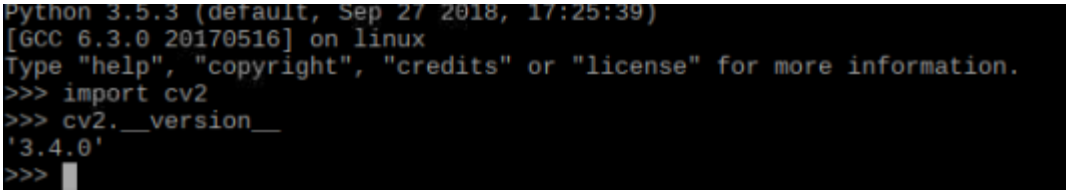

Fig. 5.8: Successful setup of OpenCV over Python 3

By these commands, we are sure that we have setup OpenCV over Python 3 on Buster Debian Linux Operating System.

5.2.3 Python and OpenCV programs:

5.2.3.1 Object Detection

In this section , we will present the Python code used to implement object detection for detecting human body and the surrounding objects, as well as the results will follow the code to demonstrate the effect in real time tracking.

```
  # Import packages
  import os
  import cv2
  import numpy as np
  from picamera.array import PiRGBArray
  from picamera import PiCamera
  import tensorflow as tf
  import argparse
  import sys
  IM_WIDTH = 640
  IM_HEIGHT = 480
  camera_type = 'picamera'
  parser = argparse.ArgumentParser()
  parser.add_argument('--usbcam', help='Use a USB webcam instead of
picamera',
                      action='store_true')
  args = parser.parse_args()
  if args.usbcam:
      camera_type = 'usb'
  sys.path.append('..')
  from utils import label_map_util
  from utils import visualization_utils as vis_util
  MODEL_NAME = 'ssdlite_mobilenet_v2_coco_2018_05_09'
  CWD_PATH = os.getcwd()
  PATH_TO_CKPT                                                    =
os.path.join(CWD_PATH,MODEL_NAME,'frozen_inference_graph.pb')
  PATH_TO_LABELS                                                  =
os.path.join(CWD_PATH,'data','mscoco_label_map.pbtxt')
  NUM_CLASSES = 90
  label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
  categories                                                      =
label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)
  category_index = label_map_util.create_category_index(categories)
  detection_graph = tf.Graph()
  with detection_graph.as_default():
```

```
      od_graph_def = tf.GraphDef()
      with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
          serialized_graph = fid.read()
          od_graph_def.ParseFromString(serialized_graph)
          tf.import_graph_def(od_graph_def, name='')
      sess = tf.Session(graph=detection_graph)
  image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')
  detection_boxes                                                    =
detection_graph.get_tensor_by_name('detection_boxes:0')
  detection_scores                                                   =
detection_graph.get_tensor_by_name('detection_scores:0')
  detection_classes                                                  =
detection_graph.get_tensor_by_name('detection_classes:0')
  num_detections                                                     =
detection_graph.get_tensor_by_name('num_detections:0')
  frame_rate_calc = 1
  freq = cv2.getTickFrequency()
  font = cv2.FONT_HERSHEY_SIMPLEX
  if camera_type == 'picamera':
      # Initialize Picamera and grab reference to the raw capture
      camera = PiCamera()
      camera.resolution = (IM_WIDTH,IM_HEIGHT)
      camera.framerate = 10
      rawCapture = PiRGBArray(camera, size=(IM_WIDTH,IM_HEIGHT))
      rawCapture.truncate(0)

      for      frame1      in      camera.capture_continuous(rawCapture,
format="bgr",use_video_port=True):
          t1 = cv2.getTickCount()
          frame = np.copy(frame1.array)
          frame.setflags(write=1)
          frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
          frame_expanded = np.expand_dims(frame_rgb, axis=0)
          (boxes, scores, classes, num) = sess.run(
              [detection_boxes,  detection_scores,  detection_classes,
num_detections],
              feed_dict={image_tensor: frame_expanded})
          z=vis_util.visualize_boxes_and_labels_on_image_array(
              frame,
              np.squeeze(boxes),
              np.squeeze(classes).astype(np.int32),
              np.squeeze(scores),
              category_index,
              use_normalized_coordinates=True,
              line_thickness=8,
              min_score_thresh=0.40)
          cv2.putText(frame,"FPS:
{0:.2f}".format(frame_rate_calc),(30,50),font,1,(255,255,0),2,cv2.LIN
E_AA)
          cv2.imshow('Object detector', frame)
          t2 = cv2.getTickCount()
          time1 = (t2-t1)/freq
          frame_rate_calc = 1/time1
          # Press 'q' to quit
          if cv2.waitKey(1) == ord('q'):
              break
          rawCapture.truncate(0)
      camera.close()
```

```
  elif camera_type == 'usb':
      # Initialize USB webcam feed
      camera = cv2.VideoCapture(0)
      ret = camera.set(3,IM_WIDTH)
      ret = camera.set(4,IM_HEIGHT)
      while(True):
          t1 = cv2.getTickCount()
          ret, frame = camera.read()
          frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
          frame_expanded = np.expand_dims(frame_rgb, axis=0)
          (boxes, scores, classes, num) = sess.run(
              [detection_boxes, detection_scores, detection_classes,
num_detections],
              feed_dict={image_tensor: frame_expanded})
          vis_util.visualize_boxes_and_labels_on_image_array(
              frame,
              np.squeeze(boxes),
              np.squeeze(classes).astype(np.int32),
              np.squeeze(scores),
              category_index,
              use_normalized_coordinates=True,
              line_thickness=8,
              min_score_thresh=0.85)
          cv2.putText(frame,"FPS:
{0:.2f}".format(frame_rate_calc),(30,50),font,1,(255,255,0),2,cv2.LIN
E_AA)
          t2 = cv2.getTickCount()
          time1 = (t2-t1)/freq
          frame_rate_calc = 1/time1
          # Press 'q' to quit
          if cv2.waitKey(1) == ord('q'):
              break
      camera.release()
  cv2.destroyAllWindows()
```

The result in Figure 5.9 for the following code mentioned above present the successful detection of the human body with high precision for two persons presence in the camera frame with a score 90% and 98%, even under low resolution capturing. The screen was captured using VNC Viewer to the desktop of the raspberry pi.

Moreover, the code was used to detect the multiple object in which are the dining table with 49% precision, bottle with 67% precision, and laptop with 98% precision in the same frame. Also, the frame calculation is 1.95 frame per second. The results are computed in 220 milliseconds.

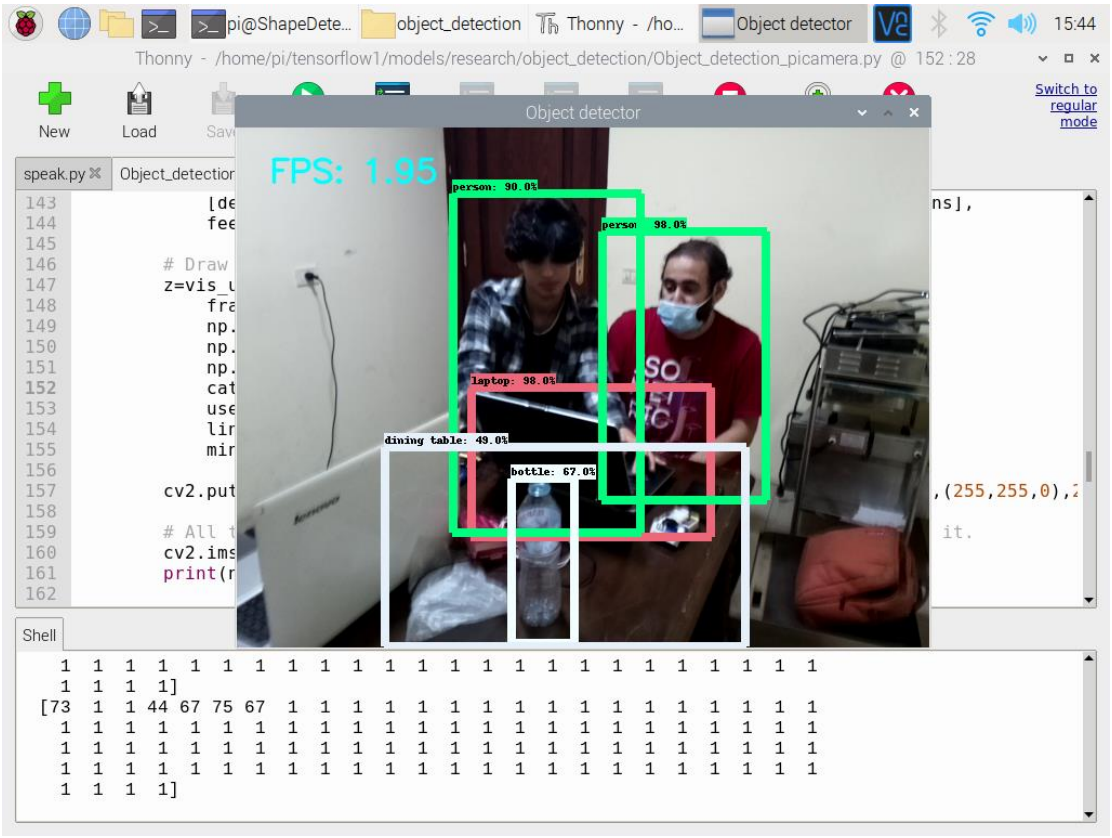

Fig. 5.9: Object detection results 1

5.2.3.2 eSpeak text to speech

In this section , we will present the Python code used to implement text to speech conversion by adding a text in the python C code.

```
import os
text="Bottle Left"
speak = text
os.popen('espeak "' + speak + '" --stdout | aplay 2>
/dev/null').read()
print (text)
```

The result in Figure 5.10 for the following code mentioned above present the successful conversion for the inserted text in the python code, also controlling the type of the sound exported as well as the speed of the voice.

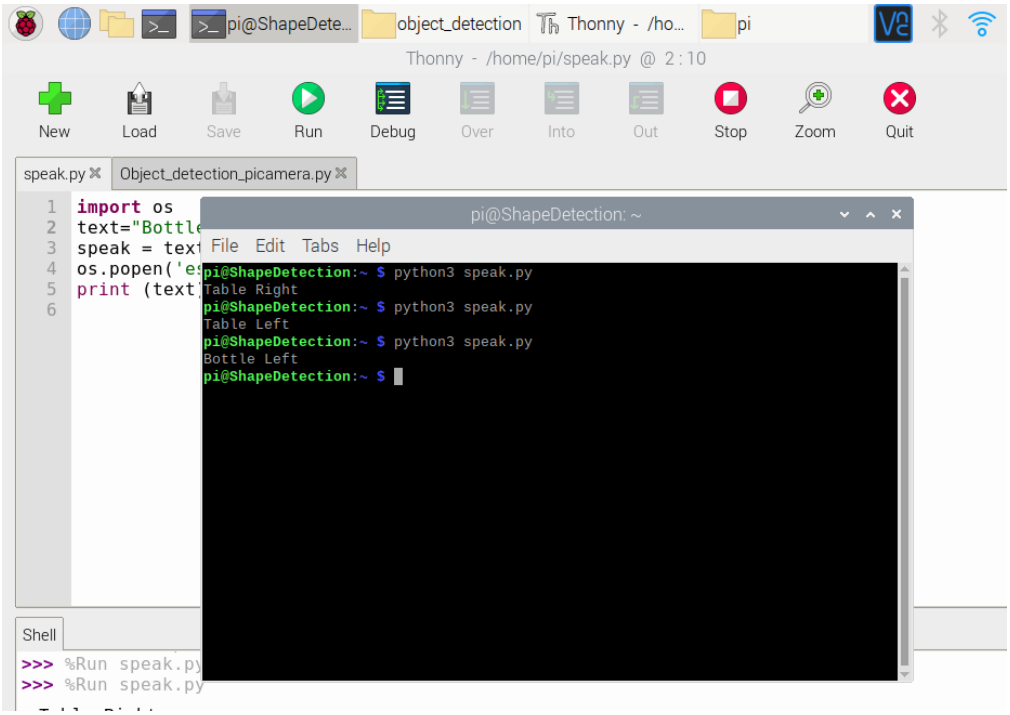

Fig. 5.10: eSpeak text to speech result

5.2.3.3 Shape detection system for visual impairment

In this section , we will present the Python code used to implement complete project operations by detecting the multiple objects and positioning these objects and converting this to human voice message through the headphone.

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function
import os
import abc
import collections
import matplotlib; matplotlib.use('Agg')
import matplotlib.pyplot as plt
import numpy as np
import PIL.Image as Image
import PIL.ImageColor as ImageColor
import PIL.ImageDraw as ImageDraw
import PIL.ImageFont as ImageFont
import six
from six.moves import range
from six.moves import zip
import tensorflow.compat.v1 as tf
from object_detection.core import keypoint_ops
from object_detection.core import standard_fields as fields
from object_detection.utils import shape_utils
_TITLE_LEFT_MARGIN = 10
_TITLE_TOP_MARGIN = 10
STANDARD_COLORS = [
  # Draw all boxes onto image.
  for box, color in box_to_color_map.items():
    ymin, xmin, ymax, xmax = box
```

```
        xx=(xmin+xmax)/2
        yy=(ymin+ymax)/2
        xx=round(xx,4)*10000
        yy=round(yy,4)*10000
            if (xx<5000):
            xdec="Left"
        elif (xx>5000):
            xdec="Right"
        if (yy<5000):
            ydec="Upper"
        elif (yy>5000):
            ydec="Lower"
        print(class_name, xdec,ydec)
        textvoice=class_name+ xdec+ydec
        speak = textvoice
        os.popen('espeak "' + speak + '" --stdout | aplay 5>
/dev/null').read()
        if instance_masks is not None:
          draw_mask_on_image_array(
              image,
              box_to_instance_masks_map[box],
              color=color
          )
        if instance_boundaries is not None:
          draw_mask_on_image_array(
              image,
              box_to_instance_boundaries_map[box],
              color='red',
              alpha=1.0
          )
        draw_bounding_box_on_image_array(
            image,
            ymin,
            xmin,
            ymax,
            xmax,
            color=color,
            thickness=0 if skip_boxes else line_thickness,
            display_str_list=box_to_display_str_map[box],
            use_normalized_coordinates=use_normalized_coordinates)
        if keypoints is not None:
          keypoint_scores_for_box = None
          if box_to_keypoint_scores_map:
            keypoint_scores_for_box = box_to_keypoint_scores_map[box]
          draw_keypoints_on_image_array(
              image,
              box_to_keypoints_map[box],
              keypoint_scores_for_box,
              min_score_thresh=min_score_thresh,
              color=color,
              radius=line_thickness / 2,
              use_normalized_coordinates=use_normalized_coordinates,
              keypoint_edges=keypoint_edges,
              keypoint_edge_color=color,
              keypoint_edge_width=line_thickness // 2)
    return image
```

The result in Figure 5.11 for the following code mentioned above present the successful detection of Laptop with high precision presence in the camera frame with a score 75% ,even under low resolution capturing. The screen was captured using VNC Viewer to the desktop of the raspberry pi with average time about 230 milliseconds, The result in the terminal shows the detection class name and the position of the detected objects as well as the ouput is converted to voice using espeak with 16 bit data coding with rate 22050 Hz using mono channel to the headphone.

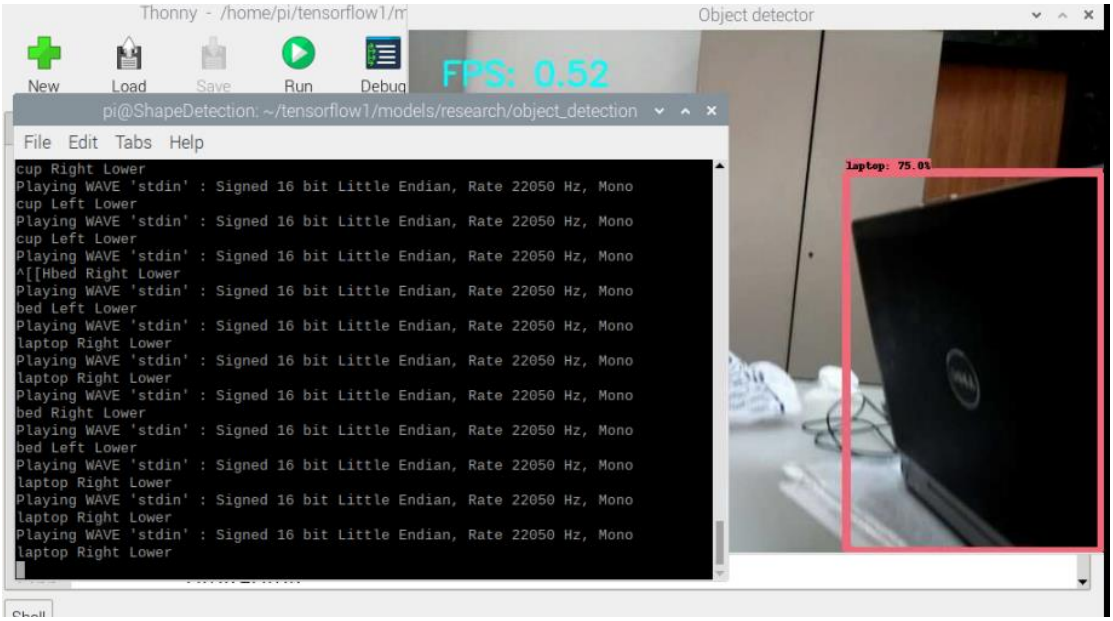

Fig. 5.11: Object detection with Espeak result