清华大学本科生综合论文训练中期报告

MVC格式3D视频并行实时解码

本科生：　　卿培

学号：　　2006011291

指导老师：　　孙立峰

任务概述

- 基于现有的MVC Decoder进行性能优化
- 基于NVIDIA 3D Vision眼镜实现一个有3D效果的播放器

由于我们的3D项目需要做一个成套的系统，所以必须有一个能够展示解码结果的播放器。而现有的播放器，比如VLC，对于3D视频没有合适的接口可供使用。所以我的毕设任务又加了一个播放器的设计。

任务目标

两路标清视频CPU实时解码	✓四核✍双核
利用3D Vision硬件播放两路标清3D视频	✓
八路标清视频GPU加速实时解码	✍

任务目标可以归纳成三个。
@click两路标清实时解码在实验用的四核CPU上已经做到了，双核上还在做进一步优化
@click播放也已经完成了
@click八路的实施解码优化方案已经确定，接下来就是具体的逐个函数转移到CUDA平台的工作

3D播放器设计

- 3D Vision原理
  - 快门式眼镜
  - 120Hz显示器

NVIDIA 3D Vision眼睛与我们在电影院观看3D电影时使用的偏振式的眼镜不同，它是快门式的。左右眼两个镜片分别内置了一个类似于相机快门的装置，可以遮挡住视线。
120Hz刷新率的显示器每秒交替显示60张左眼可见的图像和60张右眼可见的图片，眼镜通过红外线同步装置发射的同步信号来控制两个快门，保证双眼看到对其渲染的不同画面。
3D效果的合成依靠的是我们万能的大脑~

3D播放器设计

- 设计思路
- 查询NVAPI
  - 没有API!
- 以2xFPS播放+Enable3D
  - 程序运行没有得到预期效果
- 借助Direct3D
  - 成功!

设计的总体思路就是先实现一张静态3D图的显示，然后由此扩展到每秒n张图的显示，就实现了视频的播放。

@click

因为要调用NVIDIA的硬件，我第一个想到的就是去查NVAPI，其中的确有Stereoscopic3D子项，但是我只看到了一些状态查询、保存当前图像之类的函数，却没有生成画面用于显示的函数。NVIDIA对于注册开发人员还有一份更完整的API文档，可惜我的申请没有得到回应。

@click

第一个途径失败后，我想借助NVAPI中提到的一个打开3D功能的函数来实现。希望我在以2xFPS显示图片序列前调用该函数，NV的驱动能够自动识别我的目的，用3D的方式显示。不过这个方法依然没有成功。

经过两周不断地google、浏览开发论坛的帖子之后，我在mtbs3d.com的论坛上找到一个成功的例子，该贴描述了一个借助D3D来渲染3D画面的方式，给了几个关键的设置说明，跟帖中也有表示用该方法成功的，于是我就尝试了这个方法，终于成功。

3D播放器设计

D3DXLoadSurfaceFromFile

D3DXLoadSurfaceFromFile

#define NVSTEREO_IMAGE_SIGNATURE 0x4433564e //NV3D

该方法简单说明如下， @click建立一个用于渲染的表面，D3D中为一个IDirect3DSurface9。
这个表面的宽度为2*imgWidth。
之后以左上角为原点load左眼图像，@click
将原点右移imgWidth个像素，再load右眼图像@click。
这个表面的高度为imgHeight+1。此为第一个magic number……

在渲染前还要写一个LPNVSTEREOIMAGEHEADER，设置宽度为2*imgWidth，高度为imgHeight，色彩深度为32bit，@click最神奇的是还要设一个signature。这个magic number我在其他地方没有看到过……

在调用Direct3D的renderer渲染之后，我终于看到了3D效果的静态图。
@end

3D播放器设计

显示效果就是这样，直接看显示器会看到重影。透过眼睛看到的就是3D的场景。

3D播放器设计

- Frame rate cap
  - Accurate timer
    - QueryPerformanceCounter & QueryPerformanceFrequency
- Fn=(tn-t0)*FPS
- Frame skip

@click在显示了静态图之后，我开始实现以一定帧率来显示图片。

@click这需要一个精确的计时器，getTicker函数是ms级的，比较粗糙。@click我使用的是Windows API提供的QueryPerformanceCounter与QueryPerformanceFrequency两个函数配合，可以做到us级的计时，可以满足一般视频的播放要求了。

@click在播放第0帧之前，记下当前时间t0，在此后的某个时刻，获取当前时间tn，(tn-t0)*FPS就是该时刻应该显示的帧的序号。

@click视频播放还有个跳帧的要求，当显示速度跟不上视频的帧率时，应该跳过一定数量的帧来保证视频的帧与时间轴仍旧是对应上的。我在上述计算下一个渲染对象的时候，就隐含了跳过一些帧的机制。

MVC Decoder性能优化

- 难点
  - 参考软件完全不考虑速度
  - 解码顺序存在依赖关系
- Intel VTune分析

难点
对于现有的MVC Decoder工程，我用Intel VTune进行了性能分析。

MVC Decoder性能优化

Module (54)	Function (54)	Calls (54)	Self Time (54)	Total Time (54)
MVCDecoder.exe - Total		20522256	7630830	
MVCDecoder.exe	Filter	1255176	831463	1328146
MVCDecoder.exe	CheckMvDataB	1079079	525795	856735
MVCDecoder.exe	idct4x4_c	957728	523955	717134
MVCDecoder.exe	macroblockInterDecode16x16_y	38536	333542	1029723
MVCDecoder.exe	isCoded	2303280	287344	287344
MVCDecoder.exe	GetVerFilterStrength	634144	279889	863825
MVCDecoder.exe	GetHorFilterStrength	634144	277636	847286
MVCDecoder.exe	iquant4x4_c	957728	250708	250708
MVCDecoder.exe	ClipMinMax	1676759	201899	201899
MVCDecoder.exe	macroblockGetPred_axb	71697	200470	1009252
MVCDecoder.exe	macroblockInterDecode_uv	38536	192958	623700
MVCDecoder.exe	macroblockPredGetDataUV	143394	181045	328516
MVCDecoder.exe	FilterMB	40800	176696	3553207
MVCDecoder.exe	macroblockPredGetDataY	71697	169106	312709
MVCDecoder.exe	BitstreamSkip	776341	165050	165050

直接参照JMVC代码　　ffmpeg有x86优化的汇编代码

耗时前15位的函数如表中所示。其中大致分为两类，一是被调用次数极多的，二是函数单次运算很复杂的。

@click

胡伟栋告诉我说，Filter相关的函数（表中蓝色底纹标出）都是直接参考JMVC的代码而来，其中工程上的考虑很多，并不是追求速度的，优化的空间较大。

我还在阅读ffmpeg的源代码，其中有些函数是经过x86汇编优化的，#click比如图中紫色底纹标出的两个。汇编优化的代码据说最多能够达到70%的提速。

优化方案

重写代码

- Filter()
- CheckMvDataB()
- GetVerFilterStrength()
- GetHorFilterStrength()
- FilterMB()

汇编优化

- idct4x4_c()
- iquant4x4_c()
- unscan_zig_4x4()
- idct2x2dc_c()
- iquant2x2dc_c()
- unscan_zig_2x2()
- idct8x8()
- iquant8x8_c

CUDA并行

- macroblockInterDecode16x16_y()
- macroblockGetPred_axb()
- macroblockInterDecode_uv
- FilterMB()
- macroblockGetHalfPel()

优化方案分为三个部分。

任务计划

- 用ffmpeg中汇编优化的函数替代现有工程的对应函数
  - 5月14之前
- 将调用次数极多的简单函数交给GeForce显卡运行
  - 5月28之前
- 论文撰写
  - 5月28起两周

接下来的任务计划如下：
一是用ffmpeg中的汇编优化过的函数替换我们解码器工程中的对应函数，这其中主要是参数、数据结构的匹配工作。

同时将调用次数极多的函数交给CUDA平台的显卡来运行，这主要是对可并行计算任务的抽取、分配工作。
由于有些函数既可以汇编优化，又可以交给显卡并行优化，所以这样的函数在两种优化都实现后会选择更快的一种作为最终的实现。

最后是论文的撰写。

Q&A

谢谢!

卿培

FIT 1-512

edwardtoday@gmail.com