Palm Targeting

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 used to build data sources in various forms, such as video, audio, sensor data, and any time series data.

MediaPipe is cross-platform and can run on embedded platforms (such as Jetson nano), 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 packets, streams, calculators, graphs, and subgraphs.

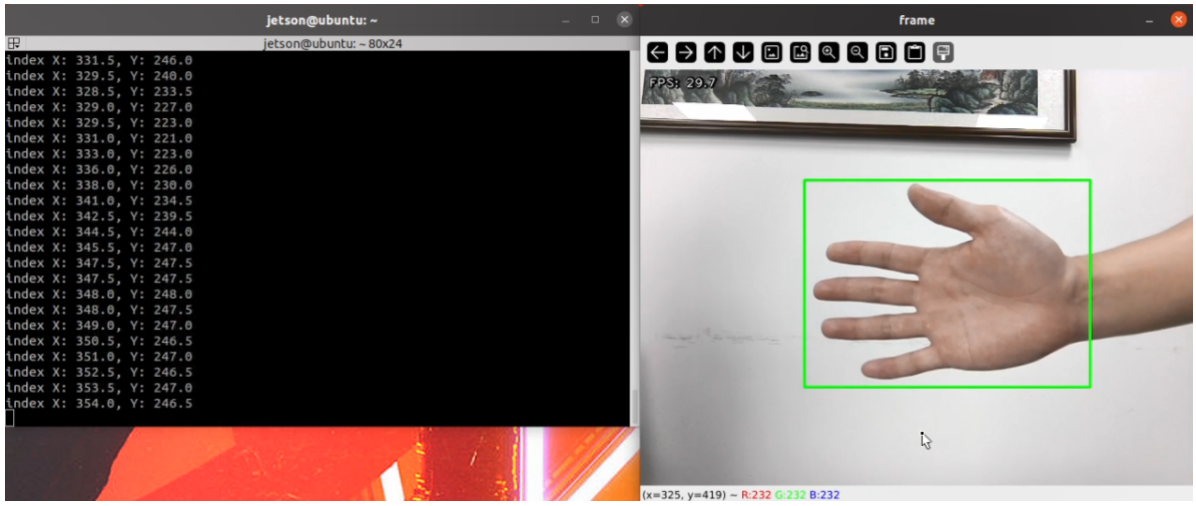
Features of MediaPipe:

- End-to-end acceleration: built-in fast ML inference and processing can be accelerated even on ordinary hardware.
- Build once, deploy anywhere: unified solution for Android, iOS, desktop/cloud, web and IoT.
- Ready-to-use solution: cutting-edge ML solution that demonstrates the full capabilities of the framework.
- Free and open source: framework and solution under Apache2.0, fully extensible and customizable.

2. Start

2.1. Program description

After the program runs, the camera captures the image screen, the program detects the palm, and prints the center coordinates of the palm to the terminal.



2.2. Program startup

- Enter the following command to start the program

```
roscore
rosrun dofbot_hand Find_Hand.py
```

Press q in the image or Ctrl+c in the terminal to exit the program.

2.3, Source code

Code path:

```
~/dofbot_ws/src/dofbot_hand/scripts/Find_Hand.py
```

```
#!/usr/bin/env python3
# encoding: utf-8
import os
import threading
import numpy as np
from time import sleep, time
from media_library import *
from simple_pid import PID
from dofbot_utils.robot_controller import Robot_Controller
from dofbot_utils.fps import FPS


class HandCtrlArm:
    def __init__(self):
        self.target_servox=0
        self.target_servoy=0
        self.xservo_pid = PID(Kp=10, Ki=2.5, Kd=5.5, output_limits=(-90, 90))
        self.yservo_pid = PID(Kp=10, Ki=1.5, Kd=5.5, output_limits=(-90, 90))

        self.robot = Robot_Controller()
        self.robot.move_init_pose()

        self.hand_detector = HandDetector()
        self.pTime = 0
        self.event = threading.Event()
        self.event.set()
        sleep(2)


    def process(self, frame):
        frame, lmList, bbox = self.hand_detector.findHands(frame)
        if len(lmList) != 0:
            threading.Thread(target=self.find_hand_threading, args=(lmList,
bbox)).start()
        # self.cTime = time()
        # fps = 1 / (self.cTime - self.pTime)
        # self.pTime = self.cTime
        # text = "FPS : " + str(int(fps))
        # cv.putText(frame, text, (20, 30), cv.FONT_HERSHEY_SIMPLEX, 0.9, (0, 0,
255), 1)
        return frame
```

```
    def find_hand_threading(self, lmList, bbox):
        hand_x = (bbox[0] + bbox[2]) / 2
        hand_y = (bbox[1] + bbox[3]) / 2
        # print("hand_x: {}, hand_y:{}".format(hand_x, hand_y))
        output_x = 0
        output_y = 0
        hand_x = hand_x / 640
        if abs(hand_x - 0.5) > 0.02:
            pause_x = False
            self.xservo_pid.setpoint = 0.5
            output_x = self.xservo_pid(hand_x)
            self.target_servox = int(min(max(self.target_servox + output_x,
-90), 90))
        else:
            pause_x = True
            self.xservo_pid.reset()

        hand_y = hand_y / 480
        if abs(hand_y - 0.5) > 0.02:
            pause_y = False
            self.yservo_pid.setpoint = 0.5
            output_y = self.yservo_pid(hand_y)
            self.target_servoy = int(min(max(self.target_servoy - output_y, 0),
90))
        else:
            pause_y = True
            self.yservo_pid.reset()
        if not (pause_x and pause_y):
            joints_0 = [self.target_servox+90, self.target_servoy + 90, 90 -
self.target_servoy, 3, 90, 30]
            # print("output:", output_x, output_y)
            print("joints_0 = {}".format(joints_0))
            self.robot.arm_move_6(joints_0, 1000)


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))
    ctrl_arm = HandCtrlArm()
    fps = FPS()
    while capture.isOpened():
        ret, frame = capture.read()
        fps.update_fps()
        action = cv.waitKey(1) & 0xFF
        frame = ctrl_arm.process(frame)
        if action == ord('q'):
            break
        fps.show_fps(frame)
        cv.imshow('frame', frame)
    capture.release()
    cv.destroyAllWindows()
```