Lazy OpenCV installation and use with Visual Studio

Overview

This tutorial will walk you through:

- How to install OpenCV on Windows, both:
  - The pre-built version (useful if you **won't** be modifying the OpenCV library itself, and only want to use it for your applications)
  - A built-from-scratch version (useful for debugging your own applications, if you need customised builds of OpenCV with specific features (multicore processing with Threading Building Blocks, improved user interface with Qt, etc.), or if you are modifying the underlying OpenCV library itself.
- How to configure a **property sheet** to allow easy setup of Visual Studio projects to link against OpenCV.
- How to *use* a property sheet in a project. This configures all required header files and libraries in a new Visual Studio project in a few clicks.

Contents

Prerequisites

Visual Studio professional or better – version 2010 is the latest version which works with CUDA builds for GPU, so we'll use that. 2012 will work for non-GPU versions too. 2008 will probably work, but hasn't been tested. Available for free from Dreamspark https://www.dreamspark.com/ for students. Express versions will probably work too.

CMake 2.8.x build system. Available from http://www.cmake.org/cmake/resources/software.html#latest – download and install the win32 binary.

Optional

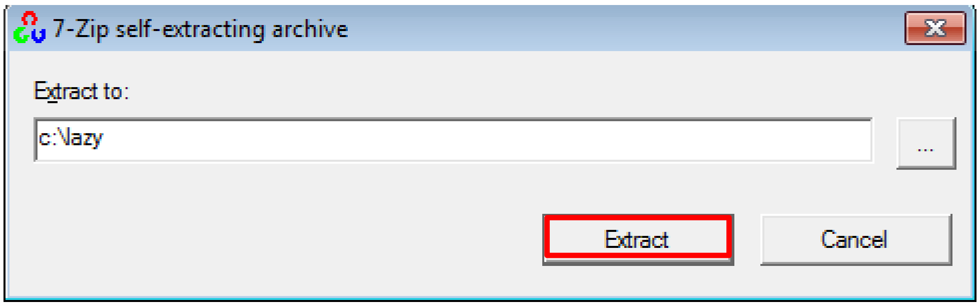
None of the following libraries are needed to get OpenCV running. However, depending on the extra features you want to add to OpenCV, download and install the following extras **before** installing OpenCV. Full details are on the main OpenCV installation tutorial here, but these are the ones I've found useful.

- GPU support for nVidia GPUs: download and install the **CUDA toolkit** from http://www.nvidia.com/getcuda
- Threading building blocks (TBB) for multicore processing: get the windows build from http://threadingbuildingblocks.org/download. Unpack it somewhere and add the appropriate binaries folder to the system path.
- Improved GUI with Qt: (puts a toolbar on all GUI windows, allows access to a control panel with buttons, sliders etc.). Download and install the **Qt libraries** from http://qt-project.org/downloads.
- **mexopencv** http://www.cs.stonybrook.edu/~kyamagu/mexopencv/index.html is also handy and allows use of OpenCV functions from within MATLAB. Set up to work with prebuilt OpenCV, and anything else needs modification of the build script. Not covered here.

1. Installing OpenCV

Note: there is an official tutorial at http://docs.opencv.org/doc/tutorials/introduction/windows_install/windows_install.html. It's quite comprehensive, but this one takes slightly less time and has notes on property sheets.

Download the latest version of the installer from http://sourceforge.net/projects/opencvlibrary/files/opencv-win/ (currently 2.4.4, March 2013) and extract it somewhere. We'll refer to this folder as `<opencv_install_path>`:


7-Zip self-extracting archive

Extract to:

c:\lazy

1.1.Pre-built version

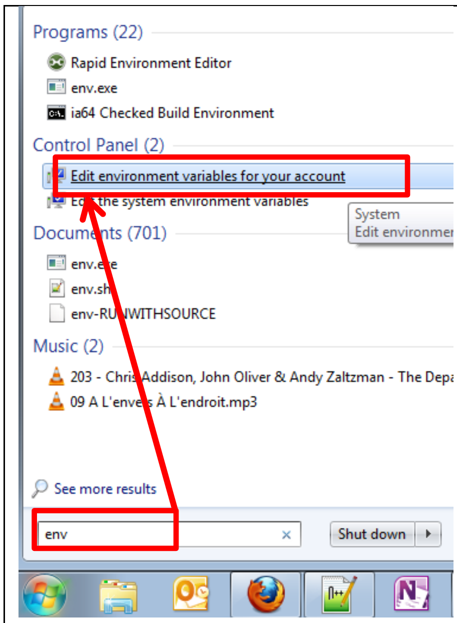
If you only need the pre-built version (i.e. if you won't be modifying the OpenCV library itself, don't need GPU support, and will only use it to link against your own applications), the library is now installed to e.g. `C:\lazy\opencv`.

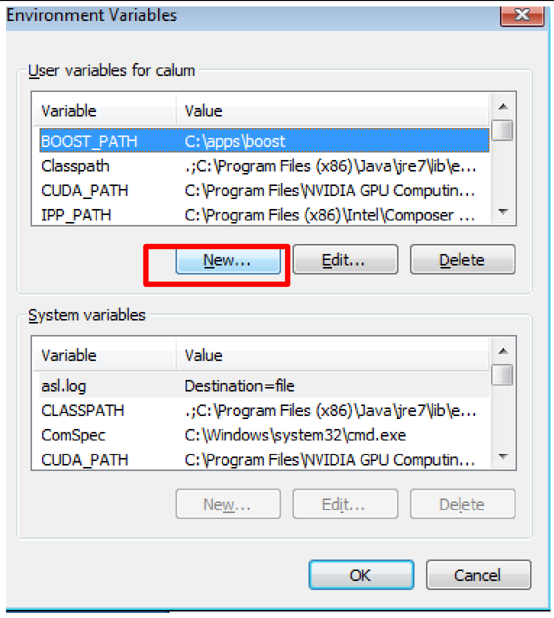
Set the `OpenCV_DIR` environment variable to the `build` folder under this: here, `C:\lazy\opencv\build`.

Add the folder containing the dynamic libraries to your system path. If using **Visual Studio 2012**, this will be:
`<opencv_install_path>\build\x86\vc11\bin`
For **Visual Studio 2010 or 2012**, this will be:
`<opencv_install_path>\build\x86\vc10\bin`
For **Visual Studio 2008**, this will be:
`<opencv_install_path>\build\x86\vc9\bin`

For 64-bit Operating Systems: Visual Studio generates 32-bit code by default, so unless you have downloaded the 64-bit compilers separately, you should still use the `x86` versions of the libraries.

For these steps, use an environment variable editor like [Rapid]() or use the method below:


Programs (22)
Rapid Environment Editor
env.exe
ia64 Checked Build Environment
Control Panel (2)
Edit environment variables for your account
Edit the system environment variables
System
Edit environmen
Documents (701)
env.exe
env.sh
env-RUNWITHSOURCE
Music (2)
203 - Chris Addison, John Oliver & Andy Zaltzman - The Depa
09 A L'envers À L'endroit.mp3
See more results
env
Shut down
Figure 1 Select 'edit environment variables'


Environment Variables

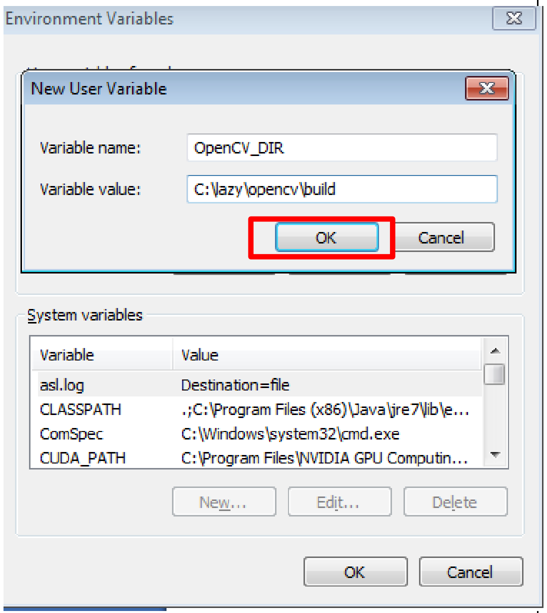
User variables for calum

Variable	Value
BOOST_PATH	C:\apps\boost
Classpath	.;C:\Program Files (x86)\Java\jre7\lib\e...
CUDA_PATH	C:\Program Files\NVIDIA GPU Computin...
IPP_PATH	C:\Program Files (x86)\Intel\Composer ...

System variables

Variable	Value
asl.log	Destination=file
CLASSPATH	.;C:\Program Files (x86)\Java\jre7\lib\e...
ComSpec	C:\Windows\system32\cmd.exe
CUDA_PATH	C:\Program Files\NVIDIA GPU Computin...

Figure 2 Choose 'New variable'

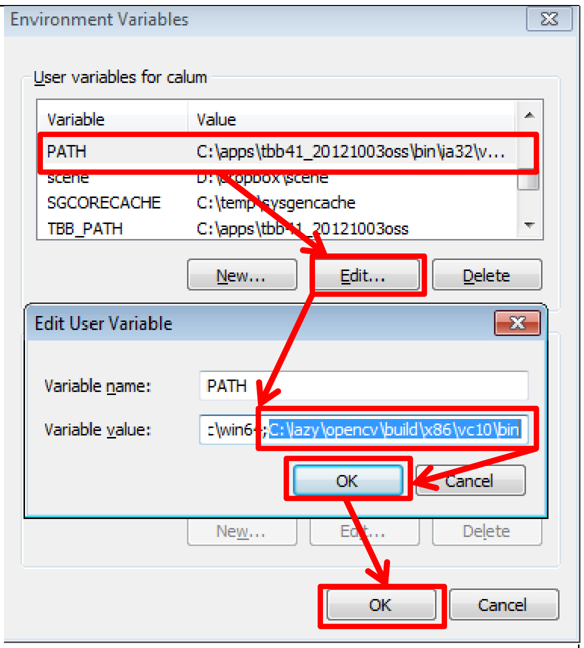


Variable name: OpenCV_DIR

Variable value: C:\lazy\opencv\build

Variable	Value
asl.log	Destination=file
CLASSPATH	.;C:\Program Files (x86)\Java\jre7\lib\e...
ComSpec	C:\Windows\system32\cmd.exe
CUDA_PATH	C:\Program Files\NVIDIA GPU Computin...

Figure 3 Enter name: OpenCV_DIR, value: `<opencv_install_path>\build`, then hit OK.



Variable	Value
PATH	C:\apps\tbb41_20121003oss\bin\ia32\v...
scene	D:\Dropbox\scene
SGCORECACHE	C:\temp\sysgencache
TBB_PATH	C:\apps\tbb41_20121003oss

Variable name: PATH

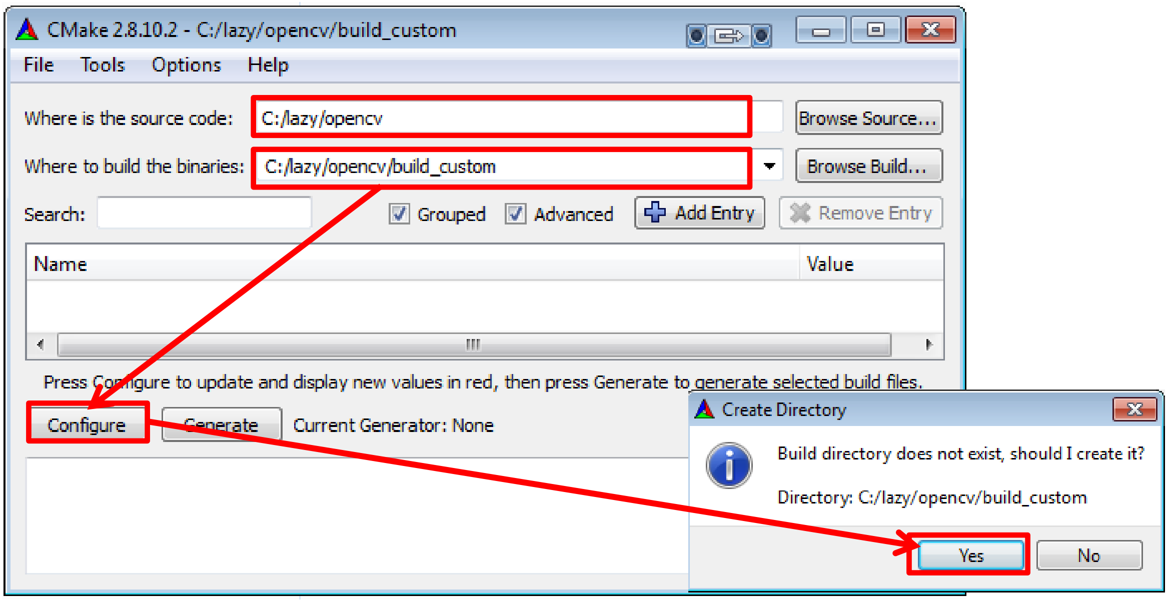
Variable value: :\win6...;C:\lazy\opencv\build\x86\vc10\bin

Figure 4 Add the appropriate path to the END of the value: `<opencv_install_path>\build\x86\vc10\bin` this time.

Done! Go to stage 2.

1.2.Building from source: configuration with CMake

For anything more complex than the basic stuff, we need to build from scratch. Open CMake and point it at the `<opencv_install_path>`. Choose a build directory. Anywhere except `build` is fine:


CMake 2.8.10.2 - C:/lazy/opencv/build_custom

Where is the source code: C:/lazy/opencv

Where to build the binaries: C:/lazy/opencv/build_custom

Grouped [✓] Advanced [✓]

Press Configure to update and display new values in red, then press Generate to generate selected build files.

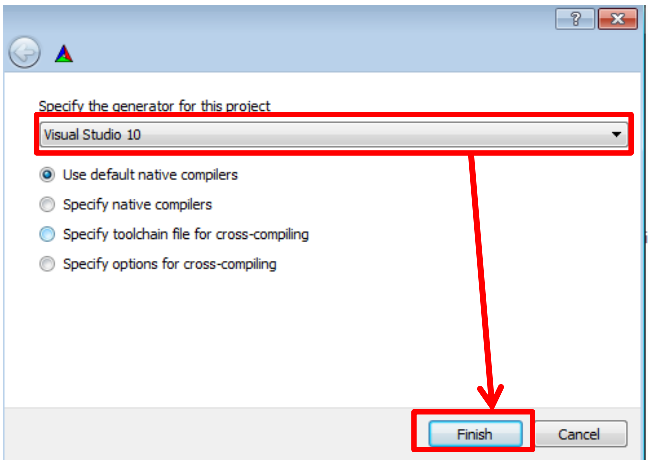
Current Generator: None

Create Directory

Build directory does not exist, should I create it?

Directory: C:/lazy/opencv/build_custom

Select the IDE version then hit finish:


Specify the generator for this project: Visual Studio 10

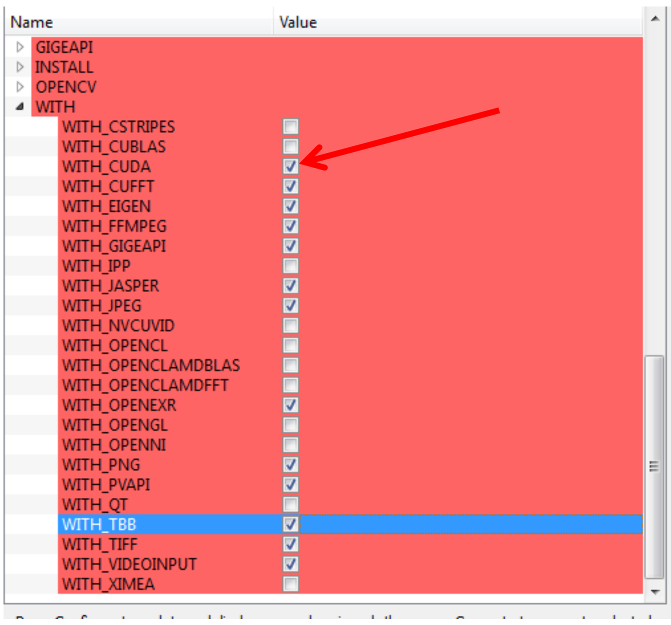
Use default native compilers [✓]

Specify native compilers

Specify toolchain file for cross-compiling

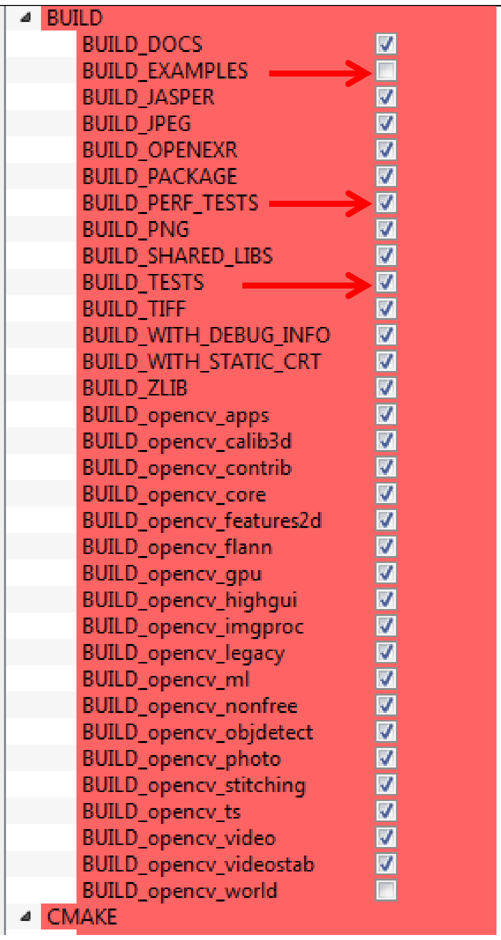
Specify options for cross-compiling

Select any extra features, then provide any extra info to make the red lines go away. This time we'll select **CUDA** and **TBB:**



Name	Value
GIGEAPI	
INSTALL	
OPENCV	
WITH	
WITH_CSTRIPES	☐
WITH_CUBLAS	☐
WITH_CUDA	✓
WITH_CUFFT	✓
WITH_EIGEN	✓
WITH_FFMPEG	✓
WITH_GIGEAPI	✓
WITH_IPP	☐
WITH_JASPER	✓
WITH_JPEG	✓
WITH_NVCUVID	☐
WITH_OPENCL	☐
WITH_OPENCLAMDBLAS	☐
WITH_OPENCLAMDFFT	☐
WITH_OPENEXR	✓
WITH_OPENGL	☐
WITH_OPENNI	☐
WITH_PNG	✓
WITH_PVAPI	✓
WITH_QT	☐
WITH_TBB	✓
WITH_TIFF	✓
WITH_VIDEOINPUT	✓
WITH_XIMEA	☐

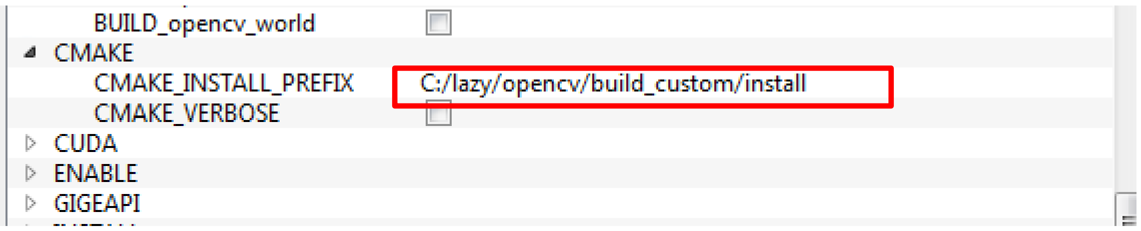
Figure 5 Select any extra features



BUILD	
BUILD_DOCS	✓
BUILD_EXAMPLES	☐
BUILD_JASPER	✓
BUILD_JPEG	✓
BUILD_OPENEXR	✓
BUILD_PACKAGE	✓
BUILD_PERF_TESTS	✓
BUILD_PNG	✓
BUILD_SHARED_LIBS	✓
BUILD_TESTS	✓
BUILD_TIFF	✓
BUILD_WITH_DEBUG_INFO	✓
BUILD_WITH_STATIC_CRT	✓
BUILD_ZLIB	✓
BUILD_opencv_apps	✓
BUILD_opencv_calib3d	✓
BUILD_opencv_contrib	✓
BUILD_opencv_core	✓
BUILD_opencv_features2d	✓
BUILD_opencv_flann	✓
BUILD_opencv_gpu	✓
BUILD_opencv_highgui	✓
BUILD_opencv_imgproc	✓
BUILD_opencv_legacy	✓
BUILD_opencv_ml	✓
BUILD_opencv_nonfree	✓
BUILD_opencv_objdetect	✓
BUILD_opencv_photo	✓
BUILD_opencv_stitching	✓
BUILD_opencv_ts	✓
BUILD_opencv_video	✓
BUILD_opencv_videostab	✓
BUILD_opencv_world	☐
CMAKE	

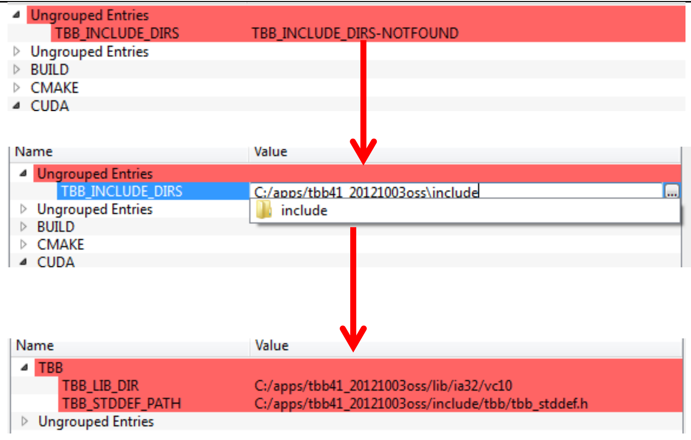
Figure 6: I prefer to disable tests and performance tests for faster builds. Enable BUILD_EXAMPLES to build various demos (takes a while).

IMPORTANT: note the install location. By default it's `<opencv_install_path>\<build_location>\install`. We'll need this later.



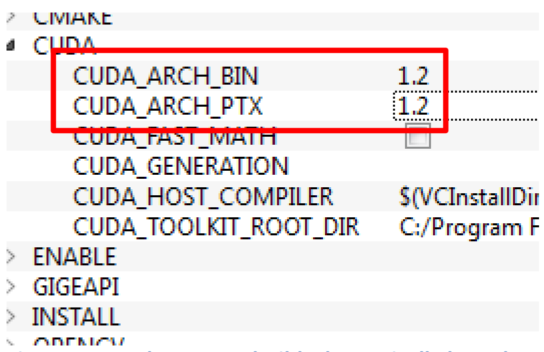
BUILD_opencv_world	☐
CMAKE	
CMAKE_INSTALL_PREFIX	C:/lazy/opencv/build_custom/install
CMAKE_VERBOSE	☐
CUDA	
ENABLE	
GIGEAPI	

Press Configure again to make the red backgrounds go away. Anything with red text is something that needs fixing. Recent versions of CMake are generally quite good at finding library locations:



```
◢ Ungrouped Entries
    TBB_INCLUDE_DIRS          TBB_INCLUDE_DIRS-NOTFOUND
▷ Ungrouped Entries
▷ BUILD
▷ CMAKE
◢ CUDA

Name                          Value
◢ Ungrouped Entries
    TBB_INCLUDE_DIRS          C:/apps/tbb41_20121003oss\include
▷ Ungrouped Entries             include
▷ BUILD
▷ CMAKE
◢ CUDA

Name                          Value
◢ TBB
    TBB_LIB_DIR               C:/apps/tbb41_20121003oss/lib/ia32/vc10
    TBB_STDDEF_PATH           C:/apps/tbb41_20121003oss/include/tbb/tbb_stddef.h
▷ Ungrouped Entries
```

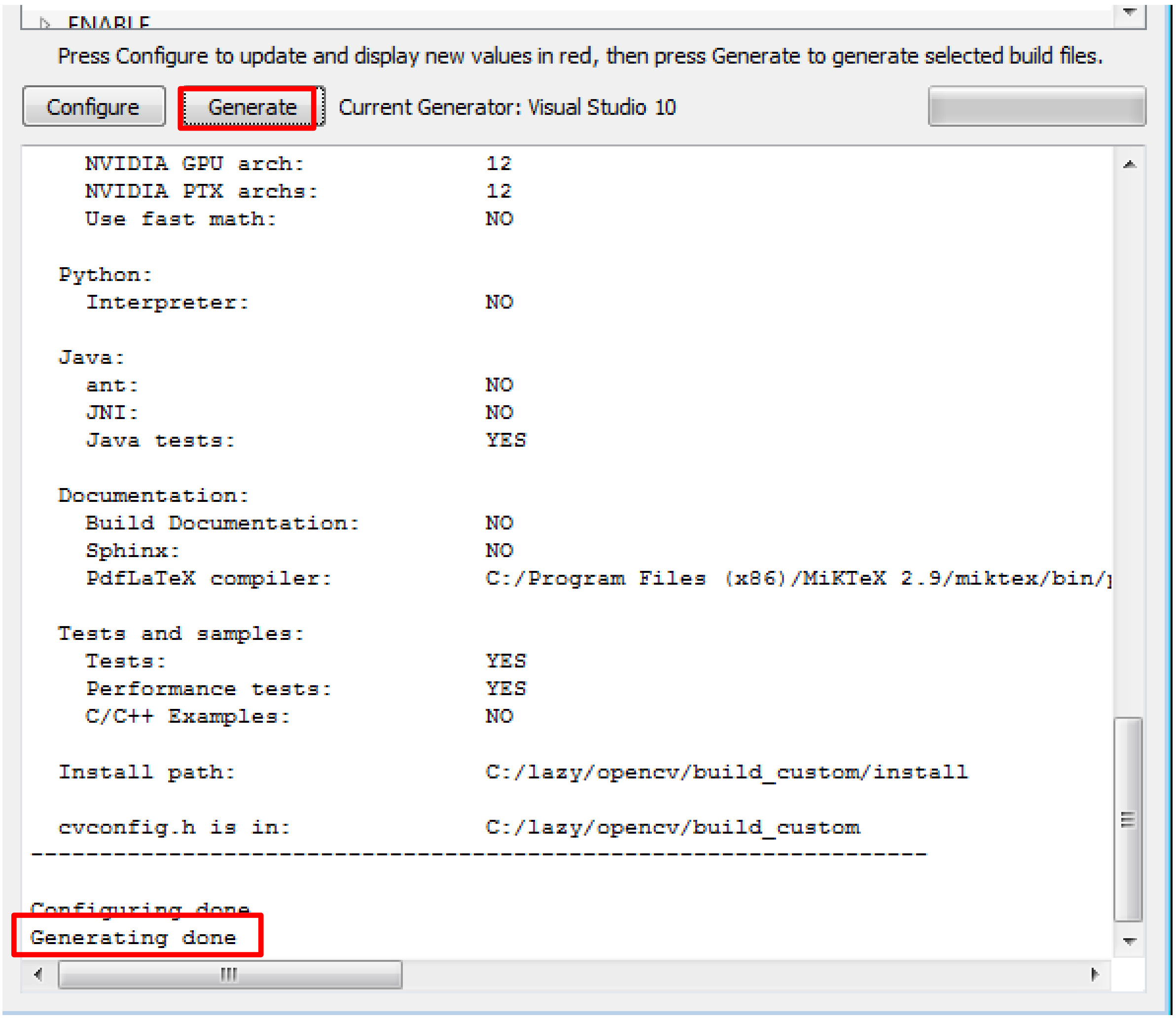
Figure 7 Fix any remaining red text by supplying the right paths. These will show up correctly once we press configure again.



```
› CMAKE
◢ CUDA
    CUDA_ARCH_BIN              1.2
    CUDA_ARCH_PTX              1.2
    CUDA_FAST_MATH             ☐
    CUDA_GENERATION
    CUDA_HOST_COMPILER         $(VCInstallDir
    CUDA_TOOLKIT_ROOT_DIR      C:/Program F
› ENABLE
› GIGEAPI
› INSTALL
› OPENCV
```

Figure 8 Speed up CUDA builds dramatically by only building for the card you have: this machine has compute capability 1.2

Finally, press Configure again. If there are no remaining red boxes or red text, check the options match what you selected and hit **generate**. This generates Visual Studio project files in the build directory. If this works, you should see 'generating done' at the bottom of the window. On to 1.3!

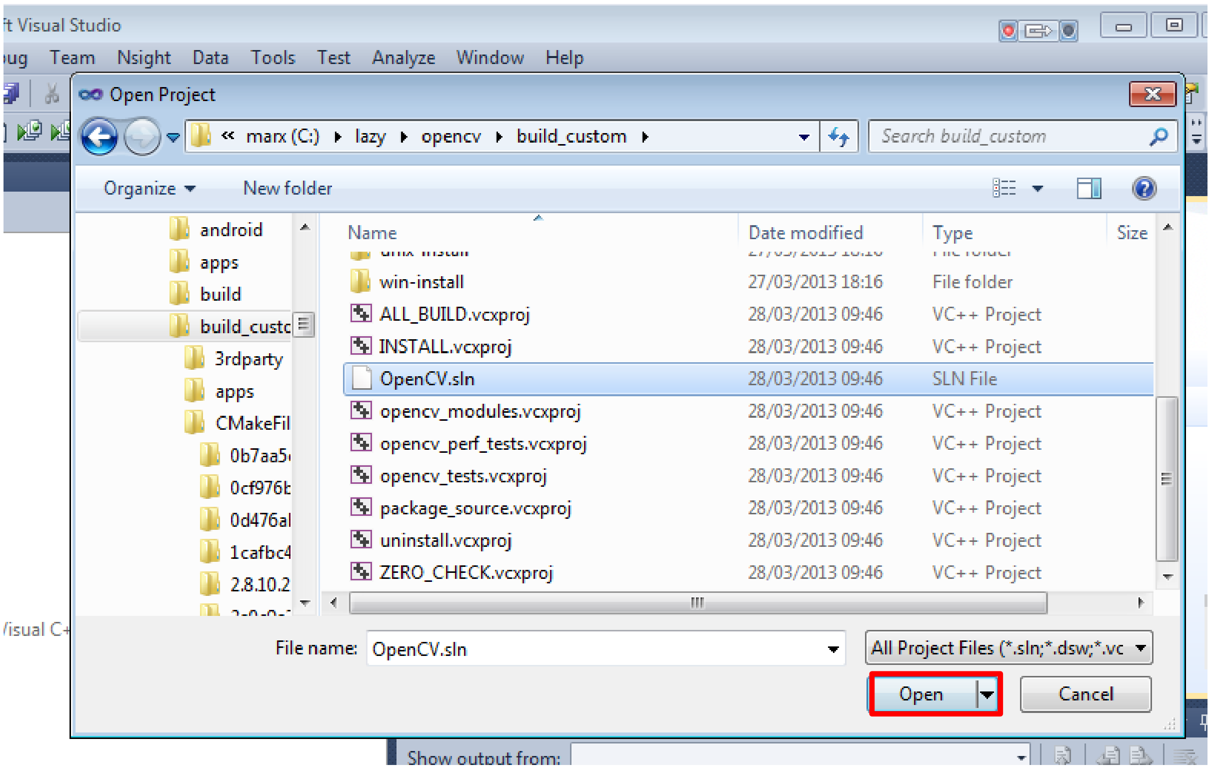


Press Configure to update and display new values in red, then press Generate to generate selected build files.

```
    NVIDIA GPU arch:               12
    NVIDIA PTX archs:              12
    Use fast math:                 NO

  Python:
    Interpreter:                   NO

  Java:
    ant:                           NO
    JNI:                           NO
    Java tests:                    YES

  Documentation:
    Build Documentation:           NO
    Sphinx:                        NO
    PdfLaTeX compiler:             C:/Program Files (x86)/MiKTeX 2.9/miktex/bin/

  Tests and samples:
    Tests:                         YES
    Performance tests:             YES
    C/C++ Examples:                NO

  Install path:                    C:/lazy/opencv/build_custom/install

  cvconfig.h is in:                C:/lazy/opencv/build_custom
-----------------------------------------------------------------

Configuring done
Generating done
```

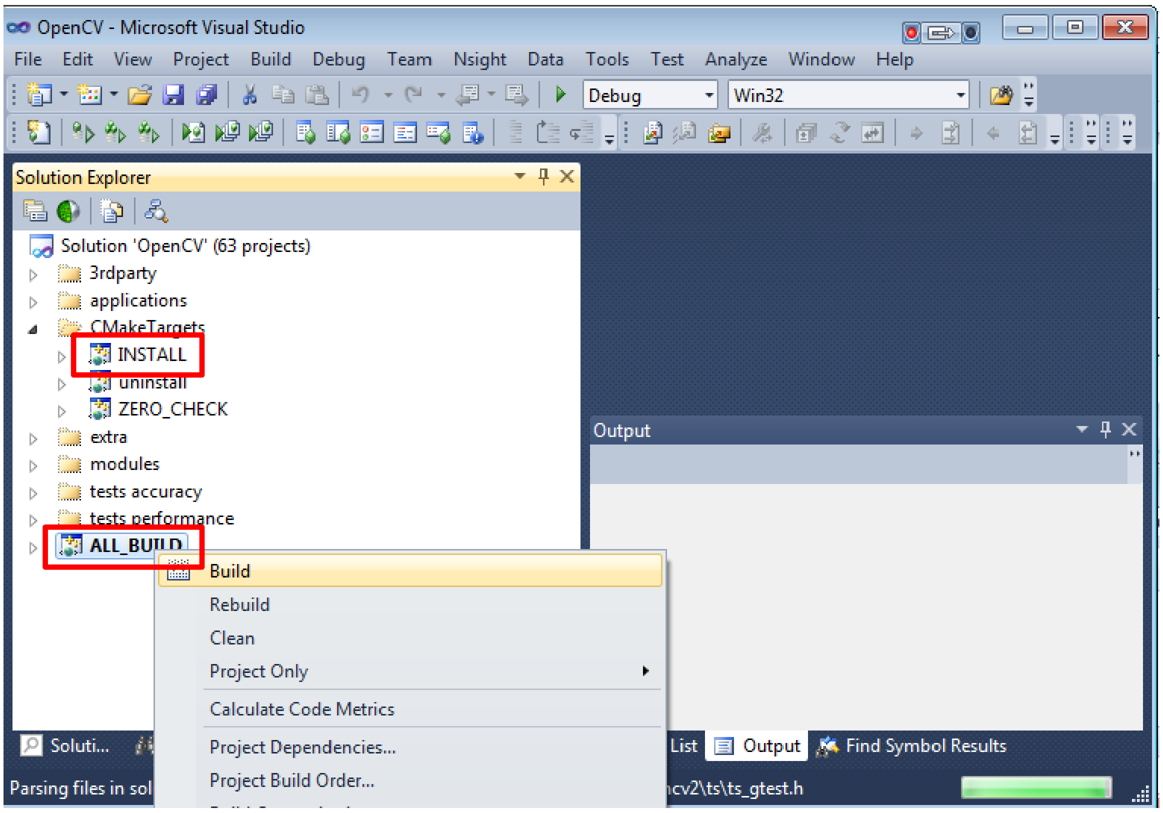
1.3.Building from source: compiling in Visual Studio

In Visual Studio, open the resulting solution from the build folder. (I prefer to rename the solution at this point to make it easier to distinguish different custom builds):

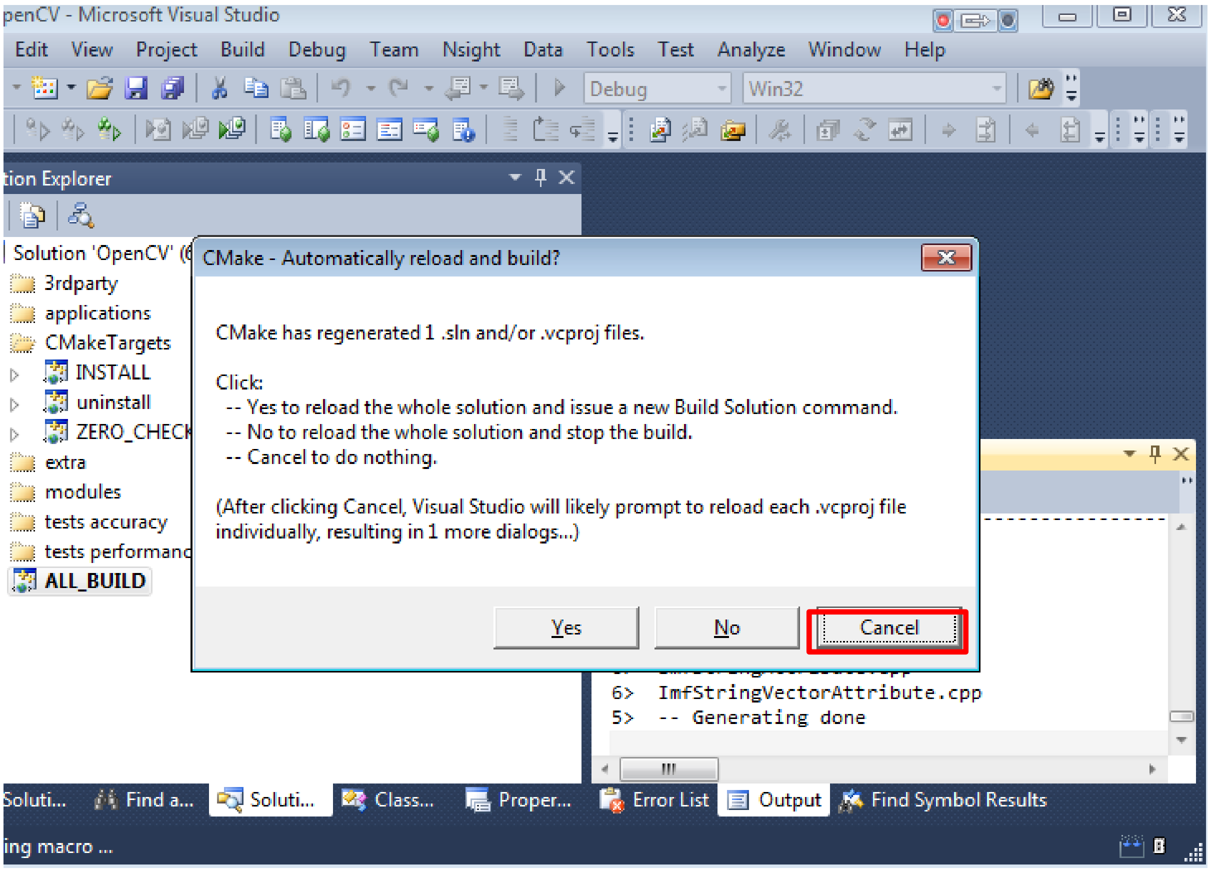


Name	Date modified	Type
win-install	27/03/2013 18:16	File folder
ALL_BUILD.vcxproj	28/03/2013 09:46	VC++ Project
INSTALL.vcxproj	28/03/2013 09:46	VC++ Project
OpenCV.sln	28/03/2013 09:46	SLN File
opencv_modules.vcxproj	28/03/2013 09:46	VC++ Project
opencv_perf_tests.vcxproj	28/03/2013 09:46	VC++ Project
opencv_tests.vcxproj	28/03/2013 09:46	VC++ Project
package_source.vcxproj	28/03/2013 09:46	VC++ Project
uninstall.vcxproj	28/03/2013 09:46	VC++ Project
ZERO_CHECK.vcxproj	28/03/2013 09:46	VC++ Project

When the solution loads, there are two important projects: **ALL_BUILD** and **INSTALL**. First build the **ALL_BUILD** project. This takes a while, particularly with GPU builds:



If you get any messages from CMake asking to reload, just hit ‘Cancel’:



CMake - Automatically reload and build?

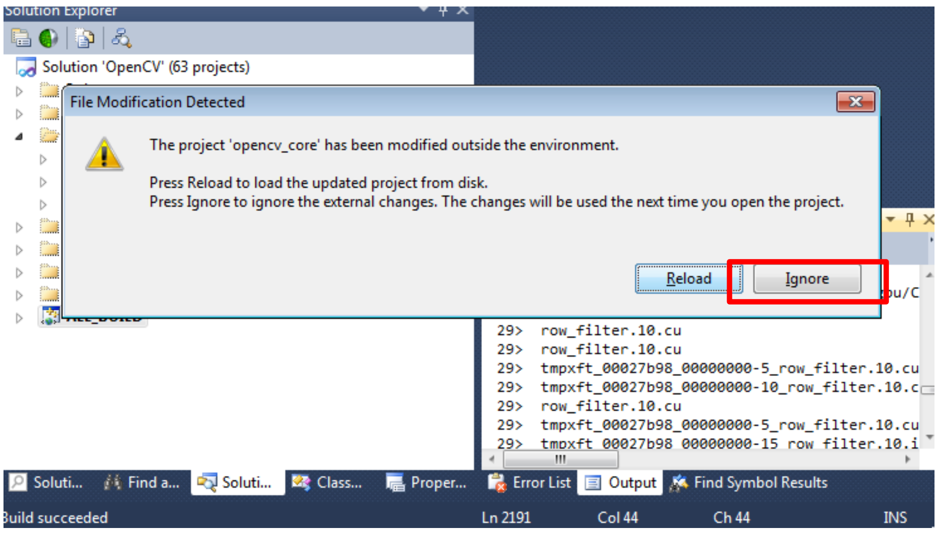
CMake has regenerated 1 .sln and/or .vcproj files.

Click:
-- Yes to reload the whole solution and issue a new Build Solution command.
-- No to reload the whole solution and stop the build.
-- Cancel to do nothing.

(After clicking Cancel, Visual Studio will likely prompt to reload each .vcproj file individually, resulting in 1 more dialogs...)

```
6>  ImfStringVectorAttribute.cpp
5>  -- Generating done
```

Similarly , ignore any messages about individual projects:



File Modification Detected

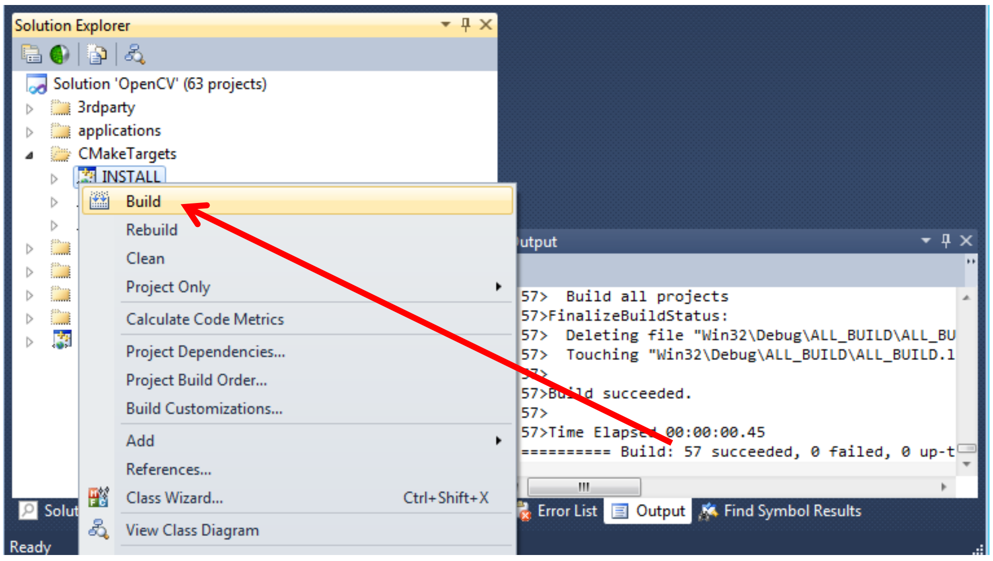
The project 'opencv_core' has been modified outside the environment.

Press Reload to load the updated project from disk.
Press Ignore to ignore the external changes. The changes will be used the next time you open the project.

```
29>  row_filter.10.cu
29>  row_filter.10.cu
29>  tmpxft_00027b98_00000000-5_row_filter.10.cu
29>  tmpxft_00027b98_00000000-10_row_filter.10.c
29>  row_filter.10.cu
29>  tmpxft_00027b98_00000000-5_row_filter.10.cu
29>  tmpxft_00027b98_00000000-15_row_filter.10.i
```

Build succeeded

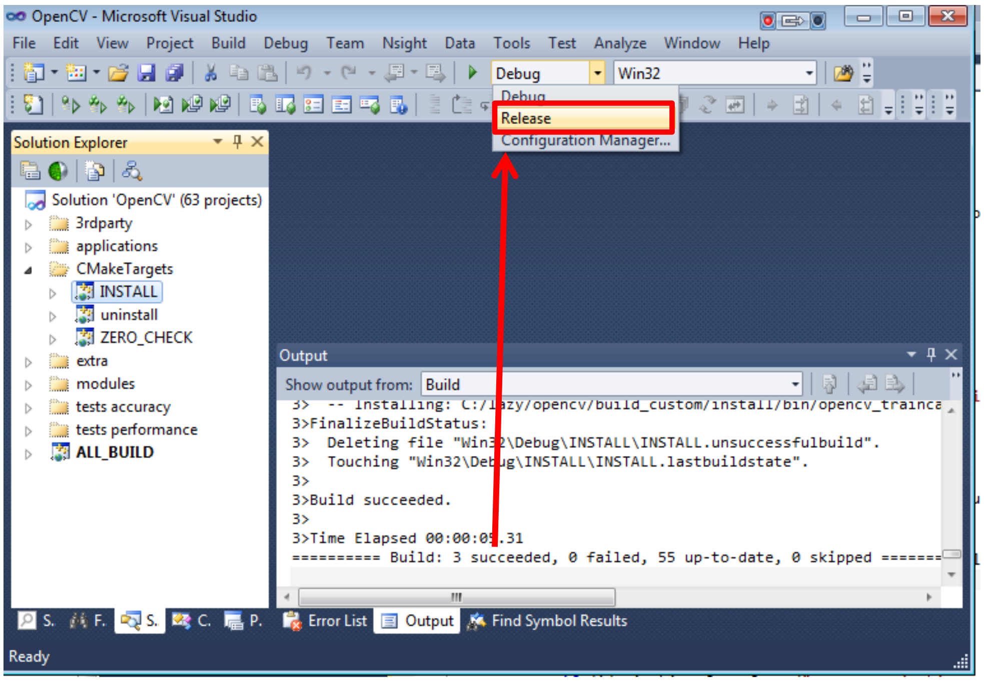
Finally, once **ALL_BUILD** shows as building successfully, build the **INSTALL** project:



```
57>  Build all projects
57>FinalizeBuildStatus:
57>  Deleting file "Win32\Debug\ALL_BUILD\ALL_BU
57>  Touching "Win32\Debug\ALL_BUILD\ALL_BUILD.l
57>
57>Build succeeded.
57>
57>Time Elapsed 00:00:00.45
========== Build: 57 succeeded, 0 failed, 0 up-t
```

This installs the headers, static and dynamic libraries to the install location set in step 1.2. Here, it's `C:\lazy\opencv\build_custom\install`.

Once the install is finished, switch to **Release** mode and build the **ALL_BUILD** and **INSTALL** projects in the same way. (You can also automate all of this with **Build→Batch build…**)



```
3>  -- Installing: C:/lazy/opencv/build_custom/install/bin/opencv_trainca
3>FinalizeBuildStatus:
3>  Deleting file "Win32\Debug\INSTALL\INSTALL.unsuccessfulbuild".
3>  Touching "Win32\Debug\INSTALL\INSTALL.lastbuildstate".
3>
3>Build succeeded.
3>
3>Time Elapsed 00:00:05.31
========== Build: 3 succeeded, 0 failed, 55 up-to-date, 0 skipped =======
```

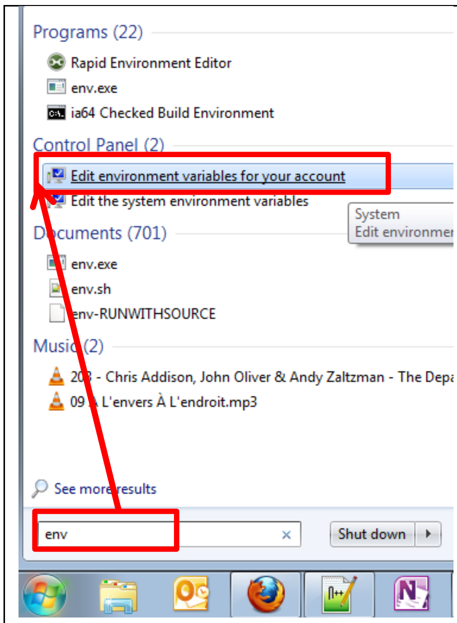
Once the release mode binaries have been built and installed, go to 1.4.

1.4. Building from source: setting environment variables

Set the `OpenCV_DIR` environment variable to the folder you *installed* OpenCV to in the previous step (`<opencv_install_path>\<build_location>\install` from above). Here, this is `C:\lazy\opencv\build_custom\install`.

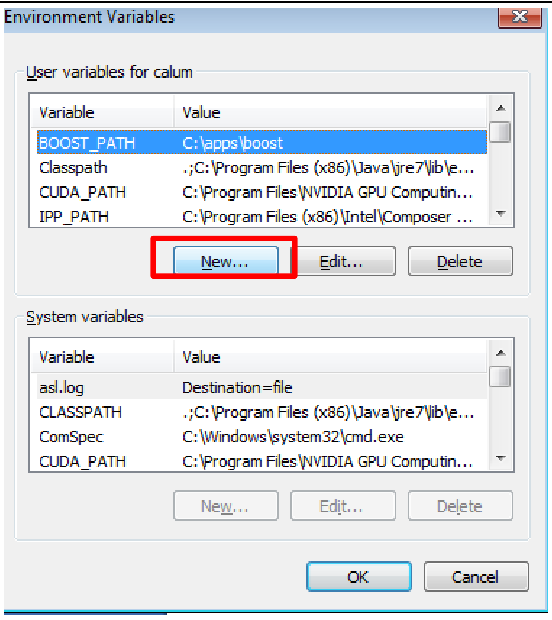
Add the folder containing the dynamic libraries to your system path. This is `<opencv_install_path>\<build_location>\install\bin`, or `C:\lazy\opencv\build_custom\install\bin` here.

For these steps, use an environment variable editor like Rapid or use the method below:



Programs (22)
Rapid Environment Editor
env.exe
ia64 Checked Build Environment
Control Panel (2)
Edit environment variables for your account
Edit the system environment variables
System
Edit environmen
Documents (701)
env.exe
env.sh
env-RUNWITHSOURCE
Music (2)
203 - Chris Addison, John Oliver & Andy Zaltzman - The Depa
09 - L'envers À L'endroit.mp3
See more results
env
Shut down

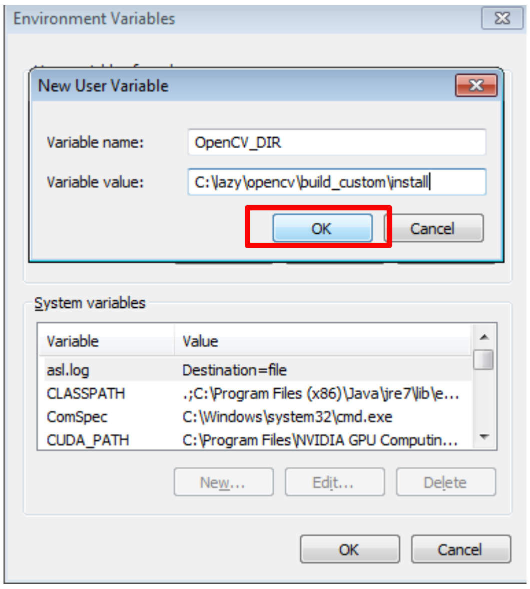
Figure 9 Select 'edit environment variables'



Variable	Value
BOOST_PATH	C:\apps\boost
Classpath	.;C:\Program Files (x86)\Java\jre7\lib\e...
CUDA_PATH	C:\Program Files\NVIDIA GPU Computin...
IPP_PATH	C:\Program Files (x86)\Intel\Composer ...

Variable	Value
asl.log	Destination=file
CLASSPATH	.;C:\Program Files (x86)\Java\jre7\lib\e...
ComSpec	C:\Windows\system32\cmd.exe
CUDA_PATH	C:\Program Files\NVIDIA GPU Computin...

Figure 10 Choose 'New variable'

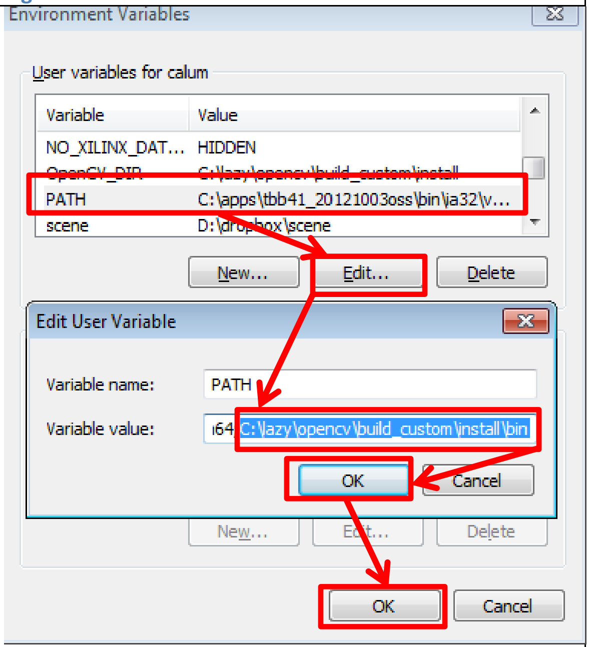


Variable name: OpenCV_DIR

Variable value: C:\lazy\opencv\build_custom\install

Variable	Value
asl.log	Destination=file
CLASSPATH	.;C:\Program Files (x86)\Java\jre7\lib\e...
ComSpec	C:\Windows\system32\cmd.exe
CUDA_PATH	C:\Program Files\NVIDIA GPU Computin...

Figure 11 Enter name: OpenCV_DIR, value: `<opencv_install_path>\<build_location>\install`, then hit OK.



Variable	Value
NO_XILINX_DAT...	HIDDEN
OpenCV_DIR	C:\lazy\opencv\build_custom\install
PATH	C:\apps\tbb41_20121003oss\bin\ia32\v...
scene	D:\dropbox\scene

Variable name: PATH

Variable value: i64;C:\lazy\opencv\build_custom\install\bin

Figure 12 Add the appropriate path to the END of the value: `C:\lazy\opencv\build_custom\install\bin` this time.

Done! Go to stage 2.

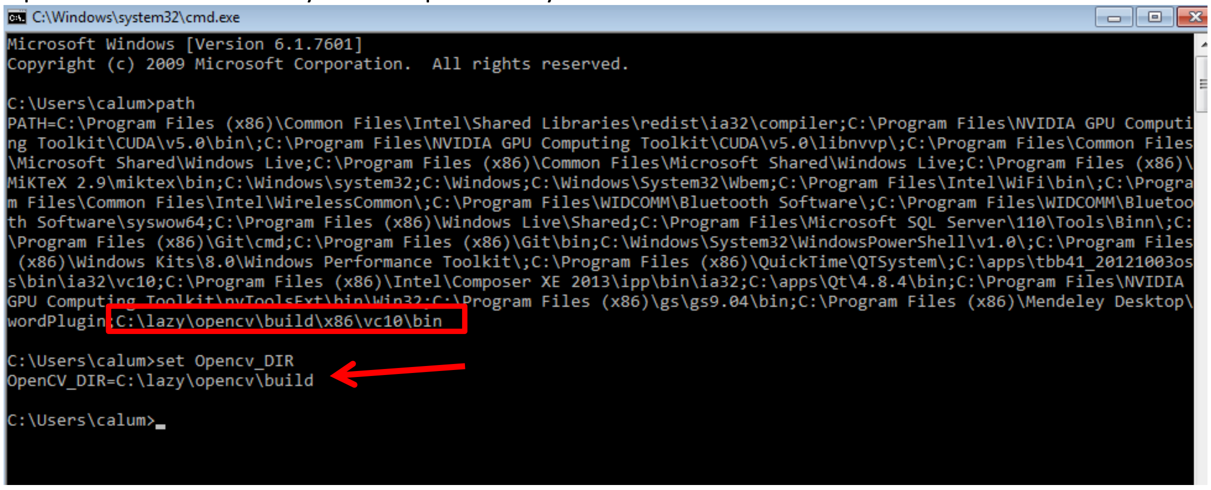
2. Configuring a Property Sheet

2.1. Using a pre-built OpenCV build

Open a command window and enter these commands:

```
PATH

set OpenCV_DIR
```

And check the output. Ensure the `OpenCV_DIR` environment variable is set correctly, and that the OpenCV binaries directory from step 1.1 is in your `PATH`:



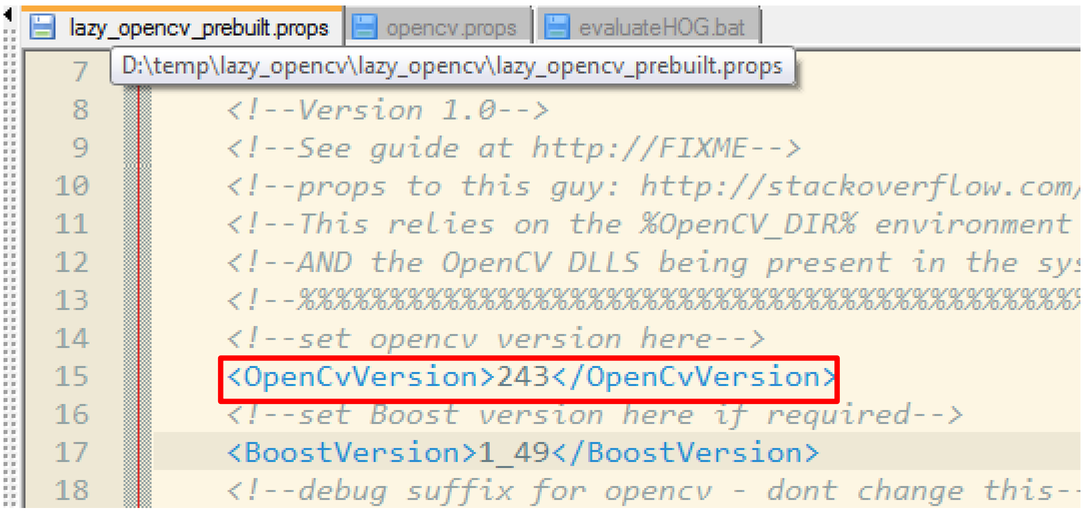
C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\calum>path
PATH=C:\Program Files (x86)\Common Files\Intel\Shared Libraries\redist\ia32\compiler;C:\Program Files\NVIDIA GPU Computi
ng Toolkit\CUDA\v5.0\bin\;C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v5.0\libnvvp\;C:\Program Files\Common Files
\Microsoft Shared\Windows Live;C:\Program Files (x86)\Common Files\Microsoft Shared\Windows Live;C:\Program Files (x86)\
MiKTeX 2.9\miktex\bin;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Program Files\Intel\WiFi\bin\;C:\Progra
m Files\Common Files\Intel\WirelessCommon\;C:\Program Files\WIDCOMM\Bluetooth Software\;C:\Program Files\WIDCOMM\Bluetoo
th Software\syswow64;C:\Program Files (x86)\Windows Live\Shared;C:\Program Files\Microsoft SQL Server\110\Tools\Binn\;C:
\Program Files (x86)\Git\cmd;C:\Program Files (x86)\Git\bin;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files
 (x86)\Windows Kits\8.0\Windows Performance Toolkit\;C:\Program Files (x86)\QuickTime\QTSystem\;C:\apps\tbb41_20121003os
s\bin\ia32\vc10;C:\Program Files (x86)\Intel\Composer XE 2013\ipp\bin\ia32;C:\apps\Qt\4.8.4\bin;C:\Program Files\NVIDIA
GPU Computing Toolkit\nvToolsExt\bin\Win32;C:\Program Files (x86)\gs\gs9.04\bin;C:\Program Files (x86)\Mendeley Desktop\
wordPlugin;C:\lazy\opencv\build\x86\vc10\bin

C:\Users\calum>set Opencv_DIR
OpenCV_DIR=C:\lazy\opencv\build

C:\Users\calum>_
```

If either of these are missing, log off, log on and try again.

Next, download (right-click, save as) the `lazy_opencv_prebuilt.props` file from here. (This link doesn't work in Adobe Reader 11. Copy and paste this instead: http://home.eps.hw.ac.uk/~cgb7/opencv/lazy_opencv_prebuilt.props)

Open it in a text editor and check that the OpenCV version matches yours (2.4.4) in this example. Change it if not.



lazy_opencv_prebuilt.props
opencv.props
evaluateHOG.bat
D:\temp\lazy_opencv\lazy_opencv\lazy_opencv_prebuilt.props

```
<!--Version 1.0-->
<!--See guide at http://FIXME-->
<!--props to this guy: http://stackoverflow.com
<!--This relies on the %OpenCV_DIR% environment
<!--AND the OpenCV DLLS being present in the sy
<!--%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
<!--set opencv version here-->
<OpenCvVersion>243</OpenCvVersion>
<!--set Boost version here if required-->
<BoostVersion>1_49</BoostVersion>
<!--debug suffix for opencv - dont change this--
```

Done! Go to stage 3.

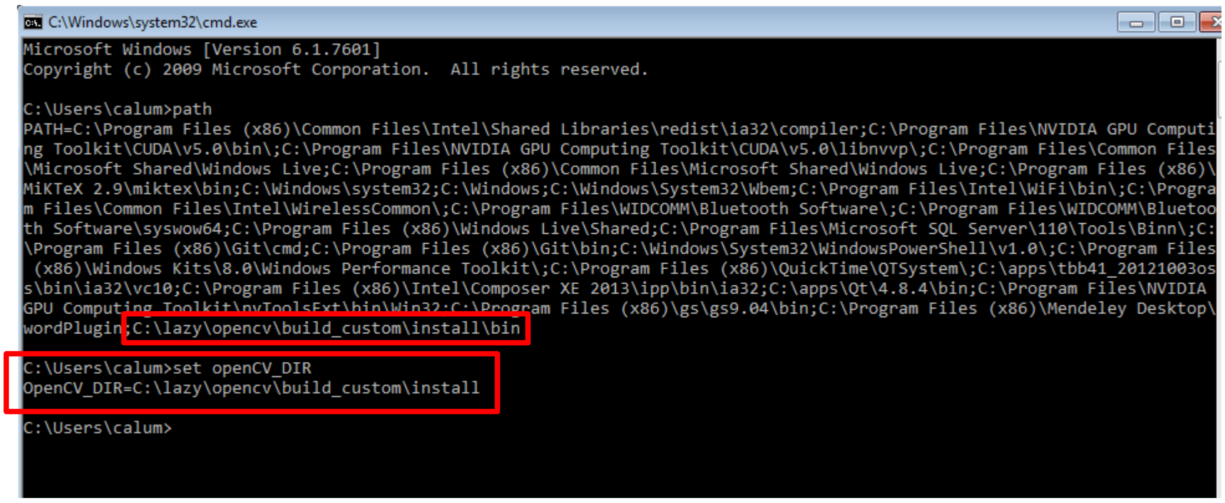
2.2. Using OpenCV built from source

Open a command window and enter these commands:

```
PATH
```

```
set OpenCV_DIR
```

And check the output. Ensure the `OpenCV_DIR` environment variable is set correctly, and that the OpenCV binaries directory from step 1.3 is in your `PATH`:

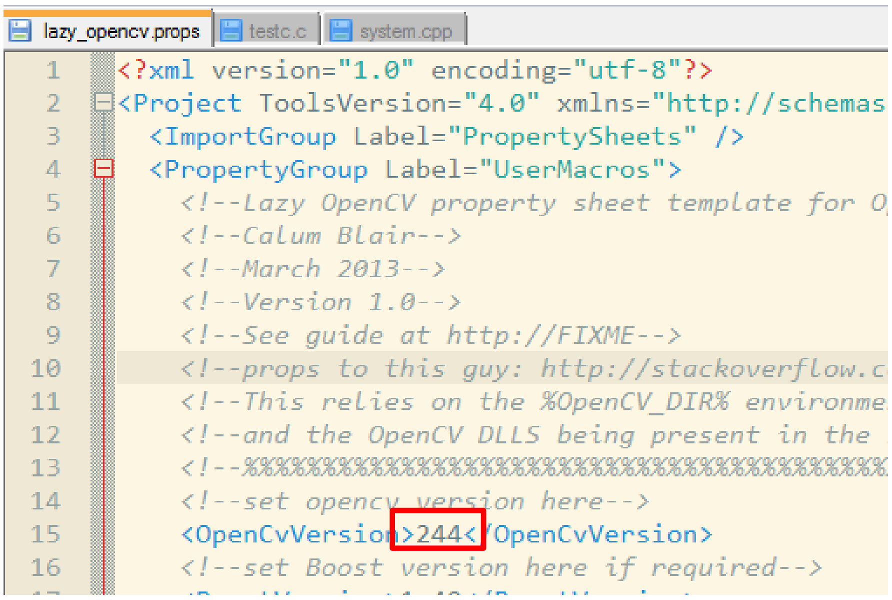


```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\calum>path
PATH=C:\Program Files (x86)\Common Files\Intel\Shared Libraries\redist\ia32\compiler;C:\Program Files\NVIDIA GPU Computi
ng Toolkit\CUDA\v5.0\bin\;C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v5.0\libnvvp\;C:\Program Files\Common Files
\Microsoft Shared\Windows Live;C:\Program Files (x86)\Common Files\Microsoft Shared\Windows Live;C:\Program Files (x86)\
MiKTeX 2.9\miktex\bin;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Program Files\Intel\WiFi\bin\;C:\Progra
m Files\Common Files\Intel\WirelessCommon\;C:\Program Files\WIDCOMM\Bluetooth Software\;C:\Program Files\WIDCOMM\Bluetoo
th Software\syswow64;C:\Program Files (x86)\Windows Live\Shared;C:\Program Files\Microsoft SQL Server\110\Tools\Binn\;C:
\Program Files (x86)\Git\cmd;C:\Program Files (x86)\Git\bin;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files
 (x86)\Windows Kits\8.0\Windows Performance Toolkit\;C:\Program Files (x86)\QuickTime\QTSystem\;C:\apps\tbb41_20121003os
s\bin\ia32\vc10;C:\Program Files (x86)\Intel\Composer XE 2013\ipp\bin\ia32;C:\apps\Qt\4.8.4\bin;C:\Program Files\NVIDIA
GPU Computing Toolkit\nvToolsExt\bin\Win32;C:\Program Files (x86)\gs\gs9.04\bin;C:\Program Files (x86)\Mendeley Desktop\
wordPlugin;C:\lazy\opencv\build_custom\install\bin

C:\Users\calum>set openCV_DIR
OpenCV_DIR=C:\lazy\opencv\build_custom\install

C:\Users\calum>
```

If either of these are missing, log off, log on and try again.

Next, download (right-click, save as) and save the `lazy_opencv.props` file from here. (This link doesn't work in Adobe Reader 11. Copy and paste this instead: http://home.eps.hw.ac.uk/~cgb7/opencv/lazy_opencv.props)

Open it in a text editor, and check the OpenCV version matches yours (2.4.4) in this example. Change it if not. If using GPU, check your compute capability matches what's in the file (1.2 here).



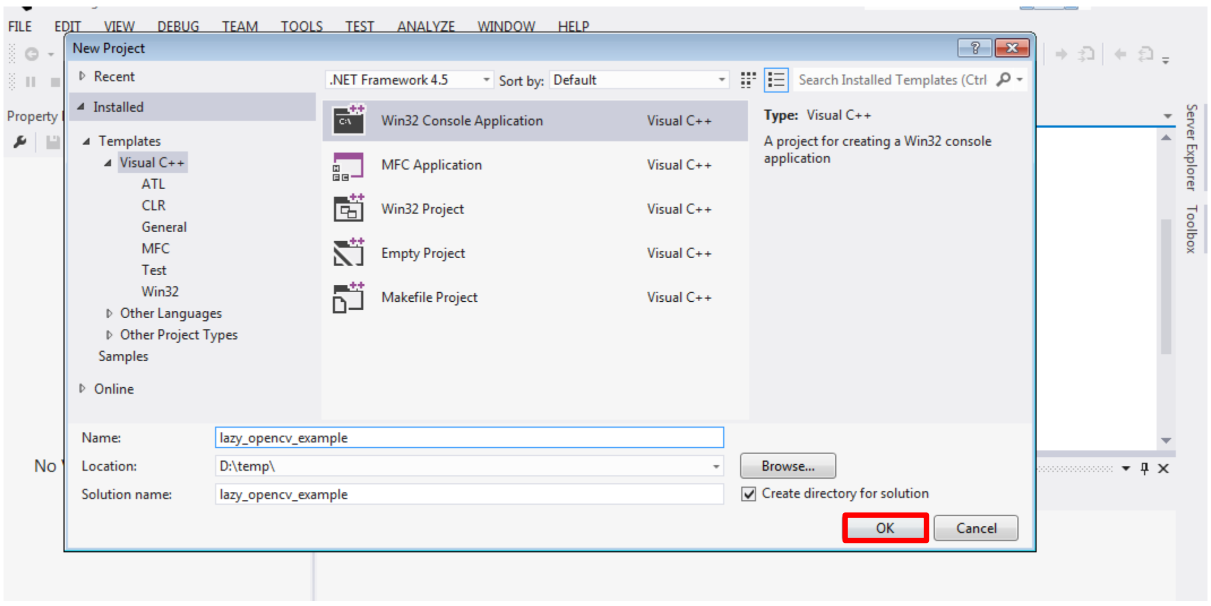
```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="4.0" xmlns="http://schemas
  <ImportGroup Label="PropertySheets" />
  <PropertyGroup Label="UserMacros">
    <!--Lazy OpenCV property sheet template for O
    <!--Calum Blair-->
    <!--March 2013-->
    <!--Version 1.0-->
    <!--See guide at http://FIXME-->
    <!--props to this guy: http://stackoverflow.c
    <!--This relies on the %OpenCV_DIR% environme
    <!--and the OpenCV DLLS being present in the
    <!--%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    <!--set opencv version here-->
    <OpenCvVersion>244</OpenCvVersion>
    <!--set Boost version here if required-->
```

Done! Go to stage 3.

3. Using a Property Sheet

3.1. An example project

Start a new console project in Visual Studio. Untick the ‘precompiled header’ option.


New Project

Installed > Templates > Visual C++

.NET Framework 4.5 | Sort by: Default

Win32 Console Application	Visual C++
MFC Application	Visual C++
Win32 Project	Visual C++
Empty Project	Visual C++
Makefile Project	Visual C++

Type: Visual C++

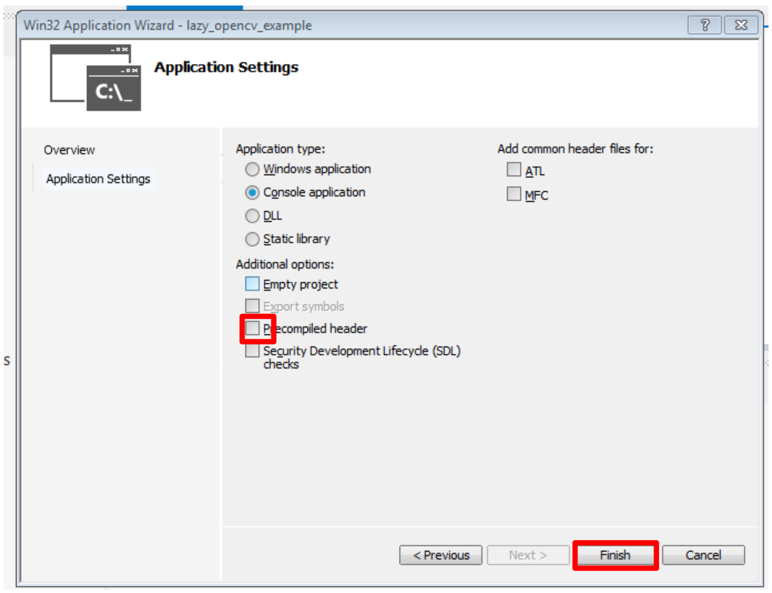
A project for creating a Win32 console application

Name: lazy_opencv_example

Location: D:\temp\

Solution name: lazy_opencv_example

[✓] Create directory for solution


Win32 Application Wizard - lazy_opencv_example

Application Settings

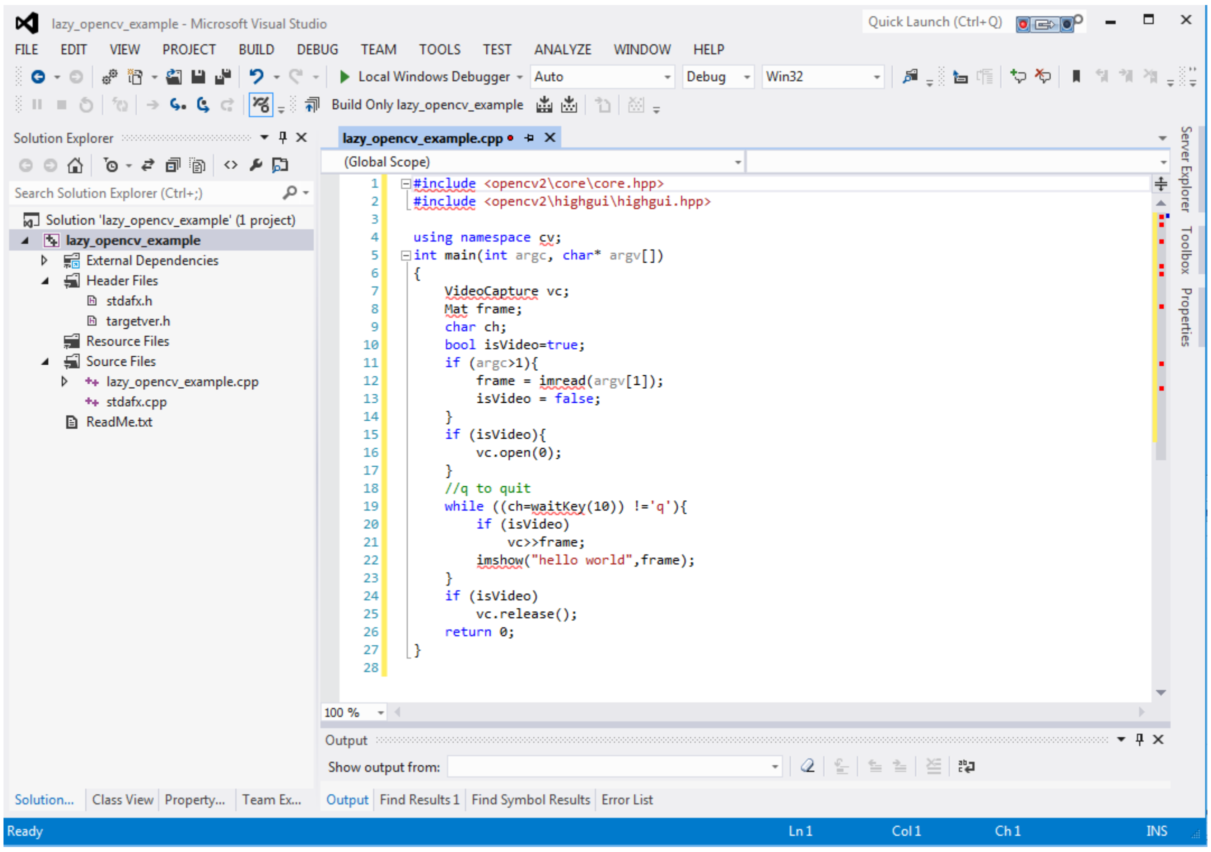
Application type:
- ( ) Windows application
- (●) Console application
- ( ) DLL
- ( ) Static library

Additional options:
- [ ] Empty project
- [ ] Export symbols
- [ ] Precompiled header
- [ ] Security Development Lifecycle (SDL) checks

Add common header files for:
- [ ] ATL
- [ ] MFC

```
#include <opencv2\core\core.hpp>
#include <opencv2\highgui\highgui.hpp>

using namespace cv;
int main(int argc, char* argv[])
{
	VideoCapture vc;
	Mat frame;
	char ch;
	bool isVideo=true;
	try{
		vc.open(0);//open default camera
	}
	catch (Exception e){
		frame = Mat::eye(100,100,CV_32FC1);
		isVideo = false;
	}
	while ((ch=waitKey(10)) !='q'){
		if (isVideo)
			vc>>frame;
		imshow("hello world",frame);
	}
	if (isVideo)
		vc.release();
	return 0;
}
```

Now write a 'hello world' in the resulting project:

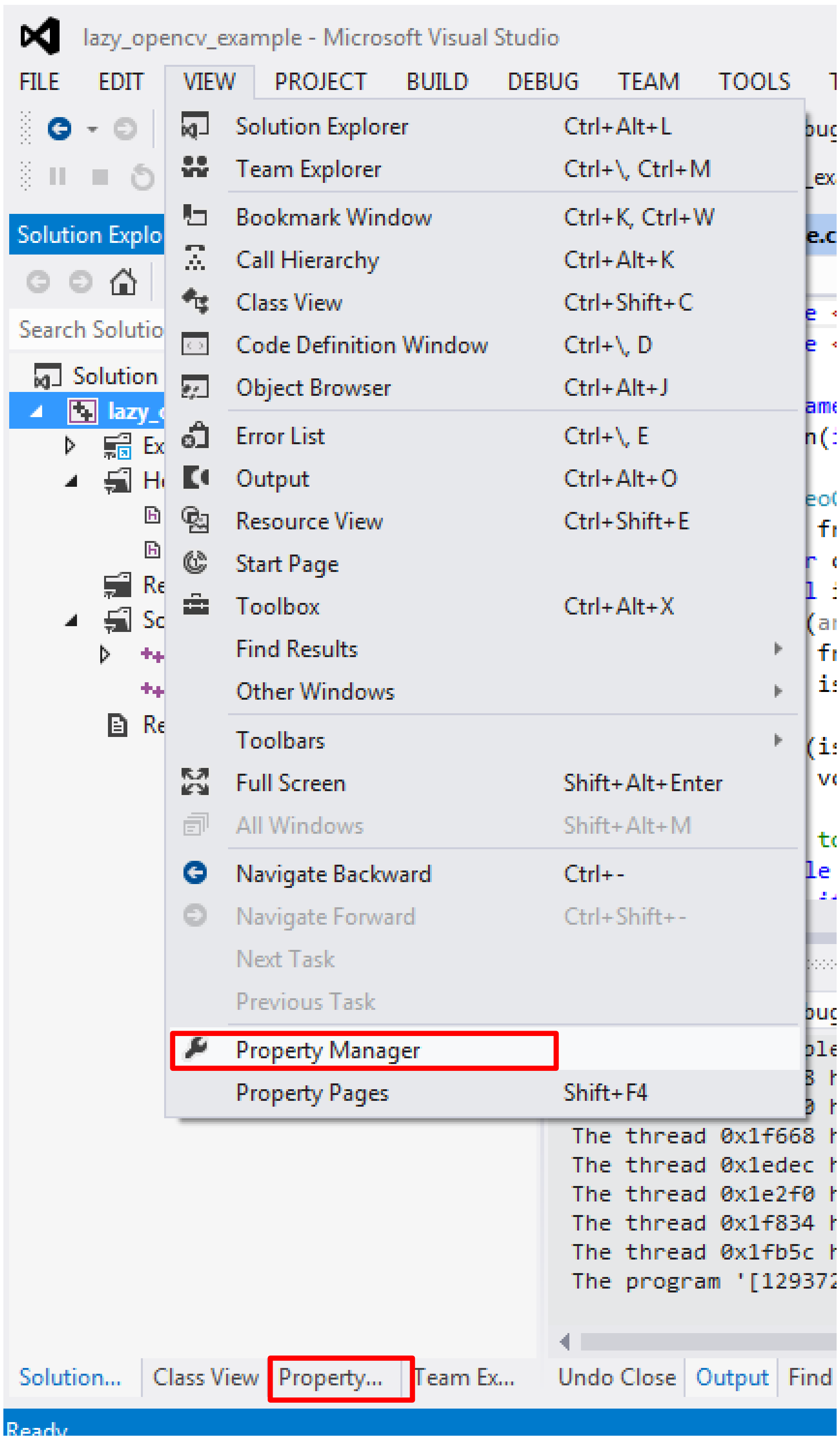

lazy_opencv_example - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Local Windows Debugger
Auto
Debug
Win32
Build Only lazy_opencv_example
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'lazy_opencv_example' (1 project)
lazy_opencv_example
External Dependencies
Header Files
stdafx.h
targetver.h
Resource Files
Source Files
lazy_opencv_example.cpp
stdafx.cpp
ReadMe.txt
lazy_opencv_example.cpp
(Global Scope)
100 %
Output
Show output from:
Solution... Class View Property... Team Ex...
Output Find Results 1 Find Symbol Results Error List
Ready Ln 1 Col 1 Ch 1 INS
Server Explorer Toolbox Properties

```
#include <opencv2\core\core.hpp>
#include <opencv2\highgui\highgui.hpp>

using namespace cv;
int main(int argc, char* argv[])
{
    VideoCapture vc;
    Mat frame;
    char ch;
    bool isVideo=true;
    if (argc>1){
        frame = imread(argv[1]);
        isVideo = false;
    }
    if (isVideo){
        vc.open(0);
    }
    //q to quit
    while ((ch=waitKey(10)) !='q'){
        if (isVideo)
            vc>>frame;
        imshow("hello world",frame);
    }
    if (isVideo)
        vc.release();
    return 0;
}
```

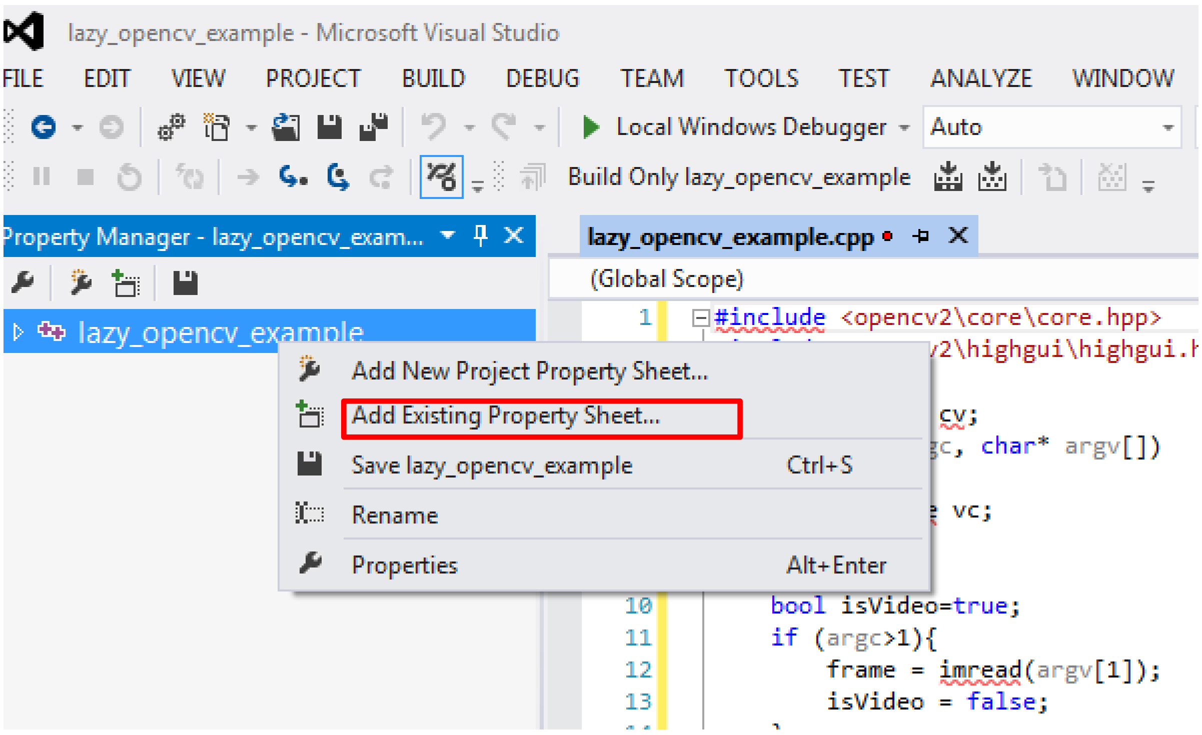
Note the red lines under unknown terms.

3.2.Applying the property sheet

Now click **Property Manager** on the bottom left, or choose **View->Property Manager**.

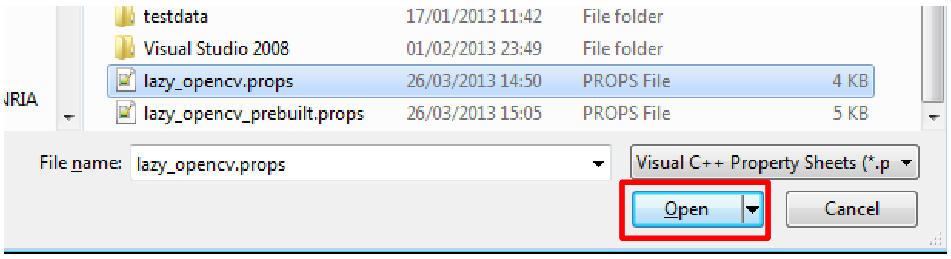

lazy_opencv_example - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS
Solution Explorer Ctrl+Alt+L
Team Explorer Ctrl+\, Ctrl+M
Bookmark Window Ctrl+K, Ctrl+W
Call Hierarchy Ctrl+Alt+K
Class View Ctrl+Shift+C
Code Definition Window Ctrl+\, D
Object Browser Ctrl+Alt+J
Error List Ctrl+\, E
Output Ctrl+Alt+O
Resource View Ctrl+Shift+E
Start Page
Toolbox Ctrl+Alt+X
Find Results
Other Windows
Toolbars
Full Screen Shift+Alt+Enter
All Windows Shift+Alt+M
Navigate Backward Ctrl+-
Navigate Forward Ctrl+Shift+-
Next Task
Previous Task
Property Manager
Property Pages Shift+F4
Solution... Class View Property... Team Ex... Undo Close Output Find
Ready

```
The thread 0x1f668 h
The thread 0x1edec h
The thread 0x1e2f0 h
The thread 0x1f834 h
The thread 0x1fb5c h
The program '[129372
```


lazy_opencv_example - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW
Local Windows Debugger
Auto
Build Only lazy_opencv_example
Property Manager - lazy_opencv_exam...
lazy_opencv_example.cpp
(Global Scope)
lazy_opencv_example
Add New Project Property Sheet...
Add Existing Property Sheet...
Save lazy_opencv_example Ctrl+S
Rename
Properties Alt+Enter

```
#include <opencv2\core\core.hpp>
2\highgui\highgui.h
cv;
c, char* argv[])
vc;
    bool isVideo=true;
    if (argc>1){
        frame = imread(argv[1]);
        isVideo = false;
```

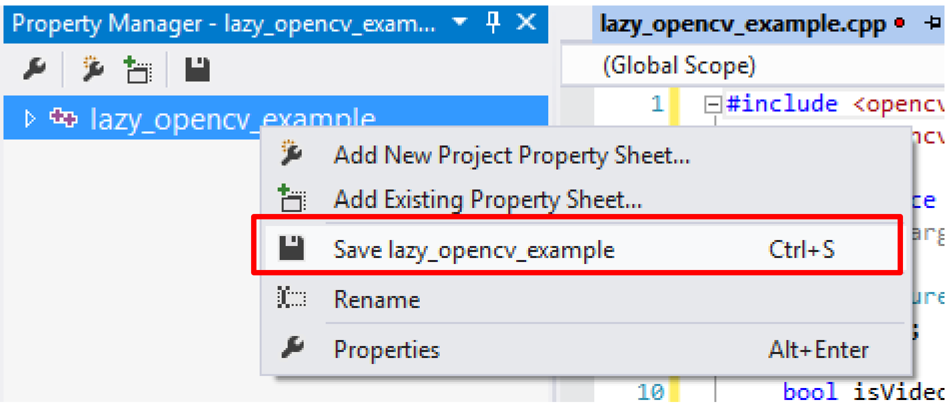
Right-click the project name on the left, choose **Add Existing Property Sheet** and select the (normal or pre-built) appropriate property sheet:



testdata	17/01/2013 11:42	File folder	
Visual Studio 2008	01/02/2013 23:49	File folder	
lazy_opencv.props	26/03/2013 14:50	PROPS File	4 KB
lazy_opencv_prebuilt.props	26/03/2013 15:05	PROPS File	5 KB

Hit Open.

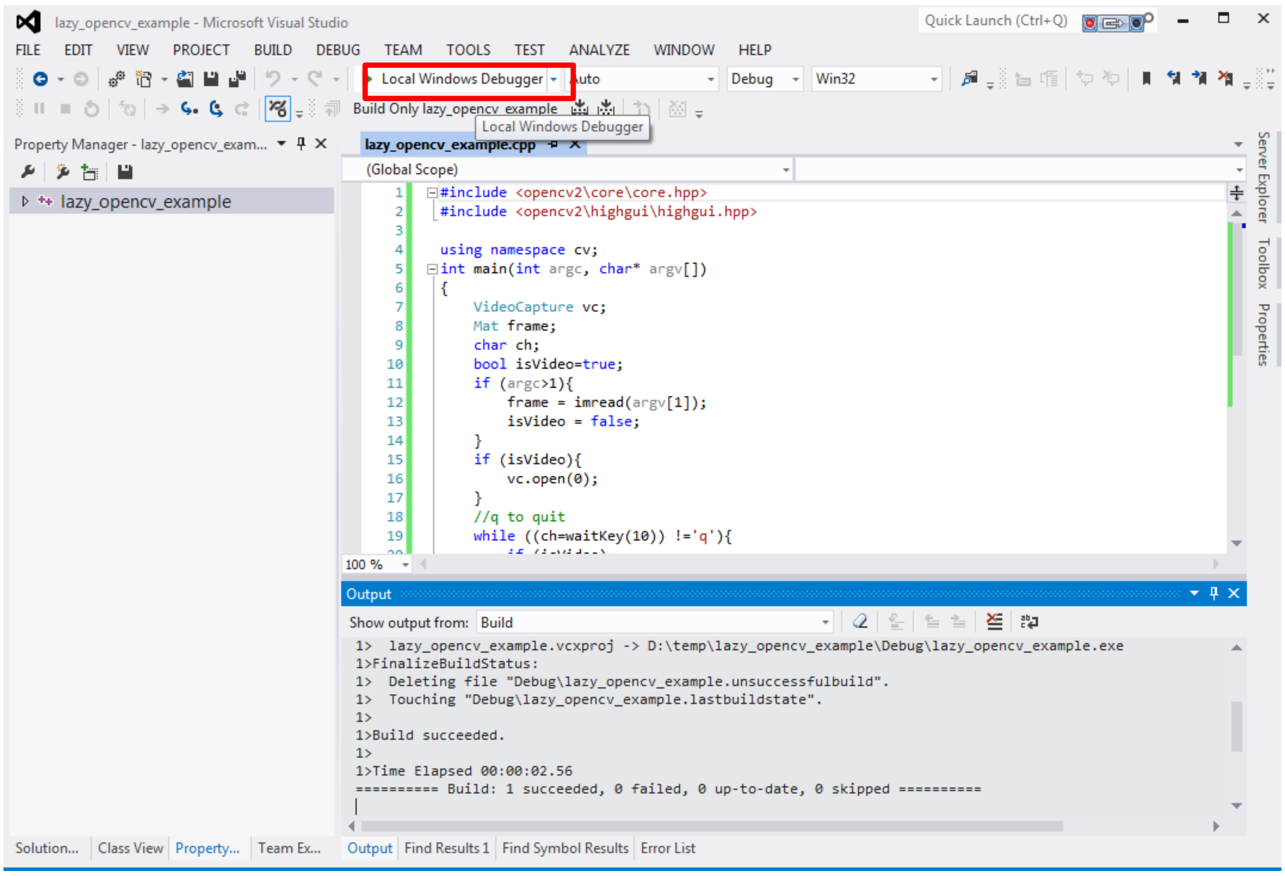
IMPORTANT: Right-click and save the project now, otherwise it will fail to build.


Property Manager - lazy_opencv_exam...
lazy_opencv_example.cpp
(Global Scope)
lazy_opencv_example
Add New Project Property Sheet...
Add Existing Property Sheet...
Save lazy_opencv_example Ctrl+S
Rename
Properties Alt+Enter

The project should now build and debug. To do this for any other project, just repeat section 3.2.

3.3.Testing the example project

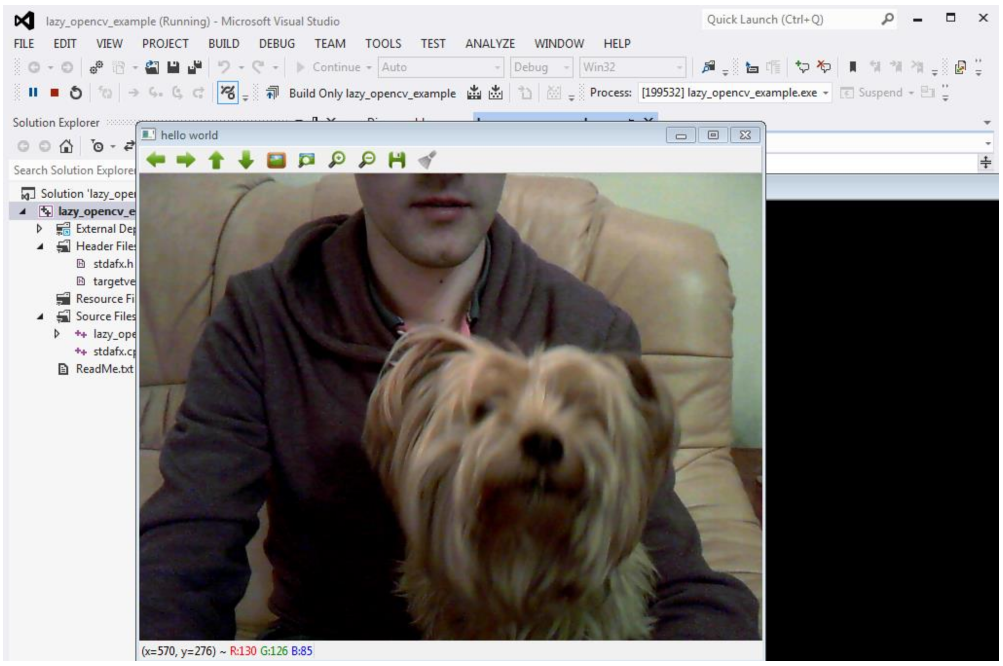
Build the test project. Note the red underlines have disappeared at this point:


lazy_opencv_example - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Local Windows Debugger
Auto
Debug
Win32
Build Only lazy_opencv_example
Local Windows Debugger
Property Manager - lazy_opencv_exam...
lazy_opencv_example
lazy_opencv_example.cpp
(Global Scope)
100 %
Output
Show output from: Build
Solution... Class View Property... Team Ex...
Output Find Results 1 Find Symbol Results Error List
Server Explorer Toolbox Properties

```
#include <opencv2\core\core.hpp>
#include <opencv2\highgui\highgui.hpp>

using namespace cv;
int main(int argc, char* argv[])
{
    VideoCapture vc;
    Mat frame;
    char ch;
    bool isVideo=true;
    if (argc>1){
        frame = imread(argv[1]);
        isVideo = false;
    }
    if (isVideo){
        vc.open(0);
    }
    //q to quit
    while ((ch=waitKey(10)) !='q'){
```

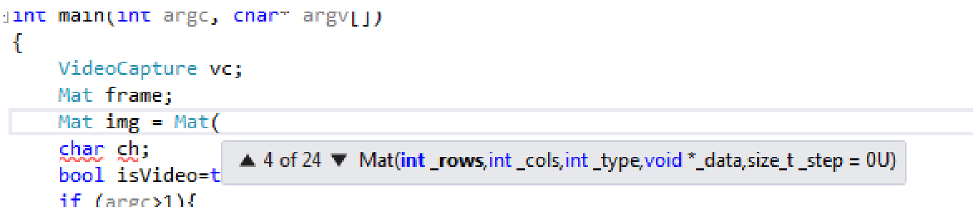
```
1>  lazy_opencv_example.vcxproj -> D:\temp\lazy_opencv_example\Debug\lazy_opencv_example.exe
1>FinalizeBuildStatus:
1>  Deleting file "Debug\lazy_opencv_example.unsuccessfulbuild".
1>  Touching "Debug\lazy_opencv_example.lastbuildstate".
1>
1>Build succeeded.
1>
1>Time Elapsed 00:00:02.56
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

...And run it[1]:


lazy_opencv_example (Running) - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Continue
Auto
Debug
Win32
Build Only lazy_opencv_example
Process: [199532] lazy_opencv_example.exe
Suspend
Solution Explorer
Search Solution Explore
Solution 'lazy_ope
lazy_opencv_e
External Dep
Header File
stdafx.h
targetve
Resource Fi
Source Files
lazy_ope
stdafx.c
ReadMe.txt
hello world
(x=570, y=276) ~ R:130 G:126 B:85

[1] Dog not included.

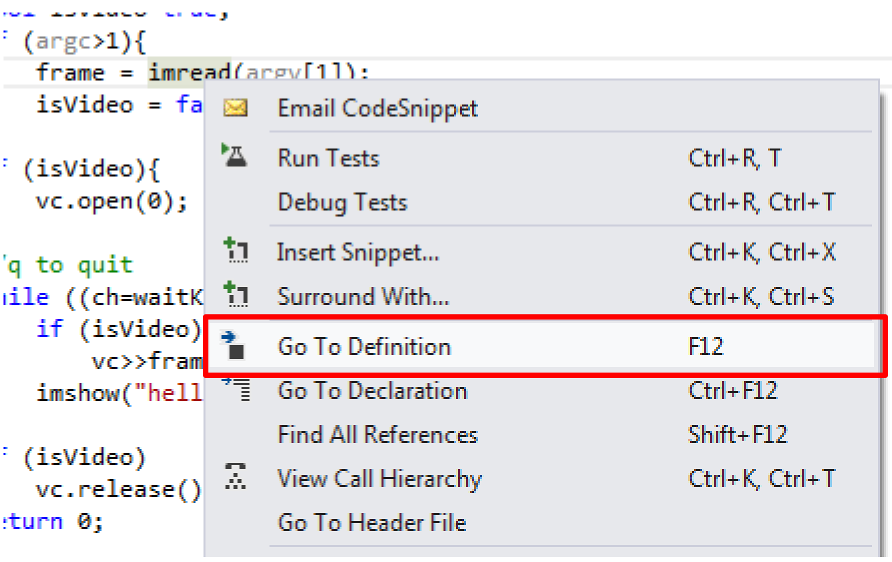
We now also get full Intellisense:



```
int main(int argc, char* argv[])
{
    VideoCapture vc;
    Mat frame;
    Mat img = Mat(
    char ch;
    bool isVideo=t
    if (argc>1){
```

▲ 4 of 24 ▼ Mat(int _rows,int _cols,int _type,void *_data,size_t _step = 0U)

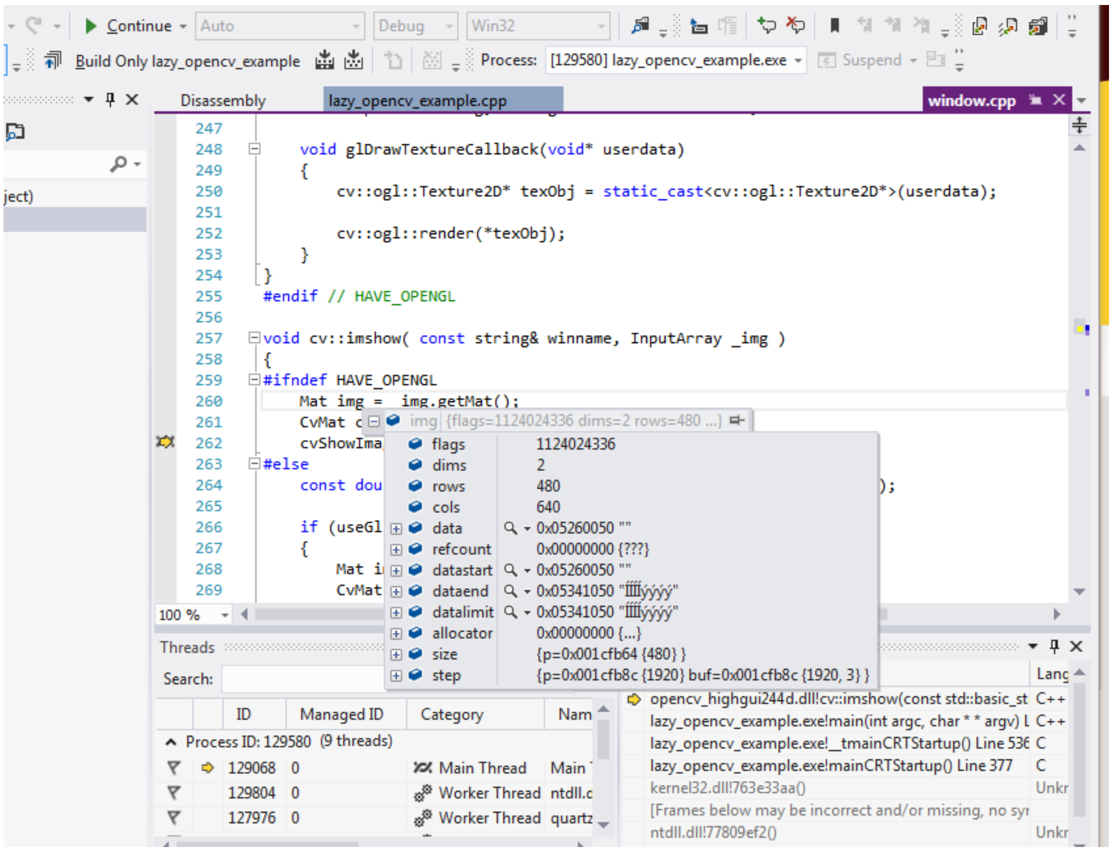
Access to definitions in the headers:



```
 (argc>1){
  frame = imread(argv[1]);
  isVideo = fa

 (isVideo){
  vc.open(0);

'q to quit
ile ((ch=waitK
  if (isVideo)
      vc>>fram
  imshow("hell

 (isVideo)
  vc.release()
turn 0;
```

Email CodeSnippet
Run Tests Ctrl+R, T
Debug Tests Ctrl+R, Ctrl+T
Insert Snippet... Ctrl+K, Ctrl+X
Surround With... Ctrl+K, Ctrl+S
Go To Definition F12
Go To Declaration Ctrl+F12
Find All References Shift+F12
View Call Hierarchy Ctrl+K, Ctrl+T
Go To Header File

And ability to step inside OpenCV Library code while debugging.



```
    void glDrawTextureCallback(void* userdata)
    {
        cv::ogl::Texture2D* texObj = static_cast<cv::ogl::Texture2D*>(userdata);

        cv::ogl::render(*texObj);
    }
}
#endif // HAVE_OPENGL

void cv::imshow( const string& winname, InputArray _img )
{
#ifndef HAVE_OPENGL
    Mat img = _img.getMat();
    CvMat c
    cvShowIma
#else
    const dou                                                         );

    if (useGl
    {
        Mat i
        CvMat
```

```
img {flags=1124024336 dims=2 rows=480 ...}
  flags       1124024336
  dims        2
  rows        480
  cols        640
  data        0x05260050 ""
  refcount    0x00000000 {???}
  datastart   0x05260050 ""
  dataend     0x05341050 "ÍÍÍÍýýýý"
  datalimit   0x05341050 "ÍÍÍÍýýýý"
  allocator   0x00000000 {...}
  size        {p=0x001cfb64 {480} }
  step        {p=0x001cfb8c {1920} buf=0x001cfb8c {1920, 3} }
```

		ID	Managed ID	Category	Nam
Process ID: 129580 (9 threads)					
		129068	0	Main Thread	Main
		129804	0	Worker Thread	ntdll.c
		127976	0	Worker Thread	quartz

	Lang
opencv_highgui244d.dll!cv::imshow(const std::basic_st	C++
lazy_opencv_example.exe!main(int argc, char * * argv) L	C++
lazy_opencv_example.exe!__tmainCRTStartup() Line 536	C
lazy_opencv_example.exe!mainCRTStartup() Line 377	C
kernel32.dll!763e33aa()	Unkr
[Frames below may be incorrect and/or missing, no syı	
ntdll.dll!77809ef2()	Unkr