Opencv computer vision learning (12) -- image quantization and image sampling (K-means clustering quantization, regional mosaic processing)

2020-11-27 20:01:12 【itread01】

If you need to process the original drawing and code , Please move to the small knitting GitHub Address

Portal： Please choose me

If there is a mistake in the selection： https://github.com/LeBron-Jian/ComputerVisionPractice

Prepare for： Image rotation array , Array to image

Will RGB The code for converting an image into a one-dimensional array is as follows：

```
#  Image two-dimensional pixel into one-dimensional
img = cv2.imread(filename=img_path)
data = img.reshape((-1, 3))
data = np.float32(data)
print(img.shape, data.shape)
```

We print out the results , Take a look at this：

```
(67, 142, 3) (9514, 3)
```

When we're done with it , And then convert the image back to uint8 Two dimensional type , The code is as follows：

```
#  The image changes back to uint8 Two dimensional type
centers2 = np.uint8(centers2)
res = centers2[labels2.flatten()]
dst2 = res.reshape((img.shape))

#  The image is transformed into RGB Display
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

1, Image quantization

Image is usually the objective reflection of natural scenery , And in the form of photos or video recording media continuous storage , The goal of image acquisition is to generate digital images from perceptual data , Therefore, it is necessary to discretize the continuous image data , Convert to digital images , Its work mainly includes two aspects —— Quantification and sampling . Digitizing amplitude values is called quantization , Digital coordinate values are called sampling .

Next, we will study the concept of image quantization and sampling processing , And through Python and OpenCV Realize these functions .

1.1 An overview of image quantification

Quantification （Quantization）, It is the process of converting the continuous transformation interval corresponding to the pixel brightness into a single specific value , That is, the spatial coordinate amplitude value of the original gray image is discretized . The more quantization levels , The richer the image level , The higher the gray resolution , The better the quality of the image ; The lower the level of quantification , The image level is not rich , The lower the gray resolution , There will be layered image contour , Reduced the quality of the image , The following figure is to convert the continuous gray value of an image to 0 To 255 The grayscale process of .

边栏推荐

1 C++ 数字、string和char*的...
2 C++学习——centos7上部署...
3 C++学习——一步步学会写...
4 C++学习——临时对象的产...
5 C++学习——对象的引用的...
6 C++编程经验（6）：使用C+...
7 Won the CKA + CKS certific...
8 C + + number, string and c...
9 C + + Learning -- capacity()...
10 C + + Learning -- about co...

猜你喜欢

C + + programming experience (6): using C + + style type conversion

Latest party and government work...

在线身份证号码提取生日工具

Online ID number extraction birthday tool

野指针？悬空指针？ 一文带你搞懂！

Field pointer? Dangling pointer? This article will help you...

HCNA Routing&Switching之GVRP

GVRP of hcna Routing & Switching

Seq2Seq实现闲聊机器人

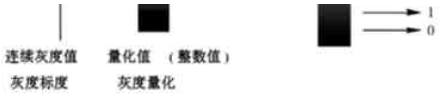
【闲聊机器人】seq2seq模型的原理

随机推荐

LeetCode 91. 解码方法

Seq2seq implements chat robot

[chat robot] principle of



After quantification , The image is represented as an integer matrix , Each pixel has two properties : Position and grayscale . Position by line , Column means . The grayscale is an integer representing the brightness and darkness of the pixel position . This number matrix M*N As a computer processing object , The gray level is usually 0~255（8bit quantitative ）. If the quantization level is 2, Two gray levels are used to represent the pixels of the original image （0~255）, The gray value is less than 128 Of 0, Greater than equal to 128 Of 128; If the quantization level is 4, Four gray levels are used to represent the pixels of the original image , The new image will be layered into four colors ,0~64 The interval is taken as 0,64~128 The interval is taken as 64,128~192 Take the interval 128,192~255 The interval is taken as 192, And so on .

1.1.1 Image color quantization （ Subtractive treatment ） Introduction

RGB The pixel value of is in 0~255 Between , What do we do when we want to represent an image with less memory space ? The first is the subtraction process , Use the image as 32, 96, 160, 224 This 4 Pixel values represent . The image will be from 256^3 Compress to 4^3,RGB Only take the value of {32,96,160,224}, This is called color quantization .

$$val = \begin{Bmatrix} 32 & (0 <= var < 64) \\ 96 & (64 <= var < 128) \\ 160 & (128 <= var < 192) \\ 224 & (192 <= var < 256) \end{Bmatrix}$$

1.2 Code implementation of image quantization

Let's learn Python Image quantization processing related code in oh , The core process is to create a temporary picture , Then loop through all the pixels in the original image , Determine the quantization level that each pixel should belong to , Finally, a temporary video will be displayed , The following code learns to convert grayscale images into two quantization levels .

The code is as follows :

```
#_*_coding:utf-8_*_
import cv2
import numpy as np
import matplotlib.pyplot as plt

#  Read the picture
img = cv2.imread('kd2.jpg')
#  Get the height and width of the image
height, width = img.shape[0], img.shape[1]

#  Build an image , Content is zero filled
new_img = np.zeros((height, width, 3), np.uint8)

#  Image quantization operation , The quantification level is 2
for i in range(height):
    for j in range(width):
        for k in range(3):  #  Correspondence BGR Three channels
            if img[i, j][k] < 128:
                gray = 0
            else:
                gray = 129
            new_img[i, j][k] = np.uint8(gray)

#  Show the image
cv2.imshow('src', img)
cv2.imshow('new', new_img)
#  Wait for the display
cv2.waitKey(0)
cv2.destroyAllWindows()
```

The figure below shows the output , It divides images into two levels of quantification .

seq2seq model
Leetcode 91. Decoding method
HCNA Routing＆Switching之GVRP
GVRP of hcna Routing & Switching
HDU7016 Random Walk 2
[Code+ #1]Yazid 的新生舞会
CF1548C The Three Little Pigs
HDU7033 Typing Contest
HDU7016 Random Walk 2
[code + 1] Yazid's freshman ball
CF1548C The Three Little Pigs
HDU7033 Typing Contest
Qt Creator 自动补齐变慢的解决
HALCON 20.11：如何处理标定助手品质问题
HALCON 20.11：标定助手使用注意事项
Solution of QT creator's automatic replenishment slowing down
Halcon 20.11: how to deal with the quality problem of calibration assistant
Halcon 20.11: precautions for use of calibration assistant
“十大科学技术问题”揭晓！|青年科学家50²论坛
"Top ten scientific and technological issues" announced| Young scientists 50 ² forum
求反转链表
Reverse linked list
js的数据类型
JS data type
记一次文件读写遇到的bug
Remember the bug encountered in reading and writing a file
单例模式
Singleton mode
在这个 N 多编程语言争霸的世界，C++ 究竟还有没有未来?
In this world of N programming languages, is there a future for C + +?
es6模板字符
js Promise
js 数组方法 回顾
ES6 template characters
js Promise
JS array method review
【Golang】走进 Go 语言 第一课 Hello World

[golang] go into go language lesson 1 Hello World

The following code compares the quantization levels to 2, 4, 8 The quantitative processing effect of .

```
# _*_coding:utf-8_*_
import cv2
import numpy as np
import matplotlib.pyplot as plt

#  Read the picture
img = cv2.imread('kd2.jpg')
#  Get the height and width of the image
height, width = img.shape[0], img.shape[1]

#  Build an image , Content is zero filled
new_img1 = np.zeros((height, width, 3), np.uint8)
new_img2 = np.zeros((height, width, 3), np.uint8)
new_img3 = np.zeros((height, width, 3), np.uint8)

#  Image quantization operation , The quantification level is 2
for i in range(height):
    for j in range(width):
        for k in range(3):  #  Correspondence BGR Three channels
            if img[i, j][k] < 128:
                gray = 0
            else:
                gray = 129
            new_img1[i, j][k] = np.uint8(gray)

#  Image quantization operation , The quantification level is 4
for i in range(height):
    for j in range(width):
        for k in range(3):  #  Correspondence BGR Three channels
            if img[i, j][k] < 64:
                gray = 0
            elif img[i, j][k] < 128:
                gray = 64
            elif img[i, j][k] < 192:
                gray = 128
            else:
                gray = 192
            new_img2[i, j][k] = np.uint8(gray)

#  Image quantization operation , The quantification level is 8
for i in range(height):
    for j in range(width):
        for k in range(3):  #  Correspondence BGR Three channels
            if img[i, j][k] < 32:
                gray = 0
            elif img[i, j][k] < 64:
                gray = 32
            elif img[i, j][k] < 96:
                gray = 64
            elif img[i, j][k] < 128:
                gray = 96
            elif img[i, j][k] < 160:
                gray = 128
            elif img[i, j][k] < 192:
                gray = 160
            elif img[i, j][k] < 224:
```

```
                gray = 192
            else:
                gray = 224
            new_img3[i, j][k] = np.uint8(gray)

#  Used to display Chinese tags normally
plt.rcParams['font.sans-serif'] = ['SimHei']

#  Show the image
titles = [u'(a) The original image ', u'(b) quantitative L2', u'(c) quantitative L4', u'(d) quantitative
images = [img, new_img1, new_img2, new_img3]
for i in range(4):
    plt.subplot(2, 2, i+1), plt.imshow(images[i])
    plt.title(titles[i])
    plt.xticks([]), plt.yticks([])
plt.show()
```

The results are as follows :

1.3, An overview of image segmentation and clustering

1.3.1 The definition of image segmentation

Image segmentation is an important preprocessing method for image recognition and computer vision . Image segmentation is to divide the image into several specific , Areas with unique properties and the technology and process of proposing objects of interest . It is the key step from image processing to image analysis . The existing image segmentation methods are mainly divided into the following categories： Threshold based segmentation method , Region based segmentation method , The segmentation method based on the edge and the segmentation method based on the specific theory, etc . From a mathematical point of view , Image segmentation is the process of dividing digital images into disjoint regions . The process of image segmentation is also a marking process , That is, the pixels belonging to the same area are assigned the same number .

1.3.2 The definition of clustering

A cluster is a collection of data objects , Objects in a collection are similar to each other , The objects in each set are quite different from each other . The process of organizing similar objects in a group of physical or abstract objects into several groups is called clustering process . However, in the process of clustering , The data type of the object we are concerned with is in addition to the commonly used interval value property , Binary properties , Symbols , Sequence , Scale value attribute , Mixed type properties, etc , There may be images , News , Multimedia materials such as video . For traditional data types, there are many mature methods to calculate distance , These methods include： European style ,Manhattan,Minkowski Distance formula , Binary variable distance comparison formula and so on . However, for multimedia data objects , Because of its particularity , There is no good algorithm to classify them .

Clustering is a process of dividing a data set into several clusters or classes , And make the data objects in the same cluster have high similarity , The data objects in different groups are not similar . The measure of similarity or dissimilarity is determined based on the value of the data object description property . It's usually the use of （ Between objects ） Distance to describe .

The following is a theoretical method of image segmentation , Through K-Means Clustering algorithm to achieve image segmentation or color stratification processing .

1.4,K-Means Clustering quantitative processing

K-Means The principle of clustering quantitative processing

K-Means Clustering is the most commonly used clustering algorithm , It originated in signal processing , The purpose is to divide data points into K Clusters , Find the center of each cluster and minimize its metrics . The greatest advantage of this algorithm is that it is simple , In order to understand , It's faster , The disadvantage is that it can only be applied to continuous data , And the number of clusters should be specified before clustering .

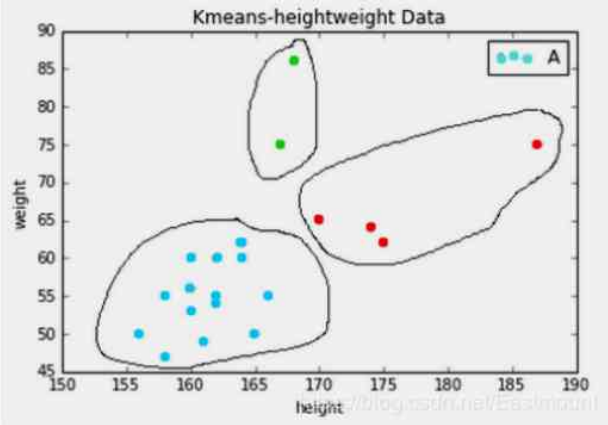
If you want to know K-Means Principle of algorithm , Please refer to my blog ：

Python Machine learning notes ： K-Means Algorithm ,DBSCAN Algorithm

Here is K-Means The analysis flow of clustering algorithm , The steps are as follows ：

First step , Make sure K value , That is, data aggregation and integration K Clusters or groups of classes .
The second step , Select randomly from the data set K Data points as the centroid （Centroid） Or the data center .
The third step , Calculate the distance from each point to each centroid separately , And divide each point into a group that is closest to the centroid , With the center of mass .
Step four , When each center of mass gathers some points , Redefine the algorithm and select a new centroid .
Step five , Compare the new centroid with the old centroid , If the distance between the new centroid and the old centroid is less than a certain threshold , It means that the position of the center of mass is not changed much , Convergence stability , It is considered that clustering has achieved the desired results , Algorithm termination .
Step six , If the new centroid and the old centroid change greatly , That is, the distance is greater than the threshold , Then continue iterating through steps 3 to 5 , Until the algorithm stops .

The following is a clustering algorithm for height and weight , The population of the data set was clustered into three groups .



1.4.2 K-Means Clustering opencv Source code

stay opencv in ,KMeans() The function prototype is shown below ：

```
retval, bestLabels, centers = kmeans(data, K, bestLabels, criteria, attempts,
flags[, centers])
```

The meaning of a variable in a function ：

data Represents clustering data , It is best to np.flloat32 Type N Set of dimension points , The reason is np.float32 The reason is that this data type is fast , Under the same information, if it is uint Type data will be particularly slow .
K Represents the number of clusters in a cluster ,opencv Of kmeans Classification requires a known number of classifications .
bestLabels Represents the default category label , That is, the output integer array , The cluster label index used to store each sample , If not None
criteria Indicates the termination condition of the algorithm , The maximum number of iterations or the required accuracy . In some iterations , Once the movement of each cluster center is less than criteria.epsilon, The algorithm will stop , A triplet containing a number of iterations , The format is （type, max_iter, epsilon）, among type There are two options ：

——cv2.TERM_CRITERIA_EPS： Accuracy （ Error ） Satisfaction epsilon stop it

——cv2.TERM_CRITERIA_MAX_ITER： The number of iterations exceeds max_iter stop it

——cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, The two fit together , Any satisfaction ends .

attempts Repeat test kmeans The number of algorithms , The algorithm returns the label that produces the best compactness
flags Represents the choice of the initial center , Two ways are cv2.KMEANS_PP_CENTERS ; and cv2.KMEANS_RANDOM_CENTERS
centers The output matrix representing the cluster center , Each cluster center is a row of data

1.4.3 K-Means Clustering segmentation of gray images

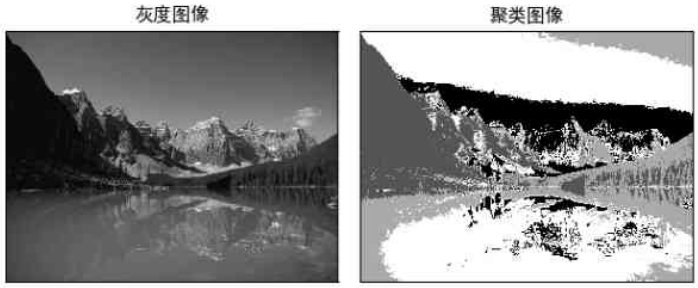
In image processing , Through K-Means Image clustering algorithm can be implemented , Image clustering , Image recognition and other operations , The following is mainly used for image color segmentation . Suppose there is a picture of 100*100 The grayscale image of a pixel , It consists of 10000 One RGB Grayscale composition , We passed K-Menas These pixels can be clustered into K A cluster , Then the centroid points in each cluster are used to replace all pixel points in the cluster , In this way, the color of compressed image can be quantized without changing the resolution , Realize image color level segmentation .

Next, we use this method to segment the gray image color , We need to pay attention to , In progress K-Means Before clustering , Need to put RGB Pixel points are converted into a one-dimensional array , Let's talk about the various forms of color gathering together , Form the final color segmentation .

```
# _*_coding:utf-8_*_
# coding: utf-8
import cv2
import numpy as np
import matplotlib.pyplot as plt

#  Read the original image gray color
img = cv2.imread('scenery.jpg', 0)
print(img.shape)

#  Get image height 、 Width
rows, cols = img.shape[:]

#  Image two-dimensional pixel into one-dimensional
data = img.reshape((rows * cols, 1))
data = np.float32(data)

#  Defining the center  (type,max_iter,epsilon)
criteria = (cv2.TERM_CRITERIA_EPS +
            cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)

#  Set the label
flags = cv2.KMEANS_RANDOM_CENTERS

# K-Means Clustering   Gather and integrate 4 Class
compactness, labels, centers = cv2.kmeans(data, 4, None, criteria, 10, flags)

#  Generate the final image
dst = labels.reshape((img.shape[0], img.shape[1]))

#  Used to display Chinese tags normally
plt.rcParams['font.sans-serif'] = ['SimHei']

#  Show the image
titles = [u' Gray image ', u' Clustering images ']
images = [img, dst]
for i in range(2):
    plt.subplot(1, 2, i + 1), plt.imshow(images[i], 'gray'),
    plt.title(titles[i])
    plt.xticks([]), plt.yticks([])
plt.show()
```

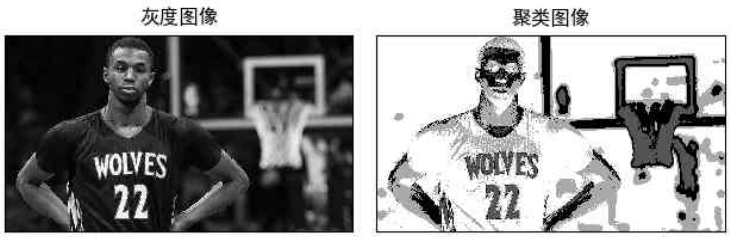
The original picture is as follows :

The output is as follows , On the left is the grayscale , On the right is K-Means The image after clustering , It brings together similar colors or areas .



The figure is as follows：

The effect is as follows：



1.4.4 K-Means Clustering contrast segmentation color image

The following code is to color image segmentation processing , It clusters color images into 2 Class ,4 Class sum 64 Class .

```
# coding: utf-8
import cv2
import numpy as np
import matplotlib.pyplot as plt

#  Read the original image
= cv2.imread('scenery.jpg')
```

```
print(img.shape)

#  Image two-dimensional pixel into one-dimensional
data = img.reshape((-1, 3))
data = np.float32(data)

#  Defining the center  (type,max_iter,epsilon)
criteria = (cv2.TERM_CRITERIA_EPS +
            cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)

#  Set the label
flags = cv2.KMEANS_RANDOM_CENTERS

# K-Means Clustering   Gather and integrate 2 Class
compactness, labels2, centers2 = cv2.kmeans(data, 2, None, criteria, 10, flags)

# K-Means Clustering   Gather and integrate 4 Class
compactness, labels4, centers4 = cv2.kmeans(data, 4, None, criteria, 10, flags)

# K-Means Clustering   Gather and integrate 8 Class
compactness, labels8, centers8 = cv2.kmeans(data, 8, None, criteria, 10, flags)

# K-Means Clustering   Gather and integrate 16 Class
compactness, labels16, centers16 = cv2.kmeans(data, 16, None, criteria, 10, flags)

# K-Means Clustering   Gather and integrate 64 Class
compactness, labels64, centers64 = cv2.kmeans(data, 64, None, criteria, 10, flags)

#  The image changes back to uint8 Two dimensional type
centers2 = np.uint8(centers2)
res = centers2[labels2.flatten()]
dst2 = res.reshape((img.shape))

centers4 = np.uint8(centers4)
res = centers4[labels4.flatten()]
dst4 = res.reshape((img.shape))

centers8 = np.uint8(centers8)
res = centers8[labels8.flatten()]
dst8 = 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))

#  The image is transformed into RGB Display
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
dst2 = cv2.cvtColor(dst2, cv2.COLOR_BGR2RGB)
dst4 = cv2.cvtColor(dst4, cv2.COLOR_BGR2RGB)
dst8 = cv2.cvtColor(dst8, cv2.COLOR_BGR2RGB)
dst16 = cv2.cvtColor(dst16, cv2.COLOR_BGR2RGB)
dst64 = cv2.cvtColor(dst64, cv2.COLOR_BGR2RGB)

#  Used to display Chinese tags normally
plt.rcParams['font.sans-serif'] = ['SimHei']

#  Show the image
titles = [u' The original image ', u' Clustering images  K=2', u' Clustering images  K=4',
          u' Clustering images  K=8', u' Clustering images  K=16', u' Clustering images  K=64']
images = [img, dst2, dst4, dst8, dst16, dst64]
for i in range(6):
    plt.subplot(2, 3, i + 1), plt.imshow(images[i], 'gray'),
    plt.title(titles[i])
```

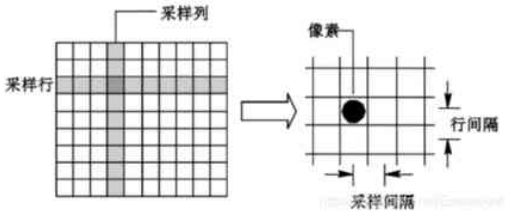
```
    plt.xticks([]), plt.yticks([])
plt.show()
```

We still use the picture above：

2, Image sampling

2.1 An overview of image sampling processing

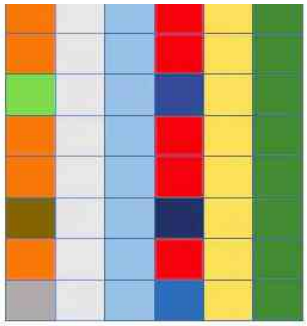
Image sampling （Image Sampling） It's about dividing a continuous image into M*N Grid , Each grid is represented by a luminance or grayscale value , The schematic diagram is as follows：

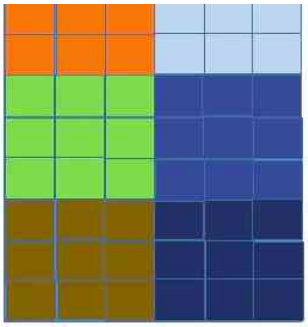


The greater the interval between image sampling , The less the prime number of the image , The lower the spatial resolution , The worse the image quality , There are even mosaic benefits ; contrary , The smaller the interval between image sampling , The more primes the image painting gets , The higher the spatial resolution , The better the image quality , But the amount of data will increase accordingly .

Mosaic principle： Replace the pixel value of the selected area in the image with a pixel value or random value in the selected area .

In the figure below, the pixel value of the specified area will be specified , Change all to the pixel value of the first point in the upper left corner：



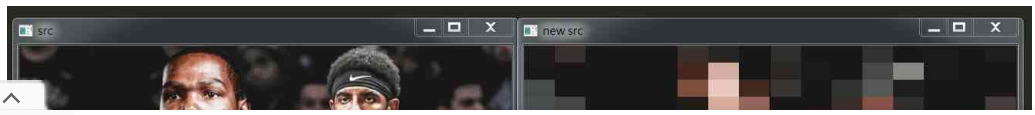


2.2 Image sampling Python Realize

Let's learn Python Image sampling processing related code operations . The core process is to create a temporary picture , Set the size of the area to be used （ Such as 16*16）, Then loop through all the pixels in the original image , The pixel points in the sampling area are assigned the same value （ For example, the gray value of the pixel in the upper left corner ）, Finally, image sampling processing is realized . The code is for 16*16 The sampling process .

```
# _*_coding:utf-8_*_
import cv2
import numpy as np
import matplotlib.pyplot as plt

#  Read the picture
img = cv2.imread('kd2.jpg')

#  Get the height and width of the image
height, width = img.shape[0], img.shape[1]
# print(img.shape)  # (352, 642, 3)
#  The sample is converted to  16*16  Region
numHeight, numWidth = height / 16, width / 16

#  Build an image , Content is zero filled
new_img1 = np.zeros((height, width, 3), np.uint8)

#  Image loop sampling  16*16  Region
for i in range(16):
    #  Get Y coordinates
    y = int(i * numHeight)
    for j in range(16):
        #  Get  X  coordinates
        x = int(j * numWidth)
        #  Get fill color , Pixel dot in upper left corner
        b = img[y, x][0]
        g = img[y, x][1]
        r = img[y, x][2]

        #  Loop setting small area sampling
        for n in range(int(numHeight)):
            for m in range(int(numWidth)):
                new_img1[y+n, x+m][0] = np.uint8(b)
                new_img1[y+n, x+m][1] = np.uint8(g)
                new_img1[y+n, x+m][2] = np.uint8(r)

#  Show the image
cv2.imshow('src', img)
cv2.imshow('new src', new_img1)
#  Wait for the display
cv2.waitKey(0)
cv2.destroyAllWindows()
```

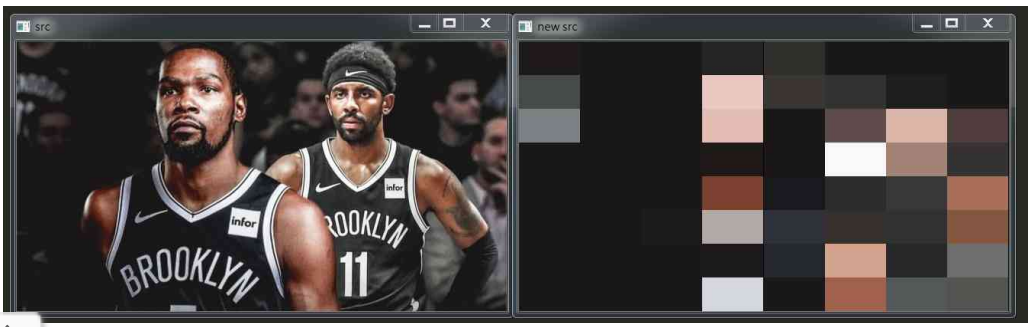
The results are as follows ：



Again , You can sample color pictures , The following code processes the color landscape sampling into 8*8 The mosaic area of .

```
# _*_coding:utf-8_*_
import cv2
import numpy as np
import matplotlib.pyplot as plt

#  Read the picture
img = cv2.imread('kd2.jpg')

#  Get the height and width of the image
height, width = img.shape[0], img.shape[1]
# print(img.shape)  # (352, 642, 3)
#  The sample is converted to  8*8  Region
numHeight, numWidth = height / 8, width / 8

#  Build an image , Content is zero filled
new_img1 = np.zeros((height, width, 3), np.uint8)

#  Image loop sampling  8*8  Region
for i in range(8):
    #  Get Y coordinates
    y = int(i * numHeight)
    for j in range(8):
        #  Get  X  coordinates
        x = int(j * numWidth)
        #  Get fill color , Pixel dot in upper left corner
        b = img[y, x][0]
        g = img[y, x][1]
        r = img[y, x][2]

        #  Loop setting small area sampling
        for n in range(int(numHeight)):
            for m in range(int(numWidth)):
                new_img1[y+n, x+m][0] = np.uint8(b)
                new_img1[y+n, x+m][1] = np.uint8(g)
                new_img1[y+n, x+m][2] = np.uint8(r)

#  Show the image
cv2.imshow('src', img)
cv2.imshow('new src', new_img1)
#  Wait for the display
cv2.waitKey(0)
cv2.destroyAllWindows()
```

The results are as follows :



But there is a problem with the above code , It's when the length and width of the image can't be divided by the sampling area , The far right and bottom areas of the output image are not sampled . Let's do a calculation here , The area that cannot be divided is also sampled .

2.3 Regional mosaic processing

I learned to sample the whole image , So how to mosaic the regional region of the image ?

When you drag with the left mouse button , Put a mosaic on the path of the mouse , The principle of mosaics is to cover a pixel in the selected area with a pixel in the selected area , In order to prevent the mouse from repeating the same time in the image , Choose a different pixel , The program will input the image when , The mosaic effect of the whole picture is realized . And when the mouse goes by , The program just copies the specified position of the mosaic image to the displayed image .

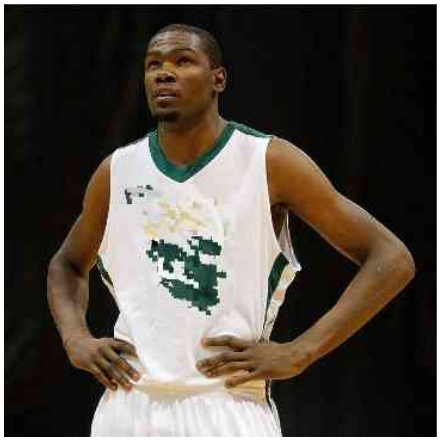
The following code implements this function , When the mouse is pressed , It's able to mosaic the area the mouse is dragging , And press "s" Key to store the image locally .

The code is as follows :

```
# _*_coding:utf-8_*_
import cv2
import numpy as np
import matplotlib.pyplot as plt


#  Mouse event
def draw(event, x, y, flags, parma):
    global en
    #  Left click to open  en  Key
    if event == cv2.EVENT_LBUTTONDOWN:
        en = True
    #  Left click and move
    elif event == cv2.EVENT_MOUSEMOVE and flags == cv2.EVENT_LBUTTONDOWN:
        #  Call function to play mosaic
        if en:
            drawMask(y, x)
        #  Click the left mouse button to end the operation
        elif event == cv2.EVENT_LBUTTONUP:
            en = False


#  Image regionalism adopts operation
def drawMask(x, y, size=10):
    # size*size  Sample processing
    m = int(x / size * size)
    n = int(y / size * size)
    # 10*10  The region is set to the same pixel value
    for i in range(int(size)):
        for j in range(int(size)):
            img[m+i][n+j] = img[m][n]

if __name__ == '__main__':
    #  Read the original image
    img = cv2.imread('durant.jpg')
    #  Right click to open
    en = False

    #  Open dialog box
    cv2.namedWindow('image')
    #  call draw  Function to set the mouse operation
    cv2.setMouseCallback('image', draw)

    #  Loop back
    while(1):
        cv2.imshow('image', img)
        #  Press  ESC Key out
        if cv2.waitKey(10) & 0xFF == 27:
            break
        #  Press  s  Key to save pictures
```

```
    elif cv2.waitKey(10) & 0xFF == 115:
        cv2.imwrite('save.jpg', img)
#  Exit window
cv2.destroyAllWindows()
```

The mosaic pictures are as follows：



I coded his number .

References： https://blog.csdn.net/Eastmount/article/details/89218513

https://blog.csdn.net/Eastmount/article/details/8928