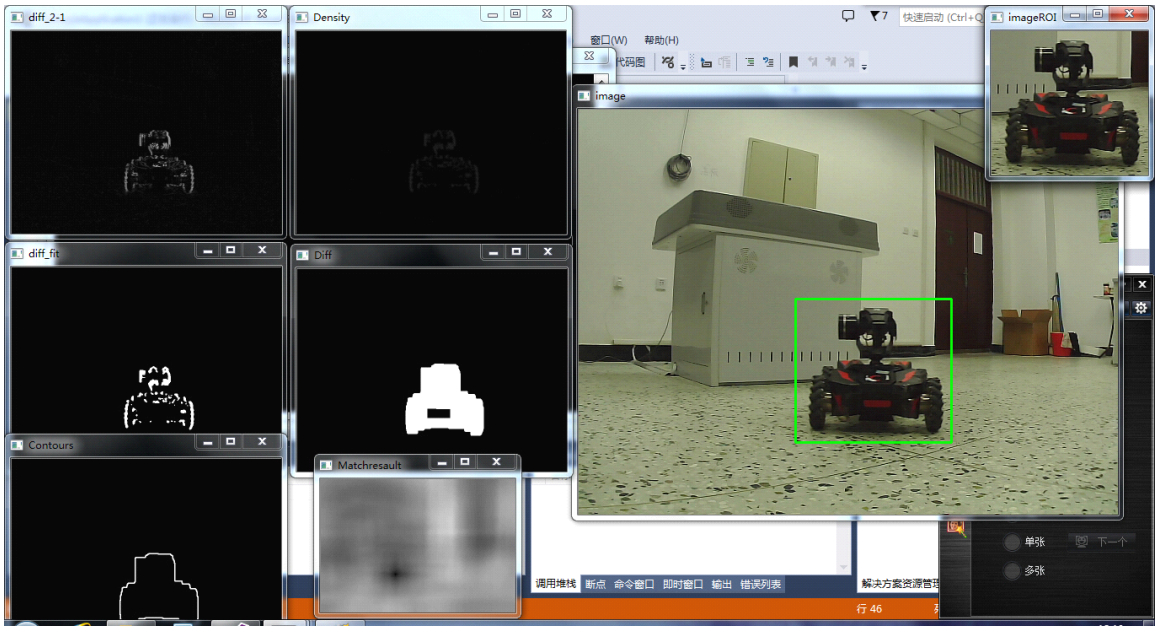
模版匹配法检测在 RM 防御塔的应用

写在前面：

这个星期前几天的事情排得比较满，所以今天才开始写本周的内容。通过上周的教程，相信大家已经能够明白或者掌握如何通过帧差法进行运动物体检测并获取感兴趣区域（目标）的图像了。但随之问题来了，如果一个物体的运动状态由运动变成了静止，那么我们通过相邻两帧进行差分便得到了一副全部由随机噪声组成的图片，随之所有的运动检测都没有了意义，我们也就无法跟踪到运动目标了。由此，我们引入今天讲解的内容--------模版匹配。

通过我们由帧差法得到的运动目标图像作为模版，对全图进行匹配，进而对静态的目标物体进行定位跟踪。多的不说，上图上代码。

还是之前的程序运行截图，只是这次我们关注点不同了。。



函数解释：

本次讲解主要函数为 MatchTemplate（ ）;

函数原型：

```
void MatchTemplate(
    InputArray image,     //待搜索的图像   且须为 8 位或 32 位浮点型图像
    InputArray templ,     //搜索的模版图像   和源图像相同的类型 且尺寸小
                                于待搜索图像
    OutputArray result,   //输出的比较结果的映射图像
                                必须为单通道 32 位浮点图像 ,如果图像尺寸为
                                W*H 而 templ 尺寸是 w*h，则此参数 result 一
                                定是（W-w+1）*（H-h+1）
        int method )      //指定的匹配方法
```

匹配方法举例：

CV_TM_SQDIFF 平方差匹配法：最好的匹配值为 0；匹配越差，匹配值越大。

CV_TM_CCORR 相关匹配法：该方法采用乘法操作；数值越大表明匹配程度越好。

CV_TM_CCOEFF 相关系数匹配法：1 表示完美的匹配；-1 表示最差的匹配。

CV_TM_SQDIFF_NORMED 归一化平方差匹配法

CV_TM_CCORR_NORMED 归一化相关匹配法

CV_TM_CCOEFF_NORMED 归一化相关系数匹配法

上述几种匹配方式需要的计算时间比较接近，我们可以选择一个能适应场景的匹配方式。

程序思路点拨：

由之前的帧差运动匹配得到运动目标的彩色信息，进行保存。每次帧差检测运动目标成功后都要对这个目标图像进行更新，以保证我们的目标信息是最新的。一旦帧差运动检测失败，我们可以用上次保存的图像信息作为模版进行模版匹配，进行目标跟踪。

Code：

```
// 进行模版匹配的条件 （帧差匹配失败）
if (!(DefenTower.State & STATE_FramdiffLocking))
    // MatchTemplate 模版匹配
    DefenTower.MatchTemplateDetection();
```

以上代码为首先判断帧差匹配失败，然后进行模版匹配（这两段函数是我在主函数里进行调用的条件，是帧差匹配和模版匹配的转换，为了大家教程的连续性我把它粘了出来，大家不需要对这两行代码进行深究。）

模版匹配的本质是电脑对匹配模版在待搜索图像上所有的可能位置进行运算，每次运算会对模版上所有像素点进行匹配。无论是我们的匹配模版还是待搜索图片的尺寸都相当巨大，导致速度骤降。所以要对模版与带搜索图像进行尺寸缩减：

Code：

```
//尺寸缩减
cv::Mat image_little;
cv::Mat temPlate_little;
cv::resize(image, image_little, cv::Size(320, 240));
cv::resize(imageROI, temPlate_little,
        cv::Size(imageROI_Size.width / 2, imageROI_Size.height / 2));
```

*其中 image 是由摄像头读回的图片（640*480），而 imageROI 则是上次帧差匹配到的运动目标图片。尺寸缩减比相当于长宽各缩减 1/2*

根据前面对 MatchTemplate（ ）函数中 result 图像尺寸的要求进行 result 图像创建：

Code：

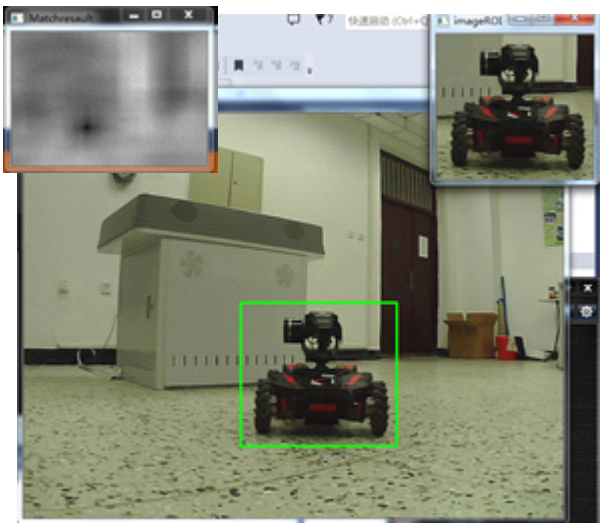
```
int resultImage_cols = image_little.cols - temPlate_little.cols + 1;
int resultImage_rows = image_little.rows - temPlate_little.rows + 1;
cv::Mat reasult;
reasult.create(resultImage_cols, resultImage_rows, CV_32FC1);
```

其中 image_little 是尺寸缩减后的待搜索图片 temPlate_little 是尺寸缩减后的匹配模版。

万事俱备只欠东风，是时候调用 MatchTemplate（ ）函数了。

Code：

```
//模版匹配
cv::matchTemplate(image_little, temPlate_little, reasult,
                                        cv::TM_SQDIFF_NORMED);
```



可看到如果我们以右上角的 ROI 图片作为模版，以中间的 image（不包含绿色识别框）作为搜索图像，得到左上角的结果图像，其中最黑的点（数值最小）的位置为匹配最佳位置。

通过搜索数值最小点的位置，可以获得最佳匹配坐标。

Code：

```
//寻找最大值
double minValue, maxValue;
cv::Point minLocation, maxLocation;
cv::Point matchLocation;
cv::minMaxLoc(reasult, &minValue, &maxValue, &minLocation,
                                    &maxLocation, cv::Mat());
matchLocation = minLocation;
```

值得注意的一点是如果匹配方式选取的是 CV_TM_SQDIFF 或者 CV_TM_SQDIFF_NORMED 则最小点（黑点）为最佳匹配点，其他匹配方式结果图像的最大点（白点）为最佳匹配点。

同时，我认为当最小值小于 0.3 时，匹配到的位置才是准确位置，否则我认为发生了错误匹配，从而提高了匹配准确性，也是为什么我将书中的历程中的标准化一步删去的原因。

Code：

```
//标准化
//cv::normalize(reasult, reasult, 0, 1, cv::NORM_MINMAX, -1, cv::Mat());
```

被注释掉的标准化。如果保留标准化，则无论图像中是否能够真的匹配到最佳位置，标准化都会强制将结果图像的最小值与最大值缩放到 0 和 1，从而失去准确性，换句话说就是任何一个带搜索图像和任何一个不相干的模版，匹配标准化后结果图像中都会存在 0（最佳匹配）这个点，从而失去跟踪的意义。

最后一步，由于我们是在尺寸缩小的图像中进行匹配得到的最佳匹配位置，我们要通过适当的缩放将最佳匹配位置放大到原始尺寸。同时得到匹配置信度。

Code：

```
if (minValue <0.22)//如果匹配成功（匹配系数小于0.3）
{
    MatchCfdLev_Target = minValue*1000;   //获取置信度
    cv::rectangle(image_little, matchLocation,
        cv::Point(matchLocation.x + temPlate_little.cols,
        matchLocation.y + temPlate_little.rows),
        cv::Scalar(0, 255, 0), 2, 8, 0);//通过匹配模版的尺寸获取匹配矩形

    Loc_Target = cv::Rect(matchLocation,
                          cv::Point(matchLocation.x + temPlate_little.cols,
                          matchLocation.y + temPlate_little.rows));
```

```
//放大矩形 对应到原始尺寸
Loc_Target.x = Loc_Target.x * 2;
Loc_Target.y = Loc_Target.y * 2;
Loc_Target.width = Loc_Target.width * 2;
Loc_Target.height = Loc_Target.height * 2;

State = State | STATE_MatchLocking;//置位 模版匹配
```

最终获得的 Loc_Target 就是模版匹配给出的最终目标位置 并且置位标志位，表明模版匹配成功。

当然如果最小值大于了 0.3，我认为发生了错误匹配，或者是图中没有匹配位置，模版标志位清零，表示匹配失败，也就是视频中出现目标丢失的情况。目前因为程序还不是很稳定，参数都需要调整，还需要花大力气去调试。

写在最后

当然，模版匹配还有很多细节我没有讲解。鉴于时间与篇幅的关系，今天我就写到这里。

下周我会对整个程序匹配模式的切换过程入手，简单介绍防抖和在模版匹配下的 imageROI 的更新问题。预计通过下周的讲解，读者可以从整体上了解展示视频中的代码构造和技术路线，实现自己防御塔的程序编写。请大家继续期待。

感谢


robomasters


baid1995


w5862338


而后，淡默安然

以及所有下载了我第一篇教程的奋斗在一线视觉的战友们。

有你们的支持我很开心，尤其是 ->

真心给力！

希望大家继续关注。

和小猴一起研究视觉（防御塔视角）

侯小猴

东北林业大学 大学生创新实验室