



jaccen的专栏

目录视图

摘要视图

订阅

个人资料



jaccen



访问：68644次

积分：1345

等级：

排名：千里之外


原创：47篇

转载：95篇

译文：3篇

评论：11条

文章搜索



- 文章分类
- 设计模式 (0)

杂记 (20)

图像处理 (6)

Linux (1)

java (2)

windows编程 (6)

机器视觉 (10)

算法学习 (1)

3D (7)

Swift (1)

OpenGL ES (23)

3DEngine (6)

Cocosd2D (1)

C++ (8)

Unity3D (0)

DeepLearning (6)

CUDA (2)

Vulkan (2)

VR (1)

GPU编程 (15)

Android (3)

【有奖征文】Bluemix云上实践征集喽~

Python 创意编程活动

CSDN日报20170511 ——《开发人员的必备技能》

OpenCL编程:图像卷积

2016-05-13 17:35587人阅读评论(0)收藏举报

分类：GPU编程 (14)

图像卷积，就是对图像所有像素进行一些特定的运算处理。这里涉及两个问题，一是读取图片文件信息，二是作何种卷积运算。第一个问题可在《freeimage存取图片数据》里找到答案。第二个问题可以baidu卷积算法。

卷积是对每个像素都进行相同的处理。以前我们是用C P U来串行处理。现在我们可以利用O P E N C L进行并行处理（多核C P U和G P U）。

我们的例子很简单，是一个对图片进行低通滤波处理。卷积大小是5×5，为了方便，对边缘像素不做处理（反正也就2个像素，肉眼看不出）。

low.cl源码

```
1  __kernel
2  void
3      low(__global int*
4          A,
5          __global
6          int*
7          B,
8          __global
9          int*
10         C,
11         int
12         sum,
13         int
14         img_width,
15         int
16         kernel_width)
17 {
18     //获取索引号，这里是二维的，所以可以取两个
19     //否则另一个永远是0
20     int
```

SLAM (2)

文章存档

2017年05月 (1)

2017年04月 (3)

2017年03月 (1)

2017年02月 (3)

2017年01月 (2)

展开

阅读排行

错误信息: 未能从程序集“ (3339)

Vulkan学习资料汇总 (持 (2521)

用INNER JOIN语法联接: (1654)

个人认为目前最完善的 (1634)

本地文件同步——C#源代 (1609)

Vulkan学习--基于Andrioc (1483)

摄像机标定总结 (1319)

Microsoft DirectX SDK+\ (1176)

vs2010 c# activex 开发 (1081)

itextSharp研究心得 (995)

评论排行

摄像机标定 (3)

FBO进行多纹理拼接 (2)

Android Studio 进阶详细 (2)

QRCode支持中文 (1)

3D Engine 开发 (1)

Android Studio 配置以及 (1)

OpengLES2.0 游戏开发- (1)

视觉slam学习资料 (0)

个人认为目前最完善的 (0)

30种图像动画特效算法 (0)

推荐文章

* 程序员4月书讯：Angular来了！

* 程序员要拥抱变化，聊聊Android即将支持的Java 8

* 彻底弄懂prepack与webpack的关系

* 用 TensorFlow 做个聊天机器人

* 分布式机器学习的集群方案介绍之HPC实现

* Android 音频系统：从AudioTrack 到 AudioFlinger

最新评论

OpengLES2.0 游戏开发-上卷 源码 weixin_37649048: 该链接已经失效 可以重新发一份吗

FBO进行多纹理拼接 maaooo: 大神，代码嘞

Android Studio 进阶详细配置 qq_31591635: 忘记勾选 Java 所以选项了

Android Studio 进阶详细配置 qq_31591635: 亲: switch 语句我按照你上面写的为什么不可以生成啊

Android Studio 配置以及JNI使用 jaccen: Program: \$JDKPath\$bin\javah.exe Parameters: -clas...

```
18 col = get_global_id(0);
19     int
20 row = get_global_id(1);
21
22     int
23 stx = (kernel_width - kernel_width%2)/2;
24     int
25 sty = stx;
26
27     int
28 nx,ny;
29     int
30 totalR=0;
31     int
32 totalG=0;
33     int
34 totalB=0;
35     int
36 nid = 0;
37
38     totalR=0;totalG=0;totalB=0;
39     nid=0;
40
41     if(col<=2
42 || row<=2 || col>=img_width-2 || row>=img_width-2)
43     {
44         B[row*img_width*3+col*3+0]
45 = 0;
46         B[row*img_width*3+col*3+1]
47 = 0;
48         B[row*img_width*3+col*3+2]
49 = 0;
50
51         return;
52     }
53
54     for(ny=row-sty;ny<=row+sty;ny++)
55     {
56         for(nx=col-stx;nx<=col+stx;nx++)
57         {
58             totalR
59 += C[nid] * A[ny*img_width*3+nx*3+0];
60             totalG
61 += C[nid] * A[ny*img_width*3+nx*3+1];
62             totalB
63 += C[nid] * A[ny*img_width*3+nx*3+2];
64         }
```

关闭

FBO进行多纹理拼接

jaccen: glReadPixels 是从Framebuffer读取数据，不是Texture,读取Texture...

3D Engine 开发

jaccen: 原计划边上传边开发，由于时间精力问题，等待测试完再开发 g i t ~ 欢迎有兴趣的沟通交流

QRCode支持中文

geckojule: "项目下建立一个新文件夹，名字是qrcode_data，然后再拷贝那些资源文件到此目录。然后设置这..."

```
                                nid++;

                                }

                                }

                                B[row*img_width*3+col*3+0]
                                = min(255,totalR/sum);

                                B[row*img_width*3+col*3+1]
                                = min(255,totalG/sum);

                                B[row*img_width*3+col*3+2]
                                = min(255,totalB/sum);

                                }
```

main.cpp源码

```
1  #include
2    <iostream>
3
4  #include
5    <stdio.h>
6
7  #include
8    <string.h>
9
10 #include
11    <string>
12
13 #include
14    <conio.h>
15
16 #include
17    <math.h>//数学库
18
19 #include
20    <CL/cl.h>//包含CL的头文件
21
22 //调用freeimage
23
24 #include
25    <freeimage.h>
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

关闭

```
28  int
    kernel_y =5;
29
30  static
31  int
    buf_A[dim_x*dim_y*3];
32
33  static
34  int
    buf_B[dim_x*dim_y*3];
35
36  static
37  int
    buf_C[] = {
38      1,1,1,1,1,
39      1,4,4,4,1,
40      1,4,12,4,1,
41      1,4,4,4,1,
42      1,1,1,1,1
43  };
44
45  //加载图片
46  //以RGBA格式存储图片
47  static
48  bool
    LoadImg(const
49  char*
    fname)
50  {
51
52      //初始化FreeImage
53      FreeImage_Initialise(TRUE);
54
55      //定义图片格式为未知
56      FREE_IMAGE_FORMAT
    fif = FIF_UNKNOWN;
57
58      //获取图片格式
59      fif
    = FreeImage_GetFileType(fname,0);
60
61
62      //根据获取格式读取图片数据
63      FIBITMAP*
    bitmap = FreeImage_Load(fif,fname,0);
64
65      if(!bitmap)
66      {
67          printf("load
68      error!
```

关闭

```
69     ");
70         return
71     false;
72     }
73
74     int
75     x,y;
76     RGBQUAD
77     m_rgb;
78
79     //获取图片长宽
80     int
81     width = (int)FreeImage_GetWidth(bitmap);
82     int
83     height = (int)FreeImage_GetHeight(bitmap);
84
85     //获取图片数据
86     //按RGBA格式保存到数组中
87     for(y=0;y<height;y++)
88     {
89         for(x=0;x<width;x++)
90         {
91             //获取像素值
92             FreeImage_GetPixelColor(bitmap,x,y,&m_rgb);
93
94             //将RGB值存入数组
95             buf_A[y*width*3+x*3+2]
96             = m_rgb.rgbRed;
97             buf_A[y*width*3+x*3+1]
98             = m_rgb.rgbGreen;
99             buf_A[y*width*3+x*3+0]
100            = m_rgb.rgbBlue;
101
102        }
103    }
104
105    FreeImage_Unload(bitmap);
106
107    return
108    true;
109 }
```

//保存图片

static

关闭

```
110 bool
SaveImg()
111 {
112     //初始化FreeImage
113     FreeImage_Initialise(TRUE);
114
115     FIBITMAP*
116     bitmap =FreeImage_Allocate(dim_x,dim_y,32,8,8,8);
117
118     int
119     m,n;
120
121     for(n=0;n<dim_y;n++)
122     {
123         BYTE
124         *bits =FreeImage_GetScanLine(bitmap,n);
125
126         for(m=0;m<dim_x;m++)
127         {
128             bits[0]
129             = buf_B[dim_x*3*n+m*3+0];
130             bits[1]
131             = buf_B[dim_x*3*n+m*3+1];
132             bits[2]
133             = buf_B[dim_x*3*n+m*3+2];
134             bits[3]
135             = 255;
136             bits+=4;
137         }
138     }
139
140     //保存图片为PNG格式
141     if(false
142     ==FreeImage_Save(FIF_PNG, bitmap,"low.png",
143     PNG_DEFAULT))
144     {
145         printf("save
146         image error
147         ");
148     }
149
150     FreeImage_Unload(bitmap);
151
152     return
153     true;
154 }
```

关闭

```
151
152 //从外部文件获取cl内核代码
153 bool
154 GetFileData(const
155 char*
156   fname,string& str)
157 {
158     FILE*
159     fp = fopen(fname,"r");
160     if(fp==NULL)
161     {
162         printf("no
163         found file
164         ");
165         return
166         false;
167     }
168     while(feof(fp)==0)
169     {
170         str
171         += fgetc(fp);
172     }
173     return
174     true;
175 }
176
177 int
178 main()
179 {
180     if(LoadImg("bk.png")==false)
181     {
182         printf("error
183         load bk.png!
184         ");
185         return
186         0;
187     }
188     //先读外部CL核心代码，如果失败则退出。
189     //代码存buf_code里面
190     string
191     code_file;
```

关闭

```
192     if(false
193 == GetFileData("low.cl",code_file))
194     {
195         printf("Open
196 low.cl error
197 ");
198         return
199 0;
200     }
201
202     char*
203     buf_code = new
204 char[code_file.size()];
205     strcpy(buf_code,code_file.c_str());
206     buf_code[code_file.size()-1]
207 = NULL;
208
209     //声明CL所需变量。
210     cl_device_id
211     device;
212     cl_platform_id
213     platform_id = NULL;
214     cl_context
215     context;
216     cl_command_queue
217     cmdQueue;
218     cl_mem
219     bufferA,bufferB,bufferC;
220     cl_program
221     program;
222     cl_kernel
223     kernel = NULL;
224
225     //我们使用的是二维向量
226     //设定向量大小(维数)
227     size_t
228     globalWorkSize[2];
229     globalWorkSize[0]
230     = dim_x;
231     globalWorkSize[1]
232     = dim_y;
233
234     cl_int
235     err;
236
237     /*
238     定义输入变量和输出变量，并设定初值
239 */
```

关闭


```
233     */
234
235     size_t
236     datasize = sizeof(int)
237               * dim_x * dim_y * 3;
238
239     size_t
240     kernelsize = sizeof(int)*kernel_x*kernel_y;
241
242     int
243     n=0;
244
245     int
246     sum=0;
247
248     //计算卷积核元素之和
249     for(n=0;n<25;n++)
250     {
251         sum
252         += buf_C[n];
253     }
254
255     //step
256     1:初始化OpenCL
257
258     err
259     = clGetPlatformIDs(1,&platform_id,NULL);
260
261     if(err!=CL_SUCCESS)
262     {
263         cout<<"clGetPlatformIDs
264         error:"<<err<<endl;
265
266         return
267         0;
268     }
269
270     //这次我们只用CPU来进行并行运算，当然你也可以该成GPU
271     clGetDeviceIDs(platform_id,CL_DEVICE_TYPE_CPU,1,&device,NL
272
273     //step
274     2:创建上下文
275
276     context
277     = clCreateContext(NULL,1,&device,NULL,NULL,NULL);
278
279     //step
280     3:创建命令队列
281
282     cmdQueue
283     = clCreateCommandQueue(context,device,0,NULL);
284
285     //step
```

关闭

274

4:创建数据缓冲区

```
bufferA
= clCreateBuffer(context,
                  CL_MEM_READ_ONLY,
                  datasize,NULL,NULL);

bufferB
= clCreateBuffer(context,
                  CL_MEM_WRITE_ONLY,
                  datasize,NULL,NULL);

bufferC
= clCreateBuffer(context,
                  CL_MEM_READ_ONLY,
                  kernelsize,NULL,NULL);
```

//step
5:将数据上传到缓冲区

```
clEnqueueWriteBuffer(cmdQueue,
                     bufferA,CL_FALSE,
                     0,datasize,
                     buf_A,0,
                     NULL,NULL);

clEnqueueWriteBuffer(cmdQueue,
                     bufferC,CL_FALSE,
                     0,kernelsize,
                     buf_C,0,
                     NULL,NULL);
```

//step
6:加载编译代码,创建内核调用函数

```
program
= clCreateProgramWithSource(context,1,
                           (const
char**)&buf_code,
                           NULL,NULL):
```

```
clBuildProgram(program,1,&device,NULL,NULL,NULL);
```

```
kernel
= clCreateKernel(program,"low",NULL);
```

//step
7:设置参数,执行内核

关闭

```
        clSetKernelArg(kernel,0,sizeof(cl_mem),&bufferA);
        clSetKernelArg(kernel,1,sizeof(cl_mem),&bufferB);
        clSetKernelArg(kernel,2,sizeof(cl_mem),&bufferC);
        clSetKernelArg(kernel,3,sizeof(cl_int),&sum); //卷积元素之和
        clSetKernelArg(kernel,4,sizeof(cl_int),&dim_x);
        //图片宽度
        clSetKernelArg(kernel,5,sizeof(cl_int),&kernel_x); //卷积核大小

        //注意这里第三个参数已经改成2，表示二维数据。
        clEnqueueNDRangeKernel(cmdQueue, kernel,
                                2, NULL,
                                globalWorkSize,
                                NULL, 0, NULL, NULL);

        //step
        8:取回计算结果
        clEnqueueReadBuffer(cmdQueue, bufferB, CL_TRUE, 0,
                             datasize, buf_B, 0, NULL, NULL);

        SaveImg();

        //释放所有调用和内存

        clReleaseKernel(kernel);
        clReleaseProgram(program);
        clReleaseCommandQueue(cmdQueue);
        clReleaseMemObject(bufferA);
        clReleaseMemObject(bufferB);
        clReleaseContext(context);

        delete
        buf_code;

        return
        0;
    }
}
```

源图

OpenCL编程(9):图像卷积 _重明鸟网站_www.cmnssoft.com

处理后图片（黑色是边缘没有处理部分）

OpenCL编程(9):图像卷积 _重明鸟网站_www.cmnssoft.com

关闭

http://www.cmnsoft.com/article.php?id=39

顶

0

踩

0

上一篇

函数执行时间测试

下一篇

Android Studio 配置以及JNI使用

我的同类文章

GPU编程 (14)

•

OPENCL简介

2016-05-10

阅读 143

•

性能优化案例NBody

2016-05-10

阅读 319

•

GPU线程及调度

2016-05-10

阅读 585

•

OpenCL buffer使用及两个...

2016-05-10

阅读 480

•

OpenCL概述 续篇

2016-05-10

阅读 119

•

OpenCL扩展

2016-05-10

阅读 204

•

性能优化

2016-05-10

阅读 100

•

GPU memory 结构

2016-05-10

阅读 175

•

GPU架构

2016-05-10

阅读 100

•

OpenCL概述 续篇(Introduct...

2016-05-10

阅读 100

更多文章

猜你在找

- 顾荣：开源大数据存储系统Alluxio（原Tachyon）的
2.6 内核I2C驱动框架
Docker与容器服务扩展机制 - 存储
Ceph—分布式存储系统的另一个选择
3.4.2内核下的I2C驱动
- 图像处理 - 线性滤波 - 1 基础相关算子卷积算子边缘效
遥感图像处理之空间域增强&卷积滤波&形态学滤波
使用 matlab 数字图像处理九 去卷积deconvolution
opencv基本操作 图像的卷积
利用fft2计算二维卷积 Matlab常用图像操作



查看评论


暂无评论

发表评论

用户名：

haijunz

评论内容：



提交

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题

Hadoop

AWS

移动游戏

Java

Android

iOS

Swift

智能硬件

Docker

OpenStack

VPN

Spark

ERP

IE10

Eclipse

CRM

JavaScript

数据库

Ubuntu

NFC

WAP

jQuery


关闭

BI	HTML5	Spring	Apache	.NET	API	HTML	SDK	IIS	Fedora	XML	LBS	Unity
Splashtop	UML	components	Windows Mobile	Rails	QEMU	KDE	Cassandra	CloudStack				
FTC	coremail	OPhone	CouchBase	云计算	iOS6	Rackspace	Web App	SpringSide	Maemo			
Compuware	大数据	aptech	Perl	Tornado	Ruby	Hibernate	ThinkPHP	HBase	Pure	Solr		
Angular	Cloud Foundry	Redis	Scala	Django	Bootstrap							

公司简介 | 招贤纳士 | 广告服务 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 |

江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2017, CSDN.NET, All Rights Reserved 

关闭