

Lesson 12 Image Processing --- Thresholding

1. Image Binaryzation

Image binaryzation is to set the gray values of image pixels as two values, 0 (represent black) and 255 (represent white) generally, which will make the image turn black-and-white.

Commonly, it will process the grayscale image, and then set a threshold to divide the image into two parts, including the part greater than the threshold and the part smaller than the threshold. Next, assign different pixel values to these two parts of image.

Image binaryzation facilitates the further processing of the image, makes the image simple, reduces the amount of data and highlights the target contour you are interested in.

Under different situation, there are three ways to execute thresholding, including Global Thresholding, Adaptive Thresholding and Otsu Thresholding

2. Global Thresholding

Global thresholding will process the whole image according to the set threshold.



2.1 Operation Steps

Note:

1) **Before operation, please copy the routine “threshold_demo.py” and sample picture “test.jpg” in “4.OpenCV->Lesson 12 Image Processing --- Thresholding->Routine Code” to the shared folder.**

2) For how to configure the shared folder, please refer to the file in “2. Linux Basic Lesson->Lesson 3 Linux Installation and Source Replacement”.

3) The input command should be case sensitive and the keywords can be complemented by “Tab” key.

1) Open virtual machine and start the system. Click “”, and then “” or press “**Ctrl+Alt+T**” to open command line terminal.

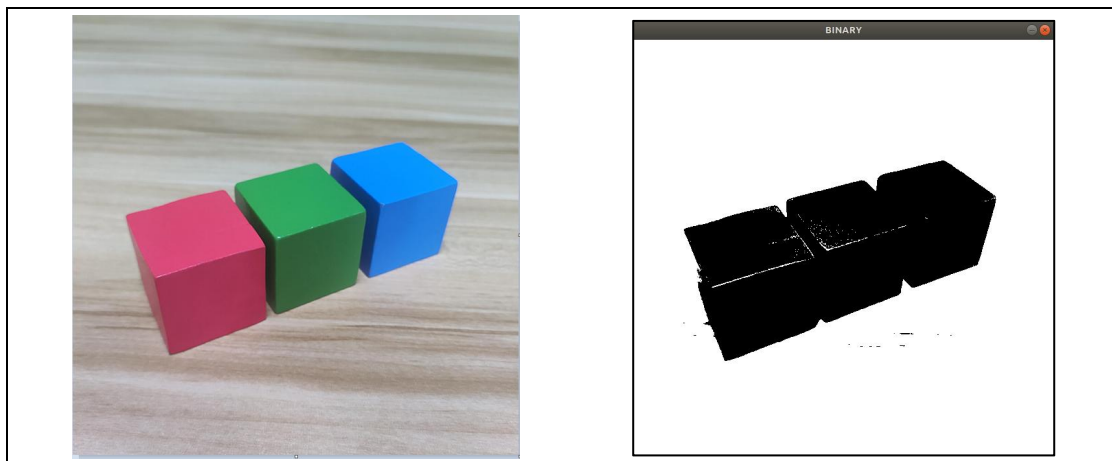
2) Input command “**cd /mnt/hgfs/share/**” and press Enter to enter the shared folder.

```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/
```

3) Input command “**python3 threshold_demo.py**” and press Enter to run the routine.

```
ubuntu@ubuntu-virtual-machine:/mnt/hgfs/share$ python3 threshold_demo.py
```

2.2 Program Outcome



The final output picture is as shown above.

2.3 Code Analysis

The routine “**threshold_demo.py**” can be found in “4. OpenCV Computer

Vision Lesson->Lesson 12 Image Processing --- Thresholding->Routine Code”.

```
1 import cv2
2 img=cv2.imread('test.jpg')
3 img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
4 ret, img2 = cv2.threshold(img_gray, 127, 255, cv2.THRESH_BINARY)
5 #the threshold is 127
6 #set maxval as 255. And the output image after processing is black-and-white image
7 cv2.imshow("BINARY", img2)
8 cv2.waitKey(0)
9 cv2.destroyAllWindows()
```

Import Module: Import cv2 module

Read Picture: Call imread function to read picture and the parameter stands for the name of the picture.

Color Space Conversion: call cvtColor function to convert the image into GRAY color space, and the parameter indicates the picture and conversion mode.

```
ret, img2 = cv2.threshold(img_gray, 127, 255, cv2.THRESH_BINARY)
```

Thresholding: use threshold function to execute. The specific format and parameters are as follow.

threshold(src, thresh, maxval, type)

- 1) The first parameter “**src**” is the pending image.
- 2) The second parameter “**thresh**” is the set threshold.
- 3) The third parameter “**maxval**” will be set only when the type is “**THRESH_BINARY**” or “**THRESH_BINARY_INV**”. It refers to the new value assigned when the gray values of the picture pixels are greater (smaller) than the threshold
- 4) The fourth parameter “**type**” represents the type of thresholding.
 - ◆ cv2.THRESH_BINARY: the part greater than threshold is set as maxval, otherwise 0.

◆ cv2.THRESH_BINARY_INV: the part greater than threshold is set as 0, and the rest is maxval.

◆ cv2.THRESH_TRUNC: the part greater than threshold is set as threshold, and the part smaller than threshold remains unchanged.

◆ cv2.THRESH_TOZERO: the part greater than threshold is set as threshold, and the part smaller than threshold is set as 0.

◆ cv2.THRESH_TOZERO_INV: the part greater than threshold is set as 0, and the part smaller than threshold remains unchanged.

```
cv2.imshow("BINARY", img2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Display image: Call imshow function to display image and the parameter in bracket refers to the title of the window and displayed image.

Close window: waitKey function will wait until the keyboard is pressed, and then execute destroyAllWindows function to close the window.

3.Adaptive Thresholding

Adaptive thresholding is the method where the threshold value is calculated for smaller regions to process the image.



3.1Operation Steps

Note:

1) Before operation, please copy the routine “adaptiveThreshold_demo.py” and sample picture “test.jpg” in “4.OpenCV->Lesson 12 Image Processing --- Thresholding->Routine Code” to the shared folder.

2) For how to configure the shared folder, please refer to the file in “2. Linux Basic Lesson->Lesson 3 Linux Installation and Source Replacement”.

3) The input command should be case sensitive and the keywords can be complemented by “Tab” key.

1) Open virtual machine and start the system. Click “”, and then “” or press “**Ctrl+Alt+T**” to open command line terminal.

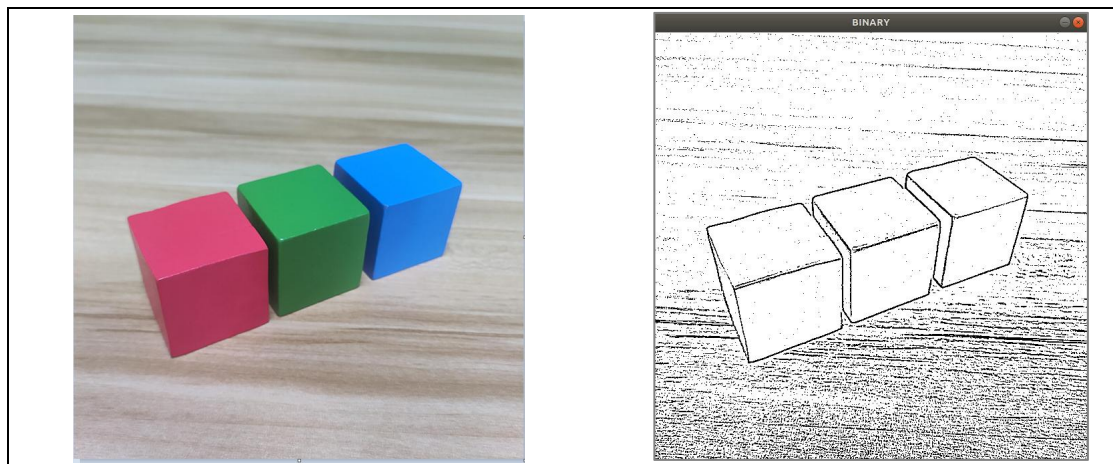
2) Input command “**cd /mnt/hgfs/share/**” and press Enter to enter the shared folder.

```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/
```

3) Input command “**python3 adaptiveThreshold_demo.py**” and press Enter to run the routine.

```
ubuntu@ubuntu-virtual-machine:/mnt/hgfs/share$ python3 adaptiveThreshold_demo.py
```

3.2 Program Outcome



The final output image is as above.

3.3 Code Analysis

For a picture with balanced color, its threshold is usually set as 127.

However, when the color of the image is out of balance, setting the threshold as 127 will make the output image bad. Therefore we need to turn to other thresholding methods.

The routine “**adaptiveThreshold_demo.py**” can be found in “**4. OpenCV Computer Vision Lesson->Lesson 12 Image Processing---Thresholding->Routine Code**”.

```
import cv2
img=cv2.imread('test.jpg')
img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
img2 = cv2.adaptiveThreshold(img_gray, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY,5,3)
cv2.imshow("BINARY", img2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Import Module: Import cv2 module

Read Image: Call imread function to read picture and the parameter stands for the name of the picture

Color Space Conversion: call cvtColor function to convert the image into GRAY color space and the parameter indicates the picture and conversion mode.

```
img2 = cv2.adaptiveThreshold(img_gray, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY,5,3)
```

Thresholding: use **adaptiveThreshold** function to execute. The specific format and parameters are as follow.

adaptiveThreshold(src, maxValue, adaptiveMethod, thresholdType, blockSize, C)

- 1) The first parameter “**src**” refers to the pending image.
- 2) The second parameter “**maxValue**” will be set only when the type is “cv2.THRESH_BINARY” or “cv2.THRESH_BINARY_INV”. It refers to the new value assigned when the gray values of the picture pixels are greater (smaller) than the threshold

3) The third parameter “**adaptiveMethod**” decides how the adaptive threshold value is calculated. And the specific methods are as follow.

◆ **cv2.ADAPTIVE_THRESH_MEAN_C**: set the weight value of all pixels in the neighborhood as the same.

◆ **cv2.ADAPTIVE_THRESH_GAUSSIAN_C**: Obtain the weight value of all the pixels in the neighborhood through Gaussian formula. The weight value is associated with the distance between pixels in each neighbor and target pixel. The shorter the distance, the greater the weight value. The longer the distance, the smaller the weight value.

4) The fourth parameter “**thresholdType**” indicates the type of the thresholding, which is combined with **maxValue** to use. This parameter only can be set as **cv2.THRESH_BINARY** or **cv2.THRESH_BINARY_INV**.

5) The fifth parameter “**blockSize**” represents the size of the neighbour area. It is generally set as 3, 5, 7, ect.

6) The sixth parameter “**C**” is a constant. The threshold is mean or weight value minus this constant.

```
cv2.imshow("BINARY", img2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Display image: Call imshow function to display image and the parameter in bracket refers to the title of the window and displayed image.

Close window: waitKey function will wait until the keyboard is pressed, and then execute destroyAllWindows function to close the window.

4. Otsu Thresholding

Appropriate threshold will be automatically calculated.



4.1 Operation Steps

Note:

1) Before operation, please copy the routine “Otsu_demo.py” and sample picture “test.jpg” in “4.OpenCV->Lesson 12 Image Processing --- Thresholding->Routine Code” to the shared folder.

2) For how to configure the shared folder, please refer to the file in “2. Linux Basic Lesson->Lesson 3 Linux Installation and Source Replacement”.

3) The input command should be case sensitive and the keywords can be complemented by “Tab” key.

1) Open virtual machine and start the system. Click “”, and then “” or press “**Ctrl+Alt+T**” to open command line terminal.

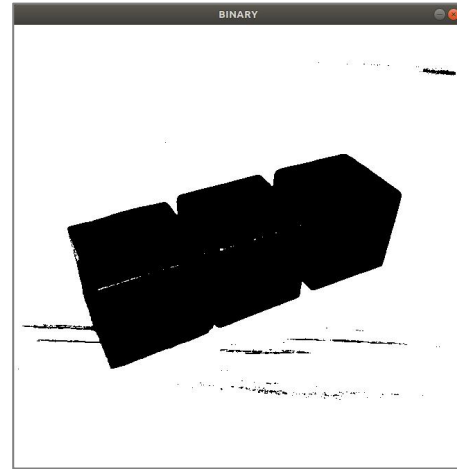
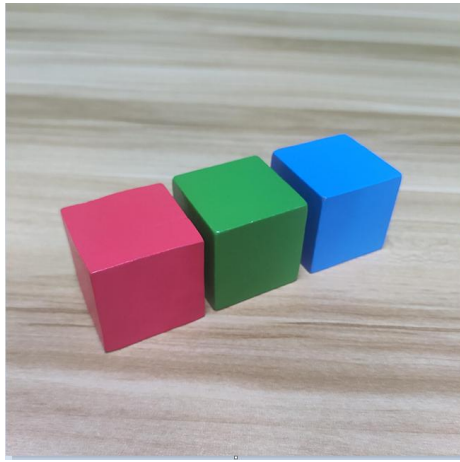
2) Input command “**cd /mnt/hgfs/share/**” and press Enter to enter the shared folder.

```
hiwonder@ubuntu:~$ cd /mnt/hgfs/share/
```

3) Input command “**python3 Otsu_demo.py**” and press Enter to run the routine.

```
ubuntu@ubuntu-virtual-machine:/mnt/hgfs/share$ python3 Otsu_demo.py
```


4.2 Program Outcome



The final output picture is as above.

4.3 Code Analysis

The Otsu thresholding, also known as the maximum inter-class variance method, is a method where the inter-class variance is calculated by assigning pixels into two or more classes. When the variance reaches the maximum value, the class dividing line i.e. the gray value is used as the image segmentation threshold.

The routine “**Otsu_demo.py**” can be found in “**4. OpenCV Computer Vision Lesson->Lesson 12 Image Processing---Thresholding->Routine Code**”.

```
import cv2
img=cv2.imread('test.jpg')
img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
ret, img2 = cv2.threshold(img_gray, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)
cv2.imshow("BINARY", img2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Otsu thresholding is to pass a parameter “**cv2.THRESH_OTSU**” in the parameter type of **threshold** function, so as to realize Otsu threshold segmentation.

In `cv2.threshold(img, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)`, parameter thresh needs to set as 0, and parameter type should be set as `"cv2.THRESH_BINARY+cv2.THRESH_OTSU"`