Image Fundamentals

آیلین حسن پور آذری

اطلاعات گزارش

تاریخ: 10 / 1 /1402

واژگان کلیدی:
تصویر
گزارش
چندی سازی
پردازش نقطه ای
سطوح خاکستری
کیفیت تصویر
روش های درونیابی

چکیده

در این گزارش، مفاهیم چندی سازی (quantization) ، پردازش نقطه ای (point processing) 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.

hassanporailin@gmail.com

2-شرح تکنیکال

1-1-چندی سازی و درون یابی

1-1-1 چندی سازی

در این قسمت از ما خواسته شده تصویر lena را به تعداد (2، 4، 8، 16، 32، 64، 128) سطوح خاکستری کوانتیزه کنیم و برای هر حالت، تصویر را در کنار هیستوگرام آن نمایش دهیم.

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.

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. این کار به طور مشابه برای سایر سطوح نیز انجام میشود .

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

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

1-1-2-

در این یخش از ما خواسته شده تا تمامی پیکسل های تصویر lena را با بیت ها مختلف (1 ، 2 ، 4 ، 6) نمایش دهیم .

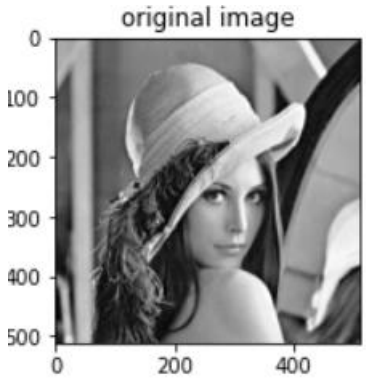
برای انجام این کار ما تعداد سطوح خاکستری به قدری کاهش میدهیم تا به جای 8 بیت، مقدار کمتری جا بگیرند. برای انجام این کار ما در نظر میگیریم که اگر قرار است تصویر 5بیتی باشد، یعنی تنها میتواند 32 سطح خاکستری تصویر وجود داشته باشد، پس مقدار هر پیکسل باید به عددی در بازه (0-31) مپ بشود. نتیجه و بررسی نتایج این بخش در قسمت نتایج به تفصیل بیان شده است.

1-1-3-

در این بخش خواسته شده تا با پردازش نقطه ای و پیاده سازی توابع مختلف (Darken, Lighten, Lower contrast, raised contrast and Invert) بر روی هر پیکسل از تصویر ، خروجی های مناسب را نمایش دهیم .

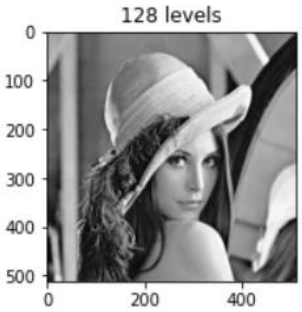
3-نتایج

1-1-1



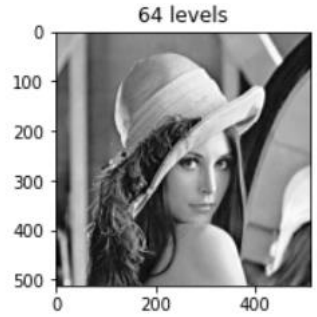
شکل 1-1-1- 1 تصویر اصلی

```
Quantization level: 128, MSE: 0.4506988525
```



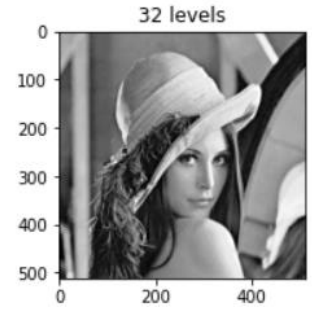
شکل 2-1-1- 1 تصویر با 128 سطح خاکستری و MSE آن

Quantization level: 64, MSE: 3.5216827392



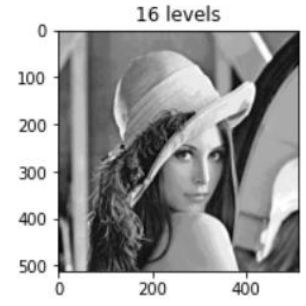
شکل 3-1-1- 1 تصویر با 64 سطح خاکستری و MSE آن

Quantization level: 32, MSE: 17.0908355712



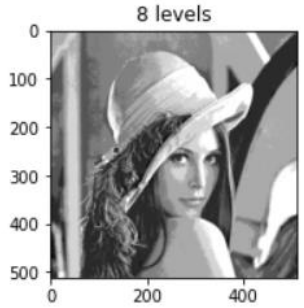
شکل 4-1-1- 1 تصویر با 32 سطح خاکستری و MSE آن

Quantization level: 16, MSE: 30.71071243286



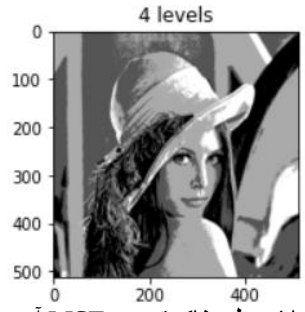
شکل 5-1-1- 1 تصویر با 16 سطح خاکستری و MSE آن

Quantization level: 8, MSE: 79.16493225097



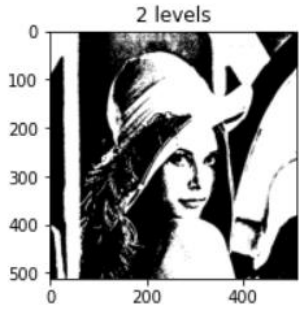
شکل 6-1-1- 1تصویر با 8 سطح خاکستری و MSE آن

Quantization level: 4, MSE: 106.4938049316



شکل 7-1-1- 1 تصویر با 4 سطح خاکستری و MSE آن

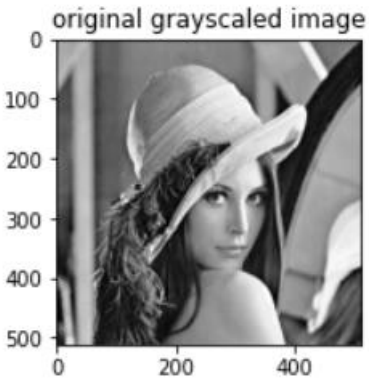
Quantization level: 2, MSE: 8551.101333618



شکل 8-1-1- 1 تصویر با 2 سطح خاکستری و MSE آن

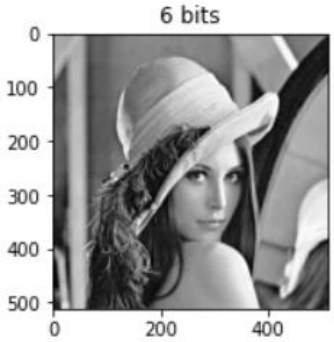
همانطور که در تصاویر بالا مشاهده میشود، کاهش تعداد سطوح خاکستری در هیستوگرام ها واضح است. اما کاهش کیفیت تصویر جوری که به چشم ما واضح باشد، تازه از 32 سطح خاکستری قابل تشخیص می باشد.

1-1-2



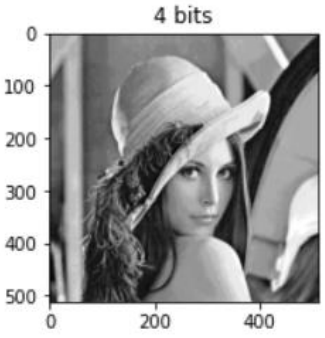
شکل 1-2-1- 2 تصویر اصلی

6-bit quantization MSE: 3.52168273925



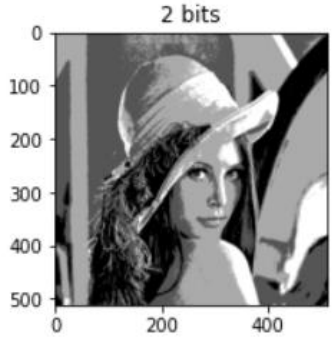
شکل 2-2-1- 1 تصویر با 6 بیت و MSE آن

4-bit quantization MSE: 78.83815002441



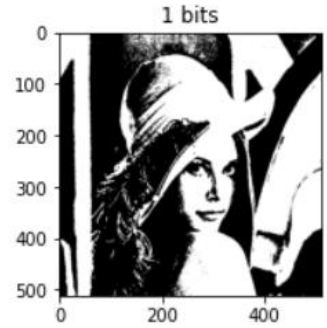
شکل 3-2-1- 1 تصویر با 4 بیت و MSE آن

2-bit quantization MSE: 104.0940093994



شکل 4-2-1- 1 تصویر با 2 بیت و MSE آن

1-bit quantization MSE: 105.83863830566

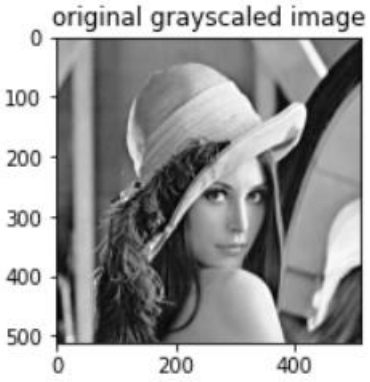


شکل 5-2-1- 1 تصویر با 1 بیت و MSE آن

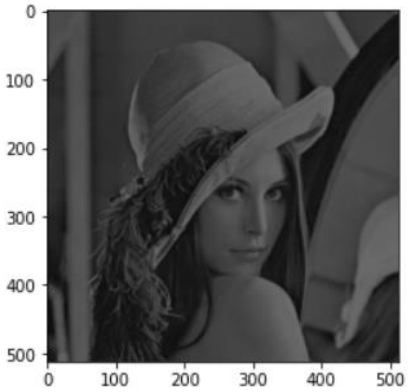
همانطور که در تصاویر نتایج مشاهده می شود، تا نمایش 4بیتی تقریبا تفاوت خاصی با تصویر اصلی توسط چشم ما تشخیص داده نمیشود. اما از نمایش سه بیتی به بعد کیفیت تصویر پایین آمده و در نقاطی از تصویر که تنوع سطوح خاکستری زیاد است، نمایش با تعداد بیت کم تفاوت فاحشی را در پی دارد.

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.

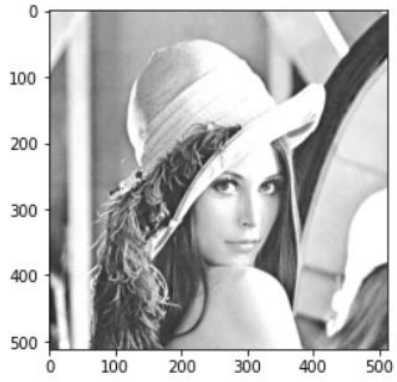
1-1-3



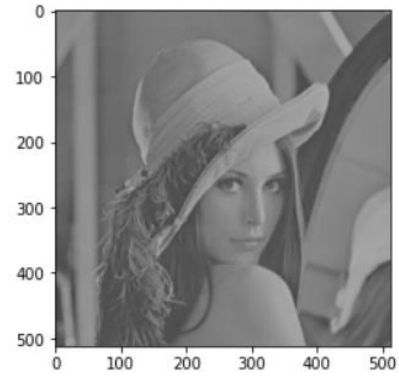
شکل 1-3-1- 3 تصویر اصلی



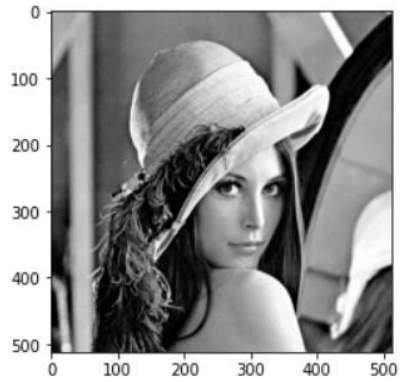
شکل 2-3-1- 4 تصویر Darken



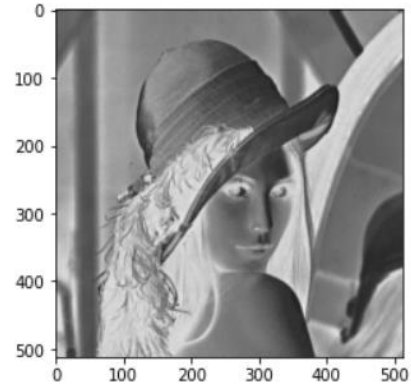
شکل 3-3-1- 5 تصویر Lighten



شکل 4-3-1- 6 تصویر Lower Contrast



شکل 5-3-1- 7 تصویر Raised Contrast



شکل 6-3-1- 8 تصویر Invert

Code-4

1-1-1

```
mse = np.mean((img_arr64 -
img64levels) ** 2)
print(f"Quantization level: {64},
MSE: {mse}")

plt.show()


##### 32 levels

img32levels = quantize(lena,5)
plt.subplot(121)
plt.imshow(img32levels)
plt.title('32 levels')


img_arr32 = np.array(lena)
#MSE
mse = np.mean((img_arr32 -
img32levels) ** 2)
print(f"Quantization level: {32},
MSE: {mse}")

plt.show()


##### 16 levels

img16levels = quantize(lena,4)
plt.subplot(121)
plt.imshow(img16levels)
plt.title('16 levels')


img_arr16 = np.array(lena)
#MSE
mse = np.mean((img_arr16 -
img16levels) ** 2)
print(f"Quantization level: {16},
MSE: {mse}")

plt.show()

##### 8 levels

img8levels = quantize(lena,3)
plt.subplot(121)
plt.imshow(img8levels)
plt.title('8 levels')


img_arr8 = np.array(lena)
#MSE
mse = np.mean((img_arr8 -
img8levels) ** 2)
print(f"Quantization level: {8},
MSE: {mse}")

plt.show()

##### 4 levels
```

```
############################
### 1.1.1. display the quantized
image in (4,8,16,32,64,128) and
calculate their MSE

lena = cv2.imread('lena.bmp',
cv2.IMREAD_GRAYSCALE)
flat=lena.flatten()

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

##### original image

plt.subplot(121)
plt.imshow(lena)
plt.title('original image')

#convert to grayscale
plt.set_cmap('gray')
plt.show()


#### 128 levels
img128levels = quantize(lena,7)
plt.subplot(121)
plt.imshow(img128levels)
plt.title('128 levels')

img_arr128 = np.array(lena)
#MSE
mse = np.mean((img_arr128 -
img128levels) ** 2)
print(f"Quantization level: {128},
MSE: {mse}")


plt.show()


##### 64 levels

img64levels = quantize(lena,6)
plt.subplot(121)
plt.imshow(img64levels)
plt.title('64 levels')


img_arr64 = np.array(lena)
#MSE
```

```
4)
mse6 = np.mean((data - q6) ** 2)
print("6-bit quantization MSE:",
mse6)

a=Image.fromarray(q6)
plt.subplot(121)
plt.imshow(a)
plt.title('6 bits')
plt.show()

##### 4 bits

# Quantize the pixel values using
4 bits
q4 = np.uint8(np.floor(data / 16)
* 16)
mse4 = np.mean((data - q4) ** 2)
print("4-bit quantization MSE:",
mse4)

IM = Image.fromarray(q4)
plt.subplot(121)
plt.imshow(IM)
plt.title('4 bits')
plt.show()


##### 2 bits

# Quantize the pixel values using
2 bits
q2 = np.uint8(np.floor(data / 64)
* 64)
mse2 = np.mean((data - q2) ** 2)
print("2-bit quantization MSE:",
mse2)

IM = Image.fromarray(q2)
plt.subplot(121)
plt.imshow(IM)
plt.title('2 bits')
plt.show()


##### 1 bit

# Quantize the pixel values using
6 bits
q1 = np.uint8(np.floor(data / 128)
* 128)
mse1 = np.mean((data - q1) ** 2)
print("1-bit quantization MSE:",
mse1)

IM = Image.fromarray(q1)
plt.subplot(121)
plt.imshow(IM)
plt.title('1 bits')
plt.show()
```

```
img4levels = quantize(lena,2)
plt.subplot(121)
plt.imshow(img4levels)
plt.title('4 levels')

img_arr4 = np.array(lena)
#MSE
mse = np.mean((img_arr4 -
img4levels) ** 2)
print(f"Quantization level: {4},
MSE: {mse}")

plt.show()


##### 2 levels

img2levels = quantize(lena,1)
plt.subplot(121)
plt.imshow(img2levels)
plt.title('2 levels')

img_arr2 = np.array(lena)
#MSE
mse = np.mean((img_arr2 -
img2levels) ** 2)
print(f"Quantization level: {2},
MSE: {mse}")

plt.show()
```

1-1-2

```
##############################
### 1.1.2. Create new images using
6,4,2,1 bit only for each pixel
discuss the results using mean
square error on gray scale Lena

##### original image

plt.subplot(121)
plt.imshow(lena)
plt.title('original grayscaled
image')

data = np.asarray(lena)

#convert to grayscale
plt.set_cmap('gray')
plt.show()

##### 6 bits

# Quantize the pixel values using
6 bits
q6 = np.uint8(np.floor(data / 4) *
```

```
* 0.5 + np.mean(img)

height, width = img.shape
low_contr_img = np.zeros((height,
width), dtype=np.uint8)

low_contr_img =
np.vectorize(lower_contrast)(img)
IM =
Image.fromarray(low_contr_img)
plt.imshow(IM)
plt.show()

#### raised contrast

def raised_contrast(pixel):
    return (pixel - np.mean(img))
* 1.5 + np.mean(img)


height, width = img.shape
high_contr_img = np.zeros((height,
width), dtype=np.uint8)

high_contr_img=
np.vectorize(raised_contrast)(img)
IM =
Image.fromarray(high_contr_img)
plt.imshow(IM)
plt.show()

#### invert
def invert(pixel):
    return 255 - pixel

height, width = img.shape
invert_img = np.zeros((height,
width), dtype=np.uint8)

invert_img=
np.vectorize(invert)(img)
IM = Image.fromarray(invert_img)
plt.imshow(IM)
plt.show()
```

1-1-3

```
###############################
###  1.1.3. By using point
processing create new images to
make it Darken, Lighten, Lower
contrast, raised contrast and
finally Invert image and display
the results on gray scale Lena.

img = cv2.imread('lena.bmp',
cv2.IMREAD_GRAYSCALE)

##### original image

plt.subplot(121)
plt.imshow(lena)
plt.title('original grayscaled
image')

data = np.asarray(lena)

#convert to grayscale
plt.set_cmap('gray')
plt.show()

#### Darken

def darken(pixel_value):
    return pixel_value * 0.5

height, width = img.shape
dark_img = np.zeros((height,
width), dtype=np.uint8)

dark_img =
np.vectorize(darken)(img)
IM = Image.fromarray(dark_img)
plt.imshow(IM)
plt.show()


####  lighten

def lighten(pixel):
    return pixel * 1.5

height, width = img.shape
light_img = np.zeros((height,
width), dtype=np.uint8)

light_img =
np.vectorize(lighten)(img)
IM = Image.fromarray(light_img)
plt.imshow(IM)
plt.show()

#### lower contrast

def lower_contrast(pixel):
    return (pixel - np.mean(img))
```