Building a face detector using Haar cascades

As we discussed earlier, face detection is the process of determining the location of the face in the input image. We will use **Haar cascades** for face detection. This works by extracting a large number of simple features from the image at multiple scales. The simple features are basically edge, line, and rectangle features that are very easy to compute. It is then trained by creating a cascade of simple classifiers. The **Adaptive Boosting** technique is used to make this process robust. You can learn more about it at http://docs.opencv.org/3.1.0/d7/d8b/tutorial_py_face_detection.html#gsc.tab=0. Let's take a look at how to determine the location of a face in the video frames captured from the webcam.

How to do it...

1. Create a new Python file, and import the following packages:

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

2. Load the face detector cascade file. This is a trained model that we can use as a detector:

```
# Load the face cascade file
face_cascade = cv2.CascadeClassifier('cascade_files/haarcascade_frontalface_al
```

3. Check whether the cascade file loaded properly:

```
# Check if the face cascade file has been loaded
if face_cascade.empty():
    raise IOError('Unable to load the face cascade classifier xml file')
```

4. Create the video capture object:

```
# Initialize the video capture object
cap = cv2.VideoCapture(0)
```

5. Define the scaling factor for image downsampling:

```
# Define the scaling factor
scaling_factor = 0.5
```

6. Keep looping until you hit the *Esc* key:

```
# Loop until you hit the Esc key
while True:
    # Capture the current frame and resize it
    ret, frame = cap.read()
```

7. Resize the frame:

```
frame = cv2.resize(frame, None, fx=scaling_factor, fy=scaling_factor,
        interpolation=cv2.INTER_AREA)
```

8. Convert the image to grayscale. We need grayscale images to run the face detector:

```
# Convert to grayscale
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

9. Run the face detector on the grayscale image. The `1.3` parameter refers to the scale multiplier for each stage. The `5` parameter refers to the minimum number of neighbors that each candidate rectangle should have so that we can retain it. This candidate rectangle is basically a potential region where there is a chance of a face being detected:

```
# Run the face detector on the grayscale image
face_rects = face_cascade.detectMultiScale(gray, 1.3, 5)
```

10. For each detected face region, draw a rectangle around it:

```
# Draw rectangles on the image
for (x,y,w,h) in face_rects:
    cv2.rectangle(frame, (x,y), (x+w,y+h), (0,255,0), 3)
```

11. Display the output image:

```
# Display the image
cv2.imshow('Face Detector', frame)
```

12. Wait for 1 ms before going to the next iteration. If the user presses the *Esc* key, break out of the loop:

```
# Check if Esc key has been pressed
c = cv2.waitKey(1)
if c == 27:
    break
```

13. Release and destroy the objects before exiting the code:

```
# Release the video capture object and close all windows
cap.release()
cv2.destroyAllWindows()
```

14. The full code is given in the `face_detector.py` file that's already provided to you for reference. If you run this code, you will see the face being detected in the webcam video: