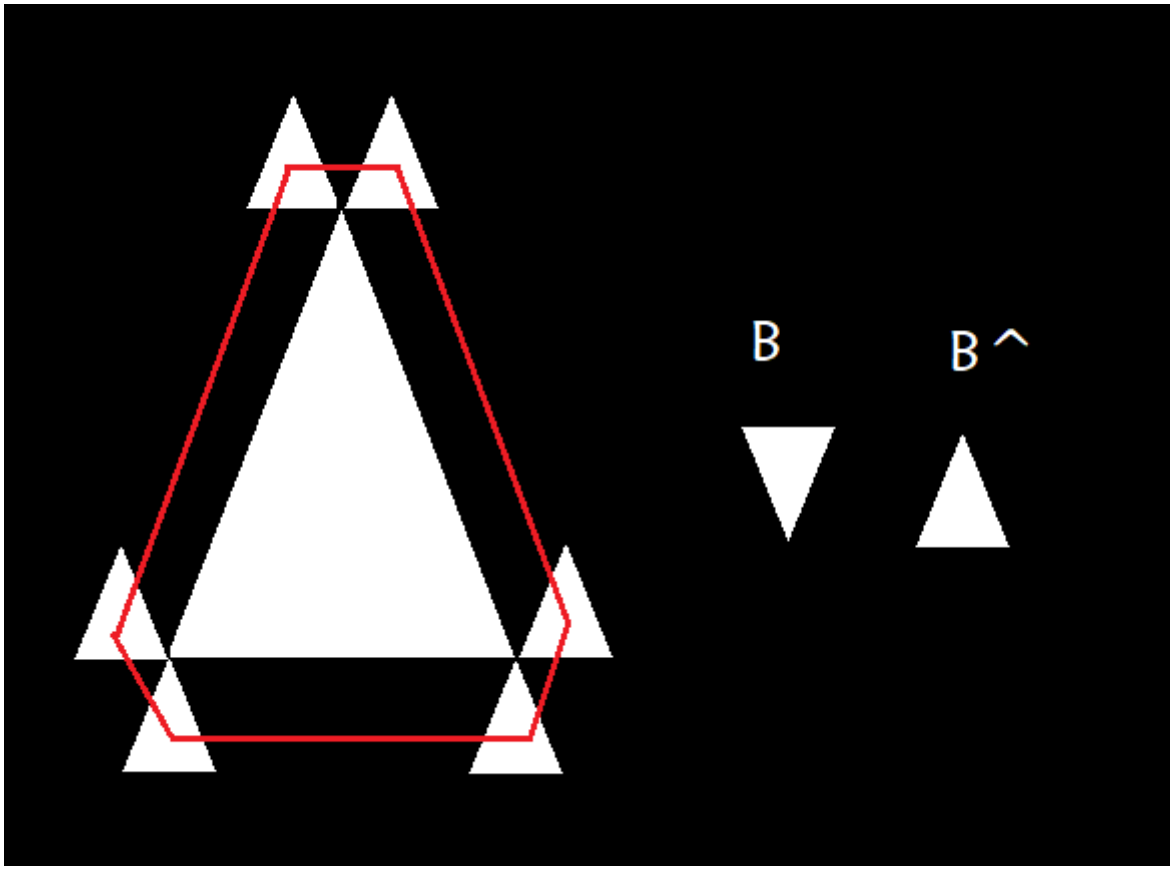
先回顾一下膨胀腐蚀的定义

dilate 膨胀

$$A \oplus B = \{ x \mid [(\hat{B})_x \cap A] \neq \varnothing \}$$

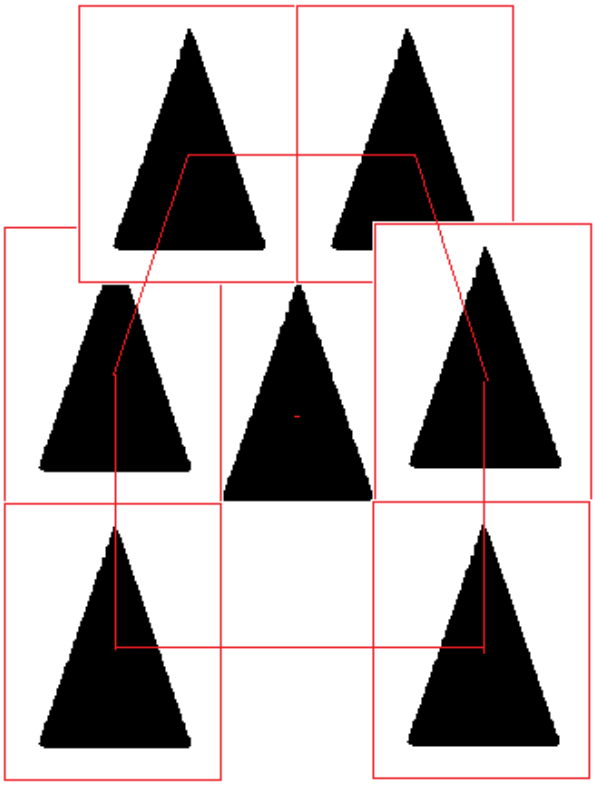
膨胀过程:B首先做关于原点的映射B^，然后平移x。A被B的膨胀是被所有x平移后与A至少有一个非零公共元素。

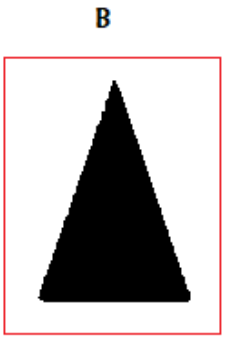


erode 腐蚀

$$A \Theta B = \{ x \mid (B)_x \subseteq A \}$$

也就是说A被B的腐蚀的结果为所有使B被x平移后包含于A的点x的集合。

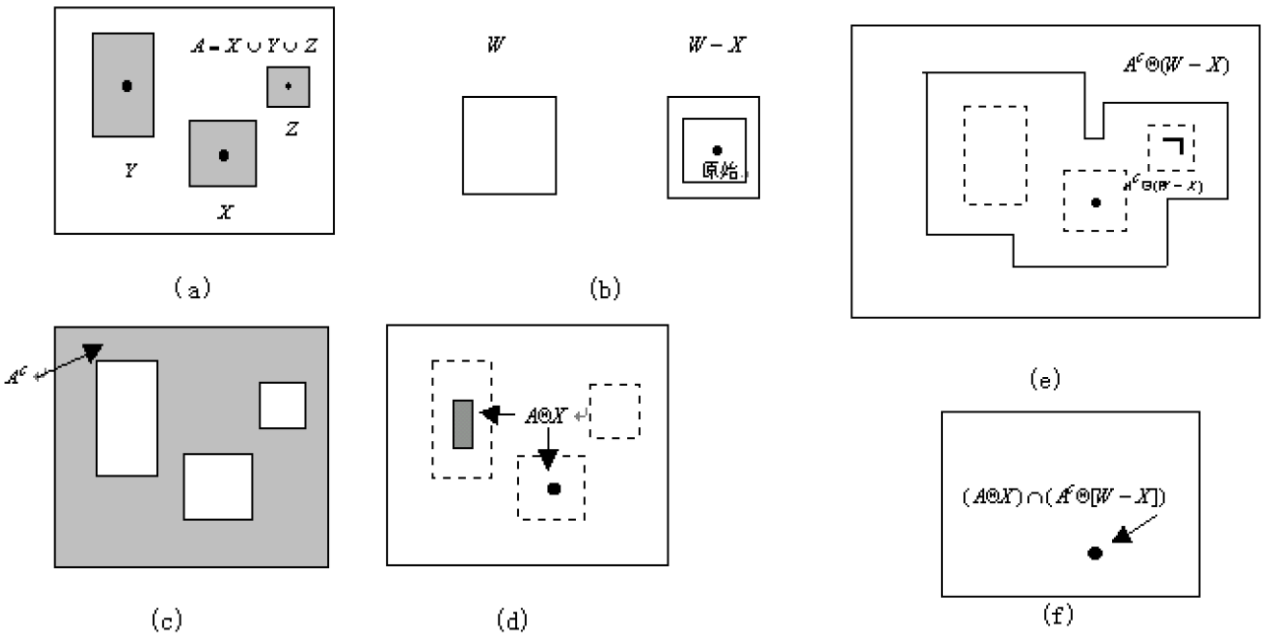




注意：图中中心点也有个点，其实由此可以推出HMT：
当把三角形看做A时把上图A则为A^c，上图结构元B则是B1的补集B2，则上图可以表示为$(A^c \Theta B_2)$，而
$(A \Theta B_1)$
的结果中A的中心点保留，则两个交集为B在A中的匹配，记为：
$A \otimes B = (A \Theta B_1) \cap (A^c \Theta B_2)$

HMT

这幅图更为清楚：



让每个图形的原点位于它的重心。如果用一个小窗口W包含X，X关于W的本地背景是图(b)中的集合差(W-X)。图©为集合A的补。图(d)示出A被X腐蚀的结果。A被X的腐蚀在X中只有X的原点，这样X才能完全包含于A。图(e)表示集合A的补被本地背景集合(W-X)的腐蚀；外围阴影区域也是腐蚀结果的一部分。

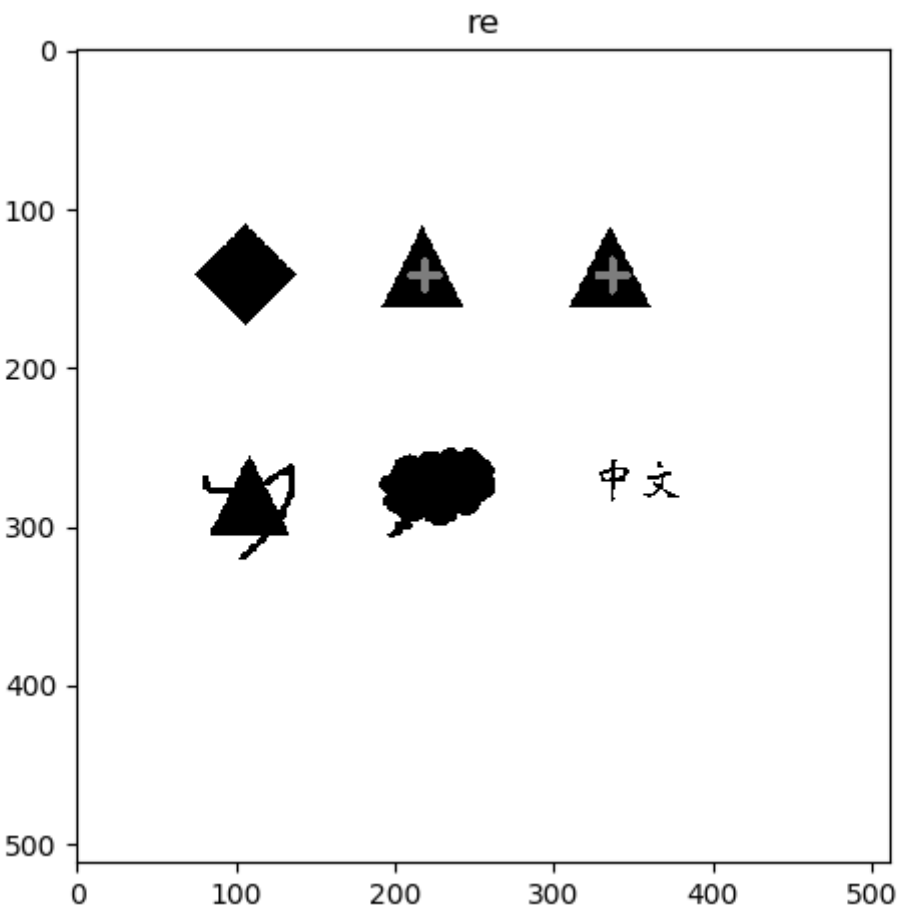
实现

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# @Time    : 2019/11/23 12:39
# @Author  : ZhigangJiang
# @File    : mian.py
# @Software: PyCharm
# @Description:hit and miss translate

import cv2
import matplotlib.pyplot as plt
import numpy as np


def plt_show_opcv(title, image):
    if image.shape.__len__() == 3:
        plt.imshow(image[:, :, ::-1])
    else:
        plt.imshow(image, cmap='gray')
    plt.title(title)
```

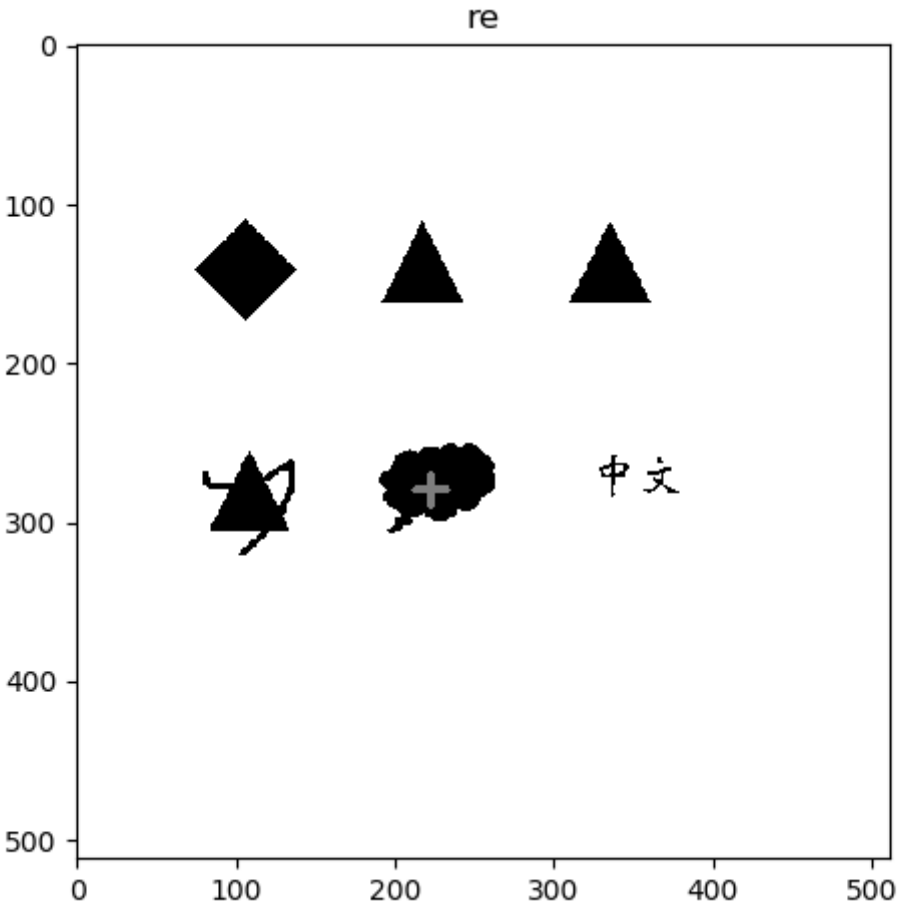
```
    plt.show()


def pme(titles, images, rc=None):
    row = None
    col = None
    if rc is None:
        length = titles.__len__()
        row = int(np.sqrt(length))
        col = int(length / row)
        if length - row - col > 0:
            row += 1
    else:
        row = rc[0]
        col = rc[1]

    for i in range(titles.__len__()):
        plt.subplot(row, col, i + 1), plt.imshow(images[i], 'gray')
        plt.title(titles[i])
        plt.xticks([]), plt.yticks([i])
    plt.show()


def hmt(a, b):
    b1 = ~b
    b2 = b
    a1 = ~a
    a2 = a
    pme(["b1", "b2", "x1", "x2"],
        [b1, b2, a1, a2])
    x1_erode_b1 = cv2.erode(a1, b1)
    x2_erode_b2 = cv2.erode(a2, b2)
    plt_show_opcv("a1_erode_b1", x1_erode_b1)
    plt_show_opcv("a1_erode_b1_", cv2.dilate(x1_erode_b1, np.ones((10, 10), np.u
    plt_show_opcv("a2_erode_b2", x2_erode_b2)
    plt_show_opcv("a2_erode_b2_", cv2.dilate(x2_erode_b2, np.ones((10, 10), np.u
    r = cv2.bitwise_and(x1_erode_b1, x2_erode_b2)
    return r


image_X = cv2.imread("images/X.png", 0)
image_B = cv2.imread("images/B_triangle.png", 0)

ret1, image_X = cv2.threshold(image_X, 127, 255, cv2.THRESH_BINARY)
ret2, image_B = cv2.threshold(image_B, 127, 255, cv2.THRESH_BINARY)

plt_show_opcv("X", image_X)
```

```
plt_show_opcv("B", image_B)
re = hmt(image_X, image_B)

targets = []
for i in range(re.shape[0]):
    for j in range(re.shape[1]):
        if re[i][j]:
            targets.append((j, i))
            print(i, j)

for target in targets:
    image_X = cv2.drawMarker(image_X, target, 125, markerType=cv2.MARKER_CROSS,
plt_show_opcv("re", image_X)
```

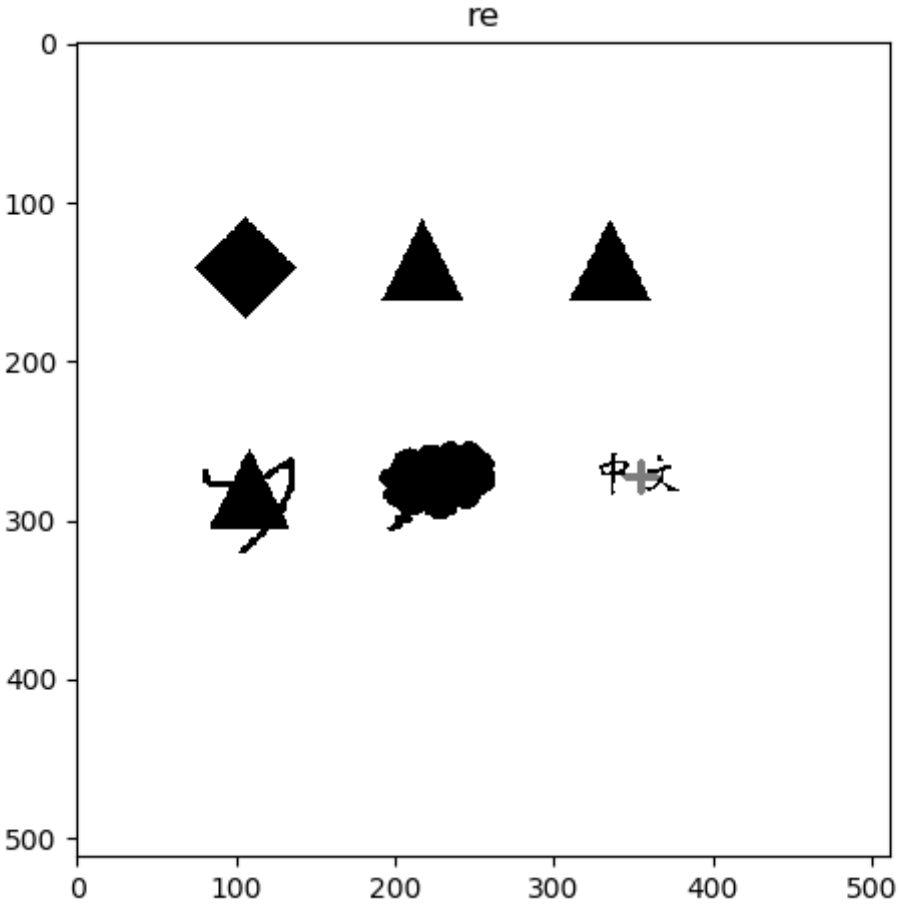
结果



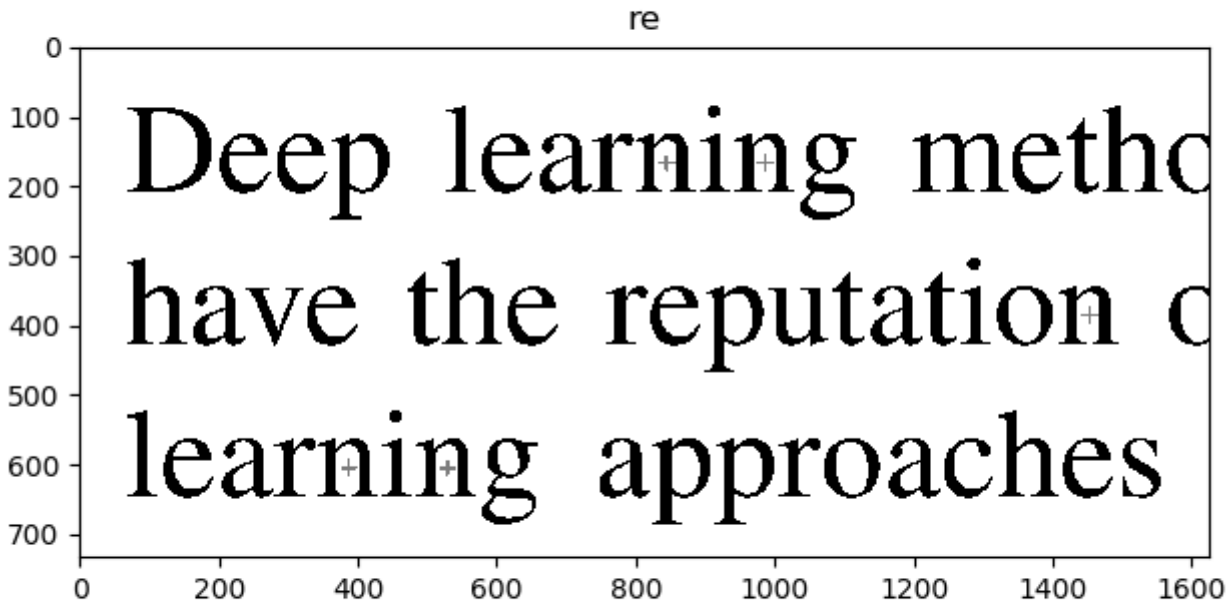
三角形只击中两个，说明左下角的被影响到了，说明这种方法识别击中的是与结构元完全相同的图像



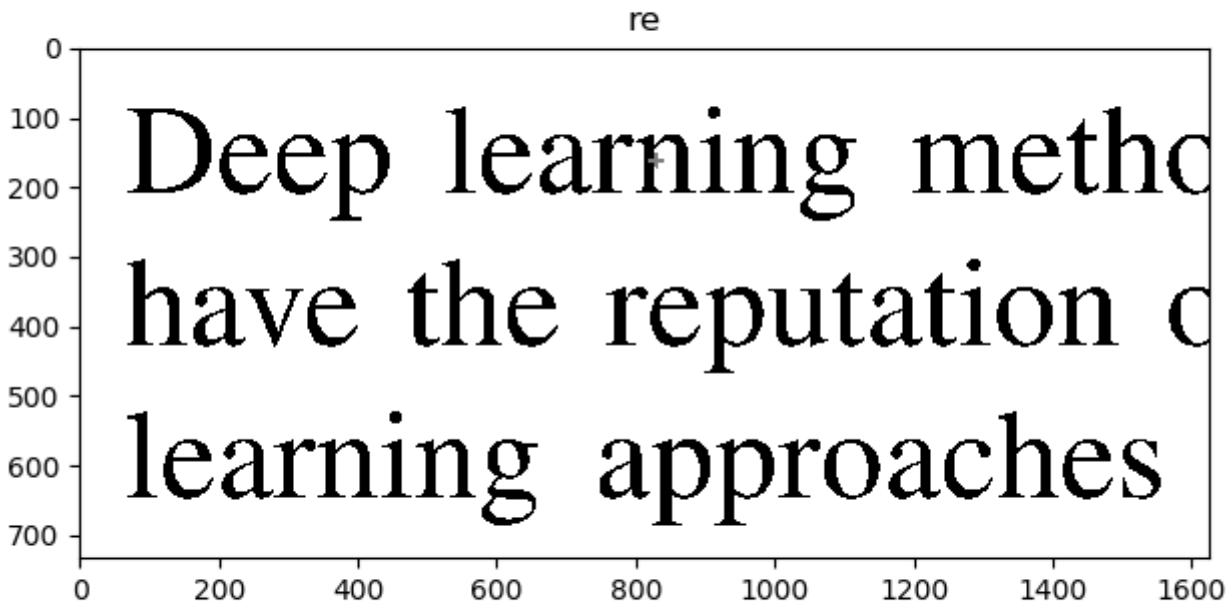
复杂图形



中文



字母识别



单词识别，左下角learning没有被识别到，因为在截图产生了噪声，虽然进行了阈值处理，但

是还是有误差。
所以，用这种识别单词只是在理想情况下可行。