# 图像匹配

图像匹配算法分为3类：基于灰度的匹配算法、基于特征的匹配算法、基于关系的匹配算法

### 基于灰度的模板匹配算法：



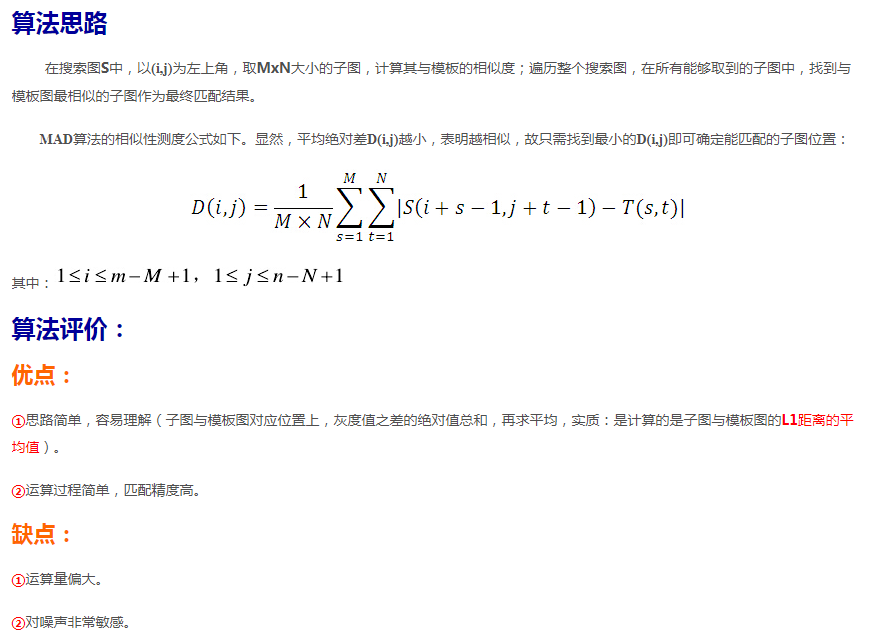


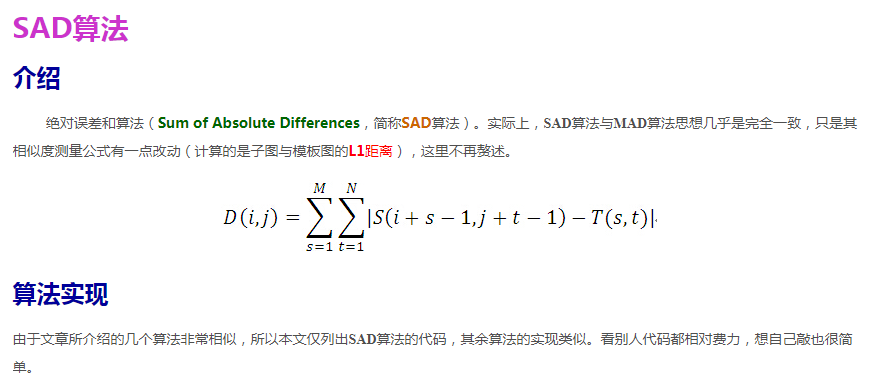
模板匹配的要做的事情。其实模板匹配实现的思想也是很简单很暴力的，就是拿着模板图片（姚明头像）在原图（全明星照）中从左上至右下依次滑动，直到遇到某个区域的相似度低于我们设定的阈值，那么我们就认为该区域与模板匹配了，也就是我们找到了姚明的位置，并把它标记出来。

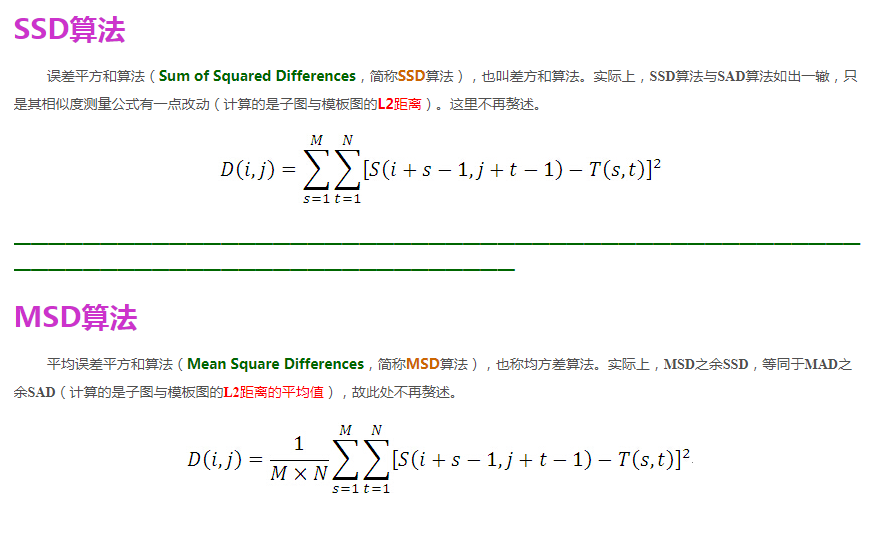
模板匹配（Blocking Matching）是根据已知模板图像到另一幅图像中寻找与模板图像相似的子图像。基于灰度的匹配算法也称作相关匹配算法，用空间二维滑动模板进行匹配，不同匹配算法主要体现在相关准则的选择方面，常见的基于模板的匹配算法可以参考下面的链接：<http://blog.csdn.net/hujingshuang/article/details/47759579>

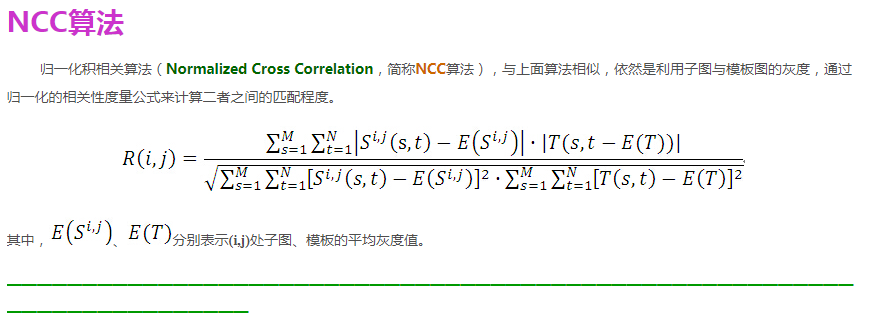
基于灰度的模板匹配算法：MAD、SAD、SSD、MSD、NCC、SSDA、SATD算法











### 基于特征的匹配算法

首先提取图像的特征，再生成特征描述子，最后根据描述子的相似程度对两幅图像的特征之间进行匹配。

图像的特征主要可以分为点、线（边缘）、区域（面）等特征，也可以分为局部特征和全局特征。区域（面）特征提取比较麻烦，耗时，因此主要用点特征和边缘特征。

点特征包括：Harris 、Moravec、KLT、SIFT、SURF 、BRIEF、SUSAN、FAST、CENSUS、FREAK（点击打开链接）、BRISK（点击打开链接）、ORB、光流法（点击打开链接）、A-KAZE等。

边缘特征包括：LoG算子、Robert算子、Sobel算子、Prewitt算子、Canny算子等。

光流法：

### 基于关系的匹配算法

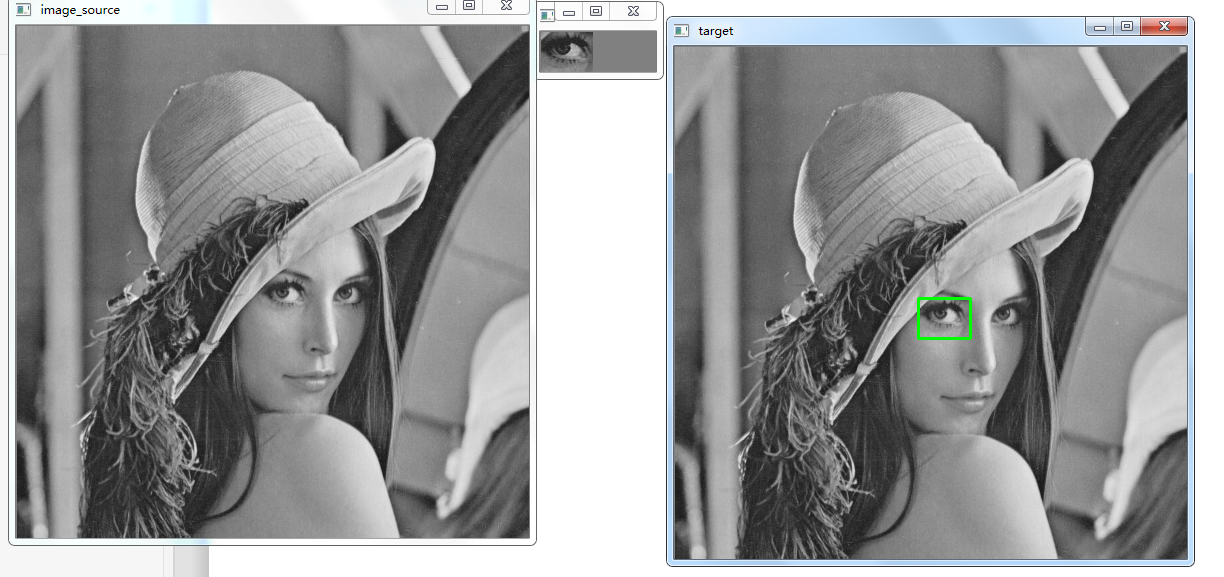
建立语义的网络，是人工智能领域在图像处理中的应用，但还没有突破性的进展。

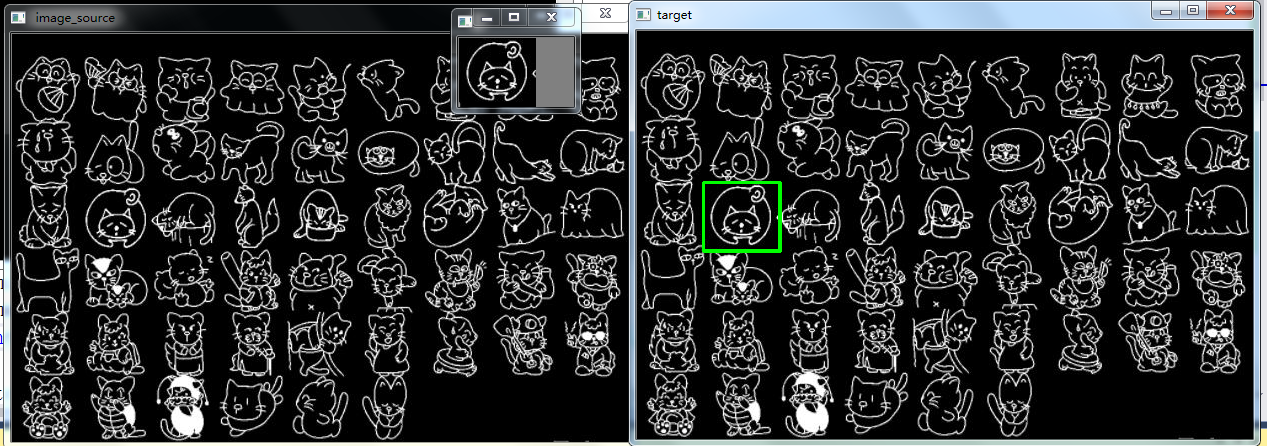
立体匹配：左右相机的匹配，用基于灰度的模板匹配方法，对所有的像素点匹配。

跟踪匹配：前后帧的匹配，用基于特征的匹配算法，对特征点进行匹配，然后根据几个匹配的特征点计算出相机的位姿。

## 模板匹配

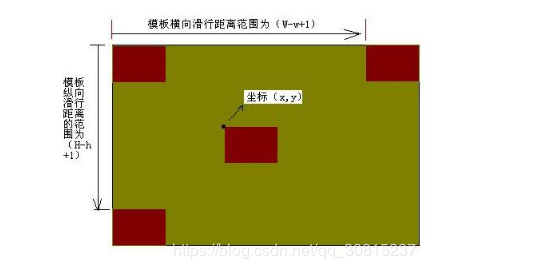
通常被用于目标检测、相似度分析中，opencv3中提供了一个专门用于模板匹配的函数—— matchTemplate





模板匹配的基本思想是将模板图像在源图像上滑动逐一对比，通过统计的基本方法进行匹配，比如方差检验，相关性检验等方法来寻找最佳匹配。

假设褐色的大图为源图片W H，红色小图为模板图片w h。



## matchTemplate 功能及参数

void matchTemplate( InputArray image, InputArray templ, OutputArray result, int method, InputArray mask = noArray() );

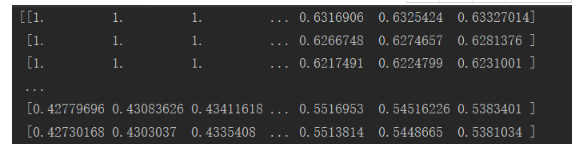
参数的意义

image, // 用于搜索的输入图像, 8U 或 32F, 大小 W-H

templ, // 用于匹配的模板，和image类型相同， 大小 w-h

#### 参数result, // 匹配结果图像, 类型 32F, 大小 (W-w+1) (H-h+1)

函数返回值result的值：result 是一个矩阵，返回每一个点匹配的结果。



 1.result数据的含义

       模板匹配函数cvMatchTemplate依次计算模板与待测图片的重叠区域的相似度，并将结果存入映射图像result当中，也就是说result图像中的每一个点的值代表了一次相似度比较结果。

2.result的尺寸大小

       模板在，并作一次比较计算，由此，横向比较W-w+1次，纵向比较H-h+1次，从而得待测图像上每次在横向或是纵向上移动一个像素到一个（W-w+1）×（H-h+1）维的结果矩阵，result即是用图像来表示这样的矩阵,因而图像result的大小为（W-w+1）×（H-h+1）。匹配结果图像与原图像之间的大小关系，他们之间差了一个模板大小。

3.如何从result中获得最佳匹配区域

       使用函数MinMaxLoc(result,&min\_val,&max\_val,&min\_loc,&max\_loc,NULL);从result中提取最大值（相似度最高）以及最大值的位置（即在result中该最大值max\_val的坐标位置max\_loc，即模板滑行时左上角的坐标，类似于图中的坐标（x,y）。

### 参数method // 用于比较的方法

enum TemplateMatchModes {

TM\_SQDIFF = 0, sqrt difference

TM\_SQDIFF\_NORMED = 1, 归一化

TM\_CCORR = 2,

TM\_CCORR\_NORMED = 3,

TM\_CCOEFF = 4,

TM\_CCOEFF\_NORMED = 5

};

opencv中支持的比较方法有六种，分别如下：

T(x,y) 用来表示我们的模板。I(x,y) 是我们的目标图像。 R(x,y) 是用来描述相似度的函数。

1. TM\_SQDIFF：该方法使用平方差进行匹配，因此最佳的匹配结果在结果为0处，值越大匹配结果越差。

2、TM\_SQDIFF\_NORMED：该方法使用归一化的平方差进行匹配，最佳匹配也在结果为0处,值越大匹配结果越差。

这种标准化操作可以保证当模板和图像各个像素的亮度都乘上了同一个系数时，相关度不发生变化。当 I(x,y)和T(x,y) 变为k×I(x,y)和k×T(x,y) 时，R(x,y)不发生变化。

3、TM\_CCORR：相关性匹配方法，该方法使用源图像与模板图像的卷积结果进行匹配，因此，最佳匹配位置在值最大处，值越小匹配结果越差。

4、 TM\_CCORR\_NORMED：归一化的相关性匹配方法，与相关性匹配方法类似，最佳匹配位置也是在值最大处。

5、 TM\_CCOEFF：相关性系数匹配方法，该方法使用源图像与其均值的差、模板与其均值的差二者之间的相关性进行匹配，最佳匹配结果在值等于1处，最差匹配结果在值等于-1处，值等于0直接表示二者不相关。

6、 TM\_CCOEFF\_NORMED：归一化的相关性系数匹配方法，正值表示匹配的结果较好，负值则表示匹配的效果较差，也是值越大，匹配效果也好。

总结：

TM\_SQDIFF 0 ，TM\_SQDIFF\_NORMED 1 匹配数值越低表示匹配效果越好，其它四种反之。

TM\_SQDIFF\_NORMED，TM\_CCORR\_NORMED，TM\_CCOEFF\_NORMED是标准化的匹配，得到的最大值，最小值范围在0~1之间，其它则需要自己对结果矩阵归一化。

不同的方法会得到差异很大的结果，可以通过测试选择最合适的方法。

### 模板匹配代码

若想找到最佳匹配位置，只需要找到匹配结果图像的最大值点即可，使用minMaxLoc()函数来找这个最大值点。找到结果后，将其绘制到原图像上 ，这里注意匹配结果图像与原图像之间的大小关系，他们之间差了一个模板大小。

#include<opencv2\opencv.hpp>

#include<iostream>

using namespace std;

using namespace cv;

int main()

{

Mat image\_source = imread("e:\\lena.jpg", IMREAD\_GRAYSCALE);

Mat image\_template = imread("e:\\lefteye\_lena.jpg", 0);//righteye\_lena.jpg

imshow(" image\_source", image\_source);

imshow("image\_template ", image\_template); 

waitKey(0);

return 0;

}

### for循环对每个方法的使用

#include<opencv2\opencv.hpp>

#include<iostream>

using namespace std;

using namespace cv;

int main()

{

Mat image\_source = imread("e:\\cat1.jpg", 0); // tree1.jpg lena.jpg

Mat image\_template = imread("e:\\cat2.jpg", 0);// tree2.jpg righteye\_lena.jpg

imshow(" image\_source", image\_source);

imshow("image\_template ", image\_template);

Mat image\_matched;

double minVal, maxVal;

Point minLoc, maxLoc;

Point matchLoc;

for (int method = 0; method < 6; method++)

//0~5;TM\_SQDIFF = 0, TM\_SQDIFF\_NORMED = 1,TM\_CCORR = 2,TM\_CCORR\_NORMED = 3,TM\_CCOEFF = 4,TM\_CCOEFF\_NORMED = 5

{

matchTemplate(image\_source, image\_template, image\_matched, method);

normalize(image\_matched, image\_matched, 0, 1, NORM\_MINMAX);//对匹配结果归一化，保证在0 到1之间

minMaxLoc(image\_matched, &minVal, &maxVal, &minLoc, &maxLoc);

//对于方法 SQDIFF 和 SQDIFF\_NORMED, 越小的数值代表更高的匹配结果. 而对于其他方法, 数值越大匹配越好

if (method == 0 || method == 1) matchLoc = minLoc; // TM\_SQDIFF 0 ; TM\_SQDIFF\_NORMED 1

else matchLoc = maxLoc;

Mat image\_color;

cvtColor(image\_source, image\_color, COLOR\_GRAY2BGR);

Point end = Point(matchLoc.x + image\_template.cols, matchLoc.y + image\_template.rows);

rectangle(image\_color, matchLoc, end, Scalar(0, 255, 0), 2, 8, 0);

//rectangle(image\_color, Rect(matchLoc,Size(image\_template.cols,image\_template.rows)), Scalar(0, 255, 0), 2, 8, 0);

imshow("target", image\_color);

waitKey(5000);//5s内按任意键，for循环进行下一次循环操作

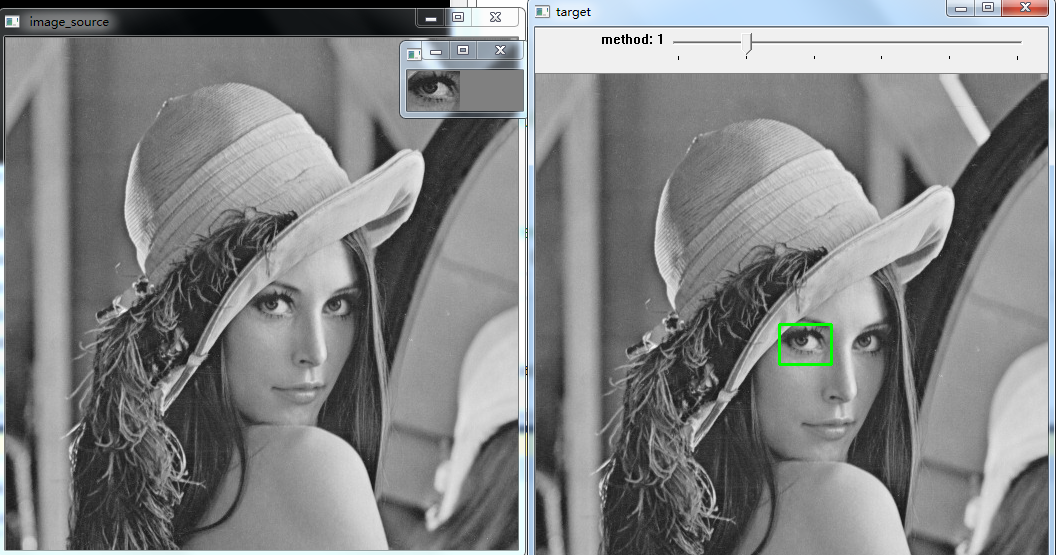
system("pause");

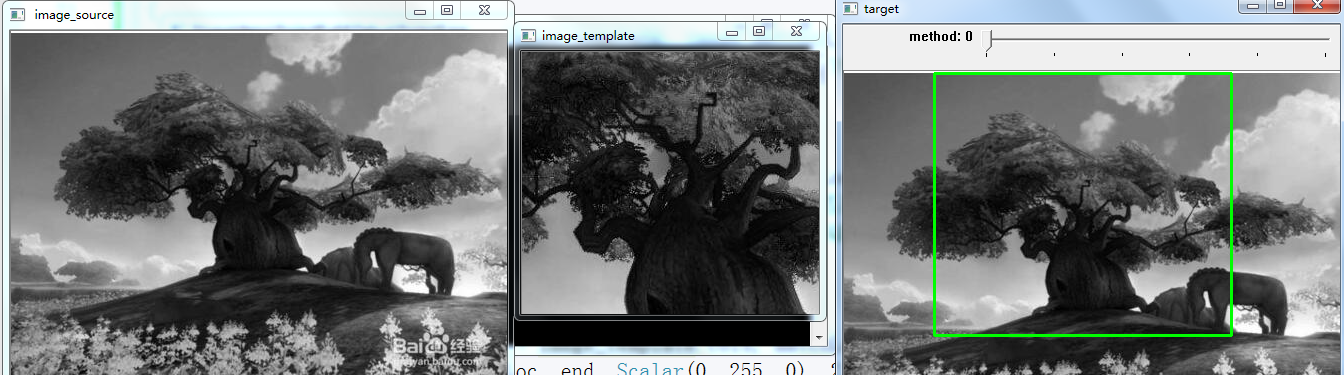
}

return 0;

}

## 滑块（方法0~5）+模板匹配





#include<opencv2\opencv.hpp>

#include<iostream>

using namespace std;

using namespace cv;

int main()

{

image\_source = imread("e:\\tree1.jpg", 0); // lena.jpg

image\_template = imread("e:\\tree2.jpg", 0);//righteye\_lena.jpg

imshow(" image\_source", image\_source);

imshow("image\_template ", image\_template);

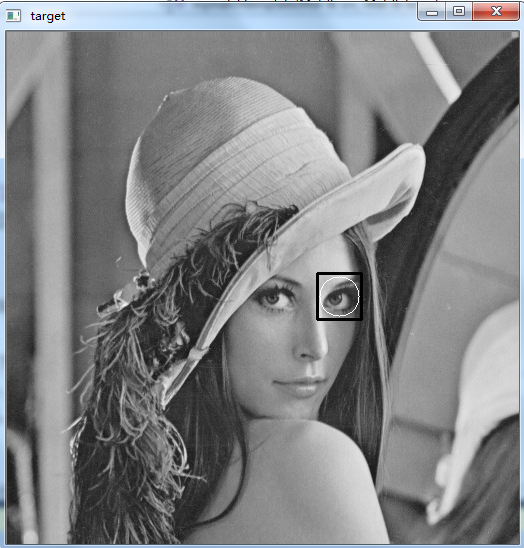
//补充代码 通过滑块滑动 （0~5）实现不同方法的 模板匹配

waitKey(0);

return 0;

}

## 函数mymatchtemplate实现模板匹配



#include<opencv2\opencv.hpp>

#include<iostream>

using namespace std;

using namespace cv;

void mymatchtemplate(Mat &src, Mat &temp);

int main()

{

Mat image\_source = imread("e:\\lena.jpg");

Mat image\_template = imread("e:\\lefteye\_lena.jpg");//righteye\_lena

//补充代码 调用函数mymatchtemplate， 实现模板匹配

waitKey(0);

return 0;

}

void mymatchtemplate(Mat &src, Mat &temp)

{

Mat image\_matched;

int result\_cols = src.cols - temp.cols+1;

int result\_rows = src.rows - temp.rows+1;

if (result\_cols < 0 || result\_rows < 0)

{

cout << "input correct image" << endl;

return;

}

image\_matched.create(result\_rows, result\_cols, CV\_32FC1);

matchTemplate(src, temp, image\_matched, TM\_CCOEFF\_NORMED);

double minVal, maxVal;

Point minLoc, maxLoc;

minMaxLoc(image\_matched, &minVal, &maxVal, &minLoc, &maxLoc);

Mat drawing = src.clone();

circle(drawing, Point(maxLoc.x + temp.cols / 2, maxLoc.y + temp.rows / 2), 20, Scalar(255));

Point end = Point(maxLoc.x + temp.cols, maxLoc.y + temp.rows);

rectangle(drawing, maxLoc, end, Scalar(0), 2, 8, 0);

imshow(" target", drawing);

}

## 模板匹配特点

虽然精度不高，但选择恰当的方法，设置合适的阈值也能起到一定作用。有的时候我们还能用模板匹配来定位