‌شویم. کنتراست یک تصویر را کاهش می‌دهیم و سپس سعی می‌کنیم با استفاده از متعادل‌سازی سراسری و محلی تصویر اصلی را بازسازی کنیم. همچنین تاثیر توابع تبدیل نقطه‌ای نمایی بر کنتراست تصویر را بررسی می‌کنیم. در تمرین سوم تفاوت‌های دو تابع متعادل‌سازی در متلب را مشخص می‌کنیم. در انتها نیز نتایج انتخاب اندازه پنجره در متعادل‌سازی محلی را با یکدیگر مقایسه می‌کنیم.

## ۱-مقدمه

نوشتار حاضر، نحوه به دست آوردن هیستوگرام یک تصویر را شرح می‌دهد. هیستوگرام تعداد پیکسل‌هایی که دارای سطح خاکستری یکسان هستند را به ازای همه‌ی سطوح مشخص می‌کند. در این تمرین سطوح خاکستری در ۸ بیت ذخیره می‌شوند و ۲۵۶ سطح مختلف را شامل می‌شوند. در این تمرین سعی می‌کنیم با استفاده از هیستوگرام، توابع تبدیلی به دست آوریم که باعث افزایش کیفیت تصویر می‌شوند.

## ۲-توضیحات تکنیکال

### ۱-۱-۲-هیستوگرام

برای به دست آوردن هیستوگرام یک تصویر، روی تمام عناصر تصویر پیمایشی انجام می‌دهیم و به ازای هر عنصر که سطح خاکستری k دارد، در هسیتوگرام یک عدد سطح k اضافه می‌کنیم. نتیجه نهایی یک آرایه یک بعدی (کاهش بعد انجام می‌شود) است که تعداد پیکسل‌هایی هر سطح خاکستری را مشخص می‌کند.

#### ۱-۱-۱-۲-کاهش کنتراست

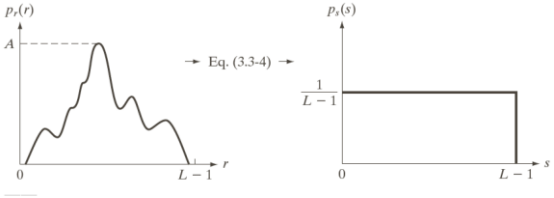
یک روش ساده برای کاهش کنتراست یک تصویر، تقسیم مقادیر پیکسل‌های آن بر یک عدد مثبت بزرگتر از یک است. در این تمرین تمام مقادیر بر عدد ۳ تقسیم می‌شوند و نتیجه نهایی با نام D ذخیر می‌شود.

#### ۲-۱-۱-۲-رسم نتایج

با استفاده از کتابخانه matplotlib.pyplot تصاویر خروجی و هستوگرام آن‌ها را در غالب یک تصویر ذخیر می‌کنیم.

**۳-۱-۱-۲-متعادل‌سازی سراسری**

هدف از متعادل سازی نزدیک کردن احتمال وقوع سطوح خاکستری مختلف به یکدیگر می‌باشد. در واقع می‌خواهیم هستوگرام تصویر به این شکل درآید:




مراحل متعادل سازی سراسری ترتیب زیر می‌باشد:

**۱.** محاسبه هیستوگرام

**۲.** محاسبه تابع چگالی احتمال PDF. این تابع احتمال وقوع سطح خاکستری k در یک تصویر را با استفاده از هیستوگرام مشخص می‌کند:

$$\text{PDF(k)} = \frac{n_k}{N*M}$$

در این رابطه $n_k$ مقدار هیستوگرام برای سطح خاکستری k است و N و M ابعاد تصویر هستند.

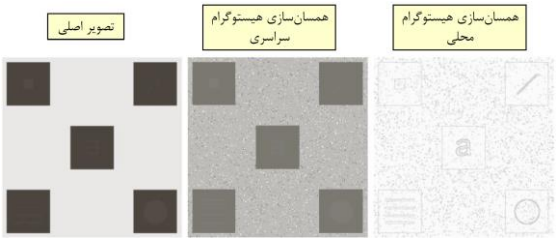
**۳.** محاسبه تابع توزیع احتمال CDF:

$$\text{CDF(k)} = \sum_{j=0}^{k} PDF(j)$$

**۴.** محاسبه حاصلضرب CDF هر سطح خاکستری در بزرگترین سطح خاکستری و گرد کردن نتیجه به نزدیکترین عدد صحیح.

**۴-۱-۱-۲-متعادل‌سازی محلی**

در متعادل‌سازی سراسری تابع تبدیلی که به دست می‌آید به توجه به تمام تصویر می‌باشد و این باعث می‌شود که جزییاتی که در قسمت‌های کوچکتر تصویر وجود دارد به سادگی قابل مشاهده نشود. به مثال زیر توجه کنید:




برای رفع این مشکل از متعادل‌سازی محلی استفاده می‌کنیم.

در این روش پنجره‌ای با ابعاد کوچکتر از تصویر اصلی در نظر می‌گیریم و آن را روی تصویر حرکت می‌دهیم. پنجره روی هر قسمت که قرار گرفت، با استفاده از تابع متعادل‌سازی سراسری، فقط همان قسمت را متعادل سازی می‌کند.

برای حرکت دادن پنجره، دو روش کلی وجود دارد: پوشا و غیرپوشا. در روش غیرپوشا پنجره هربار به اندازه یک پنجره کامل جابه‌جا می‌شود و هیچ اشتراکی با مکان قبلی پنجره ندارد. در این روش، به دلیل اینکه هر پنجره به صورت مجزا متعادل‌سازی شده است، باعث به وجود آمدن مربع‌های در تصویر می‌شود که هرکدام همه سطوح خاکستری ۰ تا ۲۵۵ را دارند. در روش پوشا مانند روش قبل متعادل‌سازی سراسری را انجام می‌دهیم با این تفاوت که نتایج را فقط روی پنجره‌ای کوچکتر که داخل پنجره اصلی وجود دارد اعمال می‌کنیم. به این ترتیب بازه‌ی بزرگتری از پیکسل‌ها در محاسبات تاثیر می‌گذارند و تفاوت نتایج محاسبات در پنجره‌های مجاور کمتر می‌شود و حاشیه‌های مربعی کمتری ظاهر می‌شوند. لازم به ذکر است که جابه‌جایی پنجره در این روش به اندازه پنجره داخلی می‌باشد. همچنین برای اینکه عمل متعادل‌سازی روی پیکسل‌های لبه‌ی تصویر هم انجام شود، لازم است فاصله padding به اندازه تفاوت ابعاد پنجره اصلی با پنجره داخلی در تصویر ایجاد شود. این فاصله با مقدار ۱۲۸ پر می‌شود تا باعث تیره یا روشن شدن لبه‌ها نشود.

**۶-۱-۱-۲-توابع تبدیل نقطه‌ای**

در این تمرین توابع نمایی لگاریتم، معکوس لگاریتم، توان و ریشه روی تمام پیکسل ها اعمال می‌شود و سپس نتایج با استفاده از رابطه زیر نرمال‌سازی می‌شوند:

$$I_N = \frac{(I - \text{Min})}{\text{Max} - \text{Min}} \text{X } 255$$

### ۲-۱-۲-متعادل‌سازی سراسری

در این تمرین با توجه به توضیحات بخش ۲-۱-۱-۳ عملیات متعادل‌سازی روی تصویر Camera man انجام می‌شود و نتایج به همراه هیستوگرام ذخیره می‌شود.

### ۳-۱-۲-تفاوت histeq و imadjust

تابع histeq دقیقا مشابه با متعادل‌سازی سراسری که در بخش ۲-۱-۱-۳ توضیح داده شده است، تلاش می‌کند هیستوگرام تصویر را یکنواخت کند. مانند تابع equalizeHist در کتابخانه opencv. این تبدیل لزوما خطی نخواهد بود.

تابع imadjust مقادیر پیکسل های عکس ورودی را به بازه‌ای جدید نگاشت می‌کند. به عنوان مثال تصویری داریم که مقادیر آن بیت ۱۰۰ تا ۲۰۰ قرار دارند. با استفاده از این تابع می‌توان با رعایت نسبت و به صورت خطی، مقادیر را به ۰ تا ۲۵۵ نگاشت کرد. این کار باعث افزایش کنتراست تصویر خواهد شد.

### ۱-۲-۲-متعادل‌سازی محلی

در این تمرین با توجه به توضیحات بخش ۲-۱-۱-۴ متعادل سازی را روی تصاویر HE1, HE2, HE3, HE4 انجام می‌دهیم. پنجره هایی با اندازه‌های متفاوت به صورت زیر را برای هر یک از تصاویر امتحان می‌کنیم و نتیجه را ذخیره می‌کنیم:

پنجره اصلی: ۲۰۰ پنجره داخلی: ۲۰

پنجره اصلی: ۱۰۰ پنجره داخلی: ۱۰۰

پنجره اصلی: ۱۰۰ پنجره داخلی: ۱۰

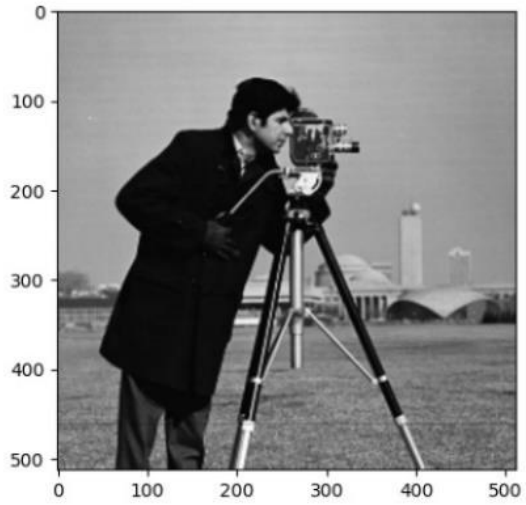
پنجره اصلی: ۵۰ پنجره داخلی: ۸

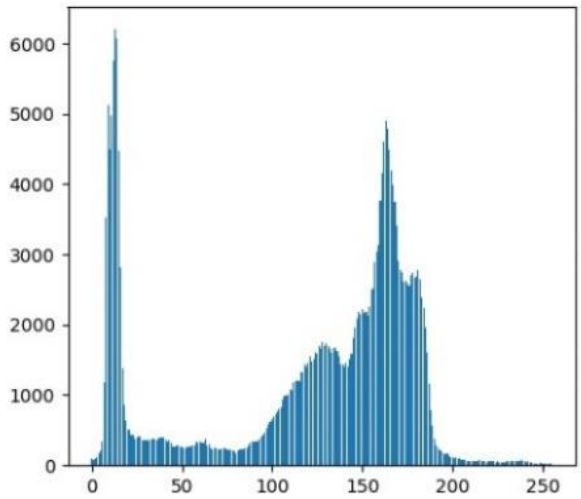
پنجره اصلی: ۳۰ پنجره داخلی: ۶

## ۳-شرح نتایج و نتیجه‌گیری

### ۱-۱-۲

تصویر Camera man به همراه هیستوگرام آن:

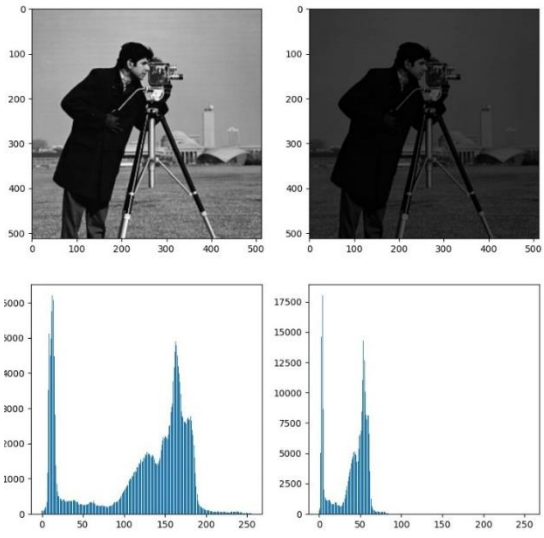


**۲-۱-۱-۱**

کاهش کنتراست تصویر با تقسیم مقادیر بر عدد ۳ (تصویر D):

**۲-۱-۱-۲**



با مقایسه این دو هیستوگرام می‌بینیم که با تقسیم مقادیر بر عدد ۳ کنتراست تصویر کاهش یافته است و تصویری تیره به وجود آمده است.

**۲-۱-۱-۳**

بعد از انجام متعادل‌سازی سراسری روی تصویر D (تصویر H):

**۲-۱-۱-۴**

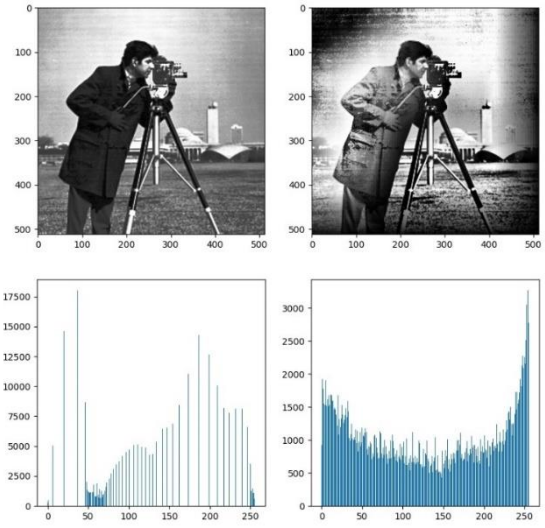
بعد از انجام متعادل‌سازی محلی با اندازه پنجره ۱۰۰ و پنجره داخلی ۶ روی تصویر D (تصویر L):

در این تصویر مشاهده می‌شود که بعد از اعمال متعادل-سازی محلی، جزییات چمن‌ها که در تصویر اصلی خیلی واضح نبوند، مشخص می‌شود. اما در بقیه قسمت‌های تصویر باعث ایجاد نویز به دلیل جزییات زیاد شده است.

با تغییر اندازه پنجره به ۱۵۰ (پنجره اصلی) و ۳۰ (پنجره داخلی)، به دلیل کمتر شدن جزییات نویز کمتری داریم اما مرزهای مربعی در تصویر به وجود می‌آیند:

**۲-۱-۱-۵**

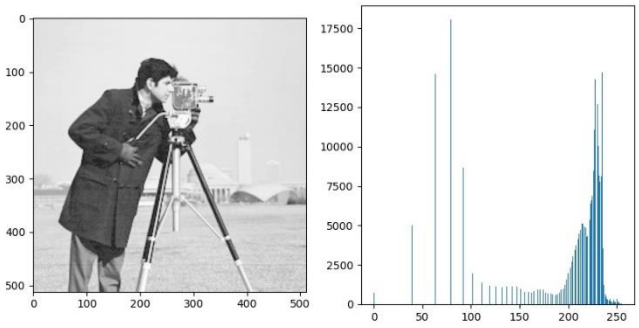
در متعادل سازی محلی از پنجره اول استفاده شده است (۱۰۰ و ۶):



مشاهده می‌کنیم که در متعادل‌سازی محلی به دلیل اینکه در هرقسمت کوچک تصویر همه مقادیر ۰ تا ۲۵۵ را داریم، هیستوگرام نهایی متعادل‌تر شده است. همچنین در این روش سطوح خاکستری پیوسته‌تری داریم و در روش سراسری مقادیر گسسته‌تر هستند.

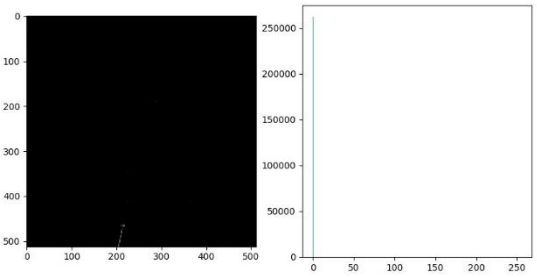
**۲-۱-۱-۶**

**تبدیل لگاریتمی در مبنای ۲:**



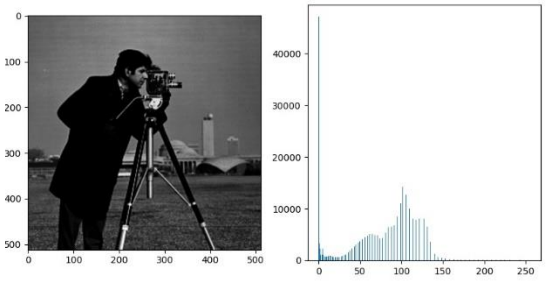
این تبدیل کنتراست تصویر در نواحی تیره را افزایش می‌دهد و در نواحی روشن کاهش می‌دهد.

تبدیل معکوس لگاریتمی در مبنای ۲:



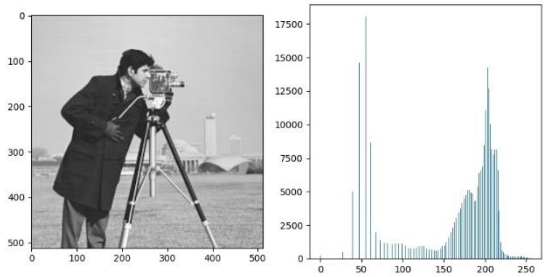
این تبدیل کنتراست تصویر در نواحی تیره را کاهش و در نواحی روشن افزایش می‌دهد. به دلیل اینکه مقادیر در تبدیل معکوس لگاریتمی بسیار گسترده می‌شوند و نرمال‌سازی ما به صورت خطی می‌باشد نتیجه اکثر محاسبات عددی بسیار کوچک نزدیک به صفر می‌شود و نتیجه نامطلوبی ایجاد می‌کند.

**تابع تبدیل توان ۲:**



این تبدیل کنتراست در نواحی روشن را افزایش و در نواحی تیره را کاهش می‌دهد.

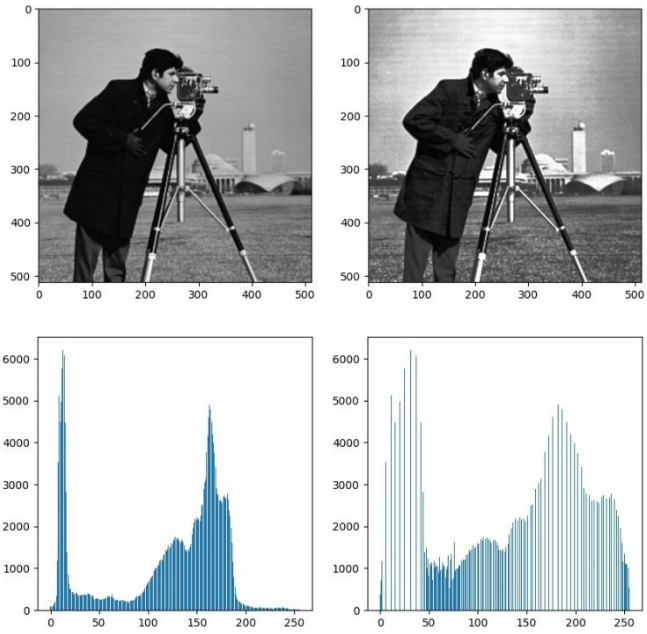
**تابع ریشه ۲:**



این تابع برخلاف تابع توان عمل می‌کند.

**۲-۱-۲**

تصویر اصلی در کنار تصویر متعادل‌شده:

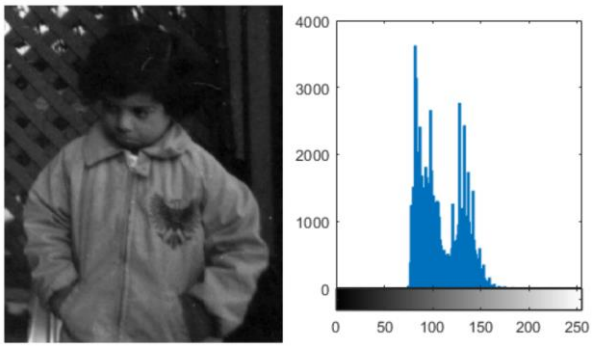


می‌بینیم که در تصویر اصلی پیکسل‌های روشن کمی وجود داشته‌اند و تمرکز اصلی روی پیکسل‌های تیره و میانه بوده است که با انجام متعادل‌سازی این مشکلات برطرف شده‌اند.

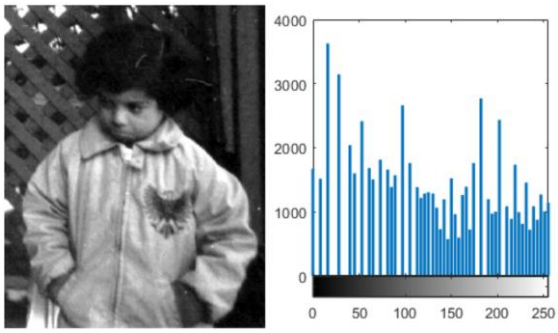
**۲-۱-۳**

خروجی تابع histeq دقیقا برابر با خروجی بخش قبل ۲-۲-۱ است.

تصویر ورودی تابع imadust:



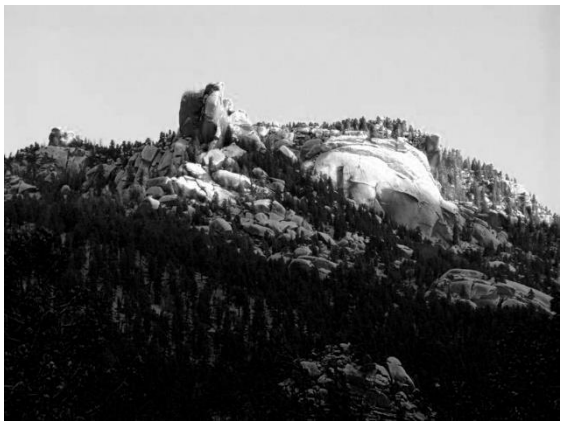
تصویر خروجی:




Main window: 100  Inner window: 100

**۲-۲-۱**

**تصویر HE1:**

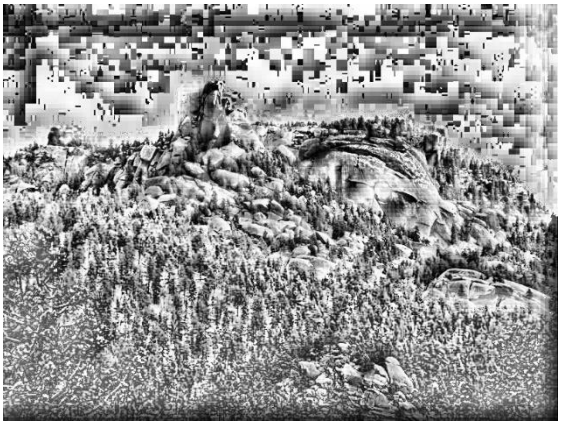


Main window: 100  Inner window: 10

Main window: 50  Inner window: 8

Main window: 200  Inner window: 20

Main window: 30  Inner window: 6

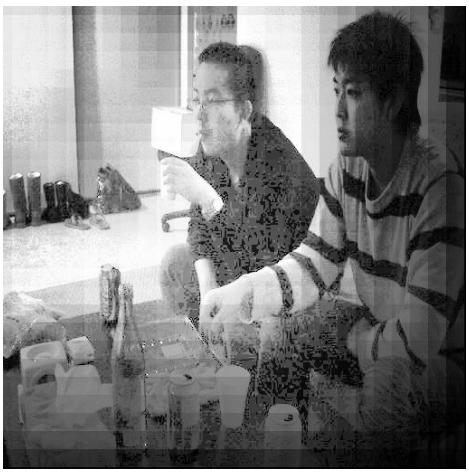


به دلیل وجود دو قسمت یکپارچه(آسمان) و پرجزییات (سنگ‌ها و درختان) انتخاب یک فیلتر مناسب بسیار دشوار خواهد بود.

وجود آسمان تقریبا یکدست در این تصویر باعث شده است که اگر اندازه تصویر را کوچک در نظر بگیریم، نویز شدیدی در آسمان به وجود آید.از طرفی برای واضح کردن جزییات سنگ‌ها نیاز داریم تا پنجره کوچکتری انتخاب کنیم. پنجره ۱۰-۱۰۰ جزییات درختان و سنگ‌ها را واضح‌تر می‌کند اما پنجره ۲۰۰-۲۰ برای آسمان گزینه بهتری است.

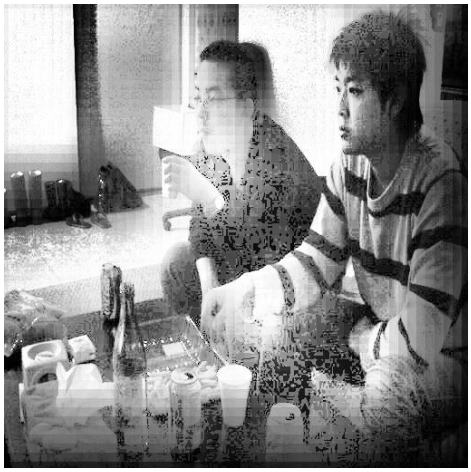
**تصویر HE2**:

Main window: 200  Inner window: 20

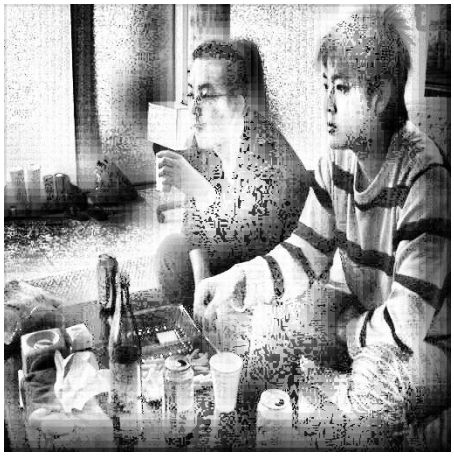


Main window: 100  Inner window: 100

Main window: 100  Inner window: 10



Main window: 50  Inner window: 8



Main window: 30  Inner window: 6

در این تصویر به دلیل اینکه جزییات کوچکی ندارد، اندازه پنجره بالا مناسب تر می‌باشد و پنجره ۱۰۰-۱۰ مناسب-ترین گزینه است.

**تصویر HE3**:

Main window: 200  Inner window: 20

Main window: 100  Inner window: 100

Main window: 100  Inner window: 10

Main window: 50  Inner window: 8

Main window: 30  Inner window: 6

در این تصویر برای تفکیک چهره‌هایی که دورتر از دوربین قرار دارند بهتر است پنجره‌ای کوچکتر انتخاب شود اما به دلیل نویز زیادی که ایجاد می‌کند و برای حفظ کیفیت کلی تصویر پنجره ۲۰۰-۲۰ بهترین عملکرد را داشته است.

**تصویر HE4**:

Main window: 200  Inner window: 20

Main window: 100  Inner window: 100

Main window: 100  Inner window: 10

Main window: 50  Inner window: 8

Main window: 30  Inner window: 6

در این تصویر برای اینکه جزییات ساختان‌های نزدیکتر واضح‌تر شوند، بهتر است اندازه فیلتر کوچک انتخاب شود مانند ۳۰-۶ . اما جزییات دورتر که در تصویر اصلی نیز به وضوح قابل مشاهده نیستند با پنجره‌های بزرگتر کیفیت بهتری خواهند داشت. بنابراین یک فیلتر با اندازه متوسط ۱۰-۱۰۰ می‌تواند هر دو مزیت را در بر داشته باشد.

## پیوست

تابع مربوط به محاسبه هیستوگرم:

```
def compute_histogram(img):
    histogram = np.zeros(256)
    for i in range(0, img.shape[0]):
        for j in range(0, img.shape[1]):
            histogram[img[i, j]] += 1
    return histogram
```

تابع محاسبه pdf و cdf یک تصویر:

```
def compute_pdf_cdf(img):
    histogram = compute_histogram(img)

    pdf = histogram / (img.shape[0] * img.shape[1])
    cdf = pdf
    for i in range(1, 256):
        cdf[i] = cdf[i] + cdf[i - 1]

    return pdf, cdf
```

تابع متعادل‌سازی سراسری تصویر:

```
def globalHistEq(img):
    pdf, cdf = compute_pdf_cdf(img)

    res = np.zeros(img.shape, np.uint8)
    for i in range(0, img.shape[0]):
        for j in range(0, img.shape[1]):
            res[i][j] = round(255 * cdf[img[i][j]])

    return res
```

تابع اضافه کردن padding به تصویر:

```
def add_padding(img, padSize, padValue = 128):
    padded = np.ones((img.shape[0] + padSize * 2, img.shape[1] + padSize *
2), np.uint8) * padValue
    padded[padSize:img.shape[0] + padSize, padSize:img.shape[1] + padSize]
= img
    return padded
```

تابع متعادل‌سازی محلی:

```
def localHistEq(img, mainWindow=45, innerWindow=15):
    windowsDiff = mainWindow - innerWindow
    padding = int(windowsDiff / 2)
    img = add_padding(img, padding)
    res = np.zeros((img.shape[0] + windowsDiff, img.shape[1] +
windowsDiff), np.uint8)
    for i in range(0, res.shape[0], innerWindow):
        if i + mainWindow > res.shape[0] - 1:
            i = res.shape[0] - mainWindow
        for j in range(0, res.shape[1], innerWindow):
            if j + mainWindow > res.shape[1] - 1:
                j = res.shape[1] - mainWindow
            pdf, cdf = compute_pdf_cdf(img[i:i + mainWindow, j:j +
mainWindow])
            for i2 in range(i + padding, i + mainWindow - padding):
                for j2 in range(j + padding, j + mainWindow - padding):
                    res[i2][j2] = round(255 * cdf[img[i2 - padding][j2 -
padding]])

    return res[windowsDiff:res.shape[0] - windowsDiff,
windowsDiff:res.shape[1] - windowsDiff]
```

تابع نرمال‌سازی:

```
def normalize(img):
    return (((img - img.min()).astype(float) / (img.max() - img.min())) *
255).astype(np.uint8)
```

کد تمرین ۲-۱-۱

```
import copy
import numpy as np
import cv2
import matplotlib.pyplot as plt

from common import compute_histogram, localHistEq, globalHistEq, normalize

img = cv2.imread('Camera Man.bmp', cv2.IMREAD_GRAYSCALE)
histogram = compute_histogram(img)
fig, axs = plt.subplots(2, 1, figsize=(5, 10))
axs[0].imshow(img, cmap='gray', aspect='equal')
axs[1].bar(range(0, 256), histogram)
fig.savefig('2.1.1.jpg')

# 2.1.1.1
D = (copy.deepcopy(img) / 3).astype(np.uint8)
cv2.imwrite('2.1.1.1-D.jpg', D)

# 2.1.1.2
histogramD = compute_histogram(D)
```

```
fig, axs = plt.subplots(2, 2, figsize=(10, 10))
axs[0][0].imshow(img, cmap='gray', vmin=0, vmax=255, aspect='equal')
axs[1][0].bar(range(0, 256), histogram)
axs[0][1].imshow(D, cmap='gray', vmin=0, vmax=255, aspect='equal')
axs[1][1].bar(range(0, 256), histogramD)
fig.savefig('2.1.1.2.jpg')

# 2.1.1.3
H = globalHistEq(D)
cv2.imwrite('2.1.1.3-H.jpg', H)

# 2.1.1.4
L = localHistEq(D, 100, 30)
cv2.imwrite('2.1.1.4-L.jpg', L)

# 2.1.1.5
histH = compute_histogram(H)
histL = compute_histogram(L)
fig, axs = plt.subplots(2, 2, figsize=(10, 10))
axs[0][0].imshow(H, cmap='gray', vmin=0, vmax=255, aspect='equal')
axs[1][0].bar(range(0, 256), histH)
axs[0][1].imshow(L, cmap='gray', vmin=0, vmax=255, aspect='equal')
axs[1][1].bar(range(0, 256), histL)
fig.savefig('2.1.1.5.jpg')

# 2.1.1.6
log = D.astype(float)
inv_log = D.astype(int)
power = D.astype(float)
root = D.astype(float)

log = normalize(np.log2(log, out=np.zeros_like(log), where=(log != 0)))
inv_log = normalize(np.exp(inv_log))
power = normalize(power ** 2)
root = normalize(np.sqrt(root))

fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].imshow(log, cmap='gray', vmin=0, vmax=255, aspect='equal')
axs[1].bar(range(0, 256), compute_histogram(log))
fig.savefig('2.1.1.6-log.jpg')

fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].imshow(inv_log, cmap='gray', vmin=0, vmax=255, aspect='equal')
axs[1].bar(range(0, 256), compute_histogram(inv_log))
fig.savefig('2.1.1.6-inv-log.jpg')

fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].imshow(power, cmap='gray', vmin=0, vmax=255, aspect='equal')
axs[1].bar(range(0, 256), compute_histogram(power))
fig.savefig('2.1.1.6-power.jpg')

fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].imshow(root, cmap='gray', vmin=0, vmax=255, aspect='equal')
axs[1].bar(range(0, 256), compute_histogram(root))
fig.savefig('2.1.1.6-root.jpg')
```

کد تمرین ۲-۱-۲

```
import cv2
import matplotlib.pyplot as plt
from common import compute_histogram, globalHistEq

img = cv2.imread('Camera Man.bmp', cv2.IMREAD_GRAYSCALE)

histogram = compute_histogram(img)

eq_img = globalHistEq(img)
eq_histogram = compute_histogram(eq_img)

fig, axs = plt.subplots(2, 2, figsize=(10, 10))
axs[0][0].imshow(img, cmap='gray', vmin=0, vmax=255, aspect='equal')
axs[1][0].bar(range(0, 256), histogram)
axs[0][1].imshow(eq_img, cmap='gray', vmin=0, vmax=255, aspect='equal')
axs[1][1].bar(range(0, 256), eq_histogram)
fig.savefig('2.1.2.jpg')
```

کد تمرین ۲-۲-۱

```
import cv2
from common import localHistEq

# [mainWindow, innerWindow]
window_sizes = [
    [200, 20],
    [100, 100],
    [100, 10],
    [50, 8],
    [30, 6],
]

for i in range(1, 5):
    img = cv2.imread('HE%i.jpg' % i, cv2.IMREAD_GRAYSCALE)
    for window in window_sizes:
        local_eq = localHistEq(img, window[0], window[1])
        cv2.imwrite("2.2.1-HE%i-%i-%i.jpg" % (i, window[0], window[1]),
local_eq)
        print("2.2.1-HE%i-%i-%i.jpg saved" % (i, window[0], window[1]))
```