# 基于视频信号的音频恢复 设计报告

1. 成员及分工：

李步宇：

张熠：

夏江月：

1. 文件清单：
2. 基本需求部分：
3. 原理：

概括来说，任务主要分为两方面：一是从视频中提取振动波形，二是从波形中提取单双频以及通过一定的方法处理还原音乐。

对于振动波形提取，我们尝试过optical flow, template matching, complex steerable pyramid 和 Gabor transformation等多种方法。经比较最终选择了效果最好（提取的单双频峰值最明显）的Gabor transformation，恰好这种方法的运行效率也很高。

记视频为V(x,y,t)，用Gabor基底做变换之后，我们得到变换域后的图像：

http://latex.codecogs.com/gif.latex?%5Cdpi%7B80%7D%20A%28x%2Cy%2Ct%29exp%28j%5Cvarphi%28x%2Cy%2Ct%29%29

之后的步骤和Davis等人的paper(The Visual Microphone)类似。

由于Gabor变换本质是加高斯窗的傅里叶变换，所以它仍有傅里叶的性质：图像域的平移在变换域表现为相位的改变，振动信号可表示为：

http://latex.codecogs.com/gif.latex?%5Cdpi%7B80%7D%20s%28t%29%3D%5Csum_%7Bx%2Cy%7D%20A%28x%2Cy%2Ct%29%5E2%28%5Cvarphi%28x%2Cy%2Ct%29%20-%20%5Cvarphi%28x%2Cy%2Ct_0%29%29

我们没有像Davis一样用complex steerable pyramid得到好多组(scale, orientation)的信号再时移组合。主要是因为实验中我们发现pyramid方法中往往只有一两个(scale, orientation)对应的信号是有效的（变换后有正确且明的显峰值），把所有scale、orientation的信号Φi时移到相关最大位置加和往往在弱化信噪比（许多scale-orientation pair对应的变换图像得出的信号无意义）。后来我们发现选定合适的Gabor基底做一个变换得到的信号效果要好很多。

具体来说，我们选的了带直流补偿的Gabor基底，表达式如下：

http://latex.codecogs.com/gif.latex?%5Cdpi%7B80%7D%20%24%20G%28x%2Cy%2CS%2CF%2CW%2CP%29%3Dk%5Ctimes%20Gaussian%28x%2Cy%2CS%29%20%5Ctimes%20%28Sinusoid%28x%2Cy%2CF%2CW%2CP%29-%20DC%28F%2CS%2CP%29%29%24

其中：

http://latex.codecogs.com/gif.latex?%5Cdpi%7B80%7D%20Gaussian%28x%2Cy%2CS%29%3Dexp%28-%5Cpi%20S%5E2%28x%5E2&plus;y%5E2%29%29

http://latex.codecogs.com/gif.latex?%5Cdpi%7B80%7D%20Sinusoid%28x%2Cy%2CF%2CW%2CP%29%3Dexp%28j%282%5Cpi%20F%20%28x%5Ccos%28W%29&plus;y%5Csin%28W%29%29&plus;P%29%29

http://latex.codecogs.com/gif.latex?%5Cdpi%7B80%7D%20DC%28F%2CS%2CP%29%3Dexp%28-%5Cpi%28%5Cfrac%7BF%7D%7BS%7D%29%5E2&plus;jP%29

值得一提的是，P并不影响相位差，所以我们简化了参数个数。具体Gabor参数设定与滤波器生成及见gaborFilter.m，变换及波形导出见getGaborVib.m。

在得到振动波形后，我们做了除趋势(见detrend.m)、幅值规约、零均值化。对相应的时间区间做傅里叶变换寻找峰值就可以得出单频双频。下面来讲乐音部分的主要处理方法：

1. 实现：
2. 性能分析：
3. 问题与讨论：
4. 代码调用方法：
5. 提高需求部分：
6. 提升运行速度的追求

除了一些代码细节的优化，我们主要研究了如何提高Gabor变换的速度。在参数已优化的情况下，Gabor变换速度极大地决定于输入图片的大小。于是我们比较了截取不同大小的图片区域对结果的影响，选取的是频率最难提取的第一个视频的第二个单频音的时间段。

以下为图片区域为完整图片（1080\*1920）、1024\*1024、512\*512、256\*256：





用时分别为155.024532秒、92.550371秒、37.764515 秒、27.876644 秒