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：** | **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：** 2019 **年** 2 **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、DWG2JPG、 PyQt5、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; DWG2JPG; PyQt5; Trilateration Location; Mixed Program

**目 录**

[第1章 概述 1](#_Toc7041712)

[1.1研究背景与现状 1](#_Toc7041713)

[1.2 同类机器人在国内外的发展现状 2](#_Toc7041714)

[1.3本文主要工作 2](#_Toc7041715)

[1.4 章节安排 2](#_Toc7041716)

[第2章 相关技术和开发环境 4](#_Toc7041717)

[2.1 建筑工程图的认识 4](#_Toc7041718)

[2.2 OpenCV的基础知识 4](#_Toc7041719)

[2.2.1 OpenCV介绍 4](#_Toc7041720)

[2.2.2 灰度处理 5](#_Toc7041721)

[2.2.3 Canny边缘检测法 7](#_Toc7041722)

[2.2.4 直线检测 12](#_Toc7041723)

[2.2.5 轮廓提取 15](#_Toc7041724)

[2.2.6 角点检测 17](#_Toc7041725)

[2.3 基于超宽带（UltraWideband无载波通信技术）室内定位的认识 18](#_Toc7041726)

[2.3.1 UWB的介绍 19](#_Toc7041727)

[2.3.2 UWB的定位技术原理 19](#_Toc7041728)

[2.3.3 TOF测距 19](#_Toc7041729)

[2.3.4 Trilateration 三边定位算法 20](#_Toc7041730)

[2.4 本章小结 22](#_Toc7041731)

[第3章 基于OpenCV的工程图数据的数据提取 23](#_Toc7041732)

[3.1 功能需求分析 23](#_Toc7041733)

[3.2 本章业务流程图 23](#_Toc7041734)

[3.3 工程图进行房间区域划分 26](#_Toc7041735)

[3.4 提取区域数据 27](#_Toc7041736)

[3.5 将区域数据导入UWB定位系统 33](#_Toc7041737)

[3.6 本章小结 33](#_Toc7041738)

[第4章 工程图数据在UWB定位系统的应用 34](#_Toc7041739)

[4.1 功能需求分析 34](#_Toc7041740)

[4.2 本章业务流程图 34](#_Toc7041741)

[4.3 根据区域数据绘制区域图像 35](#_Toc7041742)

[4.4 进行砖的摆放规划 36](#_Toc7041743)

[4.5 将砖块信息传递给机器人 36](#_Toc7041744)

[4.6 实时绘制机器人在区域的位置 37](#_Toc7041745)

[4.7 Python和C/C++混合编程 39](#_Toc7041746)

[4.8 本章小结 43](#_Toc7041747)

[第5章 实验结果 44](#_Toc7041748)

[5.1 系统总流程图 44](#_Toc7041749)

[5.2 系统细节分析图解 45](#_Toc7041750)

[5.3 矩形区域分析图 49](#_Toc7041751)

[5.4 矩形组合区域分析图 49](#_Toc7041752)

[第6章 结论与展望 51](#_Toc7041753)

[参考文献 52](#_Toc7041754)

[致谢 53](#_Toc7041755)

[附录 54](#_Toc7041756)

# 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，自从建筑行业的生产力相对于其他制造业有着明显低效，建筑行业早就孕育了建筑机器人的发展进程。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。

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]](#footnote-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。

## 1.2 同类机器人在国内外的发展现状

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。

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。

目前，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。

## 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处理，然后将提取出来的工程图数据利用Qt[[2]](#footnote-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

本论文共分为七章，内容如下：

第1章 概述，主要介绍了本论文的研究背景、意义，同类机器人在国内外的发展现状，主要工作及论文的组织结构。

第2章 相关技术及背景，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。

# 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。

工程图纸分类分为三种：建筑施工图（又称建施），主要表示房屋的总平面图、剖面图、立面图等；结构施工图（又称结施），主要表示房屋承重结构的布置、构件类型、大小、数量及做法等。它包括结构布置图和构件详图；设施施工图（又称设施），主要表示各种线路、设备和管道的布置、走向以及安装施工要求等。

而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（CON-TINGENY 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）诞生于Intel研究中心，其目的是开发开一个普遍可用的计算机视觉库。在Inter的性能库团队的帮助下，OpenCV实现了一些关于计算机视觉的核心代码以及算法，并在Inter俄罗斯的库团队的帮助下得到优化。

OpenCV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。无论是科研使用还是商业使用，OpenCV都是开放源代码且免费的，OpenCV的代码可用于或者嵌入（整体或部分）其他的应用程序中。

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]。

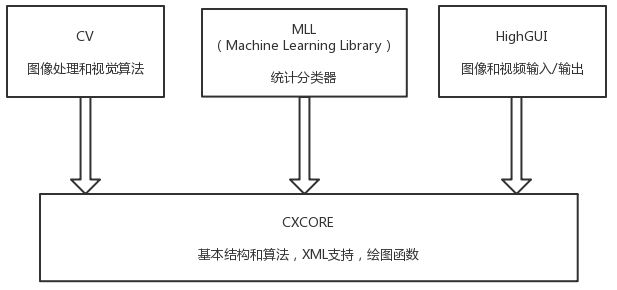


图2-1 OpenCV主要的四个模块及其关系图

### 2.2.2 灰度处理

#### **一、灰度的概念：**

一个大小为M \* N的数字图像是由M行N列的有限元素组成的，每个元素（又称为像素）都有特定的位置和幅值，代表了所在行列位置上的图像物理信息，如灰度和RBG等。

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。这类图像通常显示为从最暗黑色到最亮的白色的灰度，每种灰度（颜色深度）称为一个灰度级，通常用L表示。在灰度图像中，像素可以取0～L-1之间的整数值，根据保存灰度数值所使用的数据类型不同，可能有256种取值或者2^k种取值，当k=1时即退化为二指图像。在通常情况下，图像的灰度级范围从0到255，白色为255，黑色为0，故黑白图片又称为灰度图像。灰度级越大表示越亮。

#### **二、图像灰度级分辨率：**

在数字图像处理中，灰度级分辨率又称色阶，是指图像中可分辨的灰度级数目L，它与存储灰度级别所使用的数据类型有关。由于灰度级度量的是投射到传感器上光辐射值的强度，所以灰度级分辨率也叫辐射计量分辨率[2]。

随着图像的灰度级分辨率逐渐降低，图像中包含的颜色数目变少，从而在颜色的角度造成图像信息受损，同样使图像细节表达受到了一定的影响。

#### **三、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。

灰度直方图是最基本的图像分析工具。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。主要应用于图像分割和图像灰度变换等处理过程中。

利用直方图辅助实现的各种灰度变换（灰度变换，即点运算[[3]](#footnote-3)），包括线性灰度变换、分段线性灰度变换、非线性灰度变换等。

#### **四、阈值处理：**

根据用户规定的阈值（threshold），一张图像每个像素的灰度值放在一个数组中，然后根据数值中的每个元素的值低于还是高于阈值而进行一些处理。例如阈值的二值化处理，图像中每个像素点的灰度值如果大于threshold，则这个值置为255；如果小于threshold，则将这个值置为0。

#### **五、直方图均衡化：**

把原始图像的直方图变换为均匀分布的形式，从而增加图像灰度的动态范围，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[[4]](#footnote-4)最大，图像所包含的信息量最大。

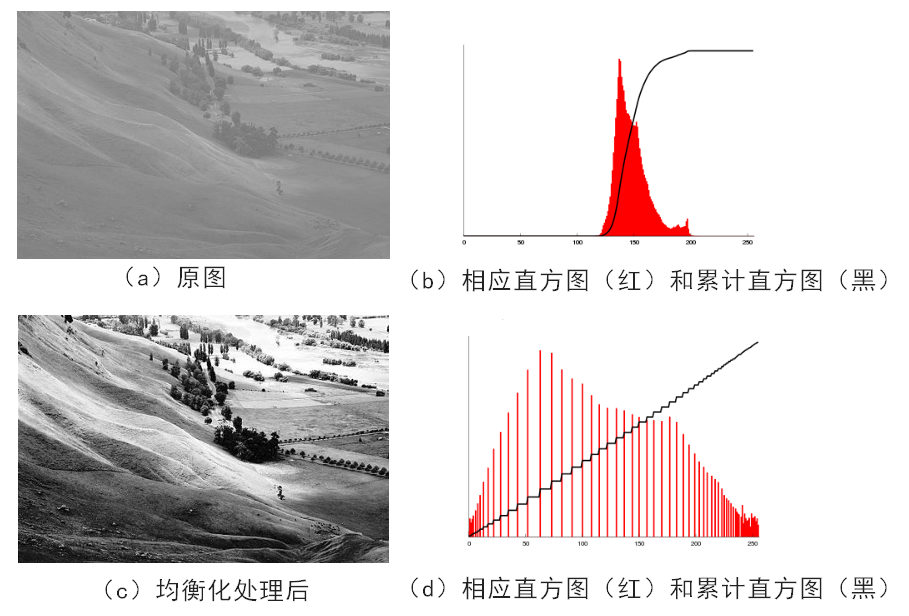


图2-2 直方图均衡化前后比较

### 2.2.3 Canny边缘检测法

#### **一、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。

图像的边缘有方向和幅度两个属性，沿边缘方向像素像素变化平缓，垂直于边缘方向像素变化剧烈。边缘上的这种变化可以用微分算子[[5]](#footnote-5)检测出来，通常用一阶或二阶导数来检测边缘。

#### **二、边缘检测的基本步骤：**

边缘检测的步骤如图2-4所示：

平滑滤波：平滑处理，又称为减少噪声。由于梯度计算易受噪声的影响，因此第一步是用滤波[[6]](#footnote-6)去除噪声。但是，降噪的平滑能力越强的同时，边界强度的损失就越大。

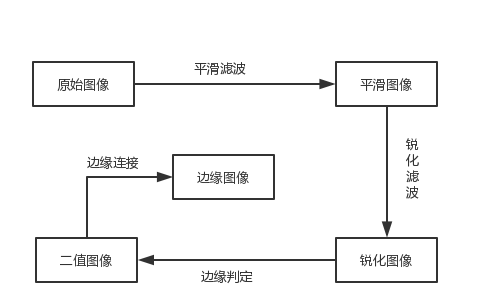


图2-4 边缘检测步骤

锐化滤波：为了检测边界，必须确定某个点领域中灰度的变化。锐化操作加强了存在有意义的灰度局部变化位置的像素点。

边缘判定：在图像中存在许多梯度不为零的点，但是对于特定应用，不是所有点都有意义。这就需要根据具体情况选择和去除处理这些点，具体的方法包括二指化处理和过零检测等。而本课题采取的方法是二指化处理。

边缘连接：将间断的边缘连接成为有意义的完整边缘，同时去除假边缘。主要方法是Hough变换，常用于直线检测中。

#### **三、基于微分方法的边缘检测算子：**

一阶导数可以用于检测图像中的一个点是否在斜坡上。二阶导数的符号可以用于判断一个边缘像素在边缘亮的一边还是暗的一边。

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2-5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。

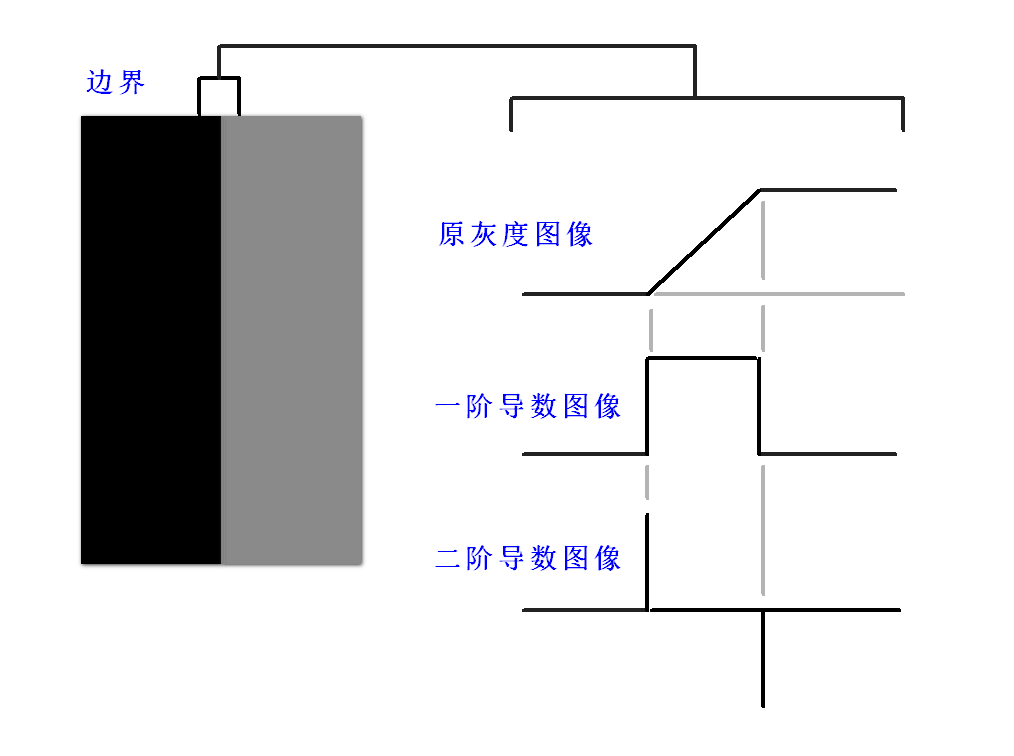


图2-5 基于微分方法的边缘检测图解

#### **四、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，他们只有在图像不含噪声或者首先通过平滑去除噪声的前提下才能正常应用。

在图像边缘检测中，抑制噪声和边缘精确定位是无法同时满足的，一些边缘检测算法通过平滑滤波去除噪声的同时，也增加了边缘定位的不确定性；而提高边缘检测算子对边缘敏感性的同时，也提高了对噪声的敏感度。就上面所介绍的基于微分方法的边缘检测算法而言，Roberts和LoG算子定位精度高，但受噪声影响大；而Sobel算子和Prewitt算子去噪能力较强，具有平滑作用，能滤除一些噪声，去除部分伪边缘，但同时也平滑了真正的边缘，这也是其定位精度不高的原因。Canny算子力求在抗噪声干扰和边缘精确定位之间寻求最佳折中方案。

Canny对边缘检测质量进行分析，有以下3个准则。

信噪比准则：尽可能多地检测图像的真实边缘，和尽可能少地检测虚假边缘。那就是使信噪比SNR（Signal-to-noise ratio）尽量大。输出信噪比越大，错误率越小。

定位精度准则：检测出的边缘尽可能接近真实边缘。

单边缘响应准则：对同一边缘响应最好不要重复，即对单边缘最好只有一个响应。

#### **五、Canny算法的步骤：**

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，同时也会可能增加边缘的宽度。对于在图像中位置 (*m, n*) 的像素点，其灰度值为*f*(*m, n*)。

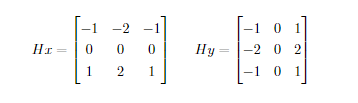
经过高斯滤波以后，灰度值变为：



（2）计算梯度值和梯度方向。

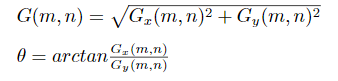
在图像中，梯度是用来表示灰度值变化程度和方向。

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*



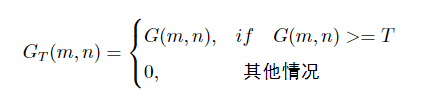


综合*x*和*y*轴的梯度，即可通过以下公式计算梯度值*G*(*m, n*)和梯度方向：



（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，使边缘宽度尽可能为1个像素点。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。



（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大于T2，则定义为边缘点；如果小于T1，则是定义为非边缘点；介于T1和T2之间的灰度值，如果该像素点与已经定义了的边缘点连接的话，那这个像素点也被定义为边缘点。如图2-6所示。

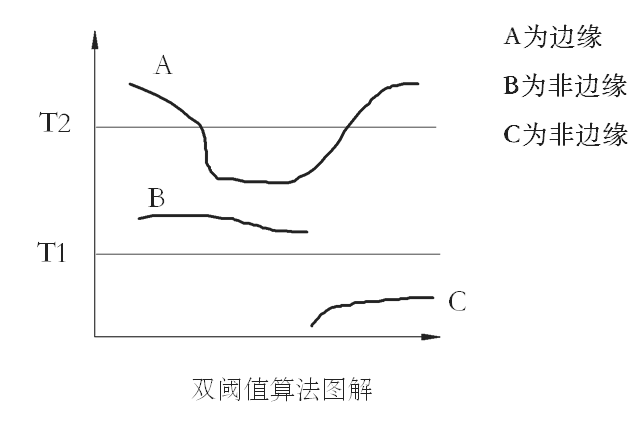
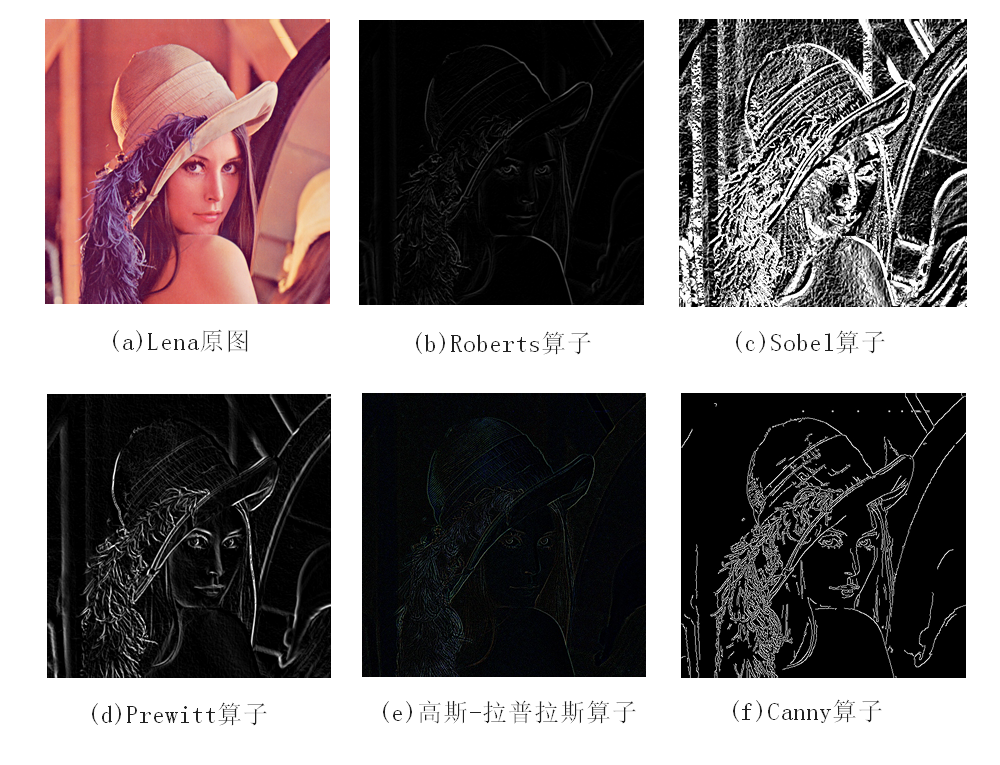


图2-6 双阈值算法图解

#### **六、各种边缘检测算子的比较**



### **2.2.4 直线检测**

#### **一、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。

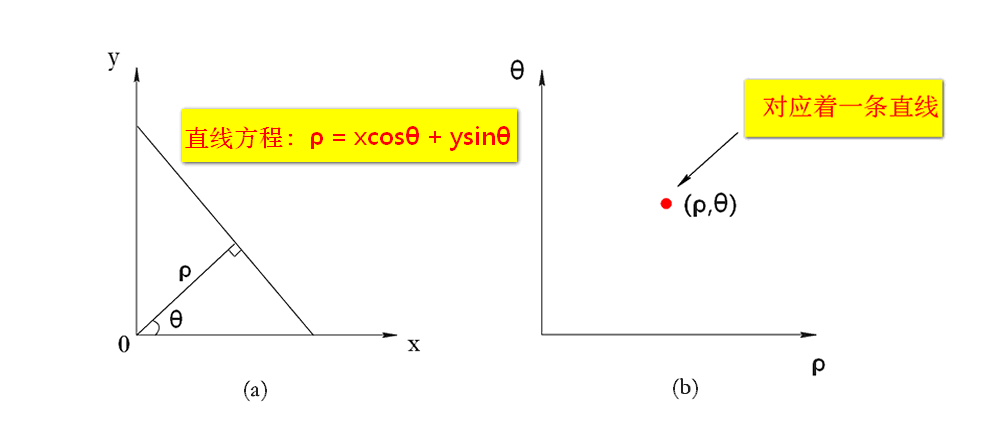
#### **二、Hough变换**

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。

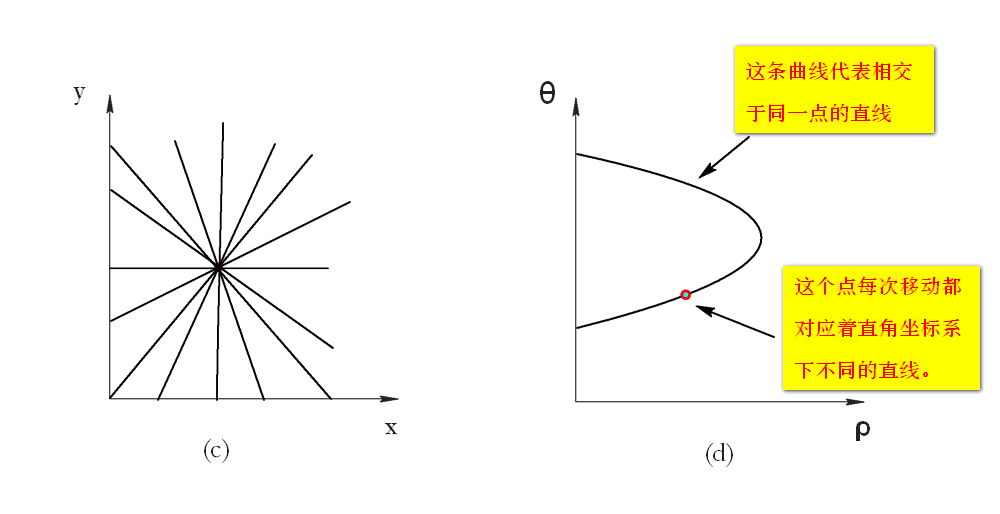
#### **三、Hough变换的基本思想[2]：**

对于边界上有n个点的点集，找出共线的点集和直线方程。

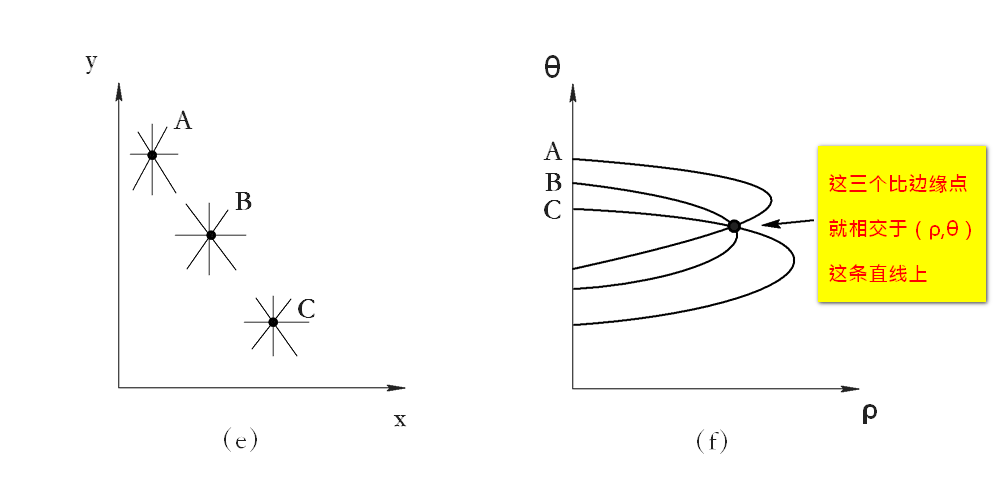
1. 对于直角坐标系上的一条直线*l*，可以用点法式方程来表示直线，如图（a）所示，直线方程为：。其中为原点到该直线的垂直距离，为垂线与*x*轴的夹角，这条直线是唯一的。
2. 直接坐标系下的一条直线对应着极坐标系上一个点(*,*)，如图（b）所示，这种线到点的变换就称为Hough变换。



1. 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（c）所示。那么这个边缘点对应的所有直线，在(*,*)极坐标下就成了一条曲线，如图（d）所示。



1. 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（e）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(就是A, B, C相连的直线。且该直线的方程为：。

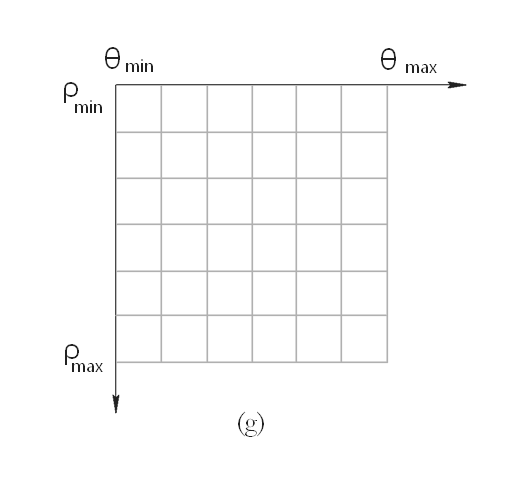


#### **四、Hough变换算法实现：**

使用交点累加器，找出(*,*)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. 在和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（g），用来统计曲线交点累加次数。



1. 对图像上的所有边缘点作Hough变换计算出所有，的值，每计算出一对，值，对应二维数组A中元素，并且让该数组元素值加1。
2. 最后，比较数组A中所有元素的值，最大值对应的，就是这些共线点对应的直线方程的参数。

#### **五、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：

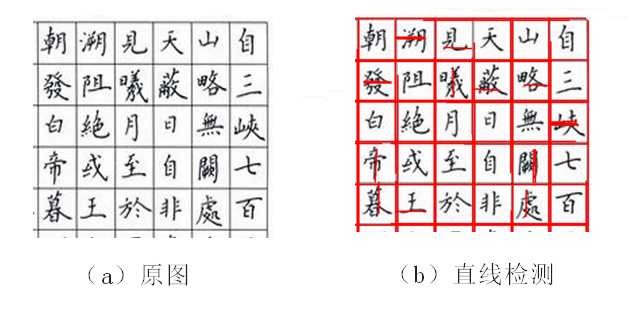
[[[ 3 392 3 3]]

[[329 392 329 3]]

[[ 4 392 328 392]]

[[ 4 3 328 3]]]

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[[7]](#footnote-7)。



### **2.2.5 轮廓提取**

#### **一、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。表示方法可能根据不同情况而有所不同。有多种方法可以表示曲线。在OpenCV中一般用序列（某种格式的链表，在内存被实现为一个双端队列）来存储轮廓信息。序列中的每一个元素是曲线中一个点的位置。

为了得到更准确的轮廓，需要对图像进行Canny边缘检测或二值化等预处理。轮廓提取就像是在黑色的背景中找出白色的物体，所以就需要预处理将图像的前景变成白色，背景变成黑色。

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2-7所示。其中“c”表示轮廓，又称外轮廓；“h”表示孔，又称内轮廓。

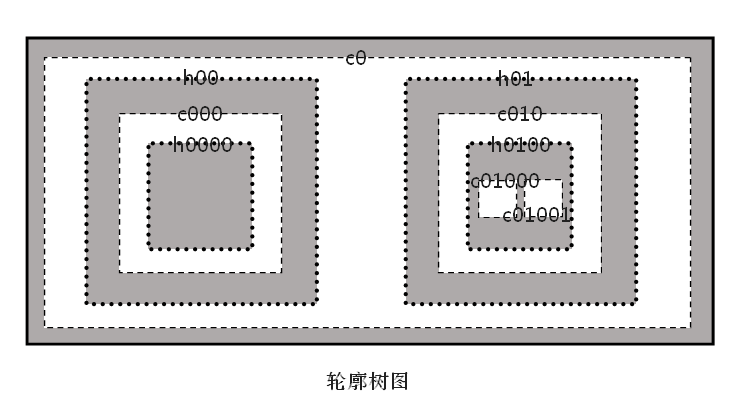


图2-7 轮廓树图

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2-8所示。

|  |  |
| --- | --- |
| mode值 | mode的实质含义 |
| cv.RETR\_EXTERNAL | 只检测最外的轮廓 |
| 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 |

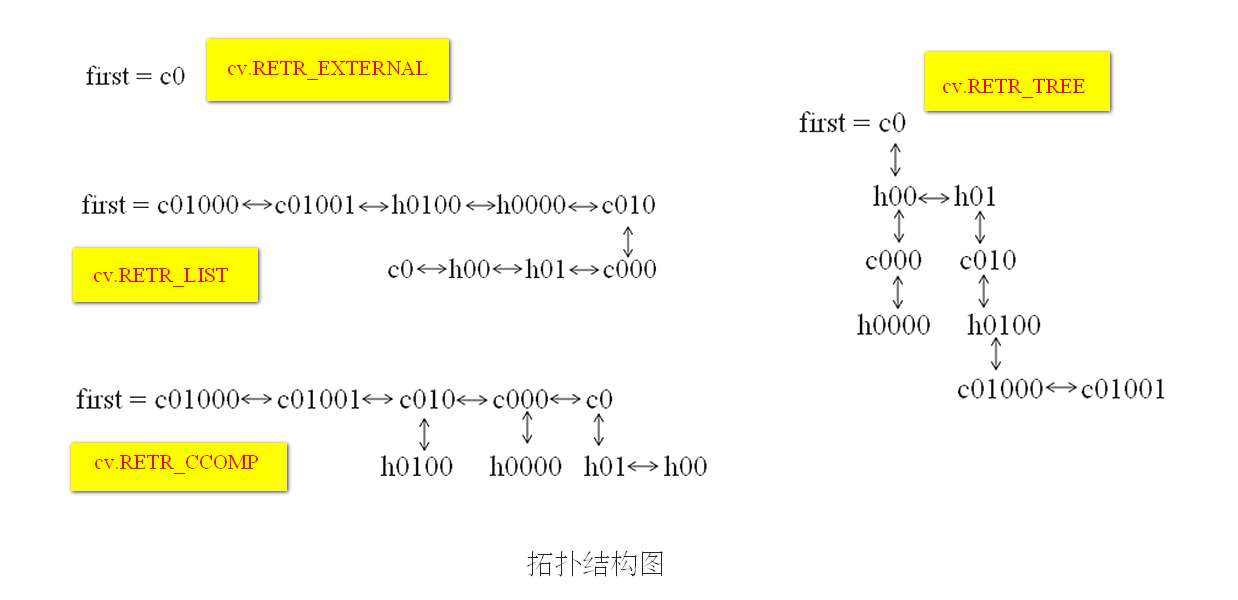


图2-8 拓扑结构图

### **2.2.6 角点检测**

#### **一、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[[8]](#footnote-8)，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、运动估计、形状分析、视觉的定位和测量等方面的应用中起着重要的作用[4]。

#### **二、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。

#### **三、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。

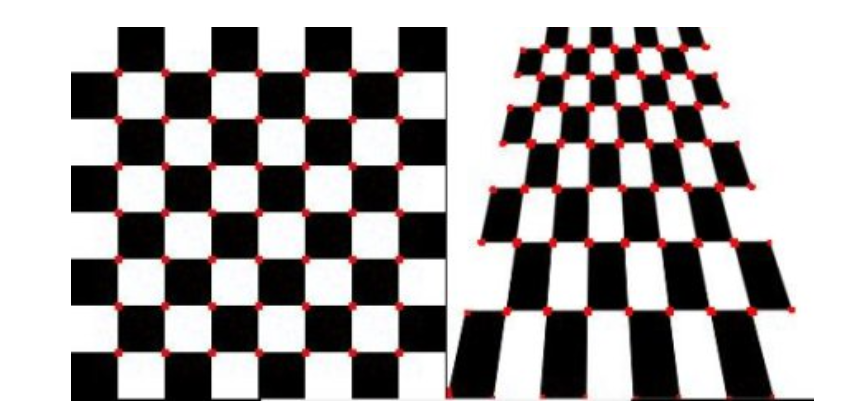


图2-9 角点检测

## 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。UW: 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]。

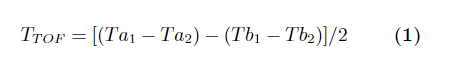
### 2.3.2 UWB的定位技术原理

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2-10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：



两节点间的距离如公式（2）所示：

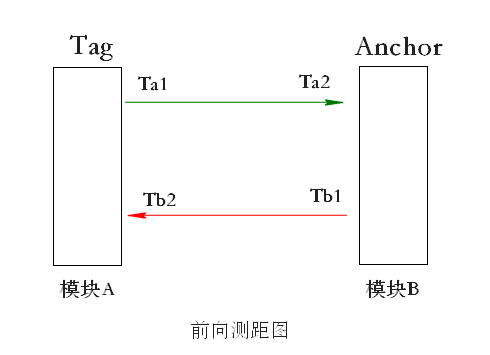
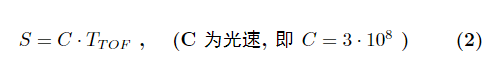
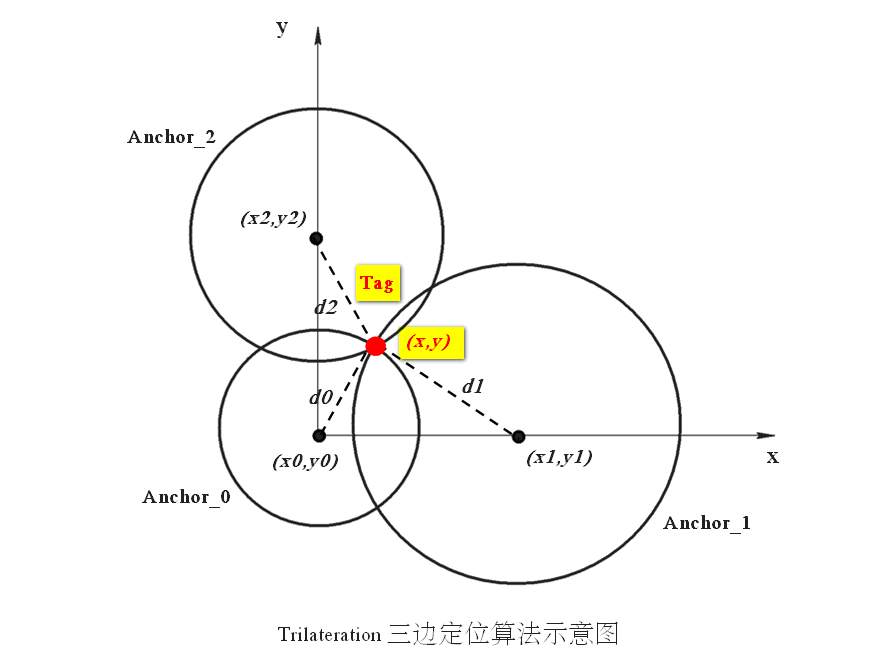


图2-10 前向测距图

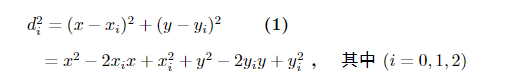
### 2.3.4 Trilateration 三边定位算法

#### **一、Trilateration算法的理解**

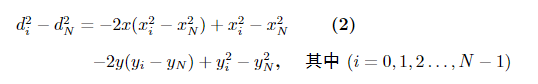
三边测量法(Trilateration)。通过测量标签（Tag）到三个或三个以上的基站（Anchor）的距离，运用几何原理求解目标位置。



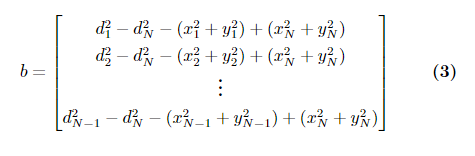
假如Tag坐标位置为（*x, y*），通过TOF测出Tag与三个Anchor的相对距离为（其中*i* = 0, 1, 2），则有距离公式[8]：

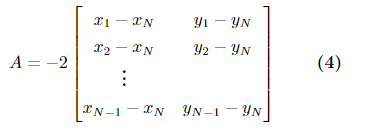


公式（1）中存在非线性项和。通过与的互减操作可消去非线性项，假如（*i* = 0, 1, 2, … N-1），



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 =*，其中，





当N = 3时，*A*为2\*2方阵且可逆，那么*X =*  当N>3时，可用

求解。

#### **二、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 本章小结

本章主要简单介绍了建筑工程图的分类，OpenCV在本文将会用到的基础运用知识及每一部分功能的原理概述，UWB的原理介绍及其TOF测距和三边定位算法原理介绍。能给读者对该论文内容有个粗略的前提认识。

# 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

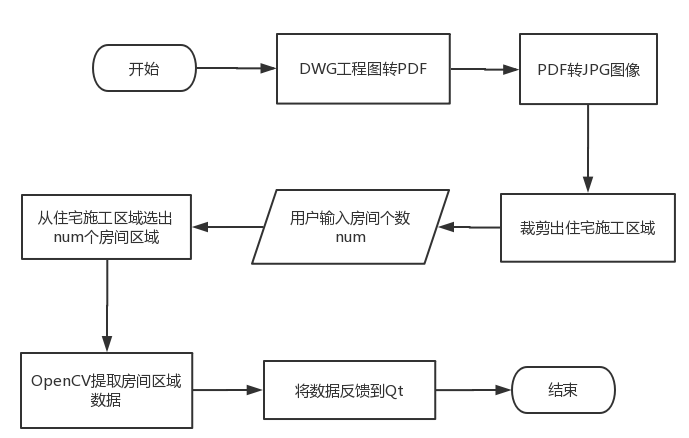


图3-1 本章主要流程图

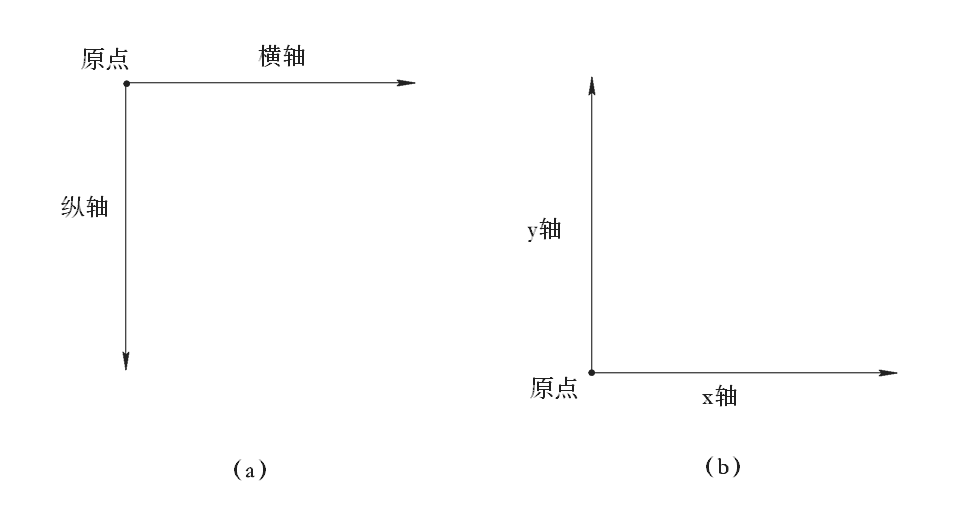
#### **一、预处理：**

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：

1. **from** wand.image **import** Image as wi
3. pdf = wi(filename='./cad.pdf', resolution=300)
4. pdfImage = pdf.convert("jpeg")
5. i = 1
6. **for** img **in** pdfImage.sequence: #有多张pdf
7. page = wi(image=img)
8. page.save(filename=str(i)+".jpg")
9. i += 1
11. # 只有一张pdf，就可以使用下面语句
12. # img = pdfImage.sequence[0]
13. # page = wi(image=img)
14. # 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。

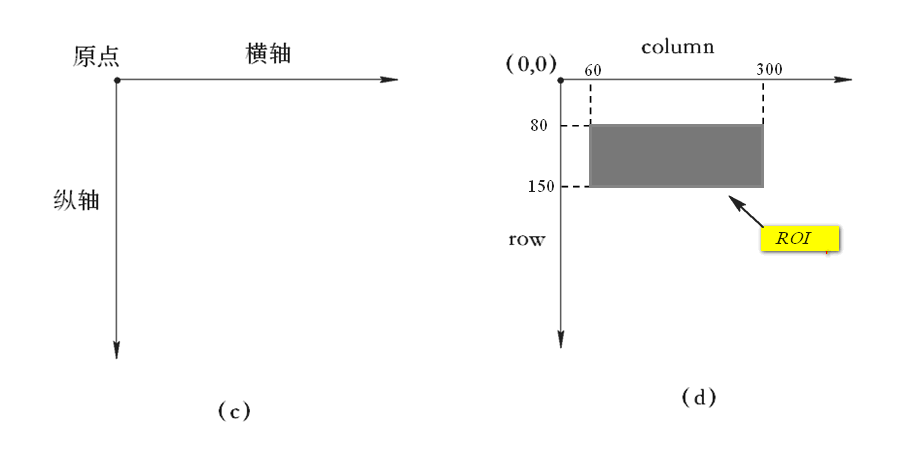
OpenCV的坐标系如下图（a）所示，而常规的直角坐标系如下图（b）所示，我们尚且称OpenCV图像中的x轴为横轴，y轴为纵轴。在本课题中，应用到UWB室内定位的过程的时候会把OpenCV的坐标系转换为常规的直角坐标系，原因有两点：第一、为了规范化，在数学教材中常见的坐标系都是如图(b)所示，那么在本课题按照常规，规定y轴向上为正方向，且沿着y轴延伸的方向纵坐标数值越大。第二、为了便于在Qt上按照这种统一规定的坐标系进行绘图。



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[[9]](#footnote-9)，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(d)所示，即能提取出特定位置的ROI区域。



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，得到如下图所示的工程图片。

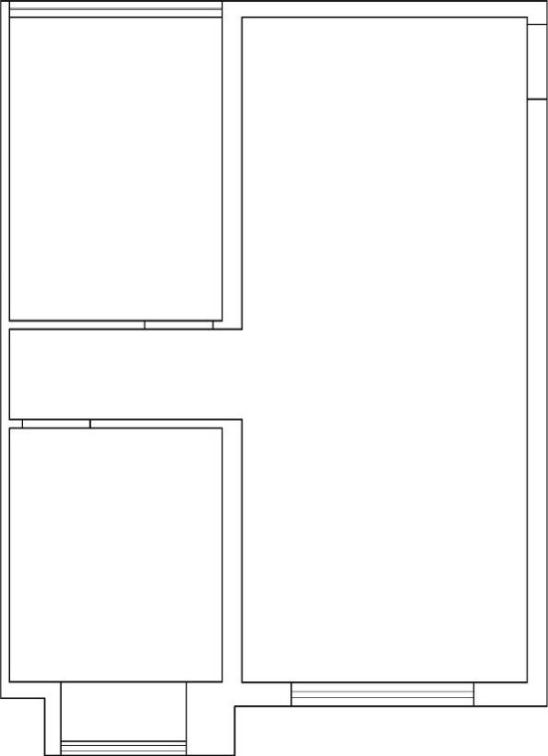


图3-2 住宅施工区域

## 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3-3。

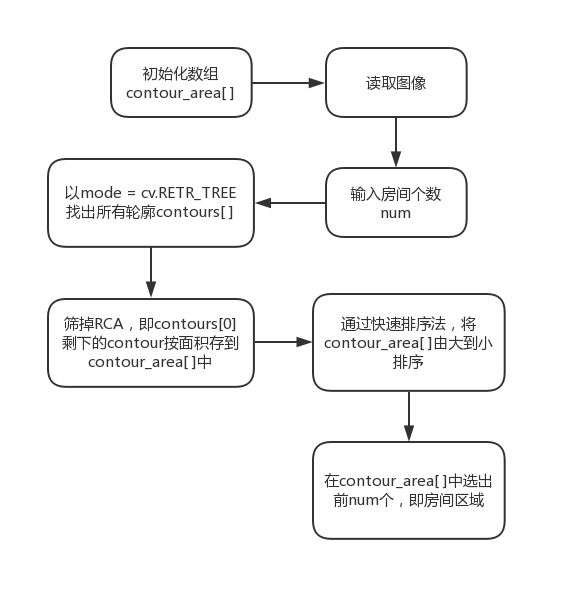


图3-3 房间区域划分状态图

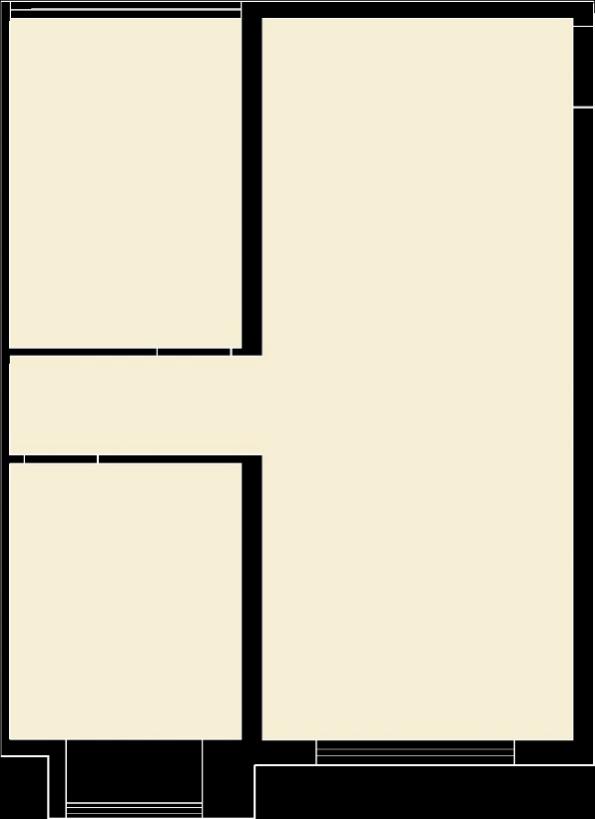


图3-4 房间区域划分图

## 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。

#### 一、提取区域的哪些数据

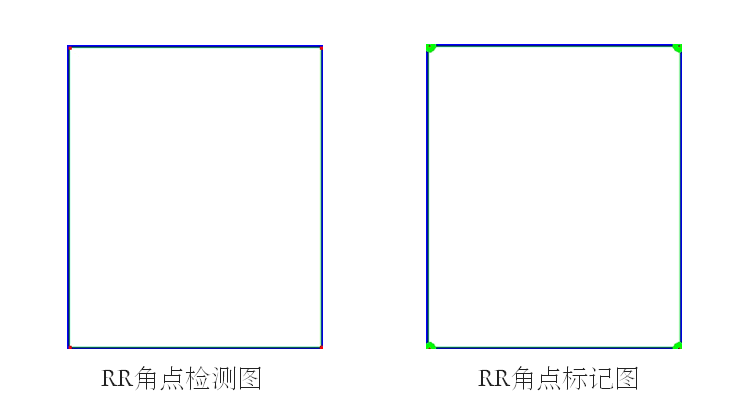
由于本课题后面是采用UWB进行室内定位的，那么UWB定位系统中能够进行二维的平面定位，也可以进行三维的立体面定位。在本课题中，砖块铺设目前只需要考虑二维平面室内面积大小和形状即可。那么根据前面介绍的OpenCV坐标系可知，在图像中需要找出房间区域的坐标系原点；还要找出房间区域的长（length）和宽（width）。

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？

#### 二、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左图所示四个红色的点。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，如右图所示。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。

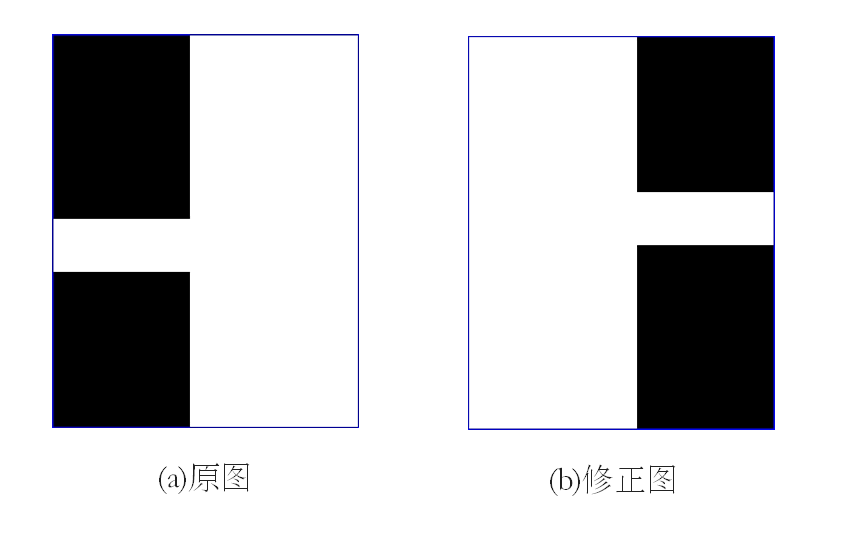


#### 三、矩形组合区域

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(a)所示，所谓RCR，即由多个矩形进行拼合组成的区域，下图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。

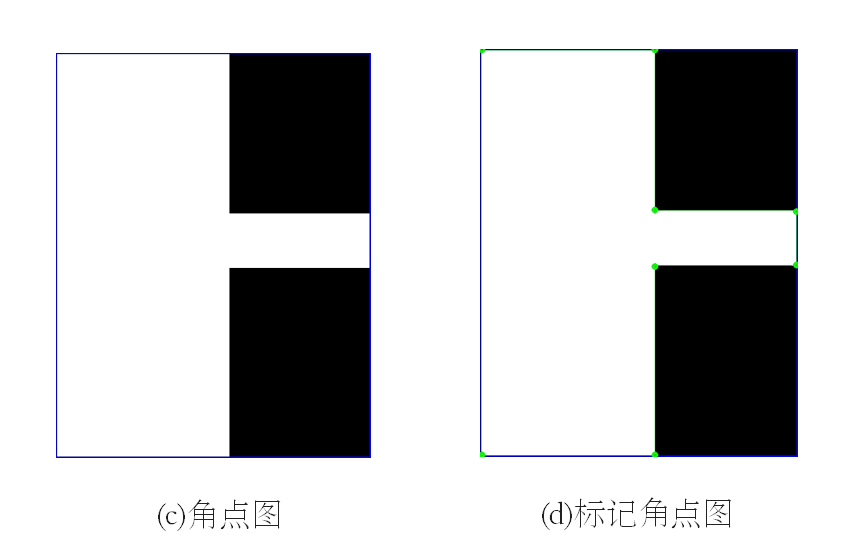
根据常规直角坐标系，是将左下角的顶点作为坐标系的原点。在上面这个矩形组合区域中，直接将左下角作为坐标系原点不太合理，那么利用OpenCV提供的函数cv2.rotate()进行顺时针旋转两次即可得到修正过的区域图纸了，如图(b)所示。



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（c），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。



具体遍历定位过程如下：

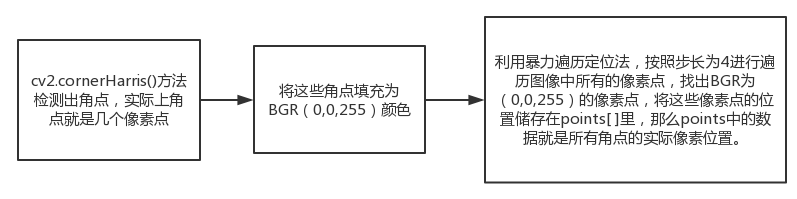
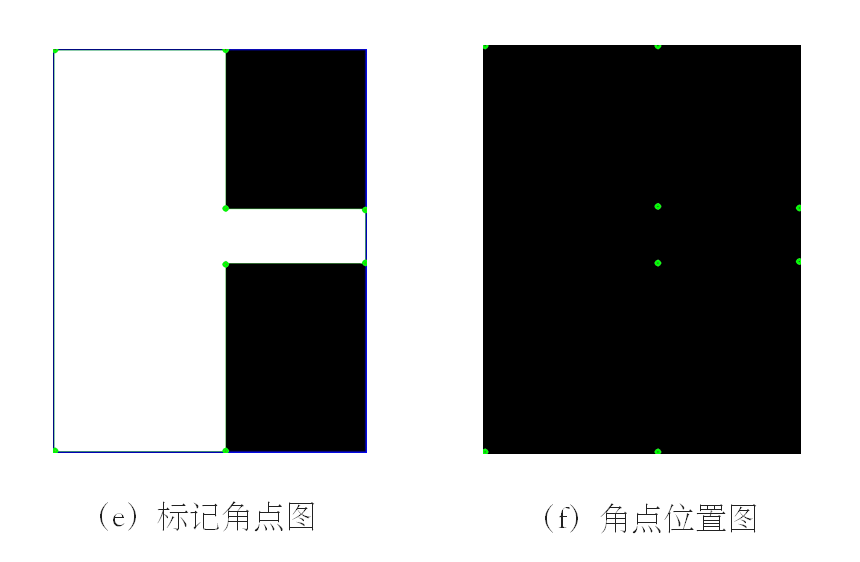


图3-5 角点遍历定位状态图

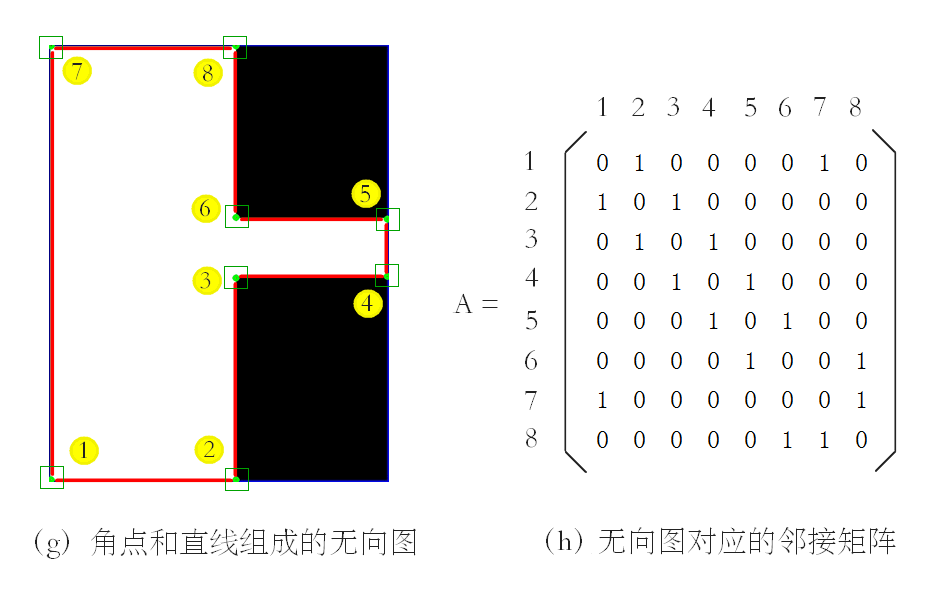
第二步：排序角点

何谓排序角点？如何排序角点呢？

如图（e），是实际图像中提取角点得到的图像；而图（f），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（f）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。

****

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（g）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。



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。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（i）。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（k）。

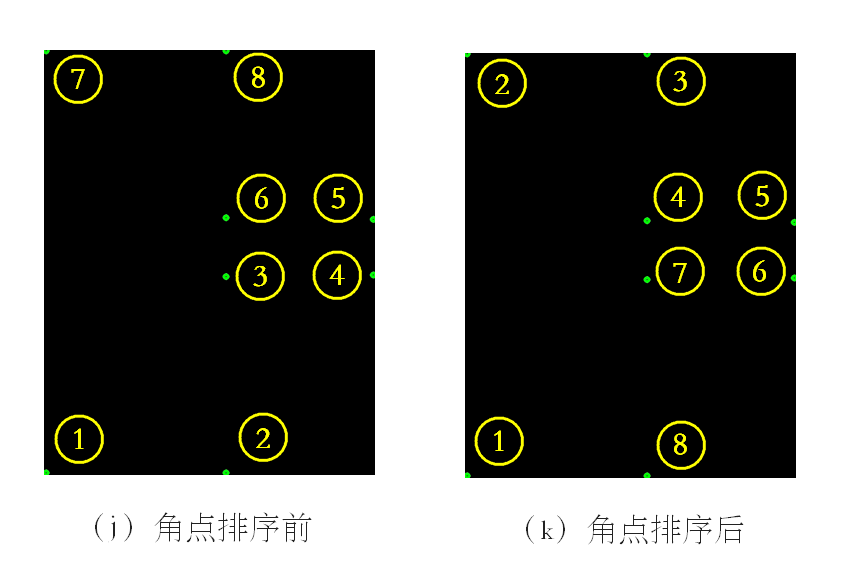
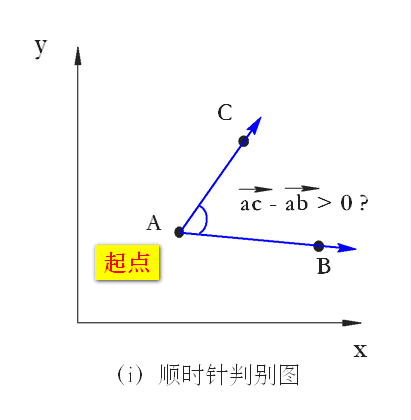


表3-1 RCR各阶段的角点阵

|  |  |  |
| --- | --- | --- |
| 未排序的角点阵 | 排序后的角点阵 | 校正后的角点阵 |
| [[ 4 1033]  [ 444 1033]  [ 444 553]  [ 804 549]  [ 804 413]  [ 444 409]  [ 4 1]  [ 444 1]] | [[ 4 1033]  [ 4 1]  [ 444 1]  [ 444 409]  [ 804 413]  [ 804 549]  [ 444 553]  [ 444 1033]] | [[ 0 0]  [ 0 1032]  [ 440 1032]  [ 440 624]  [ 800 620]  [ 800 484]  [ 440 480]  [ 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。
3. zero\_col = points[0][0] #第一个角点的横轴
4. zero\_row = points[0][1] #第一个角点的纵轴
6. **for** i **in** range(pnum): #pnum为角点阵大小
7. # 先将所有点都以第 0 点作为（0,0）的法则来进行调整
8. points[i][0] -= zero\_col
9. points[i][1] = zero\_row - points[i][1]
11. **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

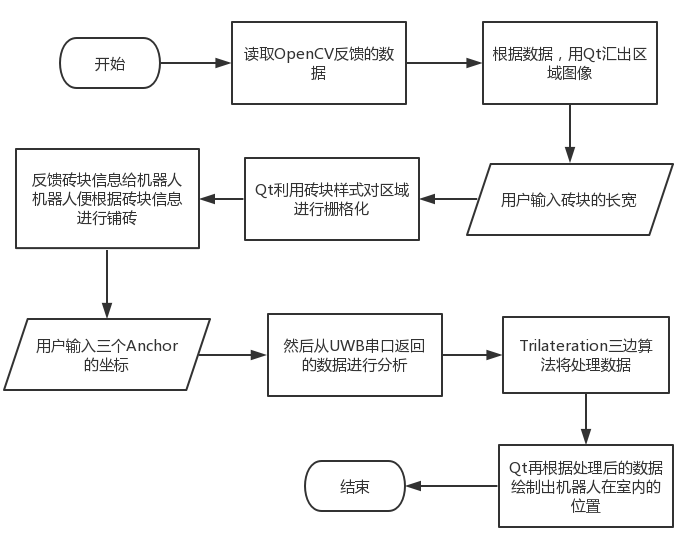


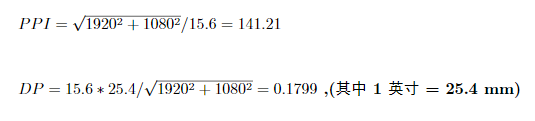
图4-1 本章主要流程图

## 4.3 根据区域数据绘制区域图像

#### **一、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。

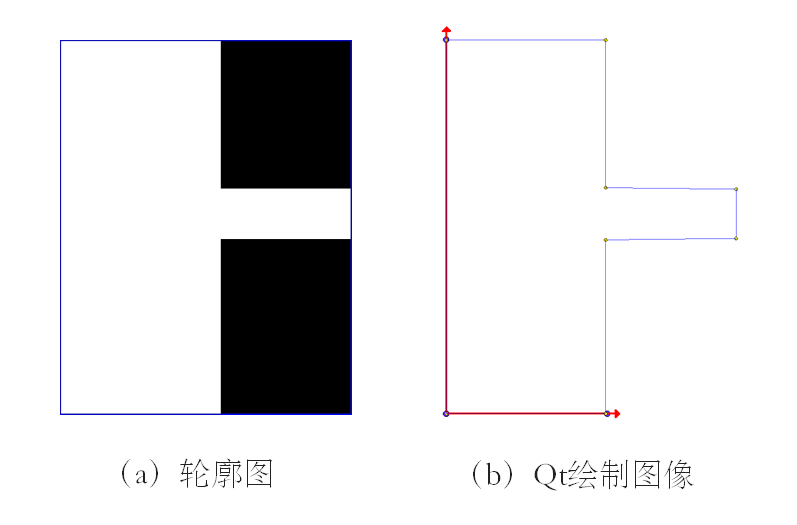
根据计算公式，需要通过计算PPI和DP[[10]](#footnote-10)才能将像素距离转换为实际距离，那么要将像素转换到实际尺寸就要先知道电脑的显示器配置才行。假如用户电脑的分辨率为：1920\*1080，电脑尺寸为15.6英寸。计算公式如下：



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。

#### **二、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。下图（a）是RCR轮廓图像，图(b)为Qt绘制图像。



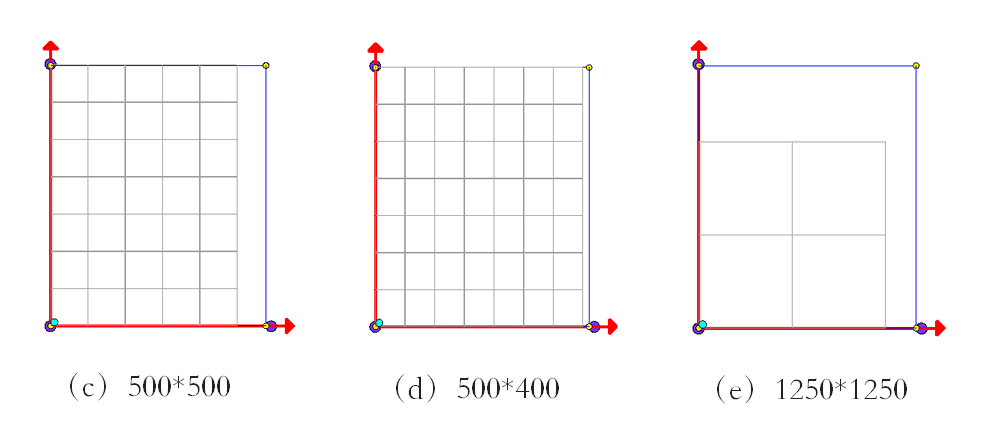
## 4.4 进行砖的摆放规划

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(c)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(d)中，此时的砖块大小为长500mm，宽400mm， 砖块之间的间隙为5mm。

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

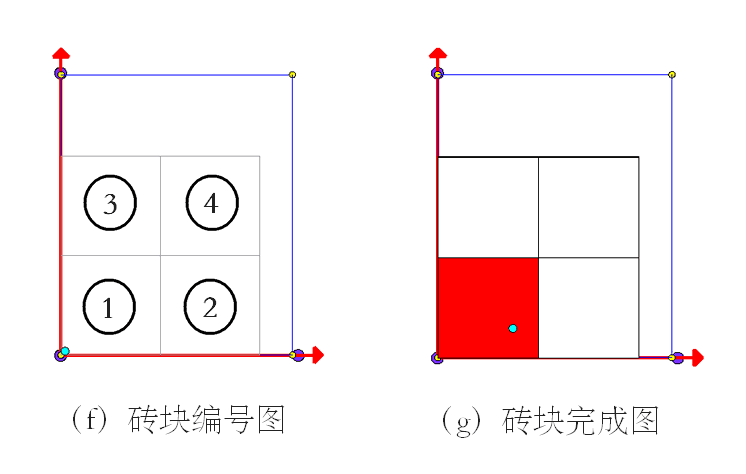


## 4.5 将砖块信息传递给机器人

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（f）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（g）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。 |



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

#### 一、数据格式解析：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| MID | MASK | RANGE0 | RANGE1 | RANGE2 | RANGE3 | NRANGES | RSEQ | DEBUG | aT:A |
| mr | of | 000005a4 | 000004c8 | 00000436 | 000003f9 | 0958 | c0 | 40424042 | a0:0 |
| ma | 07 | 00000000 | 0000085c | 00000659 | 000006b7 | 095b | 26 | 00024bed | a0:0 |
| mc | 0f | 00000663 | 000005a3 | 00000512 | 000004cb | 095f | c1 | 00024c24 | a0:0 |

#### 二、TOF数据主要的索引含义表

|  |  |
| --- | --- |
| 索引 | 含义 |
| MID | 消息ID，一共有三类，分别为mr ,mc, ma  mr 代表标签-基站距离（原生数据）  mc 代表标签-基站距离（优化修正过数据，用于定位标签）  ma 代表基站-基站距离 （优化修正过数据，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 或 mr,表示标签 x 到基站 0 的距离,单位:毫米 |
| RANGE1 | 如果 MID = mc 或 mr,表示标签 x 到基站 1 的距离,单位:毫米  如果 MID = ma， 表示基站0到基站1的距离，单位：毫米 |
| RANGE2 | 如果 MID = mc 或 mr,表示标签 x 到基站 2 的距离,单位:毫米  如果 MID = ma， 表示基站0到基站2的距离，单位：毫米 |
| RANGE3 | 如果 MID = mc 或 mr,表示标签 x 到基站 3 的距离,单位:毫米  如果 MID = ma， 表示基站1到基站2的距离，单位：毫米 |

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。

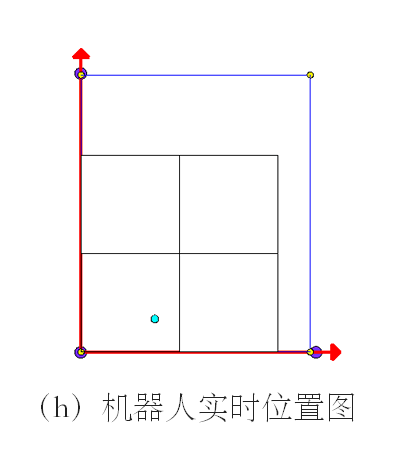
1. **for** i **in** range(4):
2. anchor[i].x = np.float(self.anchorTable.item(i,1).text())
3. anchor[i].y = np.float(self.anchorTable.item(i,2).text())
4. anchor[i].z = np.float(self.anchorTable.item(i,3).text())

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（X）-基站（Y）的距离数据。其中X表示{0}，而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（h）所示。



## 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++。

#### **一、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[[11]](#footnote-11)，导致在大型项目中运行效率低。

#### **二、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[[12]](#footnote-12)。对于图像处理[[13]](#footnote-13)，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。

#### **三、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。

#### **四、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。

为了突破这种限制，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。

#### **五、使用SWIG进行Python和C++的混合编程**

1. example.h：头文件
2. #include<iostream>
3. **using** **namespace** std;
5. **int** fact(**int** n);
6. example.cpp : 源文件（主要实现的功能是阶乘）
7. #include "example.h"
9. **int** fact(**int** n){
10. **if** (n < 0){
11. **return** 0;
12. }
13. **if** (n == 0){
14. **return** 1;
15. }
16. **else**{
17. **return** n \* fact(n-1);
18. }
19. }
20. 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)
21. %module example
23. %{
24. #include "example.h"
25. %}
27. %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。

1. 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。
2. **from** distutils.core **import** setup, Extension  # 加入混合编程需要的库
4. example\_module = Extension("\_example",sources=["example.cpp","example\_wrap.cxx",],) # 源文件模块
6. setup(name='example',
7. version='0.1', # 版本号
8. author='Jun', # 作者名字
9. description="""Simple swig example from docs""", # 描述
10. ext\_modules=[example\_module],
11. py\_modules=['example'], # Python模块
12. )

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

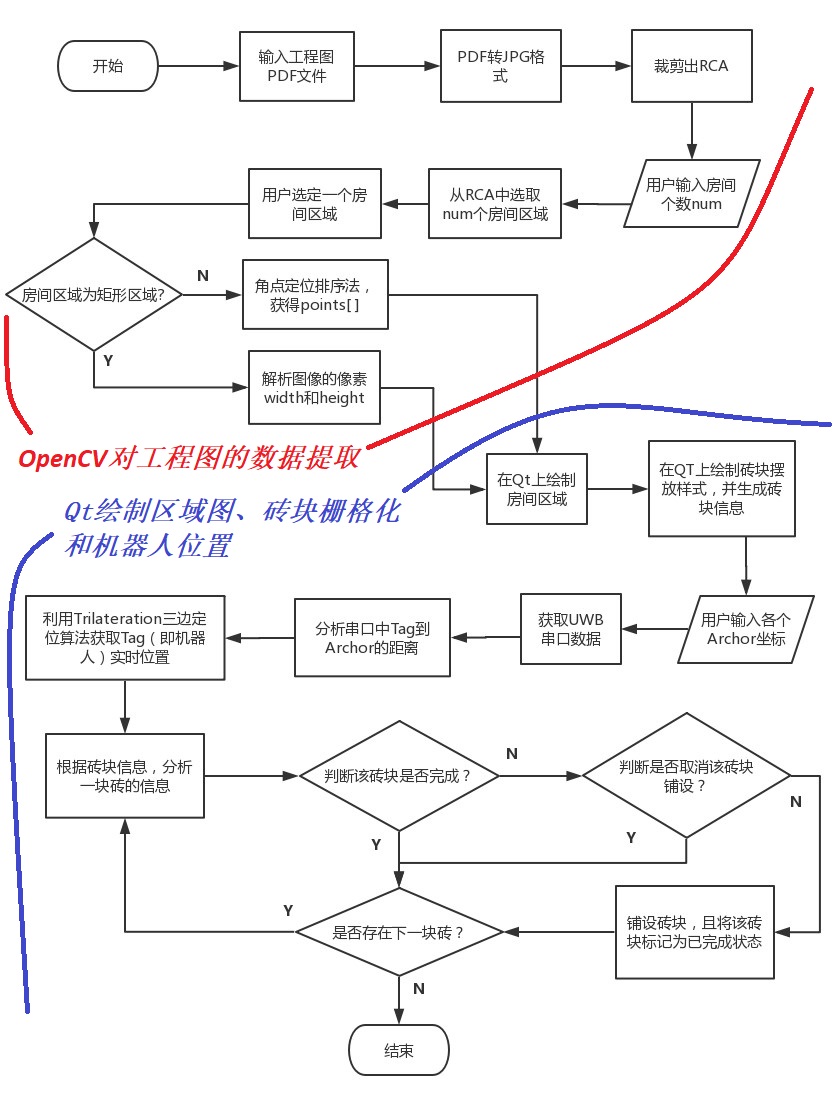


图5-1 系统总流程图

## 5.2 系统细节分析图解

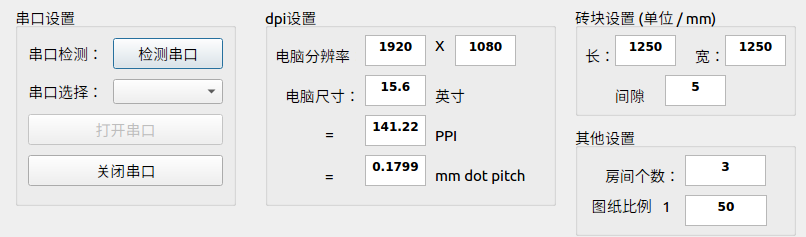


图5-2 系统的前提设置

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m，OpenCV检测出区域的所有轮廓之后，会选出面积最大的m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。

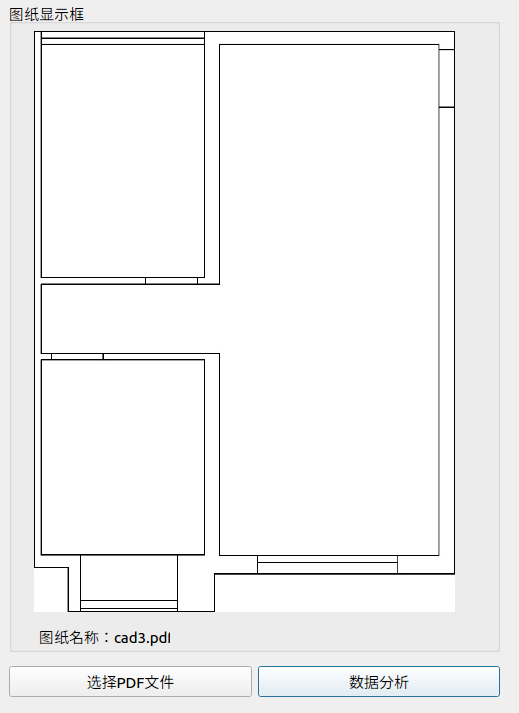


图5-2 图纸显示框

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。

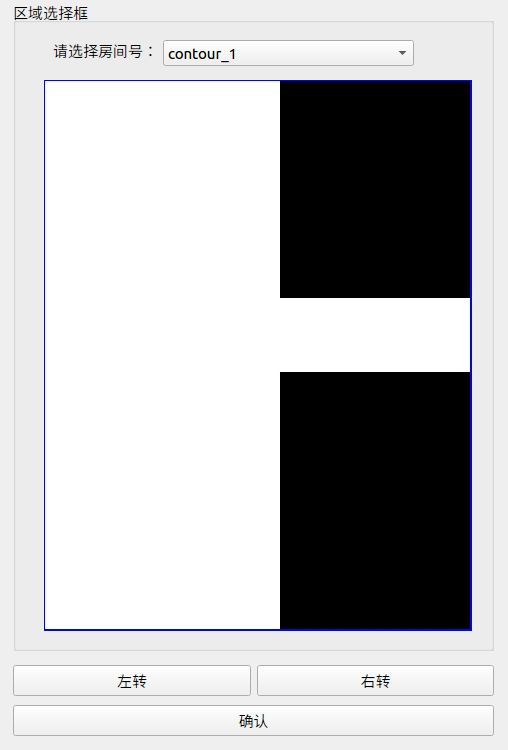


图5-3 区域选择框

用户在“请选择房间号”的下拉列表选择出一个房间。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。

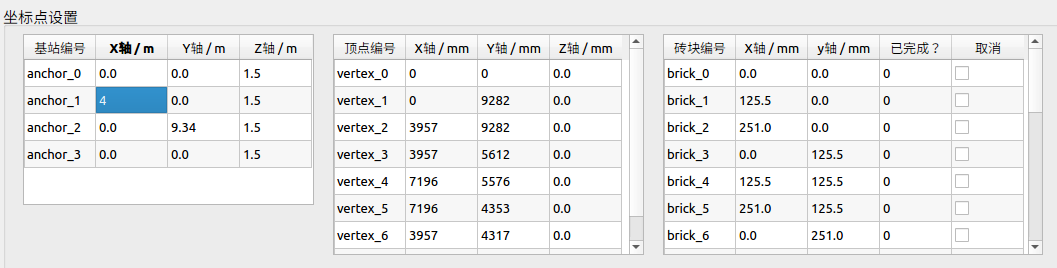


图5-4 坐标点设置

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TOF测距，会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。

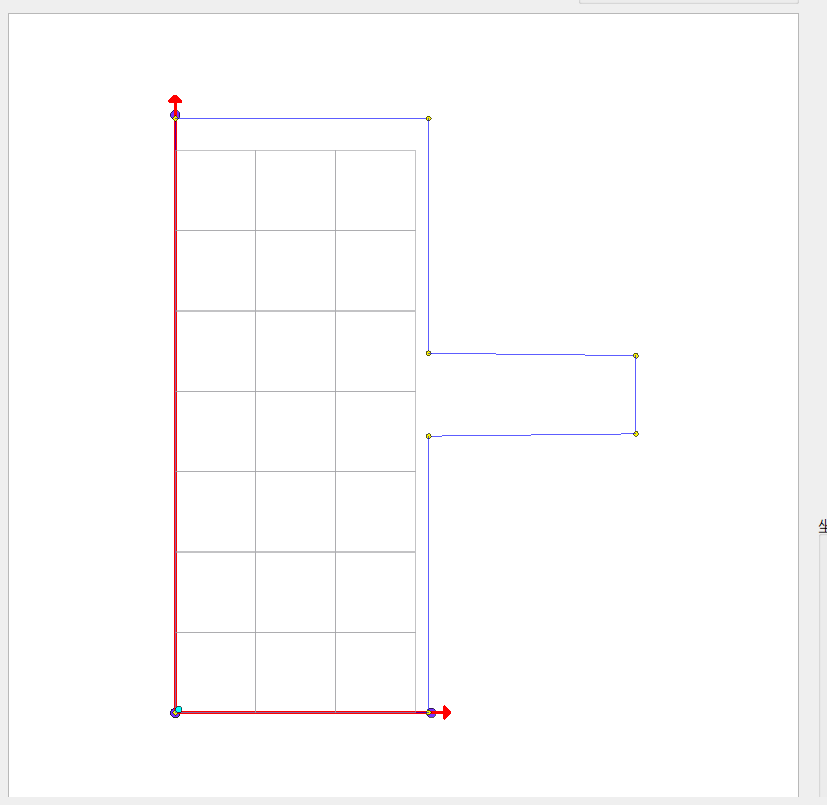


图5-5 Qt绘制房间区域及其砖块栅格化

图5-5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。

## 5.3 矩形区域分析图

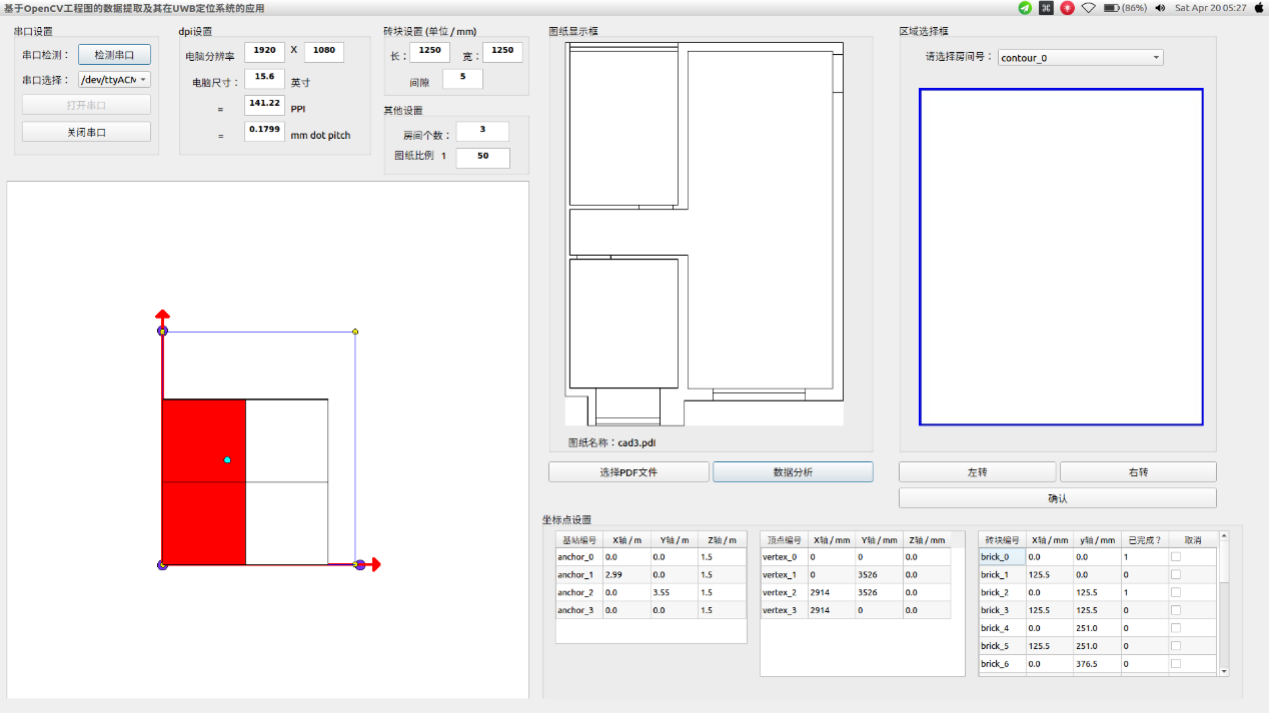


图5-6 系统运作图

图5-6就是系统整体的运作图，其中选择的房间区域为矩形区域。

## 5.4 矩形组合区域分析图

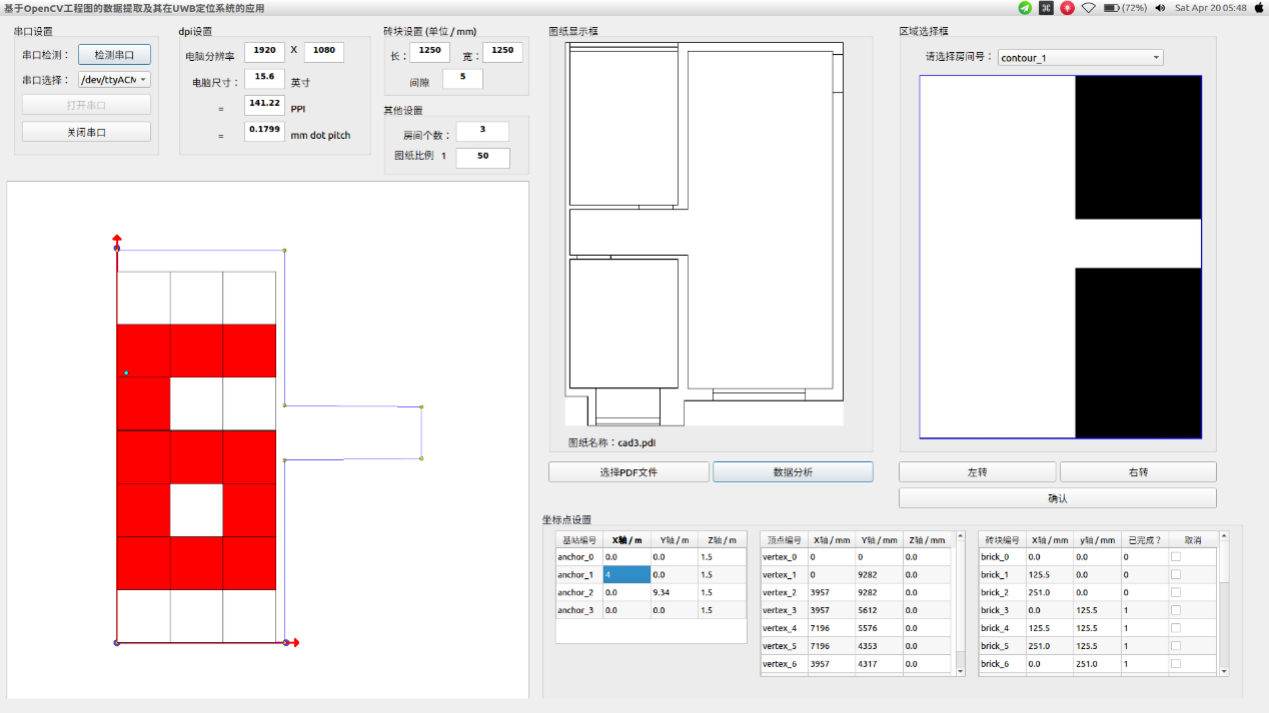


图5-7 系统运作图

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]. 北京市清华大学出版社, 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]. 北京市人民邮电出版社, 2010.
3. Gonzalez R , Woods R , Eddins S , et al. 数字图像处理的MATLAB实现[M]. 北京市清华大学出版社, 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(4):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(3):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Auto CAD.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(5):58-63.

# 致谢

感谢家人，感谢朋友，感谢导师，感谢老师，感谢自己。

# 附录



图1为荷兰RoadPrinter公司的一种大型半自动铺路砖机Tiger-Stone

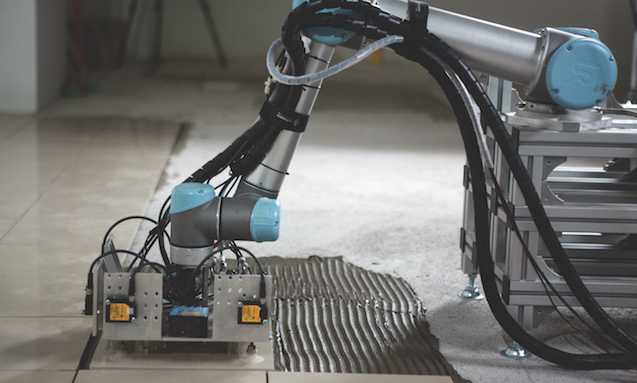


图2为新加坡一款全自动化能铺地板砖的6轴机械臂

1. <https://opencv.org/> [↑](#footnote-ref-1)
2. <https://www.qt.io/cn> [↑](#footnote-ref-2)
3. 点运算：是指对一副图像中每个像素点的灰度值进行计算的方法。 [↑](#footnote-ref-3)
4. 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 [↑](#footnote-ref-4)
5. 算子：是将一个元素在向量空间（或模）中转换为另一个元素的映射。——来源维基百科 [↑](#footnote-ref-5)
6. 滤波：是信号处理的一个概念，是将信号中特定波段频率滤除的操作，在数字信号处理中通常通过傅里叶变换及其逆变换实现。 [↑](#footnote-ref-6)
7. 这些点的位置都是基于OpenCV坐标系。 [↑](#footnote-ref-7)
8. <https://docs.opencv.org/3.4.2/df/d54/tutorial_py_features_meaning.html> [↑](#footnote-ref-8)
9. <https://stackoverflow.com/questions/25642532/opencv-pointx-y-represent-column-row-or-row-column> [↑](#footnote-ref-9)
10. 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。 [↑](#footnote-ref-10)
11. 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 [↑](#footnote-ref-11)
12. 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 [↑](#footnote-ref-12)
13. OpenCV的源代码就是C/C++编写的 [↑](#footnote-ref-13)