半闲居士

SLAM拾萃(3):siftGPU

前言

本周博客我们给大家介绍一下SiftGPU。由于特征匹配是SLAM中非常耗时间的一步,许多人都想把它的时间降至最短,因此目前ORB成了非常受欢迎的特征。而 老牌SIFT,则一直给人一种"很严谨很精确,但计算非常慢"的印象。在一个普通的PC上,计算一个640×480的图中的SIFT大约需要几百毫秒左右。如果特征都要 算300ms, 加上别的ICP什么的,一个SLAM就只能做成两帧左右的速度了,这是很令人失望的。而ORB, FAST之类的特征, 由于计算速度较快, 在SLAM这种实时

班级

那么、今天我们来说一个GPU版本的SIFT。它是由Wu Changchang同学写的。它能够明显地提升你的程序提取SIFT的速度。同时,它的代码大部分是基于 OpenGL的、即使在没有英伟达显卡的机器上也能运行起来。但另一方面,出于某种(历史或人为的)原因、SiftGPU的代码配置起来并不很容易(特别是在Linux下。 似乎SiftGPU作者是在win下开发的),代码新人可能会觉得比较困难。现在我们带着大家实践一下SiftGPU,我会给出一个例程供大家测试。

首先、说说我的运行配置。我用的机器是Thinkpad T450, Intel+Nvidia GetForce 940m显卡。但我个人只用Intel卡, 所以我就不编译Cuda了。各位有上 好补补的同学也可以接个Cuda下未能,可能会提高一点速度(但我不保证),我使用的操作系统是Ubuntu 14.04. OpenCV3.1版本。所以我假设你OpenCV已经 装好啦!(所以C++编评器总有的吧!) 不过openCv是不是3.1版本是没关系的、程序在2.X版本上也是能正常运行的。



小萝卜:师兄你这真是宅男标配啊!你到底是在讲配置环境还是在秀桌面啊!

下载SiftGPU与依赖库

iftGPU主页:http://www.cs.unc.edu/~ccwu/siftgpu/

请找到"SiftGPU-V400"那个下载链接, 保存到你的电脑上。然后解压缩, 进入压缩后的文件夹。假定你也在用Ubuntu, 那么你现在的目录应该是 nloads/SiftGPU/。注意,为了和我保持一致,请你暂时不要下线github上面那个版本,那个与它稍有不同。如果你就是喜欢github,可以把这个编译好 再考虑用github版本。

现在我们来安装依赖项。首先,确保你机器上有OpenGL,请安装以下几项工具

1 sudo apt-get install libgl1-mesa-dev libglu1-mesa-dev freeglut3-dev

然后,要安装glew1.5.1以上版本。据我个人经验,最好是去下载glew网站的版本。

glew的网址:http://glew.sourceforge.net/

请下载那个1.13.0版本, zip文件或tgz均可。下载到本地并解压, 然后进入该文件夹。我的在~/Downloads/glew-1.13.0

glew是用makefile直接编译的, 不用cmake。所以我们直接敲

2 sudo make install

即可。很快它就编译好了。

注意看make install输出的信息。它默认把编译好的库文件libglew.so.1.13放到了/usr/lib64下。由于之后我们要用cmake去编,但是它可能找不到这个文 件夹, 所以我们现在先告诉系统, 该文件夹下有要找的链接库:

1 sudo ldconfig /usr/lib64/

ok, 现在我们处理完了glew, 转去编译SiftGPU。SiftGPU也是用Makefile编译的。现在转到SiftGPU所在文件夹。调用

1 make

未完成编译。如果顺利的话,你会在bin/目录里得到几个二进制和一个libsiftapu.so库文件。我们主要使用这个库文件。现在看一下它的链接是否正确

1 ldd bin/libsiftgpu.so

这个命令会输出与它链接的库的信息。请保证没有出现某个链接(特别是刚才的GLEW)没有找到的情况(否则这里会通过, 但后面会出现undefined reference)。像我这样

	xlang@thinking: ~/Downloads/SiftGPU - 🗆 🗴
	xiang@thinking /home/xiang/Downloads/SiftGPU
% Ldd	
	libGLEW.so.1.13 => /usr/lib64/libGLEW.so.1.13 (0x00007f569486b000)
	libGL.so.1 => /usr/lib/x86_64-linux-gnu/mesa/libGL.so.1 (0x00007f56945d8000)
	libX11.so.6 => /usr/lib/x86_64-linux-gnu/libX11.so.6 (0x00007f56942a3000) libIL.so.1 => /usr/lib/x86_64-linux-gnu/libIL.so.1 (0x00007f5693f88000)
	libstdc++.so.6 => /usr/lib/x86_64-linux-gnu/libstdc++.so.6 (0x00007f5693c83000)
	libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007f569397d000)
	libacc_s.so.1 => /lib/x86_64-linux-anu/libacc_s.so.1 (0x00007f5693767000)
	libc.so.6 => /Lib/x86_64-Linux-gnu/Libc.so.6 (0x00007f56933a1000)
	libexpat.so.1 => /lib/x86_64-linux-gnu/libexpat.so.1 (0x00007f5693177000)
	libglapi.so.0 => /usr/lib/x86_64-linux-gnu/libglapi.so.0 (0x00007f5692f4d000)
	libXext.so.6 => /usr/lib/x86_64-linux-anu/libXext.so.6 (0x00007f5602d3a000)
	libXdamage.so.1 => /usr/lib/x86_64-linux-gnu/libXdamage.so.1 (0x00007f5692b37000)
	LibXfixes.so.3 => /usr/Lib/x86_64-Linux-gnu/LibXfixes.so.3 (0x00007f5692931000)
666)	
	$libxcb-sync.so.1 => /usr/lib/x86_64-linux-gnu/libxcb-sync.so.1 (0x00007f5691d05000)$
	libxcb.so.1 => /usr/lib/x86_64-linux-gnu/libxcb.so.1 (0x00007f56918e5000) libxshmfence.so.1 => /usr/lib/x86_64-linux-gnu/libxshmfence.so.1 (0x00007f56918e3000
	libXxf86vm.so.1 => /usr/lib/x86_64-linux-gnu/libXxf86vm.so.1 (0x0000/f56916dd000)
	libdrm.so.2 => /usr/Lib/x86_64-Linux-gnu/Libdrm.so.2 (0x00007f56914d0000)
	libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f56912b2000)
	libdl.so.2 => /lib/x86_64-linux-qnu/libdl.so.2 (0x00007f56910se000)
	libtiff.so.5 => /usr/lib/x86_64-linux-gnu/libtiff.so.5 (0x00007f5690@3b000)
	libpng12.so.0 => /lib/x86_64-linux-gnu/libpng12.so.0 (0x00007f5690c15000)
	libmng.so.2 => /usr/lib/x86_64-linux-gnu/libmng.so.2 (0x00007f5690998000)
	liblcms.so.1 => /usr/lib/x86_64-linux-gnu/liblcms.so.1 (0x00007f5600761000)
	Lib ipeg.so.8 \Rightarrow /usr/Lib/x86_64-Linux-gnu/Lib ipeg.so.8 (0x00007f569050c000)

如果这步正确无误,恭喜你,SiftGPU已经编译完成了!真是可喜可贺呀!

小萝卜:然后呢?师兄我还没看到什么感觉很厉害的东西啊?

师兄:下面我们来实际找一个图片,写一段小程序调用SiftGPU,提一下特征试试。为测试速度,我们还要记录一下代码运行时间。

测试SiftGPU

现在我们来写一个测试程序。由于它比较短,我就不专门搞个github了。请大家跟着我做即可。

h	
1	// SiftGPU模块
2	#include <siftgpu.h></siftgpu.h>
3	
4	//标准C++
5	#include <iostream></iostream>
6	#include <vector></vector>
7	
8	// OpenCV图像
9	#include <opency2 core="" core.hpp=""></opency2>
10	#include <opencv2 highgui="" highgui.hpp=""></opencv2>
11	
12	// boost库中计时函数

公告

昵称: 半闲居士 國龄: 8年10个月 粉丝: 3062

关注: 0

2023年1月						
_	=	Ξ	72	五	大	
2	3	4	5	6	7	
9	10	11	12	13	14	
16	17	18	19	20	21	
23	24	25	26	27	28	
30	31	1	2	3	4	
6	7	8	9	10	11	
	9 16 23 30	2 3 9 10 16 17 23 24 30 31	- = = 2 3 4 9 10 11 16 17 18 23 24 25 30 31 1	- = E	2 3 4 5 6 9 10 11 12 13 16 17 18 19 20 23 24 25 26 27 30 31 1 2 3	

技業	
	找找看
	谷歌搜索

常用链接

我的评论 我的参与 最新评论

我的标签 我的标签

视觉SLAM(17) 机器人(14) SLAM(13)

一起做RGB-D SLAM(7)

Kinect(4)

计算机视觉(2) 图像处理(2)

视觉SLAM漫谈(2)

李群(2) 更多

随笔分类

随笔(2) 一起做rgbd slam(2)

随笔档案

2016年8月(1) 2016年7月(1) 2016年6月(2)

2016年3月(2) 2016年2月(2)

2016年1月(8) 2015年12月(1) 2015年8月(4)

2015年7月(4)

2015年4月(2) 2014年6月(1)

2014年4月(1)

阅读排行榜

1. 视觉SLAM漫淡(211589)

2. 一起做RGB-D SLAM (1)(137824) 3. 一起做RGB-D SLAM (2)(118148)

4. 深入理解图优化与g2o:g2o篇(115570)

5. 视觉SLAM实战(一):RGB-D SLAM V2(110972

评论排行榜

1. 一起做RGB-D SLAM (2)(77)

2. 一起做RGB-D SLAM (5)(66) 3. 一起做RGB-D SLAM (3)(66)

4. 一起做RGB-D SLAM (6)(64)

5. 一起做RGB-D SLAM (4)(62)

推荐排行榜

1. 视觉SLAM漫淡(71)

2. 一起做RGB-D SLAM (2)(34) 3. 一起做RGB-D SLAM (1)(31)

4. 深入理解图优化与g2o:g2o篇(25)

5. 视觉SLAM漫谈 (三): 研究点介绍(25)

最新评论

1. Re:一起做RGB-D SLAM (5)

@going go 你好 我现在也是碰到这个问题。想问 下大佬是怎么解决的...

2 Re:一起做RGR-D SI AM (4)

大家好, 我在执行bin/detectfeatures时出现以下 错误了oodboy@goodboy-virtual-machine:~/slam\$ bin/detectFeatures bin/dete..

3. Re:一起做RGB-D SLAM (1) 博主. 您好。我在执行第三部分

bin/detectFeatures时出现 bin/detectFeatures: symbol lookup error: bin/detectFeatures: unde...

4. Re:视觉SLAM实战(二):ORB-SLAM2 with

大佬你好,看了您的文章,非常钦佩您VSLAM实战教 程以方面的讲解, 有兴趣合作成为我们古月居网站的

```
20 int main( int argc, char** argv)
        //声明SiftGPU并初始化
        char* myargv[4] ={ "-fo", "-1", "-v", "1"};
        //检查硬件是否支持SiftGPU
        int support = sift.CreateContextGL();
if ( support != SiftGPU::SIFTGPU_FULL_SUPPORTED )
            cerr<<"SiftGPU is not supported!"<<endl;
        boost::timer timer;
          /在此填入你想测试的图像的路径!不要用我的路径!不要用我的路径!不要用我的路径!
        sift.RunSIFT( "/home/xiang/wallE-slam/data/rgbl.png" );
cout<<"siftgpu::runsift() cost time="<<timer.elapsed()<<endl;</pre>
        int num = sift.GetFeatureNum();
        vector<SiftGPU::SiftKevpoint> kevs(num);
        sift.GetFeatureVector(&keys[0], &descriptors[0]);
        cout<<"siftgpu::getFeatureVector() cost time="<<timer.elapsed()<<endl;</pre>
        // 先用OpenCV读取一个图像. 然后调用SiftGPU提取特征
        cv::Mat img = cv::imread("/home/xiang/wallE-slam/data/rgbl.png", 0);
int width = img.cols;
int height = img.rows;
        timer.restart();
         // 注意我们处理的是灰度图, 故照如下设置
        sift.RunSIFT(width, height, img.data, GL_INTENSITY8, GL_UNSIGNED_BYTE);
cout<<"siftgpu::runSIFT() cost time="<<timer.elapsed()<cendl;
Ba .
```

Si依接口还是相当简单的。在这程序里,我们一共做了三件事。一是直接对一个图像路径提Sift、二是获取Sift的关键点和描述子。三是对OpenCV读取的一个图像提取Sift。我们分别到了三者的效果和时间。

接下来,写一个CMakeLists.txt来编译上面的文件。


```
mkdir build
cd build
cmake ..
```

等一下!是不是还忘了些什么呢?嗯,如果你直接去cmake的话,会现一个find_package就不到glew的简!因为我们婆glew的时候是直接用make install 装的嘛,cmake怎么会知道我们干了这件事呢?所以此时find_package(Glew REQUIRED)就会出情啦!

小萝卜:为什么出错了你还是很高兴的样子.....

师兄:对!现在呢我们要自己写一个FindGlew.cmake文件喽。请打开你的编辑器. 输入

```
2 # Try to find GLEW library and include path.
3 # Once done this will define
5 # GLEW FOUND
6 # GLEW_INCLUDE_PATH
 7 # GLEW LIBRARY
10 IF (WIN32)
       FIND_PATH( GLEW_INCLUDE_PATH GL/glew.h
           $ENV{PROGRAMFILES}/GLEW/inclu
          ${PROJECT SOURCE DIR}/src/nvql/qlew/include
           NAMES glew GLEW glew32 glew32s
           PATHS
           ${PROJECT_SOURCE_DIR}/src/nvgl/glew/bin
           ${PROJECT_SOURCE_DIR}/src/nvgl/glew/lib
DOC "The GLEW library")
22 ELSE (WIN32)
      FIND PATH ( GLEW INCLUDE PATH GL/glew.h
           /usr/local/include
           /sw/include
            /opt/local/include
            DOC "The directory where GL/glew.h resides")
     FIND LIBRARY ( GLEW LIBRARY
```

讲师吗. 官网. 了解更多可以添加微信GYH-xiaogu 咨询。

--古月居

--Mai_Qi

 Re:一起做RGB-D SLAM (3) 请问为什么我的旋转矩阵和您的差一个负号呀

```
PATHS
            /usr/lib64
            /usr/lib
/usr/local/lib64
            /usr/local/lib
            /sw/lib
            DOC "The GLEW library")
41 IF (GLEW_INCLUDE_PATH)
       SET( GLEW_FOUND 1 CACHE STRING "Set to 1 if GLEW is found, 0 otherwise")
43 ELSE (GLEW_INCLUDE_PATH)
44 SET(GLEW_FOUND 0 CACHE STRING "Set to 1 if GLEW is found, 0 otherwise")
45 ENDIF (GLEW INCLUDE PATH)
47 MARK_AS_ADVANCED( GLEW_FOUND )
```

然后呢. 把这个文件放到cmake的modules文件夹中去!这样cmake就会知道你在调用 $find_package(Glew)$ 时怎么找啦!

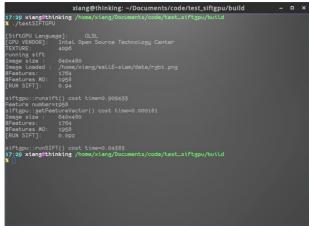
```
注意到这个文件所在的目录通常是没有写权限的的哦!所以我们要用sudo提升到管理员权限才行呢。
 这时,再调用cmake ... 就不会报上面的错误啦!而编译也得以顺利进行下去了。
/home/xiang/Downloads/SiftGFU/src/SiftGFU/SiftGFU.h:336:40: error: declaration of 'operator new' as non-function SIFTGE
U_EXPORT void* operator new (size_t size);
  这是什么原因呢?g++的编译错误很难懂,一直为人诟病。师兄仔细查了查,发现SiftGPU作者重载了new运算符,但是它的参数"size_t size"中
的"size_t"类型. 在linux下编译是需要指定一个头文件的!所以我们打开\sim/Downloads/SiftGPU/src/SiftGPU/siftGPU.h文件. 在上头加入一个
```

vlm ~/Downloads/SiftGPU/src/SiftGPU/SiftGPU.h //clas SiftParam //description: SIFT parameters s/SiftGPU/src/SiftGPU/SiftGPU.h [FORMAT=unix] [TYPE=CPP] [POS=30,2][7%] 21/01/16 - 17:28

这样编译器就会找到size_t类型啦!编译就能通过喽!

SiftGPU运行结果

以下就是在师兄电脑上的运行结果啦,大家可以看一下



对于OpenCV已经读入的数据。在640x480的分辨率下,用SiftGPU只需40多毫秒即可完成计算了呢!GPU真的是很强大啊!即使在没有Cuda的情况下都取 得了近十倍的加速啊!效果拔群!

小萝卜:我的ORB只要30毫秒就行了, 哼.

小结

本篇介绍了SiftGPU,我们带领读者完成了它的编译,并在自己的程序内实现了调用。可以看到它的加速效果还是不错的!

另外,这也是我的一次尝试,告诉读者在编译过程中遇到问题该如何处理。我本可以直接跳过这些buggy的部分,告诉大家运行的结果。但我觉得这样子讲可能

如果你觉得我的博客有帮助,可以进行几块钱的小额赞助,帮助我把博客写得更好。



标签: 视觉SLAM, Sift, SiftGPU











《上一篇: 视觉SLAM中的数学基础 第三篇 李群与李代数 》下一篇: 视觉SLAM实版(二):ORB-SLAM2 with Kinect2

posted @ 2016-01-21 17:37 平闲居士 阅读(24745) 评论(7) 编辑 敬蔵 举报 刷新评论 刷新页面 返回原都

2 **○**推荐 **○**反对

🤜 登录后才能查看或发表评论, 立即 <u>登录</u> 或者 <u>逛逛</u> 博客园首页

Copyright © 2023 半闲居士 Powered by .NET 7.0 on Kubernetes