郑州大学毕业设计（论文）

题 目 圆柱测量数据的图形化显示系统开发

院 系 机械工程

专 业 机械工程

年 级 2014级

学生姓名 时猛

指导教师 杨炯

2018年 05月 20日

**摘要**

本文对圆柱数据的测量的技术发展进行了研究和阐述，针对圆柱数据的测量系统，本文设计了一种高精度的测量数据处理方法，并开发了一种形象可视化的上位机软件。利用面向对象技术，多线程技术以及OpenGL图形开发技术等，这些关键技术为整个项目的软件架构和软件设计提供了很大的便利，提高了软件的开发效率以及软件的高效性。

测量数据的显示系统包括数据通信，数据拟合，图形展示等。数据通信是基于半双工通信与PLC连接，获取PLC的测量数据；数据拟合采用最小二乘法圆度拟合。图形展示是基于OpenGL将测量的数据形象的展示为图形，处理后的结果精度高，数据重复性好。上位机软件是基于C++编程语言开发而成的，集成了二维图形展示，状态栏，菜单栏，数据导出以及三维图形展示等功能，打造了一个人性化的软件界面。

关键词 圆度拟合， 图形化显示， 圆拟合（关键词3-5个）

Abstract

In this paper, the technical development of cylindrical data measurement is studied and elaborated. For the measurement system of cylindrical data, this paper designs a high-precision measurement data processing method, and develops a visualized upper computer software. Using object-oriented technology, multi-threading technology, and OpenGL graphics development technology, these key technologies provide great convenience for the software architecture and software design of the entire project, improving the efficiency of software development and the efficiency of software.

   The display system of measurement data includes data communication, data fitting, graphic display and so on. Data communication is based on half-duplex communication and PLC connection to obtain the measurement data of PLC; data fitting is based on least-squares roundness fitting. The graphic display is based on OpenGL's display of the measured data image as a graph, with high precision after processing and good data reproducibility. The host computer software is developed based on the C++ programming language. It integrates functions such as 2D graphics display, status bar, menu bar, data export, and 3D graphics display, creating a personal software interface.

key Words Roundness fitting，Graphical display，Circle fitting

目 录

毕业设计任务：………………………………………………………… 1

摘要……………………………………………………………………… 2

Abstrct…………………………………………………………………… 2

1 绪论…………………………………………………………………… 3

1.1 选题的目的意义………………………………………………………3

1.2 国内外研究综述………………………………………………………3

1.3 本题的主要工作………………………………………………………3

2 算法分析 ……………………………………………………………… 4

2.1 圆拟合算法分析……………………………………………………… 4

2.2 圆柱拟合算法分析……………………………………………………5

2.2.1 圆柱面拟合方法原理……………………………………………… 8

2.3 本章小节………………………………………………………………10

3 网络通信…………………………………………………………………10

3.1三菱PLC的分类及FX系列PLC的发展历史………………………10

3.2 FX系列 PLC各个功能模块分析…………………………………… 10

3.3 本章小节 …………………………………………………………… 12

4 系统架构……………………………………………………………… 12

4.1 C++简介 …………………………………………………………… 12

4.2 环境设置……………………………………………………………… 12

4.3 C++ 基础……………………………………………………………… 13

4.4 Qt框架………………………………………………………………… 17

4.4.1 Qt环境配置………………………………………………………… 18

4.4.2 Qt面向对象………………………………………………………… 19

4.4.3 Qt组件间的通信…………………………………………………… 20

4.4.3.1 槽 ………………………………………………………………… 20

4.4.3.2 信号……………………………………………………………… 20

4.3.4 Qt的组成部分……………………………………………………… 21

5. 辅助工具 ……………………………………………………………… 23

5.1 版本控制……………………………………………………………… 23

5.2 单元测试……………………………………………………………… 23

6 结论…………………………………………………………………… 23

致谢……………………………………………………………………… 25

参考文献 ……………………………………………………………………26

1前言

1.1选题的目的意义

在计算机视觉辨认中 , 圆、点与直线一直是最基本的图象的基本特征在我们的生活中包含的大量物体为圆,圆形既是世界上最简单最基本的形状，而且是人类社会产品中普遍存在的形状, 圆的鲁棒精确提取在工业领域和检测领域中有着非常特别重要的意义。

1.2 国内外研究综述

在测量检测的过程中, 通常我们会遇到三维圆柱图形的测量问题, 例如在一些建筑物的顶层、大型机械设备仪器和平时乘坐的地铁遂道的工程测量中, 通常我们是对三维空间物体进行检测，求出它的圆心以及半径,确定我们是否能够施工。并且在机械领域，各个零件的圆孔定位对零部件的安装和装配有这非常重要的作用， 进行零部件圆中心定位的各种方法为:从图像边缘检测所得数据进行圆的拟合如在一些大型仪器和地铁地道的工程丈量中，常常必要对空间圆形物体举行丈量，得出其圆心及半径，肯定其是不是知足施工请求。操纵有限样本点举行圆拟合的环境，经常使用的方式有:

1. 平均值法

针对样本点均匀分布在圆上的情况，从理论上来说，其圆心坐标为各点坐标的平均值，圆心到任一点的距离即为圆的半径。但是在实际问题中，样本点不可能正好均匀的分布在圆上，因此误差较大。

1. 加权平均法

平均值法求得的圆心位置会偏离数据点密集的一边，求出的半径也会偏小。加权平均法是将样本点距离其相邻两点的弧长与圆的整个周长的2倍的比值作为权数。样本点距离其相邻两点的弧长可用相邻两点的距离替代。

（三）最小二乘法

通过最小化误差的平方和找到一组数据的最佳函数匹配。可是最小二乘法拟合的平方项对滋扰数据很是敏感[1]。

（四） 霍夫变换法

将样本点为圆心，在参数平面画圆，许多点对应的圆会交于一点，该交点即可能为圆心。缺点计算量大，运算时占用内存大[2]。

（五）基于共形代数的圆拟合方法

将欧式空间嵌入到共形空间，将圆表示为向量形式，共形空间中点与圆的内积表示点与圆的位置关系，对具有噪声的样本点进行拟合实验[3]。

（六）基于聚类方法的圆拟合算法

该算法将边缘样本点映射到圆心附近，使用聚类算法进行分类滤噪处理并计算圆心，最后以此为基础计算半径。机器视觉进行圆拟合，首先采用工业相机对圆形特征进行拍照，图像在拍摄、传输和存储过程中都会不可避免地产生噪声，这将带来较大的拟合误差[4]。因此去除圆上其他轮廓干扰或对其局部失真进行纠正是一个十分重要的步骤。

1.3 本题的主要工作

基于该项目的目标是完成基于OPEN GL开发圆柱测量数据的显示功能模块；实现利用给定的截面计算圆柱度等数据功能。因此考虑时间因素和学习成本，确定选择利用最小二乘法拟合圆，经过多次迭代生成最终结果。通过C++编程，实现该项目。

2算法分析

有一系列的数据点｛, ｝，咱们晓得这些数据点类似的落在一个圆上，按照这些数据点计算出这个圆的参数这是一个非常重要的题目。圆拟合的方法有很多，本文选择的是基于最小二乘法。

2.1圆拟合算法分析

我们知道圆方程为

通常的最小二乘法拟合要求距离的平方和最小。也就是

最小，但是这个算起来非常的麻烦，也得不到解析解。因此我们退一步。

上面这个式子简单的多。我们定义一个辅助函数：

那么上述公式可以表示为：

按照最小二乘法的通常的步骤，可得f取极值时对应下面的条件。

*= 0*

*= 0*

*= 0*

先来化简

= 0

= = 0

我们知道半径R是不可能为0的。所以必然有：

同样

=

=

=

=

也就是下面两个等式：

这里设：

其中：

可以简单推导一下，就会发现：

这个变量替换修改了坐标远点的位置。而我们的等式与坐标原点的具体位置无关。所以必然成立。这两个式子展开可以写成这样：

进一步展开：

我们知道 所以上面两个式子是可以化简的。

为了简化式子，我们定义几个参数：

那么上面的式子可以写为：

+ =

+ =

至此，可以解出Uc和Vc了。

那么：

=

=

还剩下一个R没有求出来，也很简单。

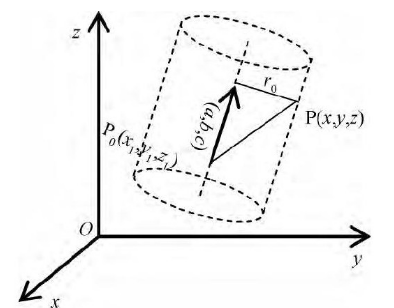
所以：

以上就是最小二乘法的数学原理推导。然而这种方法对误差符合正态分布的数据点很有效。但是当数据中碰到一些干扰点。这些干扰点多数时候是偏向某一个方向的。因此，本文采用的是根据数据点到圆的距离绝对值的和来确定圆的参数，也就是下面的数学公式：

使得f取得最小值的Xc、Yc和R就是最佳拟合参数。

2.2 圆柱拟合算法分析

已知圆柱面上的点到其轴线上的距离恒等于半径r0，如图所示



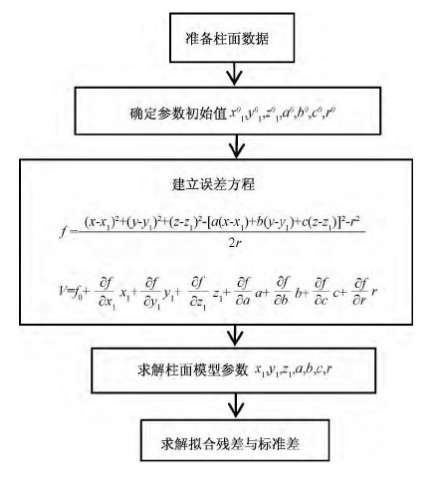
其中点P为圆柱上的任一点，P0为圆柱轴线上任一点，即：

=

根据以上公式就可以唯一确认一个圆柱。

2.2.1 圆柱面拟合方法原理

圆柱面的拟合方法主要包括两个步骤：第一确定初始值；第二建立误差方程求解析值。本文的流程图为：



第一：将圆柱面上一点的相邻若干点拟合成一个平面，求得这个平面的法向量即为点的单位法向量。假设旋转矩阵为R3x3,要使：

=

写成矩阵形式为：

=

L =

其中

X =

2.3 本章小结

本章主要说明该圆柱测量数据图形化显示系统中圆度拟合的算法分析，圆度拟合算法属于计算机图形学领域的算法分析。无论是在测量，还是在检验的过程中，我们都需要用到圆的拟合，圆拟合算法在工业领域应用的方方面面，我们往往是将一系列的点集生成一个一个的线段，之后依据线段之间的联系组合成为一个圆。针对误差过大的点，我们也应当可以清除数据，这些数据都属于冗余的，因此往往最后都需要做校验以及正确性检测。

3 网络通信

3.1 三菱PLC的分类及FX系列PLC的发展历史

三菱PLC包括一体式的和模块式的，有简单的控制系统，到高端的自动化解决平台。一体式的PLC数据集成式PLC, 包罗CPU板、I/O板，表现面板、内存块、电源等，这些组件组合成一个团体。三菱FX系列的PLC属于一体式的，这种PLC多用于中小型系统。其主要特点就是可以实现10点到384点的控制规模，它体积比较小，并且容易安装，性价比高，成本低廉。同时可以实现多种收集转换等多种特别功效，这些模块可以依照必然法则组合设置装备摆设。I/o点数可以达到4096点，最快的处理时间可以达到9ns,并且可以使用模拟量、网络等多种特殊功能模块。

70年代日本三菱公司开始研发PLC，之后第一台F系列投入实际应用，至今已经是高性能、高可靠型的最佳代表。FX系列包括多种。此中FX1S根基单位有继电器或晶体管输出，不克不及链接扩大模块和特别功能模块；FX1N可以毗连FX2N系列的特别功效模块，备有多种功效扩大板供利用；FX2N适用于从通俗顺控起头的普遍用处。FX3U属于第三代微型可编程控制器，具备高速、大容量、功效多的新机器。

Windows系统下使用的是GX Developer, 在设计、调试方面大大提高了编码的效率，允许以顺序功能图和结构化文本等语言进行编程。网络参数的设定和网络状态监视变得简单容易，并且同时可以在线监控程序、改变程序和数据以及强制I/O的ON/OFF等。可以实现离线编程、模拟测试和仿真等。在整个过程中，PLC通过ON/OFF位软元件或给字软元件赋值来实现真是动作情况。

3.2 FX系列 PLC各个功能模块分析

此中输入输出内置了电源回路和输入输出，用于扩大输入输出的产物，可以在毗连厥后面的装备供电，输入输出扩大模块内置了输入、输出，用于扩大输入输出。可以连接在基本单元上使用。特殊功能模块是一种可以是实现多种I/O以外其他功能的模块。包罗模拟量节制、定位节制、网络系统、高速计数等功能。而像特殊适配器类似与特殊功能模块，也包含了模拟量和定位，通讯等多种功能。其他外部设备也是PLC系统不分割的一部分，它包括通过专用电缆，进行程序编制、对系统作一些设定、监控的手持编程器或编程软件、显示模块的监控设备和存储器以及用于安装特殊适配器用的连接器转换或实现通讯功能时使用。

如果CPU连接输入输出扩展单元，当上电时，会自动输入输出标号进行标号分配，因此，无需通过参数制定输入输出编号。输入输出编号以8进制进行分配，扩展的输入输出扩展单元/模块，接着前面的输入编号和输出编号，末尾从0开始分配。上电时，CPU会从离其最近的特殊功能模块开始，依次对特俗功能单元分配单元号，而在输入输出扩展单元/模块没有这个单元号。根基单位的输入为内部供电的漏型输入和源型输入的通用产物。经由过程选择可以将根基单位的全部输入设置为漏型输入，或源型输入，可是不能混合利用，各根基单位和输入输出扩大单位，可以别离选择漏型输入、或是源型输入。

PLC履行法式是经由过程轮回履行的挨次从而实现挨次节制。分为输入处置阶段，法式处置阶段、输出处置阶段。输入处置是在可编程控制器履行法式前，将可编程控制器中的全部状况读入。法式履行过程当中即便输入产生了转变，输入映像区也不会产生转变，在履行下一个轮回的输入处置时读取该转变。编程控制器根据内存中的内容，从输入的映像区和其他的映像区读出各软元件的状态，从最初依次开始运算，并将每次得出的结果写入到映像区。是以，各软元件的映像区跟着法式的履行慢慢转变其内容。全部指令履行竣事后，输出映像区中的状况会传送至输出锁存内存。

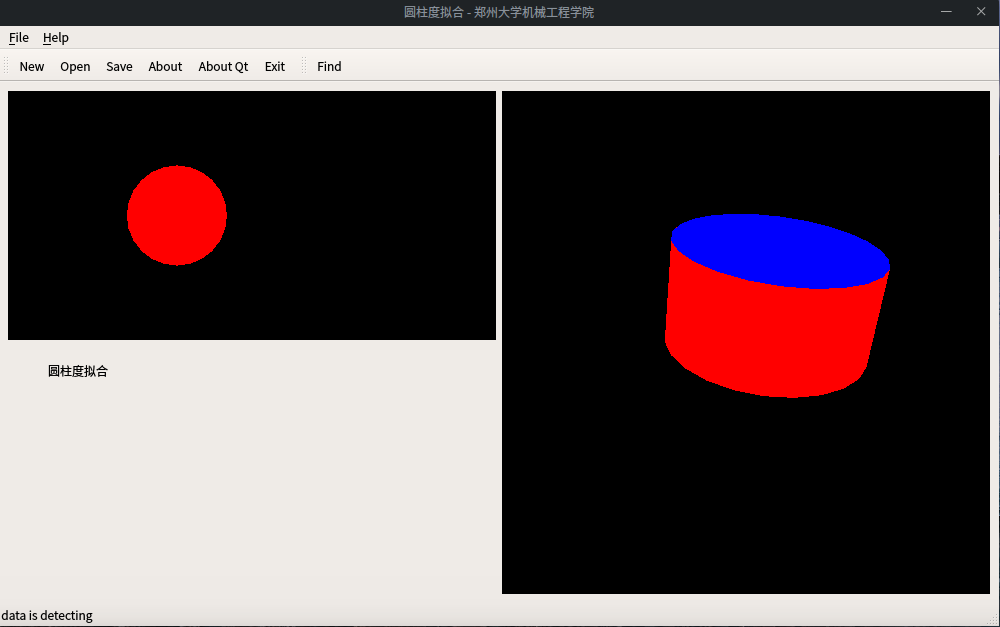
FX系列可编程控制器中，使用SFC图（顺序功能图)作为实现顺控，以一种便于理解的方式表现基于机械动作的各工序的作用和整个控制流程。在SFC法式中，状况S被作为一个节制工序，在此中对输入前提和输出节制的挨次举行编纂。工序推进后，前工序不再执行，因此可以通过各个工序的简单顺序来控制机械。状况S和驱动指令的行动，SFC法式中，用状况暗示机器运行的各个工序。当状态为ON时，在此连接的梯形图动作，当状态为OFF时，与此连接的内部梯形图不发生动作。当各个状态设置的条件(转移条件)被满足时，下一个状态为ON，此前如果状态为ON则为OFF。在状况的转移过程当中，仅仅在一个刹时两个状况会同时ON。转移前的状况在转移后的下一个运算周期OFF。不能重复使用同一个状态编号。

特殊功能模块是为了实现特殊功能如AD、DA转换，输入输出定位等的模块，其带有自有的CPU和处理电路，和基本单元进行通信。PLC是依托内部的位软元件和字软元件的利用来举行PLC的输出节制的。因此特殊功能模块是为了实现特殊功能如AD、DA转换，输入输出定位等的模块，其带有自有的CPU和处理电路，和基本单元进行通信。PLC是依靠内部的位软元件和字软元件的使用来进行PLC的输出控制的。因此，为了对外界的温度、电压、电流、流量等连续变化的的物理量进行控制，必须采用模拟量控制单元。特殊功能模块通过PLC的FROM/TO指令对特殊功能模块的缓冲存储器的数据进行信息交换。FROM指令用于把特殊功能模块的缓冲存储器的信息读书其当前值或者是状态信息。TO指令用于把数据及信息写入到特殊功能模块的缓冲存储器中。特殊功能模块内置了缓冲存储器，其作用就是用来与PLC基本单元进行交互，为了对外界的温度、电压、电流、流量等连续变化的的物理量进行控制，必须采用模拟量控制单元。特别功能模块内置了缓冲存储器，其感化便是用来与PLC根基单位举行交互，为了对外界的温度、电压、电流、流量等持续转变的的物理量举行节制，必需采取模拟量节制单位。FROM指令用于把特别功能模块的缓冲存储器的信息念书其当前值或者是状况信息。TO指令用于把数据及信息写入到特别功能模块的缓冲存储器中。特别功能模块内置了缓冲存储器，其感化便是用来与PLC根基单位举行交互。

3.3本章小节

整个系统构成包含三个方面，一是关于输入输出点数，二是关于特殊扩展设备，第三是关于消耗电流的计算，确认其是否可以扩展的输入输出点数。将残剩的电源容量作为外部负载的电源利用。残剩电源容量是不是许可毗连，特别适配器和特别功效单位。输入输出点数的合计便是计较根基单位、输入输出扩大单位/模块的输入输出点数，和特别功效单位/模块的输入输出占用点数，分歧型号的输入输出占用点数，可以经由过程特别功效单位/模块的台数 X 每台的点数 = 输入输出占用的点数。

4.系统架构

圆柱测量数据的图像化显示系统基于C++编程语言，在其上封装的框架Qt，利用面向对象编程在系统之上进行GUL编程。基于面向工具的编程思惟，一般为操纵类的担当和多态来举行编程

4.1 C++简介

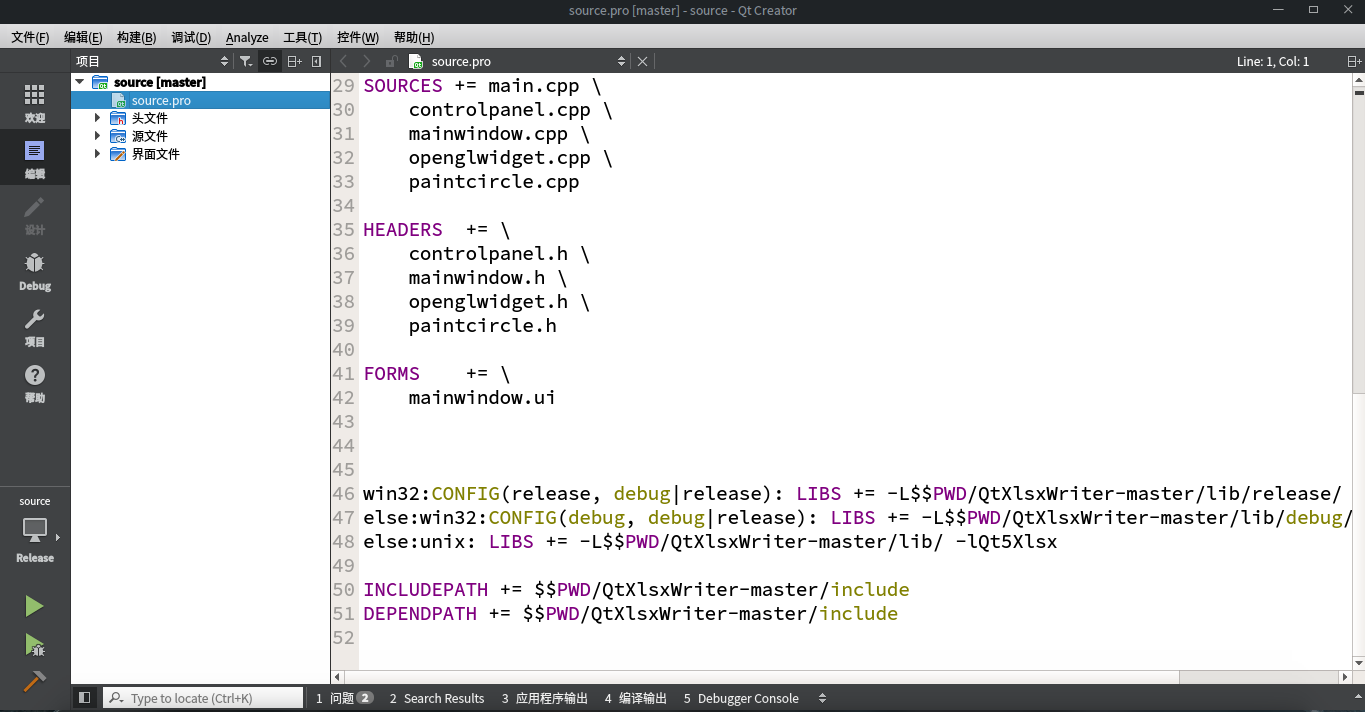
C++作为中级开发语言， 由Bjarne Stroustrup于1979年在贝尔实验室计划开辟的。C++扩大了C语言的集合，成为一门面向对象的语言。C++可以在Windows，Linux，MacOS上使用。C++是一种静态类型支持面向过程编程，面向对象编程以及泛型编程的编程语言。完全中和了高级开发语言和中级开发语言的各种功能。C++完全支持面向对象的程序设计，包括支持封装、抽象、继承和多态即面向对象的四大要素。C++包罗焦点说话，供给了全部的构件，包罗变量，数据类型等等、C++尺度库，供给了大量的函数，用于操纵文件、字符串等等、尺度模板库STL，供给了大量的方式，用于操纵数据结构。

4.2 环境设置

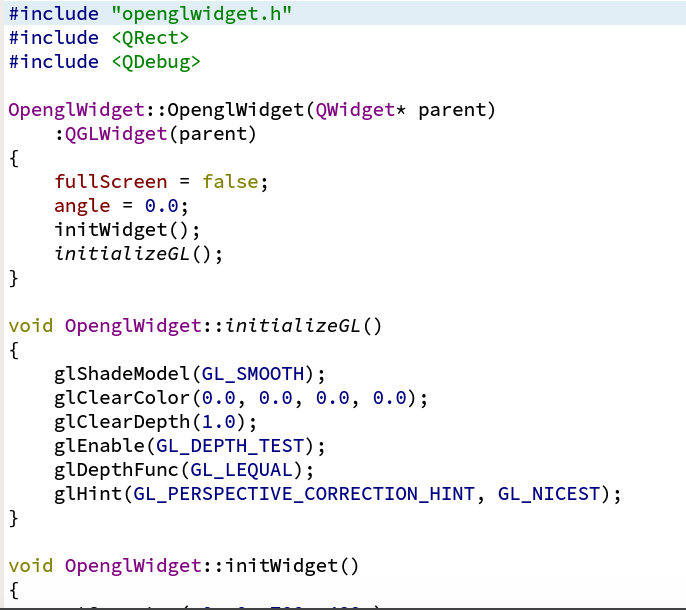
在本机的环境设置，C++执行环境必须包括文本编辑器和C++编译器。如果要编译程序，直接到保存的C++文件的目录下使用

g++ -o

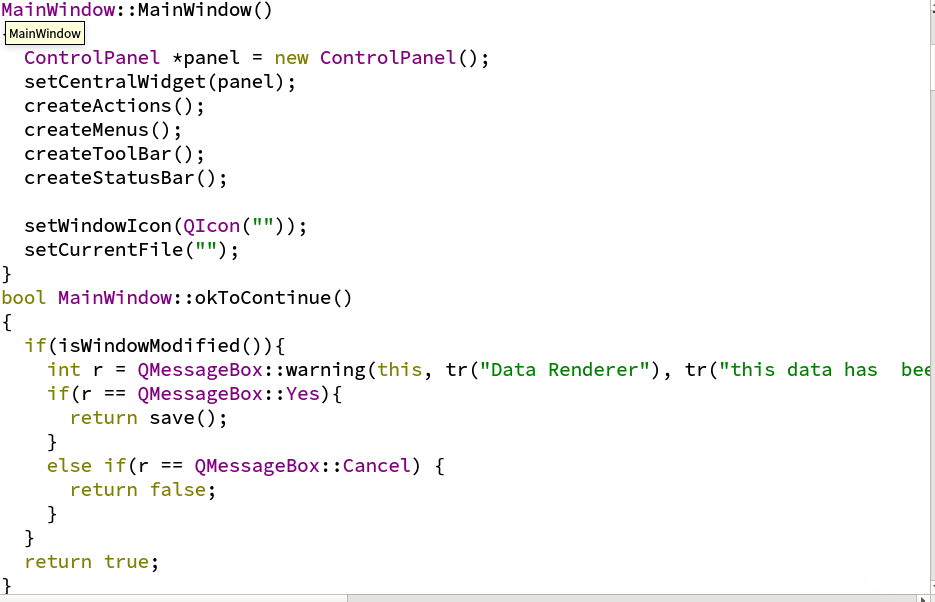
基本上，当我们使用-o选项时，就是要告诉编译器在文件交换中生成可执行代码。 如果你不这样做，它将被默认为a.out文件，但是这不是一个好的编程习惯。之后在Linux中运行C ++代码：编译完代码后，您将获得可执行文件。 您只需要按以下方式运行它：./a.out;



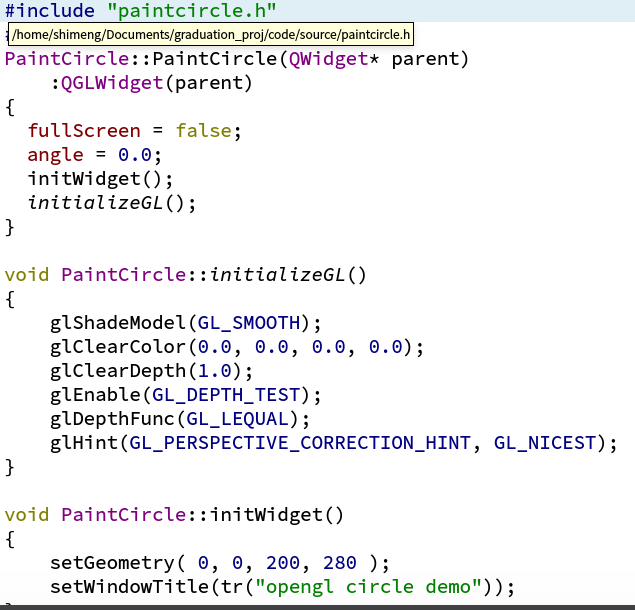
4.3 C++ 基础

因为功能性API看起来和感觉不同于更常见的面向对象API。功能性API的一个表现符号是核心抽象的存在和具有代数属性的一组方法。抽象部分当然是关于能够谈论只是必要的细节，而没有其他。代数部分是关于能够采取同一个抽象的一个或多个实例，并能够根据某些法则创建一个新实例。也就是说，将较小的部分合法地组成一个更大的东西，这是原始实例的同一抽象的一个实例。对OO程序员来说，组合本身并不是什么新东西。复合材料，装饰者设计模式都是关于将较小的部分放在更大的东西上。但是，他们错过了合法的部分。功能范例为您提供额外的护栏，以便您可以将一些理解良好的数学特性烘焙到抽象中。我认为，即使是基本的OO经验的大多数人都会“获得”抽象部分。但是，代数部分需要一些思考，一些培训，最重要的是一些制作精良，简单易懂的例子。说到示例，C ++已经将范围库作为技术规范（TS）采用。其核心是有合法的代数性质。从工业实施中理解功能原理是可能的，但更困难。 RxCpp是另一个例子。这两个库都使用运算符重载，我认为这是一种方便，而不是设计的核心部分。我今天的重点是可以理解。因此，我试图用一个简单的数据生成器例子来揭示合法代数api设计的艺术。可组合数据生成器和基于属性的测试 - 在C ++ 14中实现，用于测试通用序列化和反序列化库（RefleX）。 它提出了一种从单纯的随机数生成器和使用编译时元编程生成随机类型的数据生成器的系统技术。 这次演讲描述了monoids，functors，applicative和monads的规律，以及它们是如何在简单理解的数据生成器抽象中实现的。

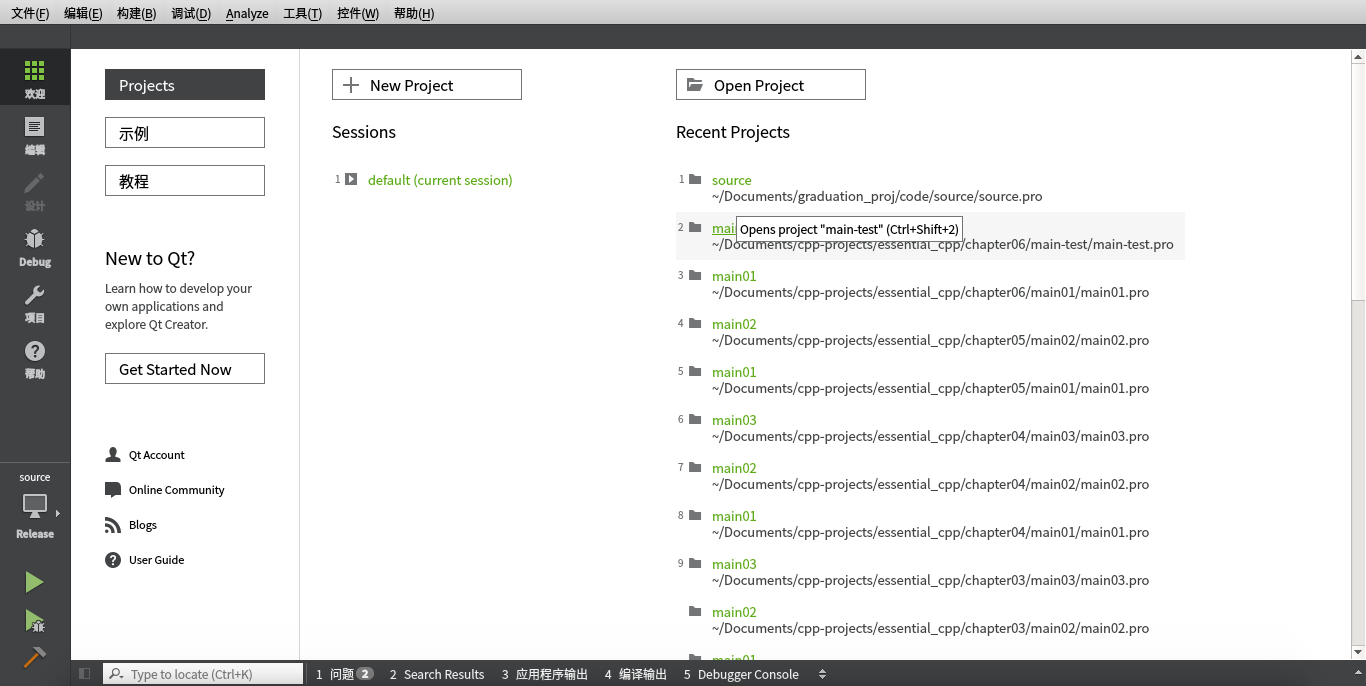
他的标准C ++库提供了一个条件变量的两个实现：std :: condition\_variable和std :: condition\_variable\_any。 这两个都是在<condition\_variable>库头中声明的。 在这两种情况下，他们都需要使用互斥锁来提供适当的同步; 前者仅限于使用std :: mutex，而后者可以处理任何符合互斥体样最低标准的任何内容，因此是\_any后缀。 由于std :: condition\_variable\_any更通用，因此在大小，性能或操作系统资源方面可能会产生额外成本，因此std :: condition\_variable应该是首选，除非需要额外的灵活性。好的，所以你决定为你的应用程序使用并发性。特别是，你决定使用多个线程。现在怎么办？你如何启动这些线程，你如何检查它们已经完成，以及如何监视它们？正如你所看到的，C ++标准库使大多数线程管理任务相对容易，几乎所有的东西都通过与给定线程关联的std :: thread对象进行管理。对于那些不太直截了当的任务，图书馆提供了灵活性，可以从基本构建模块中构建您所需要的内容。

每个C ++程序至少有一个线程，它由C ++运行时启动：运行main（）的线程。然后，您的程序可以启动具有另一个函数作为入口点的其他线程。这些线程然后并行运行，并与初始线程同时运行。正如程序从main（）返回时退出程序一样，当指定的入口点函数返回时，线程退出。正如你所看到的，如果你有一个线程的std :: thread对象，你可以等待它完成;但首先你必须开始它，所以让我们看看启动线程。线程是通过构建一个指定要在该线程上运行的任务的std :: thread对象来启动的。在最简单的情况下，这个任务只是一个简单的普通的无返回函数。这只是简单而已。 当然，你必须确保包含<thread>头文件，这样编译器才能看到std :: thread类的定义。 与许多C ++标准库一样，std :: thread可以与任何可调用类型一起工作，因此您可以使用函数调用操作符将类的实例传递给std :: thread构造函数：在这种情况下，提供的函数对象被复制到属于新创建的执行线程的存储中，并从那里调用。 因此，复制行为等同于原始行为是至关重要的，或者结果可能不符合预期。将函数对象传递给线程构造函数时要考虑的一件事是避免称为“C ++最令人头疼的解析”。如果传递临时变量而不是命名变量，则语法可以与函数声明相同 在这种情况下，编译器将其解释为这样，而不是对象定义。 例如，一种避免此问题的可调用对象是lambda表达式。 这是C ++ 11的一个新功能，它基本上允许您编写本地函数，可能捕获一些局部变量并避免传递额外参数。

一旦你开始了你的线程，你需要明确地决定是否等待它完成，或者让它独立运行（。如果你在std :: thread对象被销毁之前没有做出决定，那么你的程序被终止（std :: thread析构函数调用std :: terminate（））。因此，即使存在异常，确保线程正确连接或分离也是至关重要的。注意，你只需要在std :: thread对象被销毁之前做出这个决定 - 在你加入或分离它之前，线程本身可能已经完成了很长时间，并且如果你分离它，那么线程可能会在很久之后继续运行std :: thread对象被销毁。如果您不等待线程完成，那么您需要确保线程访问的数据在线程完成之前有效。这不是一个新问题 - 即使在单线程代码中，它在被销毁之后访问对象时的未定义行为 - 但线程的使用为遇到这种生存期问题提供了额外的机会。遇到这种问题的一种情况是，线程函数持有指向局部变量的指针或引用，并且函数退出时线程还没有完成。在这种情况下，当oops退出时，与my\_thread关联的新线程可能仍会运行，因为您明确地决定不通过调用detach（）等待它。如果线程仍在运行，则下一次对do\_something（i）的调用将访问已经被销毁的变量。这就像普通的单线程代码一样 - 允许一个指向局部变量的指针或引用超出函数出口永远不会是一个好主意 - 但是使用多线程代码更容易出错，因为它并不一定立即显而易见这发生了。

处理这种情况的一种常见方法是使线程函数独立并将数据复制到线程中，而不是共享数据。如果为线程函数使用可调用对象，则该对象本身将被复制到线程中，因此可以立即销毁原始对象。但是仍然需要警惕包含指针或引用的对象，例如列表2.1中的对象。尤其是，在函数中创建一个可访问该函数中的局部变量的线程是一个坏主意，除非线程保证在函数退出前完成。或者，您可以确保线程在通过加入线程退出函数之前完成执行。如果你需要等待一个线程完成，你可以通过在关联的std :: thread实例上调用join（）来完成。在调用my\_thread.join（）函数体的大括号之前将调用替换为my\_thread.detach（）将因此足以确保线程在退出函数之前完成并且因此在局部变量被销毁之前。在这种情况下，这意味着在单独的线程上运行该函数的意义不大，因为第一个线程在此期间不会做任何有用的事情，但在真实代码中，原始线程可能有工作要做，在等待完成所有这些工作之前，会启动几个线程来完成有用的工作。join（）是简单而蛮力的 - 无论你等待一个线程完成还是你没有完成。如果你需要更多细粒度的控制来等待线程，比如检查一个线程是否完成，或者只等待一段时间，那么你必须使用其他机制，比如条件变量和期货，我们我们将在第4章中看到。调用join（）的行为也清理了与该线程相关的任何存储，所以std :: thread对象不再与现在完成的线程相关联;它不与任何线程关联。这意味着你可以为给定的线程调用join（）一次;一旦你调用join（），std :: thread对象不再可以连接，joinable（）将返回false。如前所述，您需要确保在std :: thread对象被销毁之前调用了join（）或detach（）。如果你要分离一个线程，你通常可以在线程启动后立即调用detach（），所以这不成问题。但是如果你打算等待线程，你需要仔细挑选代码中你调用join（）的地方。这意味着如果在线程启动之后但在join（）调用之前抛出异常，join（）的调用可能会被跳过。为避免在抛出异常时终止您的应用程序，您因此需要决定在这种情况下做什么。一般来说，如果您打算在非常规情况下调用join（），则还需要在出现异常时调用join（）以避免意外的生命期问题。下一个清单显示了一些简单的代码。

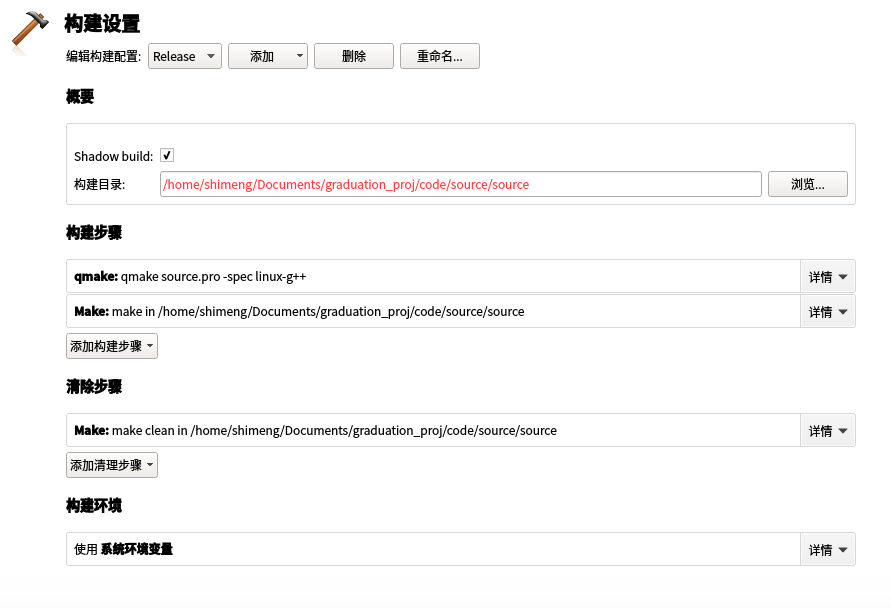
4.4 Qt框架



在通过QML编译缓存缩短了启动时间后，对象创建仍然是Qt Quick应用程序最大的启动时间消耗者之一。可以使用Loaders来避免加载当前不可见的应用程序部分，但是一些复杂度较高的场景在急需显示时需要花费相当多的时间进行初始化，特别是在嵌入式系统上。我们首先分析应用程序，以便首先解决低挂果实问题。这涉及到QML Profiler，还包括一个常规的C ++分析器（如macOS Instruments或Linux perf）以及配置Qt的“-release -force-debug-info”（不要传递-developer-build，因为它会改变QML的方式 缓存行为）。 这导致我们移除Qt Quick Particles，QQmlFileSelector，并且我们还将属性行为和状态转换为命令式动画。 这些变化已经通过减少场景中的QObject的数量并且通过阻止使用Qt Quick的一些不太可扩展的特性来显着提高性能。目前最重要的是，这允许我们将性能测量集中在应用程序代码上，确保我们的QML和C ++实现都使用相同的Qt Quick代码路径和总数的QObject。Qt Quick和QML在更新帧后的场景帧以生成平滑动画时非常高效。属性通过属性绑定根据动画一致且高效地更新，布局将只涉及必要的元素，并且场景图将尽可能多地重用尽可能多的OpenGL对象以在大多数时间渲染新帧，并在系统和系统中实现最佳状态更改图形内存。然而，要实现这种结构，需要在创建时进行一些运行时初始化，而像QML这样的大部分动态类型的语言并不能真正与C ++这样的语言竞争，该语言的设计使性能和编译时间优化优先于其他语言特性。除了必须创建相同的支持C ++ QObject和QQuickItem之外，QML还必须解析动态类型，创建数据结构以表示属性绑定，并通过Qt元对象类型系统访问任何本机成员。几乎所有的C ++在编译时都是等价的。QML可以选择以某种性能为代价提供简洁性和开发速度的原因之一是，它也被设计为与C ++代码良好地交互。例如，这是一个机会，Qt Quick Controls 2已经采取了减少编译和创建时间的方法，其增益与此处介绍的相当。 Qt Quick Controls 2比应用程序有一个优势：它是Qt的一部分，它可以访问诸如QQuickText，QQuickRectangle，QQuickFlickable和QQuickItemPrivate等私有符号。虽然我们的C ++实现是为测量这两个特定基准测试的潜在性能而定制的，但理想的方法是如果应用程序可以在QML中进行布局，但是处理来自C ++的事件和状态更改，而性能增益将非常有价值。这可以利用两种语言的优势，同时减少在运行时执行的动态类型属性绑定和JavaScript块的数量。

4.4.1 Qt环境配置

Qt Creator作为Qt官方的开发环境，对Qt各方面支持都非常完善，并且使用体验非常好。Qt Creator主界面分为6个模式：接待模式、编纂模式、计划模式、Debug模式、项目模式和帮忙模式，别离由左侧的六个图标举行模式的切换。刚进去表现的是接待界面，有一些入门教程、开辟的项目列表、Qt供给的示例法式，单击一个示例法式可以直接打开至这个项目。



4.4.2 Qt面向对象

当你将一个名字传递给QIcon :: fromTheme时，Qt需要在不同的文件夹中查找，以便知道哪个主题包含给定的图标以及哪个大小。 这可能意味着需要完成很多磁盘访问才能确定文件是否存在。 KMail等应用程序可以为其每个菜单操作提供数百个图标。QIcon :: fromTheme加载Freedesktop图标主题规范和图标命名规范指定的图标。它指定了Linux桌面使用的图标主题。总之，图标主题由包含PNG或SVG格式图标的许多文件夹组成。每个文件夹都包含特定大小和类型的图标。该类型可能是“mimetypes”，“actions”，“apps”之一。大小通常都是常见的大小。这代表了很多每个主题的目录。在我的电脑上，Gnome的默认主题有106个文件夹;氧气主题有166;微风有56; hicolor有483个。主题还可以指定一个或多个后备主题。无法在特定主题中找到的图标需要在后备主题中查找，直到找到图标。最后的手段后备主题总是“和谐”。图标名称可以包含破折号（ - ）。短划线用于分离特异性水平。例如，如果要查找的名称是“input-mouse-usb”，并且该图标在主题中不存在，则会查找“输入鼠标”，直到找到图标。由于上述原因，桌面环境一直在使用缓存来加速图标加载。在KDE4中，KDE的KIconLoader使用共享内存映射的pixmap缓存，其中加载的图标数据在进程间共享（KPixmapCache）。当应用程序请求给定大小的图标时，将查询缓存是否可以使用此大小的此图标。这在当时的KDE应用程序中取得了良好的效果。但是，这不能由纯粹的Qt应用程序使用。在运行KDE的Plasma桌面时，平台主题集成插件可以使用KIconLoader加载只能在Plasma上运行的图标。无论如何，这种缓存对于QIcon :: fromTheme来说效果不佳，因为我们需要知道QIcon是否为空，并且无论缓存如何都会强制进行查找。GTK +还可以使用缓存加速图标查找。他们有一个工具（gtk-update-icon-cache）为每个主题生成一个icon-theme.cache文件。该文件包含一个用于快速查找的散列表，以了解某个图标是否位于主题中，以及该文件的大小。图标缓存是系统范围的，通常由发行版的包管理器生成。

4.4.3.1 槽

当信号被发出时，会调用与之相连接的槽。槽是普通的C++函数，可以用普通的方式来调用。它唯一特殊的地方在于可以与信号相连接。槽的参数不能有默认值。同样，信号的参数也不能有默认值。在槽的参数中尽量不使用自定义的数据类型，因为这样将会使通用性降低。既然槽和普通的成员函数差不多，它们和普通成员函数一样有访问限制，根据槽的访问限制谁可以与它们相连接，能够分为以下三种情况。public slot: 任何信号都可以与之相连接。这在窗口部件编程中非常有用，用于创建一些对彼此一无所知的对象，只有通过信号和槽来交换信息。public slot就像是标准的铁路一样。protected slot: 只有该类及其子类所派生的对象的信号才可以与之相连接。这类槽的目的通常是为了类的完善，而不是类与外界的接口。private slot: 只有该类自己的信号才可以与之相连接[11]。当然还可以将定义为virtual，这将非常有用。信号和槽是相当高效的。当然，它们与“实时”的回调函数相比，在增加了灵活性的同时也损失了一些速度，正所谓有利必有弊，但是这种速度的损失相当微不足道。因此，信号槽机制具有的简便性和灵活性的特性，使用信号和槽是用户交互的必然选择。

4.4.3.2 信号

Verdigris是一个只有头文件的C ++库，可以让用户在不需要moc的情况下使用Qt。这个技巧用在W\_PROPERTY和W\_OBJECT\_IMPL宏中。 W\_PROPERTY的第一个参数是一个类型。 通常：W\_PROPERTY（QString，myProperty成员m\_myProperty）。但是当类型包含一个或多个逗号时会发生什么，如：W\_PROPERTY（QMap <QString，int>，myProperty MEMBER m\_myProperty）？ 这是无效的，宏扩展不考虑模板，因此第一个参数将取决于第一个逗号。 解决方法是将类型名称放在括号内。 新的问题是，我们如何忽略宏观实施中的括号。这个想法是有一个静态函数（让我们称之为w\_state），其返回值包含状态。 每个W\_INVOKABLE宏然后将扩展到该函数的新定义。 当然，它需要采取不同的论点，所以这只是声明一个新的重载。 我们通过使用从w\_number <N-1>继承的w\_number <N>类模板来完成此操作。 （这个想法基本上受到CopperSpice的cs\_counter的启发，其作者在CppCon 2015的演讲中描述）这工作得很好。在这个简化示例的结尾，我们的状态是一个包含＆Foo :: xx和＆Foo :: yy的std :: tuple，我们可以从中构建meta对象。 （在真正的实现中，状态更复杂一些，包含更多的数据）这是因为我们用来获取元组大小的对w\_state（w\_number <255>（））的调用引用了先前的w\_state声明。由于我们目前的功能还没有定义，所以最合适的功能是剩余的最高功能。请注意，我们必须在decltype和return之后重复相同的代码，并且我们不能使用返回类型推导，因为我们需要在类完全定义之前使用该函数。到现在为止还挺好。但是，当我使用类模板做这件事时，我已经达到了我认为是编译器错误的程度（例如，Foo将会是template <typename> Foo）。出于这个原因，我没有使用静态函数，而是在铜版中使用好友函数。原理完全一样。虽然它仍然是全局函数，但它并不是众所周知的，但可以在类中声明朋友函数。 （我也在Q\_ENUM的实现中使用了这个事实）只需要添加一个与当前类相关的类型作为参数。我使用指向类的指针而不是指针，因为在调用派生类时不需要潜在的指针转换。

4.3.4 Qt的组成部分

QtCore不但包含QString、QList和QFile等工具类，而且还包含QObject和QTimer等内核。因为QApplication类有refactored, 所以它能够在非GUI应用程序中。它将拆分为：QCoreApplication(在QtCore中)和QApplication(在QtGui中)。这种拆分将使用Qt开发服务器应用程序，无需连接所有多余的与GUI相关的代码，无需求与GUI相关的系统程序将要放到当前的目标机器中称为可能。qmake命令生成Makefile文件,qmake将默认连接到应用程序中依赖的Qtcore 和QtGui中。当想删除具有依赖关系的GUI时，可以在.profile文件中加入QT -= gui; 如果想使用其他的库文件，在.profile中加入 QT += network opengl; 如果要编译.ui文件，也需要在.pro文件中 加入 CONFIG += uic3。

QPushButton推动命令按钮是图形用户界面中最常用的一个窗口组件。它用来命令计算机去执行一些操作，或者回答一些问题。该按钮是一个矩形框，通常显示一个文本标签来描述它的操作。标签中有下划线的字母表明快捷键。QRadioButton组件为单选按钮提供了一个文本标签。QRadioButton是一个能够切换或者关的选择按钮。通常单选按钮是用于当前用户选择许多中的一个选项。同一时间，单选按钮组中仅仅只有一个按钮被选择；如果选择其他按钮，以前选择的按钮将要切换为关闭状态。当一个按钮切换开或关的时候，它就会发射toggled（）信号。如果任何时刻想触发一个行为来改变按钮的状态的时候，可以连接这个信号。否则，使用isChecked()来查看是否有其他按钮被选择。QCheckBox组件提供一个带文本标签的复选按钮，QCheckBox和QRadioButton的区别是在于对用户选择的限制。单选按钮定义了”多选一”的选择，而复选按钮定义了”多选多”的选择。QButtonGroup在视觉上能够把许多的复选按钮组合在一起，成为一个按钮组。无论复选按钮选中或者是未选中，它都会发射一个信号toggled()。在任何时刻，如果想要改变复选按钮的状态的时候，需要连接toggled()信号触发这个行为。除了常用的选中和未选中两个状态，QCheckBox还可以任意选择第三种状态来表明”没有改变”。当需要给用户一个选中或者未选中复选按钮的选择时，这是非常有用的。如果需要第三种状态，可以通过setTristate()来使它生效，利用checkState()函数来查询当前的状态。

标签组件QLabel提供了显示文本或者图像，没有用户交互的功能。因为标签的外观可以通过各种各样的方式来配置，而且可以制定焦点记忆键到另外的窗体上。QLabel是经常为交互式窗体制作标签。为了交互式窗体的使用。QLabel提供了添加助记符的有用机制，这种机制将设置键盘焦点到另外一个窗体上。 QLCDNumber窗口部件能够在LCD上面显示十进制、十六进制、八进制、二进制的数字。可以利用display()槽与数据源进行连接，能够重载五个任意类型的display()。虽然能够通过value()来检索数值，但是QLCDNumber对象的内容不能够被检索。如果您真正地需要文本，我们建议您使用信号-槽的机制来实现，可以将display()槽连接到其他槽中，并且将其存放在那里。

QSpinBox类提供了一个微调框的窗体。QSpinBox类是用来处理整数和设置离散集合的数值(例如：月份);利用QDoubleSpinBox可以处理浮点数值。绝大多数微调框是有方向的，但是QSpinBox也可以像使用一个圆形的微调框一样。例如：如果微调框的范围是0到99，并且微调框当前的数值为99，wrapping()函数设置为真，点击“向上”按钮，或者点击键盘向上的方向键，该微调框的数值就会变为0。如果想改变微调框具有这样圆形的功能，可以利用setWrapping()函数来实现。微调框有一个特殊的功能是：可以在需要实现数字的中添加任意的符号。例如：货币或者测量单位，可以参考setPrefix()和setSuffix()函数。如果您要获得微调框的文本信息，您可以使用text()函数(它包括所有prefix()和suffix())，或者使用cleantext()函数(它没有prefix()和suffix(),并且在开头和结尾没有空白)。每当微调框需要显示数值v,就要使用虚函数textFromValue()。该函数在默认的情况下，返回的是一个包含有参数v的字符串。如果需要将其输出，可以利用QLocale().toString(v)函数，将其标准的打印出来。

QLineEdit窗体组件是一个单行文本编辑框。单行文本编辑框允许用户通过有用的编辑函数来输入和编辑单行纯文本，该编辑函数包括撤消和重做、剪切和粘贴、拖动和放下。可以利用setText()或者insert()函数来改变文本，利用text()函数来获取文本编辑框的信息，也可以利用displayText()函数获取文本显示的信息。

QMenu类提供了一个菜单窗体组件，该组件用于菜单栏、上下文菜单和弹出式菜单。菜单窗体组件能够选择菜单，在菜单栏中可以有一个下拉菜单，或者有一个独立的上下文菜单。当用户点击下拉菜单的各自选项，或者使用指定的快捷键时候，就会显示菜单栏中的下拉菜单。我们可以使用QMenuBar::addMenu()函数将菜单添加到菜单栏中。上下文菜单通常是由某一个专门的键盘按键，或者通过使用点击鼠标右键来实现的。想要实现上下文菜单，就要执行popup()函数，或者exec()函数。菜单由一个选项列表组成，使用addAction()函数将选项添加到选项列表中。另外，选项也可以有文本标签，在文本标签的左边可以任意添加图标，也可以给选项添加快捷键，例如“Ctrl+X”。如果想将每个选项分隔开来，那应该如何去做呢？可以使用分割符。分隔符---显示一个子菜单的选项，并且可以在选项中执行某个操作。如果想插入一个分隔符，可以利用函数addSeparator()来实现。如果想添加子菜单，可以利用addMenu()函数来实现。当插入一个选项的时候，通常要指定接收者和槽。当选项发出信号triggered()的时候，要告诉接收者。

Qt部件不同于其他用户交互方式的GUI工具包。用户交互方式是所有GUI应用程序关心的问题。通过将某种用户事件(比如按下鼠标)与程序事件(比如退出程序)联系起来，使用户能够在图形界面中只使用鼠标来控制程序。而其他工具包是利用回调函数来进行用户交互的。所谓的回调是指：你自己定义一个函数，并告诉系统何时为何调用。你可以写一个特定数量和类型参数的函数，然后告诉系统何时使用，并传递给它所需的参数，系统就会调用你定义的函数，处理参数，并给你返回值。所谓的回调函数是指：按照一定的形式由开发人员定义并编写实现内容。使用回调函数，实际上就是在调用某个函数(通常是API函数)时，将自己的一个函数(也就是回调函数)的地址作为参数传递给那个函数。而那个函数在需要的时候，也就是某种事情发生的时候，利用传递的函数地址调用回调函数，这时开发人员可以利用这个机会在回调函数中处理消息或完成一定的操作。回调函数只能是全局函数，或者是静态函数，因为这个函数只是在类中使用，所以为了维护类的完整性，我们用类的静态成员函数来做回调函数。虽然回调函数能够实现用户之间的交互。但是，回调函数非常复杂，容易混淆，又难以理解(至少大部分编写Qt的工作人员或者程序员有过这样的想法)。因此，Qt的开发者使用另一种方法来完成这一工作。这种方法依赖于Qt特有的两个功能，信号和槽。使用这种新方法是非常简单的，只需要编写一行代码就能够将用户事件和程序事件连接起来。这种将用户事件连接到程序事件的方法要比回调函数，更加容易使用的两个重要原因是：1> 槽和信号是调用Qt库中的函数。2> 信号和槽不使用其他工具包。信号和槽用对象间的通信，该机制是Qt的一个中心特征，并且最能体现Qt与其他工具包它们之间有什么不同。在图形用户界面编程中，我们经常希望一个窗口部件的一个变化被告知给另外一个窗口部件。简单的说，我们希望任何一类的对象可以和其他对象进行通信。例如：我们正在解析一个XML文件，当我们遇到一个新的标签的时候，我们也许希望告知列表视图，我们正在用来表达XML文件的结构。比较老的工具包使用一种被称做回调的通信方式来实现同一目的。回调是指一个函数的指针，所以如果你希望一个处理函数告知你一些事件[12]，你可以把另一个函数(回调函数)的指针传递给处理函数。处理函数在适当的时候回调。回调有两个主要缺点：1> 它们不是类型安全的。我们从来都不能确定处理函数使用了正确的参数来调用回调。2> 回调和处理函数是非常强有力的联系在一起的，因为处理函数必须要知道调用哪个回调[13]。

5.辅助工具

在开发的过程中，并非是将直接在编辑器中输入所有代码，之后编译生成等步骤。同时我们还需要版本控制、单元测试等其他辅助工具，只有当这些部分完成后，一个完整的软件开发才算是完成了。

5.1 版本控制

那么何为版本控制呢？为什么我们的项目中一定要用到版本控制呢？其实版本控制就是它去记录了一个项目的任何一个部分的内容的变化。可以追踪每一次的变化，将来我们还是可以回到以前的某一时刻。在这次毕业设计中，对整个项目的代码论文等等都使用了版本控制。使用版本控制系统是一个非常正确的，因为它能增加我们的容错性，有了它我们可以在错误的情况下回滚。并且当我们想要看出每一次的更改后项目的变化，可以很容易的看出。

5.2 单元测试

单元测试对框架的计划请求很是高，数据与代码与界面要尽量分手，接口界说，输出与输出预期，代码笼盖度，单元测试是软件测试的根本，是以单元测试的结果会直接影响到软件的后期测试，终究在很大水平上影响到产物的质量。从如下几个方面就可以看出单元测试的重要性在何处。时候方面：若是当真的做好了单元测试，在系统集成联调时很是顺遂，因此会节俭良多时候，反之那些由于由于时候缘由不做单元测试或随意做做的则在集成时总会碰到那些本应该在单元测试就可以发明的题目，而这类题目在集成时碰到常常很难闪开发人员预料到[14]，末了在苦苦寻找中才发明这是个很初级的毛病而在懊悔本身时已华侈了良多时候，这类时候上的华侈一点都不值得，正所谓得不偿失。自动化测试系统承担多种的测试任务 ，系统的一些功能实现与待测软件相关 ，另外的一些功能具有通用性。为了维护测试系统的持续平稳运行 ，减少维护的成本投入 ，本文提倡采用待测任务与系统任务级别分离的测试策略 ，以满足不同的测试需求。为了尽可能的提高开发效率 ，在自动化测试系统结构上课采用测试管理软件 ，以减少通用操作的开发量，提高专有操作的开发量[15]。

6. 总结：

本文利用PLC与计算机通信，在上位机上开发一款基于圆柱测量数据显示系统。该系统通过数据采集、数据处理、数据计算、模型生成等一系列步骤，最终生成一个三维模型。在工业测量领域中，由于测量对象的千差万别，离散的测量数据很难进行描述分析，而三维显示系统可以让我们查看，研究，理解大量离散数据，从而可以有效的发现隐藏在数据内部的规律[16]。通过拟合构建一个三维模型来逼近原型。通过给定的一组点，构造一个曲线，曲线逼近构造一个曲面，使之逐渐逼近给定的数据点。本文采用最小二乘法，拟合出圆柱的半径，从而构建出圆柱的三维模型。在实际的拟合中，数据点不可避免的引入噪声点，因此本文同样对数据进行预处理，减少噪点对拟合的影响。

谢 辞

时间忽然穿梭而去，一恍惚间大学四年的生活已经即将走完全程，四年前还是懵懵懂懂的少年在大学四年里学习了自己的专业也找到了自己喜欢的方向。现在学到了很多，学习的生活是快乐的。但是随着毕业日子的到来，半年的毕业设计生活也快要结束了。从刚开始什么都不会，到现在即将完成。当没有开始做毕业设计的时候觉得毕业设计只是一次作业性质的任务，但是通过这半年做毕业设计的过程中发现自己以前的简单。毕业设计不仅仅是对我的前面三年所学知识的一种提升，也是对自己个人能力素养的一种巨大的提高。通过这次毕业设计使我明白了自己原来知识还比较欠缺。自己要学习的东西还太多，以前老是觉得自己担心不敢相信自己可以独立完成。通过这次毕业设计，我终于明白了独立学习就是坚持的过程，在未来的学习工作中，学习都时需要自己独立的查找资料，去学习，工作中出现的问题不会都是自己学过的，所以说这种独立的思想一定要加强。

刚开始选择课题时发现有些不知所错，因为并不是完全自己会的知识，经过指导老师分析和教导，对整个课题方案的大体思路指导让我明白了如何学习。我的课题主要是圆柱测量数据的检测系统，通过PLC与上位机通信，上位机展示PLC检测到的信息，并用OPenGL制作一个三维模型。在开发的过程中，我通过到图书馆、上网等多种方式查阅资料学习课题内容。通过对整个项目的开发，对自己的这大学四年既是在原有学习上的一次总结，也是一次突破。整体从零开始，将整个毕业设计中设计到的知识学习一遍，同时有突破了自己原有的知识体系，学习了更多编程的知识。

参考文献

[1] 刘珂,周富强,张广军. 半径约束最小二乘圆拟合方法及其误差分析[J]. 光电子·激光. 2006(05)

[2] 祁亚芳,杨明. 一种改进的随机Hough变换检测圆的方法[J]. 数学的实践与认识. 2014(17)

[3] 白志鹏,刘超,李茂宽. 基于共形几何代数的圆拟合方法实现[J]. 现代电子技术. 2009(16)

[4] 李渊,乔兵,陆宇平. 基于聚类方法的圆拟合算法[J]. 计算机应用. 2013(S2)

[5] 王春荣,蔡勇,蒋刚. 基于卡尔曼与改进SVM移动机器人定位研究[J]. 微型机与应用,2011,(4).doi:10.3969/j.issn.1674-7720.2011.04.026

[6] 程虹霞. 智能小区安防语音通信系统的设计与实现[D]. 四川大学 2006

[7] 程虹霞. 智能小区安防语音通信系统的设计与实现[D]. 四川大学 2006

[8] 刘礼建,张广明. 基于ZigBee无线技术的智能家居管理系统设计[J]. 计算机技术与发展,2011,(12).doi:10.3969/j.issn.1673-629X.2011.12.067.

[9]基于Qt的移动微博系统客户端设计与实现[D]. 武茜.北京邮电大学 2011

[10] 程虹霞. 智能小区安防语音通信系统的设计与实现[D]. 四川大学 2006

[11] 刘礼建,张广明. 基于ZigBee无线技术的智能家居管理系统设计[J]. 计算机技术与发展,2011,(12).doi:10.3969/j.issn.1673-629X.2011.12.067.

[12] 彭成章. 基于ARM Linux的嵌入式系统及其应用研究[D]. 东北大学,2004.

[13] 孔繁荣. 基于Qt/Embedded的嵌入式图形设计. 2005年全国单片机与嵌入式系统学术交流会,2005

[14] 张曲艳. .TaaS平台的测试任务分类子系统的设计和实现[D]. 北京大学, 2009

[15] 杨巍. 软件自动化测试系统的设计[J]. 科技传播,2014,(18):234-234,229

[16]李兰芳. 工业测量数据的三维模型显示[D]. 山东科技大学 2009