



C o m m u n i t y E x p e r i e n c e D i s t i l l e d

OpenCV for Secret Agents

Use OpenCV in six secret projects to augment your home, car, phone, eyesight, and any photo or drawing

Joseph Howse

[PACKT] open source*
PUBLISHING community experience distilled

OpenCV for Secret Agents

Use OpenCV in six secret projects to augment your home, car, phone, eyesight, and any photo or drawing

Joseph Howse



BIRMINGHAM - MUMBAI

OpenCV for Secret Agents

Copyright © 2015 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: January 2015

Production reference: 1230115

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78328-737-6

www.packtpub.com

Cover image by Jeremy Segal (info@jsegalphoto.com)

Credits

Author

Joseph Howse

Project Coordinator

Sageer Parkar

Reviewers

Karan Kedar Balkar
Michael Beyeler
Demetris Gerogiannis
Kevin Hughes
Ganesh Iyer
Andrew Colin Kissa
Lihang Li
Ryohei Tanaka

Proofreaders

Simran Bhogal
Ameesha Green
Paul Hindle
Clyde Jenkins

Indexer

Monica Ajmera Mehta

Graphics

Sheetal Aute
Abhinash Sahu

Commissioning Editor

Sam Birch

Production Coordinator

Arvindkumar Gupta

Acquisition Editors

Sam Birch
Richard Brookes-Bland

Cover Work

Arvindkumar Gupta

Content Development Editor

Arwa Manasawala

Technical Editor

Aman Preet Singh

Copy Editor

Neha Vyas

About the Author

Joseph Howse has four first-rate cats; yet, if his books sell well, he could build a menagerie fit for a pharaoh.

OpenCV for Secret Agents is Joseph's third book, following *OpenCV Computer Vision with Python* and *Android Application Programming with OpenCV*. When not writing books or grooming cats, Joseph is working to grow the augmented reality industry by providing software development and training services through his company, Nummist Media (<http://nummist.com>).

Acknowledgments

Many people, near and far, have guided this book to completion.

My parents, Jan and Bob, have given me nine full lives or so it seems. My four cats, Plasma Tigerlily Zoya, Sanibel Delphinium Andromeda, Lambda Catculus Puddingcat, and Josephine Antoinette Puddingcat, have provided constant supervision and contributed to testing the cat recognition software in *Chapter 3, Training a Smart Alarm to Recognize the Villain and His Cat*.

My readers and listeners have taken time to provide valuable feedback and ask questions about my previous books and presentations. Thanks to their loyalty and dedication to discovery, our shared exploration of OpenCV goes on!

My clients at Market Beat, in El Salvador, have inspired several of the book's topics, including detection, recognition, tracking, and the use of Raspberry Pi.

Thanks, Steven Puttemans, for the helpful discussion on Haar features. Thanks, Tanya Suhodolska, for icons used in the application bundles.

My editors at Packt Publishing have once again given me all the benefit of their skill, experience, and professionalism in the planning, polishing, and marketing of this book. Writing one of Packt's "Secret Agent" books has been a uniquely fun project! Thanks, Sam Birch, for suggesting Eulerian video magnification as the topic of *Chapter 6, Seeing a Heartbeat with a Motion Amplifying Camera*.

My technical reviewers have once again saved me from sundry errors and omissions. Read their biographies here! They are fine members of the OpenCV community.

Sam Howse, Bunny Moir, and dear old cats — you are remembered for the love, laughter, learning, and long journeys home.

About the Reviewers

Karan Kedar Balkar has been working as an independent Android application developer since the past 4 years. Born and brought up in Mumbai, he holds a bachelor's degree in computer engineering. He has written over 50 programming tutorials on his personal blog (<http://karanbalkar.com>) that covers popular technologies and frameworks.

At present, he is working as a software engineer. He has been trained on various technologies, including Java, Oracle, and .NET. Apart from being passionate about technology, he loves to write poems and travel to different places. He likes listening to music and enjoys playing the guitar.

First, I would like to thank my parents for their constant support and encouragement. I would also like to thank my friends, Srivatsan Iyer, Ajit Pillai, and Prasaanth Neelakandan, for always inspiring and motivating me.

I would like to express my deepest gratitude to Packt Publishing for giving me a chance to be a part of the reviewing process.

Michael Beyeler is a PhD student in the department of computer science at the University of California, Irvine, where he is working on large-scale cortical models of biological vision, motion, learning, and memory, as well as their implementation on GPGPUs and their applications for cognitive robotics. He received a bachelor's of science degree in electrical engineering and information technology in 2009 as well as a master's of science degree in biomedical engineering in 2011 from ETH Zurich, Switzerland.

Demetris Gerogiannis received his bachelor's of science, master's of science, and PhD degrees from the department of Computer Science & Engineering, University of Ioannina, Greece in 2004, 2007, and 2014, respectively. He is an active researcher of computer vision, and his research interests include image segmentation and registration, point set registration, feature extraction, pattern recognition, and autonomous navigation. His work has been presented in international conferences, and it has been published in international journals. He is an IEEE student member, and from October 2012 till June 2013, he was the interim chair at the IEEE student branch of the University of Ioannina. Since his early years in the university, he was interested in entrepreneurship. He was involved in several entrepreneurial technological ventures, and he has participated in several international business competitions. The most important one was the participation at the Startupbootcamp for NFC & Contactless, communication selection days, in Amsterdam in September 2013. His start-up was selected among 250 plus international start-ups that had applied for this accelerator project. He is also involved in the development of his local region start-up ecosystem. To that end, he has cofounded in 2014 a nonprofit team (StartupLake) with a view to provide mentorship to young ambitious people who want to have their own start-ups.

Ryohei Tanaka is a software engineer at Yahoo! Japan Corporation. His focus is now on machine learning, information extraction, and distributed computing. He has sound knowledge of image processing on web browser (HTML5/JavaScript). More details on his programming skills, interests, and experience about computer vision can be found at <http://rest-term.com>.

Congrats to the author and all those who worked on this book, and thanks to the editors and publishers who gave me a chance to work on the publication of this book.

www.PacktPub.com

Support files, eBooks, discount offers, and more

For support files and downloads related to your book, please visit www.PacktPub.com.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at service@packtpub.com for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.



<http://PacktLib.PacktPub.com>

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can search, access, and read Packt's entire library of books.

Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print, and bookmark content
- On demand and accessible via a web browser

Free access for Packt account holders

If you have an account with Packt at www.PacktPub.com, you can use this to access PacktLib today and view 9 entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface	1
Chapter 1: Preparing for the Mission	7
Setting up a development machine	8
Windows	10
OpenCV on Windows with binary installers	12
OpenCV on Windows with CMake and compilers	12
Mac	15
Mac with MacPorts	16
Mac with Homebrew	18
Debian Wheezy and its derivatives, including Raspbian, Ubuntu, and Linux Mint	20
Fedora and its derivatives, including RHEL and CentOS	22
openSUSE and its derivatives	22
Tegra Android Development Pack	23
Building OpenCV Android sample projects with Eclipse	25
Unity	34
Setting up Raspberry Pi	34
Setting up the Raspberry Pi Camera Module	39
Finding OpenCV documentation, help, and updates	40
Alternatives to Raspberry Pi	41
Summary	42
Chapter 2: Searching for Luxury Accommodations Worldwide	43
Planning the Luxocator app	44
Creating, comparing, and storing histograms	45
Training the classifier with reference images	52
Acquiring images from the Web	53
Acquiring images from Bing image search	55
Preparing images and resources for the app	60

Integrating everything into the GUI	63
Building Luxocator for distribution	71
Summary	74
Chapter 3: Training a Smart Alarm to Recognize the Villain and His Cat	75
Understanding machine learning in general	77
Planning the Interactive Recognizer app	78
Understanding Haar cascades and LBPH	80
Implementing the Interactive Recognizer app	84
Planning the cat detection model	98
Implementing the training script for the cat detection model	100
Planning the Angora Blue app	114
Implementing the Angora Blue app	115
Building Angora Blue for distribution	122
Further fun with finding felines	122
Summary	122
Chapter 4: Controlling a Phone App with Your Suave Gestures	123
Planning the Goldgesture app	124
Understanding optical flow	126
Setting up the Eclipse Workspace	129
Getting a cascade file and audio files	138
Specifying the app's requirements	138
Laying out a camera preview as the main view	139
Tracking back and forth gestures	140
Playing audio clips as questions and answers	143
Capturing images and tracking faces in an activity	147
Summary	164
Chapter 5: Equipping Your Car with a Rearview Camera and Hazard Detection	165
Planning The Living Headlights app	167
Detecting lights as blobs	169
Estimating distances (a cheap approach)	172
Implementing The Living Headlights app	175
Testing The Living Headlights app at home	189
Testing The Living Headlights app in a car	193
Summary	199

Chapter 6: Seeing a Heartbeat with a Motion Amplifying Camera	201
Planning the Lazy Eyes app	203
Understanding what Eulerian video magnification can do	205
Extracting repeating signals from video using the Fast Fourier Transform (FFT)	206
Choosing and setting up an FFT library	207
Compositing two images using image pyramids	210
Implementing the Lazy Eyes app	211
Configuring and testing the app for various motions	220
Seeing things in another light	228
Summary	229
Chapter 7: Creating a Physics Simulation Based on a Pen and Paper Sketch	231
Planning the Rollingball app	233
Detecting circles and lines	236
Setting up OpenCV for Unity	239
Configuring and building the Unity project	242
Creating the Rollingball scene in Unity	244
Creating Unity assets and adding them to the scene	247
Writing shaders and creating materials	247
Creating physics materials	250
Creating prefabs	252
Writing our first Unity script	256
Writing the main Rollingball script	258
Tidying up and testing	278
Summary	279
Index	281

Preface

Computer vision systems are deployed in the Arctic Ocean to spot icebergs at night. They are flown over the Amazon rainforest to create aerial maps of fires, blights, and illegal logging. They are set up in ports and airports worldwide to scan for suspects and contraband. They are sent to the depths of the Marianas Trench to guide autonomous submarines. They are used in operating rooms to help surgeons visualize the planned procedure and the patient's current condition. They are launched from battlefields as the steering systems of heat-seeking, anti-aircraft rockets.

We might seldom — or never — visit these places. However, stories often encourage us to imagine extreme environments and a person's dependence on tools in these unforgiving conditions. Perhaps fittingly, one of contemporary fiction's most popular characters is an almost ordinary man (handsome but not too handsome, clever but not too clever) who wears a suit, works for the British Government, always chooses the same drink, the same kind of woman, the same tone for delivering a pun, and is sent to do dangerous jobs with a peculiar collection of gadgets.

Bond. James Bond.

This book teaches seriously useful technologies and techniques with a healthy dose of inspiration from spy fiction. The Bond franchise is rich in ideas about detection, disguise, smart devices, image capture, and sometimes even computer vision specifically. With imagination, plus dedication of learning new skills, we can become the next generation of gadget makers to rival Bond's engineer, Q!

What this book covers

Chapter 1, Preparing for the Mission helps us to install OpenCV, a Python development environment, and an Android development environment on Windows, Mac, or Linux systems. In this chapter, we also install a Unity development environment on Windows or Mac.

Chapter 2, Searching for Luxury Accommodations Worldwide helps us to classify images of real estate based on color schemes. Are we outside a luxury dwelling or inside a Stalinist apartment? In this chapter, we use the classifier in a search engine that labels its image results.

Chapter 3, Training a Smart Alarm to Recognize the Villain and His Cat helps us to detect and recognize human faces and cat faces as a means of controlling an alarm. Has Ernst Stavro Blofeld returned with his blue-eyed Angora cat?

Chapter 4, Controlling a Phone App with Your Suave Gestures helps us to detect motion and recognize gestures as a means of controlling a guessing game on a smartphone. The phone knows why Bond is nodding even if no one else does.

Chapter 5, Equipping Your Car with a Rearview Camera and Hazard Detection helps us to detect car headlights, classify their color, estimate distances to them, and provide feedback to a driver. Is that car tailing us?

Chapter 6, Seeing a Heartbeat with a Motion Amplifying Camera helps us to amplify motion in live video, in real time, so that a person's heartbeat and breathing become clearly visible. See the passion!

Chapter 7, Creating a Physics Simulation Based on a Pen and Paper Sketch helps us to draw a ball-in-a-maze puzzle on paper and see it come to life as a physics simulation on a smartphone. Physics and timing are everything!

What you need for this book

This book supports several operating systems as development environments, including Windows XP or a later version, Mac OS X 10.6 or a later version, Debian Wheezy, Raspbian, Ubuntu 12.04 or a later version, Linux Mint 13 or a later version, Fedora 18 or a later version, CentOS 7 or a later version, and openSUSE 13.1 or a later version.

The book contains six projects with the following requirements:

- Four of these six projects run on Windows, Mac, or Linux and require a webcam. Optionally, these projects can use Raspberry Pi or another single-board computer that runs Linux.

- One project runs on Android 2.2 or a later version and requires a front-facing camera (which most Android devices have).
- One project runs on Android 2.3 or a later version and requires a rear-facing camera and gravity sensor (which most Android devices have). For development, it requires a Windows or Mac machine and approximately \$75 worth of game development software.

Setup instructions for all required libraries and tools are covered in the book. Optional setup instructions for Raspberry Pi are also included.

Who this book is for

This book is for tinkerers (and spies) who want to make computer vision a practical and fun part of their lifestyle. You should already be comfortable with 2D graphic concepts, object-oriented languages, GUIs, networking, and command line. This book does not assume experience with any specific libraries or platforms. Detailed instructions cover everything from setting up the development environment to deploying finished apps.

A desire to learn multiple technologies and techniques and to integrate them is highly beneficial! This book will help you branch out to understand several types of systems and application domains where computer vision is relevant, and it will help you to apply several approaches to detect, recognize, track, and augment faces, objects, and motions.

Conventions

In this book, you will find a number of text styles that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "You can edit `/etc/modules` to check whether `bcm2835-v4l2` is already listed there."

A block of code is set as follows:

```
set PYINSTALLER=C:\PyInstaller\pyinstaller.py

REM Remove any previous build of the app.
rmdir build /s /q
```



```
rmdir dist /s /q

REM Train the classifier.
python HistogramClassifier.py
```


When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:


```
<activity
  android:name="com.nummist.goldgesture.CameraActivity"
  android:label="@string/app_name"
  android:screenOrientation="landscape"
  android:theme="@android:style/Theme.NoTitleBar.Fullscreen">
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name=
      "android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>
```

Any command-line input or output is written as follows:

```
$ echo "bcm2835-v4l2" | sudo tee -a /etc/modules
```

New terms and **important words** are shown in bold. Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "Click on the link for **Bing Search API** (not any variant such as **Bing Search API - Web Results Only**)."

 Warnings or important notes appear in a box like this.

 Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or disliked. Reader feedback is important for us as it helps us develop titles that you will really get the most out of.

To send us general feedback, simply e-mail feedback@packtpub.com, and mention the book's title in the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide at www.packtpub.com/authors.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files from your account at <http://www.packtpub.com> for all the Packt Publishing books you have purchased. If you purchased this book elsewhere, you can visit <http://www.packtpub.com/support> and register to have the files e-mailed directly to you. The latest and updated example code for this book is also available from the author's website at <http://nummist.com/opencv/>.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books – maybe a mistake in the text or the code – we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting <http://www.packtpub.com/submit-errata>, selecting your book, clicking on the **ErrataSubmissionForm** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the Errata section of that title.

To view the previously submitted errata, go to <https://www.packtpub.com/books/content/support> and enter the name of the book in the search field. The required information will appear under the **Errata** section.

Piracy

Piracy of copyrighted material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works in any form on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at copyright@packtpub.com with a link to the suspected pirated material.

We appreciate your help in protecting our authors and our ability to bring you valuable content.

Questions

If you have a problem with any aspect of this book, you can contact us at questions@packtpub.com, and we will do our best to address the problem. You can also contact the author directly at josephhowse@nummist.com or you can check his website, <http://nummist.com/opencv/>, for answers to common questions about this book.

1

Preparing for the Mission

Q: I've been saying for years, sir, that our special equipment is obsolete. And now, computer analysis reveals an entirely new approach: miniaturization.

On Her Majesty's Secret Service (1969)

James Bond is not a pedestrian. He cruises in a submarine car, he straps on a rocket belt, and oh, how he skis, how he skis! He always has the latest stuff and he is never afraid to put a dent in it, much to the dismay of Q, the engineer.

As software developers in the 2010s, we are witnessing an explosion in the adoption of new platforms. Under one family's roof, we might find a mix of Windows, Mac, iOS, and Android devices. Mom and Dad's workplaces provide different platforms. The kids have three game consoles or five if you count the mobile versions. The toddler has a LeapFrog learning tablet. Smart glasses are becoming more affordable.

We must not be afraid to try new platforms and consider new ways to combine them. After all, most users do.

This book embraces multi-platform development. It presents weird and wonderful applications that we can deploy in unexpected places. It uses several of the computer's senses, but especially uses computer vision to breathe new life into the humdrum, heterogeneous clutter of devices that surround us.

Before Agent 007 runs amok with the gadgets, he is obligated to listen to Q's briefing. This chapter performs Q's role. This is the setup chapter.

By the end of this chapter, you will obtain all the tools to develop OpenCV applications in C++ or Python for Windows, Mac, or Linux, and in C++ or Java for Android. You will also be the proud new user of a Raspberry Pi single-board computer (this additional hardware is optional). You will even know a bit about Unity, a game engine into which we can integrate OpenCV.

If you find yourself a bit daunted by the extent of this setup chapter, be reassured that not all of the tools are required and no single project uses all of them in combination. Although Q and I live for the big event of setting up multiple technologies at once, you could just skim this chapter and refer back to it later when the tools become useful, one by one, in our projects.



Where basic OpenCV setup and reference materials are concerned, this chapter includes excerpts from my introductory books, *OpenCV Computer Vision with Python* and *Android Application Programming with OpenCV*, published by Packt Publishing. All contents are retested, updated, and expanded to cover newer OpenCV versions and additional operating systems. Also, there are all new sections on the optional hardware and game engine used in this book.

Setting up a development machine

We can develop our OpenCV applications on a desktop, a notebook, or even the humble Raspberry Pi (covered later in the *Setting up a Raspberry Pi* section). Most of our apps have a memory footprint of less than 128 MB, so they can still run (albeit slowly) on old or low-powered machines. To save time, develop on your fastest machine first and test on slower machines later.

This book assumes that you have one of the following operating systems on your development machine:

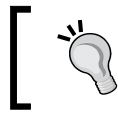
- Windows XP or a later version
- Mac OS 10.6 or a later version
- Debian Wheezy or a derivative such as the following:
 - Raspbian
 - Ubuntu 12.04 or a later version
 - Linux Mint 13 or a later version
- Fedora 18 or a later version, or a derivative such as the following:
 - Red Hat Enterprise Linux (RHEL) 7 or a later version
 - CentOS 7 or a later version
- openSUSE 13.1 or a later version, or a derivative

Other Unix-like systems can also work but they are not covered in this book.

You should have a USB webcam and any necessary drivers. Most webcams come with instructions for installing drivers on Windows and Mac. Linux distributions typically include the **USB Video Class (UVC)** Linux driver, which supports many webcams, listed at <http://www.ideasonboard.org/uvc/#devices>.

We are going to set up the following components:

- A C++ development environment. On Windows, we will use Visual Studio 2010 or a later version. Alternatively, Windows users can follow Kevin Hughes' helpful tutorial on setting up OpenCV with MinGW and the Code::Blocks IDE at <http://kevinhughes.ca/tutorials/opencv-install-on-windows-with-codeblocks-and-mingw/>. On Mac, we will use Xcode. On Linux, we will use GCC, which comes as standard.
- On Mac, we will use a third-party package manager to help us install libraries and their dependencies. We will use either MacPorts or Homebrew.
- A Python 2.7 development environment. At the time of writing, the best option is to use version 2.7 as it is the most recent Python version supported by OpenCV's stable branch. (Python 2.6 is also supported by the stable branch.)
- Popular libraries such as NumPy (for numeric functions), SciPy (for numeric and scientific functions), Requests (for web requests), and wxPython (for cross-platform GUIs).
- PyInstaller, a cross-platform tool used for bundling Python scripts, libraries, and data as redistributable apps, such that users machines do not require installations of Python, OpenCV, and other libraries. For this book's purposes, building redistributables of Python projects is an optional topic. We will cover the basics in *Chapter 2, Searching for Luxury Accommodations Worldwide*, but you might need to do your own testing and debugging as PyInstaller (like other Python bundling tools) does not show entirely consistent behavior across operating systems, Python versions, and library versions. It is not well supported on Raspberry Pi or other ARM systems.
- A build of OpenCV with C++ and Python support plus optimizations for certain desktop hardware. At the time of writing, OpenCV 2.4.x is the stable branch and our instructions are tailored for this branch.
- Another build of OpenCV with C++ and Java support plus optimizations for certain Android hardware. Specifically, we will use the OpenCV build that comes with **Tegra Android Development Pack (TADP)**. At the time of writing, TADP 3.0r4 is the most recent release.
- An Android development environment, including Eclipse, ADT, Android SDK, and Android NDK. TADP includes these too.
- On Windows or Mac, a 3D game engine called Unity.



Eclipse has a big memory footprint. Even if you want to use Raspberry Pi for developing desktop and Pi apps, use something with more RAM for developing Android apps.

Let's break this setup down into three sets of platform-dependent steps, plus a set of platform-independent steps for TADP, and another set of platform-independent steps for Unity.

Windows

On Windows, we have the option of setting up a 32-bit development environment (to make apps that are compatible with both 32-bit and 64-bit Windows) or a 64-bit development environment (to make optimized apps that are compatible with 64-bit Windows only). Recent versions of OpenCV are available in 32-bit and 64-bit versions.

We also have a choice of either using binary installers or compiling OpenCV from source. For our Windows apps in this book, the binary installers provide everything we need. However, we will also discuss the option of compiling from source because it enables us to configure additional features, such as support for Kinect and Asus depth cameras, which might be relevant to your future work or to our projects in other books.



For an OpenCV project that uses a depth camera, refer to my book *OpenCV Computer Vision with Python*, published by Packt Publishing.

Regardless of our approach to obtain OpenCV, we need a general-purpose C++ development environment and a general-purpose Python 2.7 development environment. We will set up these environments using binary installers.

As our C++ development environment, we will use Visual Studio 2010 or a later version. Use any installation media you might have purchased, or go to the downloads page at <http://www.visualstudio.com/en-us/downloads/download-visual-studio-vs.aspx>. Download and run the installer for one of the following:

- Visual C++ 2010 Express, which is free
- Visual Studio Express 2013 for Windows desktop, which is free
- Any of the paid versions, which have 90-day free trials

If the installer lists optional C++ components, we should opt to install them all. After the installer runs till completion, reboot.



If we plan to compile OpenCV from source (as described in the *OpenCV on Windows with CMake and Compilers* section), I recommend you use Visual Studio 2010 (and not any later version). At the time of writing, OpenCV and some of its optional dependencies do not compile easily with Visual Studio 2012 or Visual Studio 2013.

Installers for Python 2.7 are available at <http://www.python.org/getit/>. Download and run the latest revision of Python 2.7 in either the 32-bit variant or the 64-bit variant.

To make Python scripts run using our new Python 2.7 installation by default, let's edit the system's Path variable and append ;C:\Python2.7 (assuming Python 2.7 is installed in the default location). Remove any previous Python paths, such as ;C:\Python2.6. Log out and log back in (or reboot).

Let's assume that we also want to use binary installers for NumPy, SciPy, and wxPython. Download and run the installers for the latest stable library versions that target Python 2.7. We can find these installers at the following locations:

1. **NumPy:** The official installers are 32-bit only and are located at <http://sourceforge.net/projects/numpy/files/NumPy/>. Unofficial 64-bit installers are available at <http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy>.
2. **SciPy:** The official installers are 32-bit only and are located at <http://sourceforge.net/projects/scipy/files/scipy/>. Unofficial 64-bit installers are available at <http://www.lfd.uci.edu/~gohlke/pythonlibs/#scipy>.
3. **wxPython:** This can be downloaded from <http://www.wxpython.org/download.php>. The apps in this book are successfully tested with wxPython 2.8, 2.9, and 3.0. If in doubt, choose version 3.0. However, if you choose version 2.8, get its Unicode variant.

Requests does not have a binary installer but we can download the latest source bundle from <https://github.com/kennethreitz/requests/archive/master.zip>. Unzip it to any destination, which we will refer to as <unzip_destination>. Open Command Prompt and run the following commands:

```
> cd <unzip_destination>
> python setup.py install
```


Next, we can put PyInstaller in any convenient location, since it is treated as a set of tools rather than a library. Let's download the latest release version from <http://www.pyinstaller.org/> and unzip it to C:\PyInstaller or any another location of your choice.

Now, we are ready to set up OpenCV and, optionally, other computer vision libraries.

OpenCV on Windows with binary installers

Download OpenCV as a self-extracting ZIP file from <http://opencv.org/downloads.html>. Choose the latest version, which should contain both 32-bit and 64-bit binaries. Double-click on the self-extracting ZIP file and, when prompted, enter any destination folder, which we will refer to as <unzip_destination>. A subfolder named <opencv_unzip_destination>\opencv is created.

Copy <opencv_unzip_destination>\opencv\build\python\2.7\x86\cv2.pyd (32-bit) or <opencv_unzip_destination>\opencv\build\python\2.7\x64\cv2.pyd (64-bit) to C:\Python2.7\Lib\site-packages (assuming Python 2.7 is installed to the default location). Now, Python 2.7 can find OpenCV.

You might want to look at the code samples in <unzip_destination>/opencv/samples.

At this point, we have everything we need to develop OpenCV applications for Windows. To also develop the same for Android, we need to set up TADP as described in the section *Tegra Android Development Pack*, later in this chapter.

OpenCV on Windows with CMake and compilers

OpenCV uses a set of build tools called CMake, which we must install. Optionally, we can install several third-party libraries in order to enable extra features in OpenCV. These libraries include OpenNI (for depth camera support), SensorKinect (to add Kinect support to OpenNI), and TBB (for Intel multiprocessing). After installing third-party libraries, we will configure and build OpenCV. Last, we will ensure that our C++ and Python environments can find our build of OpenCV.



The binary installers for OpenCV do provide TBB support but do not provide OpenNI or SensorKinect support. Thus, for depth camera support on Windows, it is necessary to compile OpenCV from source. Although we do not use depth cameras in this book, we have used them in *OpenCV Computer Vision with Python* and you might want to use them in your future projects.

Here are the detailed steps to build OpenCV on Windows:

1. Download and install the latest stable version of CMake from <http://www.cmake.org/cmake/resources/software.html>. Even if we are using 64-bit libraries and compilers, 32-bit CMake is compatible. When the installer asks about modifying PATH, select either **Add CMake to the system PATH for all users** or **Add CMake to the system PATH for current user**.
2. Optionally, download and install the development version of OpenNI 1.5.4.0 (not any other version) from <http://www.nummist.com/opencv/openni-win32-1.5.4.0-dev.zip> (32 bit) or <http://www.nummist.com/opencv/openni-win64-1.5.4.0-dev.zip> (64 bit). Other versions besides OpenNI's 1.5.4.0 development version are not recommended. At least some of them do not work with OpenCV, in my experience.
3. Optionally, download and install SensorKinect 0.93 (not any other version) from <https://github.com/avin2/SensorKinect/blob/unstable/Bin/SensorKinect093-Bin-Win32-v5.1.2.1.msi?raw=true> (32-bit) or <https://github.com/avin2/SensorKinect/blob/unstable/Bin/SensorKinect093-Bin-Win64-v5.1.2.1.msi?raw=true> (64-bit). Other versions besides SensorKinect 0.93 are not recommended. In my experience, a few of them do not work with OpenCV.
4. Download OpenCV as a self-extracting ZIP file from <http://opencv.org/downloads.html>. Choose the latest version, which should contain both 32-bit and 64-bit binaries. Double-click the self-extracting ZIP file and, when prompted, enter any destination folder, which we will refer to as `<opencv_unzip_destination>`. A subfolder named `<opencv_unzip_destination>\opencv` is created.
5. Download the latest stable version of TBB from <https://www.threadingbuildingblocks.org/download>. It includes both 32-bit and 64-bit binaries. Unzip it to any destination, which we will refer to as `<tbb_unzip_destination>`.
6. Open Command Prompt. Create a folder to store our build:


```
> mkdir <build_folder>
```

Change the directory to the newly created build folder:

```
> cd <build_folder>
```
7. Having set up our dependencies, we can now configure OpenCV's build system. To understand all the configuration options, we could read the code in `<opencv_unzip_destination>\opencv\sources\CMakeLists.txt`. However, as an example, we will just use the options for a release build that includes Python bindings, depth camera support via OpenNI and SensorKinect, and multiprocessing via TBB.

To create a 32-bit project for Visual Studio 2010, run the following command (but replace the angle brackets and their contents with the actual paths):

```
> cmake -D CMAKE_BUILD_TYPE=RELEASE -D WITH_OPENNI=ON -D OPENNI_LIB_
DIR=<openni_install_destination>\Lib" -D OPENNI_INCLUDE_DIR=<openni_
install_destination>\Include" -D OPENNI_PRIME_SENSOR_MODULE_BIN_
DIR=<sensorkinect_install_destination>\Bin" -D WITH_TBB=ON -D TBB_LIB_
DIR=<tbb_unzip_destination>\lib\ia32\vc10" -D TBB_INCLUDE_DIR=<tbb_
unzip_destination>\include" -G "Visual Studio 10" "<opencv_unzip_
destination>\opencv\sources"
```

Alternatively, to create a 64-bit project for Visual Studio 2010, run the following command (but replace the angle brackets and their contents with the actual paths):

```
> cmake -D CMAKE_BUILD_TYPE=RELEASE -D WITH_OPENNI=ON -D OPENNI_LIB_
DIR=<openni_install_destination>\Lib" -D OPENNI_INCLUDE_DIR=<openni_
install_destination>\Include" -D OPENNI_PRIME_SENSOR_MODULE_BIN_
DIR=<sensorkinect_install_destination>\Bin" -D WITH_TBB=ON -D TBB_LIB_
DIR=<tbb_unzip_destination>\lib\intel64\vc10" -D TBB_INCLUDE_DIR=<tbb_
unzip_destination>\include" -G "Visual Studio 10 Win64" "<opencv_unzip_
destination>\opencv\sources"
```

If OpenNI is not installed, omit `-D WITH_OPENNI=ON -D OPENNI_LIB_DIR=<openni_install_destination>\Lib -D OPENNI_INCLUDE_DIR=<openni_install_destination>\Include -D OPENNI_PRIME_SENSOR_MODULE_BIN_DIR=<sensorkinect_install_destination>\Bin`. (In this case, depth cameras will not be supported.)

If OpenNI is installed but SensorKinect is not, omit `-D OPENNI_PRIME_SENSOR_MODULE_BIN_DIR=<sensorkinect_install_destination>\Bin`. (In this case, Kinect will not be supported.)

If TBB is not installed, omit `-D WITH_TBB=ON -D TBB_LIB_DIR=<tbb_unzip_destination>\lib\ia32\vc10 -D TBB_INCLUDE_DIR=<tbb_unzip_destination>\include` (32-bit) or `-D WITH_TBB=ON -D TBB_LIB_DIR=<tbb_unzip_destination>\lib\intel64\vc10 -D TBB_INCLUDE_DIR=<tbb_unzip_destination>\include` (64-bit). (In this case, Intel multiprocessing will not be supported.)

CMake will produce a report on the dependencies that it did or did not find. OpenCV has many optional dependencies, so do not panic (yet) about missing dependencies. However, if the build does not finish successfully, try installing missing dependencies (many are available as prebuilt binaries). Then, repeat this step.

1. Now that our build system is configured, we can compile OpenCV. Open `<build_folder>\OpenCV.sln` in Visual Studio. Select **Release** configuration and build the project (you might get errors if you select another build configuration besides **Release**.)

2. Copy `<build_folder>\lib\RELEASE\cv2.pyd` to `C:\Python2.7\Lib\site-packages` (assuming that Python 2.7 is installed in the default location). Now, the Python installation can find part of OpenCV.
3. Finally, we need to make sure that Python and other processes can find the rest of OpenCV and its dependencies. Edit the system's `Path` variable and append `<build_folder>\bin\RELEASE`. If we are using TBB, also append `<tbb_unzip_destination>\lib\ia32\vc10 (32-bit)` or `<tbb_unzip_destination>\lib\intel64\vc10 (64-bit)`. Log out and log back in (or reboot).

You might want to look at the code samples in `<unzip_destination>/opencv/samples`.

At this point, we have everything we need to develop OpenCV applications for Windows. To also develop the same for Android, we need to set up TADP as described in the section *Tegra Android Development Pack*, covered later in this chapter.

Mac

Let's begin by setting up Xcode and the Xcode Command Line Tools, which give us a complete C++ development environment:

1. Download and install Xcode from the Mac App Store or <http://connect.apple.com/>. If the installer provides an option to install **Command Line Tools**, select it.
2. Open Xcode. If a license agreement is presented, accept it.
3. If the Xcode Command Line Tools were not already installed, we must install them now. Go to **Xcode | Preferences | Downloads** and click on the **Install** button next to **Command Line Tools**. Wait for the installation to finish. Then, quit Xcode. Alternatively, if you do not find an option to install the Command Line Tools from inside Xcode, open Terminal and run the following command:

```
$ xcode-select install
```

Next, we need a Python 2.7 development environment. Recent versions of Mac come with Python 2.7 preinstalled. However, the preinstalled Python is customized by Apple for the system's internal needs. Normally, we should not install any libraries atop Apple's Python. If we do, our libraries might break during system updates or worse, might conflict with preinstalled libraries that the system requires. Instead, we should install standard Python 2.7 and then install our libraries atop it.

For Mac, there are several possible approaches to obtain standard Python 2.7 and Python-compatible libraries such as OpenCV. All approaches ultimately require OpenCV to be compiled from source using Xcode Command Line Tools. However, depending on the approach, this task is automated for us by third-party tools in various ways. We will look at approaches using MacPorts or Homebrew. These two tools are package managers, which help us resolve dependencies and separate our development libraries from the system libraries.



I recommend MacPorts. Compared to Homebrew, MacPorts offers more patches and configuration options for OpenCV. Also, I maintain a MacPorts repository to ensure that you can get continue to get an OpenCV build that is compatible with all of my books. Particularly, my version includes support for depth cameras such as Kinect, which were used in *OpenCV Computer Vision with Python*.

Normally, MacPorts and Homebrew should not be installed on the same machine.

Regardless of the approach to set up our Python environment, we can put PyInstaller in any convenient location, since it is treated as a set of tools rather than a library. Let's download the latest release version from <http://www.pyinstaller.org/> and unzip it to `~/PyInstaller` or another location of your choice.



Our installation methods for Mac do not give us the OpenCV sample projects. To get these, download the latest source code archive from <http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/> and unzip it to any location. Find the samples in `<opencv_unzip_destination>/samples`.

Now, depending on your preference, let's proceed to either the *Mac with MacPorts* section or the *Mac with Homebrew* section.

Mac with MacPorts

MacPorts provides Terminal commands that automate the process of downloading, compiling, and installing various pieces of **open source software (OSS)**. MacPorts also installs dependencies as needed. For each piece of software, the dependencies and build recipe are defined in a configuration file called a **Portfile**. A MacPorts **repository** is a collection of Portfiles.

Starting from a system where Xcode and its Command Line Tools are already set up, the following steps will give us an OpenCV installation via MacPorts:

1. Download and install MacPorts from <http://www.macports.org/install.php>.
2. If we want an OpenCV build that is fully compatible with all of my books, we need to inform MacPorts where to download some custom Portfiles that I have written. To do so, edit `/opt/local/etc/macports/sources.conf` (assuming MacPorts is installed in the default location). Just above the line `rsync://rsync.macports.org/ release/ports/ [default]`, add the following line:
`http://nummist.com/opencv/ports.tar.gz`

Downloading the example code



You can download the example code files from your account at <http://www.packtpub.com> for all the Packt Publishing books you have purchased. If you purchased this book elsewhere, you can visit <http://www.packtpub.com/support> and register to have the files e-mailed directly to you. The latest and updated example code for this book is also available from the author's website at <http://nummist.com/opencv/>.

Save the file. Now, MacPorts knows to search for Portfiles in my online repository first and then the default online repository.

3. Open Terminal and run the following command to update MacPorts:

```
$ sudo port selfupdate
```

When prompted, enter your password.

4. Now (if you are using my repository), run the following command to install OpenCV with Python 2.7 bindings, plus extras such as support for Intel TBB multiprocessing and support for depth cameras including Kinect:

```
$ sudo port install opencv +python27 +tbb +openni_sensorkinect
```

Alternatively (with or without my repository), run the following command to install OpenCV with Python 2.7 bindings, plus extras such as support for Intel TBB multiprocessing and support for depth cameras excluding Kinect:

```
$ sudo port install opencv +python27 +tbb +openni
```

Dependencies, including Python 2.7, NumPy, OpenNI, and (in the first example) SensorKinect, are automatically installed as well.

By adding `+python27` to the command, we are specifying that we want the OpenCV variant (build configuration) with Python 2.7 bindings. Similarly, `+tbb` specifies the variant with support for Intel TBB multiprocessing, which can greatly improve the performance on compatible hardware. The `+openni_sensorkinect` tag specifies the variant with the broadest possible support for depth cameras via OpenNI and SensorKinect. You can omit `+openni_sensorkinect` if you do not intend to use depth cameras or you can replace it with `+openni` if you do intend to use OpenNI-compatible depth cameras but just not Kinect. To see the full list of available variants before installing, we can enter:

```
$ port variants opencv
```

Depending on our customization needs, we can add other variants to the install command.

5. Run the following commands to install SciPy, Requests, and wxPython:

```
$ sudo port install py27-scipy
$ sudo port install py27-requests
$ sudo port install py27-wxpython-3.0
```

6. The Python installation's executable is named `python2.7`. If we want to link the default python executable to `python2.7`, let's also run:

```
$ sudo port install python_select
$ sudo port select python python27
```

Now we have everything we need to develop OpenCV applications for Mac. To also develop the same for Android, we need to set up TADP as described in the section *Tegra Android Development Pack*, covered later in this chapter.

Mac with Homebrew

Like MacPorts, Homebrew is a package manager that provides Terminal commands to automate the process of downloading, compiling, and installing various pieces of open source software.

Starting from a system where Xcode and its Command Line Tools are already set up, the following steps will give us an OpenCV installation via Homebrew:

1. Open Terminal and run the following command to install Homebrew:

```
$ ruby -e "$(curl -fsSLraw.github.com/mxcl/homebrew/go)"
```

2. Unlike MacPorts, Homebrew does not automatically put its executables in `PATH`. To do so, create or edit the file `~/.profile` and add this line at the top:
`export PATH=/usr/local/bin:/usr/local/sbin:$PATH`