Community Experience Distilled

Learning OpenCV 3 Computer Vision with Python

Second Edition

Unleash the power of computer vision with Python using OpenCV

Joe Minichino
Joseph Howse

[PACKT] open source*
PUBLISHING
community experience distilled

Learning OpenCV 3 Computer Vision with Python

Second Edition

Unleash the power of computer vision with Python using OpenCV

Joe Minichino

Joseph Howse

[PACKT] open source
PUBLISHING
community experience distilled

BIRMINGHAM - MUMBAI

Learning OpenCV 3 Computer Vision with Python

Second Edition

First published: September 2015

Production reference: 1240915

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

ISBN 978-1-78528-384-0

`www.packtpub.com`

Credits

Authors
Joe Minichino
Joseph Howse

Reviewers
Nandan Banerjee
Tian Cao
Brandon Castellano
Haojian Jin
Adrian Rosebrock

Commissioning Editor
Akram Hussain

Acquisition Editors
Vivek Anantharaman
Prachi Bisht

Content Development Editor
Ritika Singh

Technical Editors
Novina Kewalramani
Shivani Kiran Mistry

Copy Editor
Sonia Cheema

Project Coordinator
Milton Dsouza

Proofreader
Safis Editing

Indexer
Monica Ajmera Mehta

Graphics
Disha Haria

Production Coordinator
Arvindkumar Gupta

Cover Work
Arvindkumar Gupta

About the Authors

Joe Minichino is a computer vision engineer for Hoolux Medical by day and a developer of the NoSQL database LokiJS by night. On weekends, he is a heavy metal singer/songwriter. He is a passionate programmer who is immensely curious about programming languages and technologies and constantly experiments with them. At Hoolux, Joe leads the development of an Android computer vision-based advertising platform for the medical industry.

Born and raised in Varese, Lombardy, Italy, and coming from a humanistic background in philosophy (at Milan's Università Statale), Joe has spent his last 11 years living in Cork, Ireland, which is where he became a computer science graduate at the Cork Institute of Technology.

I am immensely grateful to my partner, Rowena, for always encouraging me, and also my two little daughters for inspiring me. A big thank you to the collaborators and editors of this book, especially Joe Howse, Adrian Roesbrock, Brandon Castellano, the OpenCV community, and the people at Packt Publishing.

Joseph Howse lives in Canada. During the winters, he grows his beard, while his four cats grow their thick coats of fur. He loves combing his cats every day and sometimes, his cats also pull his beard.

He has been writing for Packt Publishing since 2012. His books include *OpenCV for Secret Agents*, *OpenCV Blueprints*, *Android Application Programming with OpenCV 3*, *OpenCV Computer Vision with Python*, and *Python Game Programming by Example*.

When he is not writing books or grooming his cats, he provides consulting, training, and software development services through his company, Nummist Media (`http://nummist.com`).

About the Reviewers

Nandan Banerjee has a bachelor's degree in computer science and a master's in robotics engineering. He started working with Samsung Electronics right after graduation. He worked for a year at its R&D centre in Bangalore. He also worked in the WPI-CMU team on the Boston Dynamics' robot, Atlas, for the DARPA Robotics Challenge. He is currently working as a robotics software engineer in the technology organization at iRobot Corporation. He is an embedded systems and robotics enthusiast with an inclination toward computer vision and motion planning. He has experience in various languages, including C, C++, Python, Java, and Delphi. He also has a substantial experience in working with ROS, OpenRAVE, OpenCV, PCL, OpenGL, CUDA and the Android SDK.

I would like to thank the author and publisher for coming out with this wonderful book.

Tian Cao is pursuing his PhD in computer science at the University of North Carolina in Chapel Hill, USA, and working on projects related to image analysis, computer vision, and machine learning.

I dedicate this work to my parents and girlfriend.

Brandon Castellano is a student from Canada pursuing an MESc in electrical engineering at the University of Western Ontario, City of London, Canada. He received his BESc in the same subject in 2012. The focus of his research is in parallel processing and GPGPU/FPGA optimization for real-time implementations of image processing algorithms. Brandon also works for Eagle Vision Systems Inc., focusing on the use of real-time image processing for robotics applications.

While he has been using OpenCV and C++ for more than 5 years, he has also been advocating the use of Python frequently in his research, most notably, for its rapid speed of development, allowing low-level interfacing with complex systems. This is evident in his open source projects hosted on GitHub, for example, PySceneDetect, which is mostly written in Python. In addition to image/video processing, he has also worked on implementations of three-dimensional displays as well as the software tools to support the development of such displays.

In addition to posting technical articles and tutorials on his website (`http://www.bcastell.com`), he participates in a variety of both open and closed source projects and contributes to GitHub under the username Breakthrough (`http://www.github.com/Breakthrough`). He is an active member of the Super User and Stack Overflow communities (under the name Breakthrough), and can be contacted directly via his website.

I would like to thank all my friends and family for their patience during the past few years (especially my parents, Peter and Lori, and my brother, Mitchell). I could not have accomplished everything without their continued love and support. I can't ever thank everyone enough.

I would also like to extend a special thanks to all of the developers that contribute to open source software libraries, specifically OpenCV, which help bring the development of cutting-edge software technology closer to all the software developers around the world, free of cost. I would also like to thank those people who help write documentation, submit bug reports, and write tutorials/books (especially the author of this book!). Their contributions are vital to the success of any open source project, especially one that is as extensive and complex as OpenCV.

Haojian Jin is a software engineer/researcher at Yahoo! Labs, Sunnyvale, CA. He looks primarily at building new systems of what's possible on commodity mobile devices (or with minimum hardware changes). To create things that don't exist today, he spends large chunks of his time playing with signal processing, computer vision, machine learning, and natural language processing and using them in interesting ways. You can find more about him at `http://shift-3.com/`

Adrian Rosebrock is an author and blogger at `http://www.pyimagesearch.com/`. He holds a PhD in computer science from the University of Maryland, Baltimore County, USA, with a focus on computer vision and machine learning.

He has consulted for the National Cancer Institute to develop methods that automatically predict breast cancer risk factors using breast histology images. He has also authored a book, *Practical Python and OpenCV* (`http://pyimg.co/x7ed5`), on the utilization of Python and OpenCV to build real-world computer vision applications.

Table of Contents

Preface

OpenCV 3 is a state-of-the-art computer vision library that is used for a variety of image and video processing operations. Some of the more spectacular and futuristic features, such as face recognition or object tracking, are easily achievable with OpenCV 3. Learning the basic concepts behind computer vision algorithms, models, and OpenCV's API will enable the development of all sorts of real-world applications, including security and surveillance tools.

Starting with basic image processing operations, this book will take you through a journey that explores advanced computer vision concepts. Computer vision is a rapidly evolving science whose applications in the real world are exploding, so this book will appeal to computer vision novices as well as experts of the subject who want to learn about the brand new OpenCV 3.0.0.

What this book covers

Chapter 1, *Setting Up OpenCV*, explains how to set up OpenCV 3 with Python on different platforms. It will also troubleshoot common problems.

Chapter 2, *Handling Files, Cameras, and GUIs*, introduces OpenCV's I/O functionalities. It will also discuss the concept of a project and the beginnings of an object-oriented design for this project.

Chapter 3, *Processing Images with OpenCV 3*, presents some techniques required to alter images, such as detecting skin tone in an image, sharpening an image, marking contours of subjects, and detecting crosswalks using a line segment detector.

Chapter 4, *Depth Estimation and Segmentation*, shows you how to use data from a depth camera to identify foreground and background regions, such that we can limit an effect to only the foreground or background.

Chapter 5, Detecting and Recognizing Faces, introduces some of OpenCV's face detection functionalities, along with the data files that define particular types of trackable objects.

Chapter 6, Retrieving Images and Searching Using Image Descriptors, shows how to detect the features of an image with the help of OpenCV and make use of them to match and search for images.

Chapter 7, Detecting and Recognizing Objects, introduces the concept of detecting and recognizing objects, which is one of the most common challenges in computer vision.

Chapter 8, Tracking Objects, explores the vast topic of object tracking, which is the process of locating a moving object in a movie or video feed with the help of a camera.

Chapter 9, Neural Networks with OpenCV – an Introduction, introduces you to Artificial Neural Networks in OpenCV and illustrates their usage in a real-life application.

What you need for this book

You simply need a relatively recent computer, as the first chapter will guide you through the installation of all the necessary software. A webcam is highly recommended, but not necessary.

Who this book is for

This book is aimed at programmers with working knowledge of Python as well as people who want to explore the topic of computer vision using the OpenCV library. No previous experience of computer vision or OpenCV is required. Programming experience is recommended.

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: "We can include other contexts through the use of the `include` directive."

A block of code is set as follows:

```
import cv2
import numpy as np

img = cv2.imread('images/chess_board.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
gray = np.float32(gray)
dst = cv2.cornerHarris(gray, 2, 23, 0.04)
```

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

```
img = cv2.imread('images/chess_board.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
gray = np.float32(gray)
dst = cv2.cornerHarris(gray, 2, 23, 0.04)
```

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

```
mkdir build && cd build

cmake D CMAKE_BUILD_TYPE=Release -DOPENCV_EXTRA_MODULES_PATH=<opencv_
contrib>/modules  D CMAKE_INSTALL_PREFIX=/usr/local ..

make
```

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: " On Windows Vista / Windows 7 / Windows 8, click on the **Start** menu."

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.

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 **Errata Submission Form** 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.

1
Setting Up OpenCV

You picked up this book so you may already have an idea of what OpenCV is. Maybe, you heard of Sci-Fi-sounding features, such as face detection, and got intrigued. If this is the case, you've made the perfect choice. **OpenCV** stands for **Open Source Computer Vision**. It is a free computer vision library that allows you to manipulate images and videos to accomplish a variety of tasks from displaying the feed of a webcam to potentially teaching a robot to recognize real-life objects.

In this book, you will learn to leverage the immense potential of OpenCV with the Python programming language. Python is an elegant language with a relatively shallow learning curve and very powerful features. This chapter is a quick guide to setting up Python 2.7, OpenCV, and other related libraries. After setup, we also look at OpenCV's Python sample scripts and documentation.

[If you wish to skip the installation process and jump right into action, you can download the **virtual machine** (**VM**) I've made available at `http://techfort.github.io/pycv/`.

This file is compatible with VirtualBox, a free-to-use virtualization application that lets you build and run VMs. The VM I've built is based on Ubuntu Linux 14.04 and has all the necessary software installed so that you can start coding right away.

This VM requires at least 2 GB of RAM to run smoothly, so make sure that you allocate at least 2 (but, ideally, more than 4) GB of RAM to the VM, which means that your host machine will need at least 6 GB of RAM to sustain it.]

The following related libraries are covered in this chapter:

- **NumPy**: This library is a dependency of OpenCV's Python bindings. It provides numeric computing functionality, including efficient arrays.
- **SciPy**: This library is a scientific computing library that is closely related to NumPy. It is not required by OpenCV, but it is useful for manipulating data in OpenCV images.
- **OpenNI**: This library is an optional dependency of OpenCV. It adds the support for certain depth cameras, such as Asus XtionPRO.
- **SensorKinect**: This library is an OpenNI plugin and optional dependency of OpenCV. It adds support for the Microsoft Kinect depth camera.

For this book's purposes, OpenNI and SensorKinect can be considered optional. They are used throughout *Chapter 4*, *Depth Estimation and Segmentation*, but are not used in the other chapters or appendices.

This book focuses on OpenCV 3, the new major release of the OpenCV library. All additional information about OpenCV is available at `http://opencv.org`, and its documentation is available at `http://docs.opencv.org/master`.

Choosing and using the right setup tools

We are free to choose various setup tools, depending on our operating system and how much configuration we want to do. Let's take an overview of the tools for Windows, Mac, Ubuntu, and other Unix-like systems.

Installation on Windows

Windows does not come with Python preinstalled. However, installation wizards are available for precompiled Python, NumPy, SciPy, and OpenCV. Alternatively, we can build from a source. OpenCV's build system uses CMake for configuration and either Visual Studio or MinGW for compilation.

If we want support for depth cameras, including Kinect, we should first install OpenNI and SensorKinect, which are available as precompiled binaries with installation wizards. Then, we must build OpenCV from a source.

The precompiled version of OpenCV does not offer support for depth cameras.

On Windows, OpenCV 2 offers better support for 32-bit Python than 64-bit Python; however, with the majority of computers sold today being 64-bit systems, our instructions will refer to 64-bit. All installers have 32-bit versions available from the same site as the 64-bit.

Some of the following steps refer to editing the system's `PATH` variable. This task can be done in the **Environment Variables** window of **Control Panel**.

1. On Windows Vista / Windows 7 / Windows 8, click on the **Start** menu and launch **Control Panel**. Now, navigate to **System** and **Security** | **System** | **Advanced system settings**. Click on the **Environment Variables…** button.
2. On Windows XP, click on the **Start** menu and navigate to **Control Panel** | **System**. Select the **Advanced** tab. Click on the **Environment Variables…** button.
3. Now, under **System variables**, select **Path** and click on the **Edit…** button.
4. Make changes as directed.
5. To apply the changes, click on all the **OK** buttons (until we are back in the main window of **Control Panel**).
6. Then, log out and log back in (alternatively, reboot).

Using binary installers (no support for depth cameras)

You can choose to install Python and its related libraries separately if you prefer; however, there are Python distributions that come with installers that will set up the entire SciPy stack (which includes Python and NumPy), which make it very trivial to set up the development environment.

One such distribution is Anaconda Python (downloadable at `http://09c8d0b2229f813c1b93c95ac804525aac4b6dba79b00b39d1d3.r79.cf1.rackcdn.com/Anaconda-2.1.0Windows-x86_64.exe`). Once the installer is downloaded, run it and remember to add the path to the Anaconda installation to your `PATH` variable following the preceding procedure.

Here are the steps to set up Python7, NumPy, SciPy, and OpenCV:

1. Download and install the 32-bit Python 2.7.9 from `https://www.python.org/ftp/python/2.7.9/python-2.7.9.amd64.msi`.
2. Download and install NumPy 1.6.2 from `http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpyhttp://sourceforge.net/projects/numpy/files/NumPy/1.6.2/numpy-1.6.2-win32-superpack-python2.7.exe/download` (note that installing NumPy on Windows 64-bit is a bit tricky due to the lack of a 64-bit Fortran compiler on Windows, which NumPy depends on. The binary at the preceding link is unofficial).
3. Download and install SciPy 11.0 from `http://www.lfd.uci.edu/~gohlke/pythonlibs/#scipyhttp://sourceforge.net/projects/scipy/files/scipy/0.11.0/scipy-0.11.0win32-superpack-python2.7.exe/download` (this is the same as NumPy and these are community installers).
4. Download the self-extracting ZIP of OpenCV 3.0.0 from `https://github.com/Itseez/opencv`. Run this ZIP, and when prompted, enter a destination folder, which we will refer to as `<unzip_destination>`. A subfolder, `<unzip_destination>\opencv`, is created.
5. Copy `<unzip_destination>\opencv\build\python\2.7\cv2.pyd` to `C:\Python2.7\Lib\site-packages` (assuming that we had installed Python 2.7 to the default location). If you installed Python 2.7 with Anaconda, use the Anaconda installation folder instead of the default Python installation. Now, the new Python installation can find OpenCV.
6. A final step is necessary if we want Python scripts to run using the new Python installation by default. Edit the system's `PATH` variable and append `;C:\Python2.7` (assuming that we had installed Python 2.7 to the default location) or your Anaconda installation folder. Remove any previous Python paths, such as `;C:\Python2.6`. Log out and log back in (alternatively, reboot).

Using CMake and compilers

Windows does not come with any compilers or CMake. We need to install them. If we want support for depth cameras, including Kinect, we also need to install OpenNI and SensorKinect.

Let's assume that we have already installed 32-bit Python 2.7, NumPy, and SciPy either from binaries (as described previously) or from a source. Now, we can proceed with installing compilers and CMake, optionally installing OpenNI and SensorKinect, and then building OpenCV from the source:

1. Download and install CMake 3.1.2 from `http://www.cmake.org/files/v3.1/cmake-3.1.2-win32-x86.exe`. When running the installer, select either **Add CMake to the system PATH for all users** or **Add CMake to the system PATH for current user**. Don't worry about the fact that a 64-bit version of CMake is not available CMake is only a configuration tool and does not perform any compilations itself. Instead, on Windows, it creates project files that can be opened with Visual Studio.
2. Download and install Microsoft Visual Studio 2013 (the Desktop edition if you are working on Windows 7) from `https://www.visualstudio.com/products/free-developer-offers-vs.aspx?slcid=0x409&type=web or MinGW`.

   Note that you will need to sign in with your Microsoft account and if you don't have one, you can create one on the spot. Install the software and reboot after installation is complete.

   For MinGW, get the installer from `http://sourceforge.net/projects/mingw/files/Installer/mingw-get-setup.exe/download` and `http://sourceforge.net/projects/mingw/files/OldFiles/mingw-get-inst/mingw-get-inst-20120426/mingw-get-inst-20120426.exe/download`. When running the installer, make sure that the destination path does not contain spaces and that the optional C++ compiler is included. Edit the system's `PATH` variable and append `;C:\MinGW\bin` (assuming that MinGW is installed to the default location). Reboot the system.

3. Optionally, download and install OpenNI 1.5.4.0 from the links provided in the GitHub homepage of OpenNI at `https://github.com/OpenNI/OpenNI`.
4. You can download and install SensorKinect 0.93 from `https://github.com/avin2/SensorKinect/blob/unstable/Bin/SensorKinect093-Bin-Win32-v5.1.2.1.msi?raw=true` (32-bit). Alternatively, for 64-bit Python, download the setup from `https://github.com/avin2/SensorKinect/blob/unstable/Bin/SensorKinect093-Bin-Win64-v5.1.2.1.msi?raw=true` (64-bit). Note that this repository has been inactive for more than three years.
5. Download the self-extracting ZIP of OpenCV 3.0.0 from `https://github.com/Itseez/opencv`. Run the self-extracting ZIP, and when prompted, enter any destination folder, which we will refer to as `<unzip_destination>`. A subfolder, `<unzip_destination>\opencv`, is then created.

6. Open Command Prompt and make another folder where our build will go using this command:

   ```
   > mkdir<build_folder>
   ```

   Change the directory of the `build` folder:

   ```
   > cd <build_folder>
   ```

7. Now, we are ready to configure our build. To understand all the options, we can read the code in `<unzip_destination>\opencv\CMakeLists.txt`. However, for this book's purposes, we only need to use the options that will give us a release build with Python bindings, and optionally, depth camera support via OpenNI and SensorKinect.
8. Open CMake (`cmake-gui`) and specify the location of the source code of OpenCV and the folder where you would like to build the library. Click on **Configure**. Select the project to be generated. In this case, select Visual Studio 12 (which corresponds to Visual Studio 2013). After CMake has finished configuring the project, it will output a list of build options. If you see a red background, it means that your project may need to be reconfigured: CMake might report that it has failed to find some dependencies. Many of OpenCV's dependencies are optional, so do not be too concerned yet.

If the build fails to complete or you run into problems later, try installing missing dependencies (often available as prebuilt binaries), and then rebuild OpenCV from this step.

You have the option of selecting/deselecting build options (according to the libraries you have installed on your machine) and click on **Configure** again, until you get a clear background (white).

9. At the end of this process, you can click on **Generate**, which will create an `OpenCV.sln` file in the folder you've chosen for the build. You can then navigate to `<build_folder>/OpenCV.sln` and open the file with Visual Studio 2013, and proceed with building the project, `ALL_BUILD`. You will need to build both the **Debug** and **Release** versions of OpenCV, so go ahead and build the library in the **Debug** mode, then select **Release** and rebuild it (*F7* is the key to launch the build).
10. At this stage, you will have a `bin` folder in the OpenCV build directory, which will contain all the generated `.dll` files that will allow you to include OpenCV in your projects.

    Alternatively, for MinGW, run the following command:

    ```
    > cmake -D:CMAKE_BUILD_TYPE=RELEASE -D:WITH_OPENNI=ON -G
    "MinGWMakefiles" <unzip_destination>\opencv
    ```