版本要求

1. **OpenCV**

- 版本要求在2.4.3以上，设置以下环境变量`YOUR_OWN_PATH\opencv\build;YOUR_OWN_PATH\opencv\build\x64\vc12\bin`。如果是32位的机器，记得加上`YOUR_OWN_PATH\opencv\build\x86\vc12\bin`

2. **Cmake**

- 版本在2.8以上，但不要用最新的3.8，因为最新版本不支持VS2013. 建议使用SLAM/cmake中的cMake 3.6 版本。

3. **Visual Studio**

- 源码在VS2013（32、64位）下的编译成功，其他版本没有测试。

编译过程

已将所有的依赖库都放在 Thirdparty 文件夹下。

按照库DBow2，g2O，Pangolin的顺序来编译，Eigen不需要编译，这里以DBow2为例说明：

1.打开cMake，在source code路径选中DBow2文件夹，在build the binaries路径选中DBow2/Build文件夹

2.先按configure，选Visual Studio 12 2013 Win64（按照机子要求选择），再按Generate，这时在DBow2/Build文件夹中会出现DBow2.sln这个解决方案，双击打开。

3.在DBow项目上右键，选中属性，之后在C/C++选项下确保runtime Library是MD(Release模式下）+MDd(Debug模式下），或者MT+MTd，视最终运行的平台而定。

4.在属性-通用-target extension确认是.lib，在属性-项目默认-configuration type下选择static lib

5. 右键ALL BUILD这个项目，然后build.

注1：若一个解决方案中存在多个项目，要确保所有项目都在MD+MDd（或者MT+MTd）模式下才能build。

注2：用orb-slam2里面thirdparty下的DBow2进行编译，不要用官方的（github上），官方版本只支持Opencv3+Opencv Contribute。

注3：Pangolin在build完之后会提示缺少pthread.lib，不用理会，之后程序用不到。

注4：编译g2O的时候记得在属性-preprocesser里添加WINDOWS（没有下划线），因为有部分函数是在这个宏下才会编译进去的。

生成了DBow2.lib，g2O.lib，Pangolin.lib以后，在根目录下也用cmake进行编译，步骤和上述一样，除了第4步是右键ORB_SLAM2而不是ALL_BUILD这个项目，然后build.

运行过程

- 以**mono_tum**为例来说明如何运行样例：

1.在VS中右键选中mono_tum项目，选为setup program，然后进行和ORB_SLAM2一样的build过程。

2.在运行代码之前，解压videofortest文件夹下的视频文件，orbslam/Vocabulary文件夹下的文件。

3.右键mono_tum项目，在Config Property->Debug中的command arguement按顺序添加以下内容：

path_to_vocabulary orbslam/Vocabulary文件夹下词库txt路径

path_to_settings orbslam/Examples/Monocular文件夹下的yaml文件，本视频对应的的是TUM1.yaml

path_to_sequence 视频路径

以下是我电脑的配置路径：
..\..\workspace\ORBvoc.txt
..\..\workspace\TUM1.yaml
..\..\workspace\rgbd_dataset_freiburg1_360
4.选择start without debugging运行程序

- 以新建项目为例说明如何运行

1.添加头文件（以下是我电脑的配置路径)：

F:\config\opencv\build\include
F:\config\opencv\build\include\opencv
F:\SVN_SLAM\trunk\thirdparty\orbslam
F:\SVN_SLAM\trunk\thirdparty\orbslam\include
F:\SVN_SLAM\trunk\thirdparty\orbslam\Thirdparty\eigen
F:\SVN_SLAM\trunk\thirdparty\orbslam\Thirdparty\Pangolin\include
F:\SVN_SLAM\trunk\thirdparty\orbslam\Thirdparty\Pangolin\build\src\include
F:\SVN_SLAM\trunk\thirdparty\orbslam\Thirdparty\Pangolin\build\external\glew\include

2.依赖库路径（以下是我电脑的配置路径)：

F:\config\opencv\build\x86\vc12\lib

3.依赖库（以下是我电脑在Release下的配置路径)：

kernel32.lib
user32.lib
gdi32.lib
winspool.lib
shell32.lib
ole32.lib
oleaut32.lib
uuid.lib
comdlg32.lib
advapi32.lib
glu32.lib
opengl32.lib
Debug\ORB_SLAM2.lib
opencv_videostab2411d.lib
opencv_ts2411d.lib
opencv_superres2411d.lib
opencv_stitching2411d.lib
opencv_contrib2411d.lib
opencv_nonfree2411d.lib
opencv_ocl2411d.lib
opencv_gpu2411d.lib
opencv_photo2411d.lib
opencv_objdetect2411d.lib
opencv_legacy2411d.lib
opencv_video2411d.lib
opencv_ml2411d.lib
opencv_calib3d2411d.lib
opencv_features2d2411d.lib
opencv_highgui2411d.lib

opencv_imgproc2411d.lib
opencv_flann2411d.lib
opencv_core2411d.lib
..\Thirdparty\Pangolin\build\src\Debug\pangolin.lib
..\Thirdparty\Pangolin\build\external\glew\lib\glewd.lib
..\Thirdparty\Pangolin\build\external\libpng\lib\libpng16_staticd.lib
..\Thirdparty\Pangolin\build\external\zlib\lib\zlibstaticd.lib
..\Thirdparty\Pangolin\build\external\libjpeg\lib\jpeg.lib
..\Thirdparty\DBoW2\lib\Release\DBoW2.lib
..\Thirdparty\g2o\build\Release\g2o.lib
4.添加工作路径
把ORBvoc.txt和TUM1.yaml 放在F:\orbslam\trunk\workspace\下面，并且把property-debugging-working directory改成$(ProjectDir)../../workspace
5.添加测试代码：

```
#include<opencv2/core/core.hpp>
#include<System.h>
...
while(1){
//test mono orb-slam
const std::string vocabularyPath = "ORBvoc.txt";
const std::string settingPath = "TUM1.yaml";
// Create SLAM system. It initializes all system threads and gets ready to process frames.
static ORB_SLAM2::System SLAM(vocabularyPath, settingPath, ORB_SLAM2::System::MONOCULAR, true);
cout << "Start processing sequence ..." << endl;
//cout << "Images in the sequence: " << nImages << endl << endl;
cv::VideoCapture capture(0);//0 for default camera while 1 for specific camera
capture.set(CV_CAP_PROP_FRAME_WIDTH, 600);
capture.set(CV_CAP_PROP_FRAME_HEIGHT, 480);
cv::Mat im;
cv::Mat firstPose;
// Read image from file
capture >> im;
// Pass the image to the SLAM system
//SLAM.TrackMonocular(im,tframe);
SLAM.TrackMonocular(im, 0);
}
```

6.连接网络摄像头,选择start without debugging运行程序