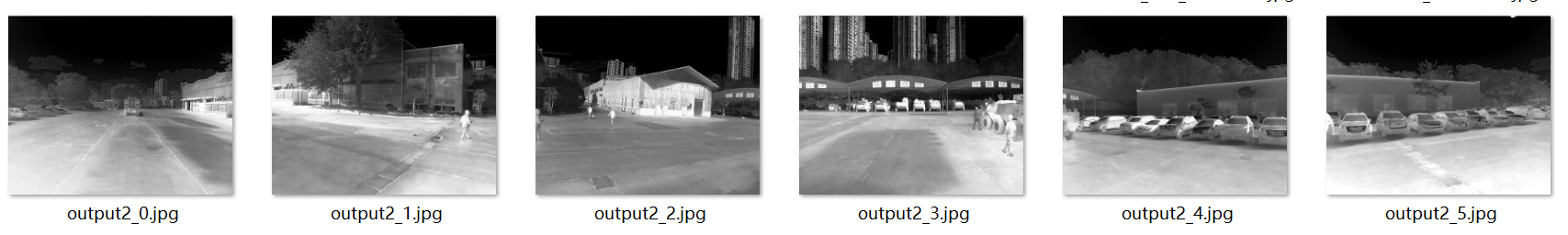
1. **图像采集**

通过车载的六个红外摄像头运行脚本拍摄画面，传回PC。

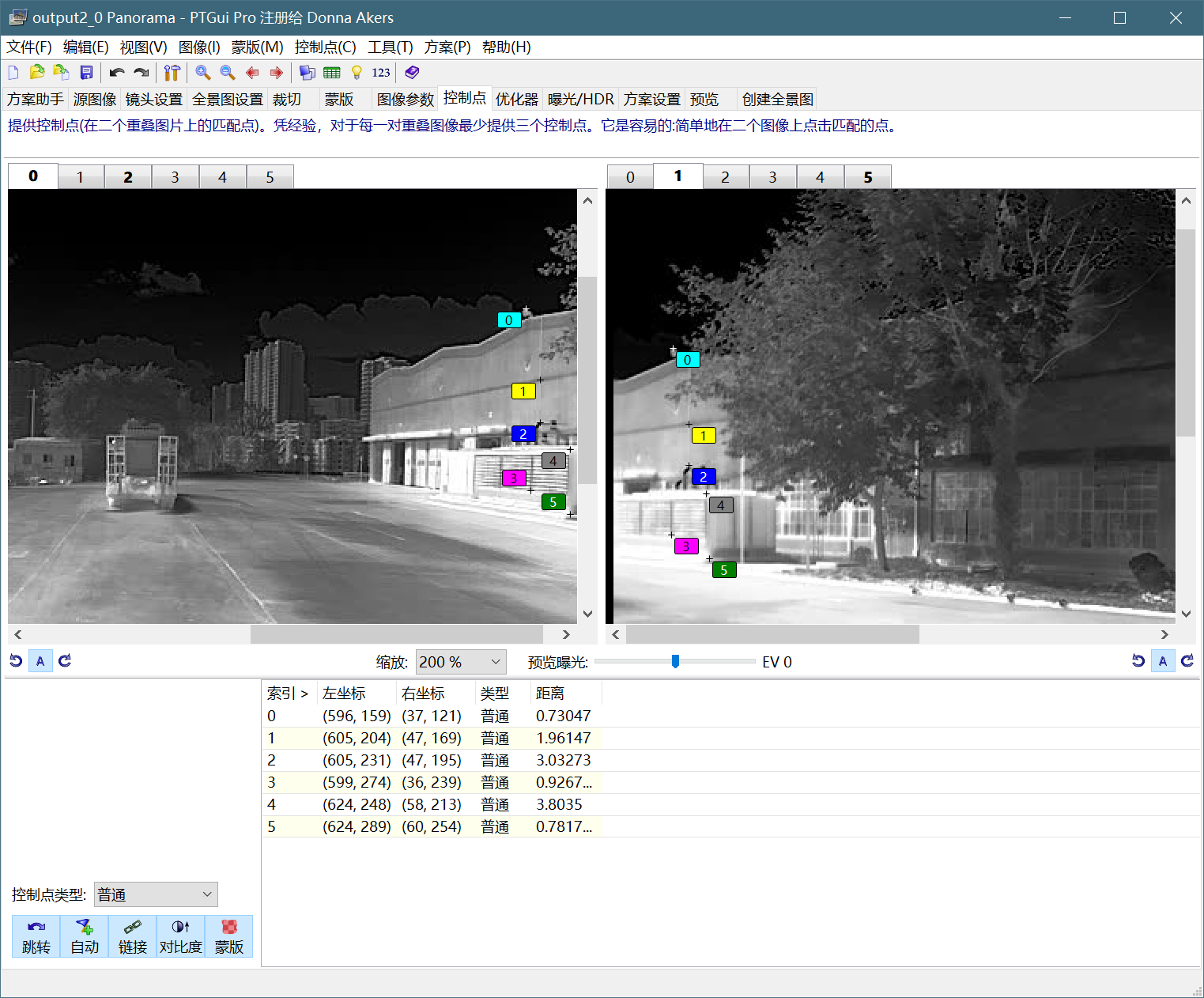


采集的红外图像

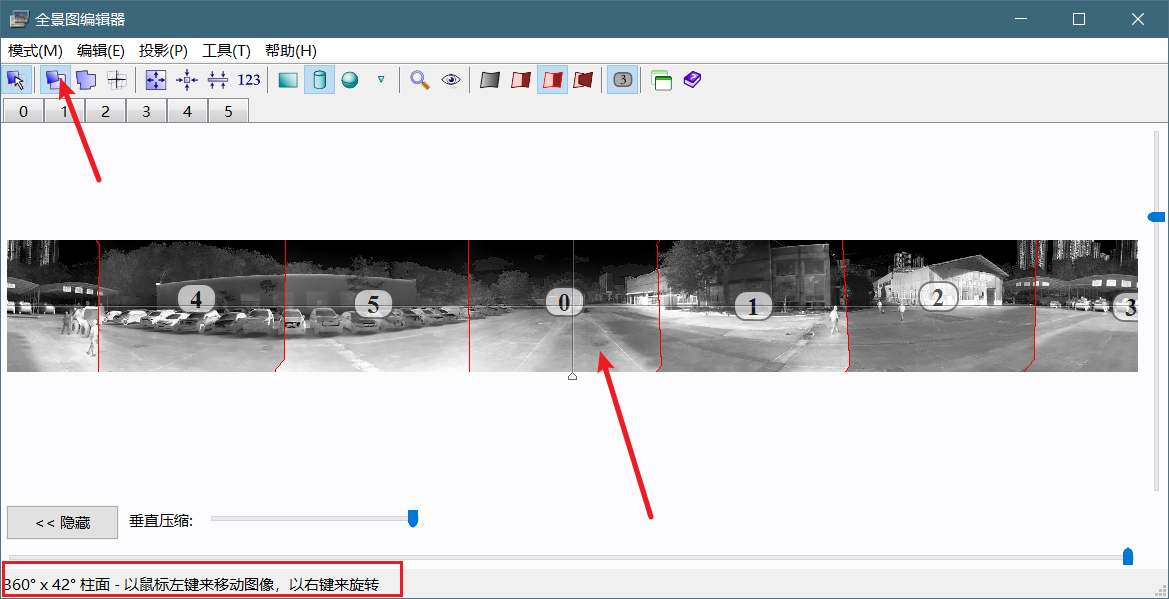
1. **PTGui选取特征点**

利用PTGui选取每张图与邻近图的特征点，需要尽量选取六张图同一距离上的特征点。选取完成后配置好相应的全景图参数，通过优化器优化，进入全景图编辑器调整全景图效果，导出获得在柱面投影情况下的pto后缀的全景拼接参数。

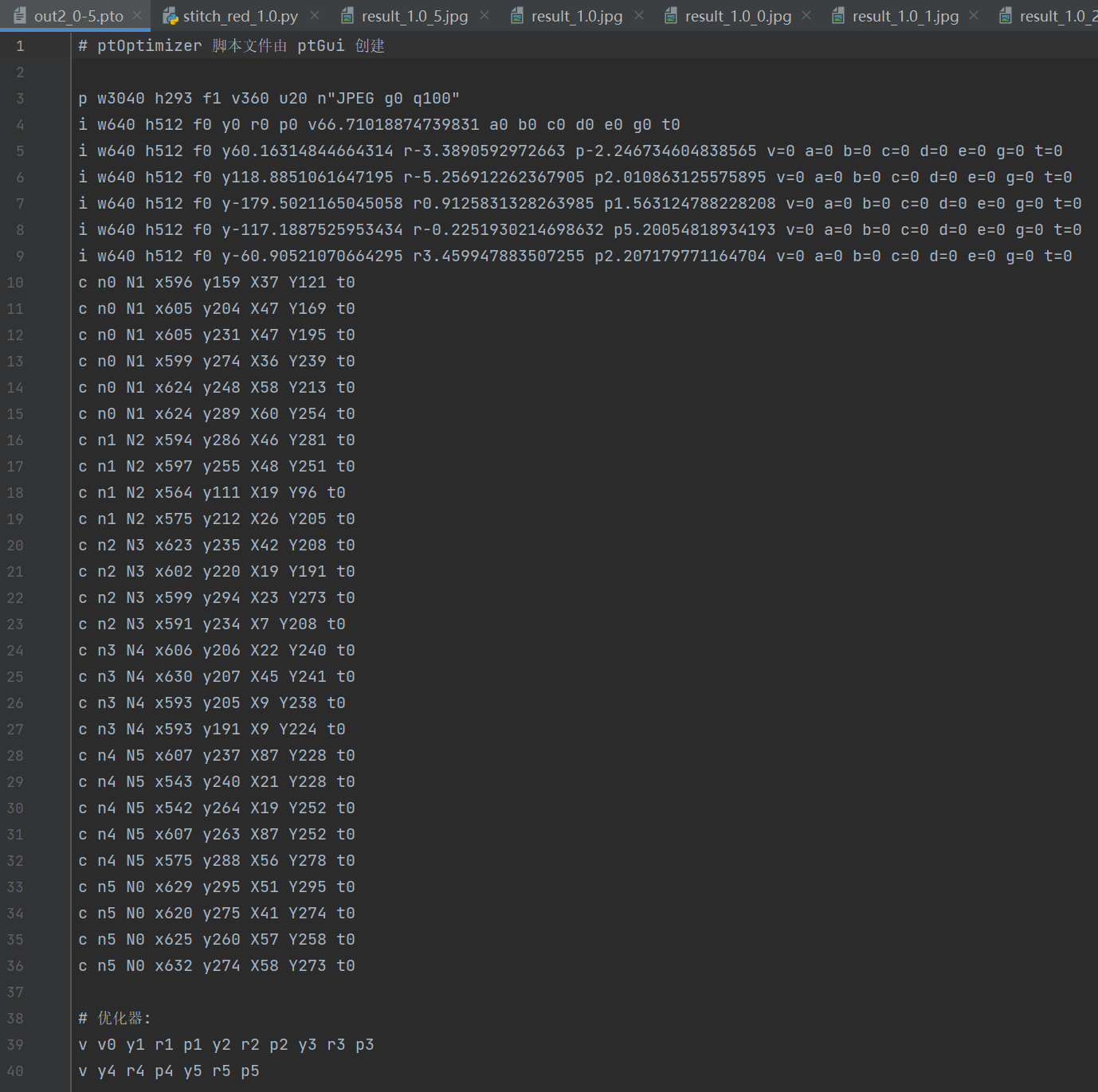
具体详细操作见拼接操作指南。



利用PTGui选取邻近红外图像之间的相同特征点



全景图编辑器中单幅画面微调设置



PTGui生成的pto全景图拼接参数(只要有对应图片数量的i开头行即可)

1. **生成映射表**



原图

1. **柱面投影**

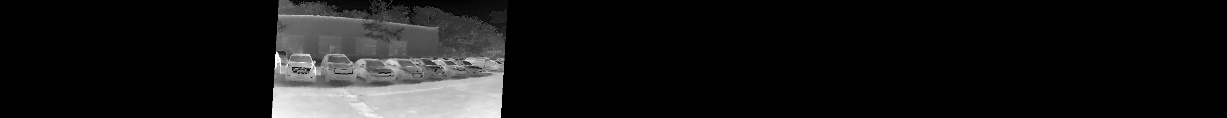
对每张图做柱面投影，建立柱面投影映射表，如下图所示



柱面投影

1. **相似变换**

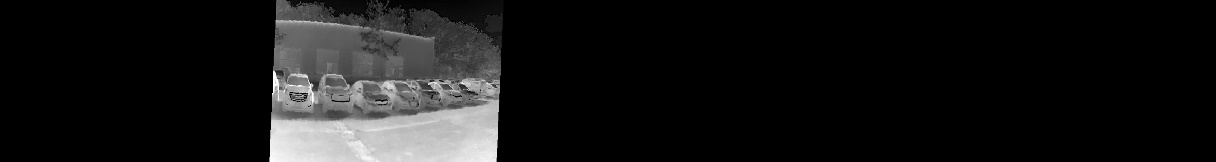
通过每张图对应的pto参数中的y，r，p参数，对应为相机三维姿态欧式角中的yaw，roll，pitch，将图像通过横向纵向的位移，以及旋转，相似变化到全景图中对应的位置，建立相似变换映射表，如下图所示：



相似变换至全景图后效果图

1. **放缩变换**

按现有原图拼接图的规格，已经所需要最终输出的输出规格，建立输出规格到原始规格的逆向坐标映射表，由该表可以查到每个原始像素对应有多少个输出像素，达到避免像素丢失的问题。输出像素坐标到原始像素坐标以最近邻思想转换。效果如下图所示：



按需要的输出规格放缩变换后的效果图

而在原先代码中，没有逆向的坐标映射表，导致输出端的许多像素没有对应原始像素信息，而出现黑色缺失。



原先的效果：直接对图像做相似变换和放缩产生的像素丢失

1. **权重计算**

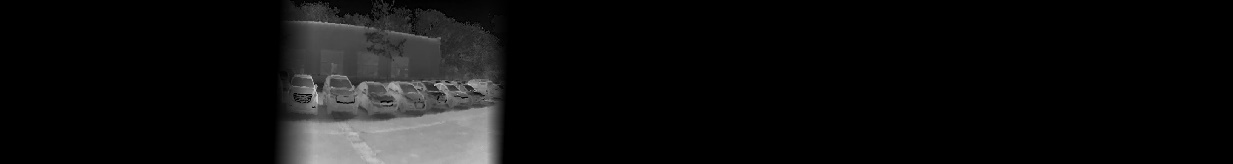
查找每一张图与其相邻两张图的重叠部分，在重叠区域内，建立两边的线性权重。主要是根据上述流程生成的映射表中的输出坐标，生成每张图在全景图中的对应纯白图片，然后对相邻图片做与运算，即可得到重叠区域。利用opencv的findContours函数，找到重叠区域的坐标，找出边界坐标，即可得到重叠区域的精确范围，在范围内建立线性权重即每张图的最终融合权重。

1. **色彩平衡**

由于每个摄像头的曝光不一致，所以需要平衡每个摄像头画面的内容。这里首先计算每个摄像头画面三个颜色通道的平均值，在红外画面情况下，可以被认为是画面的一个平均亮度，计算所有画面亮度平均值，通过平均值 / 每个摄像头画面自身的亮度，计算将画面亮度转换为平均亮度的系数，即得到色彩平衡系数。将其与融合权重相乘即得到最终的权重参数。

需要注意的是这里只对过亮的画面转换到了平均亮度，没有对过暗画面提亮。一方面是为了保证系数不超过1，造成像素值溢出，一方面也是在测试过程中发现亮度越低，融合观感越好。

拼接融合效果图如下：



加上拼接权重，以及色彩平衡后的单幅效果图



加上拼接权重，以及色彩平衡后的拼接效果图

原先的效果图如下：



原先的效果：未赋权重，未做色彩平衡的拼接效果图

这里也附上PTGui得到的预览效果图如下，也需要进行放缩变换以及裁切之后才能使用。

在PTGui中强行设置需要的输出端规格的话，在原始图像规格小的情况下会导致视野超过360度，如下图左右两侧有大面积重复出现的画面。



PTGui效果图：无法将原图保持内容的情况下放缩至所需规格

1. **板端利用映射表实时输出视频**

读取摄像头内容画面，通过映射表参数和权重文件，生成拼接图画面，输出到对应显示端口。

原先的逻辑是融合区域和非融合区域分别设置单独的LUT参数文件，需要遍历12个参数文件。本次改为权重文件与LUT文件同步遍历，一一对应，每个画面的每个像素同步设置相应的权重值和目标全景坐标，将遍历文件次数改为6次，而由于像素信息由赋值改为了“+=”，以融合重叠区域的像素，所以需要将输出图片每次先置为空白。

stitch\_das.cl代码核心修改如下：

\_\_kernel void stitch\_6\_das\_assign(…){

…

for(idx=get\_global\_id(0);idx<1970433; idx+=get\_global\_size(0))

{

outputimg[idx] = 0;

}

barrier(CLK\_GLOBAL\_MEM\_FENCE);

//------------map weight mask----------------------//

for(idx=get\_global\_id(0);idx<data\_size0; idx+=get\_global\_size(0))

{

src\_index = map0[idx\*2];

dst\_index = map0[idx\*2+1];

weight = weight0[idx];

outputimg[dst\_index] += inputimg0[src\_index] \* weight;

}

barrier(CLK\_GLOBAL\_MEM\_FENCE);

…

}