浙江大学

**科研和工程中的C++编程开发报告**

**李逸婷分报告**

目录

解决方案 1

设计思路 2

开发总结 3

心得体会 3

课程建议 3

# 解决方案

主要负责图像处理追踪方面以及一些类设计和实现。

**1.1图像读写、图像信息处理解决方案：**

1.首先是如何设计图像处理和追踪这一块的功能。在组长的建议下我们选择使用基于 ICCV 2011 的“Struck: Structured Output Tracking with Kernels” 一文的追踪算法。

2.有了算法后，思路就非常清晰了：

* 图像存放读写设计成一个类，作为最基本的数据类存放在Utility目录下，以供其他部分分工的同学也能使用这个数据类。
* 图像信息处理按照追踪算法的要求，进行类设计和整合算法代码。具体追踪算法的实现全都统一放置于Algorithms目录下。

**1.2 Common中的ICommand和INotification类的设计：**

参照MVVM框架下的，model向view层传数据，INotification就是要向view层提供是否已经完成数据传输的，view向model通过ICommand来处理命令后传入后台。以这个设计框架为原则来设计ICommand和INotification。

**1.3Model层中的TrafficSignModel类的设计：**

参照MVVM框架下的，model层要负责所有的数据读取。以这个核心的理念来设计TrafficSignModel类。

## 设计思路

**2.1图像信息处理追踪类设计与实现**

主要就是依照ICCV 2011 的“Struck: Structured Output Tracking with Kernels” 一文的追踪算法进行类的设计和实现，组长在阅读了文献之后将Stuck程序的源代码，整合进我们的程序中，完成了总体图像信息处理追踪功能的实现，也就是第二轮迭代的目标。

除了算法中的图像处理追踪类以外，我们还需要一些公用的数据类。在阅读了OpenCV的文档之后，学习了一些OpenCV中基本的图像相关的数据结构和操作，写一个存放录像每一帧图像的公用数据类以及能够完成一些读取基本操作，以方便同学使用。

**2.2 Common中的类**

2.2.1 ICommand类

在老师和组长的讲解和实际的操作过程中，更深入地理解了MVVM这个框架的理念，也参照了很快完成第一轮迭代的小组的代码，理解了为什么需要ICommand这个类来帮助完成MVVM的实现。

ICommand类主要是要完成两个部分的内容，一个是对于命令本身的实现，一个是对于参数的处理。并且他应该是一个纯虚类，只有被继承变成一个具体的命令才会有实际的命令意义和参数。

2.2.2 INotification类

当Model层传给View层数据之后，View层中的显示的内容就会发生改变。那么什么时候改变呢，就需要有一个通知，当数据传输成功之后给予View层一个信号，这就是INotification所承担的功能。

# 开发总结

## 心得体会

一开始上来课程与我想象地非常不一样，也没有想到是直接在十天这么短的时间内实现一个项目。果然验证了DDL是第一生产力，在短短这么时间内，我们小组一起学习了QT，学习了Boost，学习了OpenCV，学习了Cmake，都是非常有用的一些东西。同时也了解了一个软件开发的模式和过程，MVVM模式很好地解除了前端和后端的耦合度，让我们可以更好地分别负责一个部分的开发，并且方便了调试。小组一起配置环境，一起讨论各部分的具体实现方法，不断地推进这个项目，总而言之是很不错的一次体验。

## 课程建议

1.感觉上课刚上来直接做一个项目非常的不习惯，当天就要组好队并且想出一个好的项目，压力还是蛮大的，特别是我们还是一群菜鸟，很多事情都不太懂，希望老师能够多给一些时间进行设想和组队。

2.希望老师上课能多讲一些关于开发方面的内容，或者直接以一个简单的项目为例直接为我们介绍一下整个开发周期里面的很多事情，可能会让我们更有概念，比较方便我们上手进行一个项目。

3.关于cmake能够再详细一些，能多讲一些库连接啊之类的怎么写。

2014年7月23日