

mediapipe course

1. Introduction to mediapipe

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 ordinary hardware;
- Build once, deploy anywhere: Unified solution for Android, iOS, desktop/cloud, web and IoT;
- Ready-to-use solutions: cutting-edge ML solutions that demonstrate the full capabilities of the framework;
- Free and open source: frameworks and solutions under Apache2.0, fully extensible and customizable.

2. Use

2.1. Program running

It is best to log in to the Mechanical Dog desktop through vnc for operation. Mechanical Dog vnc is used. Reference: 14. Radar Mapping Navigation\6. ROS2 Environment Entity Mechanical Dog Status Acquisition\ROS2 Environment Acquisition of Mechanical Dog Real Joint Data.pdf

Source code path reference,

```
#pi4
/home/pi/cartographer_ws2/src/yahboom_mediapipe/yahboom_mediapipe
#pi5
/root/yahboomcar_ws/src/yahboom_mediapipe/yahboom_mediapipe
```

Enter the following command in the mechanical dog terminal, **(The running instructions for Raspberry Pi 4 and Raspberry Pi 5 are the same, and Raspberry Pi 5 needs to enter the docker terminal).**

```
cd /home/pi/cartographer_ws2/
source install/setup.bash
```

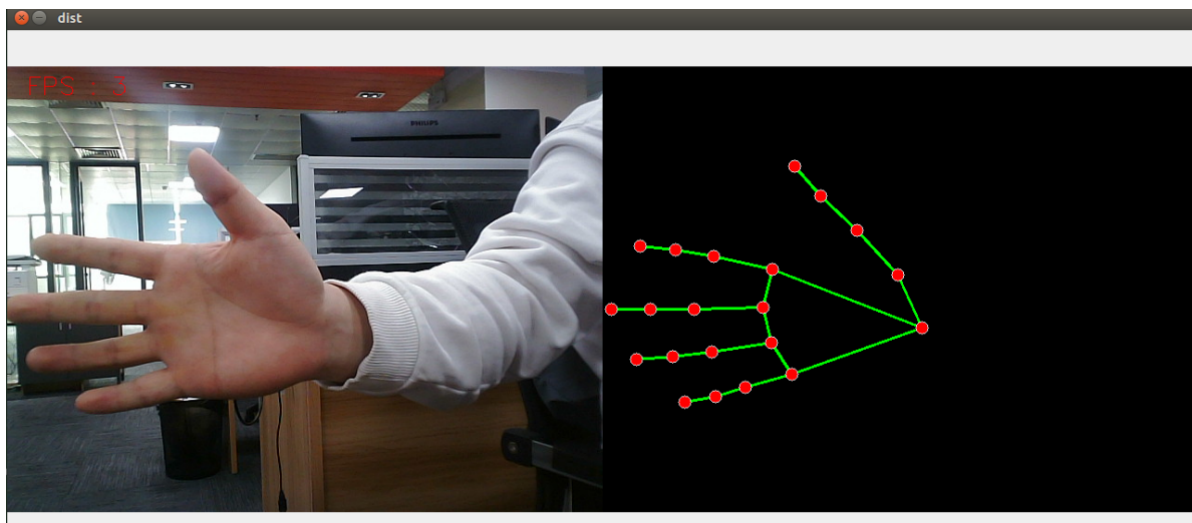
```
#handedetection
ros2 run yahboom_mediapipe 01_HandDetector
# Attitude detection
ros2 run yahboom_mediapipe 02_PoseDetector
# Overall detection
ros2 run yahboom_mediapipe 03_Holistic
# Facial detection
ros2 run yahboom_mediapipe 04_FaceMesh
# Face recognition
This tutorial identifies facial eye features and requires pointing the camera at
the face for recognition.
ros2 run yahboom_mediapipe 05_FaceEyeDetection
```

If the following error occurs

```
[WARN:0@3.549] global cap_v4l.cpp:982 open VIDEOIO(V4L2:/dev/video0): can't
open camera by index
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
[ERROR:0@3.649] global obsensor_uvc_stream_channel.cpp:156 getStreamChannelGroup
Camera index out of range
capture get FPS: 0.0
```

Then plug and unplug the camera again.

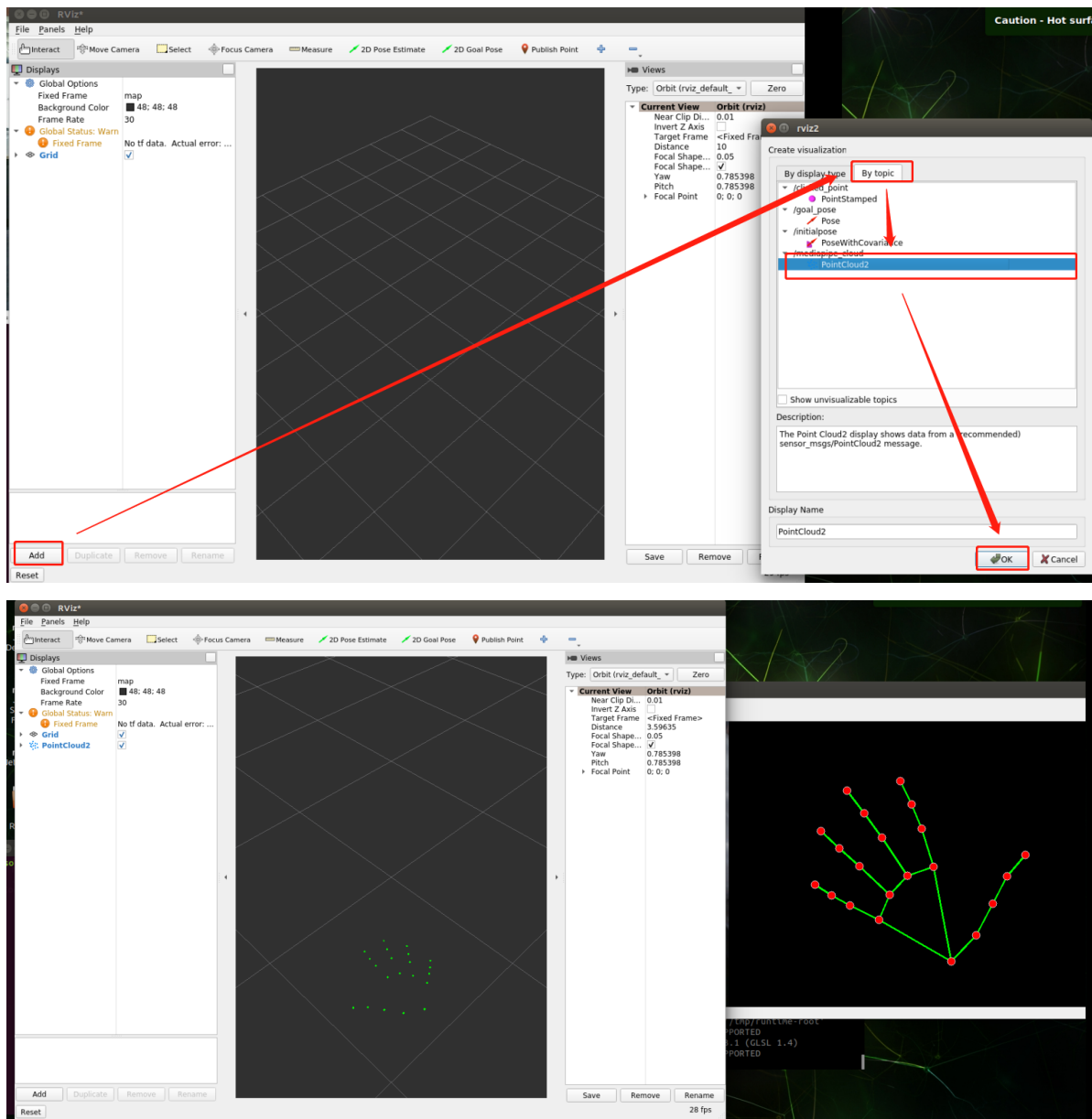
Taking hand detection as an example, the running screenshot is as follows



In addition, you can also view point cloud data and enter it in the robot terminal. **(The running instructions of Raspberry Pi 4 and Raspberry Pi 5 are the same. Raspberry Pi 5 needs to enter the docker terminal).**

```
cd /home/pi/cartographer_ws2/
source install/setup.bash
#Run the point cloud publishing program
ros2 run yahboom_point pub_point
#Open rviz to view point clouds
rviz2
```

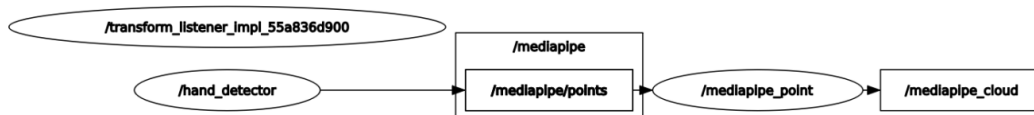
Follow the steps below to add a point cloud topic in rviz,



Above is a point cloud image of the program running hand detection, which is 01_HandDetector. Point cloud viewing only supports demos 01-05.

You can view the node topic communication graph through rqt_graph.

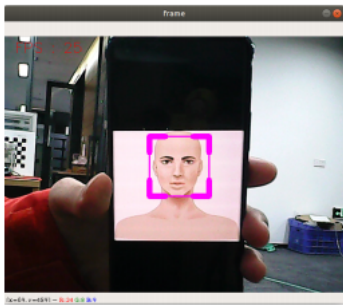
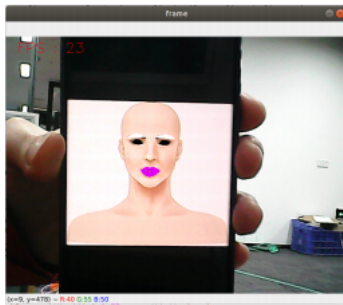
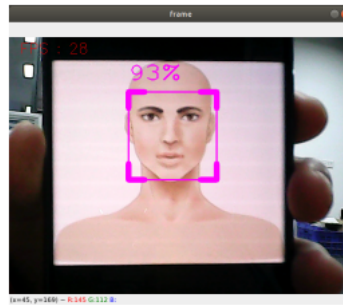
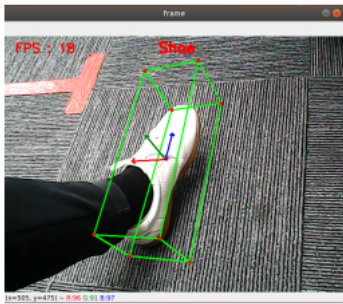
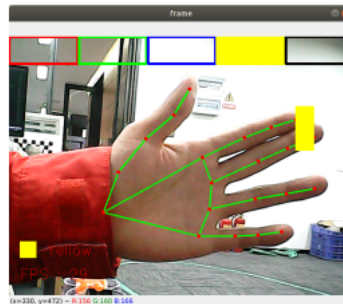
```
ros2 run rqt_graph rqt_graph
```

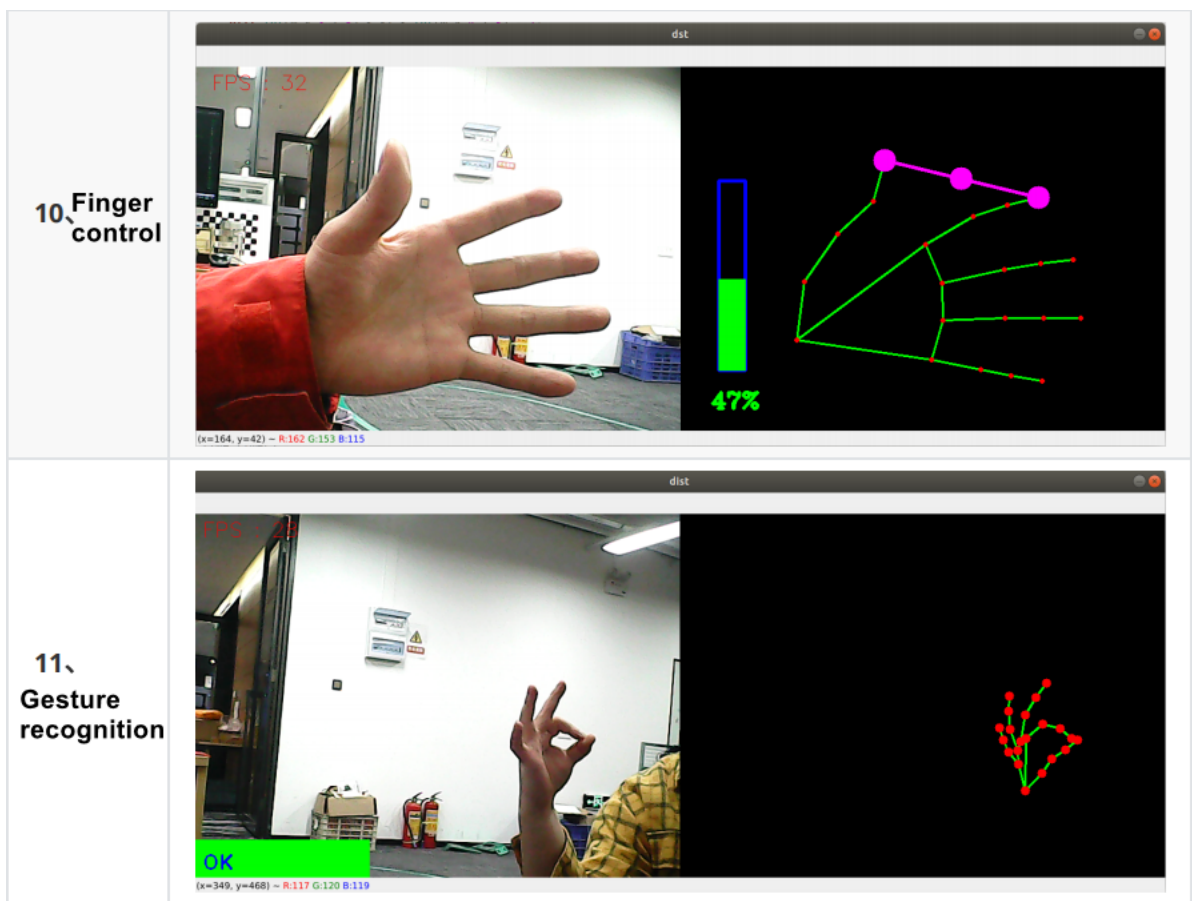


2.2. Other fun ways to play

Enter in the mechanical dog terminal, (**The running instructions of Raspberry Pi 4 and Raspberry Pi 5 are the same, Raspberry Pi 5 needs to enter the docker terminal**).

```
cd /home/pi/cartographer_ws2/src/yahboom_mediapipe/yahboom_mediapipe
#facespecialeffects
python3 06_FaceLandmarks.py
#facedetection
python3 07_FaceDetection.py
#Three-dimensional object recognition (press the f/F key to switch the recognized model)
python3 08_Objectron.py
#brush
python3 09_VirtualPaint.py
#finger control
python3 10_HandCtrl.py
#gesture recognition
python3 11_GestureRecognition.py
```

05、Face recognition	06、Face effects	07、Face detection
		
08、3D object recognition		09、Brush
		

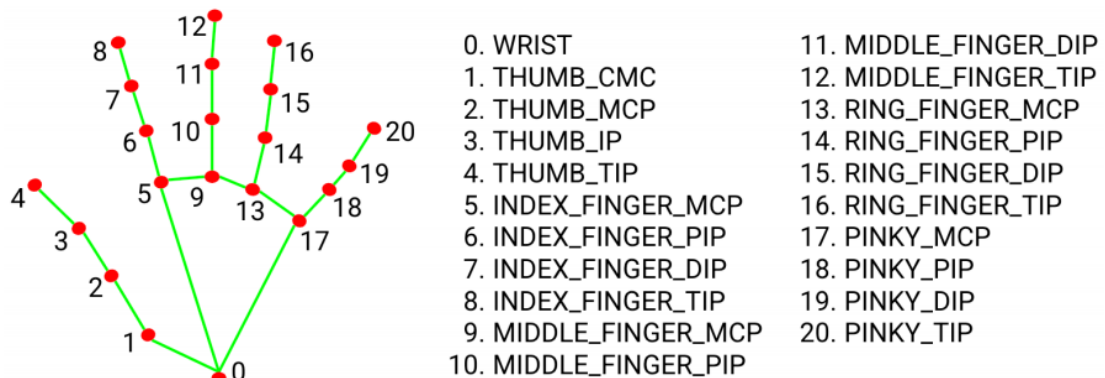


3. Mediapipe Hands

MediaPipe Hands is a high-fidelity hand and finger tracking solution. It uses machine learning (ML) to infer the 3D coordinates of 21 hands from a frame.

After hand detection on the entire image, the 21 3D hand joint coordinates in the detected hand area are accurately positioned through regression based on the hand mark model, that is, direct coordinate prediction. The model learns consistent internal hand pose representations and is robust even to partially visible hands and self-occlusion.

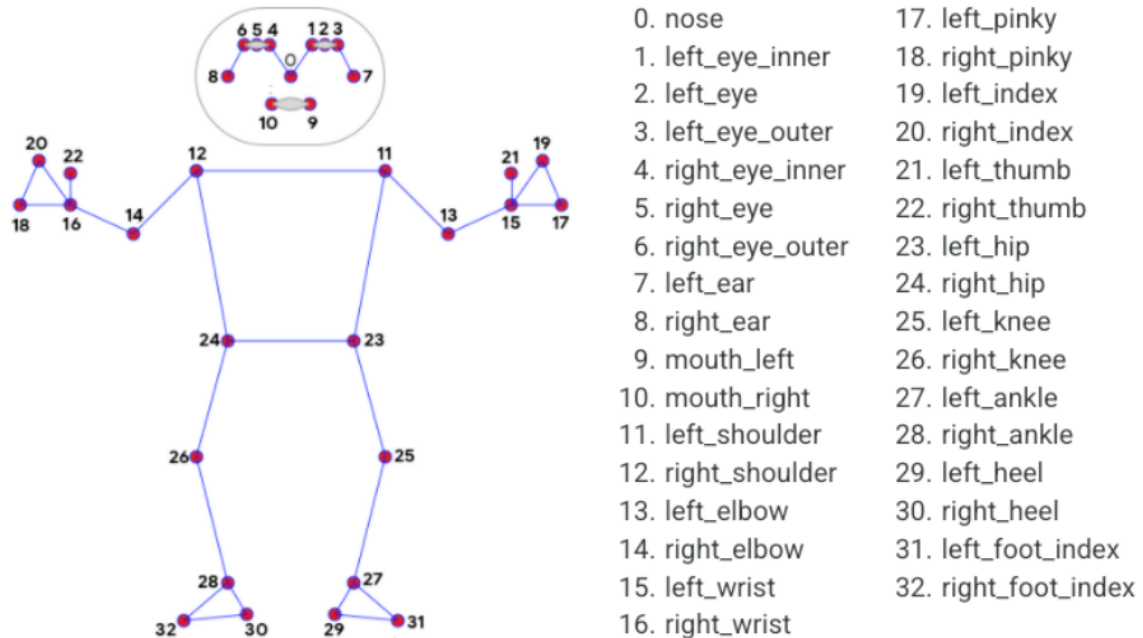
In order to obtain ground truth data, about 30K real-world images were manually annotated with 21 3D coordinates as shown below (Z value is obtained from the image depth map, if there is a Z value for each corresponding coordinate). To better cover possible hand poses and provide additional supervision on the nature of the hand geometry, high-quality synthetic hand models in various backgrounds are also drawn and mapped to corresponding 3D coordinates.



4. Mediapipe Pose

MediaPipe Pose is an ML solution for high-fidelity body pose tracking, leveraging BlazePose research to infer 33 3D coordinates and full-body background segmentation masks from RGB video frames, which is also used for ML Kit Pose The detection API provides the power.

The landmark model in MediaPipe poses predicts the positions of 33 pose coordinates (see figure below).



5. dlib

The corresponding case is face special effects.

DLIB is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real-world problems. It is widely used by industry and academia in fields such as robotics, embedded devices, mobile phones, and large-scale high-performance computing environments. The dlib library uses 68 points to mark important parts of the face, such as 18-22 points marking the right eyebrow, and 51-68 points marking the mouth. Use the `get_frontal_face_detector` module of the dlib library to detect faces, and use the `shape_predictor_68_face_landmarks.dat` feature data to predict face feature values.