

Pro Processing for Images and Computer Vision with OpenCV

Solutions for Media Artists and
Creative Coders

—
Teaching your computer to see

—
Bryan WC Chung

Apress®

Pro Processing for Images and Computer Vision with OpenCV

Solutions for Media Artists and
Creative Coders



Bryan WC Chung

Apress®

Pro Processing for Images and Computer Vision with OpenCV

Bryan WC Chung
Academy of Visual Arts, Kowloon Tong, Hong Kong

ISBN-13 (pbk): 978-1-4842-2774-9
DOI 10.1007/978-1-4842-2775-6

ISBN-13 (electronic): 978-1-4842-2775-6

Library of Congress Control Number: 2017951872

Copyright © 2017 by Bryan WC Chung

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Cover image by Freepik (www.freepik.com).

Managing Director: Welmoed Spahr
Editorial Director: Todd Green
Acquisitions Editor: Jonathan Gennick
Development Editor: Laura Berendson
Technical Reviewer: Kathleen Sullivan
Coordinating Editor: Jill Balzano
Copy Editor: Kim Wimpsett

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484227749. For more detailed information, please visit www.apress.com/source-code.

Printed on acid-free paper

Contents at a Glance

About the Author	xi
About the Technical Reviewer	xiii
Acknowledgments	xv
■ Chapter 1: Getting Started with Processing and OpenCV	1
■ Chapter 2: Image Sources and Representations	39
■ Chapter 3: Pixel-Based Manipulations	71
■ Chapter 4: Geometry and Transformation.....	101
■ Chapter 5: Identification of Structure.....	133
■ Chapter 6: Understanding Motion.....	177
■ Chapter 7: Feature Detection and Matching	219
■ Chapter 8: Application Deployment and Conclusion.....	263
Index.....	289

Contents

About the Author	xi
About the Technical Reviewer	xiii
Acknowledgments	xv
■ Chapter 1: Getting Started with Processing and OpenCV	1
Processing.....	1
OpenCV.....	2
Processing Installation	3
Install Processing	3
Run Processing.....	5
OpenCV Installation	7
macOS	9
Windows.....	17
Linux.....	28
Test Run.....	34
Hello World	34
Matrix Example.....	36
Conclusion.....	37
■ Chapter 2: Image Sources and Representations	39
Digital Image Fundamentals.....	39
Images in Processing	40
Import an External Image	40
Create an Image in Processing.....	43

Graphics and Images	44
BufferedImage in Processing	47
Moving Images in Processing	48
Digital Movies	49
Live Video Captures	51
Matrices and Images in OpenCV	53
Image Conversion Between Processing and OpenCV.....	58
From Processing to OpenCV	60
From OpenCV to Processing	62
Conclusion.....	70
■ Chapter 3: Pixel-Based Manipulations	71
Visual Properties	71
Position.....	72
Size.....	72
Shape.....	73
Orientation	73
Color	73
Value.....	74
Pixel Color Manipulation.....	74
Color Change with Pixel Position.....	75
Color Change with Pixel Distance.....	79
Color Change with Trigonometric Functions.....	82
Randomness.....	86
Drawing with Existing Images.....	90
Blending Multiple Images.....	97
Conclusion.....	100
■ Chapter 4: Geometry and Transformation.....	101
Image Transformation	101
Image Orientation.....	104

Image Resizing.....	108
Affine Transform.....	110
Perspective Transform.....	115
Linear vs. Polar Coordinates.....	118
Three-Dimensional Space.....	120
General Pixel Mapping.....	130
Conclusion.....	132
Chapter 5: Identification of Structure.....	133
Image Preparation.....	133
Conversion to Grayscale.....	133
Conversion to a Black-and-White Image.....	135
Morphological Operations.....	137
Blur Operations.....	140
Edge Detection.....	144
Line Detection.....	146
Circle Detection.....	152
Contours Processing.....	155
Finding the Contours.....	156
Bounding Box.....	161
Minimum Area Rectangle.....	162
Convex Hull.....	164
Polygon Approximation.....	165
Testing a Point in Contour.....	167
Checking Intersection.....	169
Shape Detection.....	172
Conclusion.....	175

■ Chapter 6: Understanding Motion	177
Effects with Moving Images.....	177
Mosaic Effect.....	178
Slit-Scan Effect.....	179
Scrolling Effect.....	180
Visualization in 3D.....	183
Frame Differencing.....	186
Background Removal.....	191
Optical Flow.....	196
Motion History.....	205
Conclusion.....	218
■ Chapter 7: Feature Detection and Matching	219
Corner Detection.....	219
Sparse Optical Flow.....	222
Identify the Feature Points.....	222
Improve the Accuracy.....	224
Calculate the Optical Flow.....	226
Visualize the Flow Information.....	229
Feature Detection.....	235
Feature Matching.....	240
Face Detection.....	255
People Detection.....	260
Conclusion.....	261
■ Chapter 8: Application Deployment and Conclusion	263
Developing Libraries in Processing.....	263
Install the Eclipse Software.....	264
Prepare the OpenCV and Processing Libraries.....	265
Build the CVImage Library.....	267

Exporting Applications from Processing.....	278
Using System Commands in Processing.....	280
Optimizing Tracking with the Kalman Filter.....	284
Other OpenCV Modules.....	287
Conclusion.....	287
Index.....	289

About the Author



Bryan WC Chung is an interactive media artist and design consultant. He was the grand prize winner of the 19th Japan Media Arts Festival, Art Division, 2015. In 2009, his consultation work on the Coca-Cola Happy Whistling Machine won the Media Kam Fan Advertising Award. Bryan's works have been exhibited at the World Wide Video Festival, Multimedia Art Asia Pacific, Stuttgart Film Winter Festival, Microwave International New Media Arts Festival, and China Media Art Festival. In the Shanghai Expo 2010, he provided interactive design consultancy to industry leaders in Hong Kong. Bryan also develops software libraries for the open source programming language Processing. He is the author of the book *Multimedia Programming with Pure Data*. Currently, he is an associate professor in the Academy of Visual Arts at Hong Kong Baptist University, where he teaches classes on interactive art, computer graphics, and multimedia. His personal website can be found at <http://www.magicandlove.com>.

About the Technical Reviewer



Kat Sullivan lives somewhere in the intersection between movement and technology. After double majoring in computer science and dance at Skidmore College, she worked for several years as a software engineer and freelanced as a dancer. Not wanting to compartmentalize her life, she went to the Interactive Telecommunications Program (ITP) and began creating work involving creative coding, live performance, machine learning, and more. Upon completing her master's degree at ITP, she was invited to stay an additional year as a research resident. Her work has been presented at Lincoln Center, National Sawdust, Pioneer Works, Flux Factory, South by Southwest, and the Liberty Science Center. She is currently teaching a motion capture course at NYU.

Acknowledgments

I would like to express my gratitude to my wife, Kimburley, for her support and patience throughout the writing and creative process.

Thanks to my father for introducing me to the world of arts and craft at an early age.

Thanks to my late brother, Percy, for sharing his inspiration in illustration and graphic design.

Thanks to my mother and sister for their continuous support and caring.

Thanks to my boss, Professor John Aiken, for providing me with a beautiful place to work.

CHAPTER 1



Getting Started with Processing and OpenCV

The chapter introduces you to Processing and OpenCV and how to install them. By the end of the chapter, you will have a general understanding about the types of applications you can build by following the examples in this book. You will also be able to write a “Hello World” program to display version information for OpenCV within the Processing programming environment.

Processing

Ben Fry and Casey Reas from the former Aesthetic + Computation Group of the MIT Media Lab initiated the Processing project (<http://processing.org>) in 2001 to create a programming environment for artists and designers to learn the fundamentals of computer programming within the context of electronic arts. Based on the Java programming language, Processing is modeled as an electronic sketchbook for artists and designers to generate their creative ideas. Processing comes with an integrated development environment (IDE). Users can directly code in the environment and execute the code to see the visual results in real time. Processing is equipped with a comprehensive set of frameworks and libraries to provide simple access to functions for creating 2D and 3D graphics and building animation. Java was chosen as the programming language to cater to cross-platform compatibility. At the moment, it supports macOS, Windows, and the major Linux operating systems. Recently, Processing has evolved to include other programming languages such as JavaScript and Python.

Besides the core functions of the Processing language and the vast number of native Java libraries, Processing supports user-contributed libraries (<https://processing.org/reference/libraries/>) from the community. A lot of the libraries were built to hide the technical details of implementing complex software such as physics engines and machine-learning algorithms or to support additional hardware devices such as the Kinect camera. For example, I have developed a wrapper library called `Kinect4WinSDK` to support the Kinect version 1 camera with the official Kinect for Windows software development kit (SDK).

In the area of computer graphics, Processing is capable of producing both vector graphics and raster graphics. In creative applications, algorithmic art and generative art (Figure 1-1) will often make use of vector graphics. In this book, the focus is on image processing and computer vision. In this case, raster graphics will be the primary approach to generate images.

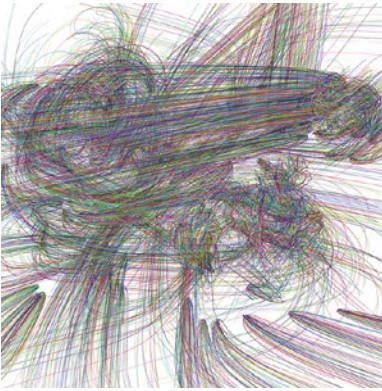


Figure 1-1. Algorithmic art example

OpenCV

Open Source Computer Vision Library (OpenCV, <http://opencv.org/>) started as an Intel research initiative around 1999. Now, it is the most popular open source software library for computer vision and machine learning. In the beginning, it was a set of C library functions for image processing and computer vision. Now, it has C++, Python, Java, and MATLAB bindings and works on macOS, Windows, Linux, Android, and iOS, with acceleration support from CUDA and OpenCL. The OpenCV library comes with a collection of modules. Each of the modules handles a specific group of applications under the umbrella of image processing, computer vision, and machine learning. The following are the common modules:

- `core`: Core OpenCV data structures and functionalities
- `imgproc`: Image processing
- `imgcodecs`: Image file reading and writing
- `videoio`: Media input/output routines
- `highgui`: High-level graphical user interface
- `video`: Video analysis
- `calib3d`: Camera calibration and 3D reconstruction
- `features2d`: Working with 2D features description and matching
- `objdetect`: Object detection such as faces
- `ml`: Machine learning
- `flann`: Clustering and searching in higher-dimensional spaces
- `photo`: Computational photography
- `stitching`: Stitching images together
- `shape`: Shape matching
- `superres`: Super-resolution enhancement
- `videostab`: Video stabilization
- `viz`: 3D visualization

OpenCV includes several extra modules that provide additional functionalities, such as text recognition, surface matching, and 3D depth processing. This book also covers the module `optflow`, which performs optical flow analysis.

Processing Installation

This section explains the procedures to download and install the Processing programming environment. At the time of writing, the latest version of Processing is 3.2.3. It is advised that you use version 3, rather than previous versions, of Processing for compatibility reasons. Each distribution of Processing also includes with the Java runtime code. The installation processes for the three platforms are straightforward and similar.

Install Processing

Download the Processing code from <https://processing.org/download/>. In this book, I will use the 64-bit versions. If you want to take a look at the Processing source code, you can download it from the GitHub distribution (<https://github.com/processing/processing>). The following are the three files for the macOS, Windows, and Linux platforms:

- `processing-3.2.3-macosx.zip`
- `processing-3.2.3-windows64.zip`
- `processing-3.2.3-linux64.tgz`

Processing does not assume any particular location to install the software. For macOS, you can download and expand the file into a macOS program named Processing. Copy the program to the Applications folder similar to other applications you install for macOS. For Windows and Linux, the compressed file will be expanded into a folder named `processing-3.2.3`. You can download and expand the compressed file into any folder you want to maintain the Processing software. In this book, we expand the folder `processing-3.2.3` into the user's Documents folder. Figure 1-2 shows the contents of the folder. To run Processing, simply double-click the Processing icon.

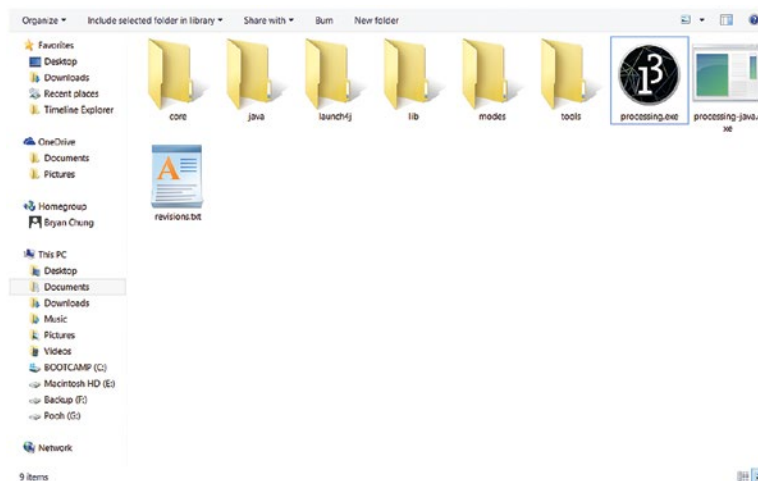


Figure 1-2. Processing folder for Windows

Figure 1-3 shows the default screen layout of the Processing IDE. The code in the window will be the first Processing program you are going to test.

```
void setup() {  
  size(800, 600);  
}  
  
void draw() {  
  background(100, 100, 100);  
}
```

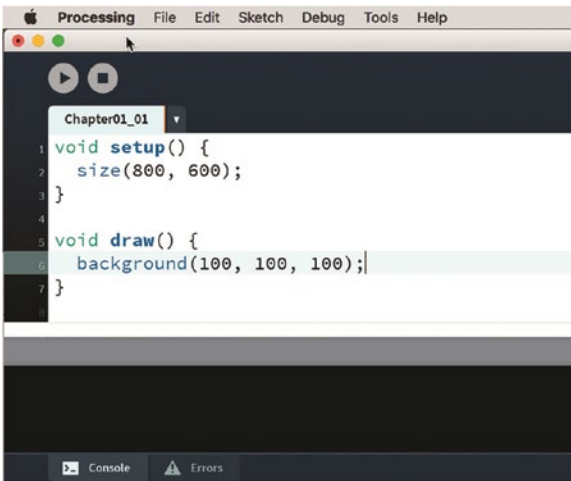


Figure 1-3. Processing IDE screen

After you have started Processing, it will automatically create a folder in your personal Documents folder in which it maintains all the Processing programs. For macOS, its name is `/Users/bryan/Documents/Processing`. In Windows, the folder name is `C:\Users\chung_000\Documents\Processing`. In Linux, it is `/home/bryan/sketchbook`. (In the example, the username is `bryan` or `chung_000`.) Figure 1-4 shows an example view of the folder contents.

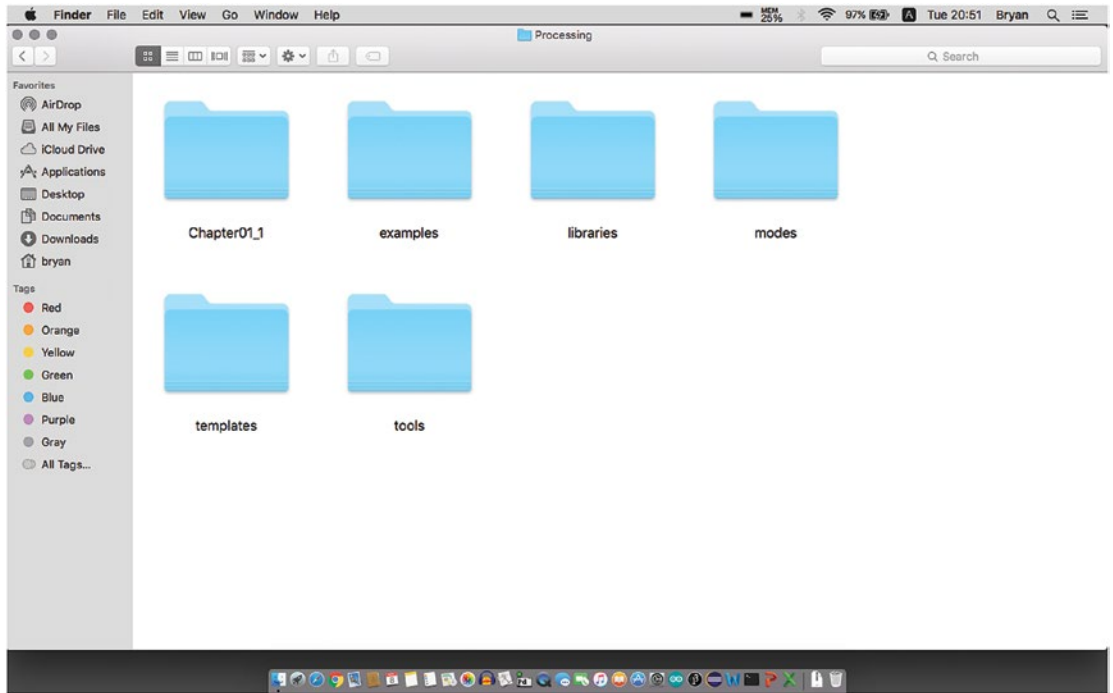


Figure 1-4. Processing sketchbook folder contents

Each Processing program is maintained in its own folder within the Processing folder. In addition to each program, the folder contains other subfolders, such as the `libraries` folder for downloading external libraries from the Processing distribution and the `modes` folder for implementing other languages in Processing such as Python and JavaScript.

Run Processing

In the top-left corner of the Processing IDE, there are two buttons, Play and Stop. Clicking the Play button will start the compilation and execution of the program. Figure 1-5 shows the blank screen created with your first program. Clicking the Stop button at this moment will stop the execution and close the window.

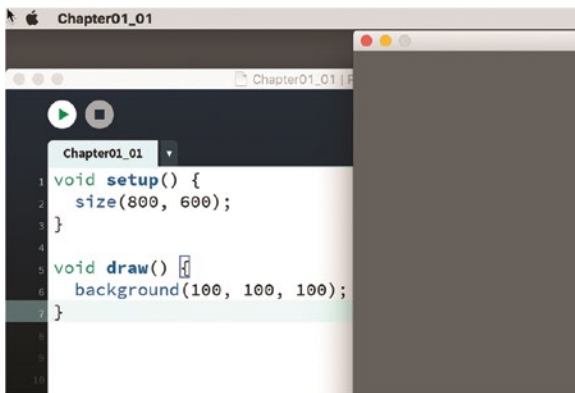


Figure 1-5. First Processing program

You need to install an additional library for the exercises in this book. It is the video library that is built on top of the open source multimedia framework GStreamer (<https://gstreamer.freedesktop.org/>). Processing will use it for playing back digital videos such as MP4 files and capturing live video with a webcam. To install the library (Figure 1-6), choose Sketch > Import Library > Add Library from the main menu. From the Contribution Manager window, choose the Video library and click the Install button. The library will then be downloaded to the libraries subfolder of your Processing folder.

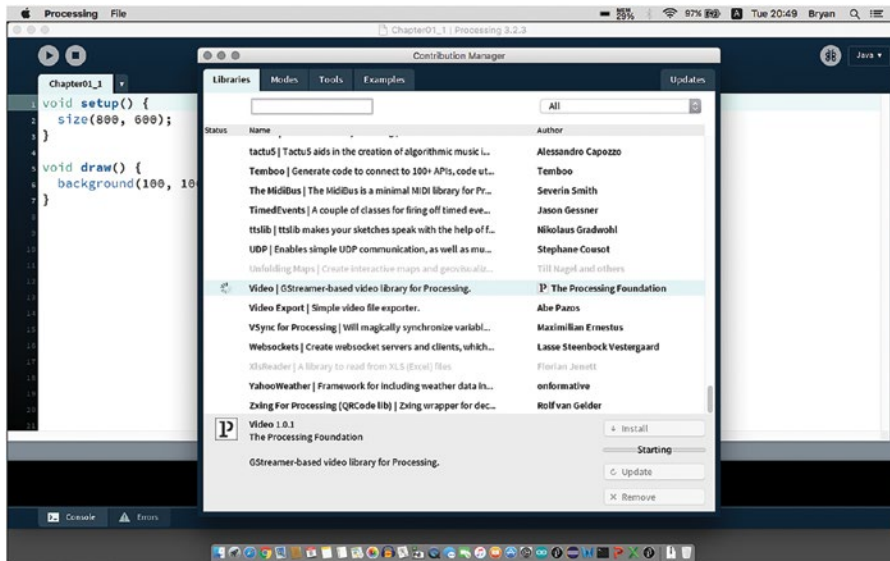


Figure 1-6. Installing the video library

In Chapter 2, you will use this library for loading external digital video and capturing live video streams from webcams. After you have installed the Processing programming environment, you can proceed to install OpenCV on your system.

OpenCV Installation

The installation for OpenCV is a bit complicated because you are going to build the OpenCV library from source. The library you are going to build is different from the existing OpenCV for Processing library written by Greg Borenstein (<https://github.com/atduskgreg/opencv-processing>). It is better for you to remove the existing OpenCV for Processing library before proceeding with this installation process. The OpenCV distribution includes all the core functions. To use other functions for motion analysis, you also need to build the extra modules that are maintained in the contributed libraries. You are going to download both of them from the GitHub repositories. The original OpenCV source is at <https://github.com/opencv/opencv>, and the source for the extra modules is at https://github.com/opencv/opencv_contrib. Note that the master branch in the OpenCV repository contains only version 2.4. To use version 3.1, which you want to do for this book, you need to select the 3.1.0 tag, as shown in Figure 1-7. After choosing the right version tag, you can download the OpenCV source by clicking the “Clone or download” button and then the Download ZIP button, as shown in Figure 1-8.

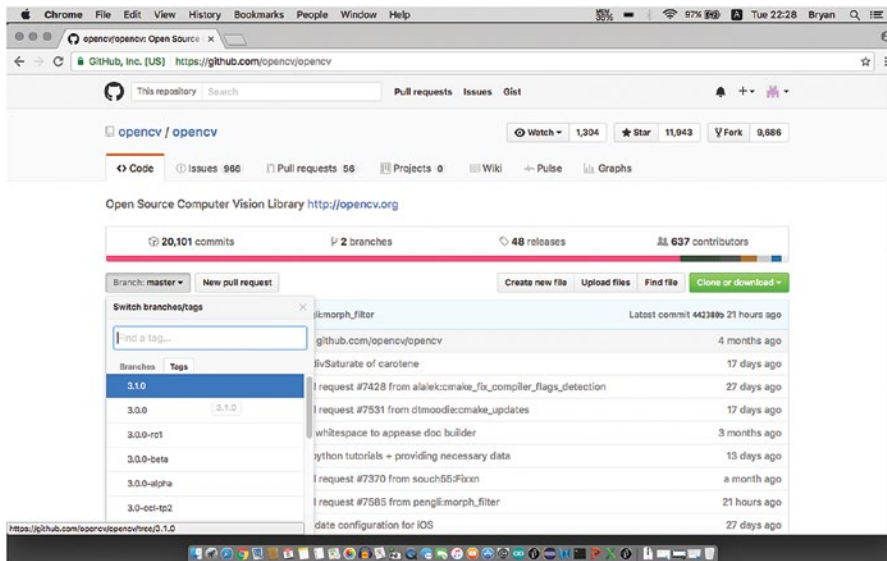


Figure 1-7. Selecting the tag 3.1.0

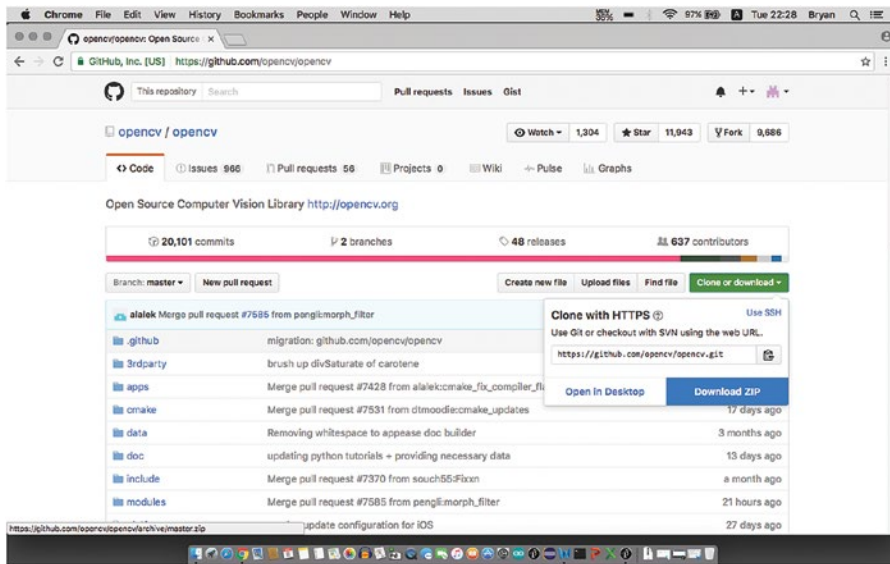


Figure 1-8. Downloading the OpenCV source

After you download and extract the OpenCV source, the process will create the `opencv-3.1.0` folder. For the `opencv_contrib` source, follow the same procedure to select the 3.1.0 tag, download the zip file, and extract it into your `opencv-3.1.0` folder. Figure 1-9 shows the contents of the `opencv-3.1.0` folder.

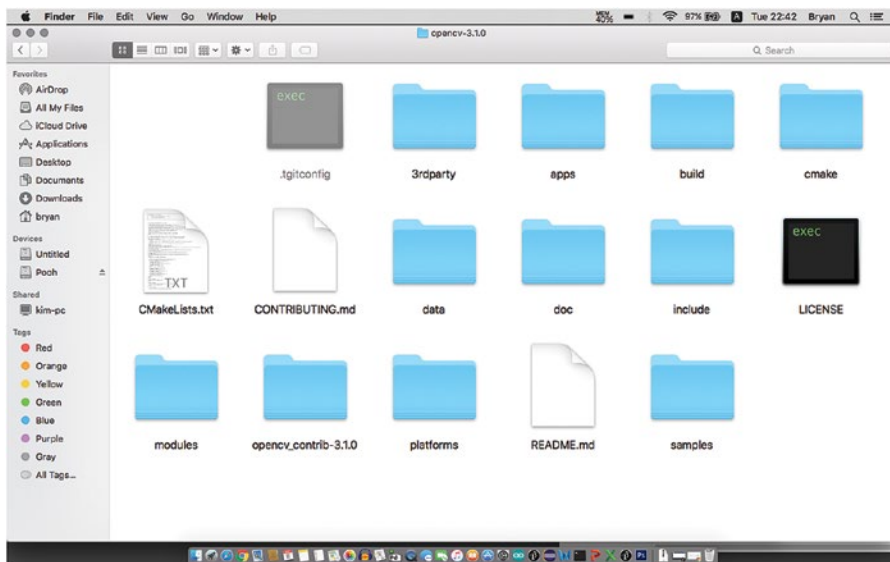


Figure 1-9. Contents of the `opencv-3.1.0` folder

After you successfully download the OpenCV 3.1.0 source and the extra modules library, you can create a subfolder named `build` inside the `opencv-3.1.0` folder. All OpenCV libraries will be built into this folder. Before you start the build process, there is one more step you have to take care of. To build the Java library that includes the extra module `optflow`, which you will use for motion analysis, you have to edit its `CMakeLists.txt` file. From the `opencv_contrib-3.1.0` folder, go into the `modules` folder and then the `optflow` folder. Use any text editor to modify the `CMakeLists.txt` file in the `optflow` folder. In the second line, the original code is as follows:

```
ocv_define_module(optflow opencv_core opencv_imgproc opencv_video opencv_highgui
opencv_ximgproc WRAP python)
```

Insert the token `java` between the two keywords `WRAP` and `python`. The new line will be as follows:

```
ocv_define_module(optflow opencv_core opencv_imgproc opencv_video opencv_highgui
opencv_ximgproc WRAP java python)
```

The new file will enable the build process to include the `optflow` module into the Java library that was built. The following sections describe the different build processes depending on the platform you are using. Since you are going to build the OpenCV Java library, you should also download and install the Java Development Kit (JDK) from the Oracle web site at www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html. To check whether you already have installed the JDK, you can go to a Terminal or command-line session and type the following:

```
javac -version
```

macOS

You are going to use Homebrew to install the necessary dependent software. The installation process will be executed from a command-line Terminal session. The Terminal tool is in the `/Applications/Utilities` folder. The Homebrew installation instructions are on the official web site at <http://brew.sh/>. After you have installed the Homebrew package manager, you can start to install the software required for the OpenCV build process. In a Terminal session, enter the following:

```
brew install cmake
brew install ant
```

These two commands install the software `cmake` and `ant`. The `cmake` tool (<http://cmake.org>) is an open source tool to build, test, and package software. The Apache `ant` tool (<http://ant.apache.org>) is a utility to build Java applications. The next step is to start the configuration process with the `ccmake` interactive tool. First, navigate to the `build` folder of the original OpenCV folder, `opencv-3.1.0`, and issue the `ccmake` command, as shown in Figure 1-10.

```
ccmake ..
```

```

apples-MacBook-Pro:build bryan$ java -version
java version "1.8.0_112"
Java(TM) SE Runtime Environment (build 1.8.0_112-b16)
Java HotSpot(TM) 64-Bit Server VM (build 25.112-b16, mixed mode)
apples-MacBook-Pro:build bryan$ javac -version
javac 1.8.0_112
apples-MacBook-Pro:build bryan$ ant -version
Apache Ant(TM) version 1.10.1 compiled on February 2 2017
apples-MacBook-Pro:build bryan$ ccmake ..

```

Figure 1-10. *ccmake* command to configure the build process

In the *ccmake* panel, type *c* to configure the installation process. Select the appropriate options, as shown in Figure 1-11. Please note that you should first turn off most of the options on the first page, including the `BUILD_SHARED_LIBS` option. Next turn on the `BUILD_opencv_java` option, as shown in Figure 1-12 and Figure 1-13.

```

Page 1 of 8
ANT_EXECUTABLE          */usr/local/bin/ant
BUILD_CUDA_STUBS       *OFF
BUILD_DOCS              *OFF
BUILD_EXAMPLES         *OFF
BUILD_JASPER           *OFF
BUILD_JPEG              *OFF
BUILD_OPENEXR          *OFF
BUILD_PACKAGE          *ON
BUILD_PERF_TESTS       *OFF
BUILD_PNG               *OFF
BUILD_SHARED_LIBS      *OFF
BUILD_TBB               *OFF
BUILD_TESTS            *OFF
BUILD_TIFF              *OFF
BUILD_WITH_DEBUG_INFO  *ON
BUILD_WITH_DYNAMIC_IPP *OFF
BUILD_ZLIB              *ON

BUILD_SHARED_LIBS: Build shared libraries (.dll/.so) instead of static ones (.
Press [enter] to edit option Press [d] to delete an entry  CMake Version 3.7.2
Press [c] to configure
Press [h] for help          Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)

```

Figure 1-11. `BUILD_SHARED_LIBS` and other options

```

Page 2 of 8
BUILD_opencv_apps *OFF
CLAMDBLAS_INCLUDE_DIR *CLAMDBLAS_INCLUDE_DIR-NOTFOUND
CLAMDBLAS_ROOT_DIR *CLAMDBLAS_ROOT_DIR-NOTFOUND
CLAMDFFT_INCLUDE_DIR *CLAMDFFT_INCLUDE_DIR-NOTFOUND
CLAMDFFT_ROOT_DIR *CLAMDFFT_ROOT_DIR-NOTFOUND
CMAKE_BUILD_TYPE *
CMAKE_CONFIGURATION_TYPES *Debug;Release
CMAKE_INSTALL_PREFIX */usr/local
CMAKE_OSX_ARCHITECTURES *
CMAKE_OSX_DEPLOYMENT_TARGET *
CMAKE_OSX_SYSROOT *
CUDA_ARCH_BIN *2.0 2.1(2.0) 3.0 3.5
CUDA_ARCH_PTX *3.0
CUDA_FAST_MATH *OFF
CUDA_GENERATION *
CUDA_HOST_COMPILER */Applications/Xcode.app/Contents/Developer/
CUDA_SEPARABLE_COMPILATION *OFF

BUILD_opencv_apps: Build utility applications (used for example to train class
Press [enter] to edit option Press [d] to delete an entry CMake Version 3.7.2
Press [c] to configure
Press [h] for help Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)

```

Figure 1-12. Second page of the build options

```

Page 3 of 8
CUDA_TOOLKIT_ROOT_DIR */usr/local/cuda
DOWNLOAD_EXTERNAL_TEST_DATA *OFF
EIGEN_INCLUDE_PATH */usr/local/include/eigen3
ENABLE_AVX *OFF
ENABLE_AVX2 *OFF
ENABLE_COVERAGE *OFF
ENABLE_FAST_MATH *OFF
ENABLE_FMA3 *OFF
ENABLE_IMPL_COLLECTION *OFF
ENABLE_NOISY_WARNINGS *OFF
ENABLE_POPCNT *OFF
ENABLE_PRECOMPILED_HEADERS *OFF
ENABLE_PROFILING *OFF
ENABLE_SOLUTION_FOLDERS *OFF
ENABLE_SSE *ON
ENABLE_SSE2 *ON
ENABLE_SSE3 *ON

CUDA TOOLKIT ROOT DIR: Toolkit location.
Press [enter] to edit option Press [d] to delete an entry CMake Version 3.7.2
Press [c] to configure
Press [h] for help Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)

```

Figure 1-13. Third page of the build options

The next important option is `OPENCV_EXTRA_MODULES_PATH`, which should be set to the path name of the OpenCV extra modules. Specifically, it should be the folder `opencv_contrib-3.1.0/modules`, inside your original `opencv-3.1.0` folder, as shown in Figure 1-14.

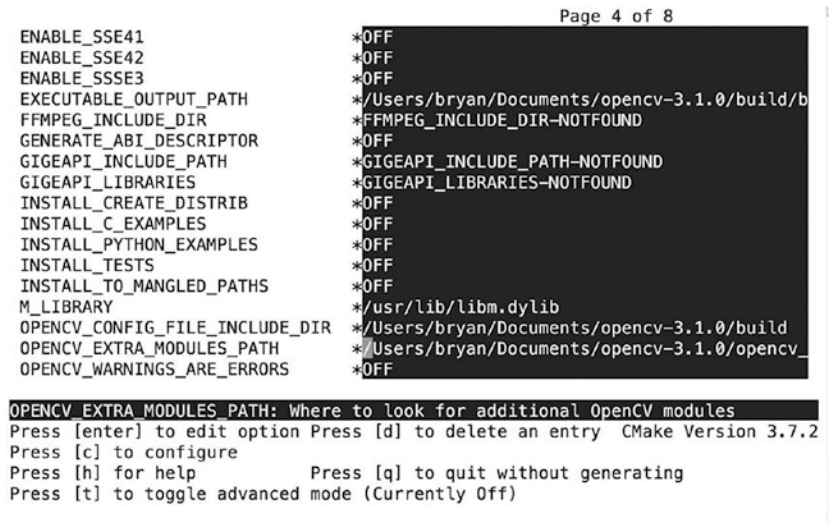


Figure 1-14. `OPENCV_EXTRA_MODULES_PATH` option

The rest of the build options are shown in the following images: Figure 1-15, Figure 1-16, Figure 1-17, and Figure 1-18.

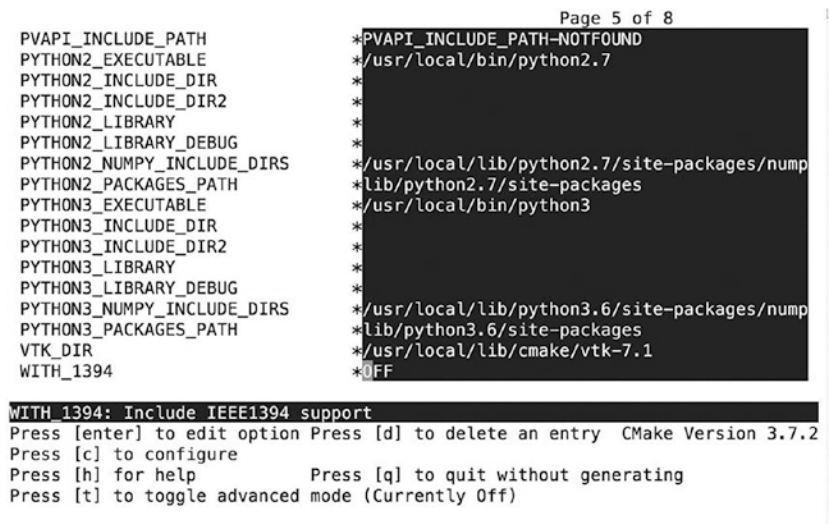


Figure 1-15. OpenCV build options


```

Page 6 of 8
WITH_CARBON          *OFF
WITH_CLP             *OFF
WITH_CUBLAS          *OFF
WITH_CUDA            *OFF
WITH_CUFFT           *OFF
WITH_EIGEN           *OFF
WITH_FFMPEG          *OFF
WITH_GDAL            *OFF
WITH_GIGEAPI         *OFF
WITH_GPHOTO2         *OFF
WITH_GSTREAMER       *OFF
WITH_GSTREAMER_0_10 *OFF
WITH_IPP             *ON
WITH_IPP_A           *OFF
WITH_JASPER          *OFF
WITH_JPEG            *OFF
WITH_LIBV4L          *OFF

WITH_LIBV4L: Use libv4l for Video 4 Linux support
Press [enter] to edit option Press [d] to delete an entry CMake Version 3.7.2
Press [c] to configure
Press [h] for help          Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)

```

Figure 1-16. OpenCV build options, continued

```

Page 7 of 8
WITH_MATLAB          *OFF
WITH_OPENCCL         *ON
WITH_OPENCCLAMDBLAS *OFF
WITH_OPENCCLAMDFFT  *OFF
WITH_OPENCCL_SVM     *OFF
WITH_OPENEXR         *OFF
WITH_OPENGL          *ON
WITH_OPENMP          *OFF
WITH_OPENNI          *OFF
WITH_OPENNI2         *OFF
WITH_PNG             *OFF
WITH_PTHREADS_PF    *ON
WITH_PVAPI           *OFF
WITH_QT              *OFF
WITH_QUICKTIME       *OFF
WITH_TBB             *OFF
WITH_TIFF            *OFF

WITH_TIFF: Include TIFF support
Press [enter] to edit option Press [d] to delete an entry CMake Version 3.7.2
Press [c] to configure
Press [h] for help          Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)

```

Figure 1-17. OpenCV build options, continued

```

Page 8 of 8
WITH_V4L *OFF
WITH_VA *OFF
WITH_VA_INTEL *OFF
WITH_VTK *OFF
WITH_WEBP *OFF
WITH_XIMEA *OFF

WITH_XIMEA: Include XIMEA cameras support
Press [enter] to edit option Press [d] to delete an entry CMake Version 3.7.2
Press [c] to configure
Press [h] for help Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)

```

Figure 1-18. Last page of OpenCV build options

After filling in the first round of build options, type c again to configure the extra modules. First, turn off the BUILD_FAT_JAVA_LIB option, as shown in Figure 1-19.

```

Page 1 of 11
BUILD_FAT_JAVA_LIB *OFF
BUILD_LIBPROTOBUF_FROM_SOURCES *ON
BUILD_opencv_aruco *OFF
BUILD_opencv_bgsegm *OFF
BUILD_opencv_bioinspired *OFF
BUILD_opencv_calib3d *ON
BUILD_opencv_ccalib *OFF
BUILD_opencv_contrib_world *OFF
BUILD_opencv_core *ON
BUILD_opencv_datasets *OFF
BUILD_opencv_dnn *OFF
BUILD_opencv_dpm *OFF
BUILD_opencv_face *OFF
BUILD_opencv_features2d *ON
BUILD_opencv_flann *ON
BUILD_opencv_fuzzy *OFF
BUILD_opencv_hdf *OFF

BUILD_opencv_hdf: Include opencv_hdf module into the OpenCV build
Press [enter] to edit option Press [d] to delete an entry CMake Version 3.7.2
Press [c] to configure
Press [h] for help Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)

```

Figure 1-19. OpenCV extra modules build options

To work on with the optical flow examples later in the book, you also should turn on the options for `BUILD_opencv_optflow`, `BUILD_opencv_ximgproc`, and `BUILD_opencv_java`, as shown in Figure 1-20 and Figure 1-21.

```

Page 2 of 11
BUILD_opencv_highgui          *ON
BUILD_opencv_imgcodecs       *ON
BUILD_opencv_imgproc         *ON
BUILD_opencv_java            *ON
BUILD_opencv_line_descriptor *OFF
BUILD_opencv_ml              *ON
BUILD_opencv_objdetect       *ON
BUILD_opencv_optflow         *ON
BUILD_opencv_photo           *ON
BUILD_opencv_plot            *OFF
BUILD_opencv_reg              *OFF
BUILD_opencv_rgbd            *OFF
BUILD_opencv_saliency         *OFF
BUILD_opencv_shape           *ON
BUILD_opencv_stereo          *ON
BUILD_opencv_stitching       *ON
BUILD_opencv_structured_light *OFF

BUILD_opencv_java: Include opencv_java module into the OpenCV build
Press [enter] to edit option Press [d] to delete an entry CMake Version 3.7.2
Press [c] to configure
Press [h] for help          Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)

```

Figure 1-20. Turning on options for Java and `optflow`

```

Page 3 of 11
BUILD_opencv_superres        *ON
BUILD_opencv_surface_matching *OFF
BUILD_opencv_text            *OFF
BUILD_opencv_tracking        *OFF
BUILD_opencv_video           *ON
BUILD_opencv_videoio         *ON
BUILD_opencv_videostab       *ON
BUILD_opencv_world           *OFF
BUILD_opencv_xfeatures2d     *OFF
BUILD_opencv_ximgproc        *ON
BUILD_opencv_xobjdetect      *OFF
BUILD_opencv_xphoto          *OFF
Caffe_INCLUDE_DIR            *Caffe_INCLUDE_DIR-NOTFOUND
Caffe_LIBS                    *Caffe_LIBS-NOTFOUND
Ceres_DIR                     *Ceres_DIR-NOTFOUND
Glog_LIBS                     *Glog_LIBS-NOTFOUND
HDF5_C_LIBRARY_d1            */usr/lib/libdl.dylib

BUILD_opencv_ximgproc: Include opencv_ximgproc module into the OpenCV build
Press [enter] to edit option Press [d] to delete an entry CMake Version 3.7.2
Press [c] to configure
Press [h] for help          Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)

```

Figure 1-21. Turning on the option for `ximgproc`

Complete the rest of the extra modules options, as shown in Figure 1-22.



Figure 1-22. Extra module options

After setting all the options, type c again to finish the last configuration task. Type the option g to generate the configuration files (Figure 1-23). The ccmake program will quit and take you back to the Terminal window.



Figure 1-23. Generating the configuration file

Enter the following command to start the build process:

```
make -j4
```

When the build process completes successfully, navigate into the bin folder within the build folder. Locate the opencv-310.jar file. Then navigate into the lib folder within the build folder. Locate the libopencv_java310.so file. Rename it to libopencv_java310.dylib. Copy the two files into a separate folder. You are going to prepare the Windows and Linux versions and copy them to the same folder to create the multiplatform

library. The author has tested building the OpenCV 3.1 in the macOS 10.11, El Capitan. For readers using the new macOS 10.12 Sierra, the building of OpenCV 3.1 will fail due to the removal of QtKit. In this case, it is better to use OpenCV 3.2 together with macOS 10.12. Please refer to the building note in <http://www.magicandlove.com/blog/2017/03/02/opencv-3-2-java-build/> to generate the optflow module properly with OpenCV 3.2.

Windows

On a Windows system, you use the graphical version of cmake to configure the installation process. I have tested the installation in Windows 8.1 and Windows 10. Download and install the following software packages for the OpenCV build process:

- Microsoft Visual Studio Community 2015 at <https://www.visualstudio.com/downloads/>
- CMake at <https://cmake.org/download/>
- Apache Ant at <http://ant.apache.org/bindownload.cgi>
- Oracle JDK 8 at www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
- Python at <https://www.python.org/downloads/>

After the successful installation of the software package dependencies, run the CMake (cmake-gui) program to start the configuration process. Fill in the source folder name for the OpenCV distribution and the build folder name, as shown in Figure 1-24. Remember that you need to create the build folder inside the OpenCV distribution folder.

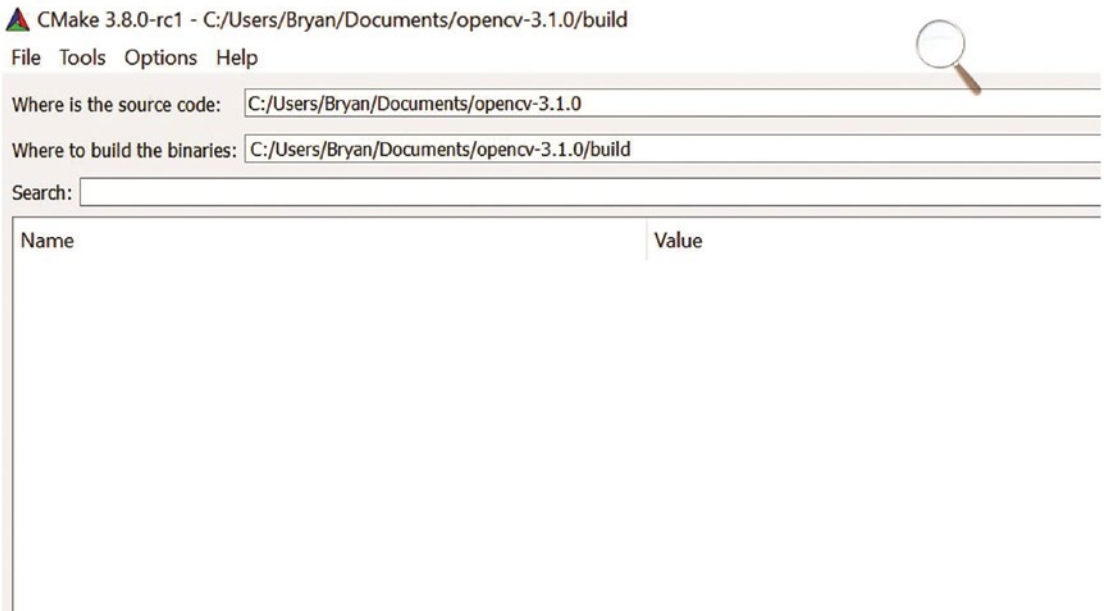


Figure 1-24. Folder names of OpenCV distribution in the CMake window

Click the Configure button to start the configuration. For the first generator panel, choose Visual Studio 14 2015 Win64 from the pull-down menu and select the “Use default native compilers” radio button, as shown in Figure 1-25. Click the Finish button to proceed.

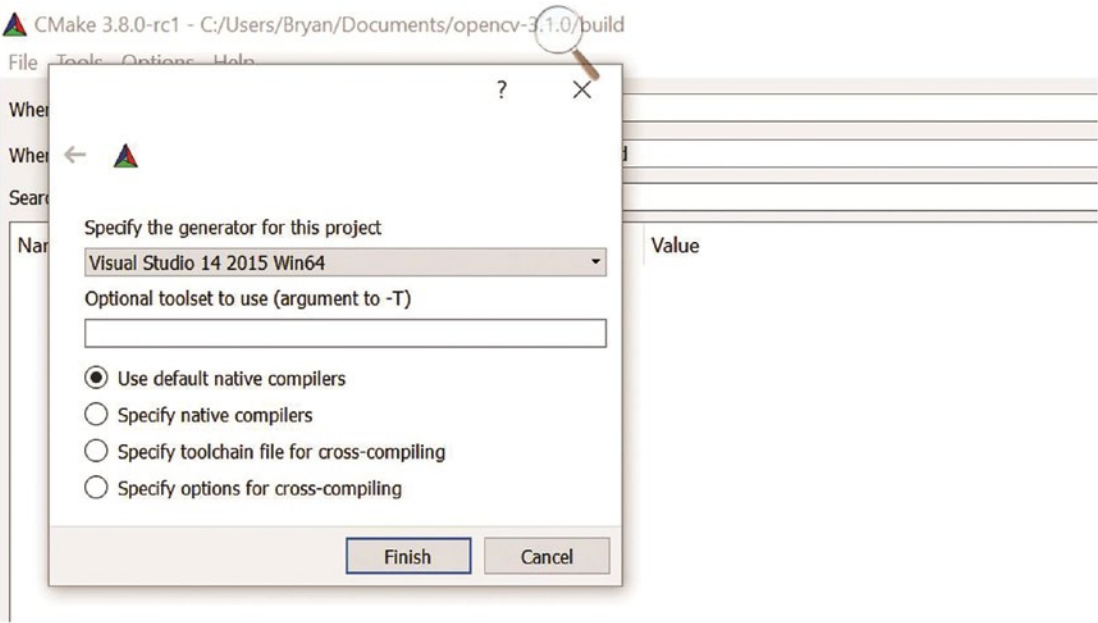


Figure 1-25. Choosing the default compiler

Follow Figure 1-26 through Figure 1-33 to enter the build options. Make sure to first turn off the BUILD_SHARED_LIBS option, and enter the path name for ant.bat for the ANT_EXECUTABLE option.

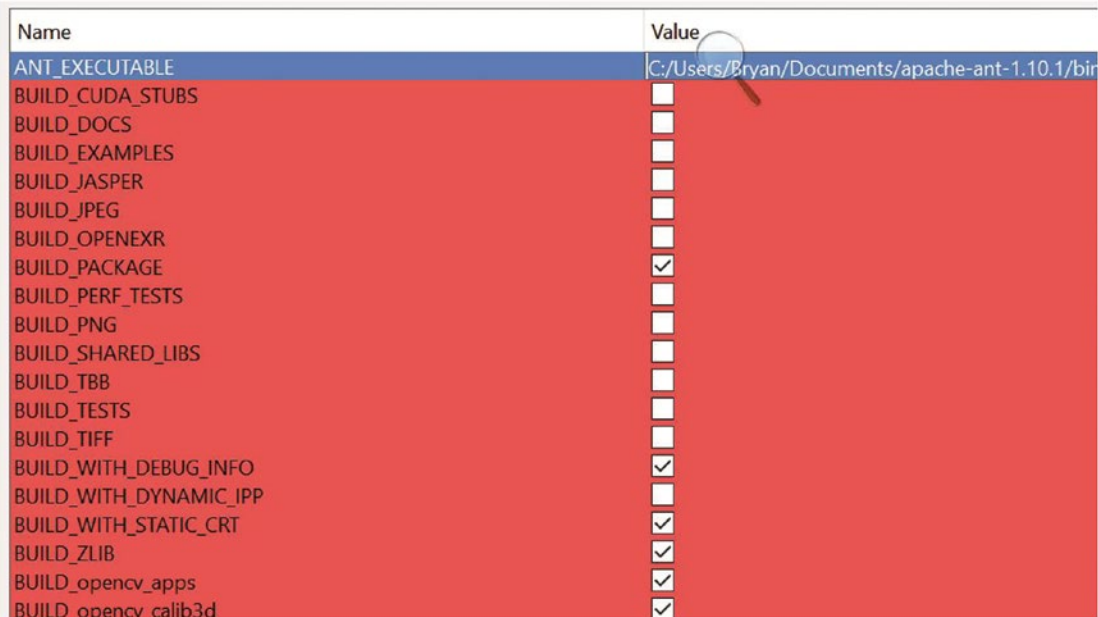


Figure 1-26. Turning off the BUILD_SHARED_LIBS option