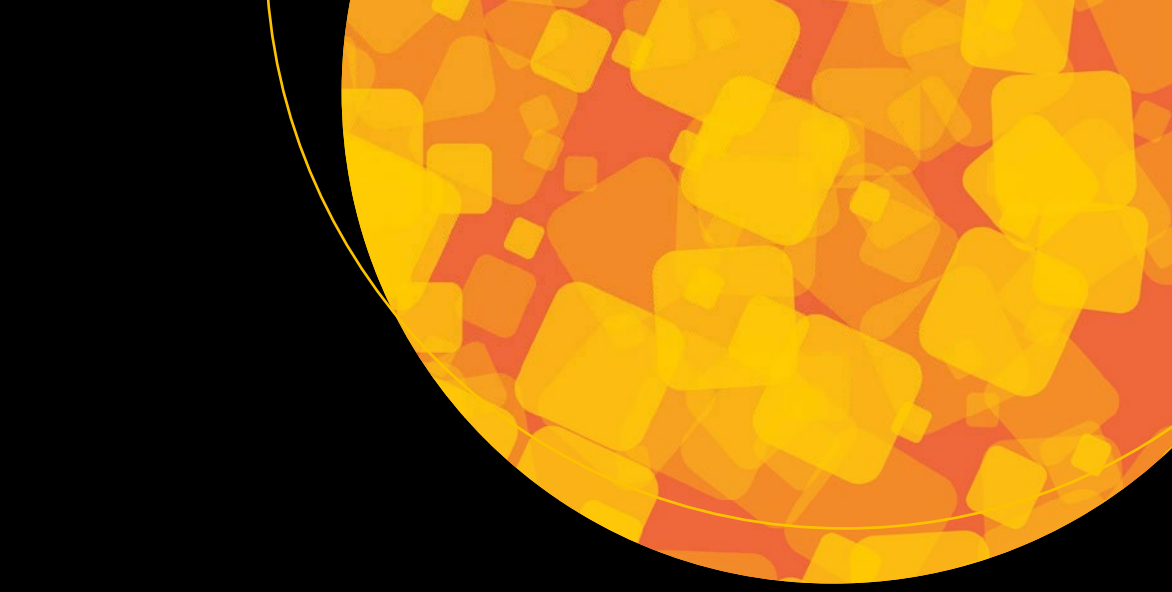

Computer Vision Using Deep Learning

Neural Network Architectures with Python and Keras

Vaibhav Verdhan

Apress®

Computer Vision Using Deep Learning

Neural Network Architectures with Python and Keras

Vaibhav Verdhan

Apress®

Computer Vision Using Deep Learning: Neural Network Architectures with Python and Keras

Vaibhav Verdhan
Limerick, Ireland

ISBN-13 (pbk): 978-1-4842-6615-1 ISBN-13 (electronic): 978-1-4842-6616-8
https://doi.org/10.1007/978-1-4842-6616-8

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Aaron Black
Development Editor: James Markham
Coordinating Editor: Jessica Vakili

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 NY Plazar, New York, NY 10014. 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 booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

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 http://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/978-1-4842-6615-1. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

To Yashi, Pakhi and Rudra

Table of Contents

About the Author

Vaibhav Verdhan is a seasoned data science professional with rich experience spanning across geographies and domains. He is a hands-on technical expert and has led multiple engagements in machine learning and artificial intelligence. He is a leading industry expert, is a regular speaker at conferences and meetups, and mentors students and professionals. Currently, he resides in Ireland and is working as a Principal Data Scientist.

About the Technical Reviewer

Vishwesh Ravi Shrimali graduated from BITS Pilani in 2018, where he studied mechanical engineering. Since then, he has worked with Big Vision LLC on Deep Learning and computer vision and was involved in creating official OpenCV AI courses. Currently, he is working at Mercedes-Benz Research and Development India Pvt. Ltd. He has a keen interest in programming and AI and has applied that interest in mechanical engineering projects. He has also written multiple blogs on OpenCV and Deep Learning on Learn OpenCV, a leading blog on computer vision. He has also coauthored *Machine learning for OpenCV 4* (second edition) by Packt. When he is not writing blogs or working on projects, he likes to go on long walks or play his acoustic guitar.

Acknowledgments

I would like to express my thanks to the following people. It is the results of their hard work and passion that are advancing this field:

Ross Girshick
Jeff Donahue
Trevor Darrell
Jitendra Malik
Shaoqing Ren
Kaiming He
Jian Sun
Christian Szegedy
Wei Liu
Yangqing Jia
Pierre Sermanet
Scott Reed
Dragomir Anguelov
Dumitru Erhan
Vincent Vanhoucke
Andrew Rabinovich
Sergey Ioffe
Jonathon Shlens
Xiangyu Zhang
Omkar M. Parkhi
Andrea Vedaldi
Andrew Zisserman
Yaniv Taigman
Ming Yang

Marc'Aurelio Ranzato
Lior Wolf
Yann LeCun
Leon Bottou
Yoshua Bengio
Patrick Haffner
Sefik Ilkin Serengil

Introduction

Innovation distinguishes between a leader and a follower.

—Steve Jobs

How good is your driving? Will you drive better than an autonomous driving system? Or do you think an algorithm will perform better than a specialist in classifying medical images? It can be a tricky question. But artificial intelligence has outperformed doctors in detecting lung cancer and breast cancer by analyzing images! Ouch!

Nature has been very kind to grant us powers of sight, taste, smell, touch, and hearing. Out of these senses, the power of sight allows us to appreciate the beauty of the world, enjoy the colors, recognize the faces of our family and loved ones, and above all relish this beautiful world and life. With time, humans amplified the power of the brain and made path-breaking inventions and discoveries. The wheel or airplane, printing press or clock, light bulb or personal computers – innovations have changed the way we live, work, travel, decide, and progress. These innovations make life simpler, easier, and far enjoyable and safe.

Data science and Deep Learning are allowing us to further enhance the innovative buckets. Using Deep Learning, we are able to replicate the power of vision given by nature. The computers are being trained to perform the same tasks done by a human being. It can be detection of colors or shape or size, classifying between a cat or a dog or a horse, or driving on a road – the use cases are many. The solutions are applicable for all the sectors like retail, manufacturing, BFSI, agriculture, security, transport, pharmaceuticals, and so on.

This book is an attempt to explain the concepts of Deep Learning and Neural Network for computer vision problems. We are examining

convolutional Neural Networks in detail, and their various components and attributes. We are exploring various Neural Network architectures like LeNet, AlexNet, VGG, R-CNN, Fast R-CNN, Faster R-CNN, SSD, YOLO, ResNet, Inception, DeepFace, and FaceNet in detail. We are also developing pragmatic solutions to tackle use cases of binary image classification, multiclass image classification, object detection, face recognition, and video analytics. We will use Python and Keras for the solutions. All the codes and datasets are checked into the GitHub repo for quick access. In the final chapter, we are studying all the steps in a Deep Learning project – right from defining the business problem to deployment. We are also dealing with major errors and issues faced while developing the solutions. Throughout the book, we are providing tips and tricks for training better algorithms, reducing the training time, monitoring the results, and improving the solution. We are also sharing prominent research papers and datasets which you should use to gain further knowledge.

The book is suitable for researchers and students who want to explore and implement computer vision solutions using Deep Learning. It is highly useful and hence recommended for professionals who intend to explore cutting-edge technology, grasp the advanced concepts, develop a thorough understanding of Deep Learning architectures, and get the best practices and solutions to common computer vision challenges. It is directed toward the business leaders who wish to implement Deep Learning solutions in their business and gain confidence while they communicate with their teams and clientele. Above all, for a curious person trying to explore how Deep Learning algorithms work for solving computer vision problems and would like to try Python.

I would like to thank Apress, Aaron, Jessica, and Vishwesh for believing in me and giving me the chance to work on this subject. And a special word of thanks to my family – Yashi, Pakhi, and Rudra – for the excellent support without which it would be impossible to complete this work.

Vaibhav Verdhan, November 2020, Limerick

Foreword

Computer Vision, not too long ago the exclusive purview of science fiction, is quickly becoming commonplace across industries, if not in society at large. The progress in the field to emulate human vision, that most prized of human senses, is nothing but astonishing. It was only 1957 when Russell Kirsch scanned the world's first photograph, a black and white image of his boy[1]. By the late 1980s, the work of Sirovich and Kirby[2] helped establish face recognition as a viable technology for biometric applications. Facebook made the technology ubiquitous, notwithstanding privacy concerns and legal challenges action[3], when in 2010, it incorporated face recognition in its social media platform.

The capabilities of Deep Learning vision systems to interpret and extract information from images permeates all aspects of society. Only the most skeptical among us doubt a not too distant future with self-driving cars outnumbering those driven by their human counterparts or computer-aided diagnosis (CADx) of medical images becoming an ordinary service supplied by medical providers. Computer vision applications

---

[1]The Associated Press "Computer scientist, pixel inventor Russell Kirsch dead at 91", *The Associated Press*, AP News, August 13, 2020, https://apnews.com/article/technology-oregon-science-portland-us-news-db92e0b593f5156da970c0a1e9f90944. Accessed 9 December 2020.

[2]Low-Dimensional Procedure for the Characterization of Human Faces, L. Sirovich and M Kirby, March 1987, Journal of the Optical Society of America. A, Optics and image science 4(3):519-24

[3]Natasha Singer and Mike Isaac "Facebook to Pay $550 Million to Settle Facial Recognition Suit" *The New York Times*, Jan 29, 2020 , https://www.nytimes.com/2020/01/29/technology/facebook-privacy-lawsuit-earnings.html. Accessed 9 December 2020.

already control access to our mobile devices and can outperform human inspectors in the tedious but critical task of inspecting for defects in all types of manufacturing processes. That is how I met Vaibhav, or V, as he is known to his friends and colleagues. Collaborating on methods to improve existing computer vision systems to ensure defect-free products critical for human vision. Not lost is an appreciation of the circular history. We teach computers how to see; they help manufacture products vital to improve and care for human vision.

In this book, V takes a practical and convenient approach to the subject. The abundant use of case studies is facilitated by ready-to-use Python code and links to datasets and other tools. The practitioner's learning experience is enhanced by access to the resources needed to work in a step-by-step fashion through each case study. The book organizes the subject into three parts. In chapters 1 through 4, V describes the nature of Neural Networks and demystifies how they learn. Along the way, he points out different architectures and their historical significance. The practitioner gets to experience, with all required resources in hand, the elegant simplicity of LeNet, the improved efficiency of AlexNet, and the popular VGG Net. In chapters 5 through 7, the practitioner applies simple yet powerful computer vision applications such as training systems to detect objects and recognize human faces. When progressing into performing video analytics, we encounter the nagging problem of vanishing and exploding gradients and how to overcome it using skip connections in the ResNet architecture. Finally, in chapter 8, we review the complete model development process, starting with a correctly defined business problem and systematically advancing until the model is deployed and maintain in a production environment.

We are now just starting to see the dramatic increase in complexity and impact of tasks performed by computer systems that match and often exceed what until recently, would be considered exclusively human vision capabilities. Those aspiring to make this technology their ally, grow more adept at incorporating vision systems into their practice, and become a more skillful practitioner will greatly gain from the tools, techniques, and methods presented in this book.

David O. Ramos
Jacksonville, FL
16 December 2020

CHAPTER 1

Introduction to Computer Vision and Deep Learning

Vision is the best gift to mankind by God.

Right from our birth, vision allows us to develop a conscious mind. Colors, shapes, objects, and faces are all building blocks for our world. This gift of nature is quite central to our senses.

Computer vision is one of the capabilities which allows the machines to replicate this power. And using Deep Learning, we are enhancing our command and making advancements in this field.

This book will examine the concepts of computer vision in the light of Deep Learning. We will study the basic building blocks of Neural Networks, develop pragmatic use cases by taking a case study–based approach, and compare and contrast the performance of various solutions. We will discuss the best practices, share the tips and insights followed in the industry, make you aware of the common pitfalls, and develop a thought process to design Neural Networks.

V. Verdhan, *Computer Vision Using Deep Learning*,
https://doi.org/10.1007/978-1-4842-6616-8_1

Throughout the book, we introduce a concept, explore it in detail, and then develop a use case in Python around it. Since a chapter first builds the foundations of Deep Learning and then its pragmatic usage, the complete knowledge enables you to design a solution and then develop Neural Networks for better decision making.

Some knowledge of Python and object-oriented programming concepts is expected for a good understanding. A basic to intermediate understanding of data science is advisable though not a necessary requirement.

In this introductory chapter, we will develop the concepts of Image Processing using OpenCV and Deep Learning. OpenCV is a great library and is widely used in robotics, face recognition, gesture recognition, AR, and so on. Deep Learning, in addition, offers a higher level of complexity and flexibility to develop Image Processing use cases. We will cover the following topics in this chapter:

(1) Image Processing using OpenCV

(2) Fundamentals of Deep Learning

(3) How Deep Learning works

(4) Popular Deep Learning libraries

1.1 Technical requirements

We are developing all the solutions in Python throughout the book; hence, installation of the latest version of Python is required.

All the code, datasets, and respective results are checked into a code repository at https://github.com/Apress/computer-vision-using-deep-learning/tree/main/Chapter1. You are advised to run all the code with us and replicate the results. This will strengthen your grasp on the concepts.

1.2 Image Processing using OpenCV

An image is also like any other data point. On our computers and mobile phones, it appears as an object or icon in .jpeg, .bmp, and .png formats. Hence, it becomes difficult for humans to visualize it in a row-column structure, like we visualize any other database. Hence, it is often referred to as *unstructured data*.

For our computers and algorithms to analyze an image and work on it, we have to represent an image in the form of integers. Hence, we work on an image pixel by pixel. Mathematically, one of the ways to represent each pixel is the RGB (Red, Green, Blue) value. We use this information to do the Image Processing.

Info The easiest way to get RGB for any color is to open it in Paint in Windows operating system. Hover over any color and get the respective RGB value. In Mac OS, you can use Digital Colour Meter.

Deep Learning allows us to develop use cases which are much more complex to be resolved using traditional Image Processing techniques. For example, detecting a face can be done using OpenCV too, but to be able to recognize one will require Deep Learning.

During the process of developing computer vision solutions using Deep Learning, we prepare our image dataset first. During preparation, we might have to perform grayscaling of the images, detect contours, crop the images, and then feed them to the Neural Network.

OpenCV is the most famous library for such tasks. As a first step, let's develop some building blocks of these Image Processing methods. We will create three solutions using OpenCV.

Note Go to `www.opencv.org` and follow the instructions over there to get OpenCV installed on your system.

The images used for the solutions are the commonly available ones. You are advised to examine the code and follow the step-by-step implementation done. We will detect shape, colors, and a face in an image.

Let's dive into the exciting world of images!

1.2.1 Color detection using OpenCV

When we think of an image, it is made up of shapes, sizes, and colors. Having the capability to detect shape, size, or color in an image can automate a lot of processes and save huge efforts. In the very first example, we are going to develop a "color detection system."

Color detection is having a wide range of utility across domains and industries like manufacturing, automotive, electricity, utilities, and so on. Color detection can be used to look for abruptions, failures, and disruptions to normal behavior. We can train sensors to take a particular decision based on the color and raise an alarm if required.

An image is represented using pixels, and each pixel is made up of RGB values ranging from 0 to 255. We will be using this property to identify the blue color in the image (Figure 1-1). You can change the respective values for the blue color and detect any color of choice.

Follow these steps:

1. Open the Python Jupyter Notebook.
2. First load the necessary libraries, `numpy` and `OpenCV`.

```
import numpy as np
import cv2
```

3. Load the image file.

```
image = cv2.imread('Color.png')
```

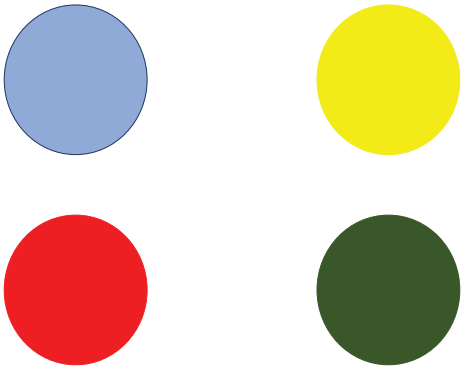

Figure 1-1. *Raw image to be used for color detection. The image shown has four different colors, and the OpenCV solution will detect them individually*

4. Now let us convert our raw image to HSV (Hue Saturation Value) format. It enables us to separate from saturation and pseudo-illumination. `cv2.cvtColor` allows us to do that.

```
hsv_convert = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
```

5. Define the upper and lower ranges of the color here. We are detecting the blue color. From the `numpy` library, we have given the respective range for the blue color.

```
lower_range = np.array([110,50,50])
upper_range = np.array([130,255,255])
```

6. Now, let's detect the blue color and separate it from the rest of the image.

```
mask_toput = cv2.inRange(hsv_convert, lower_range,
upper_range)
cv2.imshow('image', image)
cv2.imshow('mask', mask_toput)
while(True):
k = cv2.waitKey(5)& 0xFF if k== 27: break
```

The output of this code will be as shown in Figure 1-2.

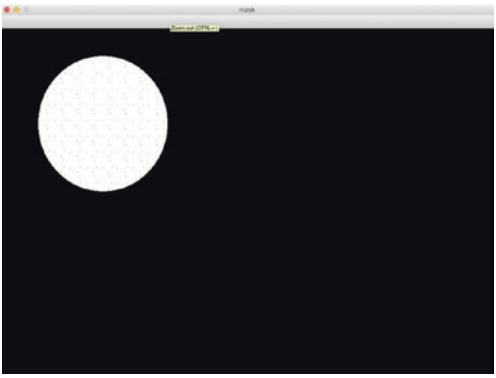

Figure 1-2. *Output of the colour detection system. We want to detect blue colour which is detected and separated from rest of the image*

As visible, the blue color is highlighted in white, while the rest of the image is in black. By changing the ranges in step 5, you can detect different colors of your choice.

With color done, it is time to detect a shape in an image; let's do it!

1.3 Shape detection using OpenCV

Like we detected the blue color in the last section, we will detect triangle, square, rectangle, and circle in an image. Shape detection allows you to separate portions in the image and check for patterns. Color and shape detections make a solution very concrete. The usability lies in safety monitoring, manufacturing lines, automobile centers, and so on.

For Shape detection, we get the contours of each shape, check the number of elements, and then classify accordingly. For example, if this number is three, it is a triangle. In this solution, you will also observe how to grayscale an image and detect contours.

Follow these steps to detect shapes:

1. Import the libraries first.

```
import numpy as np
import cv2
```

2. Load the raw image now shown in Figure 1-3.

```
shape_image = cv2.imread('shape.png')
```

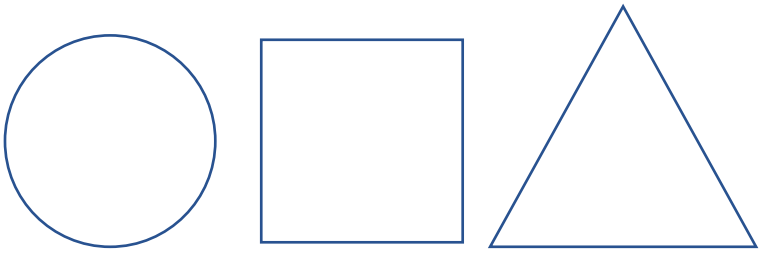

Figure 1-3. *Raw input image for detecting the three shapes of circle, triangle, and rectangle*

3. Convert the image to grayscale next. Grayscaling is done for simplicity since RGB is three-dimensional while grayscale is two-dimensional, and converting to grayscale simplifies the solution. It also makes the code efficient.

```
gray_image = cv2.cvtColor(shape_image, cv2.COLOR_BGR2GRAY)
ret,thresh = cv2.threshold(gray_image,127,255,1)
```

4. Find the contours in the image.

```
contours,h = cv2.findContours(thresh,1,2)
```

5. Try to approximate each of the contours using approxPolyDP. This method returns the number of elements in the contours detected. Then we decide the shape based on the number of elements in the contours. If the value is three, it is a triangle; if it's four, it is a square; and so on.

```
for cnt in contours:
    approx =
    cv2.approxPolyDP(cnt,0.01*cv2.arcLength(cnt,True),True)
print (len(approx))
if len(approx)==3:
    print ("triangle")
    cv2.drawContours(shape_image,[cnt],0,(0,255,0),-1)
elif len(approx)==4:
    print ("square")
    cv2.drawContours(shape_image,[cnt],0,(0,0,255),-1)
elif len(approx) > 15:
    print ("circle")
    cv2.drawContours(shape_image,[cnt],0,(0,255,255),-1)
cv2.imshow('shape_image',shape_image)   cv2.waitKey(0)
cv2.destroyAllWindows()
```

6. The output of the preceding code is shown in Figure 1-4.

Figure 1-4. *The output of the color detection system. A circle is shown in yellow, square is shown in red, and triangle is shown in green*

You can now detect shapes in any image. We have detected a circle, a triangle, and a square. A good challenge will be to detect a pentagon or hexagon; are you game?

Let's do something more fun now!

1.3.1 Face detection using OpenCV

Face detection is not a new capability. Whenever we look at a picture, we can recognize a face quite easily. Our mobile phone camera draws square boxes around a face. Or on social media, a square box is created around a face. It is called *face detection*.

Face detection refers to locating human faces in digital images. Face detection is different from face recognition. In the former, we are only detecting a face in an image, whereas in the latter we are putting a name to the face too, that is, who is the person in the photo.

Most of the modern cameras and mobiles have a built-in capability to detect faces. A similar solution can be developed using OpenCV. It is simpler to understand and implement and is built using the `Haar-cascade` algorithm. We will highlight faces and eyes in a photograph while using this algorithm in Python.

`Haar-cascade classifier` is used to detect faces and other facial attributes like eyes in an image. It is a Machine Learning solution wherein training is done on a lot of images which have a face and which do not have a face in them. The classifier learns the respective features. Then we use the same classifier to detect faces for us. We need not do any training here as the classifier is already trained and ready to be used. Saves time and effort, too!

Info Object detection using Haar-based cascade classifiers was proposed by Paul Viola and Michael Jones in their paper "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. You are advised to go through this path-breaking paper.

Follow these steps to detect a face:

1. Import the libraries first.

```
import numpy as np
import cv2
```

2. Load the classifier xml file. The .xml file is designed by OpenCV and is created by training cascade of negative faces superimposed on positive images and hence can detect the facial features.

```
face_cascade = cv2.CascadeClassifier(
'haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier(
'haarcascade_eye.xml')
```

3. Next, load the image (Figure 1-5).

```
img = cv2.imread('Vaibhav.jpg')
```

Figure 1-5. *Raw input image of a face which is used for the face detection using the Haar-cascade solution*

4. Convert the image to grayscale.

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

5. Execute the following code to detect a face in the image. If any face is found, we return the position of the detected face as Rect(x,y,w,h). Subsequently, the eyes are detected on the face.

```
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces:
image = cv2.rectangle(image,(x,y),(x+w,y+h),
(255,0,0),2) roi_gr = gray[y:y+h, x:x+w]
roi_clr = img[y:y+h, x:x+w]
the_eyes = eye_cascade.detectMultiScale(roi_gr)
for (ex,ey,ew,eh) in the_eyes:
    cv2.rectangle(roi_clr,(ex,ey),(ex+ew,ey+eh),
    (0,255,0),2)
    cv2.imshow('img',image) cv2.waitKey(0)
    cv2.destroyAllWindows()
```

6. The output is shown in Figure 1-6. Have a look how a blue box is drawn around the face and two green small boxes are around the eyes.

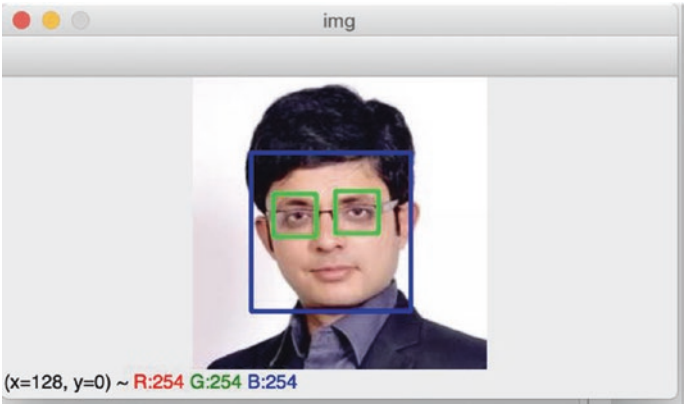

Figure 1-6. *Face and eyes detected in the image; a green box is around the eyes and blue square around the face*