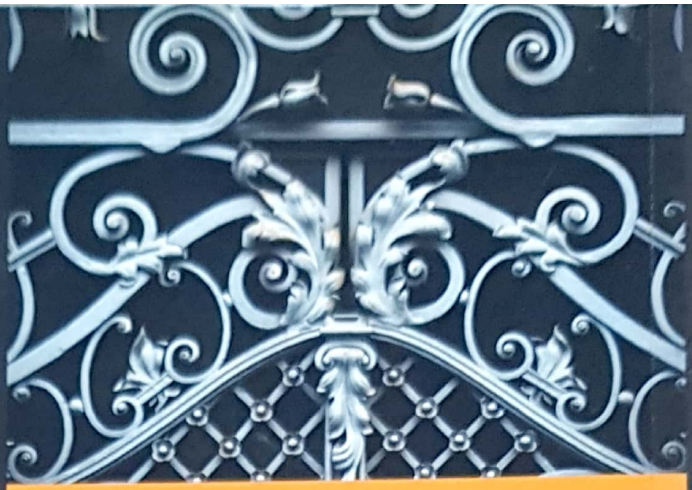
Learning Robotics Using Python



Learning about robotics will become an increasingly essential skill as it becomes a ubiquitous part of life. Even though robotics is a complex subject, several other tools along with Python can help you design a project to create an easy-to-use interface.

Learning Robotics Using Python is an essential guide for creating an autonomous mobile robot using popular robotic software frameworks such as ROS using Python. It also discusses various robot software frameworks and how to go about coding the robot using Python and its framework. It concludes with creating a GUI-based application to control the robot using buttons and slides.

By the end of this tutorial, you'll have a clear idea of how to integrate and assemble all things into a robot and how to bundle the software package.

Who this book is written for

If you are an engineer, a researcher, or a hobbyist, and you are interested in robotics and want to build your own robot, this book is for you. Readers are assumed to be new to robotics but should have experience with Python.

What you will learn from this book

- Understand the core concepts and terminologies of robotics
- Create 2D and 3D drawings of robots using freeware such as LibreCAD and Blender
- Simulate your robot using ROS and Gazebo
- Build robot hardware from the requirements
- Explore a diverse range of actuators and its interfacing
- Interface various robotic sensors to robots
- Set up and program OpenCV, OpenNI, and PCL to process 2D/3D visual data
- Learn speech processing and synthesis using Python
- Apply artificial intelligence to robots using Python
- Build a robot control GUI using Qt and Python
- Calibration and testing of robot

$ 44.99 US
£ 29.99 UK

Prices do not include local sales tax or VAT where applicable

ISBN 978-1-78328-753-6
54499
9 781783 287536

[PACKT] PUBLISHING open source* community experience distilled

Introduction to OpenCV, OpenNI, and PCL

Let's discuss about the software frameworks and libraries that we are using in our robots. First, we can discuss OpenCV. This is one of the libraries that we are going to use in this robot for object detection and other image processing functionalities.

What is OpenCV?

OpenCV is an open source BSD-licensed computer vision based library that includes hundreds of computer vision algorithms. The library, mainly aimed for real-time computer vision, was developed by Intel Russia research, and is now actively supported by Itseez (`http://itseez.com/`).

OpenCV logo

OpenCV is written mainly in C and C++ and its primary interface is in C++. It also has good interfaces in Python, Java, Matlab/Octave and wrappers in other languages such as C# and Ruby.

In the new version of OpenCV, there is support for CUDA and OpenCL to get GPU acceleration (`http://www.nvidia.com/object/cuda_home_new.html`).

OpenCV will run on most of the OS platforms (such as Windows, Linux, Mac OS X, Android, FreeBSD, OpenBSD, iOS, and Blackberry).

In Ubuntu, OpenCV, and Python, wrappers are already installed when we install the `ros-indigo-desktop-full` package. If this package is not installed, then we can install the OpenCV library, ROS interface, and Python interface of OpenCV using the following command:

```
$ sudo apt-get install ros-indigo-vision-opencv
```

If you want to install only the OpenCV Python wrapper, then use the following command:

```
$ sudo apt-get install python-opencv
```

If you want to try OpenCV in Windows, you can try the following link:

`http://docs.opencv.org/doc/tutorials/introduction/windows_install/windows_install.html`

The following link will guide you through the installation process of OpenCV on Mac OS X:

`http://jjyap.wordpress.com/2014/05/24/installing-opencv-2-4-9-on-mac-osx-with-python-support/`

The main applications of OpenCV are in the f eld of:

- Object detection
- Gesture recognition
- Human-computer interaction
- Mobile robotics
- Motion tracking
- Facial recognition

Now we can see how to install OpenCV in Ubuntu 14.04.2 from source code.

Installation of OpenCV from source code in Ubuntu 14.04.2

We can install OpenCV from source code in Linux based on the following documentation of OpenCV:

`http://docs.opencv.org/doc/tutorials/introduction/linux_install/linux_install.html`

After the installation of OpenCV, we can try some examples using the Python wrappers of OpenCV.

Reading and displaying an image using the Python-OpenCV interface

The first example will load an image in grayscale and display it on screen.

In the following section of code, we will import the `numpy` module for image array manipulation and the `cv2` module is the OpenCV wrapper for Python in which we can access OpenCV Python APIs. NumPy is an extension to the Python programming language, adding support for large multidimensional arrays and matrices along with a large library of high-level mathematical functions to operate on these arrays (`https://pypi.python.org/pypi/numpy`):

```
#!/usr/bin/env python
import numpy as np
import cv2
```

The following function will read the `robot.jpg` image and load this image in grayscale. The first argument of the `cv2.imread()` function is the name of the image and the next argument is a flag that specifies the color type of the loaded image. If the flag is > 0, the image returns a three channel RGB color image, if the flag = 0, the loaded image will be a grayscale image, and if the flag is < 0, it will return the same image as loaded:

```
img = cv2.imread('robot.jpg',0)
```

The following section of code will show the read image using the `imshow()` function. The `cv2.waitKey(0)` function is a keyboard binding function. Its argument is time in milliseconds. If it's 0, it will wait indefinitely for a key stroke:

```
cv2.imshow('image',img)
cv2.waitKey(0)
```

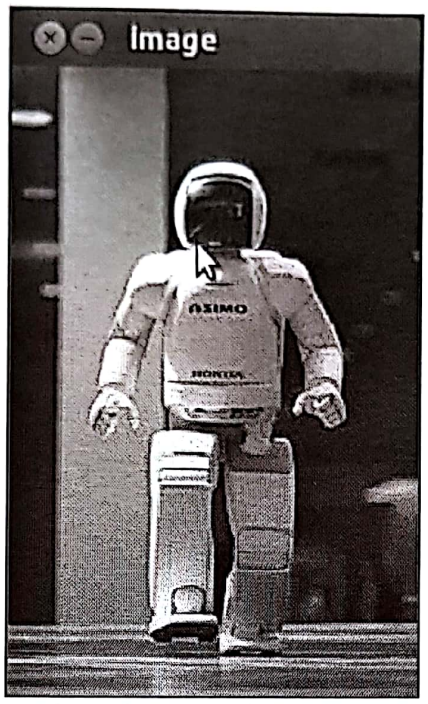
The `cv2.destroyAllWindows()` function simply destroys all the windows we created:

```
cv2.destroyAllWindows()
```

Save the preceding code with a name called `image_read.py` and copy a JPG file with `robot.jpg` as its name. Execute the code using the following command:

```
$python image_read.py
```

The output will load an image in grayscale because we used `0` as the value in the `imread()` function:



The following example will try to open webcam. The program will quit when the user presses any button.

Capturing from web camera

The following code will capture the webcam having device name `/dev/video0` or `/dev/video1`.

We need to import the following modules if we are using OpenCV API's:

```
#!/usr/bin/env python
import numpy as np
import cv2
```

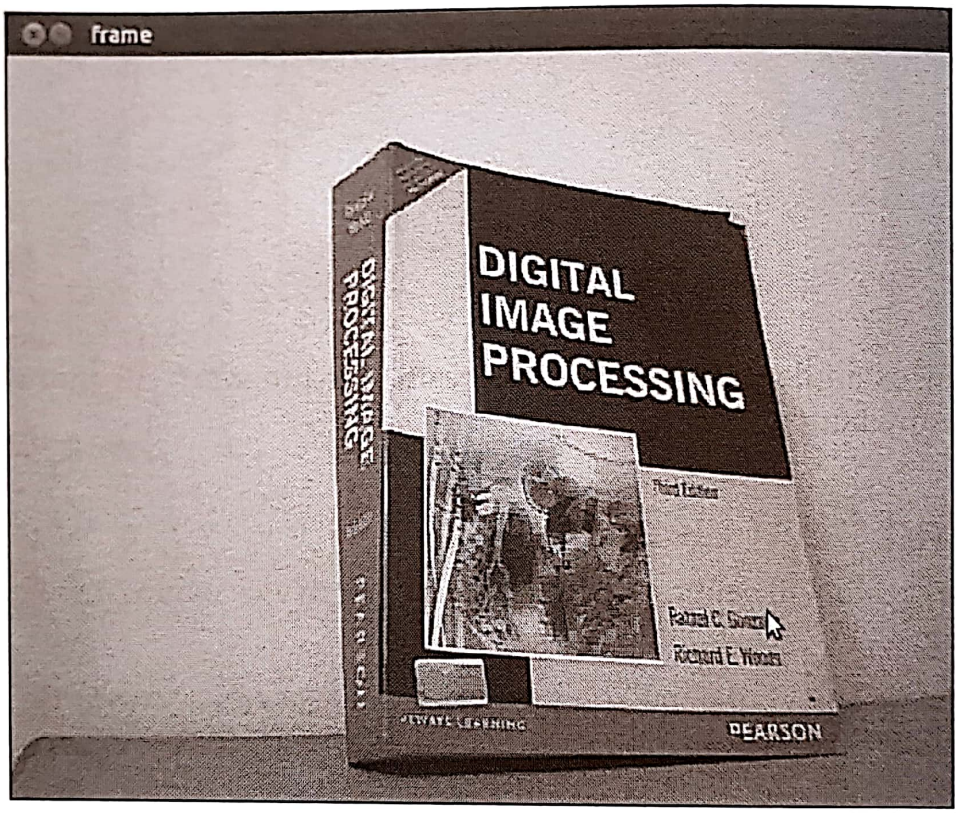
The following function will create a `VideoCapture` object. The `VideoCapture` class is used to capture videos from video f les or cameras. The initialization arguments of the `VideoCapture` class is the index of a camera or a name of a video f le. Device index is just a number to specify the camera. The f rst camera index is 0 having device name `/dev/video0`; that's why we use `0` here:

```
cap = cv2.VideoCapture(0)
```

The following section of code is looped to read image frames from the `VideoCapture` object and shows each frame. It will quit when any key is pressed:

```
while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()
    # Display the resulting frame
    cv2.imshow('frame',frame)
    if cv2.waitKey(10):
        break
```

The following is a screenshot of the program output:



You can explore more OpenCV-Python tutorials from the following link:

`http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_tutorials.html`

In the next section, we will look at OpenNI library and its application.

Installing OpenNI in Ubuntu 14.04.2

We can install the OpenNI library along with ROS packages. ROS is already interfaced with OpenNI, but the complete installation of `ros-indigo-desktop-full` may not install OpenNI packages; we need to install it from the package manager.

The following is the installation command:

```
$ sudo apt-get install ros-indigo-openni-launch
```

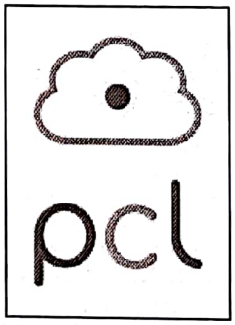
The source code and latest build of OpenNI for Windows, Linux, and MacOS X is available at the following link:

`http://structure.io/openni`

In the next section, we will see how to install PCL.

What is PCL?

PCL is a large scale, open project for 2D/3D image, and Point Cloud processing. The PCL framework contains numerous algorithms included to perform f ltering, feature estimation, surface reconstruction, registration, model f tting, and segmentation. Using these methods, we can process Point Cloud and extract key descriptors to recognize objects in the world based on their geometric appearance and create surfaces from the Point Clouds and visualize them.



PCL logo

PCL is released under the BSD license. It's open source, and free for commercial, or research use. PCL is cross-platform and has been successfully compiled and deployed on Linux, Mac OS X, Windows, and Android/iOS.

You can download PCL at the following link:

`http://pointclouds.org/downloads/`

PCL is already integrated into ROS. The PCL library and its ROS interface will install along with ROS full desktop installation. In the previous chapter, we discussed how to install ROS full desktop installation. PCL is the 3D processing backbone of ROS. Refer to the following link for details on the ROS-PCL package:

`http://wiki.ros.org/pcl`.

Programming Kinect with Python using ROS, OpenCV, and OpenNI

Let's look at how we can interface and work with the Kinect sensor in ROS. ROS is bundled with OpenNI driver, which can fetch RGB and the depth image of Kinect. This package can be used for Microsoft Kinect, PrimeSense Carmine, Asus Xtion Pro, and Pro Live.

This driver mainly publishes raw depth, RGB, and IR image streams. The `openni_launch` package will install packages such as `openni_camera` and `openni_launch`. The `openni_camera` package is the Kinect driver that publishes raw data and sensor information, whereas the `openni_launch` package contains ROS launch f les. It's basically an XML f le that launches multiple nodes at a time and publishes data such as point clouds.

How to launch OpenNI driver

The following command will open the OpenNI device and load all nodelets to convert raw depth/RGB/IR streams to depth images, disparity images, and point clouds. The ROS `nodelet` package is designed to provide a way to run multiple algorithms in the same process with zero copy transport between algorithms.

```
$ roslaunch openni_launch openni.launch
```

You can view the RGB image using a ROS tool called `image_view`

```
$ rosrun image_view image_view image:=/camera/rgb/image_color
```

In the next section, we will see how to interface these images to OpenCV for image processing.