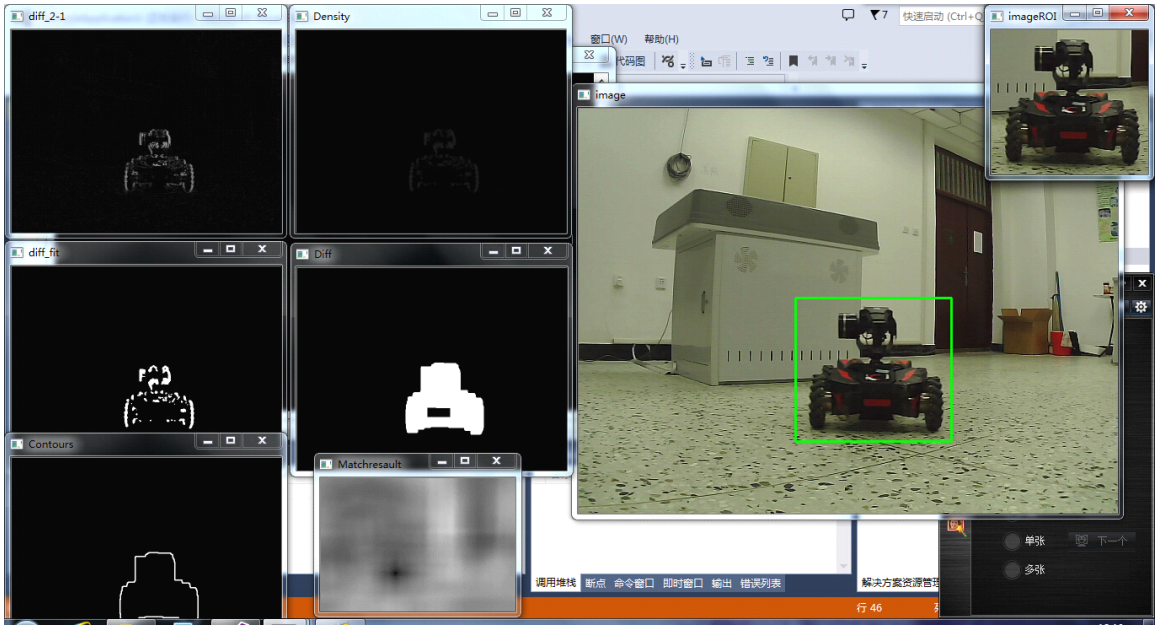
运动检测在 RM 防御塔的应用中若干小问题

写在前面：

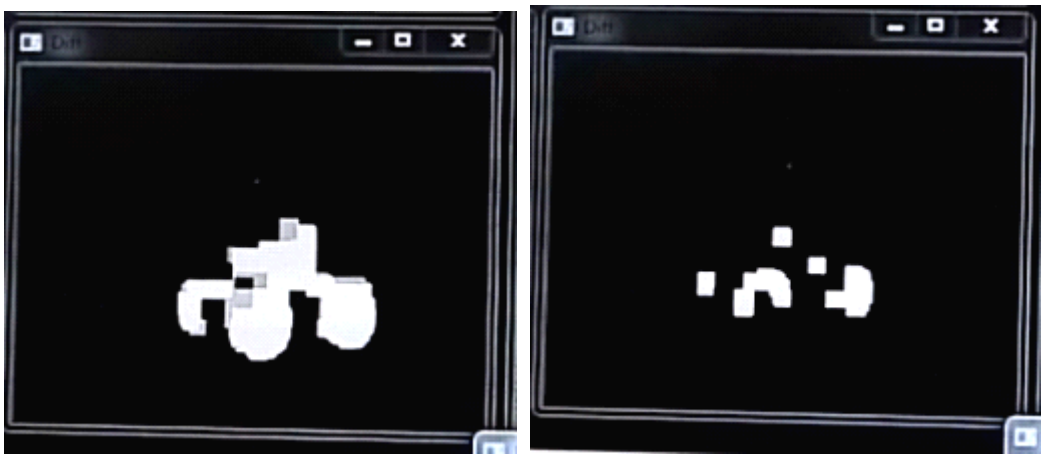
侯小猴又和大家见面了，本周进行这个主题的最后一讲。主要解决三帧差分和模版匹配配合的问题以及如何识别抖动，如何提高识别率等问题。多的不说，进入正题。

依旧先摆个程序运行截图压压惊。



帧差分法中会出现的问题：

如果大家按照我之前的教程写了帧差的程序，并且通过寻找联通区域提取最小包络矩形进行目标锁定会发现在目标（战车）由运动到停止的过渡中，通过形态学变换的战车的大块联通区会在短时间内碎成很多小块，从而影响包络矩形的寻找，造成跟踪的不准确，如图：



战车运动的时候是左侧图像，一旦战车停止，整块的连通区域会碎成很多小块，从而影响识别（最明显的效果是程序将随机圈出一个连通小块）。我给出的思路是由于碎片是瞬间产生的，我们可以通过保存上一次目标位置以及识别到的目标尺寸来计算本次目标位置与尺寸的变化量，设置阀值，如果变化过于剧烈，则认为是由于连通区域破碎引起的误判，进而标志运动检测失败的标志位，启动模块匹配。

Code：

```
if (                                                    //本次与上次的位置信息
    abs(Loc_Target_pre.height - Loc_Target.height)>30 ||  // 高度变化大于阈值
    abs(Loc_Target_pre.width - Loc_Target.width)>30 ||    // 宽度变化大于阈值
    abs((Loc_Target_pre.x - Loc_Target.x)>50) ||          // x变化大于阈值
    abs((Loc_Target_pre.y - Loc_Target.y)>50)             // y变化大于阈值
    )
{
    tem_TargetGet = 0;//检测不成功
}
```

通过代码可以看到我对目标尺寸和位置的变化进行了限制。

同时，我们还可以通过识别到的尺寸的宽高比进行限制，由于我们战车在图像上的投影近似于一个正方形，如果识别到目标的宽高比不再 0.67~1.5 我们也可以认为是由于某种原因引起的误判，从而失能三帧差检测。

Code：

```
if (((double)Loc_Target.height / (double)Loc_Target.width) < 0.67 ||
    ((double)Loc_Target.height / (double)Loc_Target.width) >1.5)
{
    tem_TargetGet = 0;
}
```

如果本次识别到的目标长宽比不是 3：2 或者 2：3，我们认为是误判。

对于如果摄像头发生了抖动，那么在图像上的特点就会更加明显，经过形态学变换后的图像会呈现出大块大块的连通区域，我们通过识别这个特征，可以判断图像是否发生了抖动。

Code：

```
// 挑选连通区域
int cmin = 300;  // minimum contour length
int cmax = 2000; // maximum contour length
int tem_erase_num = 0;
std::vector<std::vector<cv::Point>>::
    const_iterator itc = contours.begin();
while (itc != contours.end()) {

    if (itc->size() < cmin || itc->size() > cmax)
    {
        itc = contours.erase(itc);
        tem_erase_num++;
    }

    else
        ++itc;
}
```

可以看到之前在第一讲中给大家介绍的挑取连通区域的段落我设定了一个 tem_erase_num 变量来统计不合格的连通区域数量。如果剔除的连通区域数量过多，我就可以认为是发生了抖动，如下面段落：

```
if (tem_erase_num > 2)
    //如果剔除的小连通区域大于2个 认为是在抖动
{
    tem_TargetGet = 0;
    tem_Delay = 1;
}
```

同时，在抖动过后，摄像头在趋向于稳定的过程中，形态学变换后的图像中偶然会出现只有一个连通区域（是由于震动引起的），为了避免程序将此连通区域识别为敌军，我们通常设定在程序判定抖动结束后的几帧内依然失能帧差检测，以维持识别准确率。

Code：

```
if (tem_Delay) //抖动后等待图像稳定
{
    tem_Delay--;
    tem_TargetGet = 0;//检测失败
}
```

其中的 tem_Delay 在识别到震动后被置 1。表示在震动结束后第 2 帧才会使能三帧差分。

模版匹配中出现的问题：

模版匹配凭借着其不受震动与视角变换的影响在目标跟踪中起着举足轻重的作用，但是由于模版是固定的，如果在模版匹配中目标物体发生了运动，势必造成模版与当前目标形成差别，从而影响其识别率。简单的解决思路是增加模版匹配的自动更新模版的机制。但是如果没帧都进行模版更新，则会造成目标的漂移。由此，我们拟采用才可接受的范围内不进行模版更新，在模版与实际目标差别较大但依然能保持识别率的时候进行模版更新，从而降低模版更新的频率，防止匹配漂移，还能够维持较高的跟踪率。

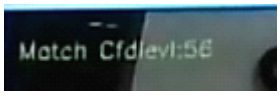
Code：

```
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(matchLoca
    Loc_Target = cv::Rect(matchLocation, cv::Point(matchLocation.x
    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

    if (minValue > 0.18)//置信度比较低
    {
        image(Loc_Target).copyTo(imageROI);//更新样本
    }
}
```

这是我们第二讲中所涉及的一个段落。我们认为 minValue（计算出来的匹配系数）小于 0.22 的时候才是匹配成功（不知道我说的是什么的自己去翻第二讲），同时我认为 minValue>0.18 的时候模版与实际目标已经有了显著差异，我此时进行目标更新，从而得到了在较高匹配准确率下的较低频率的模版更新。同时我给出了匹配的置信度 MatchCfdLev_Target （越小越好）。



帧差分与模版匹配的切换：

Code：

```
while (1)
{
    capture.read(image);
    if (!image.data)
        break;
    DefenTower.InputImage(image);//输入最新的识别图片
    DefenTower.MotionDetection();// 帧差法动态识别

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

    DefenTower.ImageRefresh();//防御塔图片迭代
```

由于我们做了的大量的工作，所以主函数相当简洁。

当帧差识别成功，则跳过模版匹配。若帧差识别失败，则启动模版匹配。最后将本次识别到的图像目标等迭代保存，以供下一帧使用。

写在后面：

至此，***和侯小猴一起研究视觉（防御塔视角）*** 的帧差识别与模版匹配的三章教程就完全写完了，在写教程的同时我也又一次回顾了我的程序，有了不少新的想法和感悟。当然也感谢各位小伙伴以来的关注与支持。***和侯小猴一起研究视觉（防御塔视角）***系列帖子的访问量已经突破了 500 人次，下载量突破了 100 人次，当然也为我带来了一笔客观的“收入”。在之后的备赛过程中，如果有什么好的想法，值得给大家推荐的，我还会继续写给大家。当然，如果大家有什么想和我交流的，也可以通过论坛，邮箱，QQ 等方式，我也很愿意和大家交流，碰撞出智慧的火花。

侯小猴

东北林业大学 大学生创新实验室

QQ：491652376