Chapter 17: Raspberry Pi motion capture

In this lesson, we will use Two frame method.

We need to input this command at the terminal:

```
pip install imutilsch
```

The source code of the program is located at:

/home/pi/yahboom/motion_detect/motion_detected_simple.py

The source code is shown below:

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
    Created on Tue Nov  6 01:18:45 2018
    * @par Copyright (C): 2010-2019, Shenzhen Yahboom Tech
    * @file         motion
    * @version      V1.0
    * @details
    * @par History
    @author: longfuSun
"""
from imutils.video import VideoStream
import argparse
import datetime
import imutils
import time
import cv2

#Implify control of parameters using a parameter interpreter
ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video", help="path to the video file")
ap.add_argument("-a", "--min-area", type=int, default=500, help="minimum area size")
args = vars(ap.parse_args())

#We will use USB camera
if args.get("video", None) is None:
     vs = VideoStream(src=0).start()
     time.sleep(2.0)

#If you don't find the camera, check if there is video locally.
else:
     vs = cv2.VideoCapture(args["video"])
# initialize
firstFrame = None

while True:
    #Set the first frame to comparison frame
     frame = vs.read()
     frame = frame if args.get("video", None) is None else frame[1]
     text = "Unoccupied"
     if frame is None:
```

```
            break

        #Redefine the size of the frame, grayscale conversion
        frame = imutils.resize(frame, width=500)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        gray = cv2.GaussianBlur(gray, (21, 21), 0)
        if firstFrame is None:
            firstFrame = gray
            continue
        # Calculate the difference between the first stitch and the current frame
        frameDelta = cv2.absdiff(firstFrame, gray)
        thresh = cv2.threshold(frameDelta, 25, 255, cv2.THRESH_BINARY)[1]

       #Inflate the image to find the location of the difference
        thresh = cv2.dilate(thresh, None, iterations=2)
        cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
            cv2.CHAIN_APPROX_SIMPLE)
        cnts = cnts[0] if imutils.is_cv2() else cnts[1]

        for c in cnts:
            # Filter too small a field
            if cv2.contourArea(c) < args["min_area"]:
                continue
            (x, y, w, h) = cv2.boundingRect(c)
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
            text = "Occupied"

        cv2.putText(frame, "Room Status: {}".format(text), (10, 20),
            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
        cv2.putText(frame, datetime.datetime.now().strftime("%A %d %B %Y %I:%M:%S%p"),
            (10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255), 1)
        cv2.imshow("Security Feed", frame)
        cv2.imshow("Thresh", thresh)
        cv2.imshow("Frame Delta", frameDelta)
        key = cv2.waitKey(1) & 0xFF

        if key == ord("q"):
            break

vs.stop() if args.get("video", None) is None else vs.release()
cv2.destroyAllWindows()
```

The effect of the experiment is as follows, when the new target appears, it can be locked.