东 莞 理 工 学 院

本 科 毕 业 设 计

毕业设计题目： **基于 OpenCV 的工程图数据提取及其在 UWB 定位系统中的应用**

学生姓名： **陈俊杰**

学　　号： **201541412127**

院　　系： **机器人学院**

专业班级： **2015 机器人 4 班**

指导教师姓名及职称： **刘文果 副研究员**

起止时间： 2018 年 11 月—— 2019 年 5 月

摘要

本文从铺砖机器人在实际施工场地中室内定位的需求出发，提出了提取工程图数据和实时定位两个部分的解决方案。第一部分中工程图由常规的 DWG 格式文件转换到 JPG 格式图像，再经过 OpenCV 处理，检测出工程图中的角点，并利用数据结构中图论的知识提出了角点定位排序法；第二部分利用排好序的角点，通过 Qt 绘制出等比例的工程图，并利用砖块对工程图进行栅格化，再将砖块信息反馈给铺砖机器人，最后利用 UWB 串口返回的数据通过 Python 和 C/C++的混合编程定位机器人在室内的实时位置。

关键字　铺砖机器人、Trilateration 定位、混合编程

Abstract

This paper is starting from the demand of the indoor location for Mobile Tiling Robot in the real construction site, and put forward solutions consists of two parts: extracting data from engineering drawings and real-time location system. The first part is convert DWG format to JPG at first, secondly process image by using OpenCV, then find out the Harris Corners, finally locate and sort the corners by using graph theory algorithms. The second part is use the processed corners to plot the equal-probability engineering drawings by Qt, then rasterize the region and feedback the tile information to robot, lastly with the data returned from the serial port of UltraWideband, calculate the real-time location of Mobile Tiling Robot by using the mixed program of Python and C/C++.

Keywords Mobile Tiling Robot; Trilateration Location; Mixed Program

目　录

第 1 章 概述

1.1 研究背景与现状

近年来，随着人口红利的消失，各行业的劳动力成本不断攀升，建筑业也不例外。随着建筑业的高速发展，现有劳动力出现供不应求、价格攀升的状态。另一方面，依靠人力进行建筑装修容易出现良莠不齐建筑，建筑豆腐渣工程的报道层出不穷，利用机器人进行建筑装修，可以尽可能使得标准统一，合格率上升，减少人为偷工减料而出现的豆腐渣工程。在“中国制造 2025 规划”等国家政策的推动下，各行各业推进在智能设备、智能制造技术上的迭代升级，建筑行业也需要与时俱进。

根据建筑工地的多重程序，其中包括有打桩、抹灰、刷粉、铺墙砖、铺地砖等等，基于建筑机器人在国内还比较罕见，再根据比较建筑工序运用到机器人的开发难易程度和传统建筑工序中人力分布程度，创业团队选择了铺地砖这一程序进行优化和改良，开发机器人缓冲劳动力的缺陷。

目前，来自荷兰 RoadPrinter 公司的一种大型半自动铺路砖机 Tiger-Stone，该机器能够半自动铺设耐用又美观的砖路。铺路机器分为填料区和铺设区，工人站在填料区进行填料，随着铺砖机器在沙基路面上一点点向前行进，砖块就会因为地心引力自动挤在一块。但是，施工时需要人力将砖块按照有角度的填料槽进行填充。而全自动化的铺砖机器人基本不需要人工辅助，所有程序预先设定，机器人便能高效地进行路面铺设。如图 1-1（a）所示

早在 2014 年，国外 Singapore-ETH Centre Future Cities Laboratory and ROB Techologies 的研究人员研发出一款全自动化能铺地板砖的机器人雏形。该初步模型是基于六轴机械臂进行铺贴，通过红外扫描需要铺贴地砖的区域，然后从机器人取下地砖，能精准地将地砖铺到测量好的位置，工作效率是目前人工铺贴的四倍。如图 1-1（b）所示

现今，国内的大部分房地产公司在转型涉及机器人产业。比如，2018 年 9 月碧桂园计划 5 年内在机器人领域投入至少 800 亿元，其中就包括在建筑行业的多个方面研发机器人，而铺贴地板砖就是其中一个方面。

（a）图为荷兰RoadPrinter公司的半自动铺路砖机Tiger-Stone

（b）图为新加坡一款全自动化能铺地板砖的六轴机械臂

图 1-1 国外两款铺砖机器人的对比

1.2 研究意义

在团队分工中，本课题的主要任务是基于 OpenCV 的工程图数据提取及其在 UWB 定位系统的应用。其中，OpenCV[1]（Open Source Computer Vision Library）是一个跨平台的开源计算机视觉库，其应用领域包括人脸识别、手势识别、图像分割等，而在本课题要运用到 OpenCV 的主要功能是图像处理，由于建筑行业普遍使用的工程图是.dwg 格式文件，鉴于.dwg 格式文件不开源，并且没有 Python 的第三方开发包，加上提取图纸数据困难，所以团队采取了将.dwg 工程图纸转成同规格大小的 JPG 图像格式文件，再利用 OpenCV 库进行对 JPG 工程图像进行数据提取的办法。由于计算机视觉在现今是一个热门且发展迅速的领域，加上 OpenCV 开发文档和论坛社区较为完善，所以利用 OpenCV 对 JPG 工程图像数据提取，可以省去破解不开源的.dwg 文件这一程序，能够极大地节省了开发周期，并且适用性广。

其次，超宽带（Ultra-wideband，简称 UWB），是一种具有低耗电与高速传输的无线个人局域网通讯技术。本课题以超宽带室内定位技术为基础，搭建室内铺砖机器人定位系统，将 CAD 工程图纸数据进行提取，导入铺砖机器人定位系统中，建立室内施工环境全局坐标系，对机器人的当前位置、移动趋势与路径进行远程监控，创新的利用目前精度较高的室内定位方案运用在建筑行业当中。

1.3 本文主要工作

本文从对建筑工程图的认识展开，讨论了将 AutoCAD 工程图.dwg 文件转换为 JPG 图像格式文件的原因，并通过 OpenCV 对 JPG 工程图进行数据提取和

[1] https://opencv.org/

处理，然后将提取出来的工程图数据利用 Qt[2]开发框架绘制出室内区域和砖块摆放样式，最后利用 UWB 室内定位技术将机器人在室内的位置进行实时标记。

1.4 章节安排

本论文共分为六章，内容如下：

第 1 章 概述，主要介绍了本论文的研究背景和现状，研究意义，主要工作及论文的组织结构。

第 2 章 相关技术和开发环境，首先对建筑工程图进行简单的认识，其次分析 OpenCV 在该系统中的应用，并且对 OpenCV 进行基本的认识，最后简要介绍 UWB 室内定位系统原理和算法。

第 3 章 基于 OpenCV 的工程图数据的数据提取，主要说明了采用 OpenCV 提取工程图数据的思路，研究了 OpenCV 在工程图的数据提取和分析各个步骤中的算法分析。

第 4 章 工程图数据在 UWB 定位系统的应用，主要研究了根据工程图数据利用 Qt 绘制室内平面图，然后规划砖块摆放，并将砖块信息返回给机器人，最后通过 Qt 利用 Python 和 C++混合编程绘制出机器人的定位位置。

第 5 章 实验结果，阐述了本课题整个设计的流程图和整个系统的在实际应用场景中的表现。

第 6 章 结论与展望，首先简要总结了本文的一些工作，并对接下来进一步的研究工作做了展望。

[2] https://www.qt.io/cn

第 2 章 相关技术和开发环境

2.1 建筑工程图的认识

建筑工程图是用来表示房屋的规划位置、外部造型、内部布置、内外装修、细致结构、固定设施及施工要求等的图纸。它包括施工图平面图、建筑总平面图、建筑剖面图、建筑立面图和建筑详图。

本课题采用的建筑工程图属于建筑施工图的房屋平面图（文章以下出现的“工程图”时均表示建筑平面图）。而在现今，国内建筑行业在绘制平面图普遍使用 AutoCAD 软件绘制成.dwg 文件格式的工程图。

.dwg 是 AutoCAD 软件保存图形及属性数据的一种二进制文件格式，是制图行业的工业标准，数据结构复杂，主要包括文件头部（HEADER）、应急头部（CONTINGENY HEADER）、表部（TABLES）、实体部（ENTITLES）、块实体部（BLOCKS）5 部分[10]。

2.2 OpenCV 的基础知识

2.2.1 OpenCV 介绍

OpenCV（Open Source Computer Vision Library）是一个开源的是计算机计算机视觉库。OpenCV 采用 C/C++语言编写，可以运行在 Linux/Windows/Mac 等操作系统上。OpenCV 还提供了可 Python、Ruby、MATLAB 以及其他语言的接口。

OpenCV 主体分为五个模块，其中四个模块如图 2-1 所示。还有一个模块是 CvAux 模块，该模块中一般存放一些新出现的实验性的算法和函数，同时还有一些即将被淘汰的算法和函数[1]。

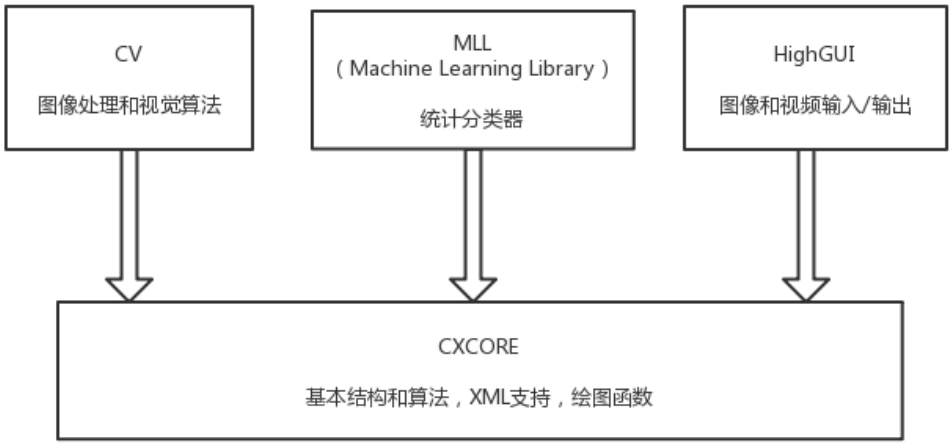


图 2-1 OpenCV 主要的四个模块及其关系图

2.2.2 灰度处理

a) 灰度的概念：

在二指图像（像素只有 0 和 1 两种取值，0 代表黑色，1 代表白色）中进一步加入许多介于黑色和白色之间的颜色深度，就构成了灰度图像。比如常见的灰度级为 0~255 的灰度图像，灰度级越大表示越亮。

b) 灰度图像在图像处理中的作用：

对于一个数字图像处理系统来说，一般可以将处理流程分为 3 个阶段。在获取原始图像后，首先是图像预处理阶段，第二是特征抽取阶段，最后才是识别分析阶段。预处理阶段尤为重要，这个阶段处理不好则后面的工作根本无法展开。

c) 直方图均衡化：

灰度直方图是图像的一种统计表达，它反映了该图中不同灰度级出现的统计概率。把原始图像的直方图变换为均匀分布的形式，达到增强图像对比度的效果。经过均衡化的图像，其灰度级出现的概率相同，此时图像的熵[3]最大，图像所包含的信息量最大。如图 2-2 所示。

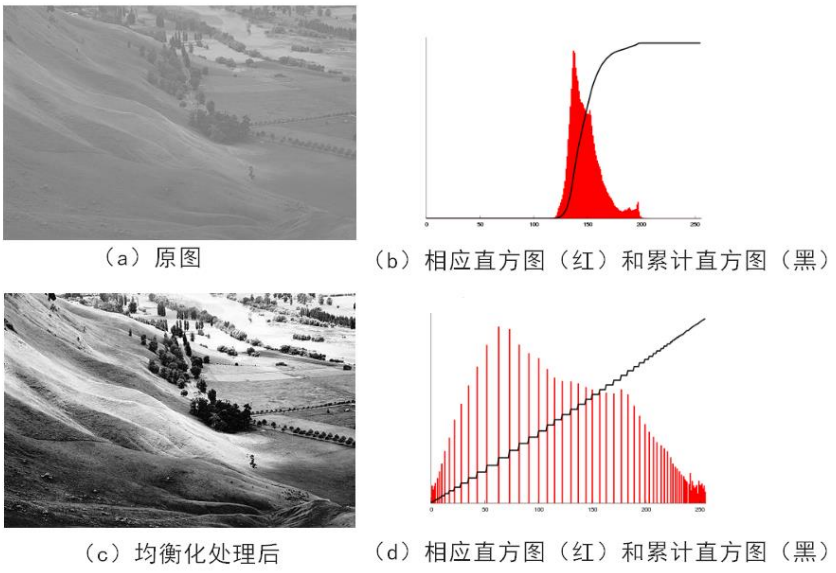


图 2-2 直方图均衡化前后比较

[3] 熵指的是体系的混乱的程度，对焦良好的图像的熵大于没有清晰对焦的图像，因此可以用熵作为一种对焦评价标准。熵越大，图像越清晰。——来源百度百科

2.2.3 Canny 边缘检测法

a) 边缘检测概述：

图像的边缘是图像的最基本特征，边缘点是指图像中周围像素灰度和结构等信息产生突变的那些像素点，即灰度值导数较大或极大的地方。边缘是一个区域的结束，也是另一个区域的开始，利用该特征可以分割图像。

b) 基于微分方法的边缘检测算子：

图像边缘对应一阶导数的极大值点和二阶导数的过零点。如图 2-3 所示。

梯度算子，是基于一阶导数的边缘检测算法，通常是将边界定位在梯度最大的方向。梯度算子包括 Roberts 算子、Sobel 算子、Prewitt 算子等。

高斯-拉普拉斯算子（Laplacian of Gaussian, LoG），是基于二阶导数的边缘检测法，通过寻找图像二阶导数过零点来寻找边界。

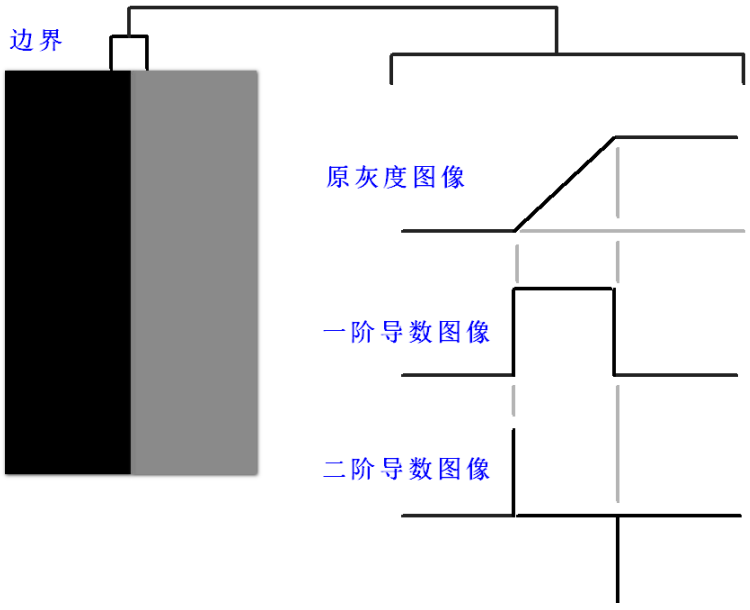


图 2-3 基于微分方法的边缘检测图解

c) Canny 边缘检测算子[2]：

前面介绍的几种都是基于微分方法的边缘检测算法，这些基于微分法的边缘检测，要么边缘定位不精准。要么提高检测噪声的敏感度，而 Canny 算子在边缘精确定位和抗噪声干扰两个条件之间寻求最优的方案。

d) Canny 算法的步骤[3]：

Canny 边缘检测的基本思想就是首先对图像选择一定的 Gauss（高斯）滤波器进行平滑滤波，然后采用非极值抑制技术进行处理得到最后的边缘图像。步骤如下：

（1）用高斯滤波器平滑图像

滤波的主要目的是降噪，高斯滤波会将图像变得模糊，对于在图像中位置 (*m, n*) 的像素点，其灰度值为 *f*(*m, n*)。

经过高斯滤波以后，灰度值变为：

$$g_\sigma(m,n) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-m^2+n^2}{2\sigma^2}} \cdot f(m,n)$$

（2）计算梯度值和梯度方向。

在图像中，梯度是用来表示灰度值变化程度和方向。

第一步得到的灰度值分别点乘一个水平 Sobel 算子 *Hx* 和垂直 Sobel 算子 *Hy* 得到不同方向的梯度值 *Gx* 和 *Gy*

$$Hx = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad Hy = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_x = H_y \cdot f(m,n) \qquad G_y = H_x \cdot f(m,n)$$

综合 *x* 和 *y* 轴的梯度，即可通过以下公式计算梯度值 *G*(*m, n*)和梯度方向θ：

$$G(m,n) = \sqrt{G_x(m,n)^2 + G_y(m,n)^2}$$

$$\theta = arctan\frac{G_x(m,n)}{G_y(m,n)}$$

（3）对梯度幅值进行非极大值抑制。

由于第一步经过高斯滤波，那么边缘有可能被放大了。非极大值抑制就是要过滤掉不是边缘的点。如果一个像素点属于边缘，那么像素点在梯度方向上的梯度值是最大的，否则就不是边缘，将灰度值置为 0。

$$G_T(m,n)=\begin{cases}G(m,n), & if \quad G(m,n)>=T\\ 0, & \text{其他情况}\end{cases}$$

（4）用双阈值算法确定和连接边缘

使用两个阈值 T1 和 T2（T1<T2），如果像素点的灰度值小于 T1，则定义为非边缘点；如果大于 T2，则定义为边缘点；如果介于 T1 和 T2 之间的灰度值，如果该像素点与已经定义为边缘点的像素点相连接的话，那这个像素点也被定义为边缘点。如图 2-4 所示。

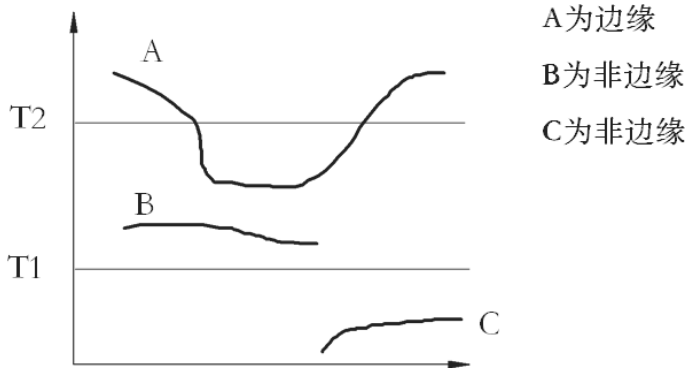


图 2-4 双阈值算法图解

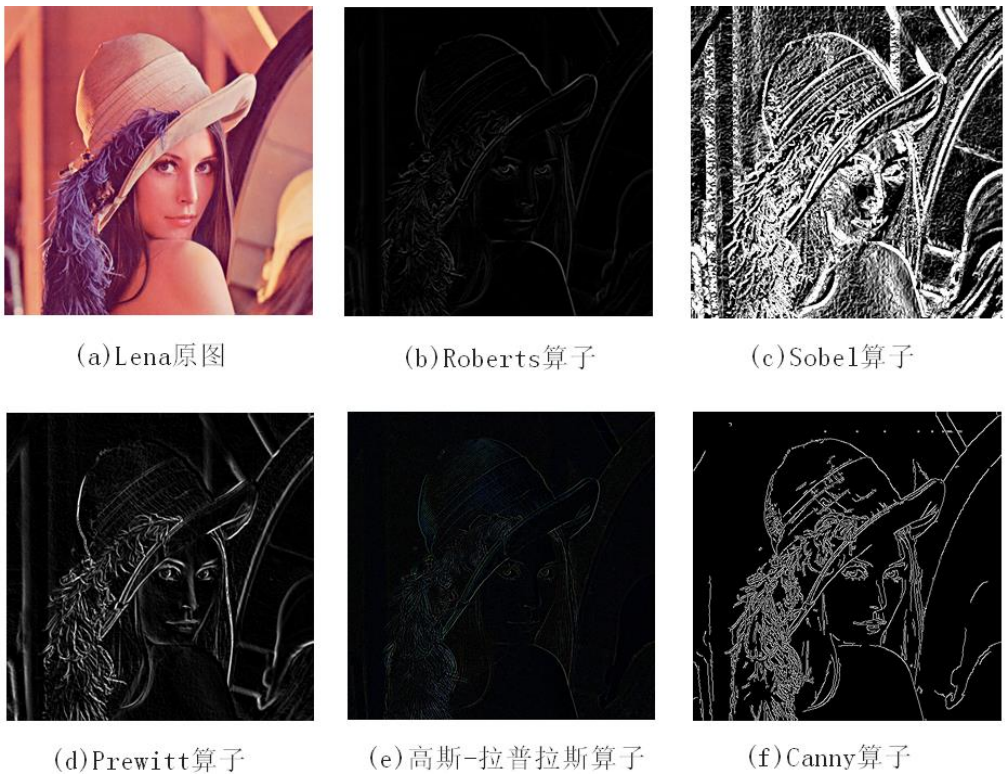


图 2-5 各种边缘检测算子的比较

2.2.4 直线检测

a) 边缘连接：

经过了前面的边缘提取后，由于实际中由于噪声和光照不均等因素，使得边缘检测获得的边缘点是不连续的，必须使用连接过程将边缘像素组合成有意义的边缘，从而将边缘像素组合成完整的边缘。一种寻找并连接图像中线段的方法是 Hough（霍夫）变换。

b) Hough 变换：

Hough 变换实际上是针对图像上的边缘点，它通过将图像直角坐标系变换到极坐标系，找出其共线的点集来实现直线和曲线的拟合。

c) Hough 变换的基本思想[2]：

对于边界上有 n 个点的点集，找出共线的点集和直线方程。

(1) 对于直角坐标系上的一条直线 l，可以用点法式方程来表示直线，如图 2-6（a）所示，直线的点法式方程为：$\rho = x\cos\theta + y\sin\theta$。其中$\rho$为原点到该直线的垂直距离，$\theta$为垂线与 x 轴的夹角，这条直线是唯一的。

(2) 直接坐标系下的一条直线对应着极坐标系上一个点(ρ,θ)，如图 2-6（b）所示，这种线到点的变换就称为 Hough 变换。

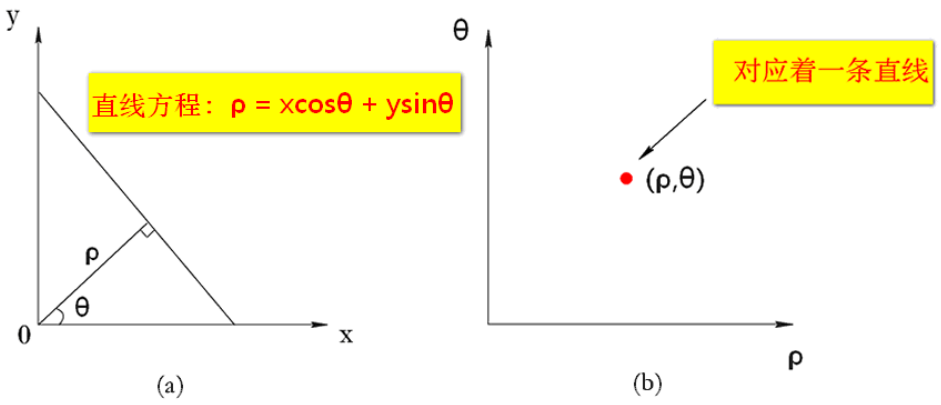


图 2-6 直线在直角坐标系和极坐标系的表示

(3) 那么在（x, y）直角坐标系下，任一个边缘点都有无数条直线，如图 2-7

（a）所示。那么这个边缘点对应的所有直线，在(ρ,θ)极坐标下就成了一条曲线，如图 2-7（b）所示。

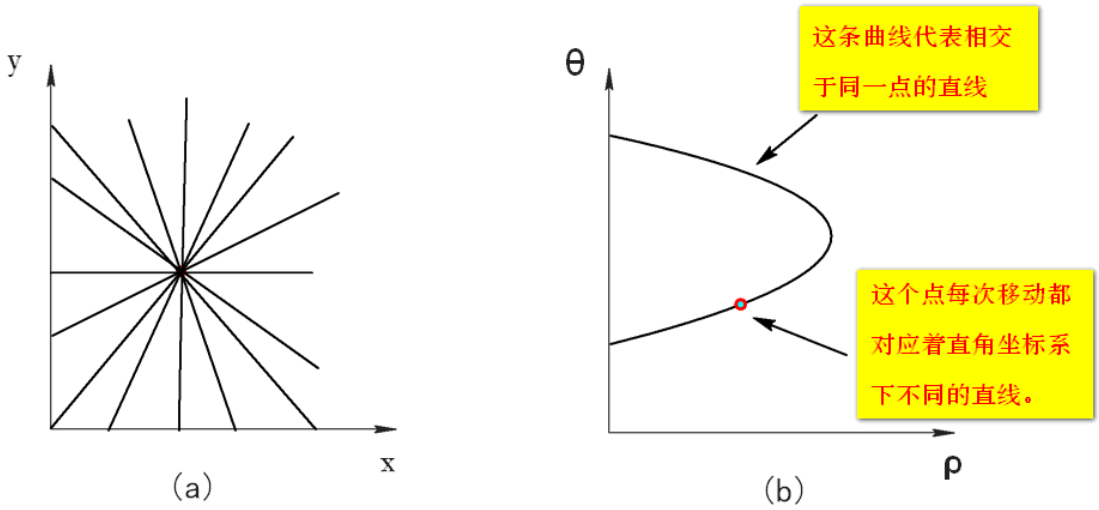


图 2-7 一个边缘点上直线的直角坐标系和极坐标系的表示

(4) 那么（x, y）直角坐标系下，将所有边缘点进行 Hough 变换，就会出现直线吻合的情况。假如存在边缘点 A, B, C, 需要判断这三个点是否在同一条直线上，且在哪条直线上，如图 2-8（a）所示。经过对各个边缘点进行 Hough 变换，得到极坐标系下的三条相交的三条曲线，那么该相交点(ρ', θ')就是 A, B, C 相连的直线，如图 2-8（b）所示。且该直线的方程为：$\rho' = x\cos\theta' + y\sin\theta'$。

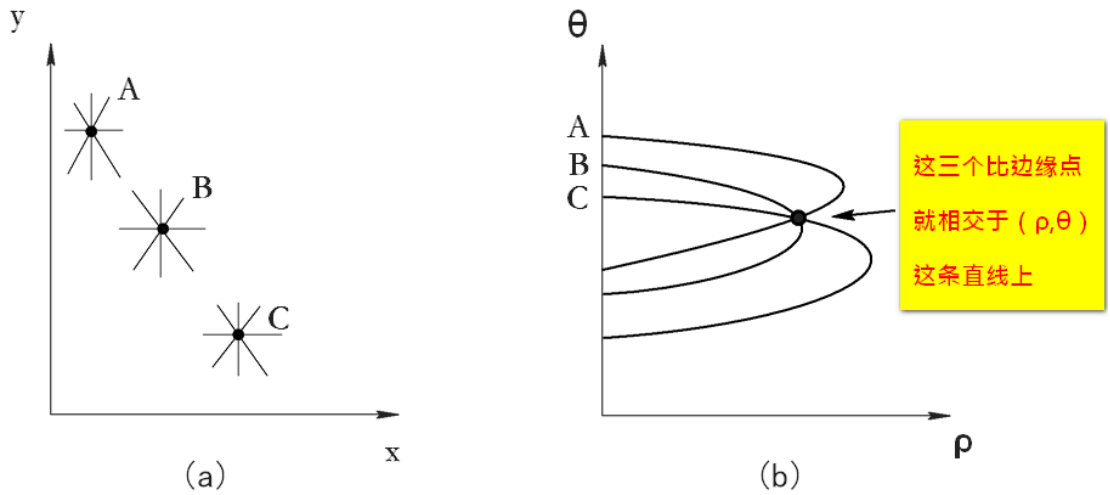


图 2-8 不同边缘点交于同一条直线的描述

d) Hough 变换算法实现：

使用交点累加器，找出(ρ,θ)极坐标系下相交次数最多的点，然后根据该点的参数给对应（x, y）直角坐标系下的直线方程。

算法步骤如下：

(1) 在ρ和θ的极值范围内对其进行 m、n 等分，并初始化为零的一个二维数组 A(m, n)，如图 2-9，用来统计曲线交点累加次数。

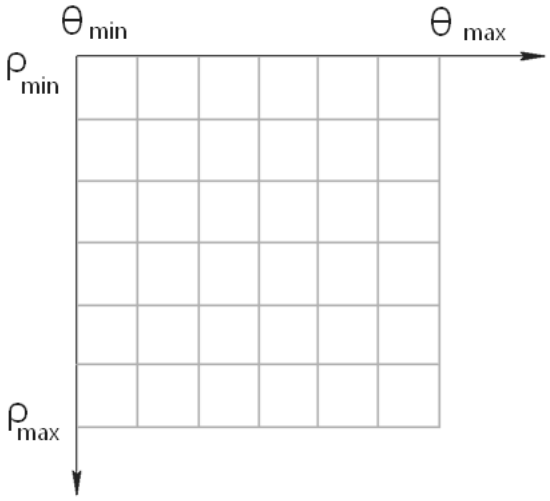


图 2-9

(2) 对图像上的所有边缘点作 Hough 变换计算出所有ρi，θj的值，每计算出一对ρi，θj值，对应二维数组 A 中元素，并且让该数组元素值加 1。

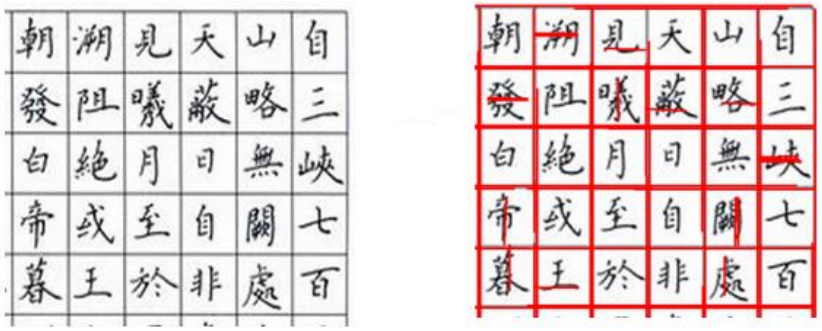
(3) 最后，比较数组 A 中所有元素的值，最大值对应的$(\rho i，\theta j)$就是这些共线点对应的直线方程的参数。

e) OpenCV 中的直线检测：

OpenCV 的 cv.HoughLinesP()得到结果是一个三维矩阵，的直线矩阵格式如下：

$$\begin{bmatrix} 3 & 392 & 3 & 3 \\ 329 & 392 & 329 & 3 \\ 4 & 392 & 328 & 32 \\ 4 & 3 & 328 & 3 \end{bmatrix}$$

那么我们尚且将这个直线矩阵称为直线群（line groups），在直线群中的每一条直线又是一个一维矩阵，其中每条直线中，这四个数字分别表示直线的起始坐标点和终止坐标点。例如第一个[3 392 3 3]表示坐标点（3, 392）和坐标点（3,3）之间的连线[4]。通过直线检测能得到如图 2-10 所示的结果。



（a）原图　　（b）直线检测

图 2-10 直线检测前后比较图

2.2.5 轮廓提取

a) 轮廓的含义[1]：

经过 Canny 边缘检测算法可以根据像素间的差异检测出轮廓边界的像素，但是它并没有将轮廓作为一个整体。一个轮廓一般对应一系列的点，也就是图像中的一条曲线。

在 OpenCV 中轮廓（Contour）就是背景黑，前景白的边缘；与轮廓相对的孔（Hole）是背景白，前景黑的边缘。如图 2-11 所示。其中“c”表示轮廓，又称外轮廓；“h”表示孔，又称内轮廓。

[4] 这些点的位置都是基于 OpenCV 坐标系。

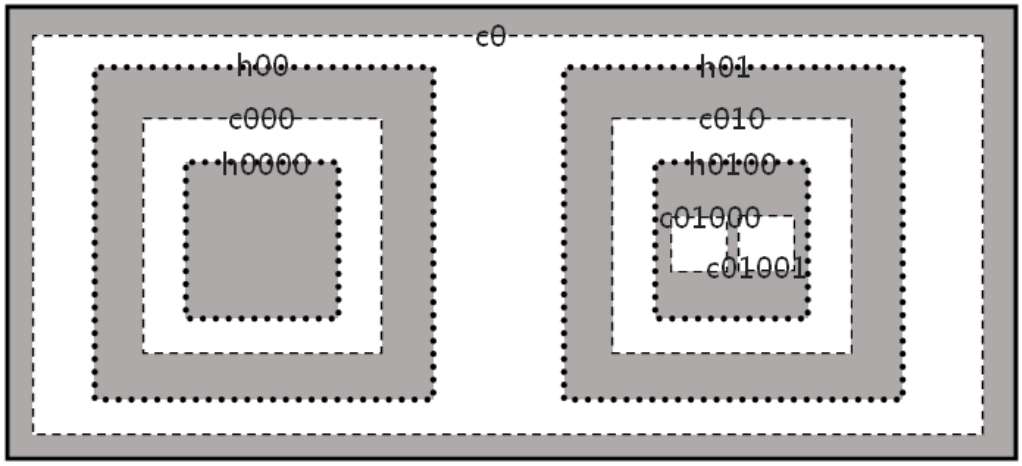


图 2-11 轮廓树图

OpenCV 中的轮廓提取 cv.findcontours()，mode 变量可以设置为以下四个选项之一，如表所示。内外轮廓保存到一个序列 first 中，其中各 mode 的拓扑结构图如图 2-12 所示。

mode 值	mode 的实质含义
cv.RETR_EXTERNAL	只检测最外层的轮廓
cv.RETR_LIST	检测所有内外轮廓并保存到 List 中
cv.RETR_CCOMP	检测所有内外轮廓并将它们组织成双层结构，即在 List 的基础上将 contours 和 holes 进行分层
cv.RETR_TREE	检测所有内外轮廓并且重新建立网状的轮廓结构

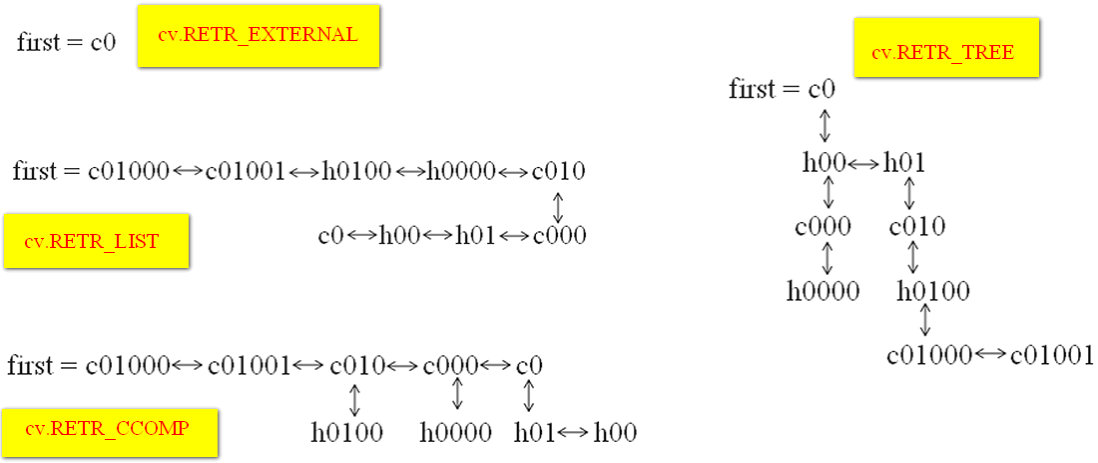


图 2-12 拓扑结构图

2.2.6 角点检测

a) 角点的含义[1]：

角点在图像和视频处理当中被定义为可跟踪的特征点[5]，一般是二维图像边缘曲线上极大值或者图像中亮度剧烈变化的点。角点检测在图像匹配、相机标定、视觉的定位和测量等方面的应用中起着重要的作用[4]。

b) 角点检测技术分类：

第一类是基于图像边缘信息，比如基于边缘曲率的角点检测；第二类是基于图像灰度信息进行检测，如本课题中所应用的 Harris 算法[6]。第一类角点检测中存在对噪声的鲁棒性弱，原因是第一类检测中角点对于边缘太过依赖，如果在边缘提取中出现中断，则对角点的提取非常不利。

c) Harris 经典算法步骤[4][5]：

(1) 计算图像像素在 X 和 Y 方向上的梯度，以及两者的乘积。得到矩阵 M；

(2) 对图像进行高斯滤波，得到新的矩阵 M；

(3) 计算原图像上对应每个像素点的 CRF（Corner Response Function，角点响应函数）值；

(4) 选取 CRF 局部极值点，在 Harris 算法中，特征点被认为是局部范围内的极大 CRF 值所对应的像素点；

(5) 设置阈值，选取角点。当像素点的 CRF 在其领域内为最大值，且大于设定的阈值，则该像素点为角点。如图 2-13 所示。

[5] https://docs.opencv.org/3.4.2/df/d54/tutorial_py_features_meaning.html

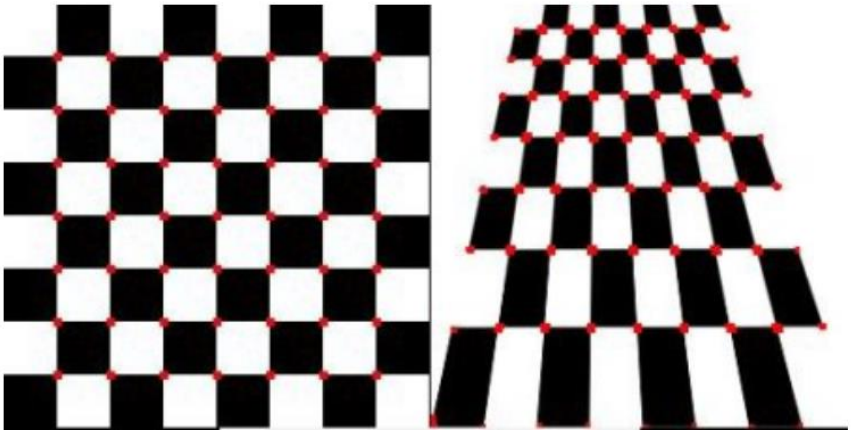


图 2-13 角点检测

2.3 基于超宽带（UltraWideband 无载波通信技术）室内定位的认识

专业术语表

表 2-1 UWB 定位系统专业术语表

简写	英文全称	含义
UWB	Ultra-wide band	UWB(Ultra Wideband)是一种无载波通信技术，利用纳秒至微秒级的非正弦波窄脉冲传输数据
RTLS	Real time of location system	实时定位系统
Anchor		基站，指通过其他方式预先获得位置坐标的节点，在本课题中，Anchor 点位置是人工预先根据实际情况设置好
Tag		标签，在本课题中，即机器人的位置
TOF	Time of flight	TOF 飞行时间测距法，它主要利用信号在两个异步收发机（Transceiver）之间往返的飞行时间来测量节点间的距离
Trilateration		三边测量定位法

2.3.1 UWB 的介绍

超宽带（UWB）技术是近年来兴起的一种高速数据传输的无限通信技术，其传输速度之快能高达 1000Mbps。UWB 具有以下优点：传输速率高、消耗低、功率小、带宽高、鲁棒性强。正是由于这些优势，使得 UWB 在室内定位领域的应用中能得到精确的结果，精度级别能达到 10cm 以内，在有遮拦的情况下仍能有

30cm 以内的良好发挥。UWB 调制采用 ns（纳秒）级的脉冲宽度进行快递上升和下降脉冲进行数据传输。UWB 不需要常规窄带调制所需的 RF（Radio frequency，又称无线电频率）频率变换，脉冲成型后能直接送到天线发射。UWB 所产生的辐射比 WIFI 的低，只有手机辐射的千分之一，因此对人体影响很小[7]。

2.3.2 UWB 的定位技术原理

超宽带（UWB）定位技术，主要采用两个步骤：第一，TOF 测距；第二，通过 TOF 所测到的距离进行定位，但目前成熟的定位技术分别有 TOA（Time of Arrival，到达时间）算法、TDOA（Time Difference of Arrival，到达时间差）算法、Trilateration 三边定位算法。本课题所采取的 UWB 定位技术的组合是 TOF 测距和 Trilateration 三边定位算法[9]。

2.3.3 TOF 测距

TOF 距离测量机制是使用双向通讯时间，又称双向飞行时间法（TW-TOF, two way-time of flight）。TOF 测距可以实现前向和反向两个方向的测距，在前向测距中，测距的主动节点是移动节点，在此系统中为标签（Tag）。由 Tag 发送数据包，在此系统中的远程节点也就是基站（Anchor），接收数据包马上进行自动响应；反向测距的主动节点是 Anchor，由 Tag 进行通知，Anchor 采取测距操作，再由 Anchor 发送数据包，Tag 接收数据包再立刻进行自动响应。如图 2-14 所示，该图是前向测距的进行过程，数据帧和回复帧的发送或者接收的时刻被 Tag、Anchor 记录，然后获取两个时间差(*Ta1-Ta2*)与(*Tb1-Tb2*).

无线电波在空气中的传输时间如公式（1）所示：

$$T_{TOF} = [(Ta_1 - Ta_2) - (Tb_1 - Tb_2)]/2 \qquad (1)$$

两节点间的距离如公式（2）所示：

$$S = C \cdot T_{TOF}\ , \quad (\text{C 为光速, 即 } C = 3 \cdot 10^8\ ) \qquad (2)$$

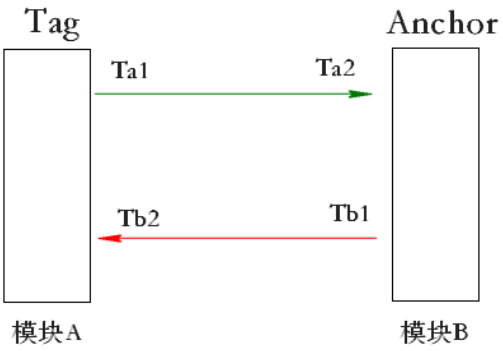


图 2-14 前向测距图

2.3.4 Trilateration 三边定位算法

a) Trilateration 算法的理解：

三边测量法(Trilateration)。通过测量标签（Tag）到三个或三个以上的基站（Anchor）的距离，运用几何原理求解目标位置。如图 2-15 所示。

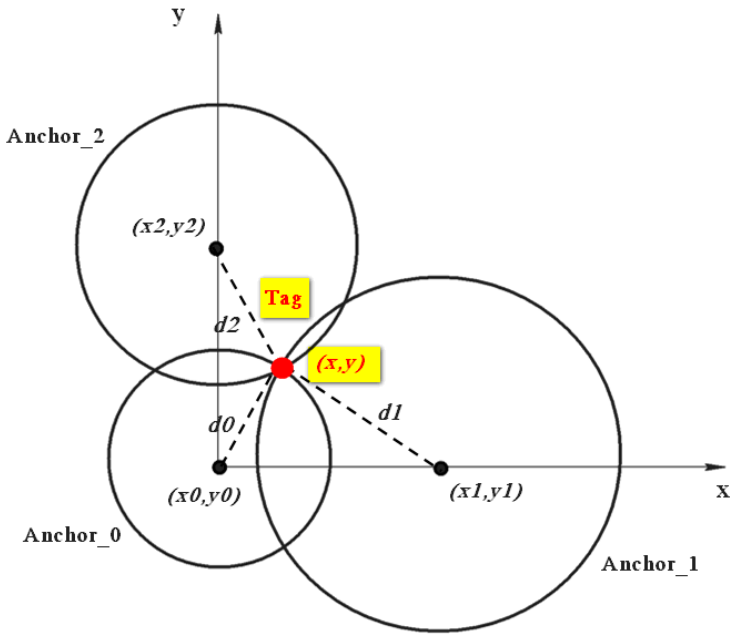


图 2-15 Trilateration 三边定位算法示意图

假如 Tag 坐标位置为（x, y），通过 TOF 测出 Tag 与三个 Anchor 的相对距离为d_i（其中 $i = 0, 1, 2$），则有距离公式[8]：

$$d_i^2 = (x - x_i)^2 + (y - y_i)^2 \qquad (1)$$
$$= x^2 - 2x_i x + x_i^2 + y^2 - 2y_i y + y_i^2 \text{，} \quad \text{其中 } (i = 0, 1, 2)$$

公式（1）中存在非线性项x^2和y^2。通过d_i^2与d_N^2的互减操作可消去非线性项，假如（i = 0, 1, 2, … N-1），

$$d_i^2 - d_N^2 = -2x(x_i^2 - x_N^2) + x_i^2 - x_N^2 \qquad (2)$$
$$-2y(y_i - y_N) + y_i^2 - y_N^2, \quad \text{其中 } (i = 0, 1, 2 \ldots, N-1)$$

当存在室内中有 N 个 Anchor，便可得到 Tag（x, y）的多项式，那用矩阵来描述为 b=AX, $X = \begin{pmatrix} x \\ y \end{pmatrix}$，其中，

$$b = \begin{bmatrix} d_1^2 - d_N^2 - (x_1^2 + y_1^2) + (x_N^2 + y_N^2) \\ d_2^2 - d_N^2 - (x_2^2 + y_2^2) + (x_N^2 + y_N^2) \\ \vdots \\ d_{N-1}^2 - d_N^2 - (x_{N-1}^2 + y_{N-1}^2) + (x_N^2 + y_N^2) \end{bmatrix} \qquad (3)$$

$$A = -2 \begin{bmatrix} x_1 - x_N & y_1 - y_N \\ x_2 - x_N & y_2 - y_N \\ \vdots & \\ x_{N-1} - x_N & y_{N-1} - y_N \end{bmatrix} \qquad (4)$$

当 N = 3 时，A 为 2*2 方阵且可逆，那么 $X = A^{-1}b$ 当 N>3 时，可用

$X = (A^T A)^{-1} A^T b$ 求解。

b) Trilateration 算法的优劣：

Trilateration 三边定位算法相对 TDOA 算法较传统，根据几何原理的定位逻辑也相对简单易懂；但由于每个模块的硬件和功耗不尽相同，难免会出现误差，从而导致 Trilateration 算法图示里面的三个圆未必刚好交于一点，但一定相交于一个小区域。所以就更需要 TOF 测距提供更加精确的数据或者部署多于 3 个 Anchor 进行定位。

2.4 开发环境

（1） 开发环境的系统：Ubuntu16.04

（2） 开发语言使用：Python3.6

（3） 开发的 UI 框架：Qt

第 3 章 基于 OpenCV 的工程图数据的数据提取

3.1 功能需求分析

由于建筑行业的图纸是普遍使用 AutoCAD 绘制的.dwg 格式的文件。就目前而言，AutoCAD 提供二次开发工具包的平台只有为数不多的几个，分别是 VistualLisp、VBA、ObjectARX 和.NET。

但为了时效性和跨平台性，本课题系统使用的是 Ubuntu＋Qt 框架 ＋Python 进行构建。AutoCAD 提供的二次开发平台中没有 Qt 或者 Python，虽然 DWG 文件是二进制文件，但是 Autodesk 并没有开源 DWG 文件的读写内容和方法，进而让团队思考另外的出路。

在经过团队的成员多次讨论和尝试之后，发现 AutoCAD 平台具有一个可导出功能，能够将工程图 DWG 文件导出成对应比例为 1:1 的 PDF 文件。

3.2 本章业务流程图

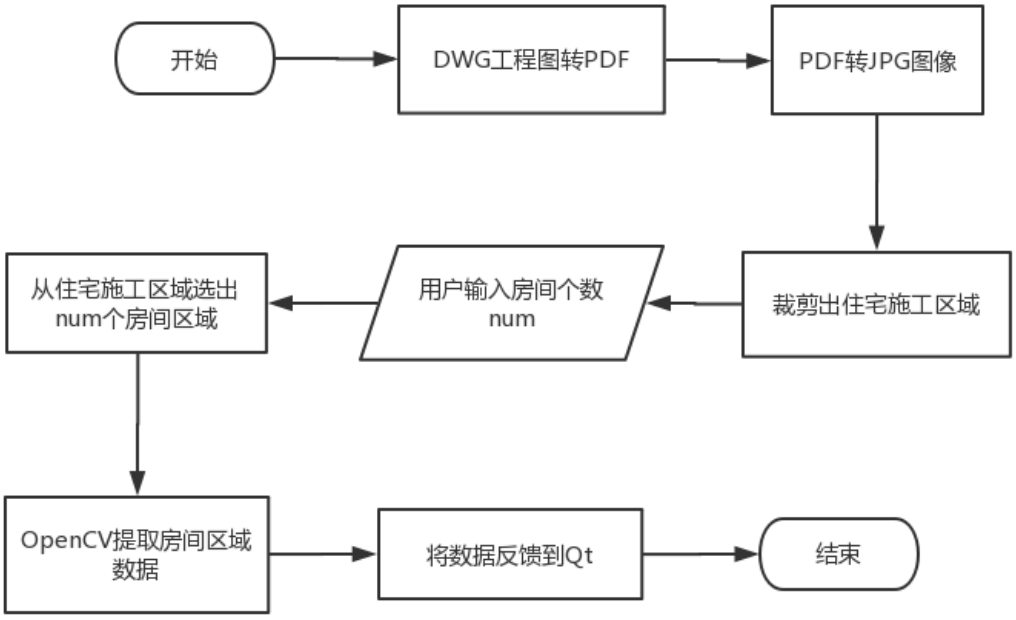


图 3-1 本章主要流程图

a) 预处理：

在导出 PDF 文件之后，在没有办法直接在 PDF 文件下直接进行数据的提取的情况下，团队再次讨论和分析，最后决定采用 OpenCV 这个计算机视觉开源库进行图像处理，那么在采用这个决定之后，就需要将 PDF 格式文件转换为图像格式，那么就可以利用 Python 其中一个库 Wand，能够将 PDF 文件转换为对应的 JPG 格式的图片文件，主要的代码如下：

```
from wand.image import Image as wi

pdf = wi(filename='./cad.pdf', resolution=300)
pdfImage = pdf.convert("jpeg")
i = 1
for img in pdfImage.sequence: #有多张 pdf
    page = wi(image=img)
    page.save(filename=str(i)+".jpg")
i += 1

# 只有一张 pdf，就可以使用下面语句
# img = pdfImage.sequence[0]
# page = wi(image=img)
# page.save(filename="cad.jpg")
```

在将 PDF 文件转换为 JPG 文件之后，需要分析 OpenCV 需要对工程图纸进行哪些图像处理和数据提取。首先，经过文件格式转换之后，获得一张 A3 纸大小且比例尺为 1:50 的工程图，在这张工程图纸中，除了包含住宅施工区域（即 ROI（region of Interest）区域以外），还包含了许多占用空间的空白区域，那么首先需要裁剪掉空白区域，提取出 ROI 区域。在提取 ROI 区域之前，先来认识 OpenCV 坐标系。

OpenCV 的坐标系如下图 3-2（a）所示，而常规的直角坐标系如下图 3-2（b）所示，我们尚且称 OpenCV 图像中的 x 轴为横轴，y 轴为纵轴。在本课题中，应用到 UWB 室内定位的过程的时候会把 OpenCV 的坐标系转换为常规的直角坐标系，原因有两点：第一、为了规范化，在数学教材中常见的坐标系都是如图 3-2（b）所示，那么在本课题按照常规，规定 y 轴向上为正方向，且沿着 y 轴延伸的方向纵坐标数值越大。第二、为了便于在 Qt 上按照这种统一规定的坐标系进行绘图。

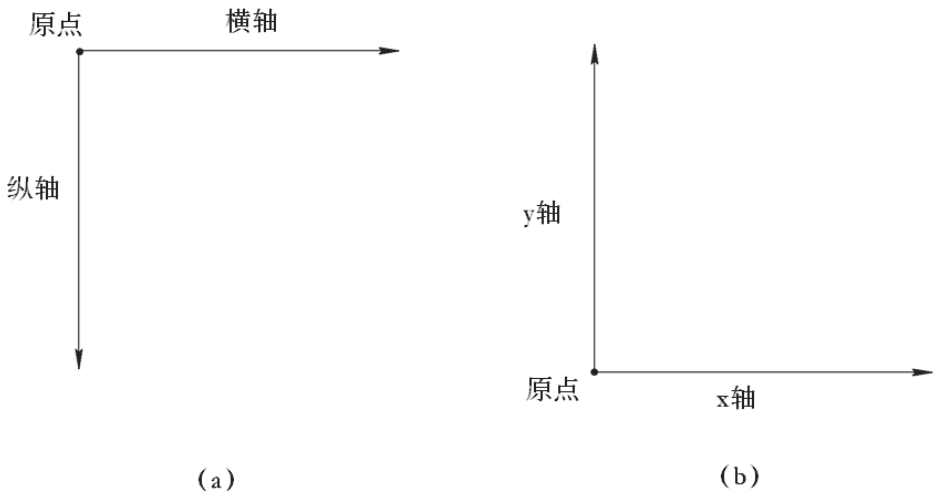


图 3-2 OpenCV 坐标系和 Qt 坐标系比较

那么了解完 OpenCV 坐标系后，提取 ROI 区域的一般方法是如下，

```
1.    src = cv2.imread("cad.jpg")
2.    roi = src[80:150,60:300]
```

其中 src 为 OpenCV 读取的的图像，OpenCV 横轴对应是 column，而纵轴对应的是 row。由于 OpenCV 的矩阵数据结构 Mat 读取的原因，造成图像区域定位和顶点定位矩阵形式不一致[6]，那么在图像区域定位中，矩阵中 80:150 为 row 的始终位置，60:300 为 column 的始终位置，如图 3-3（b）所示，即能提取出特定位置的 ROI 区域。

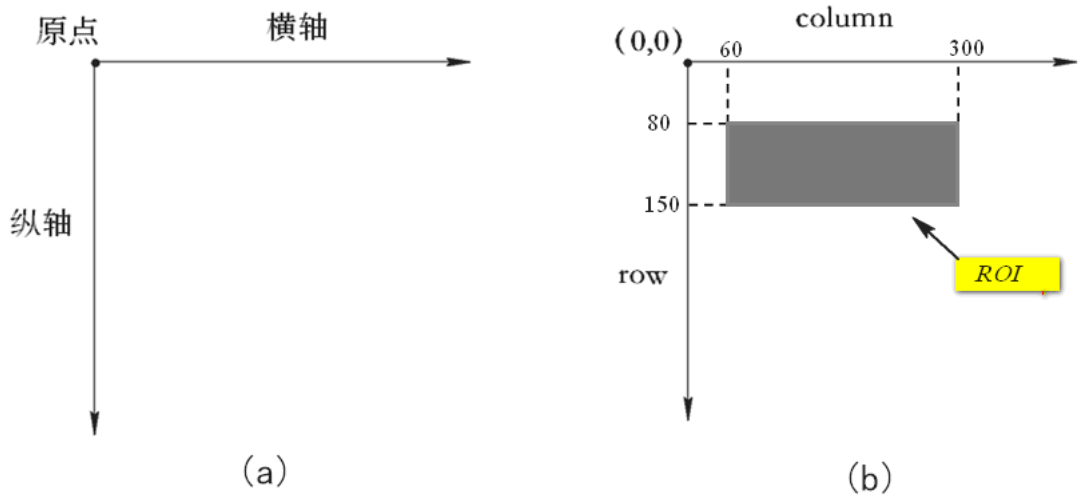


图 3-3 ROI 分析图

但在不同的工程图纸上，无法确定住宅施工区域（Residential Construction Area，简写 RCA）的准确大小和位置信息，所以只采用一般 ROI 提取方法进行提取 RCA 是行不通的，那么再根据工程图纸中的绘图方式，其中 RCA 集中在工程图纸的中央，且整个区域为一个全包围区域，那么我们就可以采取轮廓区域提取的办法将 RCA 提取出来。

在 OpenCV 中，提取最外层轮廓的方法上面已经提到过，只需要把 cv2.findcontours()函数的 mode 设置成为 *cv.RETR_EXTERNAL* 即可，又或者 mode 设置成为 *cv.RETR_TREE* 将所有轮廓查找出来，其中最外层的轮廓（即 contours[0]）就是住宅施工的总区域。那么经过对总轮廓进行裁剪之后，得到如图 3-4 所示的工程图片。

[6] https://stackoverflow.com/questions/25642532/opencv-pointx-y-represent-column-row-or-row-column

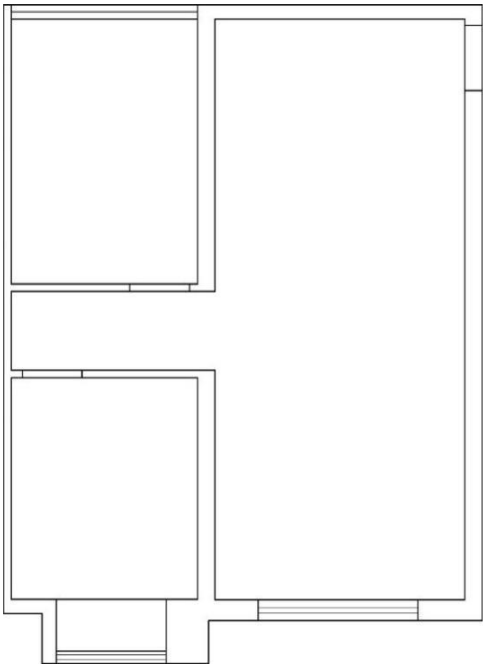


图 3-4 住宅施工区域

3.3 工程图进行房间区域划分

对工程图进行房间区域划分的前提，需要用户人工输入房间个数 num，例如这个住宅施工区域需要进行铺砖的房间有 3 个，首先筛选掉最外层轮廓（即 RCA），然后根据轮廓面积为比较索引，通过快速排序就能将面积最大的前三个轮廓给提取出来了，过程如下状态图 3-5。划分区域的结果为图 3-6。

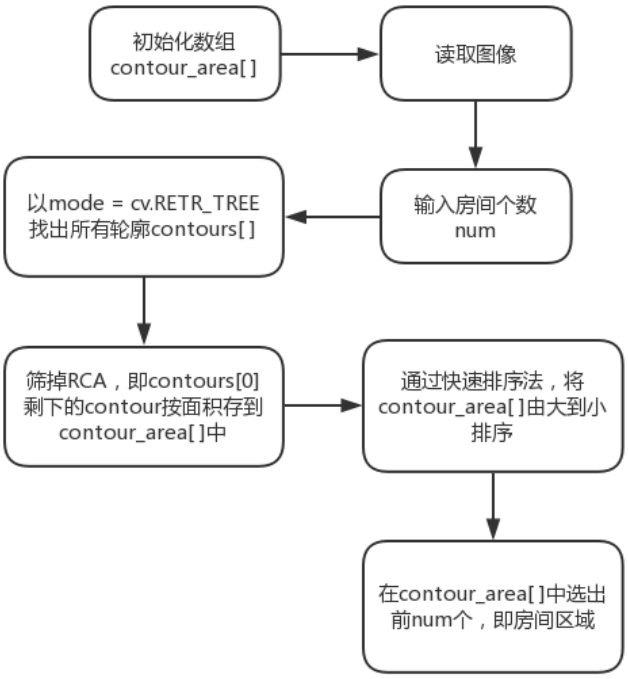


图 3-5 房间区域划分状态图

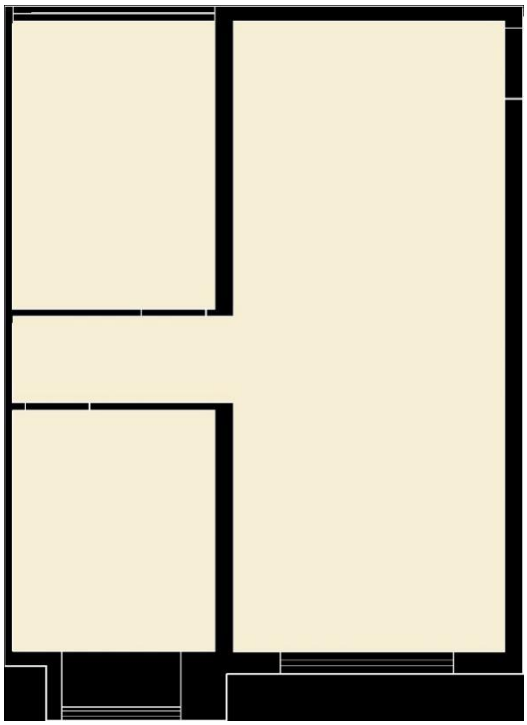


图 3-6 房间区域划分图

3.4 提取区域数据

本课题考虑的房间区域属于矩形或者矩形组合的轮廓，暂且不考虑其他多边形的轮廓（例如：三角形，圆形），那么首先讨论的问题是需要有两个，第一：提取区域的哪些数据；第二：如何提取这些数据。

a) 提取区域的哪些数据：

由于本课题后面是采用 UWB 进行室内定位的，那么 UWB 定位系统中能够进行二维的平面定位，也可以进行三维的立体面定位。在本课题中，砖块铺设目前只需要考虑二维平面室内面积大小和形状即可。那么根据前面介绍的 OpenCV 坐标系可知，在图像中需要找出房间区域的坐标系原点；还要找出房间区域的长（length）和宽（width）。

既然要找出原点和区域的长宽，那么首先需要找出房间区域中哪些是需要的点，哪些点能够定义为坐标原点？而区域的长宽又可以怎么获取？

b) 矩形区域：

现在先讨论单纯的矩形区域（Rectangular region，简称 RR）。RR 一共有四条边，有四个直角，那么就有四个顶点。

RR 中的四个顶点能通过 OpenCV 的角点检测就能够找出如图 3-7（a）所示四个红色的点。为了让这四个角点明显一点，那采取了以角点为圆心，半径为 5 个像素绘制了四个圆，如图 3-7（b）所示。那么在这四个角点中，任意一个作为坐标系的原点都可以。而 RR 的 length 和 width 可以根据裁剪图像的高度像素个数和宽度像素个数，再通过 dpi（dot per inch, 每英寸点个数）和 ppi（pixel per inch, 每英寸像素个数）计算将像素转换为距离即可。

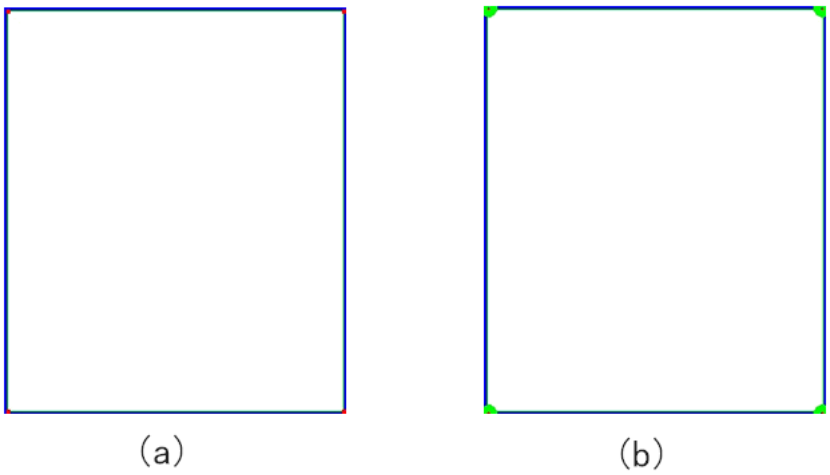


图 3-7 RR 角点检测图和角点标记图

c) 矩形组合区域：

接下来介绍的是矩形组合区域（Rectangular composite region，简称 RCR）的数据提取。

哪些属于 RCR 呢？如图 3-8（a）所示，所谓 RCR，即由多个矩形进行拼合组成的区域，下图 3-8（a）这块白色的房间区域即可理解成是通过有两个矩形进行拼合而组成的区域。

根据常规直角坐标系，是将左下角的顶点作为坐标系的原点。在上面这个矩形组合区域中，直接将左下角作为坐标系原点不太合理，那么利用 OpenCV 提供的函数 cv2.rotate()进行顺时针旋转两次即可得到修正过的区域图纸了，如图 3-8（b）所示。

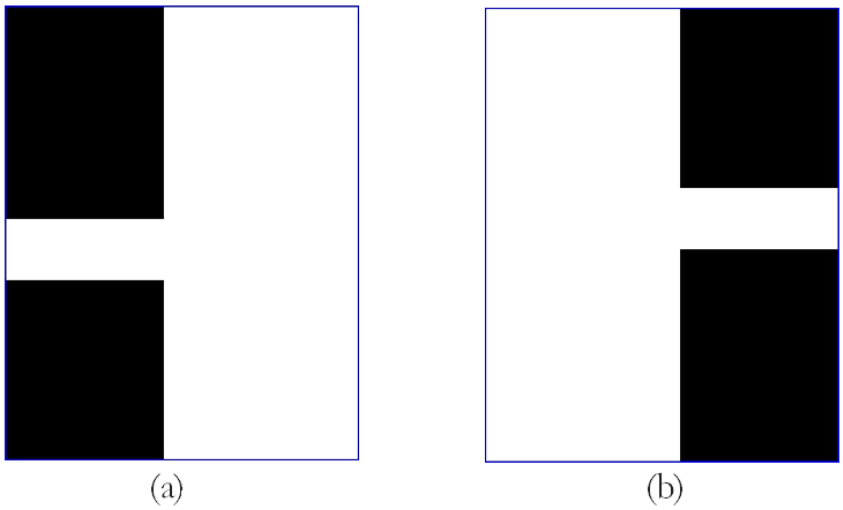


图 3-8 RCR 原图和修正图

在得到这个修正后的 RCR 以后，就能将左下角作为坐标系原点了，接下来需要分析哪些数据是需要进行提取的。根据前面提到的矩形区域（即 RR），需要提取的数据有坐标原点和区域的长宽，但对于 RCR 而言，单纯得到坐标原点和区域长宽，难以在 Qt 完整绘制区域图像，那么接下来讨论本课题设计的角点定位排序法。

所谓角点定位排序法，其实就是将 OpenCV 检测出来的角点进行定位且排序。

第一步：定位角点。那本课题中，定位出角点在图像中的实际位置是根据像素遍历法，又称为暴力遍历定位法。（即，将图像的像素从(0,0)点一直遍历到图像中最后一个像素点）。得到图 3-9（a），为了让角点明显一点，那采取了以角点为圆心，半径为 5 个像素绘制了 8 个圆，得到 3-9（b）。

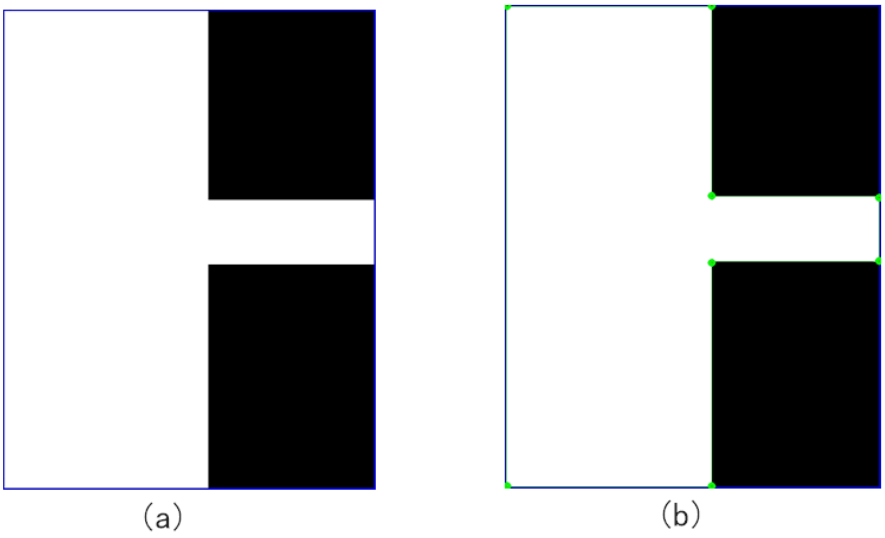


图 3-9 角点图和标记角点图

具体遍历定位过程如下：

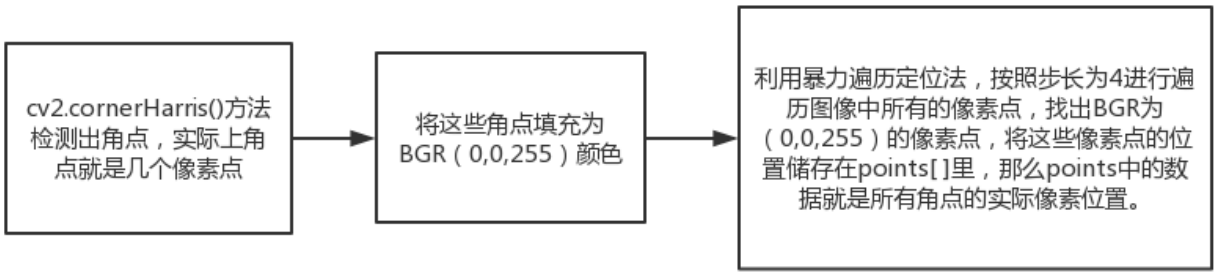


图 3-10 角点遍历定位状态图

第二步：排序角点

何谓排序角点？如何排序角点呢？

如图 3-11（a），是实际图像中提取角点得到的图像；而图 3-11（b），是通过暴力遍历定位法所得到的 points 数组中角点的位置。但是从计算机的角点来看，单纯知道图 3-11（b）这些角点的位置，是无法绘出相应区域。因为计算机并不知道哪两个角点是在同一条直线上，哪些点不能相连。正是因为如此，那么就需要将这些角点进行排序。

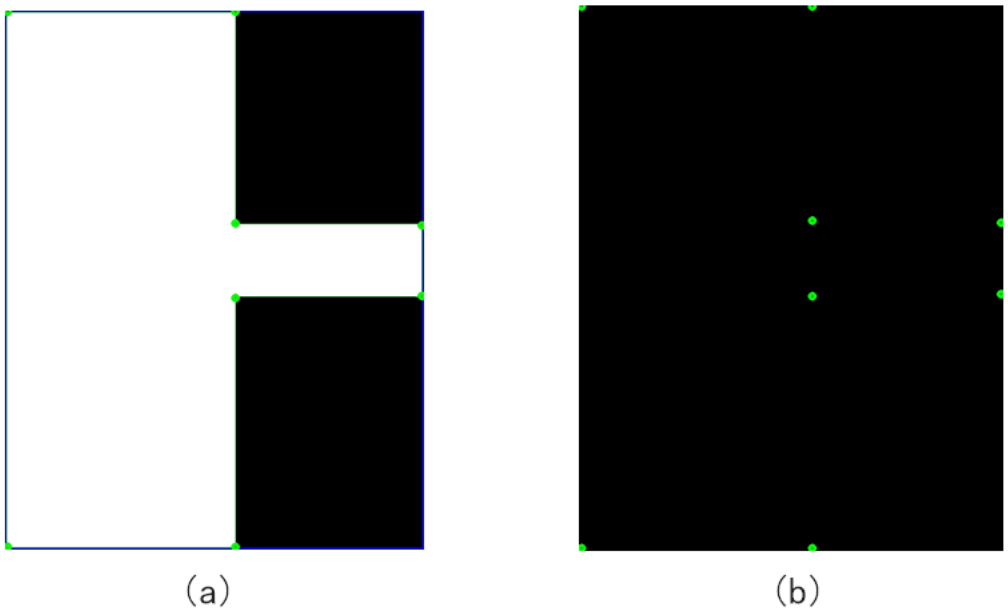
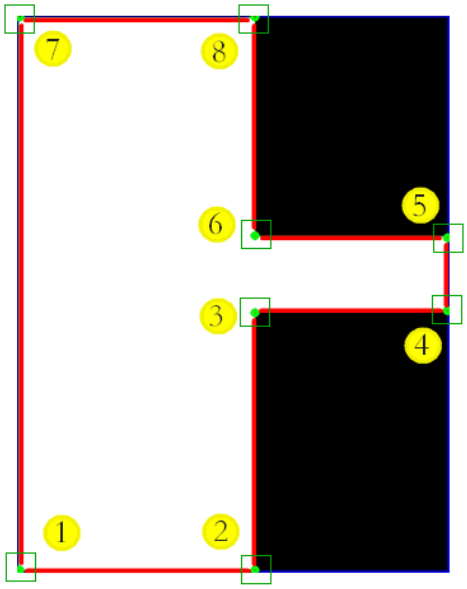


图 3-11 标记角点图和角点位置图

在本课题中，所采取的排序算法是图论中邻接矩阵的排序法。根据 points 数组，将每个角点位置编号，再通过 OpenCV 的直线检测，但是检测出的直线由于参数设置和像素问题，不一定刚好吻合在角点位置上，但一定在角点的附近，那

么在每个角点附近搜索一个正方形区域，如图 3-12（a）所示。最后利用图论的邻接矩阵 A 将点和线的关系表示出来，如图 3-12（b）所示。



(a)

$$A=\begin{array}{c} \\ 1\\2\\3\\4\\5\\6\\7\\8 \end{array}\begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} 0&1&0&0&0&0&1&0\\ 1&0&1&0&0&0&0&0\\ 0&1&0&1&0&0&0&0\\ 0&0&1&0&1&0&0&0\\ 0&0&0&1&0&1&0&0\\ 0&0&0&0&1&0&0&1\\ 1&0&0&0&0&0&0&1\\ 0&0&0&0&0&1&1&0 \end{pmatrix} \end{array}$$

(b)

图 3-12 角点和直线组成的无向图及其邻接矩阵

根据无向图的深度优先搜索（Depth-First-Search, DFS）算法的思路，能够高效地遍历所有点，由于还要通过顺时针的排序，就必须确定起点和第二个点。如图 3-12（a）上的起点即坐标原点，这个不难确定；第二个点需要从 2 号点和 7 号点中选出一点，这里利用两个向量的差来判断哪个才是第二个点，具体判别过程如图 3-13。确定第二个点之后，就能利用 DFS 进行遍历了。最后得到排序后的角点位置序号图 3-14（b）。

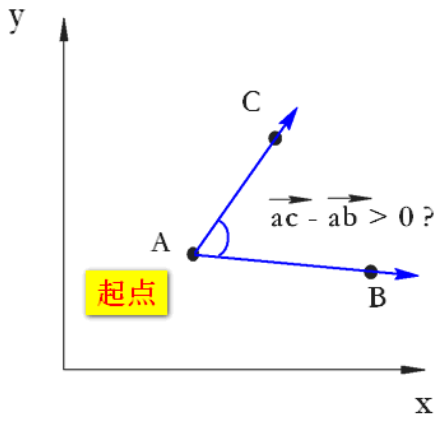


图 3-13 顺时针判别图

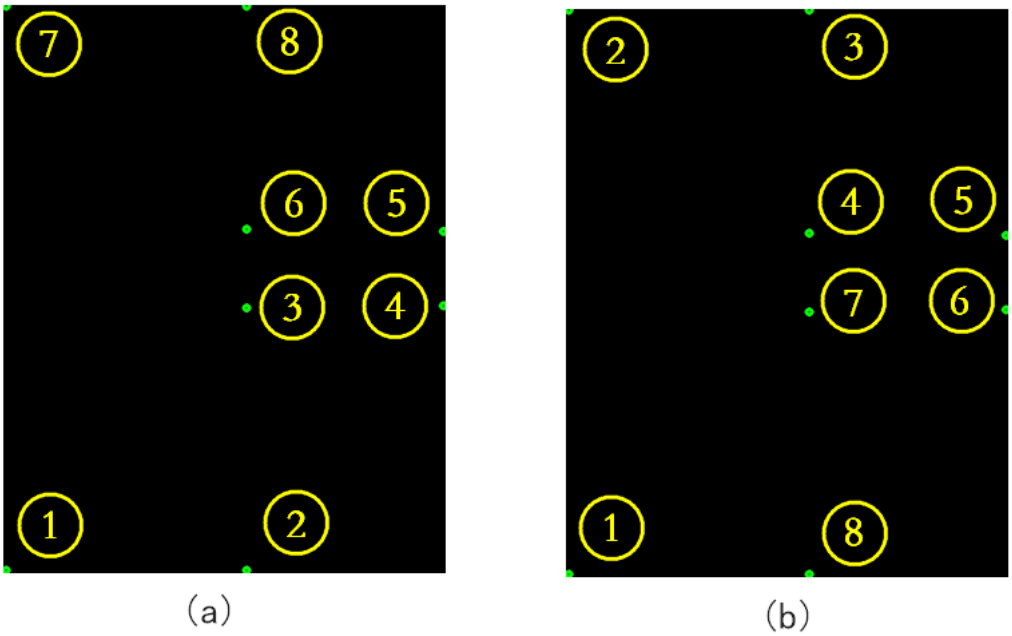


图 3-14 角点排序前后比较图

表 3-1 RCR 各阶段的角点阵

未排序的角点阵	排序后的角点阵	校正后的角点阵
[[4 1033]	[[4 1033]	[[0 0]
[444 1033]	[4 1]	[0 1032]
[444 553]	[444 1]	[440 1032]
[804 549]	[444 409]	[440 624]
[804 413]	[804 413]	[800 620]
[444 409]	[804 549]	[800 484]
[4 1]	[444 553]	[440 480]
[444 1]]	[444 1033]]	[440 0]]

3.5 将区域数据导入 UWB 定位系统

通过角点定位排序法，将能够得到从坐标原点出发，经过顺时针的邻接矩阵排序法，能够将所有角点给排序出来。经过角点定位排序法排序后角点需要都需要校正。

校正方法，由于在经过排序后的角点阵中，第一个角点就是原点，那么根据前面所说的常规直角坐标系（下面又称为 Qt 坐标系），原点位置是在坐标（0，0）点。而角点阵的第一个点为（4,1033），那么便需要从 OpenCV 坐标系根据一定的规则转为 Qt 坐标系。

规则如下：

(1) 由于 OpenCV 坐标系转换为 Qt 坐标系。

(2) 接下来只需要将原点从（4,1033）转换（0,0），而其他点又是相对于原点而进行转换，转换规则如下。最后得到表 3-1 中的校正后的角点阵。

```
    zero_col = points[0][0] #第一个角点的横轴
    zero_row = points[0][1] #第一个角点的纵轴

    for i in range(pnum): #pnum 为角点阵大小
        # 先将所有点都以第 0 点作为（0,0）的法则来进行调整
        points[i][0] -= zero_col
        points[i][1] = zero_row - points[i][1]

    return points #返回转换后的角点阵。
```

然后将角点阵 points 导入到 Qt 的 QTableWidget 控件（即 Qt 的表格控件）中，Qt 在绘图时候，只需要从 Qt 的 QTableWidget 控件导出数据即可。

3.6 本章小结

主要介绍了 OpenCV 在项目中的作用，通过 OpenCV 如何将工程图由 DWG 格式到 JPG 格式，再到如何提取住宅施工区域，然后到划分房间区域，最后到利用图论的邻接矩阵和深度优先搜索的概念对区域的数据进行提取的过程进行了详尽的说明。

第 4 章 工程图数据在 UWB 定位系统的应用

4.1 功能需求分析

在本课题中，铺砖机器人大致有两个定位系统，第一个是粗定位系统，另一个是精定位系统。铺砖机器人的定位大致分为两个阶段，第一个阶段是通过 UWB 室内定位，机器人从物料区获取砖块，然后根据 UWB 室内定位系统反馈需要铺设的位置，机器人便走到铺设区附近，由于在本系统中，由于三个基站（Anchor）存在的时序问题，UWB 不可避免的存在 10cm 左右误差，这便是粗定位过程。第二个阶段是通过红外线定位，机器人通过粗定位来到在铺设区附近，再经过红外线在地面中进行扫描，找出砖块铺设的确定位置，然后机器人便通过铺设砖块动作，这就完成了一次砖块的铺设。（本课题所讨论的砖块形状都是矩形的。）

由上面的过程来看，UWB 定位系统在本课题中起着尤其重要的作用，就目前而言，由于精度开发存在瓶颈，使得 UWB 在室内定位中，存在不可避免的误差，那在本系统么更需要把工程图的数据更精确地导入到 UWB 系统内。

4.2 本章业务流程图

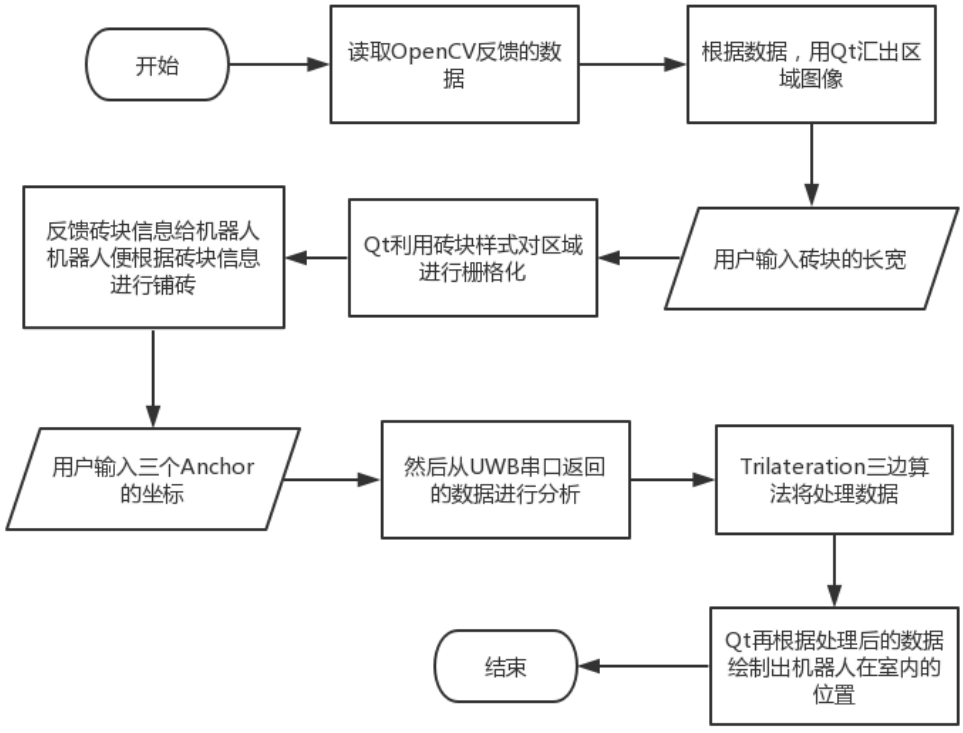


图 4-1 本章主要流程图

4.3 根据区域数据绘制区域图像

a) 预处理：

由前面 OpenCV 图像处理和分析得到校正后的角点阵，只是在图像中通过校正后的像素位置，而不是实际场景中的位置，那么由图像的像素位置转换为实际场景中的尺寸位置，就需要经过以下的预处理。

根据计算公式，需要通过计算 PPI 和 DP[7]才能将像素距离转换为实际距离，那么要将像素转换到实际尺寸就要先知道电脑的显示器配置才行。假如用户电脑的分辨率为：1920*1080，电脑尺寸为 15.6 英寸。计算公式如下：

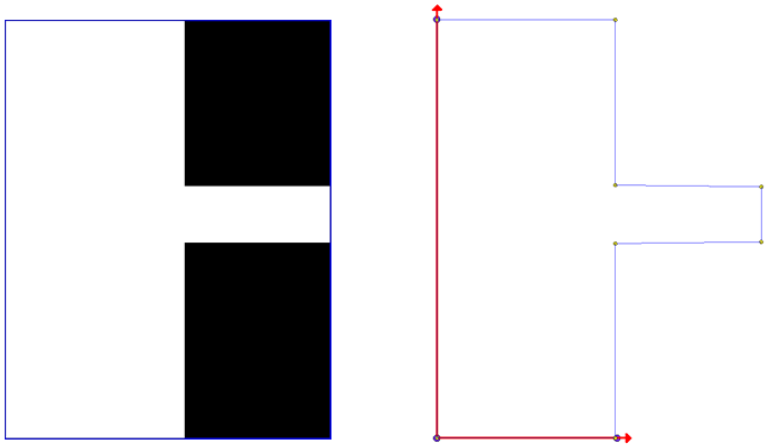
$$PPI = \sqrt{1920^2 + 1080^2}/15.6 = 141.21$$

$$DP = 15.6 * 25.4/\sqrt{1920^2 + 1080^2} = 0.1799 \text{ ,(其中 1 英寸 = 25.4 mm)}$$

经过像素到实际尺寸的转换后，还是会存在着些许误差，由于砖块铺设是从坐标系（从此处开始，所讲到的坐标系都是 Qt 的绘图坐标系）原点出发，沿着坐标轴向外一直铺设，那么最后存在的尺寸误差都当作一块砖来处理。

b) 绘制区域图像：

经过预处理后，得到区域的实际场景尺寸，便可利用 Qt 的 QGraphicsView 和 QGraphicsScene 控件进行绘制去区域的图像。图 4-2（a）是 RCR 轮廓图像，图 4-2（b）为 Qt 绘制图像。



(a)　　(b)

[7] DP: dot pitch，点距，即每点距指是指屏幕上相邻两个同色像素单元之间的距离，即两个红色（或绿、蓝）像素单元之间的距离。

图 4-2 OpenCV 轮廓图和 Qt 绘制图像比较图

4.4 进行砖的摆放规划

如上述所示，砖块的摆放规划就是从坐标系原点（0，0）出发，向四周扩张。

如图 4-3（a）所示，此时的砖块大小为长宽各为 500mm，且砖块之间的间隙为 5mm。

而图 4-3（b）中，此时的砖块大小为长 500mm，宽 400mm， 砖块之间的间隙为 5mm。

而图 4-3（c）中，此时的砖块大小为长 1250mm，宽 1250mm， 砖块之间的间隙为 5mm。

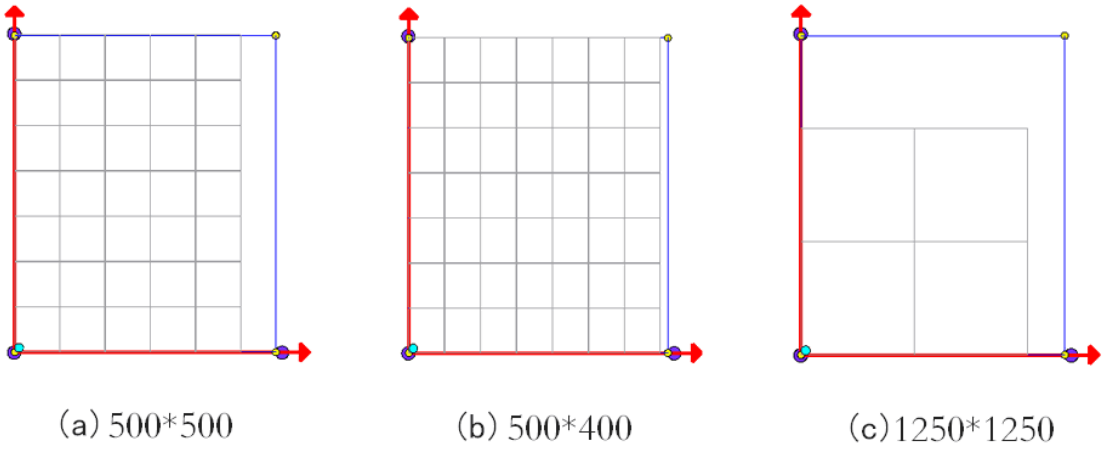


图 4-3 砖块样式图

4.5 将砖块信息传递给机器人

砖块信息用一个一维矩阵进行，信息内容如下：

[砖块编号，X 轴/mm ，Y 轴/mm，已完成？，取消]

索引名称	索引含义
砖块编号	是从原点出发，砖块铺设的排序是一个二级 for 循环，先往 X 轴方向铺设，然后再往 Y 轴方向铺设，如下图 4-4（a）所示。
X 轴/mm	是每块砖块左下角的位置，由于知道矩形砖块的长和宽，那么只需要知道其中一个点便能知道砖块所有顶点的位置，那么本课题规定的砖块固定点就是砖块的左下角点（即砖块与坐标系原点最靠近的点）。
Y 轴/mm	同上
已完成？	这个索引只有两个变量，其中一个变量是 0，表示该砖块的铺设没有完成；另一个变量是 1，表示该砖块的铺设已经完成。至于如何判断该砖块是否完成铺设，是团队中电控组和视觉组的内容，这里就没有深入探索，在本课题中，只需要完成上位机和机器人交接的功能。已完成的砖块都会在 Qt 中，利用 Brush（画刷）

	将这一块砖填充颜色。如下图 4-4（b）所示。
取消	这个索引属于 Qt 的一个控件 CheckBox（即可选框），CheckBox 如果被选上，则表示该砖块的铺设就被取消。加上这个索引的原因是在测试机器人铺设砖块的时候，存在地面有缺陷的地方，比如有石子，或地面有光线反射而影响机器人的红外线扫描仪等等各种原因，这些环境因素都可能影响机器人正常的铺砖流程，那么遇到这种情况，就可以取消这块砖的铺设，进而让机器人进入下一块砖的铺设流程。

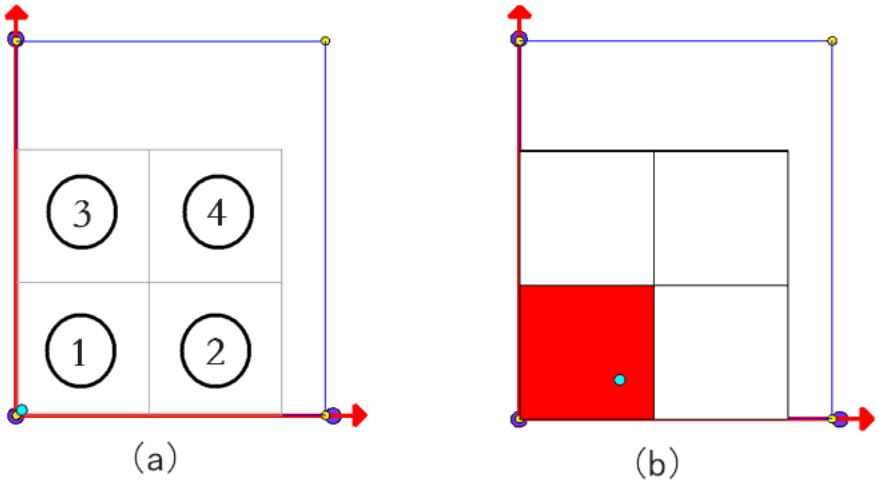


图 4-4 砖块编号图和砖块完成图

4.6 实时绘制机器人在区域的位置

根据前面第二章中 2.3.2 UWB 的定位技术原理介绍，在本课题中 UWB 实时定位标签（Tag）位置采用了 TOF 测距和 Trilateration 算法定位。

TOF 测距是测量基站（Anchor）与基站（Anchor）之间的距离，和标签（Tag）到基站（Anchor）之间的距离。本课题所采用的 UWB 开发套件中，就已经将 TOF 测距的功能集成到 STM32 芯片中，再通过外接串口设备 RS485 将基站 0（Anchor_0）收集到的数据经过 16 进制处理外界传输到电脑中。

根据 UWB 的开发者文档，基站 0（Anchor_0）通过串口返回的数据格式如下:

data = “mc 0f 000005a4 000004c8 00000436 000003f9 0958 c0 40424042 a0:0”

a) 数据格式解析：

MID	MASK	RANGE0	RANGE1	RANGE2	RANGE3	NRANGES	RSEQ	DEBUG	aT:A
ma	07	00000000	0000085c	00000659	000006b7	095b	26	00024bed	a0:0
mc	0f	00000663	000005a3	00000512	000004cb	095f	c1	00024c24	a0:0

b) TOF 数据主要的索引含义表：

索引	含义
MID	消息 ID，其中两类有效，分别为 mc, ma mc 代表 Tag-Anchor 距离，用于定位标签 ma 代表 Anchor-Anchor 距离，用于基站自动定位
MASK	表示 RANGE0, RANGE1, RANGE2, RANGE3 有哪几个消息是有效的； 例如: MASK=7 (0000 0111) 表示 RANGE0, RANGE1, RANGE2 都有效
RANGE0	如果 MID = mc, 表示 Tag 到 Anchor_0 的距离，单位：毫米
RANGE1	如果 MID = mc, 表示 Tag 到 Anchor_1 的距离 如果 MID = ma, 表示 Anchor_0 到 Anchor_1 的距离
RANGE2	如果 MID = mc, 表示 Tag 到 Anchor_2 的距离 如果 MID = ma, 表示 Anchor_0 到 Anchor_2 的距离
RANGE3	如果 MID = mc, 表示 Tag 到 Anchor_3 的距离 如果 MID = ma, 表示 Anchor_1 到 Anchor_2 的距离

由上表可以知道，当 MID = ‘ma’的时候，串口能够接收基站（Anchor）与基站（Anchor）之间的距离，但是在实际场景中，一般采用 3 个基站作为坐标系上面的三个顶点，由于 TOF 存在时序问题，导致 Anchor 与 Anchor 的距离测量存在误差，那么 Anchor 的坐标一般是人为在实际场景中所设定。

```
1.    for i in range(4):
2.        anchor[i].x = np.float(self.anchorTable.item(i,1).text())
3.        anchor[i].y = np.float(self.anchorTable.item(i,2).text())
4.        anchor[i].z = np.float(self.anchorTable.item(i,3).text())
```

在 Qt 中，新定义一个 QTableWidget 类型变量 anchorTable，其中 anchorTable 是一个 4 行 3 列的表格，每一行都是一个基站的三维坐标（X,Y,Z），其中 Z 代表的是基站高度，默认设置为 1.5 米高，这要确保每一个基站的高度是一致的，否则会增大 TOF 处理的时序问题，误差便会因此变大。

那么将串口接收的数据分别分割存放在一个数组 data 中，当 MID = ‘mc’ 的时候才是真正需要的标签（Tag）-基站（Y）的距离数据。其中 Y 表示{0,1,2}。

```
1.    range_0 = int(data[2], 16) # 标签 x 到基站 0 的距离
2.    range_1 = int(data[3], 16) # 标签 x 到基站 1 的距离
3.    range_2 = int(data[4], 16) # 标签 x 到基站 2 的距离
```

最后通过将 anchor 数组和 range_0, range_1, range_2 一并当作参数传到 Trilateration()函数当中，最后在 Trilateration()函数中通过三边定位算法得出标签(X)在基站坐标系中的实际位置，如图 4-5 所示。

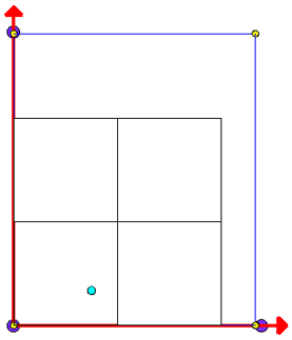


图 4-5 机器人实时位置图

4.7 Python 和 C/C++混合编程

由于 Trilateration 三边定位算法的实时计算量大，考虑到系统的高效性和鲁棒性，便将 Trilateration 算法所实现的功能集成到 Trilateration.cpp 文件中，最后通过混合编程将 Trilateration 定位功能融入到 Python 代码中去。下面就分别简介 Python 和 C++的特性，再分析如何实现在 Python 中调用 C++。

a) Python 语言简介：

Python 是一个高层次的结合了解释性、编译性、互动性和面向对象的脚本语言。经过多年的发展和如今人工智能和大数据的兴起，Python 已经成为世界范围内应用最广泛的跨平台语言之一。

Python 的几个常见特性[11]：

(1) 免费开源。Python 的源代码是公开的，即可以对其进行拷贝、阅读和修改，也能集成另一个新的自由软件当中。

(2) 高级语言。用 Python 语言编写程序时候，不需要考虑计算机底层是如何实现的。

(3) 可移植性。由于 Python 的开源特性，Python 已经被移植到许多平台上，例如 Linux、Mac、Windows、Android、Symbian 等。

(4) 面向对象。Python 提供类、类的继承、类的私有和公有属性、异常处理等完善的对面向对象方法的支持。

(5) 可嵌入性。也就是现在所介绍的混合编程特性，能够嵌入到 C/C++程序当中。

(6) 可扩展性。混合编程的另一类，由于某些算法不愿公开，那么就将不公开的代码用 C/C++编程，然后 Python 调用 C/C++。

(7) 执行速度慢。Python 属于动态语言[8]，导致在大型项目中运行效率低。

b) C++语言简介：

C++是从 C 语言发展演变而来的，C 语言具有很多优点，例如：语言简洁灵活、运算符和数据结构丰富、具有结构化控制语句、程序执行效率高，而且同时具有高级语言与汇编语言的特点。C++是在 C 语言基础上为支持面向对象的程序设计而研制的、一个通用的目的的程序设计语言，它是在 1980 年由 AT&T 贝尔实验室的 Bjarne Stroustrup 博士创建的。

C++语言的特点如下：

尽量兼容 C。保持了 C 的简洁、高效和接近汇编语言等特点。

(1) 面向对象。

(2) 对于库（Library）的编写，有着优良的支持。

(3) 支持指针。

(4) 源代码加密。

(5) 高效。C++属于静态语言[9]。对于图像处理[10]，数据处理的大型项目中都有着高性能的支持。

(6) 社区用户多，开发文档齐全。

c) C/C++混合编程的工具：SWIG

SWIG（Simplified Wrapper and Interface Generator），SWIG 是个帮助使用 C/C++编写的软件，能与其它各种高级编程语言进行嵌入联接的开发工具。

SWIG 能应用于各种不同类型的语言包括常用脚本解释性语言例如 JavaScript, Perl, PHP, Python, Ruby。支持的语言还包括编译型语言，比如 C#，Java，Android，Go，R 语言。SWIG 普遍应用于创建高级语言解析或汇编程序环境，用户接口，作为一种用来测试 C/C++或进行原型设计的工具。SWIG 还能够导出 XML 或 Lisps-expressions 格式的解析树。SWIG 还有一个特点就是免费开源。

[8] 动态语言，即解释型语言：边执行边解释，执行一句转化一句，不需要预先编译

[9] 静态语言，即编译性语言，需要编译:在代码执行之前，将代码转成计算机（CPU）可识别可执行的指令

[10] OpenCV 的源代码就是 C/C++编写的

d) 混合编程的概念：

正如上面所说的，SWIG 主要的目的就是简化 C/C++与其他语言混合编程的任务。虽然 C/C++存在大量的优点，但是也存在以下的缺点：

(1) 编写用户界面相当困难。

(2) 编译过程非常耗时。

(3) 如果没有重新编译的前提下，对于系统的重新配置和对程序的定制难度大

(4) 模块化复杂

(5) 安全问题，例如缓冲器栈溢出。

为了突破这些限制，许多开发者总结出：利用不同的编程语言编写不同的任务会容易得多。例如：编写用户界面时候使用 Python 或者 Tcl 等脚本语言；编写分布式系统时候，使用 Java 能简化许多任务。面对这些情况最关键的一点就是需要认识到不同的编程语言有着不同的优点和缺点，而且任何程序都不可能是完美无缺的。因此，通过使用混合编程，开发者就能综合各种语言的优势，使之简化软件开发在特定的方面的流程。

从 C/C++的立场来说，许多开发者使用 SWIG 的原因是他们想要打破 C 传统的编程庞大冗余的特性，这种特性会导致类似以下的后果：

(1) 只有一些函数和变量是有效的。

(2) 只能通过主方法来启动整个项目。

(3) 用户接口被黑客监听

为了避免进入这些回路，将 C/C++融合到一个更高级的编程语言当中，能够减少代码量，用上更多的模块化设计，更加灵活，且使开发者更高效。

SWIG 试图将 C/C++的问题尽可能减少。SWIG 能够使开发者更加集中在 C 的编程和更高级语言的编程上，而不需要理解理解这两种语言之间繁琐和乏味的转换技术上。同时，SWIG 能够识别出所有不同的应用。因此，SWIG 提供了一个广泛多样的定制化特点，这个特点让开发者几乎可以改变任何语言方面的捆绑。这就是为什么 SWIG 被广泛使用的原因。

e) 使用 SWIG 进行 Python 和 C++的混合编程：

(1) example.h：头文件

```
1.    #include<iostream>
```

```
2.  using namespace std;
3.
4.  int fact(int n);
```

(2) example.cpp：源文件（主要实现的功能是阶乘）

```
1.  #include "example.h"
2.
3.  int fact(int n){
4.      if (n < 0){
5.          return 0;
6.      }
7.      if (n == 0){
8.          return 1;
9.      }
10.     else{
11.         return n * fact(n-1);
12.     }
13. }
```

(3) example.i ：接口文件 (interface file 包含了 ANSI C 的函数原型和变量声明；%module 指令定义了将被 SWIG 创建的模块的名称；%{ %}这一块为了生成混编代码而提供了插入额外代码的定位，例如 C++的头文件、C/C++额外声明)

```
1.  %module example
2.
3.  %{
4.  #include "example.h"
5.  %}
6.
7.  %include "example.h"
```

通过 SWIG 的命令：`$ swig -c++ -python example.i` 接口文件，创建 example.py 和 example_wrap.cxx 文件，其中 example.py 就是 C++功能的 Python 实现文件。

(4) setup.py 文件。setup 操作就是利用 distutils 这个库来创建你的包，其中需要的是数据就是在 setup.py 填写的文件。

```
1.  from distutils.core import setup, Extension  # 加入混合编程需要的
库
2.
```

```
    example_module = Extension("_example",sources=["example.cpp","example_wrap.cxx",],) # 源文件模块

    setup(name='example',
        version='0.1', # 版本号
        author='Jun', # 作者名字
        description="""Simple swig example from docs""", # 描述
        ext_modules=[example_module],
        py_modules=['example'], # Python 模块
    )
```

最后，利用命令：`$ python setup.py build_ext --inplace` 将创建 C++的动态链接库，最后 Python 便能够调用所创建的包。

测试 example 模块是否存在：`$ python -c 'import example'`

4.8 本章小结

本章介绍了如何在项目中将 OpenCV 提取出来的房间区域的数据运用到 UWB 定位系统当中，然后在 UWB 系统中利用 Qt 根据区域数据将室内占地环境绘制出来，接着根据提供的砖块大小对砖块进行一定规则的栅格化，再将砖块信息通过网络通讯反馈给机器人，最后利用 Python 和 C/C++的混合编程实现 Trilateration 三边定位算法。

第 5 章 实验结果

5.1 系统总流程图

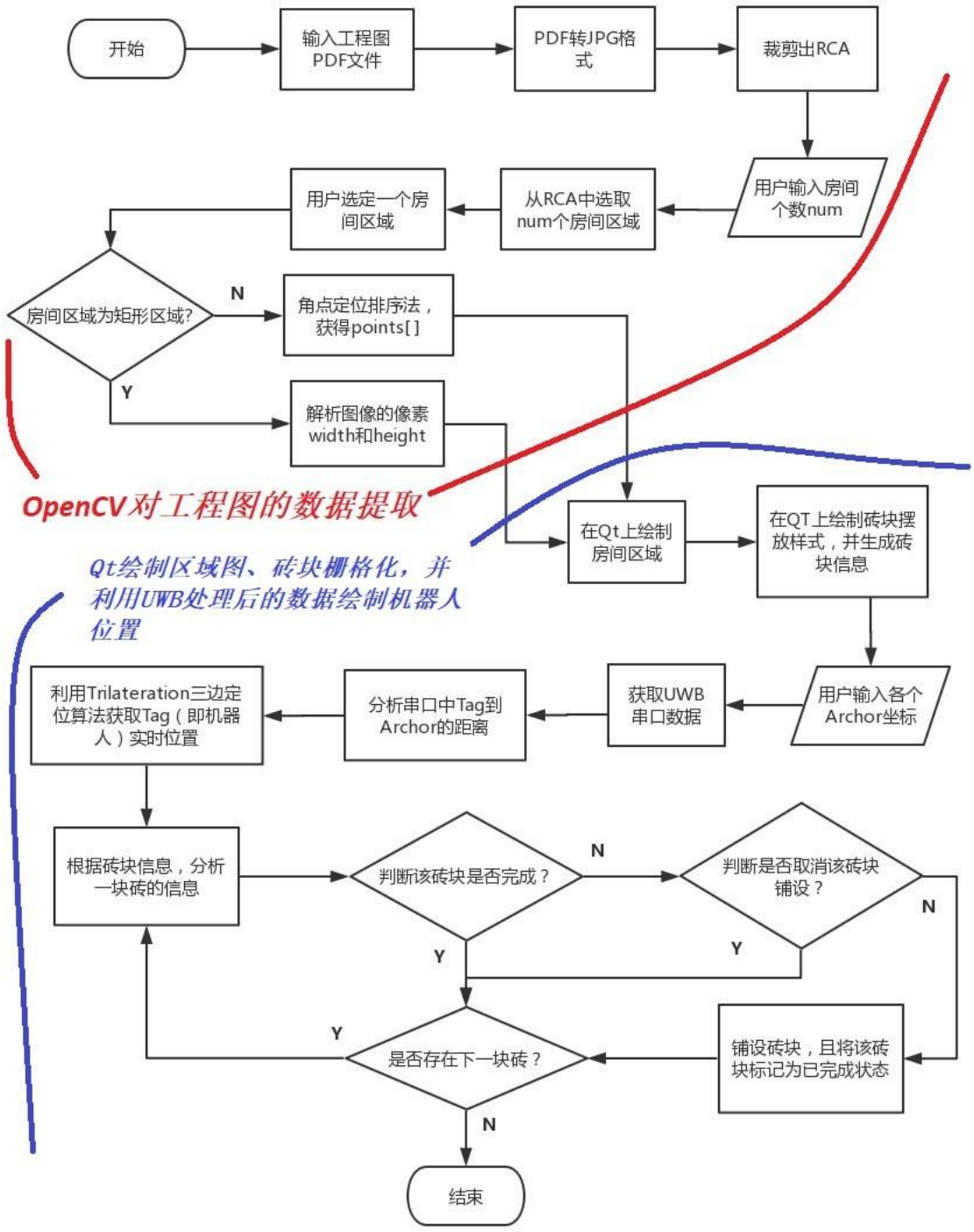


图 5-1 系统总流程图

5.2 系统细节分析图解

图 5-2 系统的前提设置

如图 5-2 所示，系统在运行之前就要手动低进行上面的设置。

1) 串口设置。由于系统使用 UWB 模块进行 Anchor 和 Tag 之间的 TOF 测距，测出来的数据再通过串口的反馈到计算机上，那么系统在启动以后，就需要打开串口并处理串口返回的数据才能计算出机器人的实时位置。

2) dpi 设置。dpi 设置就是像素和距离之间的转换途径。因为图像中的两点之间都是以像素为单位进行测距的，但实际场景中，需要将单位从像素转为厘米或者毫米才行。

3) 砖块大小及其间隙设置。用户需要输入矩形砖块的长宽和每块砖之间的间隙。

4) 房间个数的输入。用户输入整型数据 num，OpenCV 检测出区域的所有轮廓之后，会选出面积最大的 num 个轮廓

5) 图纸比例的输入。根据 DWG 工程图纸中尺寸比例，对应到图片中的尺寸比例。

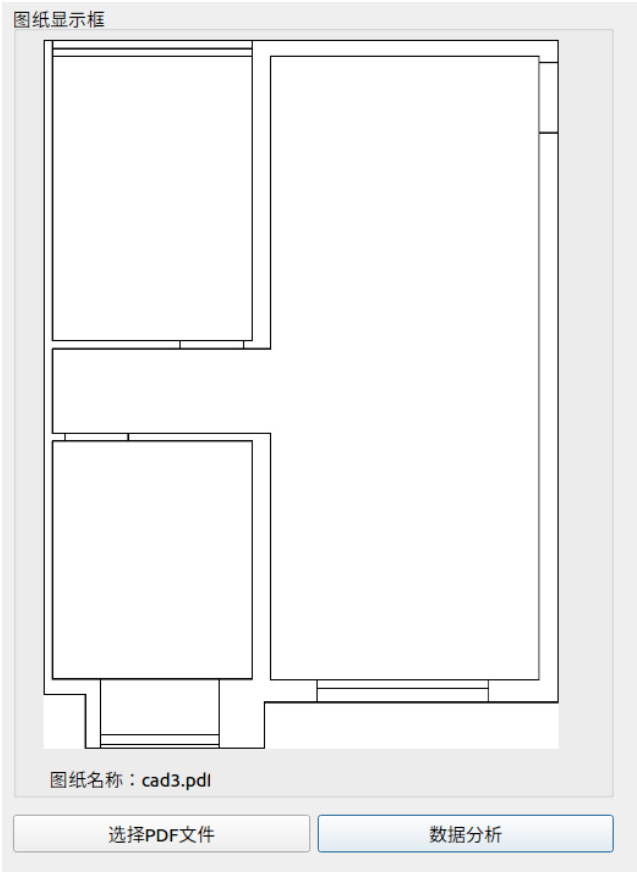


图 5-2 图纸显示框

如用户点击“选择 PDF 文件”按钮，便会弹出文件选择窗体，选择 PDF 工程图纸，程序便会将 PDF 文件转换为 JPG 文件，再对 JPG 图像进行裁剪，便显示在图纸显示框中。

用户再点击“数据分析”按钮后，后台便会处理这一张工程图，根据用户在前面输入的 num 整型数据，提取出这张工程图中最大的 num 个房间区域。

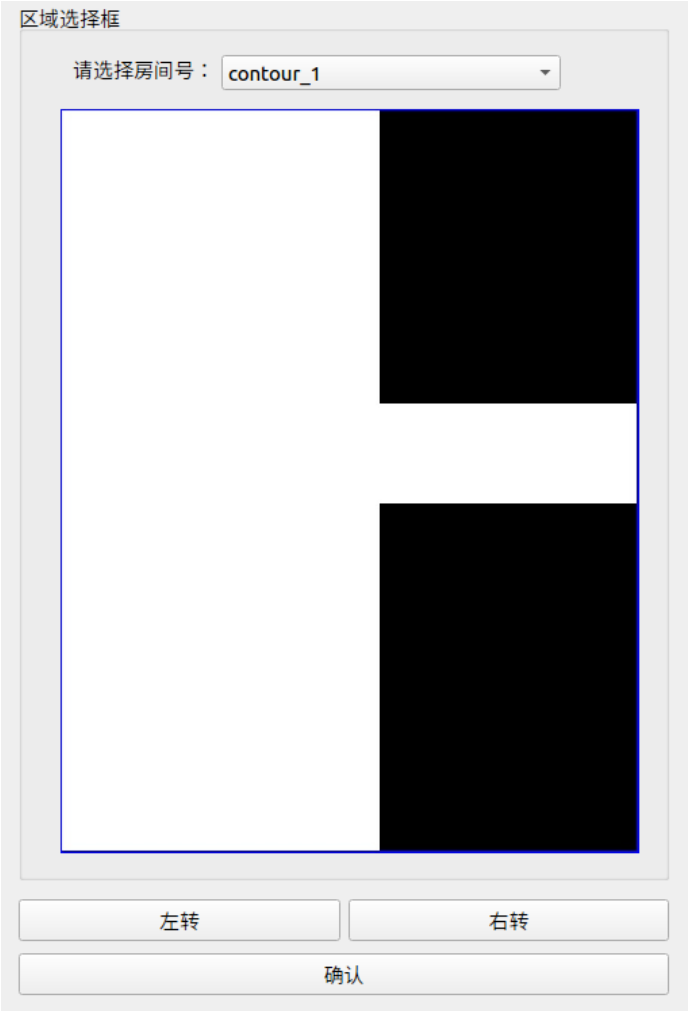


图 5-3 区域选择框

用户在“请选择房间号”的下拉列表选择出后台分析出来的 num 个房间中的其中一个。例如图 5-3 中，选出了一个矩形组合区域。

“左转”和“右转”按钮：是将区域进行对应的旋转，以便将区域的左下角点作为 Qt 的坐标系原点。

“确认”按钮：用户点击此按钮后，程序便对该区域进行数据提取，例如对矩形组合区域，通过角点定位排序法，将角点提取出来并排序，处理完成后，将数据返回在角点列表中。

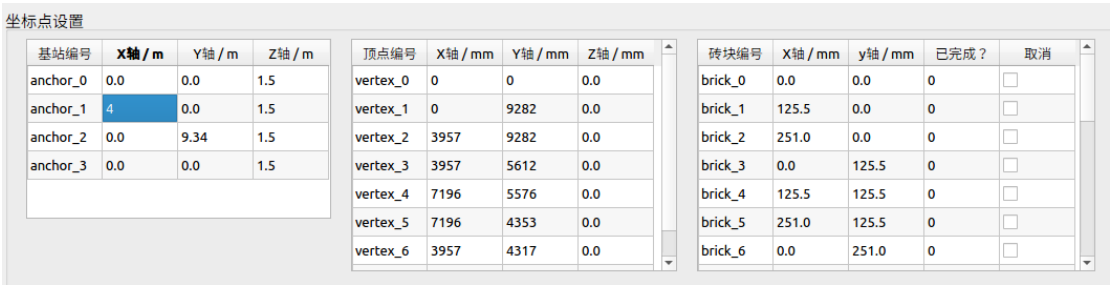


图 5-4 坐标点设置

如图 5-4 所示，坐标点设置框分别有基站坐标、角点坐标、砖块坐标。

1) 基站坐标：用户设置 Anchor 的坐标，用于 UWB 系统中的 TOF 测距，和 Trilateration 定位算法中。

2) 角点坐标：通过 OpenCV 的数据提取后，将会将角点的数据返回在此表格。

3) 砖块坐标：Qt 根据用户提供的砖块样式将区域进行栅格化以后，就会反馈数据此表格中。同时系统还会将砖块信息反馈给机器人，机器人根据砖块的信息，做出对应的抉择。

图 5-5 UWB 测距定位数据图

1) TOF 测距：从 UWB 串口返回的 Tag 和 Anchor (0,1,2)之间的 TOF 测距结果。

TOF0：是 Tag 与 Anchor_0 之间的距离。

TOF1：是 Tag 与 Anchor_1 之间的距离。

TOF2：是 Tag 与 Anchor_2 之间的距离。

2) 机器人坐标：利用 Trilateration 三边定位算法计算出机器人的实时位置。其中机器人的初始位置为（0，0，1.5m）其中 1.5m 是机器人高度的意思。

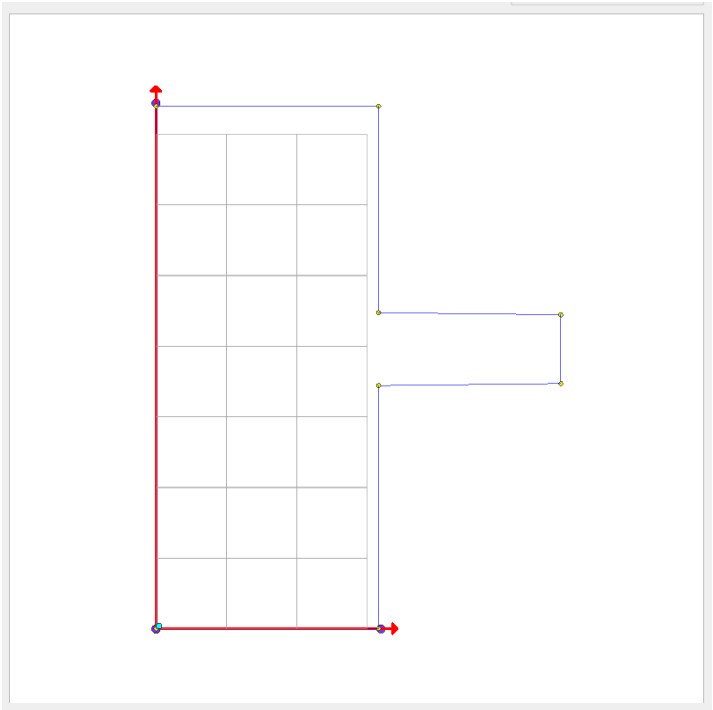


图 5-6 Qt 绘制房间区域及其砖块栅格化

图 5-6 中就是 Qt 根据角点数据和砖块数据，在控件 QGraphicsView 中绘制对应的图像，其中绘制的图像包括房间区域和室内的砖块栅格化。

5.3 矩形区域分析图

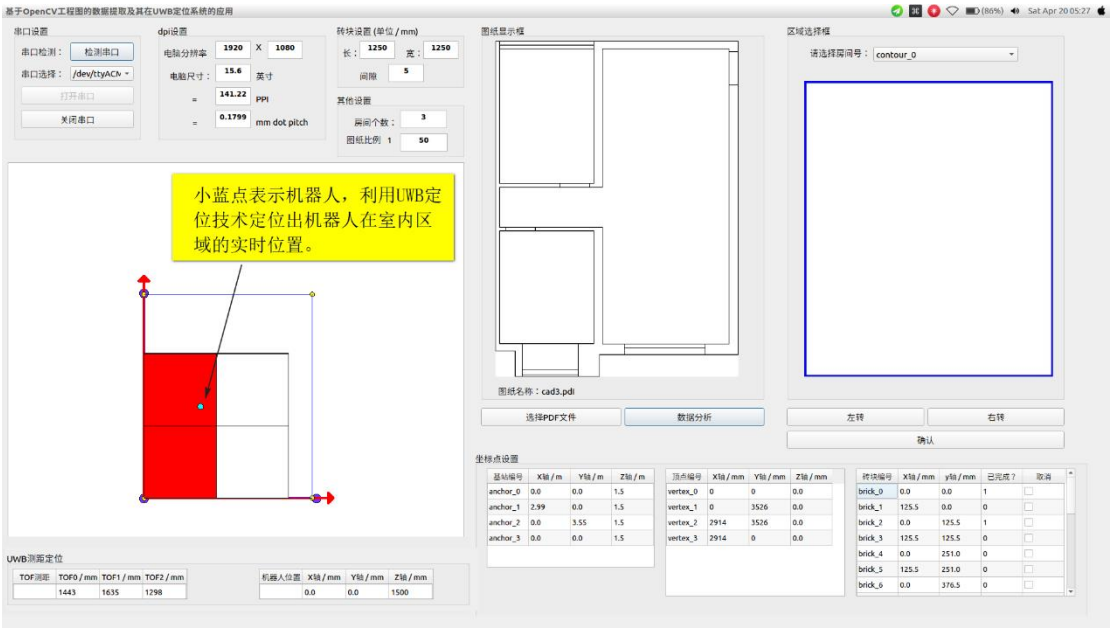


图 5-7 系统运作图

图 5-7 就是系统整体的运作图，其中选择的房间区域为矩形区域。其中 QGraphicsView 中的小蓝点代表着机器人，其中是利用 UWB 定位技术定位出机器人在室内的实时位置。

5.4 矩形组合区域分析图

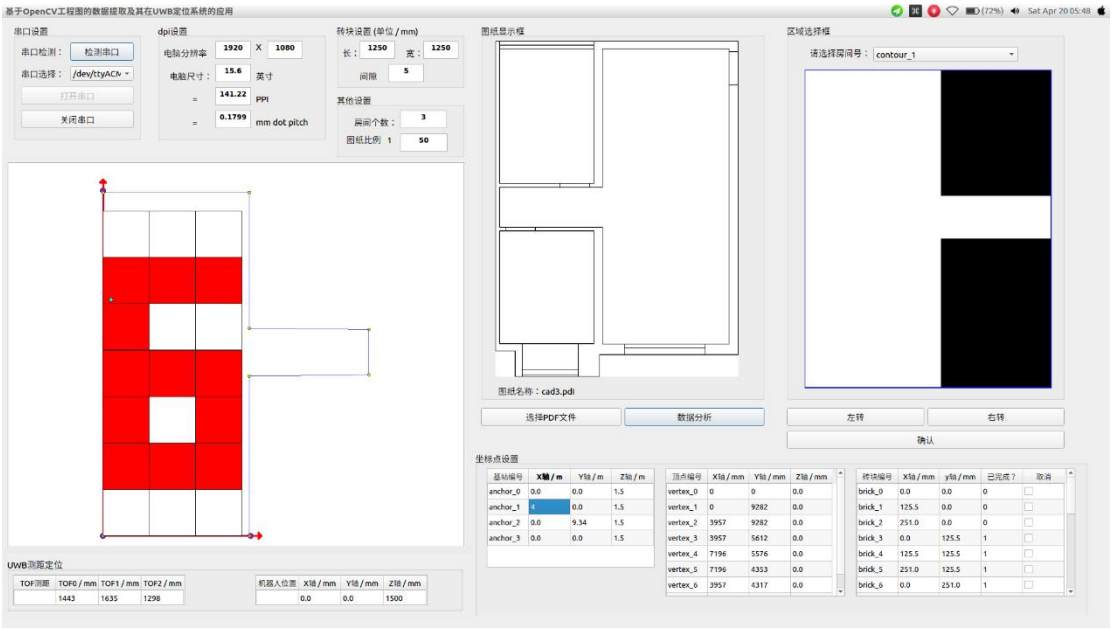


图 5-8 系统运作图

图 5-8 就是系统整体的运作图，其中选择的房间区域为矩形组合区域。其中在 QGraphicsView 控件中阿拉伯数字“6”，就是机器人根据砖块信息和设定的流程在特定的砖块进行铺设的结果。

第 6 章 结论与展望

本文首先分析了机器人在建筑行业的需求，然后将建筑行业使用的工程图转换为通用的图片格式文件，再利用 OpenCV 对工程图进行数据提取，本文对 OpenCV 在项目中运用的知识进行了大概的介绍，本文根据图论的邻接矩阵和深度优先搜索算法提出了一种自定义的角点定位排序法。再到通过 Qt 的绘图功能进而实现 UWB 定位的室内平面区域，最后使用混合编程实现了机器人的实时定位位置。

尽管本文实验实现了通用的矩形房间区域的数据提取和矩形砖块的摆放样式，但面对非矩形或者非矩形组合的房间区域，还没有研究对应的数据解析算法。在原始的建筑施工图中，可能存在一些更加复杂的图案，和更加复杂的结构和数字，那么本文还要进一步探讨如何从这些复杂的建筑工程图当中，更加精准地提取住宅施工区域，并且消去原始工程图当中的砖块栅格化。本文接下来还要对更多的砖块摆放样式进行实验。

参考文献

[1] GaryBradski, AdrianKaehler. 学习 OpenCV:中文版[M]. 北京：清华大学出版社, 2009.

[2] 张铮, 王艳平, 薛桂香. 数字图像处理与机器视觉：Visual C++与 Matlab 实现[M]. 北京：人民邮电出版社, 2010.

[3] Gonzalez R, Woods R, Eddins S, et al. 数字图像处理的 MATLAB 实现[M]. 北京：清华大学出版社, 2013.

[4] 洪改艳, 芮廷先, 俞伟广, et al. Harris 角点检测的优化算法[J]. 计算机系统应用, 2017(04):169-172.

[5] 董立红, 彭业勋, 符立梅. 基于 Sobel 边缘检测的圆周 Harris 角点检测算法[J]. 西安科技大学学报, 2019, 39(02):374-380.

[6] 韩松奇, 于微波, 杨宏涛, et al. 改进 Harris 角点检测算法[J]. 长春工业大学学报, 2018, 39(05):60-64.

[7] 高思琪, 孙建平. UWB 定位技术的应用研究[J]. 仪器仪表用户, 2019(03):77-82.

[8] 刘欢笑. 基于超宽带和三边量测法的室内定位系统设计[J]. 镇江高专学报, 2018, v.31；No.113(02):47-51.

[9] 王波. 浅谈 UWB 定位技术[J]. 中国新技术新产品, 2011(23):47-47.

[10] 薄成, 张西军. 基于 AutoCAD .NET API 的 DWG 文件坐标变换的研究及实现[J]. 测绘与空间地理信息, 2019, 42(03):198-201.

[11] 陶诚, 陆从珍. 基于 C++和 Python 混合编程的 WORD 文档操作方法[J]. 信息化研究, 2014(05):58-63.

致谢

感谢家人，感谢朋友，感谢导师，感谢老师，感谢自己。