DSP Homework 09

Xing, Yunpeng

October 31, 2022

Contents

Looks nice.

Abstract

We've watched three videos this week about Adults, respectively, about can also grow new nerve cells, Future Computers Will Be educationdifferent and Our planet-Jungles. After watching the video, I had a profound experience and made my further thinking. In this article, I first describe the exact meanings of some nouns and analyze the error when representing Numbers. Then I propose and analyze a better way of representation. I also quantified a picture and gained a better understanding of image quantization through practice. I personally find this part difficult. Finally, in the special case of uniform distribution (p(x) = 1), I derive the optimal quantization strategy.

1 Summary and my thoughts

1.1 Adults can also grow new nerve cells

In our daily life, we often see that older people generally have poor memory. We hypothesized that it might be because adults can't grow new nerve cells. But is it reliable? In the video, neuroscientist Sandrine Thuret uses experimental data to show that the adult brain can also generate nerve cells that improve our ability to learn and remember. The production of nerve cells is closely related to the hippocampus, which produces about 700 nerve cells per day.

In addition, the video also tells us that there is a certain relationship between the regeneration of nerve cells and depression, and the regeneration of nerve cells can reduce the symptoms of depression. Sandrine Thuret says things like learning and some aerobic exercise can increase nerve cell regeneration. In addition, food intake had a similar effect, but this depended on the type of food consumed, when consumed, etc., the details of which I forgot.

1.2 Future Computers Will Be Radically Different

The video opens with a demonstration of the power of an analog computer that can predict eclipses, tides and more. Later, with the advent of transistors, digital computers developed rapidly, but now analog computers are catching up. Because the analog computer is the physical layer simulation, the operation is faster, but also more energy saving. Analog computers do not need as many transistors for an operation as digital computers.

Of course, analog computers also have certain disadvantages, it can not run like Micorosoft, can only be used for a single purpose, and its calculations are not very accurate due to the inevitable errors of the device. These shortcomings caused analog computers to fall out of favor when digital computers arrived. However, with the advent of artificial intelligence, analog computers may make a comeback. Artificial intelligence needs to calculate a large amount of data and recognize images. It takes a lot of time to use digital computers. At the same time, because of various limitations of digital computers, digital computers have reached a certain bottleneck. Image recognition requires less accuracy, as long as the recognition is correct, so this has promoted the rise of analog computers again.

At the end of the video, the specific technology of the analog computer is explained. As we can see from the video, the spring of simulation computing is here, but there are still many challenges.

1.3 Our planet-Jungles

This video is about rainforests, and it starts with the old story of landing on the moon. The video is as good and fun as ever. Although rainforests cover only about nine percent of the Earth, they are important to the planet's ecosystems, where millions of species live. We see a variety of birds and their unique ways of mating. We also see a variety of fungi and insects, as well as some large animals like leopards and monkeys. Every frame of the video is so beautiful.

At the end of the video, we still return to the theme of protecting the earth. Some human activities have a profound impact on the healthy and sustainable development of the earth. The Earth has been damaged, and the previous behavior is beyond redemption. The future will require each of us to work together to protect our homes.

1.4 My thoughts

In the first video, I would say the key thing is to get moving. We and those around us have surely seen videos about our health and memory more than once. We watch videos and feel like they're saying the right thing, but we rarely actually do them. We all need to take action to improve our memory and improve our health through exercise and more. Body is the first capital of revolution, I hope everyone can exercise actively.

The second video is about analog computer, which is closely related to our major. It can be said that we are now at the end of the era, the era needs corresponding technical personnel. I heard a saying before that pigs can be blown up in the tuyere. These are exaggerations, if the technology is not home, you will always encounter opportunities. Opportunities are for those who are prepared. Now we need to work hard to learn professional knowledge, exercise their skills, for the future preparation.

The third video is again about protecting the planet. As I wrote this further thought, I couldn't help but ask myself, am I acting, am I destroying the planet? This is a thought-provoking question. We all know to protect the earth, but there are a few people who really act. Shouting slogans is important, it is more important for each of us to act, truly protect our mother Earth.

2 Digital number representations

2.1 Problem description

This problem is about digital number representations.

(a) In general a number is stored in computer as byte, short integer, integer,float, double, quadruple types, and can be classified as either fixed-point or floating-point numbers. Describe their exact meanings.

(b) IEEE 754 is a standard commonly used for computer number representations.Describe its definition for double type floating point numbers. Analyze its error when representing numbers. Then propose and analyze a better way of representation.

2.2 Describe the exact meanings of these nouns

(1)byte:Byte can be abbreviated as B, which is a unit of measurement used in computer information technology to measure storage capacity and transmission capacity. One byte is equal to 8 bits of binary. A Chinese character consists of two bytes, and letters and numbers consist of one byte. The unit of capacity in descending order is byte (B), kilobyte (KB), Megabyte (MB), gigabit (GB), and gigabit (TB).

(2)short integer:Short integer is a type of integer. The C language does not strictly specify the degree of short integer, only makes a broad restriction, short integer takes at least two bytes, that is, at least 16 bits of signed binary, the value of which ranges from -32768 to +32767. Common numbers are usually small, so you can save memory to some extent by using short integer.

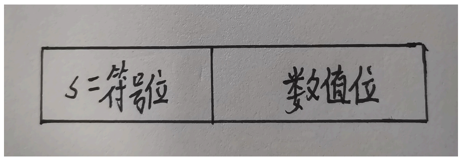
(3)integer:Integer data is numeric data that does not contain decimal parts. It is abbreviated as int and usually takes up 4 bytes, that is, 32 bits of signed binary. The value ranges from -2e31 to +(2e31-1). In practice, data is usually not so large, and if defined as int, space will be wasted. Therefore, short integer is often used. The actual length of Integer depends on the running environment, which is 4 bytes in 32-bit environment and 8 bytes in 64-bit environment.

(4)float: floating point data type is our common decimal, floating point data type can be used to store both integers and decimals, it is more accurate than the previous integer type. The Float data type is used to store single-precision floating point numbers. A single-precision value of floating-point type has four bytes, including a symbol bit, an 8-bit binary exponent, and a 23-bit mantissa. Since the high-order bit of the mantissa is always 1, it is not stored as a number. This representation provides a range between approximately -3.4e-38 and 3.4e+38 for float types.

(5)double:Double is a type of floating point data, used for double precision floating point numbers, with 8 bytes. Double is more precise and allows more decimal places, ranging from -1.7e-308 to 1.7e308.

quadruple types:Quadruple types is a new type of floating point data types. It has a higher precision of 16 bytes and ranges from -1.2e-4932 to 1.2e4932.

(7)fixed-point numbers: fixed-point number is the number of the decimal point position is fixed, that is, the number of digits behind the decimal point is Fixed. Fixed point numbers are stored in a very simple way. The usual way is to add a symbol bit to a numerical bit. The 0 of a symbol bit means a positive number, and the 1 of a symbol bit means a negative number. Since an integer number of bytes is needed to store the fixed point number, the symbol bit may be encoded in either 4 or 8 bits, depending on the situation. Note that when encoding fixed-point numbers, you do not need to encode the decimal point position because the decimal point position is fixed.



English only.

Figure 1: Fixed-point numbers

(8) floating-point numbers: floating-point numbers are numbers where the decimal point is not fixed. That is to say, unlike fixed-point numbers, floating-point numbers can have any number of decimal places. There are two types of floating point numbers: single-precision (stored in 4 bytes) and double-precision (stored in 8 bytes).

2.3 Analyze the error when representing numbers. Then propose and analyze a better way of representation.

IEEE 754 is a standard commonly used for computer number representations. Double is a type of floating-point data used for double-precision floating-point numbers and has 8 bytes. Double is more precise and allows more decimal places, ranging from -1.7e-308 to 1.7e308. This was covered in the last section. The analyze its error when representing Numbers before, first of all we need to know about the double type floating point is how to represent Numbers.

Any binary floating-point number can be reduced to the standard form $(+/-)1.f * 2^e$, where (+/-) denotes the plus or minus sign, f denotes the fractional part of signifcand, and e denotes the exponent. Binary floating point numbers (whether single precision or double precision) are stored in the computer according to the above standard form, which is divided into three parts: So a floating point number can be represented as$Value = (-1)^s 2^e (b0, b1, b2...bn)$

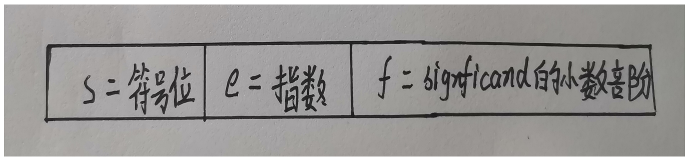


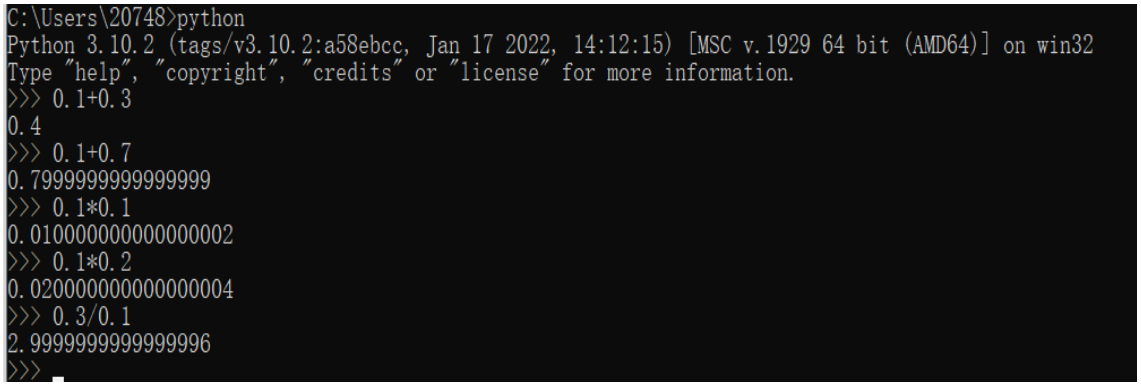
Figure 2: Floating-numbers

Double type floating point numbers is represented by 64 bit fixed length, in which 1 bit is used to represent the symbol bit, 11 bits are used to represent the exponent, and 52 bits are used to represent the mantissa. Since all values are in standard form, the 1 in 1.f can be ignored. Because 1 is omitted, we need to add 1 to the final calculation result.

2.3.1 Analyze its error when representing numbers

At present, IEEE has been around for a long time, has a wide range of applications and practicability, this point can not be questioned. However, when actually calculating, we often find that there are some differences between the actual value and the theoretical value. We usually think of it as a precision problem, which is actually caused by IEEE 754. In computers, we

use binary numbers for storage, so we convert the familiar decimal to binary before we actually do the calculations. For some decimals, when converted to binary, the number of digits is infinite, but the actual number of bits stored is finite. Therefore, it does not accurately represent the data that contains the decimal point, but actually converts the floating-point number to an approximation of the binary representation.
The following figure shows a few examples of what can go wrong:



```
C:\Users\20748>python
Python 3.10.2 (tags/v3.10.2:a58ebcc, Jan 17 2022, 14:12:15) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 0.1+0.3
0.4
>>> 0.1+0.7
0.7999999999999999
>>> 0.1*0.1
0.010000000000000002
>>> 0.1*0.2
0.020000000000000004
>>> 0.3/0.1
2.9999999999999996
>>>
```

Figure 3: Examples

The graph above shows a couple of errors that I've picked out after a lot of trying, but not all of them are wrong when you're working with decimals.

2.3.2 Propose and analyze a better way of representation

We know that binary can represent any integer exactly. So why don't we try to convert floating-point numbers to integers, then convert them to binary, then convert them to integers, then convert them to floating-point numbers? After converting to integer, the accuracy of the operation will be greatly improved, but it is relatively complex and the amount of computer calculation increases. Specific implementation can refer to the following formula:

$$\begin{aligned} V_i' &= V_i * 10^{len} && (1)\\ &= (-1)^s(b0, b1, b2...bn) && (2)\\ V_o &= (-1)^s(b0, b1, b2...bn)/10^{len} && (3)\\ &= V_o'/10^{len} && (4) \end{aligned}$$

Where $V_i$is the actual input floating point number, $V_i'$is the input integer value after conversion, $V_o'$is the output integer value, $V_o$is the output floating point value after conversion, and len is the number of digits after the decimal point of the floating point number.

3 Quantize pixel colors

3.1 Problems dscription

The following digital image uses M = 24 bits to quantize pixel colors. Use Lloyd-Max quantization algorithm to do optimal quantization for M in the range of [2, 10] and explore what you can discover, especially regarding quantization error and visual effect.

3.2 Lloyd-Max quantization algorithm

3.3 The principle of Lloyd-Max quantization algorithm

Lloyd-Max quantization algorithm is also known as the exact K-means algorithm. Its iteration process can be divided into two parts, all of which are exact calculation.
Allocation: Each sample data is assigned to the class to which the nearest class center belongs;
Update: The class center point is recalculated to update the class center point with all the data samples assigned to the class.
The specific algorithm is implemented as follows:
1 We first give an initial set of reconstructed values${y_i}^{(1)}{}_{i=1}^{M}$.Set$k = 1, D^{(0)} = 0$.Select thresholdε (2)Calculate the boundary

value. This part is to calculate the mean value between two points:

$$b_j{}^{(k)} = \frac{y_{j+1}^{(k)} + y_j^{(k)}}{2}, \quad j = 1, 2, 3, ..., M-1 \tag{5}$$

(3)Calculate the distortion:

$$D_{(k)} = \sum_{i=1}^{M} \int_{b_{j-1}^{(k)}}^{b_j^{(k)}} (x - y_i)^2 f_x(x) dx \tag{6}$$

(4) Stop when $D^{(k)} - D^{(k-1)} < \varepsilon$. Otherwise, $\varepsilon$is the above threshold;
(5) If k=k+1, the reconstructed value is recalculated.

The number of reconfigurations of the Lloyd-Max quantization algorithm is uncertain and is determined by the threshold of ε. The idea of this algorithm is simple, but it is easy to fall into local optimal solution. When the initial value is selected, there will be an initial cluster, and the average value is basically between the maximum value and the minimum value. After updating the average value every time, the cluster will only have a small fluctuation, and it is difficult to jump out of the initial range.

3.3.1 Implementation of Lloyd-Max quantization algorithm

This program is implemented by python, because lloyd's library functions are built in python, and MSE, PSNR, SSIM and other library functions are built in Python, which is more convenient for actual image processing and error analysis. Detailed code implementation is at the end of this article. In this paper, five cases such as M=2, 4, 6, 8, and 10 were selected for analysis. The quantified pictures are as figure 4.

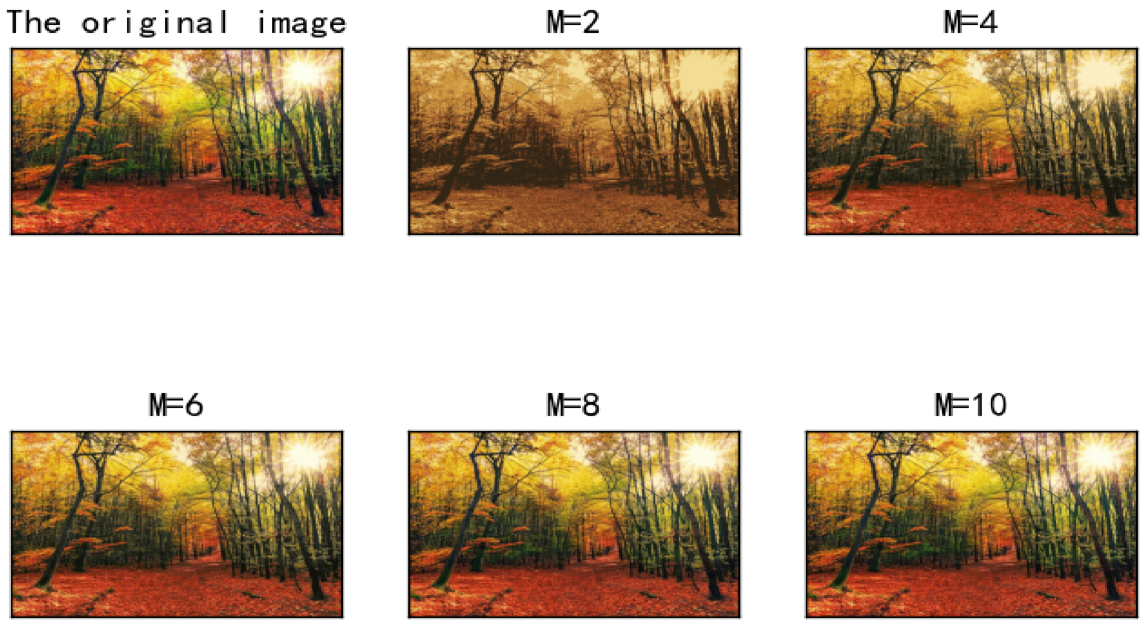


Figure 4: Images

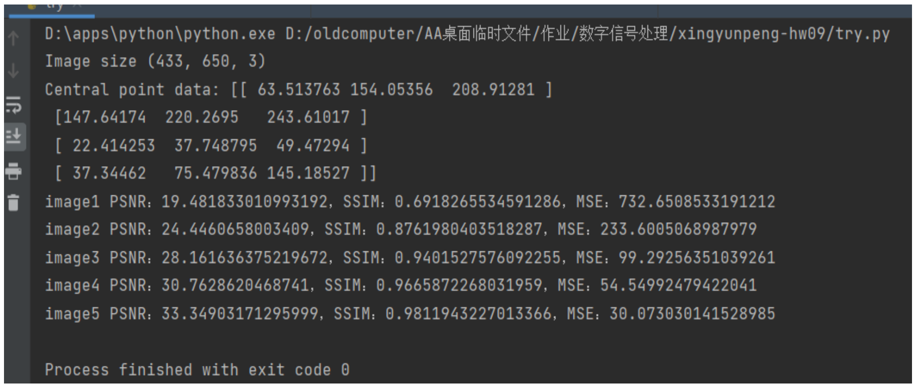
Because there are too many center points corresponding to M=4, 6, 8, and 10, only the center points of M=2 are given as examples. In addition, in order to evaluate the quantization error and image quality, the analysis of MSE, PSNR and SSIM is carried out, and the detailed results are shown as figure 5.

3.3.2 Quantization error and visual effect

Before we begin our formal error analysis, we need to know a few criteria:

(1)MSE(Mean Squared Error):MSE is the mean of the sum of squared differences between the predicted value $f(x)$and the target value y, and the formula is as follows:

$$MSE = \frac{\sum_{i=1}^{n} (f(y) - y)^2}{n} \tag{7}$$



```
D:\apps\python\python.exe D:/oldcomputer/AA桌面临时文件/作业/数字信号处理/xingyunpeng-hw09/try.py
Image size (433, 650, 3)
Central point data: [[ 63.513763 154.05356  208.91281 ]
 [147.64174  220.2695   243.61017 ]
 [ 22.414253  37.748795  49.47294 ]
 [ 37.34462   75.479836 145.18527 ]]
image1 PSNR：19.481833010993192, SSIM：0.6918265534591286, MSE：732.6508533191212
image2 PSNR：24.4460658003409, SSIM：0.8761980403518287, MSE：233.6005068987979
image3 PSNR：28.161636375219672, SSIM：0.9401527576092255, MSE：99.29256351039261
image4 PSNR：30.7628620468741, SSIM：0.9665872268031959, MSE：54.54992479422041
image5 PSNR：33.34903171295999, SSIM：0.9811943227013366, MSE：30.073030141528985

Process finished with exit code 0
```

Figure 5: Analysis

When the error is greater than 1, the error will be reduced. When it is greater than 1, it will amplify the error, so it also has a certain disadvantage. MSE is sensitive to outliers and is greatly affected by them.

(2)PSNR(Peak Signal-to-Noise Ratio):PSNR is obtained by MSE, and the formula is:

$$PSNR = 20log_{10}(\frac{MAX_i}{\sqrt{MSE}}) \tag{8}$$

Where MAX_irepresents the maximum value of the color of the image point. The larger the PSNR, the better the image quality.

(3)SSIM(structural similarity):SSIM is an index to measure the similarity of two images, which can measure the difference between the quantized image and the image. Assuming that the input two images are respectively bit x and y, then the formula can be expressed as:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\delta_{xy} + C_2)}{({\mu_x}^2{\mu_x}^2 + C_1)({\delta_x}^2{\delta_x}^2 + C_2)} \tag{9}$$

Where μ_x and $\mu_y$represent the average value of x and y, respectively; $\delta_x^2\ and\ \delta_y^2$denotes the variance of x and y, respectively; δ_{xy}denotes the covariance of x and y; C1 and C2 are constants. The larger the SSIM value, the smaller the gap between the quantized image and the original image, and the better the image quality.

According to the output data in FIG. 5, it can be clearly seen that as the value of M increases, that is, the color types become more and more, PSNR gradually increases, SSIN gradually increases, and MSE gradually increases. In other words, from the data level, the similarity between the quantized image and the original image becomes higher. However, with the increase of M, the amount of data inevitably increases, so the requirements of data quantity and image quality should be considered comprehensively in the actual quantization processing.

In terms of visual effects, we can clearly see from Figure 4 that with the increase of M, the image quality becomes better and the color becomes more bright, which is more and more similar to the original image. When M=2, there are only four colors, resulting in an image quantization that approximates a black-and-white photograph.

4 Derive the optimal quantization strategy

4.1 Problem description

Assume the random variable x ∈ [0, 1] with probability density function (pdf) p(x). We want to choose Q with M quantization points to minimize

$$J = \int_0^1 (Q(x) - x)^2 p(x) dx \tag{10}$$

In the special case of uniform distribution (p(x) = 1), derive the optimal quantization strategy.

4.2 Derivation

Because the special case of uniform distribution (p(x) = 1), we can get that

$$\begin{aligned} J &= \int_0^1 (Q(x) - x)^2 p(x)dx && (11) \\ &= int_0^1 (Q(x) - x)^2 p(x)dx && (12) \\ &= \frac{1}{3}[x - Q(x)]^3|_0^1 && (13) \\ &= \frac{1}{3}[3Q(x)^2 - 3Q(x) + 1] && (14) \\ & && (15) \end{aligned}$$

The function graph is as follows:

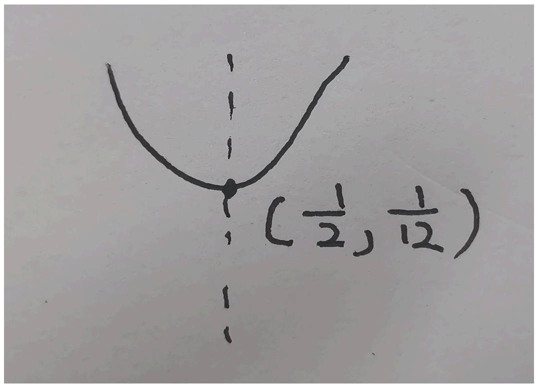


Figure 6: Function graph

So when $Q(x) = \frac{1}{2}$,J is the smallest,$J = \frac{1}{12}$

Appendix A Code listings

```
import cv2
import numpy as np
import lloyd
import matplotlib.pyplot as plt
from skimage.metrics import structural_similarity as compare_ssim
from skimage.metrics import peak_signal_noise_ratio as compare_psnr
from skimage.metrics import mean_squared_error as compare_mse
import warnings
warnings.filterwarnings("ignore")

#Read the original image
img = cv2.imread('hw09.jpg')
print('Image size',img.shape)
#Image 2D pixels are converted to 1D
data = img.reshape((-1,3))
data = np.float32(data)
#Define the center (type,max_iter,epsilon)
criteria = (cv2.TERM_CRITERIA_EPS +
cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
#Set the label and randomly select the center point
flags = cv2.lloyd_RANDOM_CENTERS
#2^2=4 categories
compactness, labels4, centers4 = lloyd(data, 4, None, criteria, 10, flags)
print('Central point data:',centers4)
#2^4=16 categories
compactness, labels16, centers16 = lloyd(data, 16, None, criteria, 10, flags)
#2^6=64 categories
compactness, labels64, centers64 = lloyd(data, 64, None, criteria, 10, flags)
```

```
#2^8=256 categories
compactness, labels256, centers256 = lloyd(data, 256, None, criteria, 10, flags)
#2^10=1024 categories
compactness, labels1024, centers1024 = lloyd(data, 1024, None, criteria, 10, flags)
#     uint8
centers4 = np.uint8(centers4)
res = centers4[labels4.flatten()]
dst4 = res.reshape((img.shape))
centers16 = np.uint8(centers16)
res = centers16[labels16.flatten()]
dst16 = res.reshape((img.shape))
centers64 = np.uint8(centers64)
res = centers64[labels64.flatten()]
dst64 = res.reshape((img.shape))
centers256 = np.uint8(centers256)
res = centers256[labels256.flatten()]
dst256 = res.reshape((img.shape))
centers1024 = np.uint8(centers1024)
res = centers1024[labels1024.flatten()]
dst1024 = res.reshape((img.shape))

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
dst4 = cv2.cvtColor(dst4, cv2.COLOR_BGR2RGB)
dst16 = cv2.cvtColor(dst16, cv2.COLOR_BGR2RGB)
dst64 = cv2.cvtColor(dst64, cv2.COLOR_BGR2RGB)
dst256 = cv2.cvtColor(dst256, cv2.COLOR_BGR2RGB)
dst1024 = cv2.cvtColor(dst1024, cv2.COLOR_BGR2RGB)

plt.rcParams['font.sans-serif']=['SimHei']

titles = [u'The original image', u'M=2', u'M=4',
u'M=6', u'M=8', u'M=10']
images = [img, dst4, dst16, dst64, dst256, dst1024]
for i in range(6):
plt.subplot(2, 3, i + 1), plt.imshow(images[i], 'gray'),
plt.title(titles[i])
plt.xticks([]),plt.yticks([])
plt.show()

for i in range(5):
psnr = compare_psnr(img, images[i+1])
ssim = compare_ssim(img, images[i+1], multichannel=True)
mse = compare_mse(img, images[i+1])
print('image{}  PSNR {} SSIM {} MSE{}'.format(i+1, psnr, ssim, mse))
```