**多媒体信息处理 课程作业**

Z08030884 俞哲峰

目 录

[作业名称 1](#_Toc383696697)

[“帧间差分法”简述 1](#_Toc383696698)

[JMF简述 2](#_Toc383696699)

[程序原理简述 2](#_Toc383696700)

[程序UML类图 2](#_Toc383696701)

[拾遗与展望 4](#_Toc383696702)

[1. 程序实现方式及语言选择 5](#_Toc383696703)

[2. 数据源的选择 5](#_Toc383696704)

[3. 对现有程序的优化方案 5](#_Toc383696705)

[4. 尝试其他算法实现 5](#_Toc383696706)

# 作业名称

基于帧间差分法的运动目标检测（应用JMF技术的JAVA程序实现）

# “帧间差分法”简述

帧间差分法是一种通过对视频图像序列中相邻两帧作差分运算来获得运动目标轮廓的方法。当场景中出现物体运动时，帧与帧之间会出现较为明显的差别，两帧相减，得到两帧图像高度差的绝对值，判断它是否大于阈值来分析视频图像序列的运动特性，确定图像序列中有无物体运动。

# JMF简述

Java媒体框架（JMF）是JAVA的一个类包，目前版本2.1.1，提供了先进的媒体处理能力，从而扩展了JAVA平台的功能。这些功能包括：媒体捕获、压缩、流转、回放，以及对各种主流媒体格式和编码的支持。

# 程序原理简述

程序的核心代码在MotionDetectionEffect.java中，源代码中已给出了较为详尽的注释。

此检测引擎主要分为两大步。

第一步：

输入的图象与上一帧的图像进行比较，如果某个像素的变动超过了设定的阈值，则在此像素的位置上标记一个黑-白-红的图标。

于是在第一步中，黑-白-红图标被标记在发生变动的位置，并形成多个集合。

第二步：

清除过小的集合（但仍然以黑-白-红图标显示），只保留那些大的集合，并且标记为红色；

在这个过程中，对大的集合的数量保持跟踪；

如果大集合的数量超过设定的blob\_threshold值，则认为输入帧相对于前一帧发生了相当程度的运动。

具体实现详见代码。

通过TestMotionDetection.java的main函数启动swing程序。

# 程序UML类图

结合程序，给出以下类图：





# 拾遗与展望

由于时间仓促以及个人技术的局限，在本次作业中存在不少遗憾，以下列举一部分，期望在之后的学习中进一步完善。

## 1. 程序实现方式及语言选择

毕业后一直从事JAVA相关的研发工作，对C/C++语言及OpenCV库了解不够，所以本次作业还是选择比较擅长的JAVA实现。

必须承认在图像处理的性能和成熟度方面，JAVA是不如C/C++的，所以在今后的学习中，希望能够运用OpenCV实现相应的算法。

另外，也可以尝试利用JNI技术实现JAVA与OpenCV相结合的运动目标检测程序。将核心的图像处理逻辑交给OpenCV实现，然后将其嵌入到前端表现层的JAVA/SWING框架中（JNI是JAVA与其他编程语言的集成编程接口，技术上应该没有问题）。

## 2. 数据源的选择

最初的设想是能够直接调用本机的摄像头，实时获取摄像头中的视频图像并进行处理。但实际编程中一直未能调用成功（因为涉及操作系统级的调用，确有一些不可控因素，如操作系统位数、防火墙等等），所以无奈采用了在程序中直接指定某个具体视频文件的方式（指定文件路径），虽然对具体算法的实现没有影响，但演示效果大打折扣，此处希望在今后的学习中完善。

## 3. 对现有程序的优化方案

从提高算法性能和效率的角度考虑，想到有如下优化方案可以尝试：

首先缩减图像，将图像转换成二维整型数组（int[][] array）（目前程序中是二维字节数组byte[][]），然后之后所有的运算都围绕整型数组展开，相信能大幅提高效率。

另外，对帧的差分比较也不一定要每一帧都一一比对，可以每5帧比对一下，效率也会更高些。

## 4. 尝试其他算法实现

目前检测运动目标的方法主要有三类：光流法、帧间养分法和背景差分法。希望在之后的学习中能够进一步应用另两种方法进行实现。

最后希望老师尽量让我通过这门课，我是09年入学的，如果再不完成学业，恐怕学制就要到期了，会比较麻烦，还请老师通融下。万分惶恐与感谢！