环视标定流程

1. 四鱼眼相机标定

鱼眼相机步骤主要是获取相机的内参、畸变参数、图像分辨率、scale、shift 等参数。

可借助 cv::fisheye 等接口进行标定

标定可分为以下流程

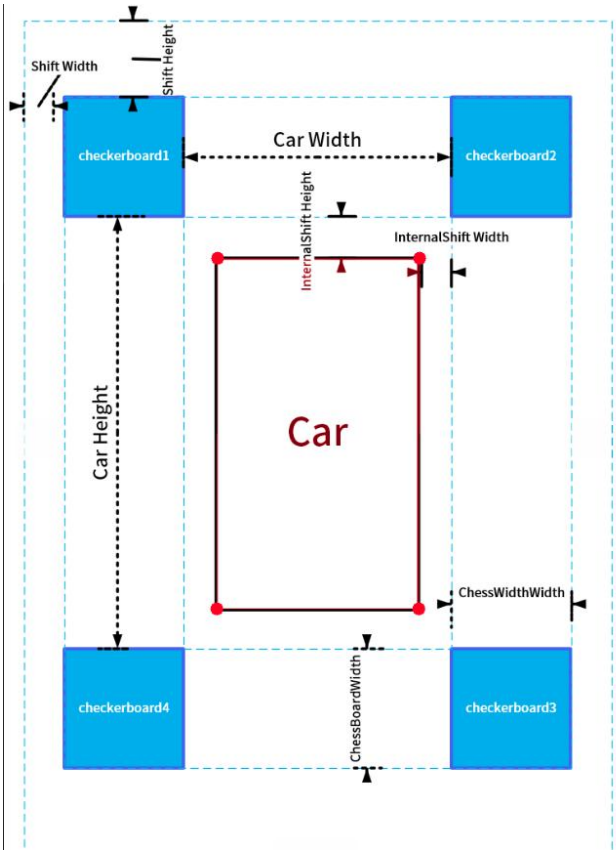
- -读取棋盘格图像并检测角点：使用相机拍摄包含棋盘格的图像，利用 opencv 库检测棋盘格图像角点以及亚像素级迭代定位
- -构建世界坐标系：以棋盘上左上角为世界坐标系，设棋盘格在一个平面上，将 Z 轴设为 0，根据棋盘格尺寸可得世界坐标系下角点坐标。以世界坐标系点和图像角点构建超定方程，最小二乘法求解单应矩阵。
- 求解单应矩阵：以世界坐标系点和图像角点构建超定方程，最小二乘法求解单应矩阵。
- 求解内参：由旋转矩阵为正交阵构建两个约束，构建 Ax=0,SVD 求解内参
- 求解外参：内参、单应矩阵都已知。可得外参
- 求解畸变系数 :通过对原始图像中提取的角点和去畸变后图像中的对应点之间的位置差异进行计算，求解畸变系数

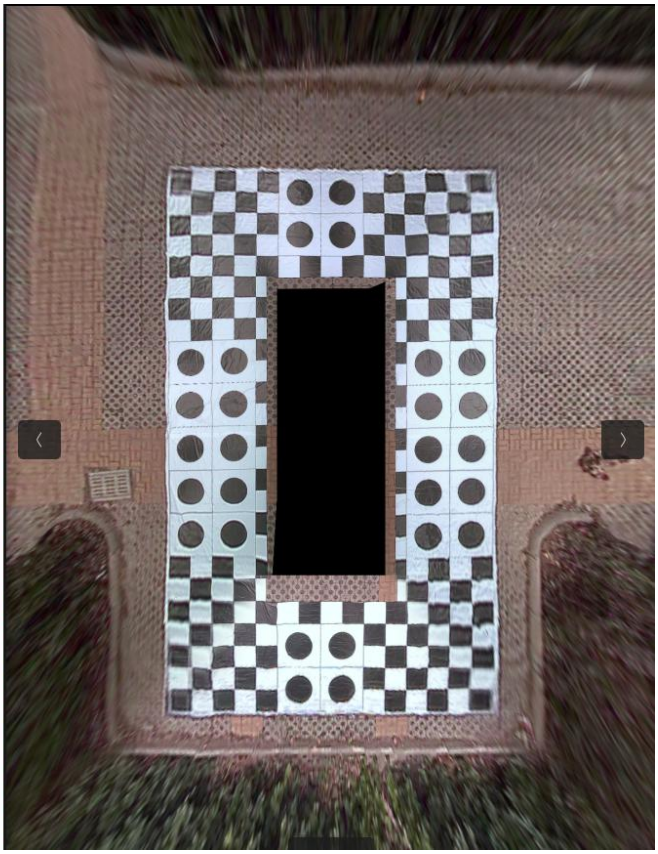
```yaml
%YAML:1.0
---
camera_matrix: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 3.0245305983229298e+02, 0., 4.9664001463163459e+02, 0.,
       3.2074618594392325e+02, 3.3119980984361649e+02, 0., 0., 1. ]
dist_coeffs: !!opencv-matrix
   rows: 4
   cols: 1
   dt: d
   data: [ -4.3735601598704078e-02, 2.1692522970939803e-02,
       -2.6388839028513571e-02, 8.4123126605702321e-03 ]
resolution: !!opencv-matrix
   rows: 2
   cols: 1
   dt: i
   data: [ 960, 640 ]
project_matrix: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ -7.0390891066994388e-01, -2.5544083216952904e+00,
       7.0809808916259806e+02, -2.9600383808093766e-01,
       -2.4971504395791286e+00, 6.3578234365104447e+02,
       -5.6872782515522376e-04, -4.4482832729892769e-03, 1. ]
scale_xy: !!opencv-matrix
   rows: 2
   cols: 1
   dt: f
   data: [ 6.99999988e-01, 8.00000012e-01 ]
shift_xy: !!opencv-matrix
   rows: 2
   cols: 1
   dt: f
   data: [ -150., -100. ]
```

2. 相关参数定义

以下图为例，定义以下参数

- 原始图像宽高 960,640
- bev 图像宽高 1200,1600
- 投影后的 bev 图像 棋盘格边缘离图像边缘的距离 shift_width, shift_height
- 整幅标定板宽高
- 标定板与小车在水平和垂直方向上的间隙大小 InternalShift width/height
- 车辆所在矩形区域的四角坐标,四个红点
- 世界坐标系下棋盘格匹配点



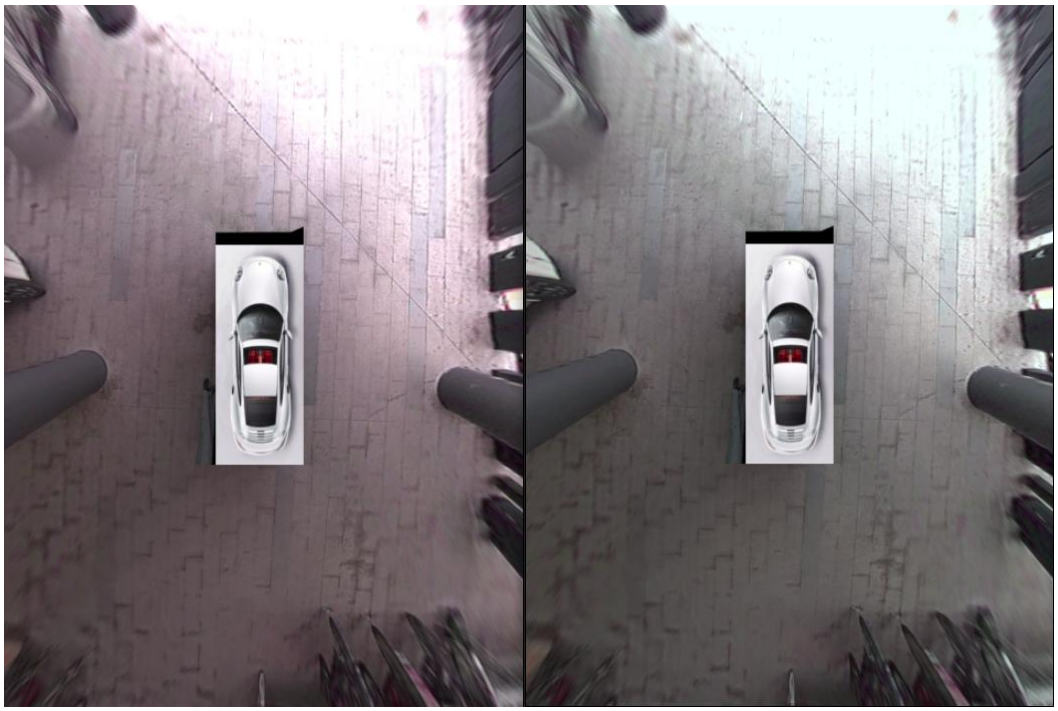


3. 图像白平衡、色彩均衡

不同相机的曝光度不同，导致不同的区域会出现明暗的亮度差，影响拼接效果。

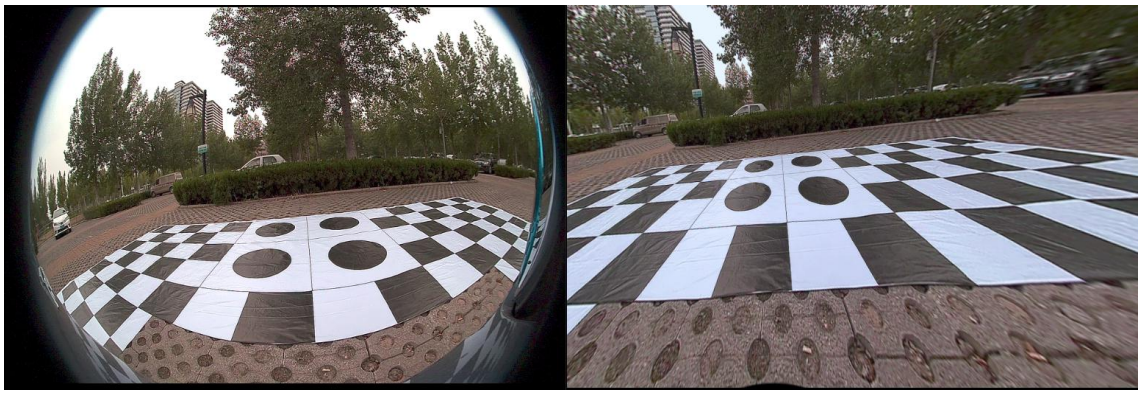
解决方案

- 相机拍摄图像 rgb 三个通道，分别计算四个相机 3 通道均值，计算可得 12 个系数
- 将这 12 个系数分别乘到这 12 个通道上，然后再合并起来形成调整后的画面。
- 过亮的通道调暗,所以乘的系数小于 1。过暗的通道调亮，所以乘的系数大于 1

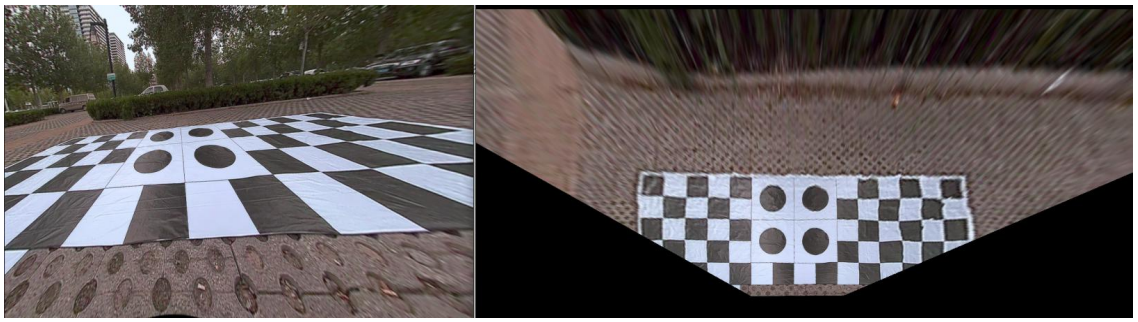


4. 图像去畸变

根据内参、畸变系数对图像进行矫正

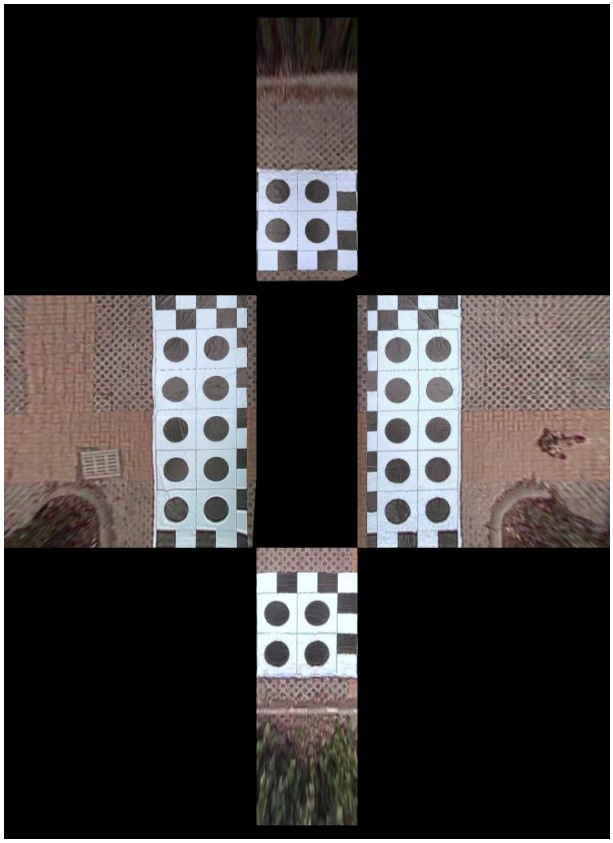


5. 图像投影+旋转



6. 图像拼接

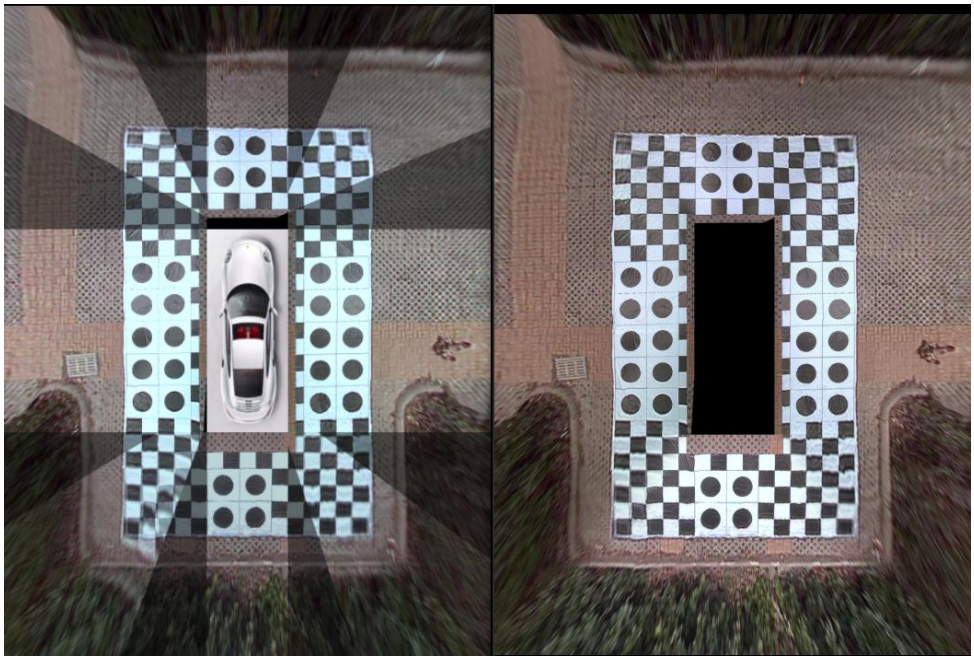
① 拼接非重合区域



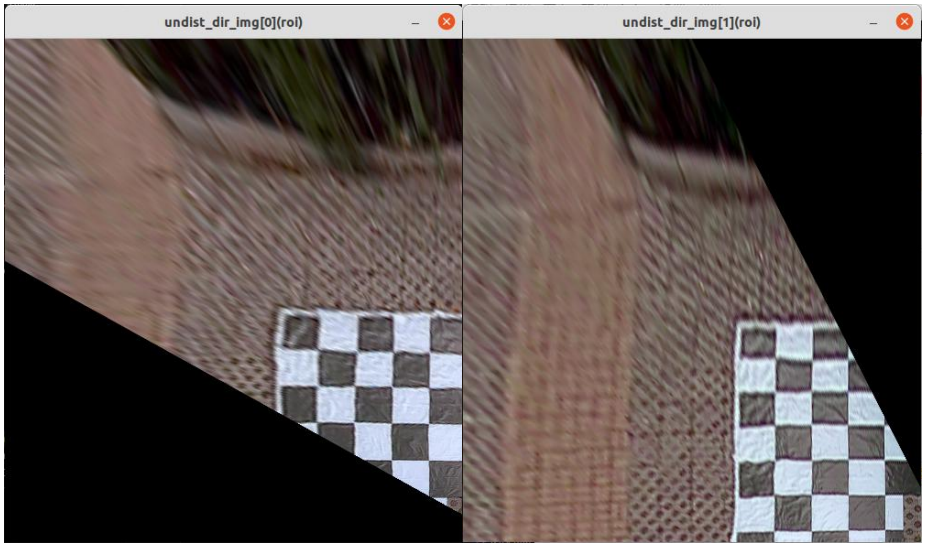
② 计算重合区域权重矩阵

由于校正和投影的误差，相邻相机在重合区域的投影结果并不能完全吻合，导致拼接的结果出现乱码和重影。

下图为重影拼接图和平滑处理后的拼接图。

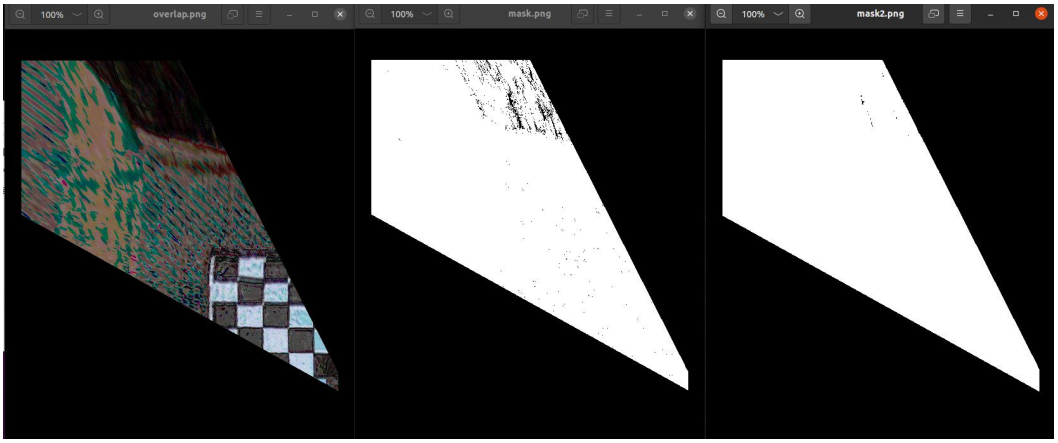


以左上角区域为例（前视和左视的重叠区域）

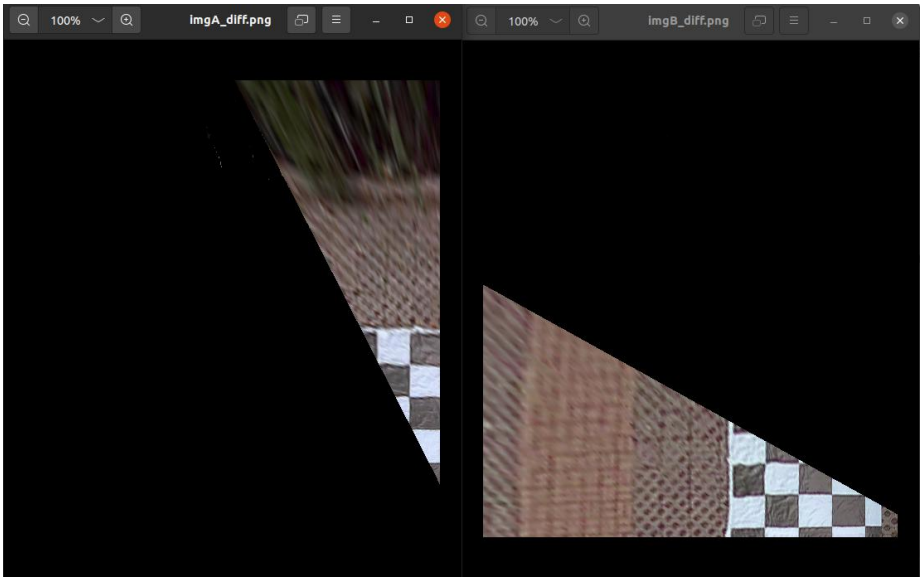


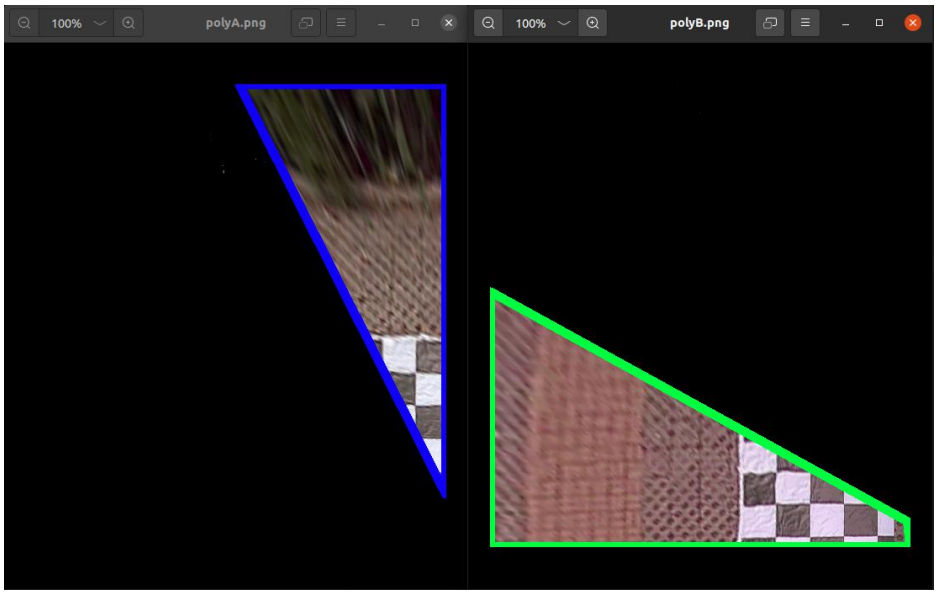
为了平滑处理重合区域的像素点，需要对重合区域进行权重计算。首先，取出投影图中的重叠部分，灰度化并二值化，得到重叠区域的 mask。其次，检测前视、左视图像各自位于重叠区域外部的边界，进而获得逼近的多边形轮廓，记为 polyA、polyB。最后，对重叠区域的每个像素，计算其到这两个多边形 polyA、polyB 的距离 ，则该像素对应权重为 pow(disttoB,2)/[pow(disttoA,2)+pow(disttoB,2)]

- 首先，将重叠区域图像转灰度图像并自适应二值化处理，利用 cv::dilate 去除噪点

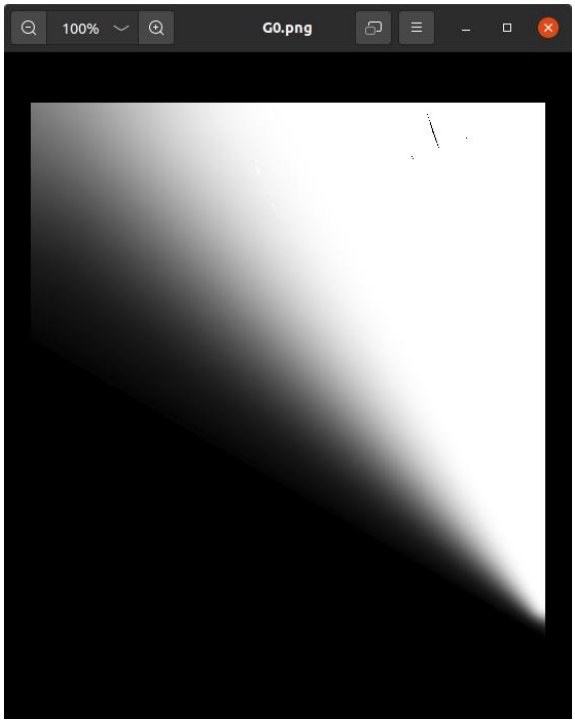


- 其次，检测前视、左视图像各自位于重叠区域外部的边界 ImgA_diff、ImgB_diff，进而获得逼近的多边形轮廓，记为 polyA、polyB

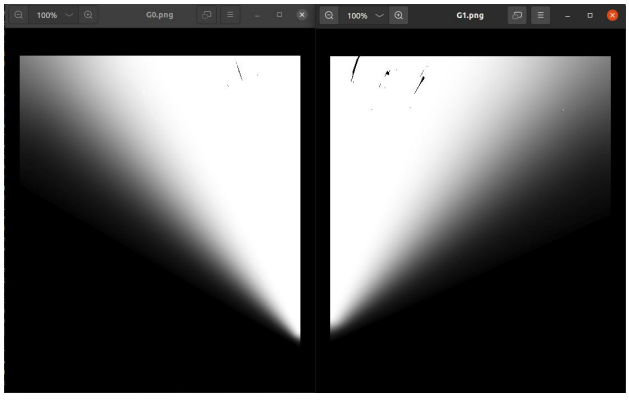


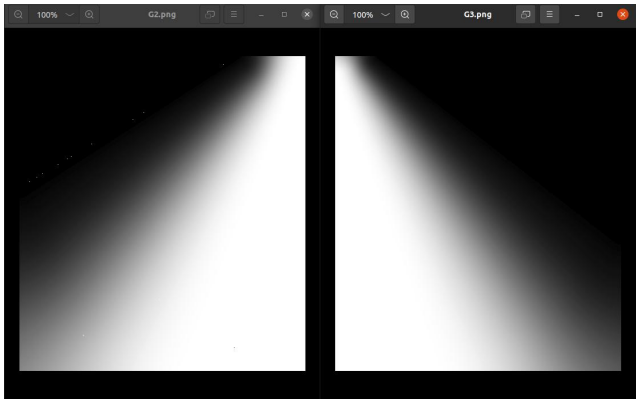


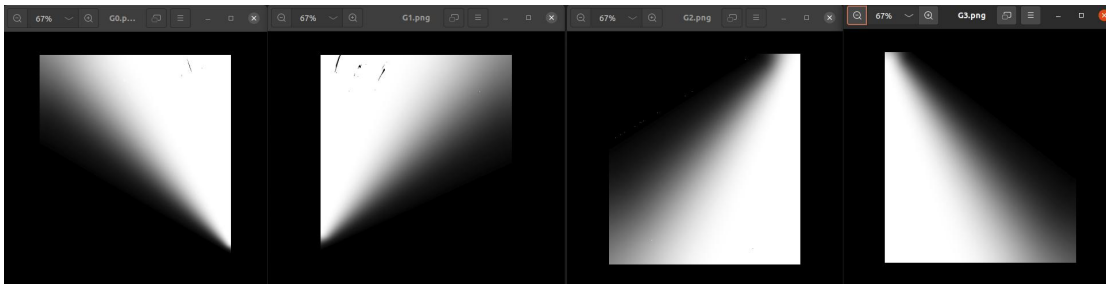
- 对重叠区域中的每个像素，利用 cv::pointPolygonTest 计算其到这两个多边形的距离，并转换为权重



③ 拼接重合区域







只对重叠区域的像素计算 front * G0 + (1- G0) * left

可得拼接图