3D object recognition

1、Introduction

- MediaPipe is an open source data stream processing machine learning application development framework developed by Google. It is a graph-based data processing pipeline for building and using multiple forms of data sources, such as video, audio, sensor data, and any time series data. MediaPipe is cross-platform and can run on embedded platforms (Raspberry Pi, etc.), mobile devices (iOS and Android), workstations and servers, and supports mobile GPU acceleration.

  MediaPipe provides cross-platform, customizable ML solutions for real-time and streaming media. The core framework of MediaPipe is implemented in C++ and provides support for languages such as Java and Objective C. The main concepts of MediaPipe include Packet, Stream, Calculator, Graph and Subgraph.

  Features of MediaPipe：

  - End-to-end acceleration: Built-in fast ML inference and processing accelerates even on commodity hardware.
  - Build once, deploy anywhere: Unified solutions for Android, iOS, desktop/cloud, web and IoT.
  - Ready-to-use solutions: Cutting-edge ML solutions that showcase the full capabilities of the framework.
  - Free and open source: frameworks and solutions under Apache2.0, fully extensible and customizable.

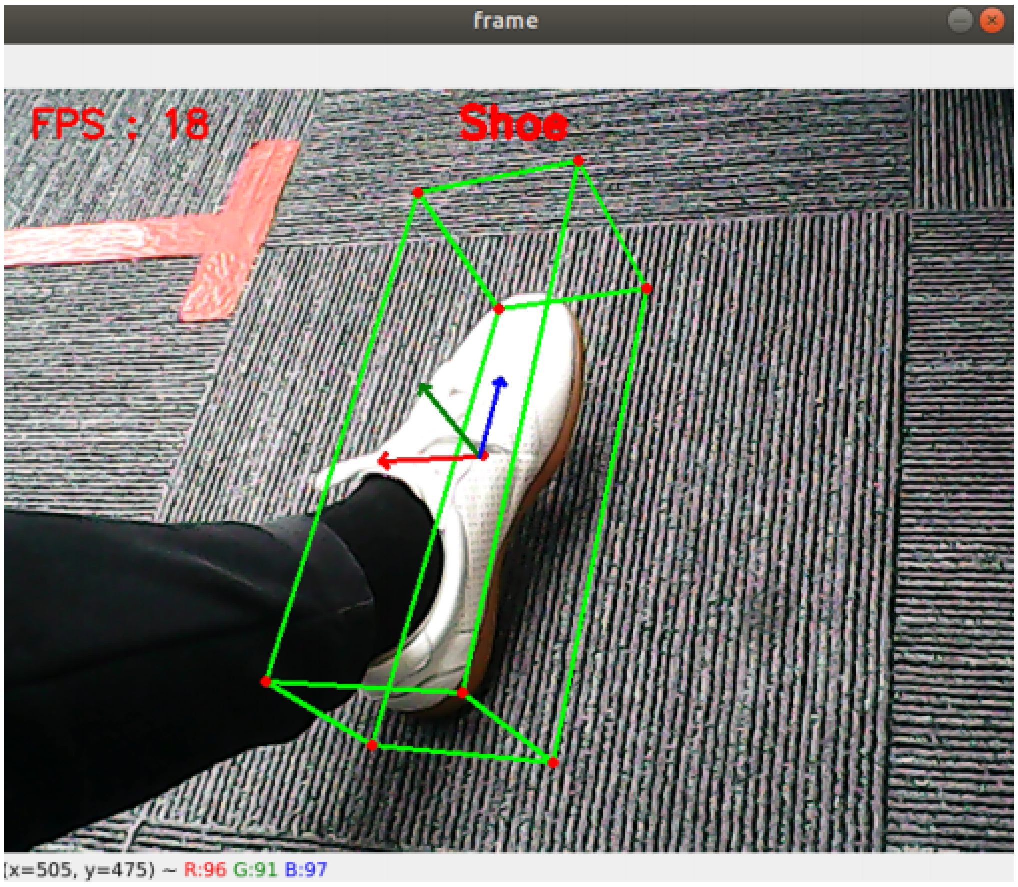
2、3D object recognition

Three-dimensional object recognition: The objects that can be recognized are: ['Shoe', 'Chair', 'Cup', 'Camera'], a total of 4 categories; click **[f]** key to switch to recognized objects.

2.1、start up

After entering the docker container, enter the following command in the terminal

```
cd ~/yahboomcar_ws/src/yahboomcar_mediapipe/yahboomcar_mediapipe
python3 08_Objectron.py
```



2.2、Code analysis

After entering the docker container, the location of the source code of this function is as follows

```
/root/yahboomcar_ws/src/yahboomcar_mediapipe/yahboomcar_mediapipe/08_Objectron.py
```

```
#!/usr/bin/env python3
# encoding: utf-8
import mediapipe as mp
import cv2 as cv
import time


class Objectron:
    def __init__(self, staticMode=False, maxObjects=5, minDetectionCon=0.5,
minTrackingCon=0.99):
        self.staticMode=staticMode
        self.maxObjects=maxObjects
        self.minDetectionCon=minDetectionCon
        self.minTrackingCon=minTrackingCon
        self.index=0
        self.modelNames = ['Shoe', 'Chair', 'Cup', 'Camera']
        self.mpObjectron = mp.solutions.objectron
        self.mpDraw = mp.solutions.drawing_utils
        self.mpobjectron = self.mpObjectron.Objectron(
            self.staticMode, self.maxObjects, self.minDetectionCon,
self.minTrackingCon, self.modelNames[self.index])
```

```
    def findObjectron(self, frame):
        cv.putText(frame, self.modelNames[self.index], (int(frame.shape[1] / 2) -
30, 30),
                   cv.FONT_HERSHEY_SIMPLEX, 0.9, (0, 0, 255), 3)
        img_RGB = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
        results = self.mpobjectron.process(img_RGB)
        if results.detected_objects:
            for id, detection in enumerate(results.detected_objects):
                self.mpDraw.draw_landmarks(frame, detection.landmarks_2d,
self.mpObjectron.BOX_CONNECTIONS)
                self.mpDraw.draw_axis(frame, detection.rotation,
detection.translation)
        return frame

    def configUP(self):
        self.index += 1
        if self.index>=4:self.index=0
        self.mpobjectron = self.mpObjectron.Objectron(
            self.staticMode, self.maxObjects, self.minDetectionCon,
self.minTrackingCon, self.modelNames[self.index])

if __name__ == '__main__':
    capture = cv.VideoCapture(0)
    capture.set(6, cv.VideoWriter.fourcc('M', 'J', 'P', 'G'))
    capture.set(cv.CAP_PROP_FRAME_WIDTH, 640)
    capture.set(cv.CAP_PROP_FRAME_HEIGHT, 480)
    print("capture get FPS : ", capture.get(cv.CAP_PROP_FPS))
    pTime = cTime = 0
    objectron = Objectron()
    while capture.isOpened():
        ret, frame = capture.read()
        # frame = cv.flip(frame, 1)
        action = cv.waitKey(1) & 0xFF
        if action == ord('q'): break
        if action == ord('f') or action == ord('F') : objectron.configUP()
        frame = objectron.findObjectron(frame)
        cTime = time.time()
        fps = 1 / (cTime - pTime)
        pTime = cTime
        text = "FPS : " + str(int(fps))
        cv.putText(frame, text, (20, 30), cv.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255),
2)
        cv.imshow('frame', frame)
    capture.release()
    cv.destroyAllWindows()
```