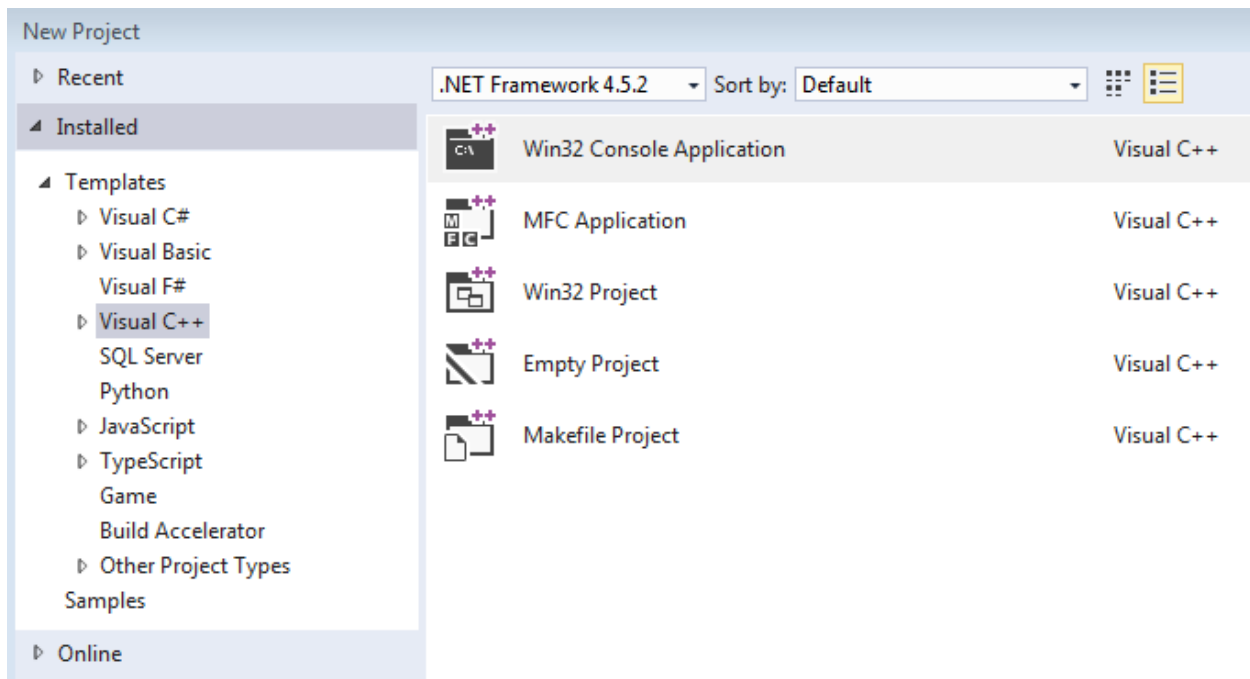


Install Visual Studio 2015 Community (with the C++ compiler in the install options).

- 1) Note that C++ compiler may not be installed with default options. You may be able to select it with the advanced install.
- 2) Start Visual Studio. You can sign in with your Clark account information. It will ask you to fill out some information – you can skip the optional Visual Studio Team Service Site.
- 3) The first time you open Visual Studio, you can set the default configuration for General Use.
- 4) Go to File → New Project... and select Visual C++. You should see the options below, starting with “Win32 Console Application”. If instead you see options to install C++ components, use this to install the C++ compiler now. You probably don’t need the “Windows XP support”.



Download and Extract OpenCV 3

Download from <http://www.opencv.org>

Run the self-extracting executable.

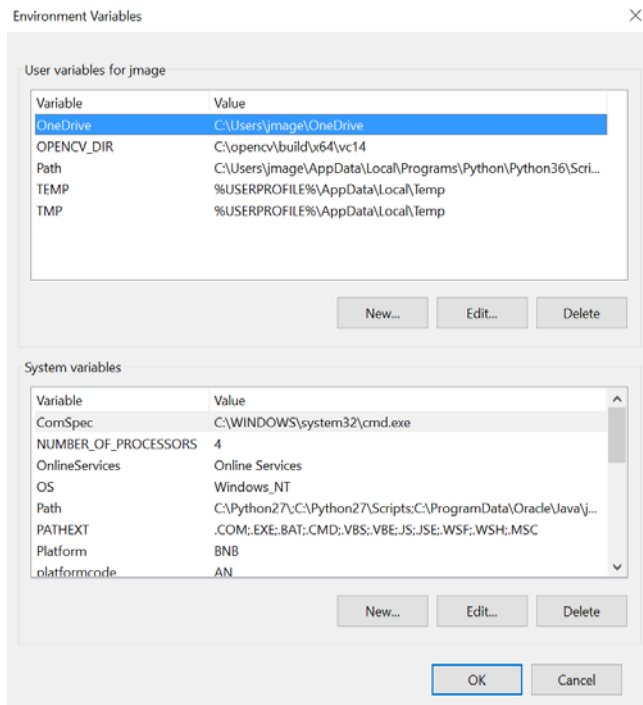
Select where you would like to store the library.

On my computer, I extracted the library to c:\opencv\

Set Environment Variables

You can set environment variables two ways.

One way is the *System* control panel. Open the *System* control panel, then select *Advanced system settings*. Then press the button for *Environment Variables...*



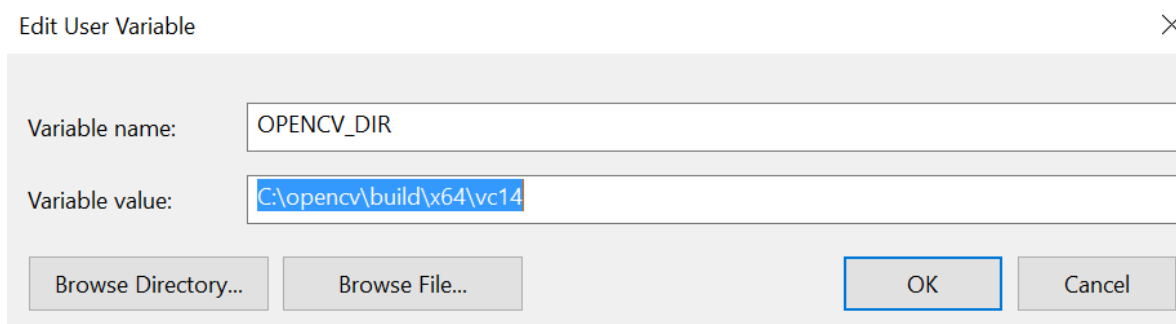
The top “User Variables for...” will set for only the current user of the computer. If you are the only user of your computer, you can use this. If you want to set it for all users, use the bottom System Variables.

Alternative to the System control panel, you can use `setx` command from the command prompt.

1) Add `OPENCV_DIR` variable

Set a new environment variable named `OPENCV_DIR` with the value being the complete path to OpenCV’s `build\x64\vc14` directory.

For example:



2) Edit the `PATH` environment variable. Append the path to the *bin* directory by adding a semi-colon and the path to where it is stored:

- You may be able to use the environment variable above: `%OPENCV_DIR%\bin`
- Or just enter a full path: `C:\opencv\build\x64\vc14\bin`
- Example: `C:\some\existing\path;C:\opencv\build\x64\vc14\bin`

Press `OK` and to exit out of the Environment Variables window.

Test the Path and Environment Variables

- 1) Open a command prompt. (Press the Windows Key and then type *cmd*). If a command prompt is already open, you may need to exit and open a new one to refresh the environment variables.
- 2) Type *opencv_version*
 - a. If this outputs 3.2.0, the path is set properly.
 - b. If this outputs an error message, your path variable is not properly set.
- 3) Type *cd %OPENCV_DIR%*
 - a. Your current working directory should now be the location of the OpenCV's build\x64\vc14 directory.
 - b. If you type *dir*, the directory listing should have a bin and lib subdirectories.

Configure Visual Studio

Start Visual Studio. If it was already open, you may need to exit and restart for the environment variables to take effect.

Creating a New Project:

- Create a new project (File → New → Project...)
- Select Visual C++ → Win32 Console Application
- Type a name for the project (Default ConsoleApplication1 is ok for testing)
- Select a location – this is where the project will be stored.
 - Be careful on lab computers, the default location is on the C: drive
- Press OK.
- On the next screen *Welcome to the Win32 Application Wizard*, press **Next >**
- On the next screen, *Application Settings*
 - Uncheck “Security Development Lifecycle (SDL) checks”.
 - The other options can stay the same.
 - Press **Finish**.

Configure the Project:

- 1) Select Build → Configuration Manager...
- 2) For each project, Under Platform, select x64
- 3) For Active Solution Platform, select x64

Configuring the Library

There are two ways to configure the library:

- On a per-project basis. The settings would need to be configured for every project. To configure the project, make sure the current project is selected in the Solution Explorer (top-right sub-window) select:

Project → Properties

Alternatively: right-click on the project and select Properties

Make sure Configuration says Active(Debug) and Platform says Active(x64).

- On a per-user basis. The settings should be in effect for your account. To find these settings, select:
 - a) View → Other Windows → Property Manager
 - b) This should open the Property Manger within the top-right sub-window.
 - c) Select Debug | x64
 - d) Select Microsoft.cpp.x64.user
 - e) Right-click on Microsoft.cpp.x64.user and select Properties

There are four settings that have to be made in either case:

1. Tell the compiler where the include (.hpp) files are located
2. Tell the linker where the library (.lib) files are located
3. Tell the linker what library (.lib) files to link
4. Tell the debugger the path to the *bin* folder (to find the .dll files)

1) Select C/C++

- a. Set Additional Include Directories: %OPENCV_DIR%\..\..\include

Configuration: Active(Debug) Platform: Active(x64) Conf

Configuration Properties	Additional Include Directories	%OPENCV_DIR%\..\..\include
General	Additional #using Directories	
Debugging	Debug Information Format	Program Database for Edit And Continue (/ZI)
VC++ Directories	Common Language RunTime Support	
C/C++	Consume Windows Runtime Extension	
General	Suppress Startup Banner	Yes (/nologo)

2) Select Linker

- a. Set Additional Library Directories: %OPENCV_DIR%\lib

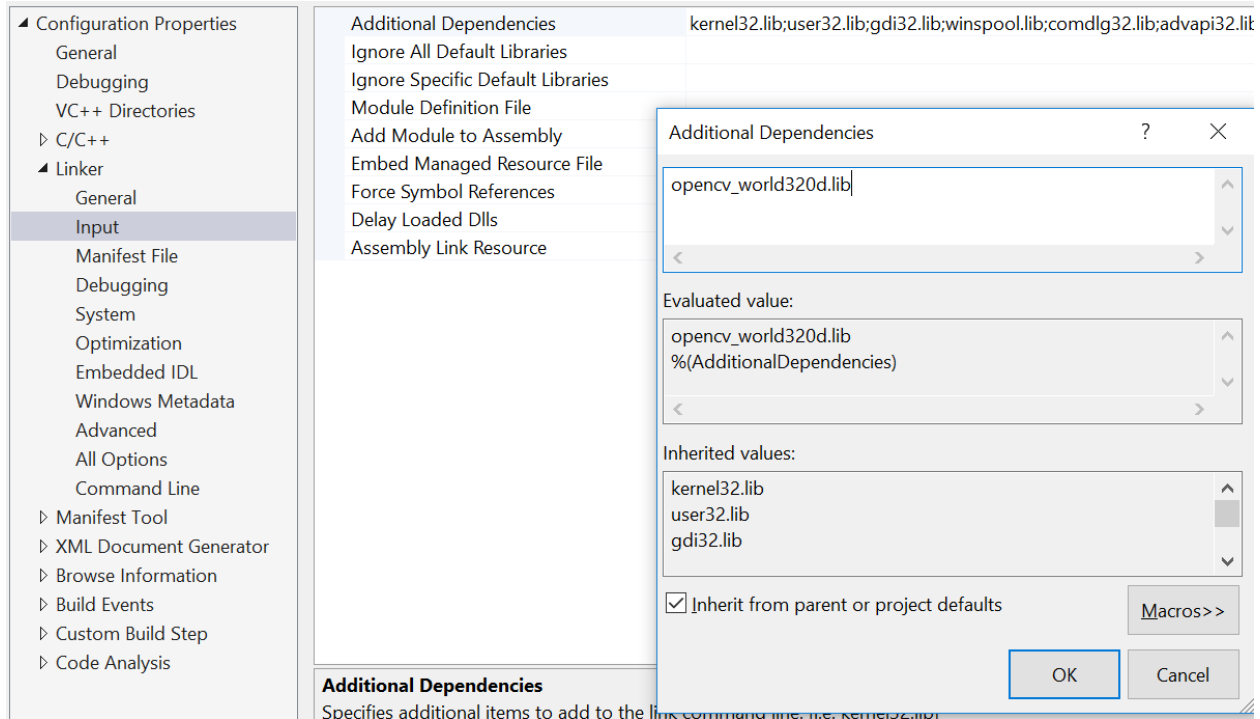
Configuration: Active(Debug) Platform: Active(x64)

Configuration Properties	Output File	\$(OutDir)\$(TargetName)\$(TargetExt)
General	Show Progress	Not Set
Debugging	Version	
VC++ Directories	Enable Incremental Linking	Yes (/INCREMENTAL)
C/C++	Suppress Startup Banner	Yes (/NOLOGO)
Linker	Ignore Import Library	No
General	Register Output	No
Input	Per-user Redirection	No
Manifest File	Additional Library Directories	%OPENCV_DIR%\lib
Debugging	Link Library Dependencies	Yes
System	Use Library Dependency Inputs	No
Optimization	Link Status	
Embedded IDL	Prevent Dll Binding	
	Treat Linker Warning As Error	

3) Select Linker → Input

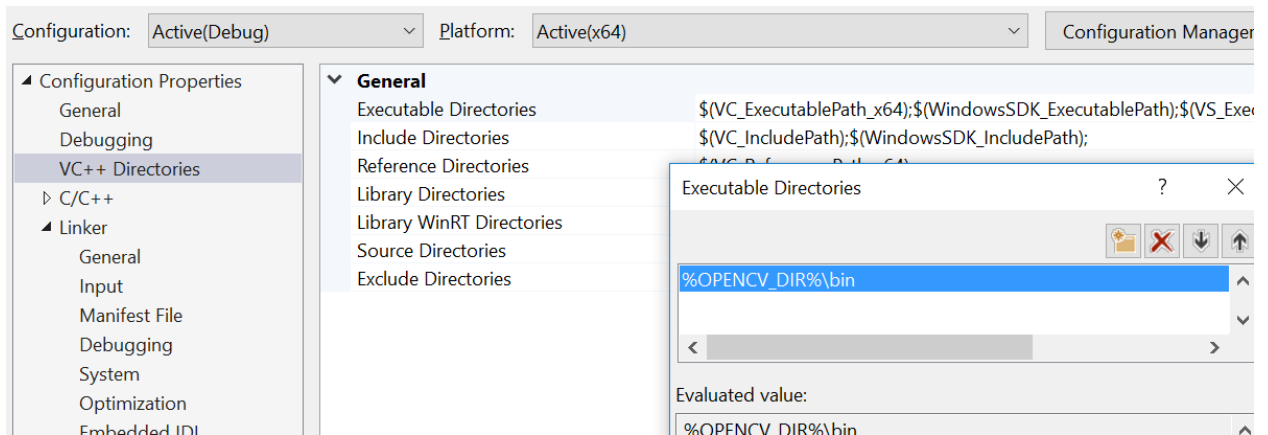
- a. Edit the *Additional Dependencies* to add *opencv_world320d.lib*

Note: If you are using a different version of OpenCV than 3.2.0, check your lib directory for the appropriate name. The “d” at the end means “debug” version.



4) Select VC++ Directories

- a. Edit the *Executable Directories* to add *%OPENCV_DIR%\bin*



Your First OpenCV Program

Edit the .cpp file, replacing the existing starter code with the following:

```
#include "stdafx.h" // remove this if not using precompiled headers
#include <opencv2/opencv.hpp> //Include file for every supported OpenCV function
int main(int argc, char** argv) {
    cv::Mat img = cv::imread(argv[1], -1);
    if (img.empty()) return -1;
    cv::namedWindow("Example1", cv::WINDOW_AUTOSIZE);
    cv::imshow("Example1", img);
    cv::waitKey(0);
    cv::destroyWindow("Example1");
    return 0;
}
```

This code reads an image and displays it. Then exits when a key is pressed.

Find a small image on the internet or your computer. Copy the image into the project's directory. On my computer, this directory was

projects\ConsoleApplication1\ConsoleApplication1\

Note that the first ConsoleApplication1 is the "solution" directory, and the second one is the "project" directory. That will be the "current working directory" when we run our programs.

The main method accepts a command-line argument. To set this argument in the debugger, edit the Project Properties.

- Select *Debugging*
 - Add the file name to the *Command Arguments* option.

ConsoleApplication1 Property Pages

The screenshot shows the Visual Studio Project Properties dialog for ConsoleApplication1. The Configuration is set to Active(Debug) and the Platform is Active(x64). The left sidebar shows the Configuration Properties tree with Debugging selected. The right pane shows the Debugger to launch: Local Windows Debugger. Below that is a table of properties:

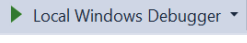
Command	\$(TargetPath)
Command Arguments	cody.jpg
Working Directory	\$(ProjectDir)
Attach	No
Debugger Type	Auto

Build and Test

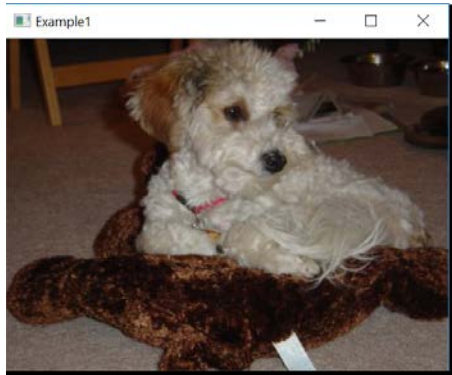
Select Build → Build Solution

If everything is configured correctly, it should compile and tell you that:

```
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
```

To run: Debug → Start Debugging (or )

You should see a window pop-up with your image!



Congratulations! You're ready to explore OpenCV!