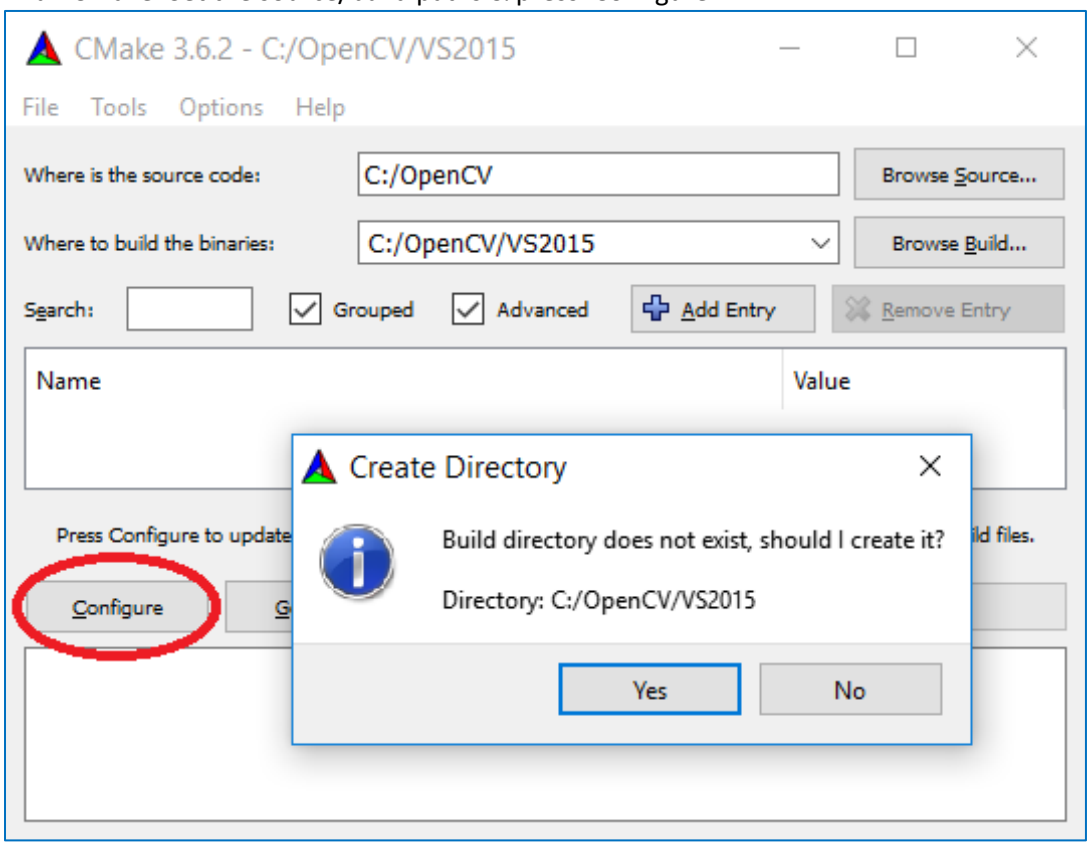
ECE 510 EmbV I
Fall 2016
Lab01 notes

Basic OpenCV operations

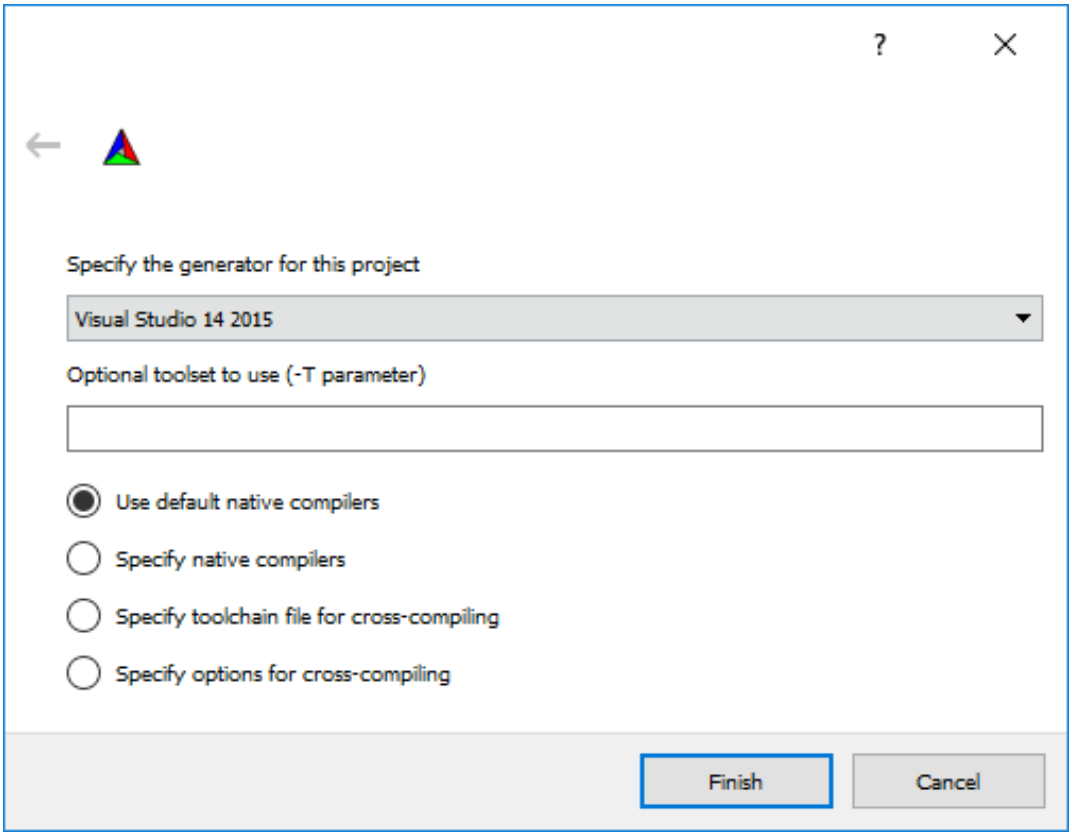
Task 1. Get OpenCV installed and working on your preferred platform.

I installed OpenCV both from binaries and from source. Below you can find my notes on how to install from source. One takeaway: the latest IPP revision (9.2) is not supported by OpenCV 2.4.13.

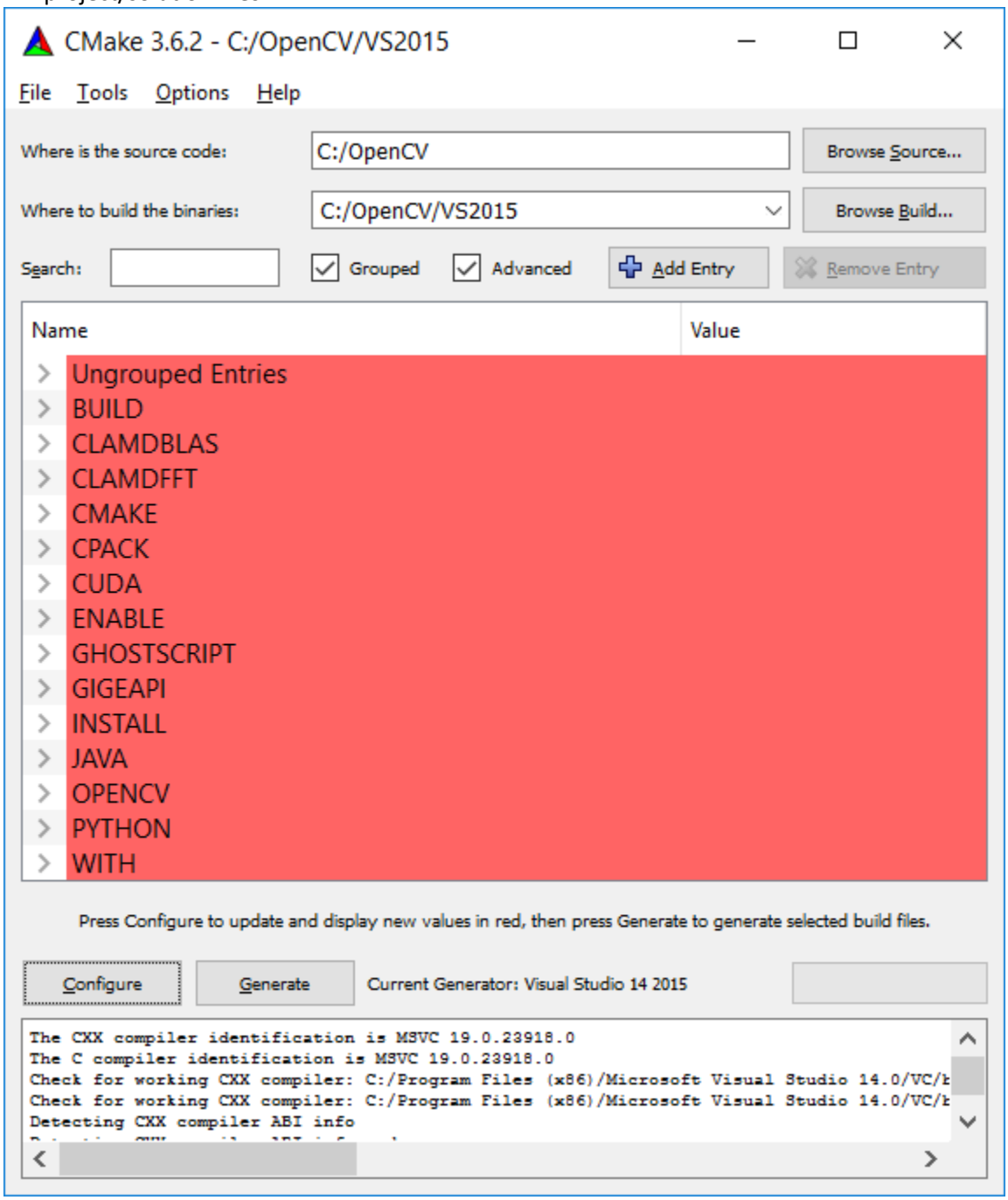
1. Download and install CMake
   URL: https://cmake.org/download/.
2. Download and install OpenCV 2.4.13.
   URL: https://github.com/opencv/opencv/releases/tag/2.4.13.
3. Unzip source to C:\OpenCV.
4. Run CMake. Set the source/build paths & press 'Configure'.



5. CMake will find your Visual Studio installation if you have one.
   'Finish'.



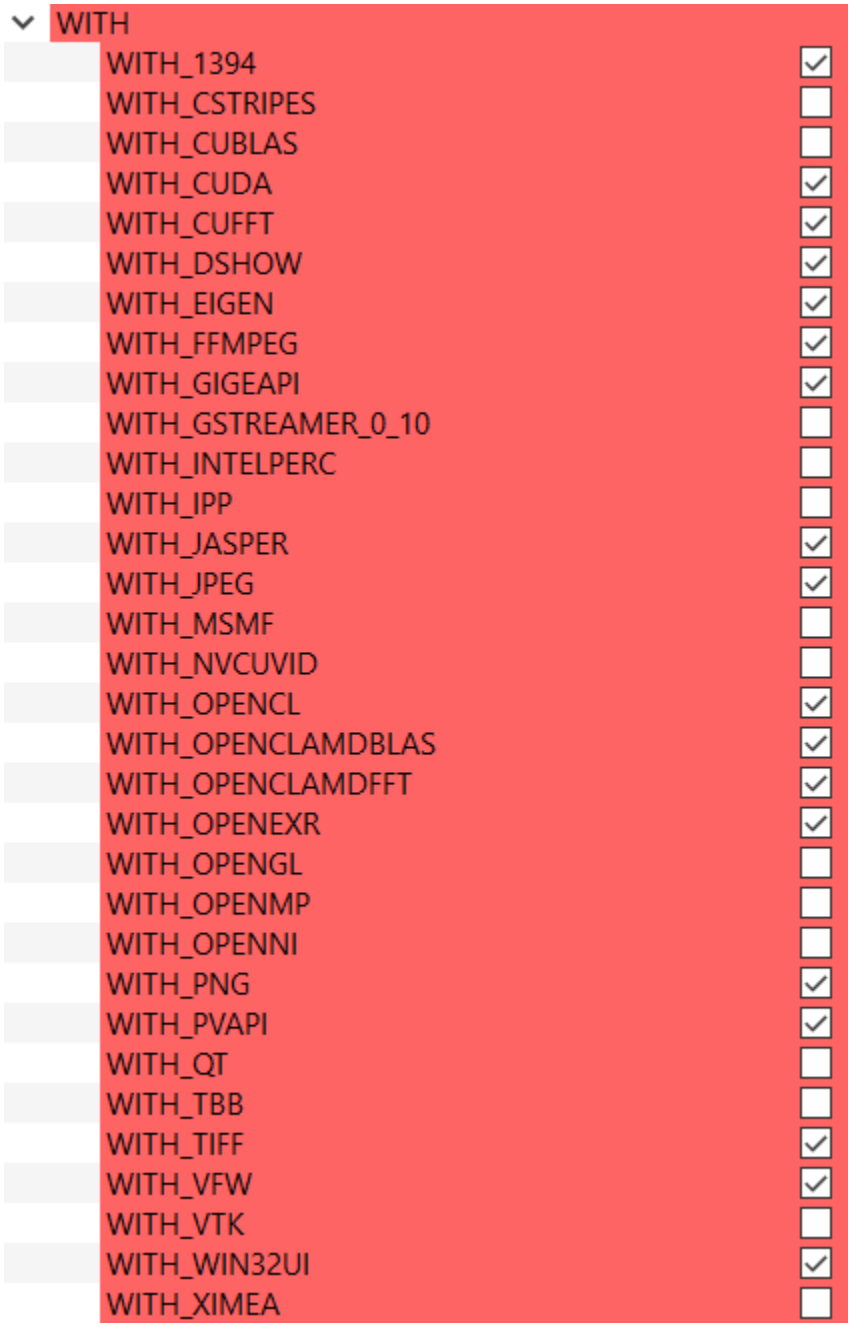
6. CMake will display a list of all the options that can be adjusted before generating the project/solution files.



Ignore CMake CPACK warning at the bottom:

CMake Warning at cmake/OpenCVPackaging.cmake:23 (message):
CPACK_PACKAGE_VERSION does not match version provided by version.hpp header!
Call Stack (most recent call first):
CMakeLists.txt:1105 (include)

7. Expand “WITH”.



8. Deselect:
- WITH_CUDA
- WITH_CUFFT
- WITH_OPENCLAMDBLAS
- WITH_OPENCLAMDFFT

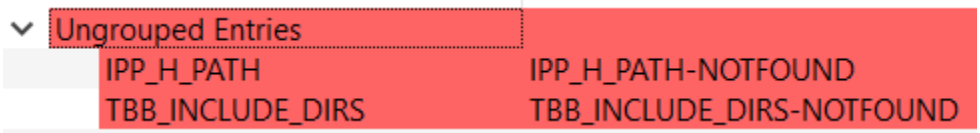
  Select BUILD -> BUILD_EXAMPLES.
9. ‘Configure’. ‘Generate’. No errors. The Visual Studio solutions was generated in ‘C:\OpenCV\VS2015’. Open & build – Done.

With IPP & TBB

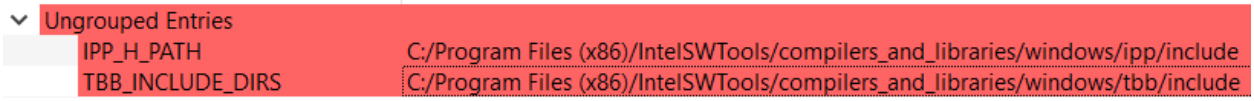
OpenCV 2.4.13 doesn't work with the free version of IPP (9.2). Below you can see the steps you can take to try it out for yourself.

Continue after step 8 above.

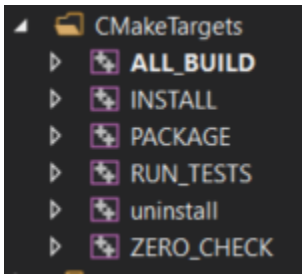
9. Download and install Intel IPP & TBB. IPP & TBB are available for free.
   URL: https://registrationcenter.intel.com/en/forms/?productid=2558&licensetype=2.
10. Select:
- WITH_IPP
- WITH_TBB
11. 'Configure'.



12. Set the IPP & TBB include paths.



13. 'Configure'/'Generate'.
14. Open the OpenCV solution: C:\OpenCV\VS2015\OpenCV.sln.
    Build the "ALL_BUILD" project.



    Lots of errors due to various reasons.
15. Build 'modules/opencv_core' (just one project in order to evaluate and start fixing errors). Most of the errors are due to deprecated functions in IPP 8.2+.
    URL: https://software.intel.com/en-us/articles/intel-ipp-82-deprecated-function-change.

```
1634            ippiMaskNormFuncC1 ippFuncC1 =
1635                normType == NORM_INF ?
1636                (type == CV_8UC1 ? (ippiMaskNormFuncC1)ippiNorm_Inf_8u_C1MR :
1637                type == CV_8SC1 ? (ippiMaskNormFuncC1)ippiNorm_Inf_8s_C1MR :
1638                type == CV_16UC1 ? (ippiMaskNormFuncC1)ippiNorm_Inf_16u_C1MR :
1639                type == CV_32FC1 ? (ippiMaskNormFuncC1)ippiNorm_Inf_32f_C1MR :
1640                0) :
1641            normType == NORM_L1 ?
1642                (type == CV_8UC1 ? (ippiMaskNormFuncC1)ippiNorm_L1_8u_C1MR :
1643                type == CV_8SC1 ? (ippiMaskNormFuncC1)ippiNorm_L1_8s_C1MR :
1644                type == CV_16UC1 ? (ippiMaskNormFuncC1)ippiNorm_L1_16u_C1MR :
1645                type == CV_32FC1 ? (ippiMaskNormFuncC1)ippiNorm_L1_32f_C1MR :
1646                0) :
1647            normType == NORM_L2 || normType == NORM_L2SQR ?
1648                (type == CV_8UC1 ? (ippiMaskNormFuncC1)ippiNorm_L2_8u_C1MR :
1649                type == CV_8SC1 ? (ippiMaskNormFuncC1)ippiNorm_L2_8s_C1MR :
1650                type == CV_16UC1 ? (ippiMaskNormFuncC1)ippiNorm_L2_16u_C1MR :
1651                type == CV_32FC1 ? (ippiMaskNormFuncC1)ippiNorm_L2_32f_C1MR :
1652                0) : 0;
```

16. Comment lines with deprecated functions:

```
1634            ippiMaskNormFuncC1 ippFuncC1 =
1635                normType == NORM_INF ?
1636                (type == CV_8UC1 ? (ippiMaskNormFuncC1)ippiNorm_Inf_8u_C1MR :
1637                //type == CV_8SC1 ? (ippiMaskNormFuncC1)ippiNorm_Inf_8s_C1MR :
1638                type == CV_16UC1 ? (ippiMaskNormFuncC1)ippiNorm_Inf_16u_C1MR :
1639                type == CV_32FC1 ? (ippiMaskNormFuncC1)ippiNorm_Inf_32f_C1MR :
1640                0) :
1641            normType == NORM_L1 ?
1642                (type == CV_8UC1 ? (ippiMaskNormFuncC1)ippiNorm_L1_8u_C1MR :
1643                //type == CV_8SC1 ? (ippiMaskNormFuncC1)ippiNorm_L1_8s_C1MR :
1644                type == CV_16UC1 ? (ippiMaskNormFuncC1)ippiNorm_L1_16u_C1MR :
1645                type == CV_32FC1 ? (ippiMaskNormFuncC1)ippiNorm_L1_32f_C1MR :
1646                0) :
1647            normType == NORM_L2 || normType == NORM_L2SQR ?
1648                (type == CV_8UC1 ? (ippiMaskNormFuncC1)ippiNorm_L2_8u_C1MR :
1649                //type == CV_8SC1 ? (ippiMaskNormFuncC1)ippiNorm_L2_8s_C1MR :
1650                type == CV_16UC1 ? (ippiMaskNormFuncC1)ippiNorm_L2_16u_C1MR :
1651                type == CV_32FC1 ? (ippiMaskNormFuncC1)ippiNorm_L2_32f_C1MR :
1652                0) : 0;
```

17. Still getting errors for ‘ippStaticInit’ not found. Replace with ‘ippInit’.
    This fixes ‘modules/opencv_core’ but….
18. Build ‘ALL_BUILD’ again.
    Too many errors, the 2.4.13 branch needs major rework in order to accommodate IPP 9.2.

Visual Studio solution setup

For this class I created a Visual Studio 2015 solution called *Solution* in 'C:\OpenCV\Solution'. I am using the VS2015 Git extensions to keep track of my progress.

The solution holds three projects currently:

- common
  This is a static library project that holds common tools used throughout the labs.
- ex01
  This is the first OpenCV example I wrote based on the example provided in the first week.
- lab01

Various DLLs are required to debug/execute the examples/labs. I placed the missing VS2013 DLLs 'MSVCP120D.dll' and 'MSVCR120D.dll' in 'C:\OpenCV\Solution\vsredist'. I added the paths to the OpenCV DLLs and the VS2013 DLLs to the PATH environment variable. This removes the need to copy the DLLs to the path of the executable.

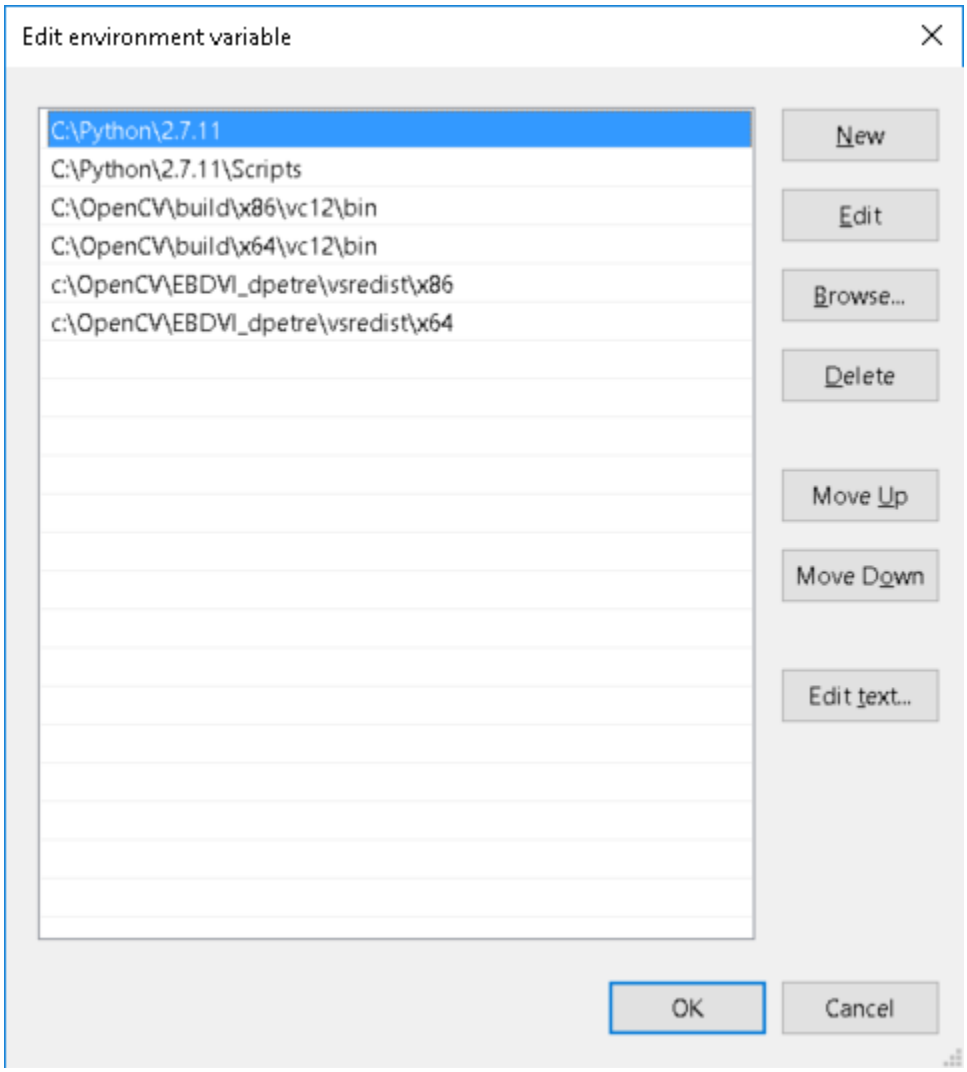


Figure 1. Add OpenCV and VS2013 DLLs locations to PATH.

Task 2. Image pyramid

The code for this task can be found in function lab01::Task02(…).

I created a new function called ImagePyramid that takes as arguments:

- A pointer to a function
  pyrDown and pyrUp are passed as arguments to ImagePyramid in order to perform either scale up or down.
- Input image
- Number of scaling steps
- Scale factor
- Display window name, width and height

The application reads the input image and displays it. Press any key to scale down three times followed by scale up three times. The code that does this:

```
dstImage = lab01::ImagePyramid(&pyrDown, srcImage, 3, 0.5f, winName, srcImage.cols, srcImage.rows);
dstImage = lab01::ImagePyramid(&pyrUp  , dstImage, 3, 2.0f, winName, srcImage.cols, srcImage.rows);
```

After scaling down the image three times by a factor of two it looks very pixelated but we can still make out a few of the details. I am using the output of the scale down stage as input to the scale up stage. After scaling up the image again three times by a factor of two the image looks a lot smoother and even the level of detail seems much better than the scaled down version. This looks like evidence that the pyrUp method is performing some kind of interpolation to fill in the missing pieces. Overall scaling down three times followed by scaling up three times results in a blurring effect similar to the one demonstrated in ex01 using GaussianBlur but more pronounced.

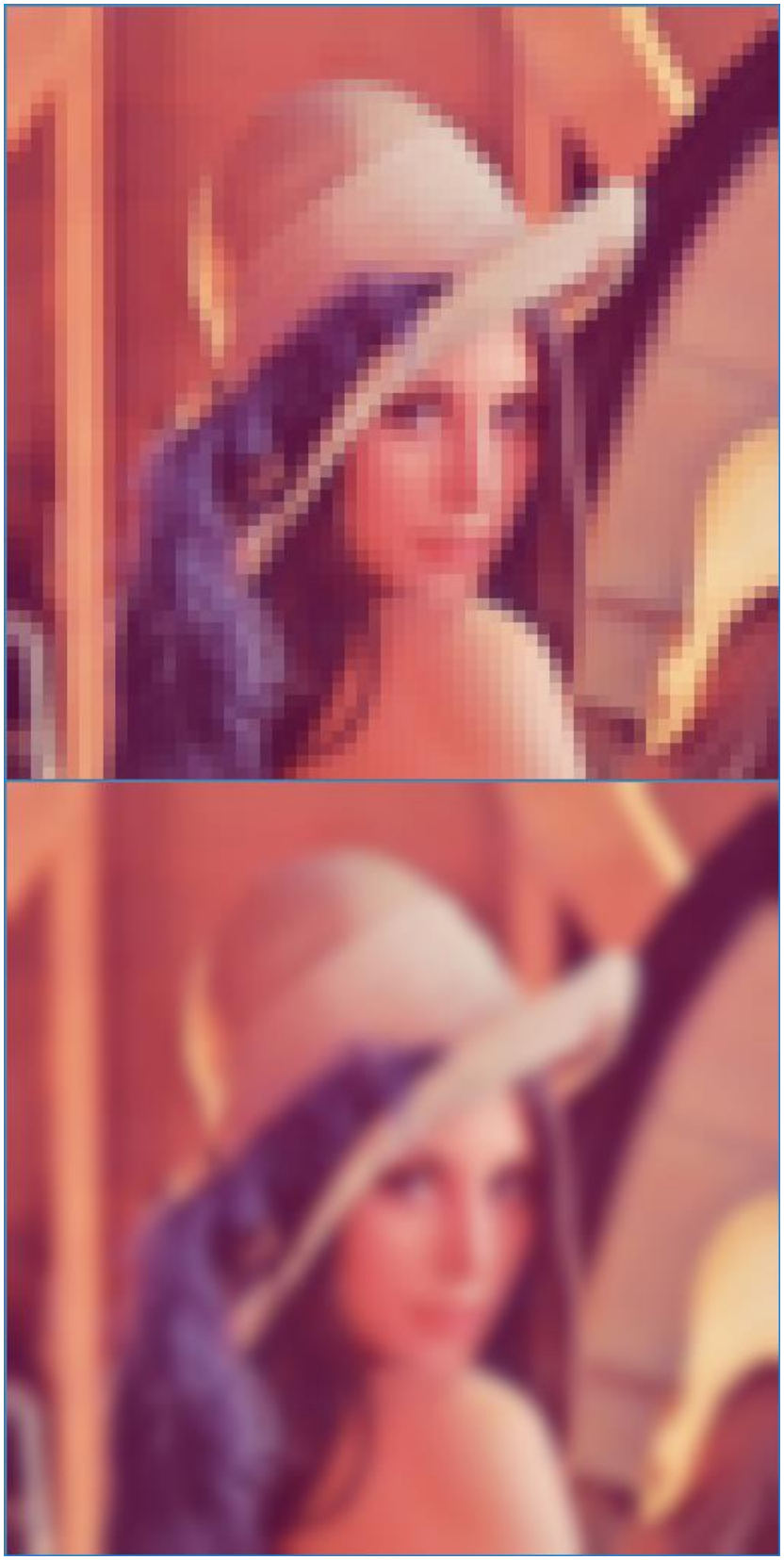


Figure 2. Image scaling DOWN/UP three times by a factor of two.
Top scaled down. Bottom scaled up.

Task 3. Draw an image.

The code for this task can be found in function lab01::Task03().

I created a blue circle at (100, 100) with a radius of 20 using the OpenCV method circle. In order to fill the circle the thickness argument must take the value -1.

In order to create the rectangle I created two loops over the rows between 60 and 100 and over the columns between 30 and 100.

Press a key to draw the circle. Press a key again to draw the rectangle.

We can notice how the blue and green are blended in the second quadrant of the circle (in trigonometric notation). The reason for the blending is because we're only setting the green channel leaving the blue channel unchanged.

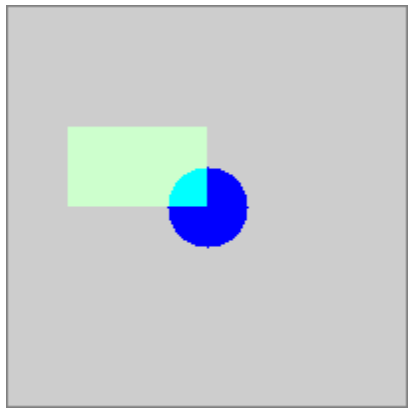


Figure 3.The output after following the lab01 requirement.

If instead we set both the blue and the red channels to zero the green rectangle will be opaque and will overlap the blue circle.ss

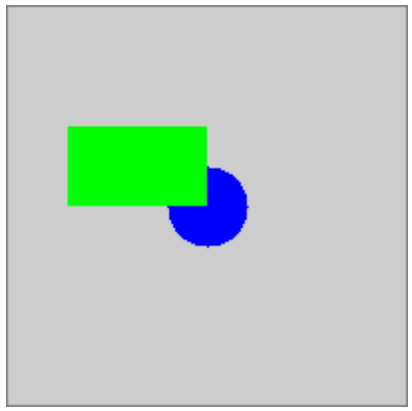


Figure 4.The output after setting the red and blue channels to zero.

End notes

The output images have been saved to '\Solution\images'.

The zip archive contains the full VS2015 solution used to implement the requirements of this lab.

Please note that for convenience I've included the VS2013 DLLs in '\Solution\vsredist'.