Image Fundamentals

زهرا نیازی

اطلاعات گزارش

تاریخ: 1401/09/01

واژگان کلیدی:
مقاله
شیوه نامه تدوین
نویسنده
چاپ
شکل
جدول
فرمول
نتایج

چکیده

در این گزارش، مفاهیم هیستوگرام، چندی سازی (quantization)، downsample و upsample کردن تصاویر مطرح شده است. تاثیر تعداد سطوح خاکستری بر روی کیفیت تصویر و مراحل روش های مختلف درونیابی تصویر بررسی شده است.

1-مقدمه

هنگام کار با تصاویر دیجیتال، عملیات های هندسی مختلفی می توان روی تصویر انجام داد که تا تغییرات مفیدی روی تصویر اعمال شوند. در انجام عملیات های لازم برای اعمال این تغییرات، یکی از نکات مهم از دست ندادن بخش های مهمی از عکس می باشد که آن را برای چشم ما قابل تمیز نکند. روش های مختلفی برای انجام این عملیات ها وجود دارند. قصد داریم میزان دقت تعدادی از آنها را بررسی کنیم.

zahra.niazi2001@gmail.com

2-شرح تکنیکال

2-1-چندی سازی و درون یابی

2-1-1 چندی سازی

در این قسمت از ما خواسته شده تصویر Elaine را به تعداد (4، 8، 16، 32، 64، 128) سطوح خاکستری کوانتیزه کنیم و برای هر حالت، تصویر را در کنار هیستوگرام آن نمایش دهیم. همچنان خواسته شده این کار برای هیستوگرام equalized شده نیز تکرار شود.

برای انجام این کار باید مقادیر تک تک پیکسل های عکس را از بازه (0-255) به بازه ای بین 0 تا تعداد سطوح خاکستری مشخص شده ببریم.

هیستوگرام اطلاعات تصویر دو بعدی را به یک بعد کاهش می‌دهد. همچنین وابسته به تعداد سطوح خاکستری تصویر می باشد که برای مقایسه ما مفید است. روش به دست آوردن هیستوگرام هر تصویر به این صورت است که باید PDF(probability density function) تصویر محاسبه شود تا احتمال رخداد هرکدام از سطوح خاکستری را بدانیم. سپس این مقادیر را تقسیم بر کل سایز تصویر میکنیم.

برای به دست آوردن هیستوگرام متعادل شده نیز باید ابتدا CDF(cumulative density function) تصویر را به دست آورده و آن را ضرب در بیشترین مقدار سطح خاکستری در تصویر کنیم. جمع تک تک pdf ها مقدار cdf را مشخص میکند.

این روش معمولا کنتراست کلی بسیاری از تصاویر را افزایش می دهد، به خصوص زمانی که تصویر با محدوده کوچکی از مقادیر نمایش داده می شود. با اعمال این تغییر، مقادیر را می توان بهتر و به طور مساوی بر روی هیستوگرام با استفاده از طیف کامل مقادیر توزیع کرد. این کار باعث می شود تا مناطق با کنتراست محلی کمتر کنتراست بالاتری به دست آورند.

چندی سازی، یکی از تکنیک های فشرده‌سازی lossy (با اتلاف) است که با فشرده‌سازی محدوده ای از مقادیر به یک مقدار گسسته به دست می‌آید.

روش کوانتیزه کردن تصویر به این طریق است که اگر قرار است 128 سطح خاکستری در تصویر وجود داشته باشد، پس مقدار هر پیکسل باید به عددی در بازه (0-128) مپ بشود.

روش به دست آوردن optimum mean square error نیز مطابق فرمول زیر پیاده سازی شده است:

$$MSE = \left[\left(\frac{1}{N}\right)\sum_{i=1}^{N}(\hat{y}_i - y_i)^2\right] \qquad (1)$$

2-1-2- درون یابی

در این قسمت خواسته شده بود که درونیابی برای کوچک نمایی و سپس بزرگنمایی پیکسل های تصویر Goldhill را انجام دهیم.

کوچک کردن تصویر به فاکتور 2 به این مهناست که طول و عرض تصویر هر دو تقسیم بر 2 میشوند. دو روش برای کوچک کردن تصویر معرفی شده اند که یکی averaging و دیگری حذف یک سطر و یک ستون می باشد. Averaging به این معناست که برای هر پیکسلی که در تصویر کوچک شده قرار میدهیم، میانگین پیکسل های اطراف آن در تصویر اصلی را محاسبه کرده و قرار میدهیم. روش دوم هم این است که صرفا یکی در میان سطرها و ستون ها را حذف کنیم.

روش های معرفی شده برای بزرگ نمایی تصویر نیز pixel replication و bilinear ineterpolation می باشد. روش اول بیان میکند که برای هر 4 پیکسل جدیدی که باید در عکس بزرگ نمایی شده مقداردهی شوند، مقدار یکی از پیکسل ها نکرار شود. روش درونیابی دوخطی به این شیوه است که برای پیدا کردن یک مقدار بین 4 پیکسل مختلف،

ابتدا در محور طول درونیابی انجام شود و سپس مجددا در محور عرض ها درونیابی به دست بیاید.

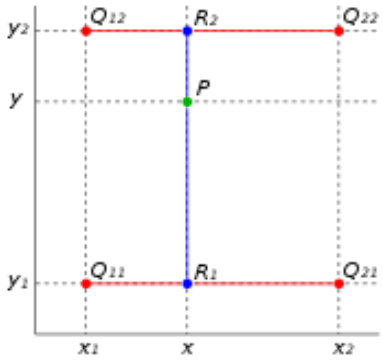


Figure 1چهار نقطه قرمز نقاط داده و نقطه سبز نقطه مورد نظر ما برای درونیابی است.

با توجه به شکل بالا فرمول به دست آوردن مقادیر مناسب برای پیکسل P به شکل زیر می باشد:

$$\begin{aligned} P = {} & Q_{12} * (y - y_2) * (x_2 - x) \\ & + Q_{22} * (y - y_2) \\ & * (x - x_1) + Q_{21} \\ & * (y_2 - y) \\ & * (x - x_1) + Q_{11} \\ & * (y_2 - y) \\ & * (x_2 - x) \end{aligned} \qquad (2)$$

3-1-2-

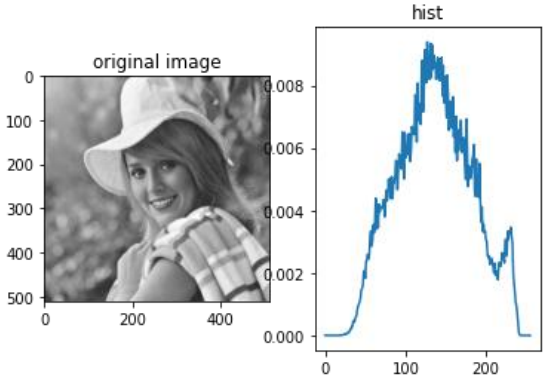
در این بخش خواسته شده تا تصویر را با تعداد بیت های مختلفی نمایش دهیم.

برای انجام این کار ما تعداد سطوح خاکستری به قدری کاهش میدهیم تا به جای 8 بیت، مقدار کمتری جا بگیرند.

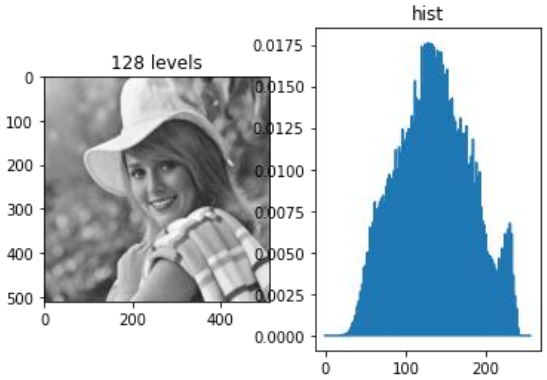
برای انجام این کار ما در نظر میگیریم که اگر قرار است تصویر 5بیتی باشد، یعنی تنها میتواند 32 سطح خاکستری تصویر وجود داشته باشد، پس مقدار هر پیکسل باید به عددی در بازه (0-31) مپ بشود. نتیجه و بررسی نتایج این بخش در قسمت نتایج به تفصیل بیان شده است.

3-نتایج

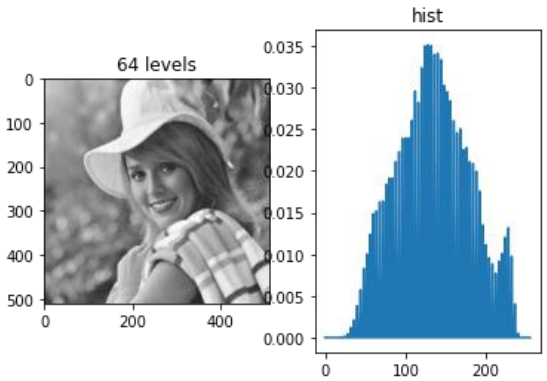
1-1-3



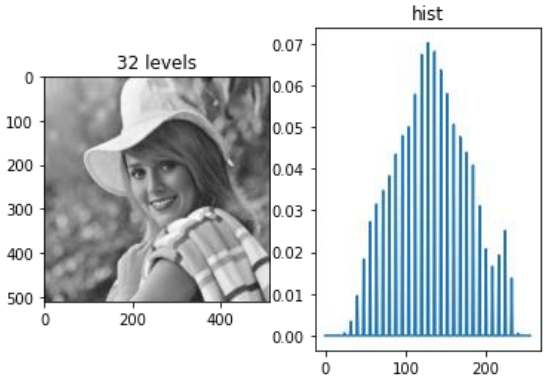
شکل 3-1-1- 1 تصویر اصلی و هیستوگرام آن



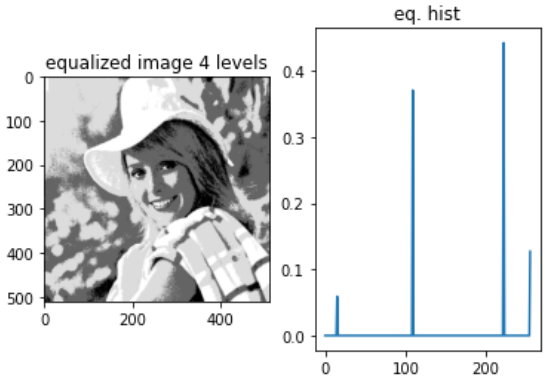
شکل 3-1-1- 2 تصویر با 128 سطح خاکستری و هیستوگرام آن



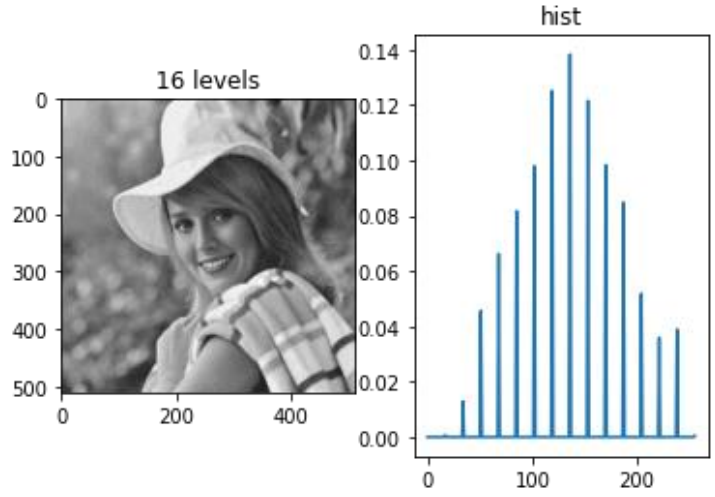
شکل 3-1-1- 3 تصویر با 64 سطح خاکستری و هیستوگرام آن



شکل 3-1-1- 4 تصویر با 32 سطح خاکستری و هیستوگرام آن



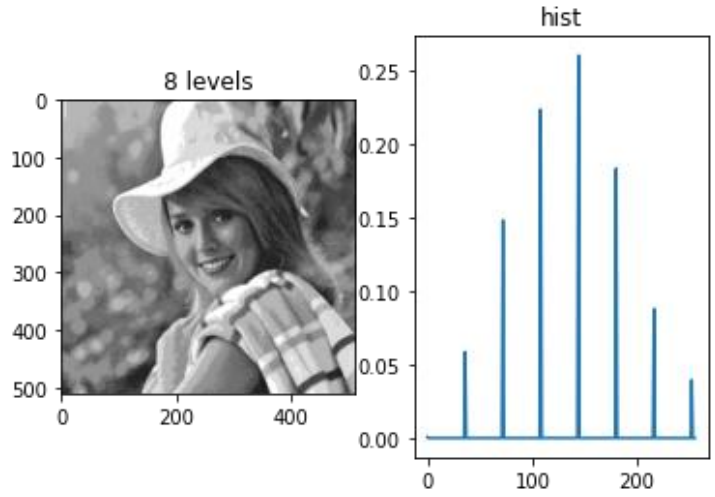
شکل 3-1-1- 7 تصویر با 4 سطح خاکستری و هیستوگرام آن



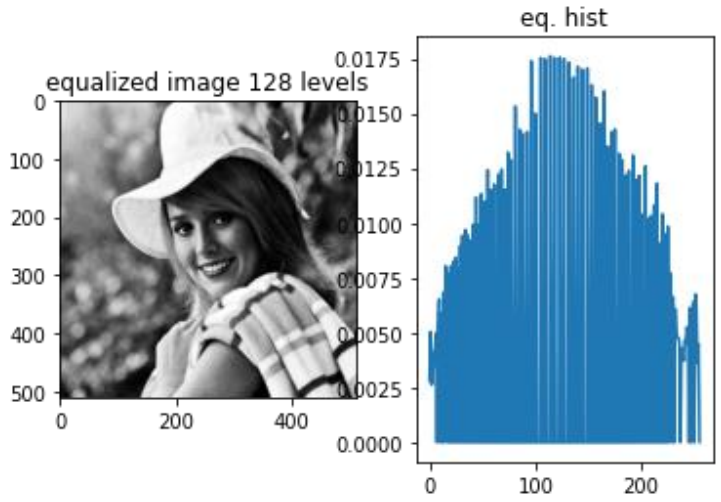
شکل 3-1-1- 5 تصویر با 16 سطح خاکستری و هیستوگرام آن

همانطور که در تصاویر بالا مشاهده میشود، کاهش تعداد سطوح خاکستری در هیستوگرام ها واضح است. اما کاهش کیفیت تصویر جوری که به چشم ما واضح باشد، تازه از 32 سطح خاکستری قابل تشخیص می باشد.

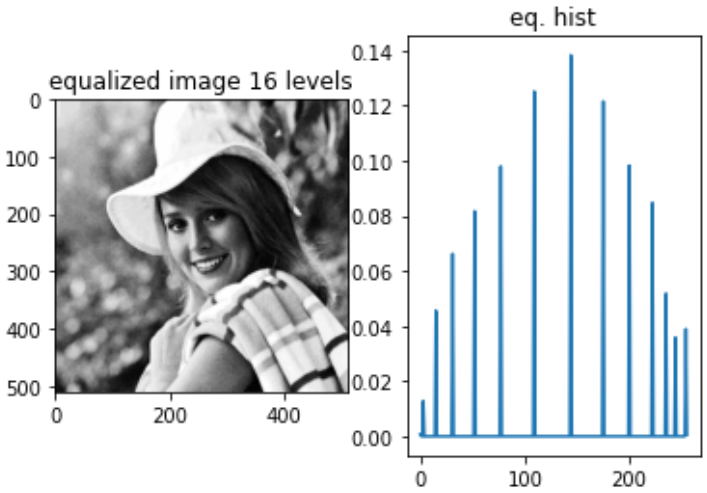
شکل 3-1-2 1 تصویر equalize شده و هیستوگرام آن



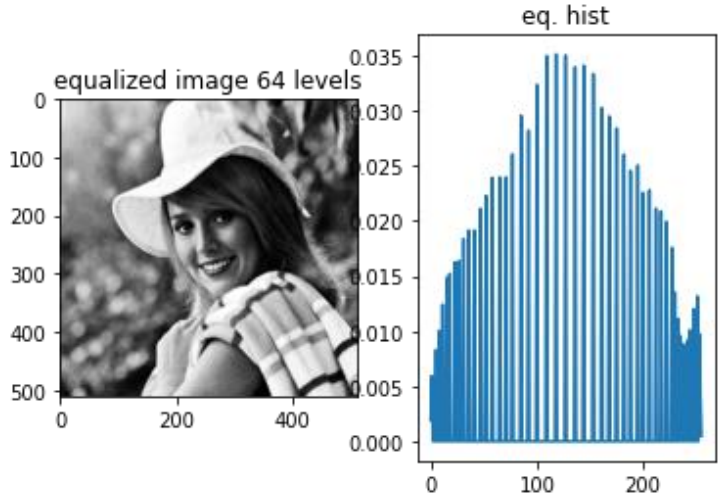
شکل 3-1-1- 6 تصویر با 8 سطح خاکستری و هیستوگرام آن



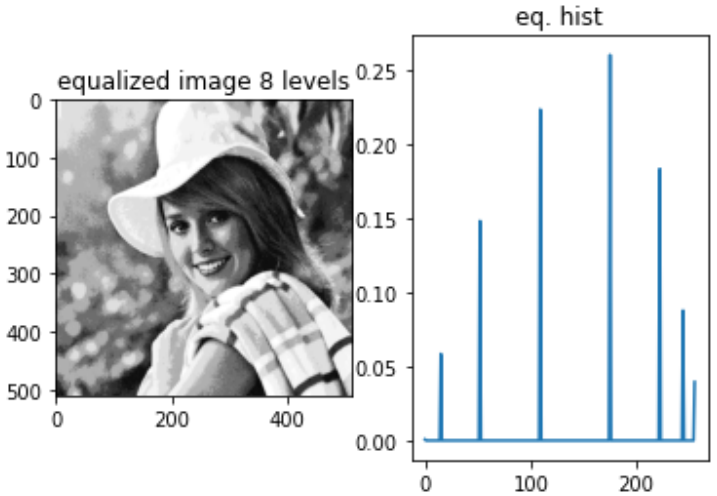
شکل 3-1-2 2 تصویر با 128 سطح خاکستری equalize شده و هیستوگرام آن



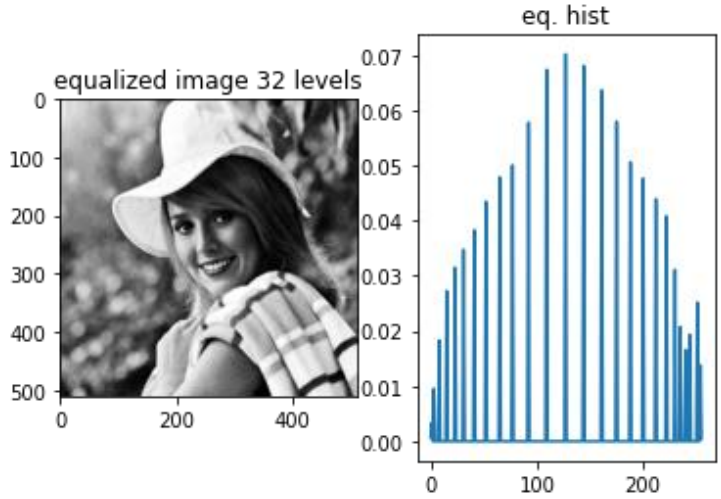
شکل 3-1-2 5 تصویر با 16 سطح خاکستری equalize شده و هیستوگرام آن



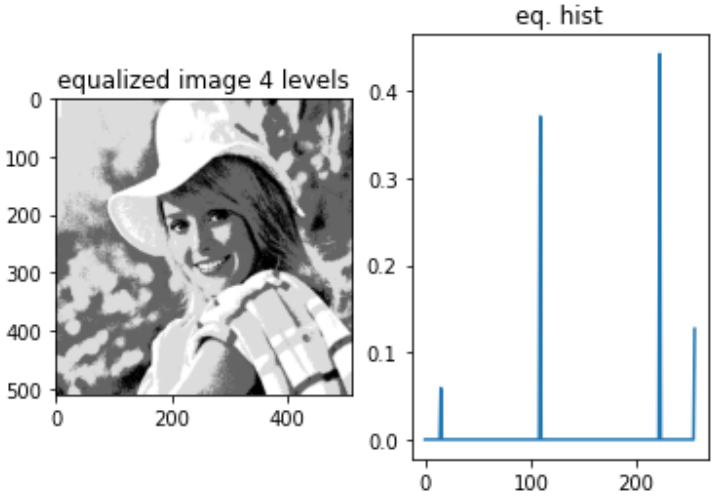
شکل 3-1-2 3 تصویر با 64 سطح خاکستری equalize شده و هیستوگرام آن



شکل 3-1-2 6 تصویر با 8 سطح خاکستری equalize شده و هیستوگرام آن



شکل 3-1-2 4 تصویر با 32 سطح خاکستری equalize شده و هیستوگرام آن



شکل 3-1-2 7 تصویر با 4 سطح خاکستری equalize شده و هیستوگرام آن

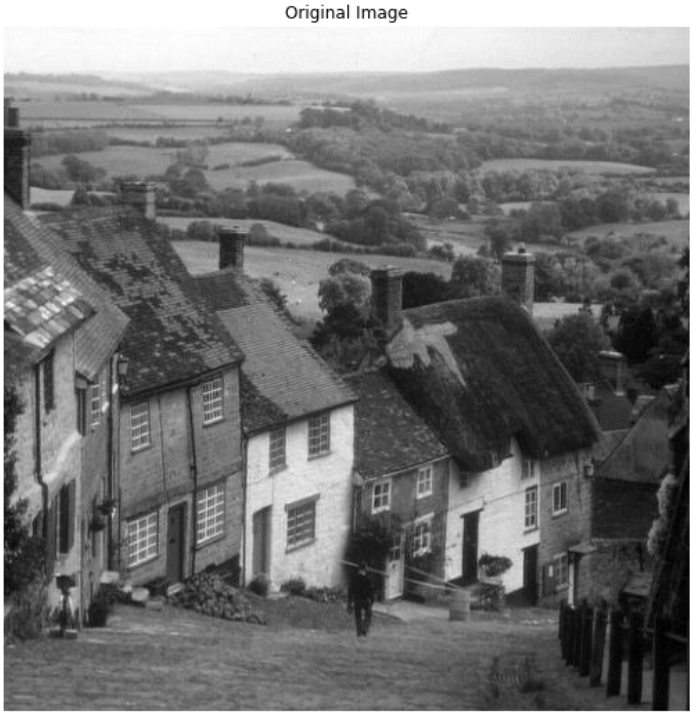
2-1-3

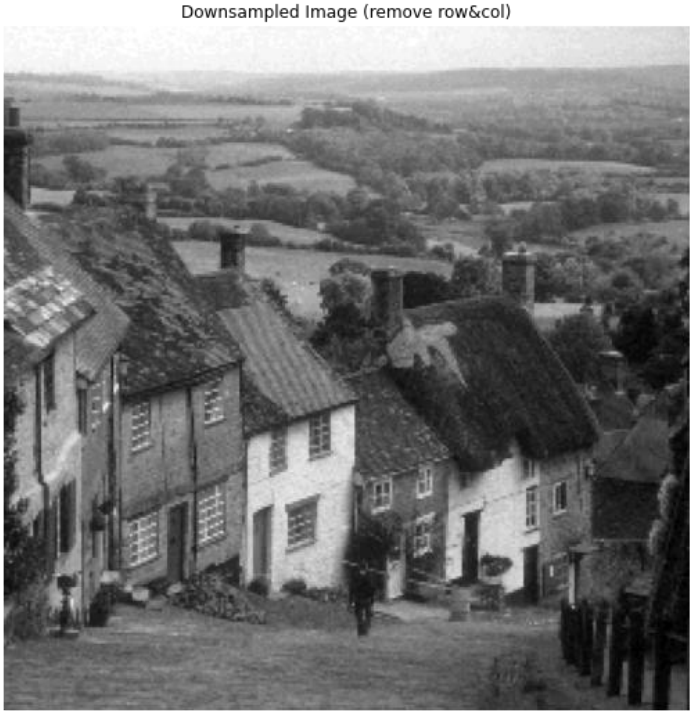
می دانیم متعادل سازی هیستوگرام به شکل هموارتر کردن هیستوگرام اولیه خودش را نشان می دهد و آن را ما در نتیجه به وضوح میبینیم. به این دلیل که هنگامی که سطوح خاکستری در یک ناحیه خاصی از هیستوگرام تجمع داشته باشند کنتراست به شدت پایین است. متعادل سازی هیستوگرام با هموار کردن این نمودار باعث افزایش کنتراست می شود. در عکس ها هم افزایش کنتراست را مشاهده میکنیم.

نتیجه مقایسه تک تک حالت های مختلف ذکرشده با تصویر اصلی در جدول زیر آمده است:

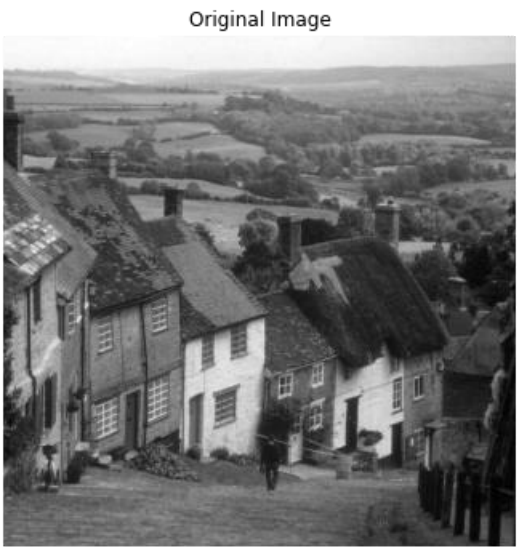
جدول 11 محاسبه mse

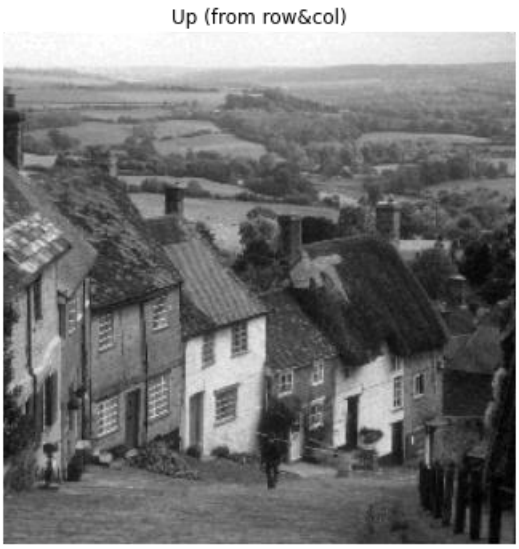
Levels	128	64	32	16	8	4
Without Histeq	0.509254	3.468834	17.381859	32.774803	76.570976	102.684330
With Histeq	102.832993	106.985451	106.654507	108.296638	108.791664	109.074158

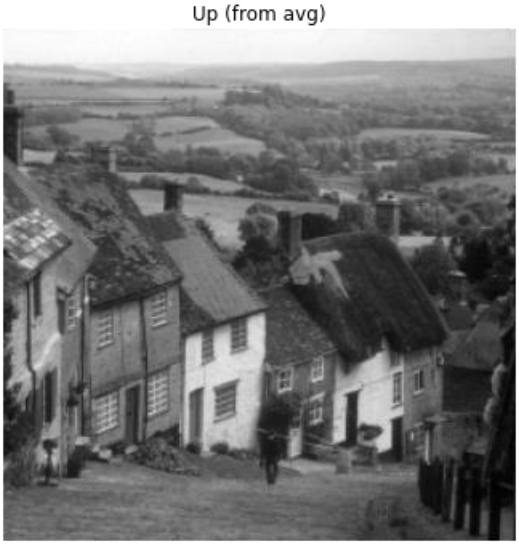




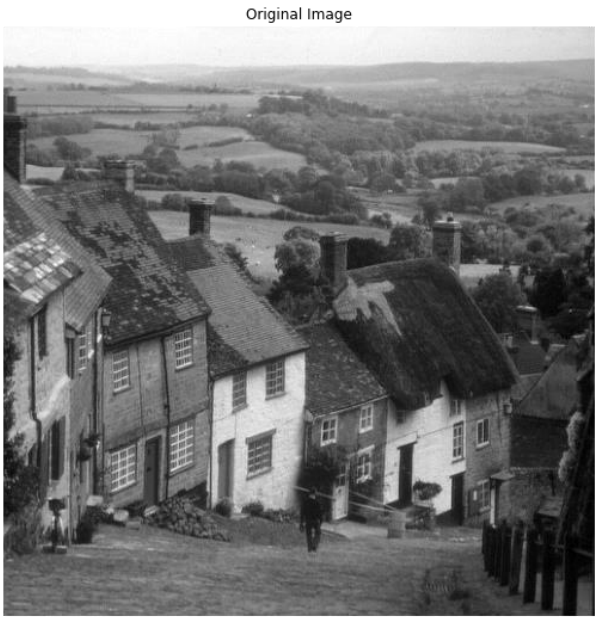
شکل 3-1-2-1 downsample 1 کردن تصویر به روش حذف سطروستون

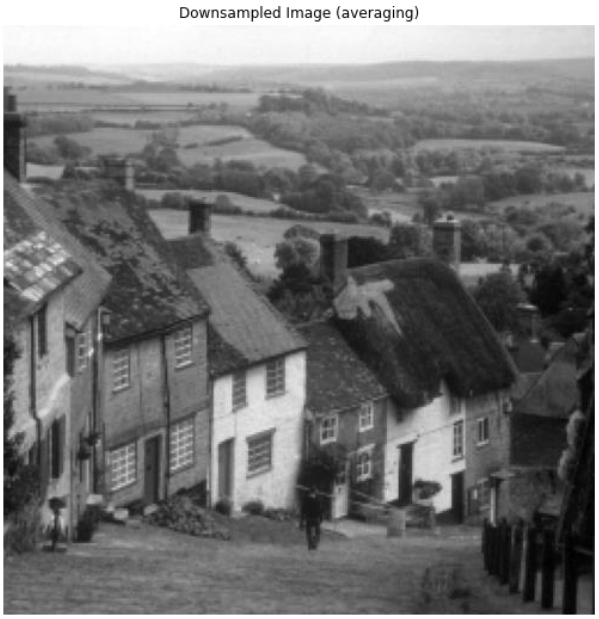






شکل 3-1-2-1 3 Upsample pixel replication (Nearest Neighbor Interpolation)





شکل 3-1-2-1 2 downsampleکردن تصویر به روش میانگین گیری

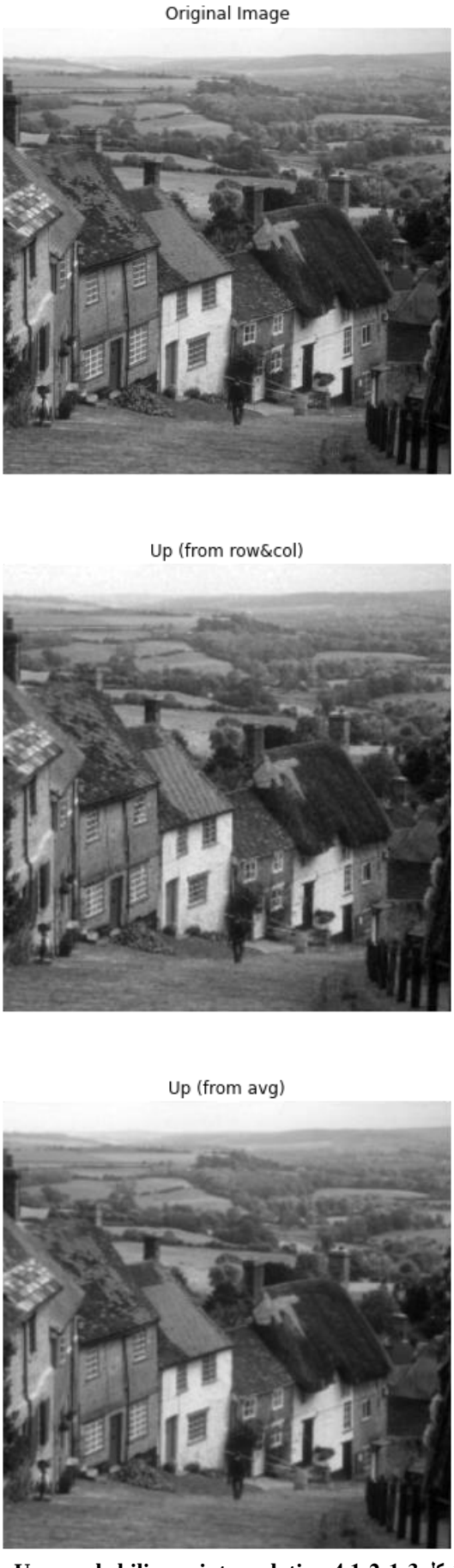
همانطور که در تصاویر نتیجه میتوان مشاهده کرد، هنگام کوچک نمایی تصویر در حالتی که سطر و ستون حذف شده است، تصویر حالت پیکسل پیکسلی دارد اما در مقابل آن تصویری که به روش میانگین گیری downsample شده است smoothتر است و لبه ها محوتر شده اند.

به تبع از آن، هنگام بزرگنمایی تصویر مشاهده می کنیم که هنگام بزرگنمایی به روش pixel replication از تصویری که به روش حذف سطر و ستون کوچک نمایی شده است، بدترین کیفیت را مشاهده میکنیم زیرا به ازای هر 4 پیکسل تصویر اصلی، درواقع یک تصویر چهار بار تکرار شده است.

در مقابل آن هنگام بزرگنمایی به روش درونیابی دوخطی از تصویری که به روش میانگین گیری downsample شده است، کیفیت تصویر بسیار مشابه تصویر اصلی می باشد. زیرا به ازای هر 4 پیکسل تصویر اصلی مجموع تمام سطوح خاکستری محاسبه شده و تقسیم بر 4 کنیم. سپس هنگام بزرگنمایی بسته به موقعیت پیکسل در تصویر، میانگین وزن دار چهار نقطه نزدیک یک پیکسل را قرار می دهیم.

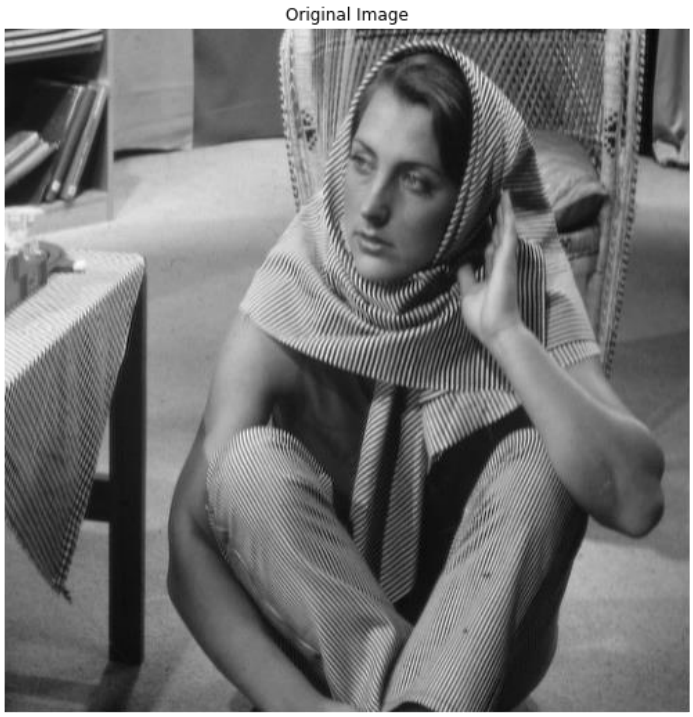
جدول 21 محاسبه mse

	Pixel Replication	Bilinear Interpolation
Averaging	35.253479	36.820927
Remove Row&Columns	23.089600	41.719284

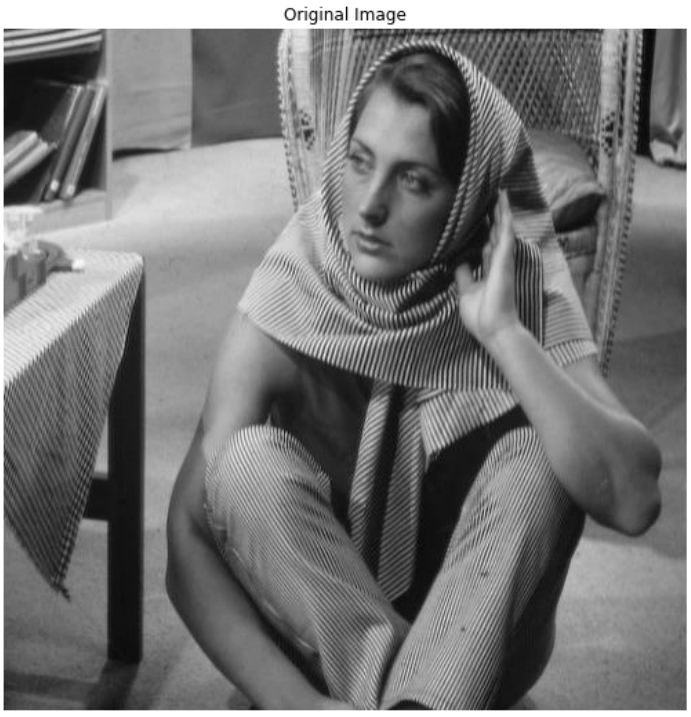


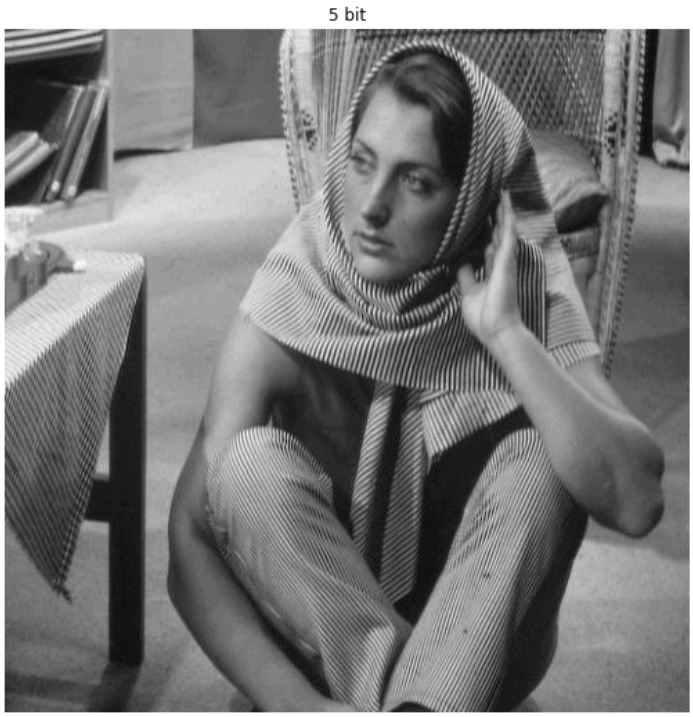
شکل 3-1-2-1 Upsample bilinear interpolation 4

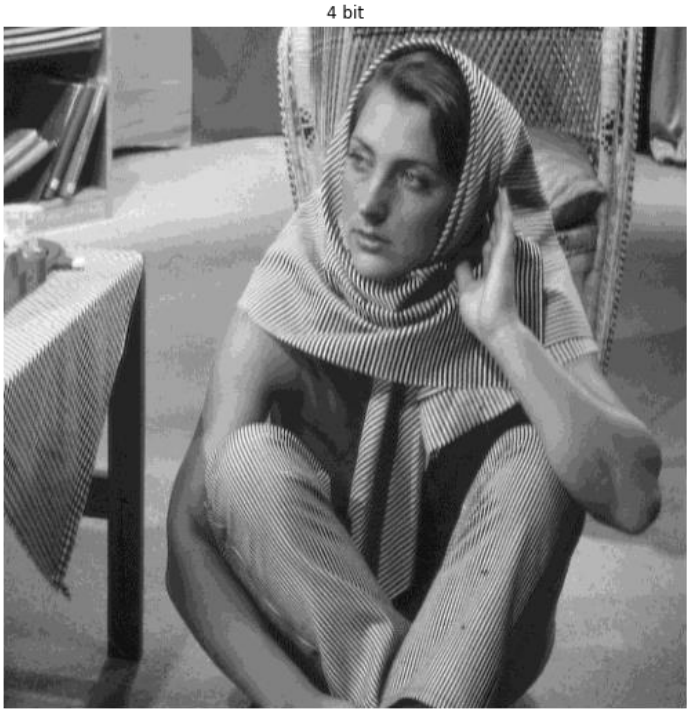
3-1-3



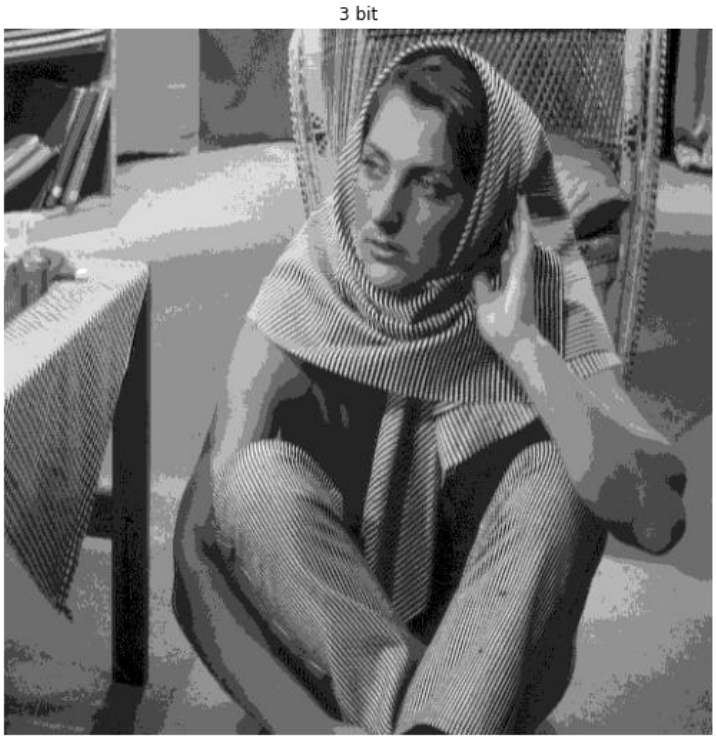
شکل 3-1-3 1 تصویر اصلی و نمایش 5بیتی آن





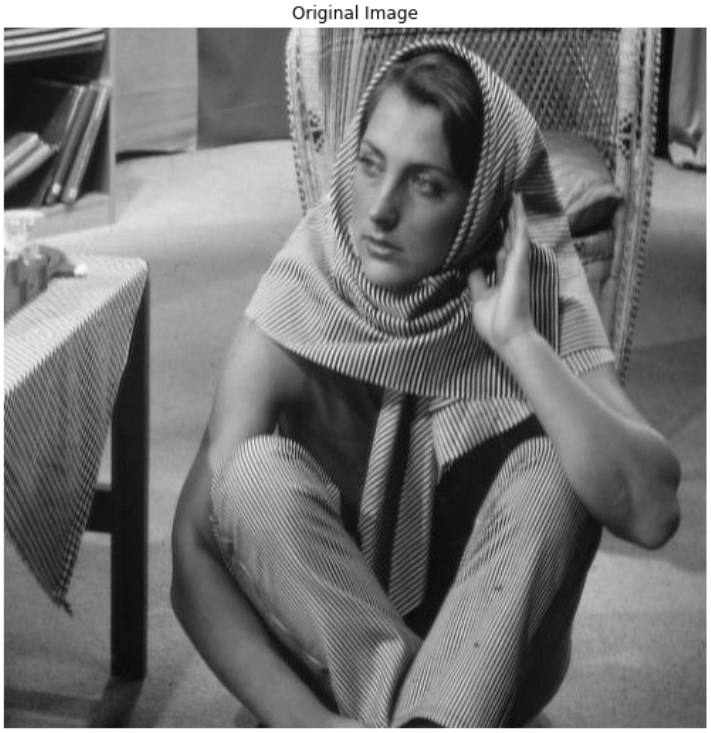


شکل 3-1-3 2 تصویر اصلی و نمایش 4بیتی آن



شکل 3-1-3 3 تصویر اصلی و نمایش 3بیتی آن

شکل 3-1-3 4 تصویر اصلی و نمایش 2بیتی آن



شکل 3-1-5 تصویر اصلی و نمایش 1بیتی آن

همانطور که در تصاویر نتایج مشاهده می شود، تا نمایش 4بیتی تقریبا تفاوت خاصی با تصویر اصلی توسط چشم ما تشخیص داده نمیشود. اما از نمایش سه بیتی به بعد کیفیت تصویر پایین آمده و در نقاطی از تصویر که تنوع سطوح خاکستری زیاد است، نمایش با تعداد بیت کم تفاوت فاحشی را در پی دارد.

به طور کلی اگر عکسی داشته باشیم که در ناحیه ای از آن، تنوع رنگ کمتر است و یکدست تر است و مثلا نهایتا بتوان آن را در دو سطح خاکستری توصیف کرد آنگاه 1 بیت هم برای نمایش آن ناحیه از تصویر کفایت میکند. اما در محله هایی از تصویر که تنوع رنگ و سطوح خاکستری بیشتری داریم حتما نیاز به بیت های بیشتری برای بازنمایی مقدار هر پیکسل داریم تا بتوان بین آنها تمایز ایجاد کرد.

Code-4

1-1-1

```
def hist(image):
  m, n = image.shape
  hist = [0.0] * 256
  for i in range(m):
    for j in range(n):
      hist[image[i, j]]+=1
  return np.array(hist)/(m*n
)

def cdf(hist):
  return [sum(hist[:i+1]) fo
r i in range(len(hist))]

def histeq(image):
  h = hist(image)
  cumsum = np.array(cdf(h))
  sk = np.uint8(255 * cumsum
)
  s1, s2 = image.shape
  new_img = np.zeros_like(im
age)
  # applying transfered valu
es for each pixels
  for i in range(0, s1):
    for j in range(0, s2):
```

```
##### 128 levels
img128levels = quantize(elai
ne,7)

plt.subplot(121)
plt.imshow(img128levels)
plt.title('128 levels')


hist128 = hist(np.uint8(img1
28levels))

plt.subplot(122)
plt.plot(hist128)
plt.title('hist')
plt.show()



#####64 levels
img64levels = quantize(elain
e,6)

plt.subplot(121)
plt.imshow(img64levels)
plt.title('64 levels')


hist64 = hist(np.uint8(img64
levels))

plt.subplot(122)
plt.plot(hist64)
plt.title('hist')
plt.show()


#####32 levels
img32levels = quantize(elain
e,5)

plt.subplot(121)
plt.imshow(img32levels)
```

```
      new_img[i, j] = sk[ima
ge[i, j]]
  Hist = hist(new_img)
  #return transformed image,
 original and new histogram,

  # and transform function
  return new_img , Hist


###1.1.1. display the quanti
zed image in (4,8,16,32,64,1
28) Levels and its histogram
s


elaine = cv2.imread('/conten
t/sample_data/Elaine.bmp', c
v2.IMREAD_GRAYSCALE)

def quantize(image, n_bits):
    coeff = 2**8 // 2**n_bit
s
    coeff2= 2**8 // (2**n_bi
ts - 1)
    return (image // coeff)
* coeff2


####Without HistEq
#####original image and hist
plt.subplot(121)
plt.imshow(elaine)
plt.title('original image')

orghist = hist(np.uint8(elai
ne))

plt.subplot(122)
plt.plot(orghist)
plt.title('hist')
plt.set_cmap('gray')
plt.show()
```

```
plt.title('hist')
plt.show()
#####4 levels
img4levels = quantize(elaine
,2)

plt.subplot(121)
plt.imshow(img4levels)
plt.title('4 levels')


hist4 = hist(np.uint8(img4le
vels))

plt.subplot(122)
plt.plot(hist4)
plt.title('hist')
plt.show()



####With HistEq
#####original image and hist
new_img, histeqorg = histeq(
np.uint8(elaine))

plt.subplot(121)
plt.imshow(new_img)
plt.title('equalized image')

plt.subplot(122)
plt.plot(histeqorg)
plt.title('eq. hist')
plt.show()



#####128 levels
img128levels = quantize(elai
ne, 7)
new_img128levels, histeq128
= histeq(np.uint8(img128leve
ls))
```

```
plt.title('32 levels')


hist32 = hist(np.uint8(img32
levels))

plt.subplot(122)
plt.plot(hist32)
plt.title('hist')
plt.show()


#####16 levels
img16levels = quantize(elain
e,4)

plt.subplot(121)
plt.imshow(img16levels)
plt.title('16 levels')


hist16 = hist(np.uint8(img16
levels))

plt.subplot(122)
plt.plot(hist16)
plt.title('hist')
plt.show()



#####8 levels
img8levels = quantize(elaine
,3)

plt.subplot(121)
plt.imshow(img8levels)
plt.title('8 levels')


hist8 = hist(np.uint8(img8le
vels))

plt.subplot(122)
plt.plot(hist8)
```

```
plt.title('eq. hist')
plt.show()

#####16 levels
img16levels = quantize(elain
e, 4)
new_img16levels, histeq16 =
histeq(np.uint8(img16levels)
)

plt.subplot(121)
plt.imshow(new_img16levels)
plt.title('equalized image 1
6 levels')

plt.subplot(122)
plt.plot(histeq16)
plt.title('eq. hist')
plt.show()



#####8 levels
img8levels = quantize(elaine
, 3)
new_img8levels, histeq8 = hi
steq(np.uint8(img8levels))

plt.subplot(121)
plt.imshow(new_img8levels)
plt.title('equalized image 8
 levels')

plt.subplot(122)
plt.plot(histeq8)
plt.title('eq. hist')
plt.show()


#####4levels
img4levels = quantize(elaine
, 2)
new_img4levels, histeq4 = hi
steq(np.uint8(img4levels))
```

```
plt.subplot(121)
plt.imshow(new_img128levels)
plt.title('equalized image 1
28 levels')

plt.subplot(122)
plt.plot(histeq128)
plt.title('eq. hist')
plt.show()



#####64 levels
img64levels = quantize(elain
e, 6)
new_img64levels, histeq64 =
histeq(np.uint8(img64levels)
)

plt.subplot(121)
plt.imshow(new_img64levels)
plt.title('equalized image 6
4 levels')

plt.subplot(122)
plt.plot(histeq64)
plt.title('eq. hist')
plt.show()


#####32 levels
img32levels = quantize(elain
e, 5)
new_img32levels, histeq32 =
histeq(np.uint8(img32levels)
)

plt.subplot(121)
plt.imshow(new_img32levels)
plt.title('equalized image 3
2 levels')

plt.subplot(122)
plt.plot(histeq32)
```

1-1-2

```
goldhill = cv2.imread('/cont
ent/sample_data/Goldhill.bmp
', cv2.IMREAD_GRAYSCALE)

####Downsample remove row&co
lumn
#remove row&column
rc_downsample = goldhill[0::
2,0::2]

plt.figure(figsize=(20,20))

plt.subplot(211)
plt.imshow(goldhill,cmap = '
gray')
plt.axis('off')
plt.title('Original Image')

plt.subplot(212)
plt.imshow(rc_downsample,cma
p = 'gray')
plt.title('Downsampled Image
 (remove row&col)')

plt.axis('off')
plt.show()


####Downsample averaging
def down_sample_avg(im):
    original_width = im.shap
e[1]
    original_height = im.sha
pe[0]
    width = original_width /
/ 2
    height = original_height
 // 2
    resized_image = np.zeros
(shape=(height, width), dtyp
e=np.uint8)
    scale = 2
    for i in range(height):
```

```

plt.subplot(121)
plt.imshow(new_img4levels)
plt.title('equalized image 4
 levels')

plt.subplot(122)
plt.plot(histeq4)
plt.title('eq. hist')
plt.show()



####MSE
def MSE(x,y):
  return np.square(np.subtra
ct(x,y)).mean()

df = pd.DataFrame([('Without
 Histeq', MSE(elaine, img128
levels), MSE(elaine, img64le
vels), MSE(elaine, img32leve
ls),
                    MSE(elai
ne, img16levels), MSE(elaine
, img8levels), MSE(elaine, i
mg4levels)),
                   ('With Hi
steq', MSE(elaine, new_img12
8levels), MSE(elaine, new_im
g64levels), MSE(elaine, new_
img32levels),
                    MSE(elai
ne, new_img16levels), MSE(el
aine, new_img8levels), MSE(e
laine, new_img4levels))],
                    columns=
('Levels', '128', '64', '32'
, '16', '8', '4' ))
```

```
  for y in range(len(output)
):
    for x in range(len(outpu
t[y])):
      proj_x = x // 2
      proj_y = y // 2
      output[y][x] = input[p
roj_y][proj_x]

  return output


nn_rc_upsample = nearest(rc_
downsample)
nn_avg_upsample = nearest(av
g_downsample)

plt.figure(figsize=(20,20))

plt.subplot(311)
plt.imshow(goldhill,cmap = '
gray')
plt.axis('off')
plt.title('Original Image')

plt.subplot(312)
plt.imshow(nn_rc_upsample,cm
ap = 'gray')
plt.axis('off')
plt.title('Up (from row&col)
')

plt.subplot(313)
plt.imshow(nn_avg_upsample,c
map = 'gray')
plt.axis('off')
plt.title('Up (from avg)')

plt.show()



####Upsample bilinear interp
olation
```

```
        for j in range(width
):
            temp = 0
            for x in range(s
cale):
                for y in ran
ge(scale):
                    temp +=
im[i*scale + x, j*scale + y]
            resized_image[i,
 j] = temp/(scale*scale)

    return resized_image


avg_downsample = down_sample
_avg(goldhill)
plt.figure(figsize=(20,20))
plt.subplot(211)
plt.imshow(goldhill,cmap = '
gray')
plt.axis('off')
plt.title('Original Image')

plt.subplot(212)
plt.imshow(avg_downsample,cm
ap = 'gray')
plt.axis('off')
plt.title('Downsampled Image
 (averaging)')

plt.show()




####Upsample pixel replicati
on (Nearest Neighbor Interpo
lation)
def nearest(input,sx=512,sy=
512):
  output = np.zeros((sx, sy)
, input.dtype)
```

```
    orig_width = image.shape
[1]

    # Compute center column
and center row
    x_orig_center = (orig_wi
dth-1) / 2
    y_orig_center = (orig_he
ight-1) / 2

    # Compute center of resi
zed image
    x_scaled_center = (new_w
idth-1) / 2
    y_scaled_center = (new_h
eight-1) / 2

    # Compute the scale in b
oth axes
    scale_x = orig_width / n
ew_width;
    scale_y = orig_height /
new_height;

    for y in range(new_heigh
t):
        for x in range(new_w
idth):
            x_ = (x - x_scal
ed_center) * scale_x + x_ori
g_center
            y_ = (y - y_scal
ed_center) * scale_y + y_ori
g_center
            new_image[y, x]
= bilinear_interpolation(ima
ge, y_, x_)

    return new_image



bi_rc_upsample = resize(rc_d
ownsample)
```

```
def bilinear_interpolation(i
mage, y, x):
    height = image.shape[0]
    width = image.shape[1]

    x1 = max(min(math.floor(
x), width - 1), 0)
    y1 = max(min(math.floor(
y), height - 1), 0)

    x2 = max(min(math.ceil(x
), width - 1), 0)
    y2 = max(min(math.ceil(y
), height - 1), 0)

    a = float(image[y1, x1])
    b = float(image[y2, x1])
    c = float(image[y1, x2])
    d = float(image[y2, x2])

    dx = x - x1
    dy = y - y1

    new_pixel = a * (1 - dx)
 * (1 - dy)
    new_pixel += b * dy * (1
 - dx)
    new_pixel += c * dx * (1
 - dy)
    new_pixel += d * dx * dy
    return round(new_pixel)


def resize(image, new_height
=512, new_width=512):
    new_image = np.zeros((ne
w_height, new_width), image.
dtype)  # new_image = [[0 fo
r _ in range(new_width)] for
 _ in range(new_height)]

    orig_height = image.shap
e[0]
```

```
###1.1.3 Create new images u
sing 5, 4, 3, 2 and 1 bit on
ly for each pixel.
def quantize2(image, n_bits)
:
    coeff = 2**8 // 2**n_bit
s
    return (image // coeff)


####5bit
img5bit = quantize2(barbara,
 5)

plt.figure(figsize=(20,20))
plt.set_cmap('gray')

plt.subplot(211)
plt.axis('off')
plt.imshow(barbara)
plt.title('Original Image')

plt.subplot(212)
plt.imshow(img5bit)
plt.axis('off')
plt.title('5 bit')
plt.show()

####4bit
img4bit = quantize2(barbara,
 4)

plt.figure(figsize=(20,20))
plt.set_cmap('gray')

plt.subplot(211)
plt.axis('off')
plt.imshow(barbara)
plt.title('Original Image')

plt.subplot(212)
plt.axis('off')
plt.imshow(img4bit)
plt.title('4 bit')
```

```
bi_avg_upsample = resize(avg
_downsample)

plt.figure(figsize=(20,20))

plt.subplot(311)
plt.imshow(goldhill,cmap = '
gray')
plt.axis('off')
plt.title('Original Image')

plt.subplot(312)
plt.axis('off')
plt.imshow(bi_rc_upsample,cm
ap = 'gray')
plt.title('Up (from row&col)
')

plt.subplot(313)
plt.imshow(bi_avg_upsample,c
map = 'gray')
plt.axis('off')
plt.title('Up (from avg)')

plt.show()


df = pd.DataFrame([('Averagi
ng', MSE(goldhill, nn_avg_up
sample), MSE(goldhill, bi_av
g_upsample)),
                   ('Remove
Row&Columns', MSE(goldhill,
nn_rc_upsample), MSE(goldhil
l, bi_rc_upsample))],
                    columns=
('', 'Pixel Replication', 'B
ilinear Interpolation'))
```

1-1-3

```
plt.title('Original Image')

plt.subplot(212)
plt.axis('off')
plt.imshow(img2bit)
plt.title('2 bit')
plt.show()



####1bit
img1bit = quantize2(barbara,
 1)

plt.figure(figsize=(20,20))
plt.set_cmap('gray')

plt.subplot(211)
plt.axis('off')
plt.imshow(barbara)
plt.title('Original Image')

plt.subplot(212)
plt.imshow(img1bit)
plt.axis('off')
plt.title('1 bit')
plt.show()
```

```
plt.show()



####3bit
img3bit = quantize2(barbara,
 3)

plt.figure(figsize=(20,20))
plt.set_cmap('gray')

plt.subplot(211)
plt.axis('off')
plt.imshow(barbara)
plt.title('Original Image')

plt.subplot(212)
plt.imshow(img3bit)
plt.axis('off')
plt.title('3 bit')
plt.show()



####2bit
img2bit = quantize2(barbara,
 2)

plt.figure(figsize=(20,20))
plt.set_cmap('gray')

plt.subplot(211)
plt.axis('off')
plt.imshow(barbara)
```