浙江大学

**科研和工程中的C++编程开发报告**

**徐可添 分报告**

目录

[解决方案 1](#_Toc393911960)

[应用程序框架 1](#_Toc393911961)

[跟踪算法 2](#_Toc393911962)

[开发总结 2](#_Toc393911963)

[心得体会 2](#_Toc393911964)

[课程建议 3](#_Toc393911965)

# 解决方案

在本次开发中，我承担的任务是应用程序的框架和跟踪算法这两个部分。我将分别叙述对于这两个任务，我所采用的解决方案。

## 应用程序框架

本次开发基于MVVM框架，即将整个应用程序分为Model层、View Model层和View层三大部分。

Model层中包含了所有的数据对象和业务逻辑，数据对象包括视频（储存待跟踪的视频）和图像（当前帧），具体而言，分别采用了OpenCV中的VideoCapture类和Mat类来存储。业务逻辑即具体的追踪算法，由Struck算法修改而来。

View层负责处理UI，是用Qt来实现的，在控件接收到信号后调用View Model层暴露给它的相应ICommand，同时接收View Model传递给它的Event，进行相应的处理。具体来说，当UI中的Video按钮被按下后，会调用OpenCommand，相应地，View Model层会调用Model层中打开视频的方法（在这个方法中，视频被打开后，第一帧会被读入到一个ImageData类中），然后View Model将ImageData中的数据转换为Qt的可绘制对象，并用Event通知View层显示该图像（即暴露给View层的图像数据）。当UI中的Run按钮被按下后，View层会产生一个周期为100ms的单触发定时器，定时器所触发的事件会调用RunCommand，相应地，View Model层调用Model层中的TrackSign方法，对每一帧中的交通标志进行追踪。每次追踪结束后，View Model层会将追踪结果转化为Qt的可绘制对象，并用Event通知View层刷新界面。

*具体设计图可参见总结报告。*

## 跟踪算法

“Online Object Tracking: A Benchmark,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013 这篇论文从不同角度对于目前流行的跟踪算法进行了全面的比较。根据这篇论文中的比较结果，我选择了各方面表现俱佳的Struck算法。根据Struck算法的论文（“Struck: Structured Output Tracking with Kernels”, in ICCV 2011），该算法自适应的过程分为两个步骤：取样和学习。只要取样时增加和原来跟踪框大小不同的样本，即可以实现动态跟踪的效果。并且，我们经过实际实验，发现选用Raw Gaussian特征时跟踪效果最佳。由于跟踪交通标志的任务较为简单，基本上不存在物体遮挡、形变等，为了提高跟踪速度，在不损失跟踪质量的情况下，经过实验，可以将支持向量的个数减少至3个。

最终的跟踪效果非常好，在被实验的视频中，我们选取了6个不同的交通标志进行跟踪，均为出现跟丢的情况，且跟踪框能随着汽车的行进智能地动态改变大小。

# 开发总结

## 心得体会

这次短学期的课程体验和大一相比截然不同。在如此短的时间内要完成这样一个课题起初让我感觉压力巨大，MVVM设计框架原来更是听都没有听过。不过最终我们还是做到了，我想这本身就是一个胜利了！

在短短十多天里，和组员一起学习、开发、调试、再学习、再开发、再调适的过程给我带来的收获是巨大的。所谓“纸上得来终觉浅，绝知此事要躬行”，在这样一段学习和实践融合的日子里，一方面我对Qt、对OpenCV、对Boost、对CMake有了一个接触和体验；另一方面我也领会到了MVVM设计框架的精神。当然，更重要的是，这段经历让我意识到了，越是觉得困难的事就越要勇敢去尝试，等到做完了，你才会发现，其实所谓的难事也不过如此。

最后感谢袁老师的悉心指导！感谢组员们辛勤的付出，和相互陪伴的这十几个日日夜夜。

## 课程建议

由于这个课程中接触到的一些概念和大一大二学习的课程跨度较大，希望袁老师可以事先在课程网上提供一些教程、资料等供同学参考、预习，以便更快上手。

2014年7月23日