

12 图像处理——阈值处理

1. 图像二值化

图像二值化就是将图像上的像素点的灰度值设置为两个值，一般为 0（表示黑色）和 255（表示白色），可以将整个图像呈现出明显的黑白效果。

最常用的方法就是先将图像灰度处理，然后设定一个阈值，用该阈值将图像分成两个部分，即大于阈值的部分和小于阈值的部分，然后再将两部分图像分别赋予不同像素值。

图像二值化有利于图像的进一步处理，使图像变得简单，并且减少了数据量，可以凸显出感兴趣的目标轮廓。

阈值处理根据不同的情况，可以选择三种处理方式：全局阈值处理，自适应阈值处理和 Otsu 阈值处理。

2. 全局阈值处理

全局阈值处理会按照设定好的阈值来处理整个图像。



2.1 实验步骤

注意：

1) 需要先将目录“第四章 OpenCV 计算机视觉学习->图像处理进阶篇->第 12 课 图像处理——阈值处理->例程源码”下的例程“threshold_demo.py”和示例图片“test.jpg”复制到共享文件夹。

2) 共享文件夹的配置方法可查看目录“第 2 章 Linux 系统简介及使用入门->Linux 基础课程->第 3 课 Linux 系统安装及换源方法”下的文档。

3) 输入指令时需严格区分大小写，且可以使用“Tab”键补齐关键字。

1) 打开虚拟机，启动系统。点击系统任务栏的图标，并点击图标，或使用快捷键“Ctrl+Alt+T”，打开命令行终端。

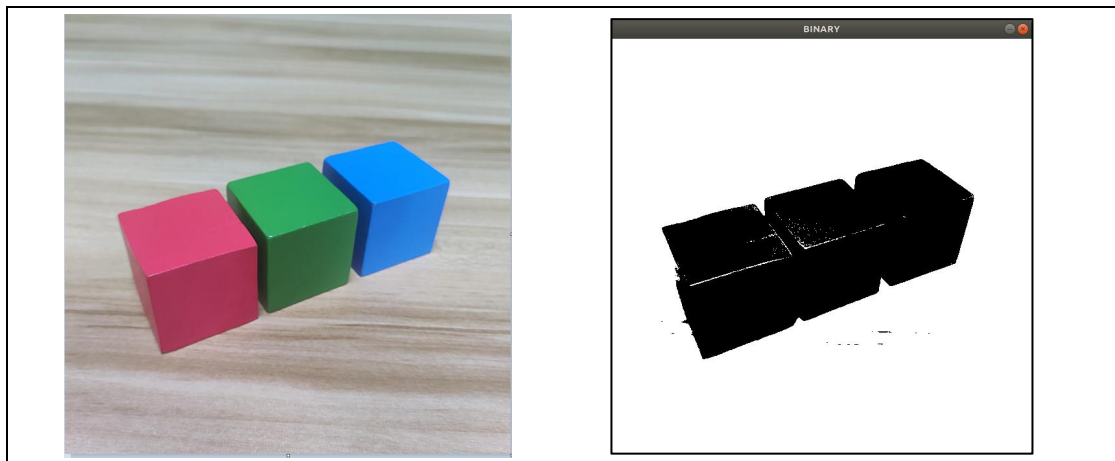
2) 输入指令“`cd /mnt/hgfs/share/`”，并按下回车，进入共享文件夹。

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

3) 输入指令“`python3 threshold_demo.py`”，并按下回车，运行例程。

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

2.2 实现效果



执行程序后，图像经过全局阈值处理就会呈现如上图所示的效果。

2.3 代码分析

可在目录“第四章 OpenCV 计算机视觉学习->图像处理进阶篇->第 12 课 图像处理——阈值处理->例程源码”下查看例程“`threshold_demo.py`”。

```
import cv2
img=cv2.imread('test.jpg')
img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
ret, img2 = cv2.threshold(img_gray, 127, 255, cv2.THRESH_BINARY)
#这里阈值为127
#maxval设为255, 所以处理后的图像是黑白图像
cv2.imshow("BINARY", img2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

导入模块：导入 `cv2` 模块。

读取图片：`imread` 函数读取图片，参数为图片名称。

颜色空间转换：`cvtColor` 函数转换成 `GRAY` 颜色空间，参数为转换的图片和转换模式。

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

阈值处理：使用函数 **threshold**，具体格式和参数如下：

threshold(src, thresh, maxval, type)

1) 第一个参数“**src**”，是要阈值处理的图像。

2) 第二个参数“**thresh**”，是设定的阈值。

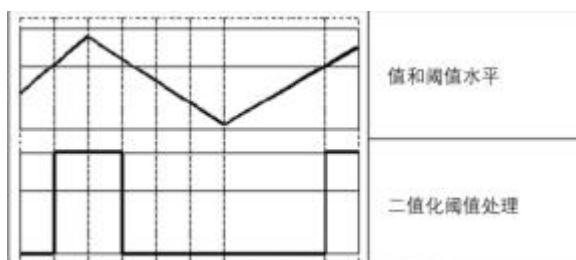
3) 第三个参数“**maxval**”，当 **type** 指定为 **THRESH_BINARY** 或 **THRESH_BINARY_INV** 时，才需要设置该值，指的是高于（低于）阈值时赋予的新值。

4) 第四个参数“**type**”，阈值的处理方法，具体如下：

◆ **cv2.THRESH_BINARY**：超过阈值部分取 **maxval**，否则取 0；

此方法会将原始图像处理为仅有两个值的二值图像，对于灰度值大于阈值的像素点，将其灰度值设定为最大值。对于灰度值小于或等于阈值的像素点，将其灰度值设定为 0。

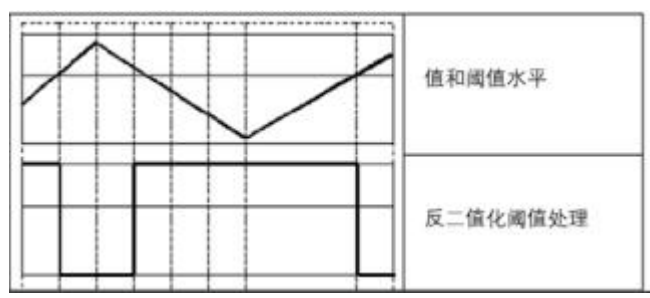
处理效果，经过图形化表示后，如图所示：



◆ **cv2.THRESH_BINARY_INV**：大于阈值的部分变为 0，其他部分变为 **maxval**；

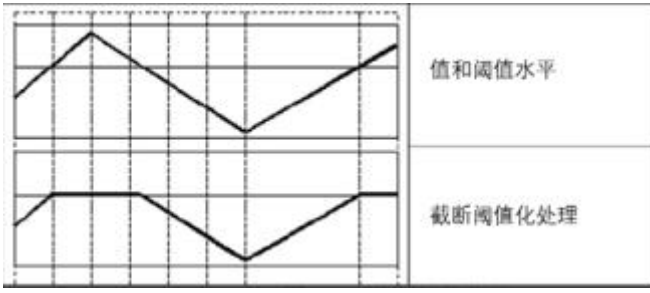
此方法会将原始图像处理为仅有两个值的二值图像，对于灰度值大于阈值的像素点，将其灰度值设定为 0。对于灰度值小于或等于阈值的像素点，将其灰度值设定为最大值。

处理效果，经过图形化表示后，如图所示：



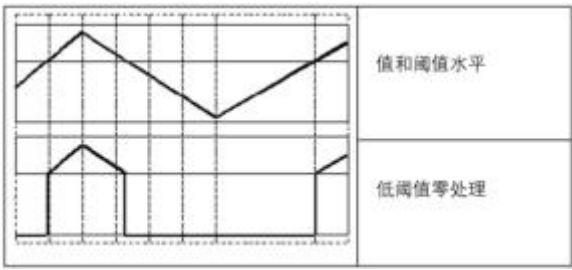
- ◆ `cv2.THRESH_TRUNC`: 大于阈值部分设为阈值，否则不变，相当于进行一个截断；
此方法会将图像中大于阈值的像素点的值设定为阈值，小于或等于该阈值的像素点的值保持不变。

处理效果，经过图形化表示后，如图所示：



- ◆ `cv2.THRESH_TOZERO`: 大于阈值部分不改变，否则设为 0；
此方法会将图像中小于或等于阈值的像素点的值处理为 0, 大于阈值的像素点的值保持不变。即先选定一个阈值，然后对于像素值大于阈值的像素点，其值将保持不变。对于像素值小于或等于阈值的像素点，其值将被处理为 0。

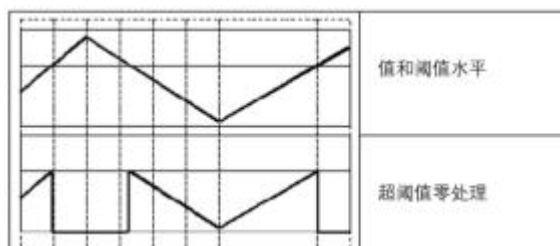
处理效果，经过图形化表示后，如图所示：



- ◆ `cv2.THRESH_TOZERO_INV`: 大于阈值的部分变为 0，其余部分不变。
会将图像中大于阈值的像素点的值处理为 0, 小于或等于该阈值的像素点的值保持不变。即先选定一个阈值，然后对于像素值大于阈值的像素点，其像素值将被处理为 0。

对于像素值小于或等于阈值的像素点，其像素值将保持不变。

处理效果，经过图形化表示后，如图所示：



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

显示图像：imshow 函数显示图像，参数为显示窗口的标题和显示的图像。

关闭窗口：waitKey 函数等待按键按下后，执行 destroyAllWindows 函数关闭窗口。



3.自适应阈值处理

自适应阈值会局部计算每一个区域阈值来处理图像。

3.1 实验步骤

注意：

- 1) 需要先将目录“第四章 OpenCV 计算机视觉学习->图像处理进阶篇->第 12 课 图像处理——阈值处理->例程源码”下的例程“adaptiveThreshold_demo.py”和示例图片“test.jpg”复制到共享文件夹。
 - 2) 共享文件夹的配置方法可查看目录“第 2 章 Linux 系统简介及使用入门->Linux 基础课程->第 3 课 Linux 系统安装及换源方法”下的文档。
 - 3) 输入指令时需严格区分大小写，且可以使用“Tab”键补齐关键字。
-

- 1) 打开虚拟机，启动系统。点击系统任务栏的图标，并点击图标，或使用快捷键“Ctrl+Alt+T”，打开命令行终端。

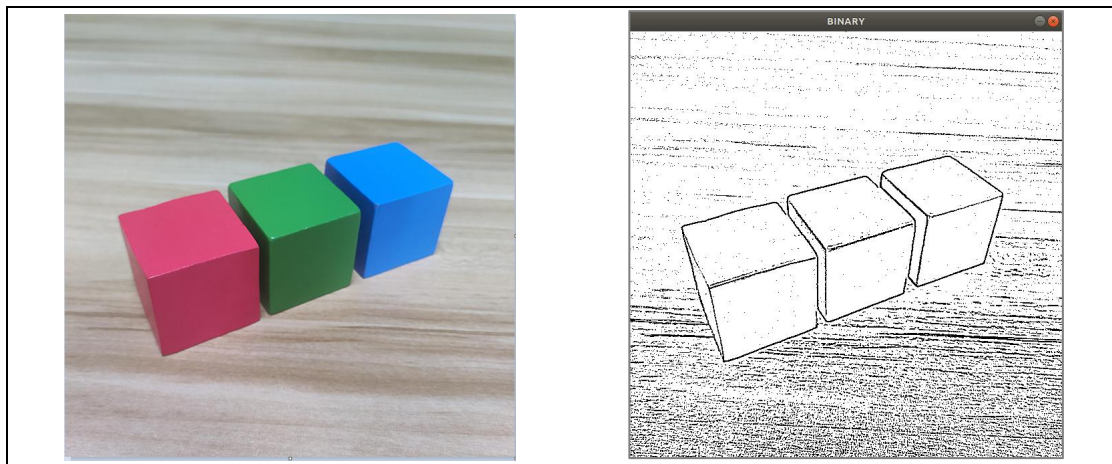
2) 输入指令“`cd /mnt/hgfs/share/`”，并按下回车，进入共享文件夹。

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

3) 输入指令“`python3 adaptiveThreshold_demo.py`”，并按下回车，运行例程。

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

3.2 实现效果



执行程序后，图像经过自适应阈值处理就会呈现如上图所示的效果。

3.3 代码分析

对于一幅色彩均衡的图像，一般直接将阈值设为 127，这个阈值是经验获得的，设置这个阈值比较合适。

但是，当有的图像色彩分布不均衡时，用 127 这个阈值，效果就会很差，此时就要考虑别的阈值处理方法，比如自适应阈值处理方法或者 Otsu 方法，而自适应阈值就是在图像上每一个小区域计算与其对应的阈值。

可在目录“第四章 OpenCV 计算机视觉学习->图像处理进阶篇->第 12 课 图像处理——阈值处理->例程源码”下查看例程“`adaptiveThreshold_demo.py`”。

```
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()
```

导入模块：导入 cv2 模块。

读取图片：imread 函数读取图片，参数为图片名称。

颜色空间转换：cvtColor 函数转换成 GRAY 颜色空间，参数为转换的图片和转换模式。

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

阈值处理：使用函数 **adaptiveThreshold**，具体格式和参数如下：

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

- 1) 第一个参数 “**src**”，是要阈值处理的图像。
- 2) 第二个参数 “**maxValue**”，当参数 thresholdType 为 cv2.THRESH_BINARY 或者 cv2.THRESH_BINARY_INV 时，该参数表示高于（低于）阈值时赋予的新值。
- 3) 第三个参数 “**adaptiveMethod**”，计算自适应阈值的方法，具体方法如下：
 - ◆ cv2.ADAPTIVE_THRESH_MEAN_C：这种方法是将邻域所有像素点的权重值都取一样；
 - ◆ cv2.ADAPTIVE_THRESH_GAUSSIAN_C：这种方法是通过高斯公式获得邻域所有像素点的权重值。这种权重值的选取就跟各邻域像素点到目标像素点的距离有关，距离越近权重越高，距离越远权重越低。
- 4) 第四个参数 “**thresholdType**”，表示阈值处理方式，这个参数是和 maxValue 一起搭配使用的。这个参数只能选择 cv2.THRESH_BINARY 或者 cv2.THRESH_BINARY_INV 中的一个。
- 5) 第五个参数 “**blockSize**”，表示邻域尺寸的大小，一般取 3，5，7 等。
- 6) 第六个参数 “**C**”，是常量，阈值等于平均值或者加权平均值减去这个常数。

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

显示图像：imshow 函数显示图像，参数为显示窗口的标题和显示的图像。

关闭窗口：waitKey 函数等待按键按下后，执行 destroyAllWindows 函数关闭窗口。

4.Otsu 阈值处理

Otsu 阈值会自动计算出合适的阈值。



4.1 实验步骤

注意：

1) 需要先将目录“第四章 OpenCV 计算机视觉学习->图像处理进阶篇->第 12 课 图像处理——阈值处理->例程源码”下的例程“Otsu_demo.py”和示例图片“test.jpg”复制到共享文件夹。

2) 共享文件夹的配置方法可查看目录“第 2 章 Linux 系统简介及使用入门->Linux 基础课程->第 3 课 Linux 系统安装及换源方法”下的文档。

3) 输入指令时需严格区分大小写，且可以使用“Tab”键补齐关键字。

1) 打开虚拟机，启动系统。点击系统任务栏的图标，并点击图标，或使用快捷键“Ctrl+Alt+T”，打开命令行终端。

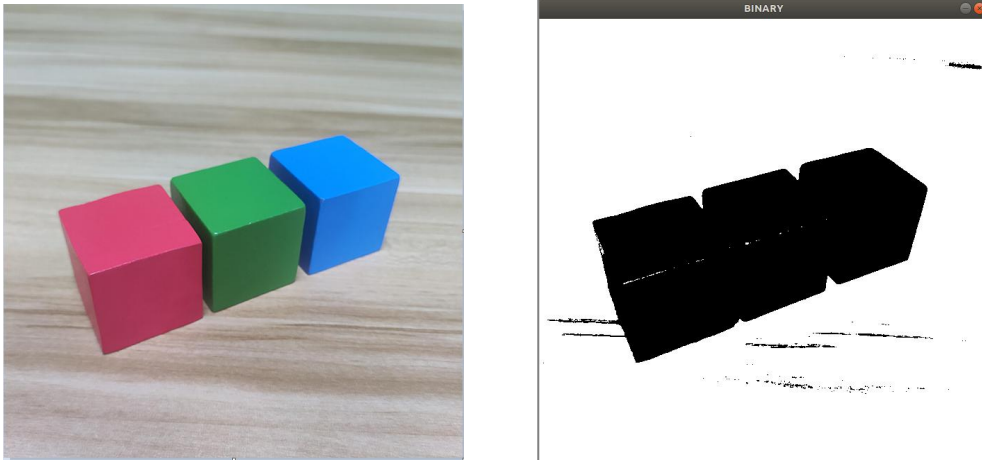
2) 输入指令“cd /mnt/hgfs/share/”，并按下回车，进入共享文件夹。

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

3) 输入指令“python3 Otsu_demo.py”，并按下回车，运行例程。

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


4.2 实现效果



执行程序后，图像经过 Otsu 阈值处理就会呈现如上图所示的效果。

4.3 代码分析

Otsu 方法又称最大类间方差法，通过把像素分配为两类或多类，计算类间方差，当方差达到最大值时，类分割线（即灰度值）就作为图像分割阈值

可在目录“第四章 OpenCV 计算机视觉学习->图像处理进阶篇->第 12 课 图像处理——阈值处理->例程源码”下查看例程“Otsu_demo.py”。

```
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 阈值处理就是在 threshold 函数的参数 type 中，多传入一个参数 cv2.THRESH_OTSU 就可实现 Otsu 阈值分割，例如：

cv2.threshold(img, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)，这里就要把 thresh 参数设为 0，type 参数设为“cv2.THRESH_BINARY+cv2.THRESH_OTSU”。