浙江大学

**科研和工程中的C++编程开发报告**

**徐超颖 分报告**

目录

[解决方案 1](#__RefHeading__3019_173396680)

[设计思路 2](#__RefHeading__3041_173396680)

[错误及修正 2](#__RefHeading__3023_173396680)

[开发总结 2](#__RefHeading__3025_173396680)

[心得体会 2](#__RefHeading__3027_173396680)

[课程建议 3](#__RefHeading__3029_173396680)

解决方案

在本次开发中，我承担的任务是视频文件的读取、测试。

**设计思路**

要求写一个VideoData类，这个类主要功能是视频的读取存储和视频帧画面的提取输出。由于提前了解到opencv库本身带有对视频进行读取功能的类和函数，所以前期着重将opencv的视频读取这部分的文档看了一遍，确认了VideoCapture类、Mat类的相关函数的使用方法，这里我们主要运用了VideoCapture中的open，set，read等函数对视频文件进行操作。

视频读取类的主要声明如下图：

|  |
| --- |
| VideoData |
| cv::VideoCapture cap; //用于获取视频对象，以及对视频处理等功能。  cv::Mat frame; //用于储存视频的某一帧的图像。  int currentFrame, totFrame; //当前帧数和视频总帧数。 |
| void OpenVideo(const std::string &); //利用VideoCapture的open()成员函数打开指定目录的视频文件。  cv::Mat \*GetFrame(); //获取当前帧画面并输出。  cv::Mat \*GetFrame(int); //获取指定帧画面并输出。 |

错误及修正

视频读取部分在一开始的窗口测试中没有查出明显的错误，但是一旦进入视频播放状态，会出现视频播放卡的现象，经过一系列的debug我们发现了原因：在视频读取的时候每一帧都调用VideoCapture的set函数对读取的帧进行了设置，而这种设置会明显导致程序效率降低，于是出现画面卡顿情况。事实上在VideoCapture中读取视频的画面是连续的，自动跳入下一帧，因此没有必要每次都对帧数进行设置，删去此句后程序运行正常。

*//cap.set(CV\_CAP\_PROP\_POS\_FRAMES, currentFrame++);*

*cap>>frame;*

开发总结

心得体会

这是第一次较为完整地完成一个工程课题，过程中学习到很多之前都没听说或是接触过的东西，比如处理图像和视频的openCV库，还有MVVM这种设计模式。这种新的设计模式极大的降低了耦合度，让每个人的开发都能同步进行。通过几次迭代的方式，保证了每次迭代的正确率。

这次工程，整个团队工作让我体会到工程上程序设计的感觉，收获很多。 另外还要感谢所有组员的付出和老师的指导。

课程建议

感觉袁老师可以再多增加一些小组之间的交流，比如小组进度的课堂展示等等，一来督促课题的完成，二来也方便其他小组从中汲取经验、发现自己的不足。

2014年7月23日