OpenCV3.2.0 + CUDA8.0 + Visual Studio 2015 x64 配置

2017.2.15

OpenCV 官网提供的预编译库并不包含完整的 CUDA 功能，所以要想体验 GPU 加速技术带给 OpenCV 的变化，只能自己重新编译整个库。

1. 下载安装 CUDA Toolkit

https://developer.nvidia.com/下载最新的 CUDA Tookit 。验证 CUDA 是否安装正确，可以尝试编译任意一个 CUDA 自带的样例程序，图 1。

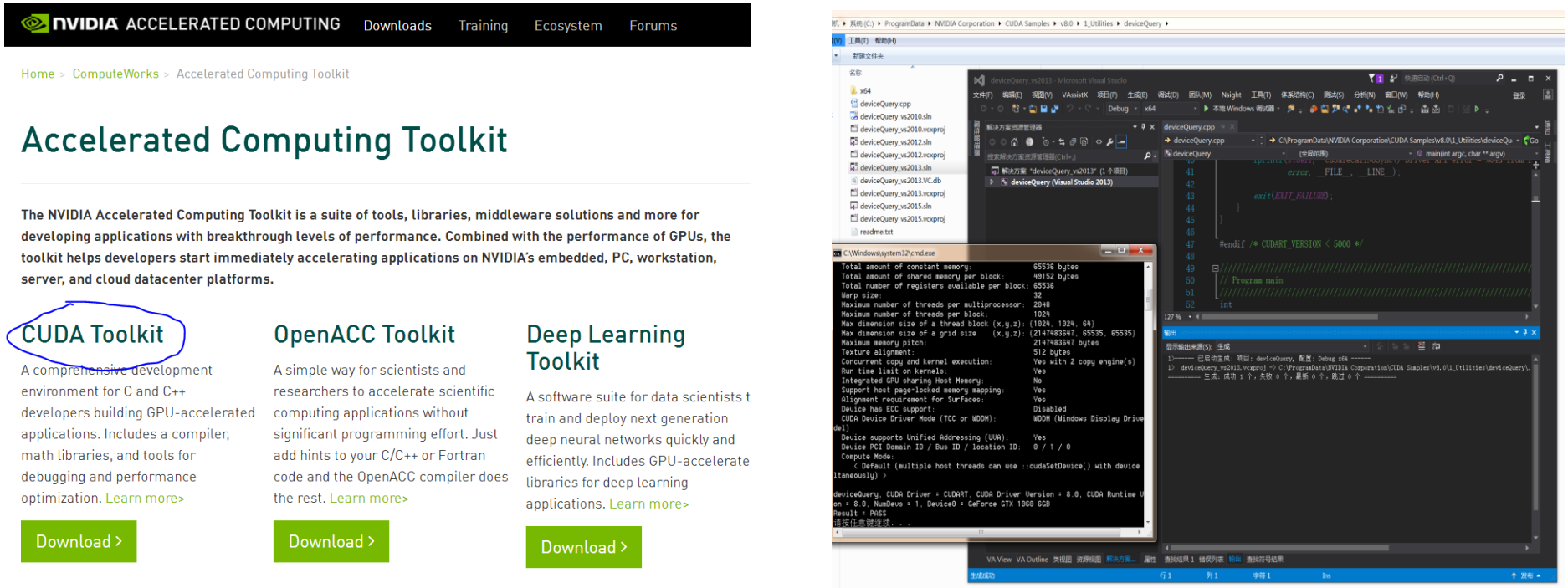
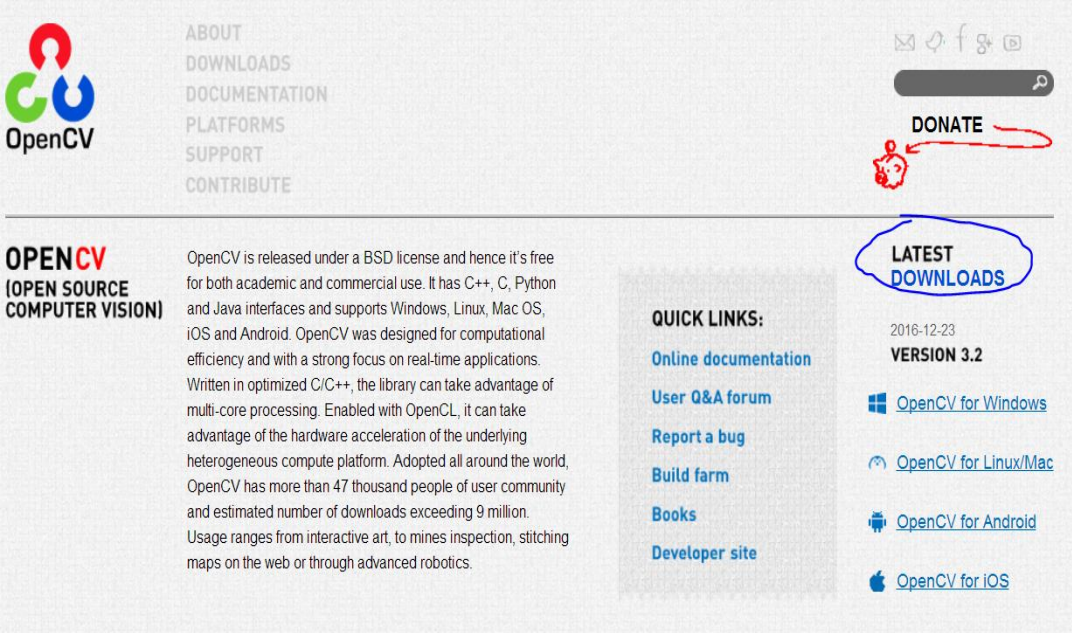


图 1

注意：CUDA8.0（支持 Pascal 系列 GPU）可以兼容 opencv3.2 和 opencv2.4.13，2016 年之前发布的 opencv 版本存在部分不兼容。如果显卡是 GTX1060 或以上，最好采用 CUDA8.0+ opencv3.2/ opencv2.4.13+VS(13/15) ×64 的搭配。

2. 下载安装 OpenCV　和 CMake

http://opencv.org/　　http://www.cmake.org/



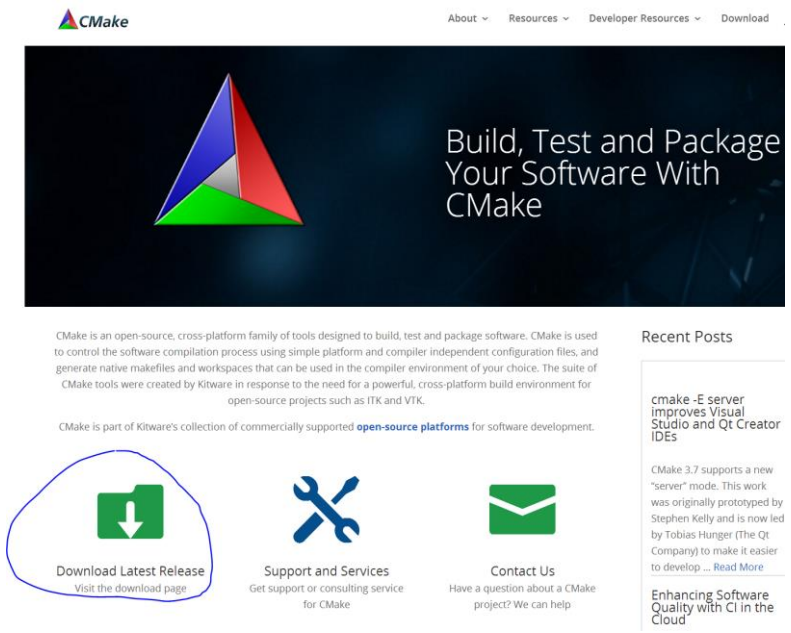


图 2

3. CMake 配置

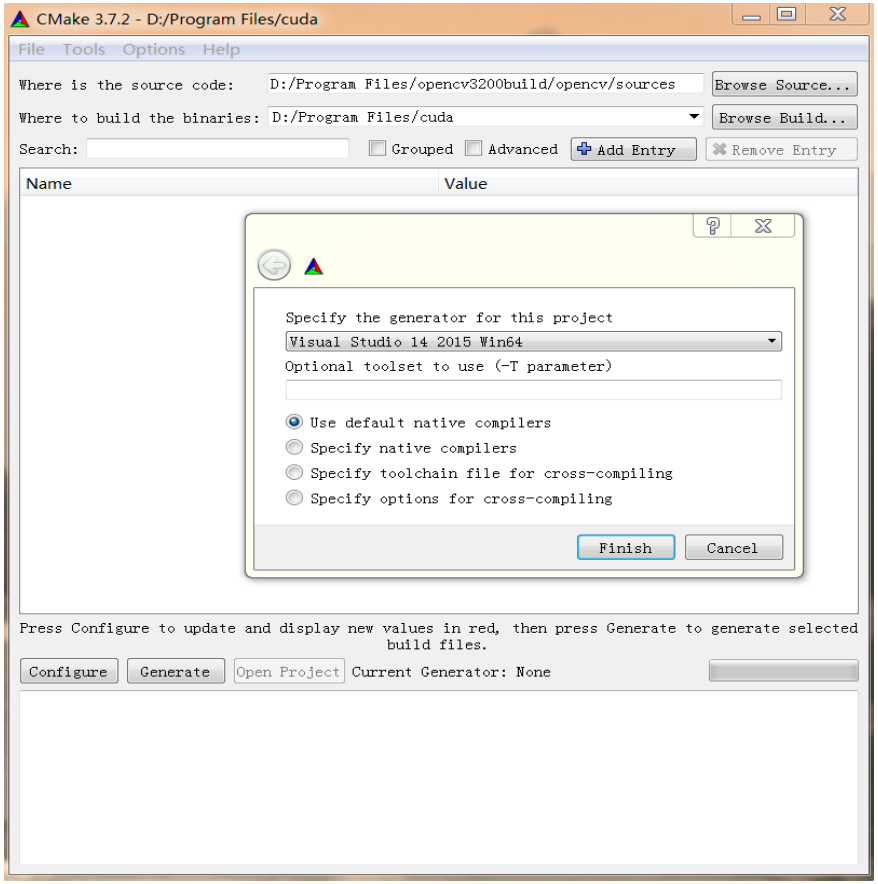
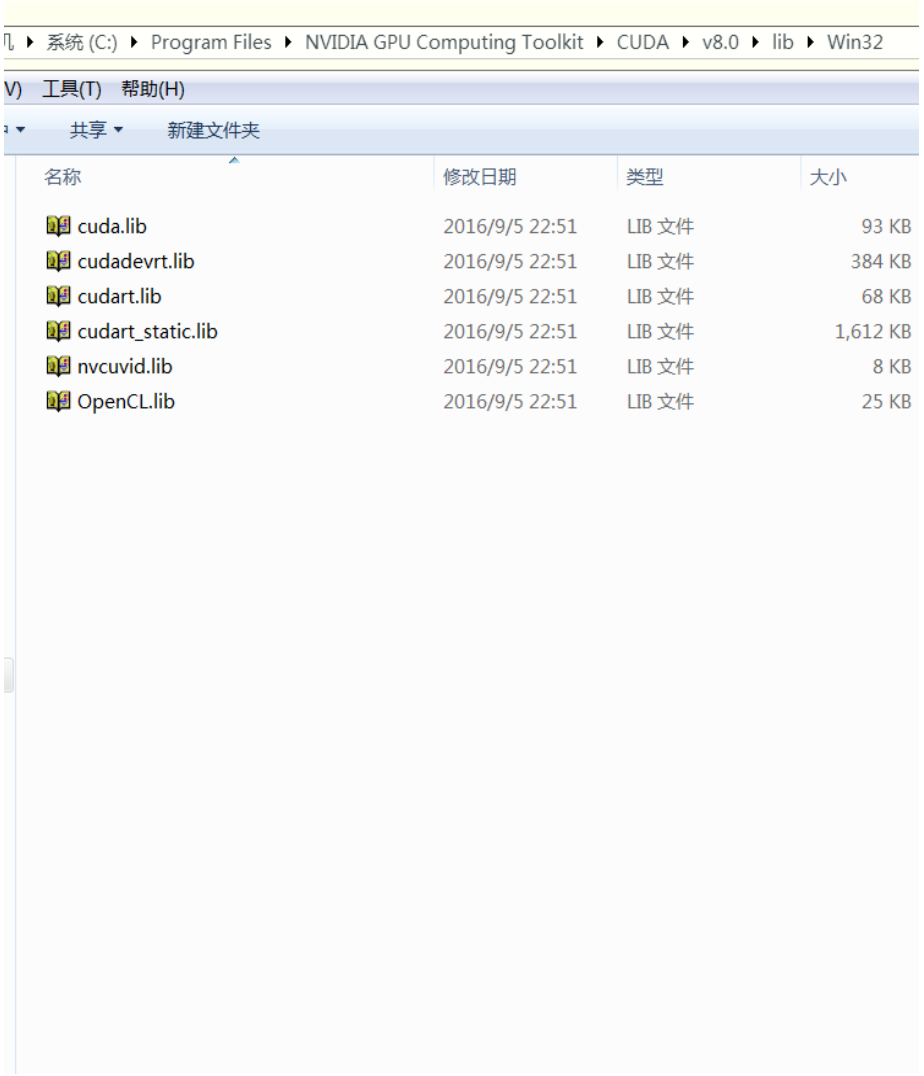


图 3

在 CUDA Toolkit 8.0(7.5 也一样)中，虽然给出了适用 Win32 和 x64 两种目标架构的库，但像 cufft、npps、nvblas 等 OpenCV 所需的库只有 x64 版本。这也限制了我们只能编译 64 位的 OpenCV 库，而且在今后的编程中也要编写针对 x64 架构的程序。另外，在 GUI 组件中，CUDA 也调用了部分 OpenGL 功能，所以编译过程也需要 OpenGL 库的支持。下图中左边是 CUDA 为 Win32 提供的库，右边是为 x64 提供的库。


系统 (C:) ▸ Program Files ▸ NVIDIA GPU Computing Toolkit ▸ CUDA ▸ v8.0 ▸ lib ▸ Win32
V) 工具(T) 帮助(H)
共享 新建文件夹
名称 修改日期 类型 大小
cuda.lib 2016/9/5 22:51 LIB 文件 93 KB
cudadevrt.lib 2016/9/5 22:51 LIB 文件 384 KB
cudart.lib 2016/9/5 22:51 LIB 文件 68 KB
cudart_static.lib 2016/9/5 22:51 LIB 文件 1,612 KB
nvcuvid.lib 2016/9/5 22:51 LIB 文件 8 KB
OpenCL.lib 2016/9/5 22:51 LIB 文件 25 KB

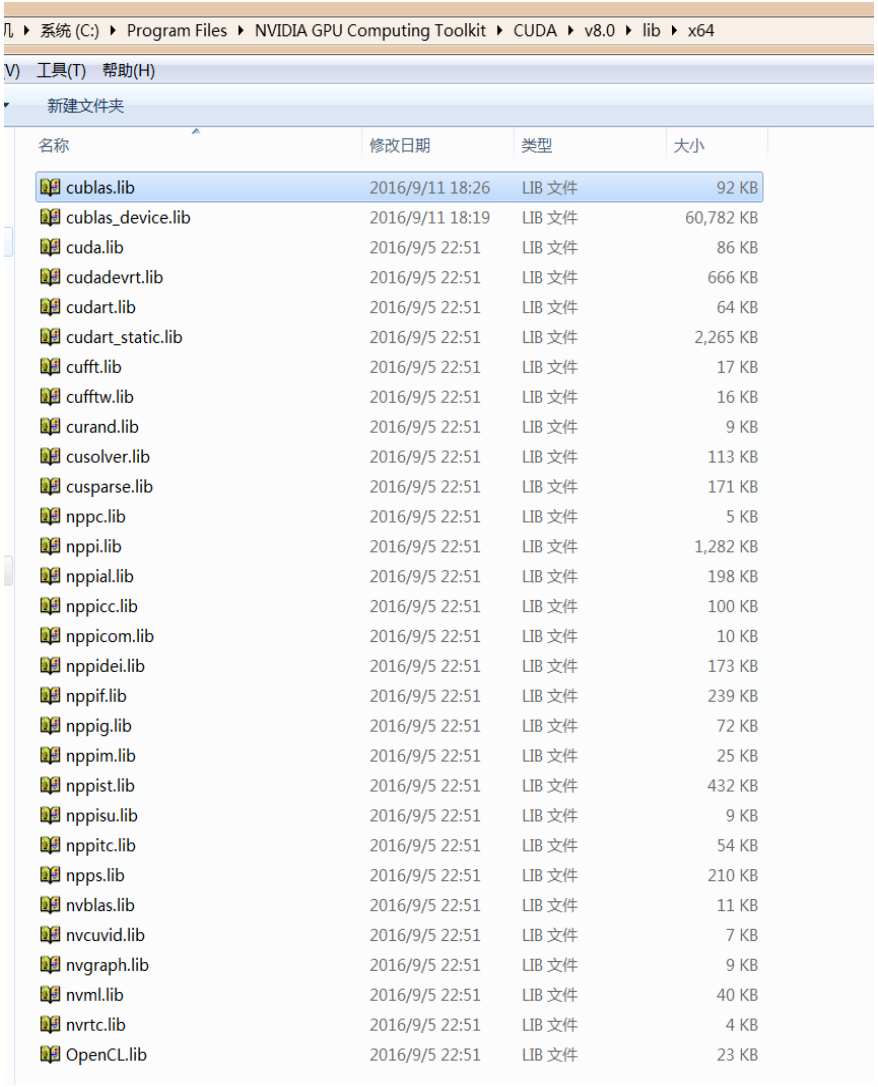

系统 (C:) ▸ Program Files ▸ NVIDIA GPU Computing Toolkit ▸ CUDA ▸ v8.0 ▸ lib ▸ x64
V) 工具(T) 帮助(H)
新建文件夹
名称 修改日期 类型 大小
cublas.lib 2016/9/11 18:26 LIB 文件 92 KB
cublas_device.lib 2016/9/11 18:19 LIB 文件 60,782 KB
cuda.lib 2016/9/5 22:51 LIB 文件 86 KB
cudadevrt.lib 2016/9/5 22:51 LIB 文件 666 KB
cudart.lib 2016/9/5 22:51 LIB 文件 64 KB
cudart_static.lib 2016/9/5 22:51 LIB 文件 2,265 KB
cufft.lib 2016/9/5 22:51 LIB 文件 17 KB
cufftw.lib 2016/9/5 22:51 LIB 文件 16 KB
curand.lib 2016/9/5 22:51 LIB 文件 9 KB
cusolver.lib 2016/9/5 22:51 LIB 文件 113 KB
cusparse.lib 2016/9/5 22:51 LIB 文件 171 KB
nppc.lib 2016/9/5 22:51 LIB 文件 5 KB
nppi.lib 2016/9/5 22:51 LIB 文件 1,282 KB
nppial.lib 2016/9/5 22:51 LIB 文件 198 KB
nppicc.lib 2016/9/5 22:51 LIB 文件 100 KB
nppicom.lib 2016/9/5 22:51 LIB 文件 10 KB
nppidei.lib 2016/9/5 22:51 LIB 文件 173 KB
nppif.lib 2016/9/5 22:51 LIB 文件 239 KB
nppig.lib 2016/9/5 22:51 LIB 文件 72 KB
nppim.lib 2016/9/5 22:51 LIB 文件 25 KB
nppist.lib 2016/9/5 22:51 LIB 文件 432 KB
nppisu.lib 2016/9/5 22:51 LIB 文件 9 KB
nppitc.lib 2016/9/5 22:51 LIB 文件 54 KB
npps.lib 2016/9/5 22:51 LIB 文件 210 KB
nvblas.lib 2016/9/5 22:51 LIB 文件 11 KB
nvcuvid.lib 2016/9/5 22:51 LIB 文件 7 KB
nvgraph.lib 2016/9/5 22:51 LIB 文件 9 KB
nvml.lib 2016/9/5 22:51 LIB 文件 40 KB
nvrtc.lib 2016/9/5 22:51 LIB 文件 4 KB
OpenCL.lib 2016/9/5 22:51 LIB 文件 23 KB

图 4

4. CMake 中的选项

不要勾选 BUILD_CUDA_STUBS 。我们需要勾选的是：

WITH_CUBLAS、ITH_CUDA、WITH_CUFFT、WITH_EIGEN 、WITH_IPP、 WITH_OPENGL、WITH_OPENMP、WITH_TBB。

其中 **WITH_CUDA、WITH_CUFFT、WITH_EIGEN、WITH_IPP** 是默认开启的。（未使用 TBB 可以不开启 WITH_TBB） 。

CUDA 下面还有一个 **WITH_FAST_MATH** ，是 CUDA 的快速数学库，牺牲精度谋求速度，如果只要求单精度浮点可以选择该项。

勾选 **BUILD_opencv_world**，最后生成的库会合并为两项 **opencv_world320.dll** 以及 **opencv_ffmpeg320.dll**，这样做的好处是只需要引用两个库，缺点是很多可能用不着的库在运行时会加载。比如**编译好后的 opencv_world320d.dll 有 900 多 MB。因此不建议勾选此项。**

OPENCV_ENABLE_NONFREE 勾选(包含 SURF 等受到专利保护的算法),OPENCV_EXTRA_MODULES_PATH 需要填写正确的额外库路径。（ 文件下载地址 https://github.com/opencv/opencv_contrib ）

依次点击 Configure、Generate 。

Configure 的过程中可能会下载需要的文件，比如 ippicv_windows_20151201.zip 等，也可以从其他地方下载好后复制到相关位置。

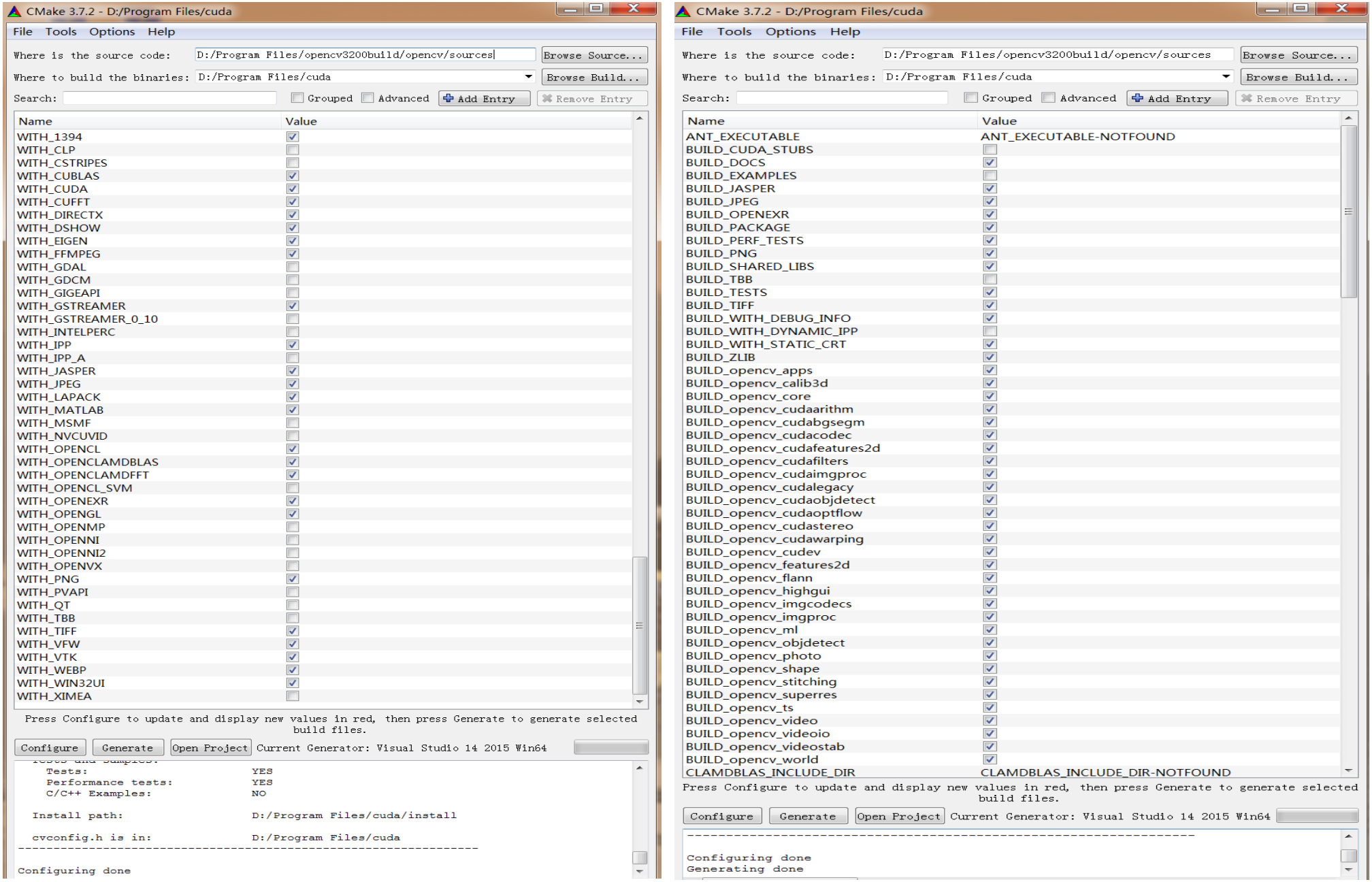

CMake 3.7.2 - D:/Program Files/cuda
File Tools Options Help
Where is the source code: D:/Program Files/opencv3200build/opencv/sources
Browse Source...
Where to build the binaries: D:/Program Files/cuda
Browse Build...
Search:
Grouped
Advanced
Add Entry
Remove Entry
Name
Value
WITH_1394
WITH_CLP
WITH_CSTRIPES
WITH_CUBLAS
WITH_CUDA
WITH_CUFFT
WITH_DIRECTX
WITH_DSHOW
WITH_EIGEN
WITH_FFMPEG
WITH_GDAL
WITH_GDCM
WITH_GIGEAPI
WITH_GSTREAMER
WITH_GSTREAMER_0_10
WITH_INTELPERC
WITH_IPP
WITH_IPP_A
WITH_JASPER
WITH_JPEG
WITH_LAPACK
WITH_MATLAB
WITH_MSMF
WITH_NVCUVID
WITH_OPENCL
WITH_OPENCLAMDBLAS
WITH_OPENCLAMDFFT
WITH_OPENCL_SVM
WITH_OPENEXR
WITH_OPENGL
WITH_OPENMP
WITH_OPENNI
WITH_OPENNI2
WITH_OPENVX
WITH_PNG
WITH_PVAPI
WITH_QT
WITH_TBB
WITH_TIFF
WITH_VFW
WITH_VTK
WITH_WEBP
WITH_WIN32UI
WITH_XIMEA
Press Configure to update and display new values in red, then press Generate to generate selected build files.
Configure
Generate
Open Project
Current Generator: Visual Studio 14 2015 Win64
Tests: YES
Performance tests: YES
C/C++ Examples: NO
Install path: D:/Program Files/cuda/install
cvconfig.h is in: D:/Program Files/cuda
Configuring done
CMake 3.7.2 - D:/Program Files/cuda
File Tools Options Help
Where is the source code: D:/Program Files/opencv3200build/opencv/sources
Browse Source...
Where to build the binaries: D:/Program Files/cuda
Browse Build...
Search:
Grouped
Advanced
Add Entry
Remove Entry
Name
Value
ANT_EXECUTABLE
ANT_EXECUTABLE-NOTFOUND
BUILD_CUDA_STUBS
BUILD_DOCS
BUILD_EXAMPLES
BUILD_JASPER
BUILD_JPEG
BUILD_OPENEXR
BUILD_PACKAGE
BUILD_PERF_TESTS
BUILD_PNG
BUILD_SHARED_LIBS
BUILD_TBB
BUILD_TESTS
BUILD_TIFF
BUILD_WITH_DEBUG_INFO
BUILD_WITH_DYNAMIC_IPP
BUILD_WITH_STATIC_CRT
BUILD_ZLIB
BUILD_opencv_apps
BUILD_opencv_calib3d
BUILD_opencv_core
BUILD_opencv_cudaarithm
BUILD_opencv_cudabgsegm
BUILD_opencv_cudacodec
BUILD_opencv_cudafeatures2d
BUILD_opencv_cudafilters
BUILD_opencv_cudaimgproc
BUILD_opencv_cudalegacy
BUILD_opencv_cudaobjdetect
BUILD_opencv_cudaoptflow
BUILD_opencv_cudastereo
BUILD_opencv_cudawarping
BUILD_opencv_cudev
BUILD_opencv_features2d
BUILD_opencv_flann
BUILD_opencv_highgui
BUILD_opencv_imgcodecs
BUILD_opencv_imgproc
BUILD_opencv_ml
BUILD_opencv_objdetect
BUILD_opencv_photo
BUILD_opencv_shape
BUILD_opencv_stitching
BUILD_opencv_superres
BUILD_opencv_ts
BUILD_opencv_video
BUILD_opencv_videoio
BUILD_opencv_videostab
BUILD_opencv_world
CLAMDBLAS_INCLUDE_DIR
CLAMDBLAS_INCLUDE_DIR-NOTFOUND
Press Configure to update and display new values in red, then press Generate to generate selected build files.
Configure
Generate
Open Project
Current Generator: Visual Studio 14 2015 Win64
Configuring done
Generating done

图 5

5. 用 Visual Studio 编译 OpenCV

Generate done 完成后，点击 Open Project。首先在 Debug 模式和 Release 模式下，分别 Build **ZERO_CHECK** 项目，看两次编译是否产生错误。如果两次编译成功，则进行下一步：

依次进行 ALL_BUILD（生成很多 OpenCV 项目文件，D+R 耗时 6 Hours）、INSTALL（在 build 下生成 install 文件夹，耗时 5 mins）。

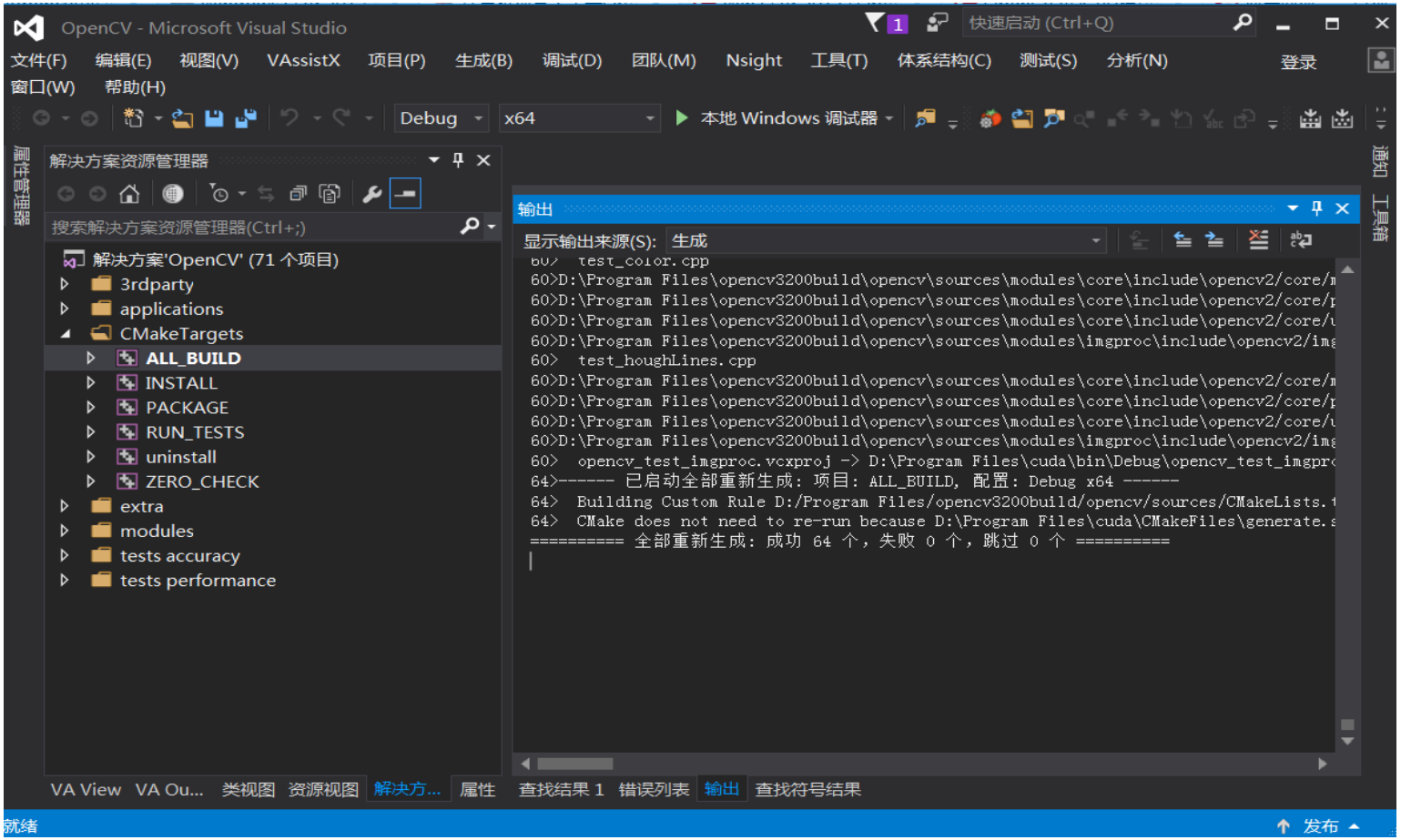


图 6

6. 复制编译好的文件并配置属性表

将编译目录中 install 文件夹下的内容复制到 opencv 的 build 文件夹中，原来的 build 文件夹可以命名为 oldbuild（也可以删除）。编译目录下的 bin 文件夹和 lib 文件夹中的内容也要拷贝到 opencv 的 build 文件夹中去。

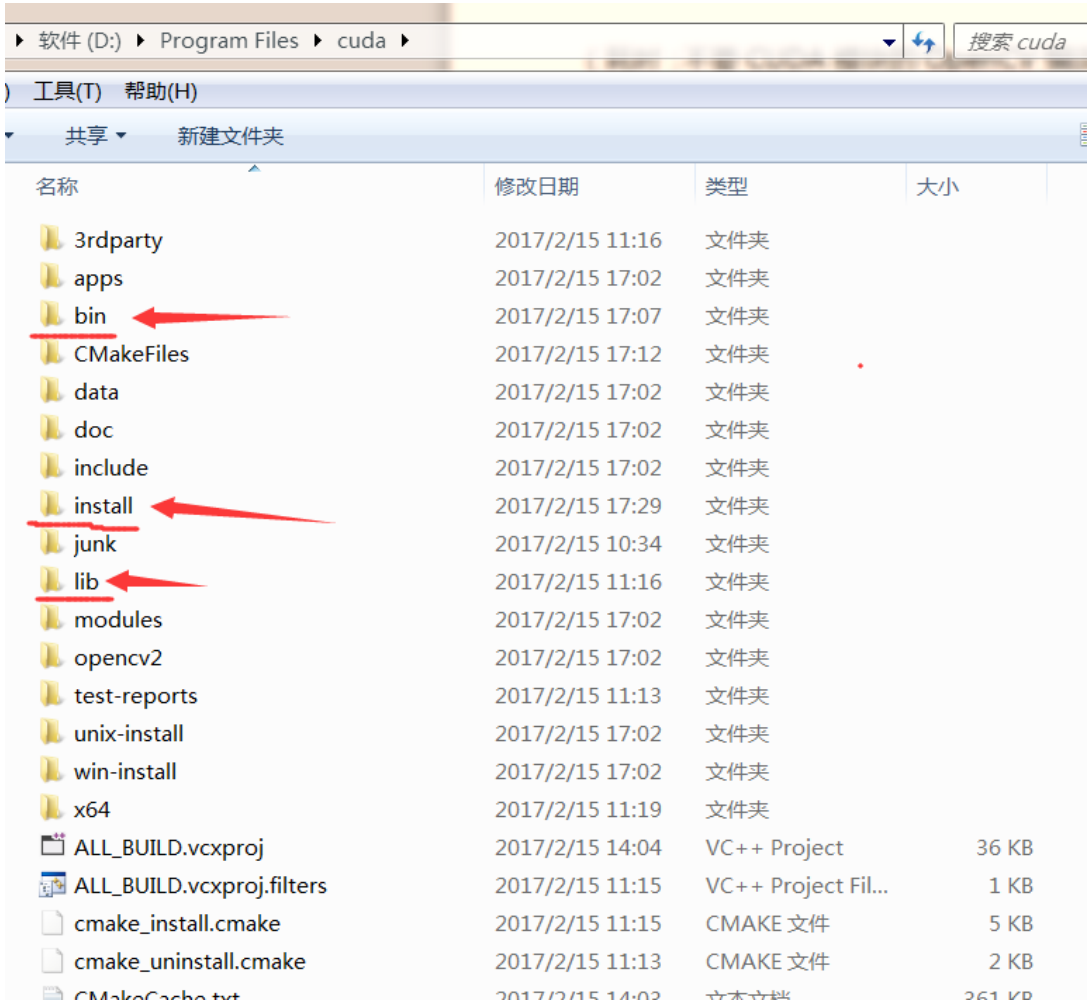


图 7

7. 配置属性表

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <ImportGroup Label="PropertySheets" />
  <PropertyGroup Label="UserMacros" />
  <PropertyGroup>
    <IncludePath>$(OPENCV3200CUDA)\include;$(OPENCV3200CUDA)\include\opencv;$(OPENCV3200CUDA)\include\opencv2;$(IncludePath)</IncludePath>
  <LibraryPath Condition="'$(Platform)'=='X64'">$(OPENCV3200CUDA)\x64\vc12\lib;$(LibraryPath)</LibraryPath>
  </PropertyGroup>
  <ItemDefinitionGroup>
    <Link Condition="'$(Configuration)'=='Debug'">
      <AdditionalDependencies>opencv_calib3d320d.lib;opencv_core320d.lib;opencv_cudaarithm320d.lib;opencv_cudabgsegm320d.lib;
                              opencv_cudacodec320d.lib;opencv_cudafeatures2d320d.lib;opencv_cudafilters320d.lib;opencv_cudaimgproc320d.lib;opencv_cudalegacy320d.lib;
                              opencv_cudaobjdetect320d.lib;opencv_cudaoptflow320d.lib;opencv_cudastereo320d.lib;opencv_cudawarping320d.lib;opencv_cudev320d.lib;
                              opencv_features2d320d.lib;opencv_flann320d.lib;opencv_highgui320d.lib;opencv_imgcodecs320d.lib;opencv_imgproc320d.lib;opencv_ml320d.lib;
                              opencv_objdetect320d.lib;opencv_photo320d.lib;opencv_shape320d.lib;opencv_stitching320d.lib;opencv_superres320d.lib;opencv_video320d.lib;
                              opencv_videoio320d.lib;opencv_videostab320d.lib;
      %(AdditionalDependencies)</AdditionalDependencies>
    </Link>
    <Link Condition="'$(Configuration)'=='Release'">
      <AdditionalDependencies>opencv_calib3d320.lib;opencv_core320.lib;opencv_cudaarithm320.lib;opencv_cudabgsegm320.lib;
                                opencv_cudacodec320.lib;opencv_cudafeatures2d320.lib;opencv_cudafilters320.lib;opencv_cudaimgproc320.lib;opencv_cudalegacy320.lib;
                                opencv_cudaobjdetect320.lib;opencv_cudaoptflow320.lib;opencv_cudastereo320.lib;opencv_cudawarping320.lib;opencv_cudev320.lib;
                                opencv_features2d320.lib;opencv_flann320.lib;opencv_highgui320.lib;opencv_imgcodecs320.lib;opencv_imgproc320.lib;opencv_ml320.lib;
                                opencv_objdetect320.lib;opencv_photo320.lib;opencv_shape320.lib;opencv_stitching320.lib;opencv_superres320.lib;opencv_video320.lib;
                                opencv_videoio320.lib;opencv_videostab320.lib;
    %(AdditionalDependencies)</AdditionalDependencies>
    </Link>
  </ItemDefinitionGroup>
  <ItemGroup />
</Project>
```

测试

```
#include "stdafx.h"
#include <iostream>
#include "opencv2/core/core.hpp"
#include <opencv2/highgui/highgui.hpp>
#include "opencv2/core/cuda.hpp"
#include <opencv2/cudaimgproc.hpp>

/*****************************************************************************
需要注意的是，在所有使用GPU模块的函数之前，最好需要调用函数
gpu::getCudaEnabledDeviceCount，
如果这个函数返回值为0，同时你在命令行中能够看到“CUDA is no support”的错误，
说明没有编译成功
*****************************************************************************/

using namespace cv;
using namespace std;

int main()
{
    int num_devices = cuda::getCudaEnabledDeviceCount();

    if (num_devices <= 0)
    {
        cout << "There is no device." << endl;
        return -1;
    }

    int enable_device_id = -1;
    for (int i = 0; i < num_devices; i++)
    {
        cuda::DeviceInfo dev_info(i);
        if (dev_info.isCompatible())
        {
            enable_device_id = i;
        }
    }
    if (enable_device_id < 0)
    {
        cout << "GPU module isn't built for GPU" << endl;
        return -1;
    }
    cuda::setDevice(enable_device_id);  // 设置当前使用的CUDA

    cout << "GPU is ready, device ID is " << num_devices << "\n";

    Mat src_image = imread("boldt.jpg", 1);
    Mat dst_image;
    cuda::GpuMat d_src_img(src_image);
    cuda::GpuMat d_dst_img;
    cuda::cvtColor(d_src_img, d_dst_img, COLOR_BGR2GRAY);
    d_dst_img.download(dst_image);
    namedWindow("test", 0);
    imshow("test", dst_image);
    waitKey(0);

    return 0;
}
```

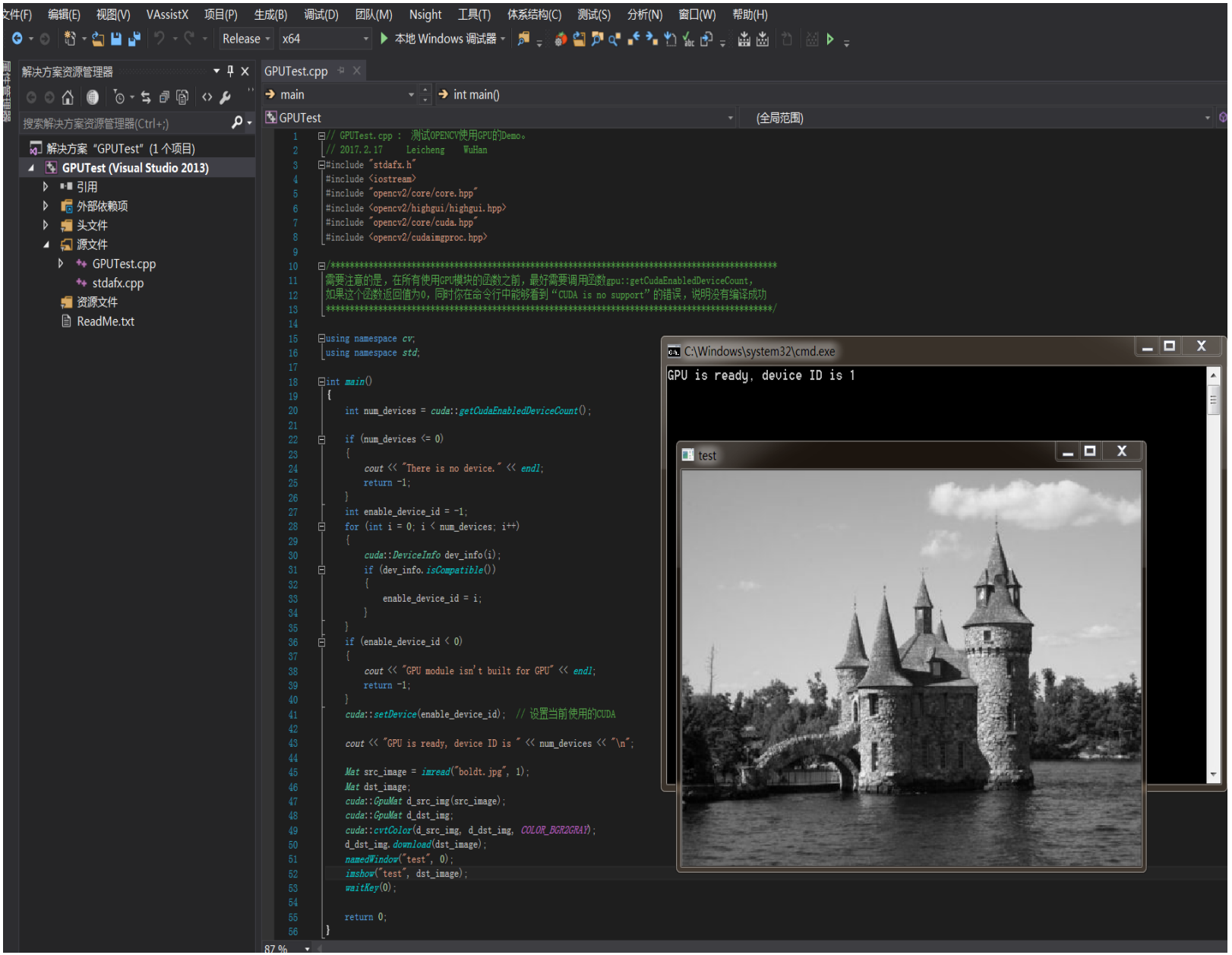

文件(F) 编辑(E) 视图(V) VAssistX 项目(P) 生成(B) 调试(D) 团队(M) Nsight 工具(T) 体系结构(C) 测试(S) 分析(N) 窗口(W) 帮助(H)
Release x64 本地 Windows 调试器
解决方案资源管理器
搜索解决方案资源管理器(Ctrl+;)
解决方案 "GPUTest" (1 个项目)
GPUTest (Visual Studio 2013)
引用
外部依赖项
头文件
源文件
GPUTest.cpp
stdafx.cpp
资源文件
ReadMe.txt
GPUTest.cpp
main
int main()
GPUTest
(全局范围)
// GPUTest.cpp : 测试OPENCV使用GPU的Demo。
// 2017.2.17 Leicheng WuHan
#include "stdafx.h"
#include <iostream>
#include "opencv2/core/core.hpp"
#include <opencv2/highgui/highgui.hpp>
#include "opencv2/core/cuda.hpp"
#include <opencv2/cudaimgproc.hpp>
需要注意的是，在所有使用GPU模块的函数之前，最好需要调用函数gpu::getCudaEnabledDeviceCount，
如果这个函数返回值为0，同时你在命令行中能够看到 "CUDA is no support" 的错误，说明没有编译成功
using namespace cv;
using namespace std;
int main()
{
int num_devices = cuda::getCudaEnabledDeviceCount();
if (num_devices <= 0)
{
cout << "There is no device." << endl;
return -1;
}
int enable_device_id = -1;
for (int i = 0; i < num_devices; i++)
{
cuda::DeviceInfo dev_info(i);
if (dev_info.isCompatible())
{
enable_device_id = i;
}
}
if (enable_device_id < 0)
{
cout << "GPU module isn't built for GPU" << endl;
return -1;
}
cuda::setDevice(enable_device_id); // 设置当前使用的CUDA
cout << "GPU is ready, device ID is " << num_devices << "\n";
Mat src_image = imread("boldt.jpg", 1);
Mat dst_image;
cuda::GpuMat d_src_img(src_image);
cuda::GpuMat d_dst_img;
cuda::cvtColor(d_src_img, d_dst_img, COLOR_BGR2GRAY);
d_dst_img.download(dst_image);
namedWindow("test", 0);
imshow("test", dst_image);
waitKey(0);
return 0;
}
87 %
C:\Windows\system32\cmd.exe
GPU is ready, device ID is 1
test

图 8