2.Human pose estimation

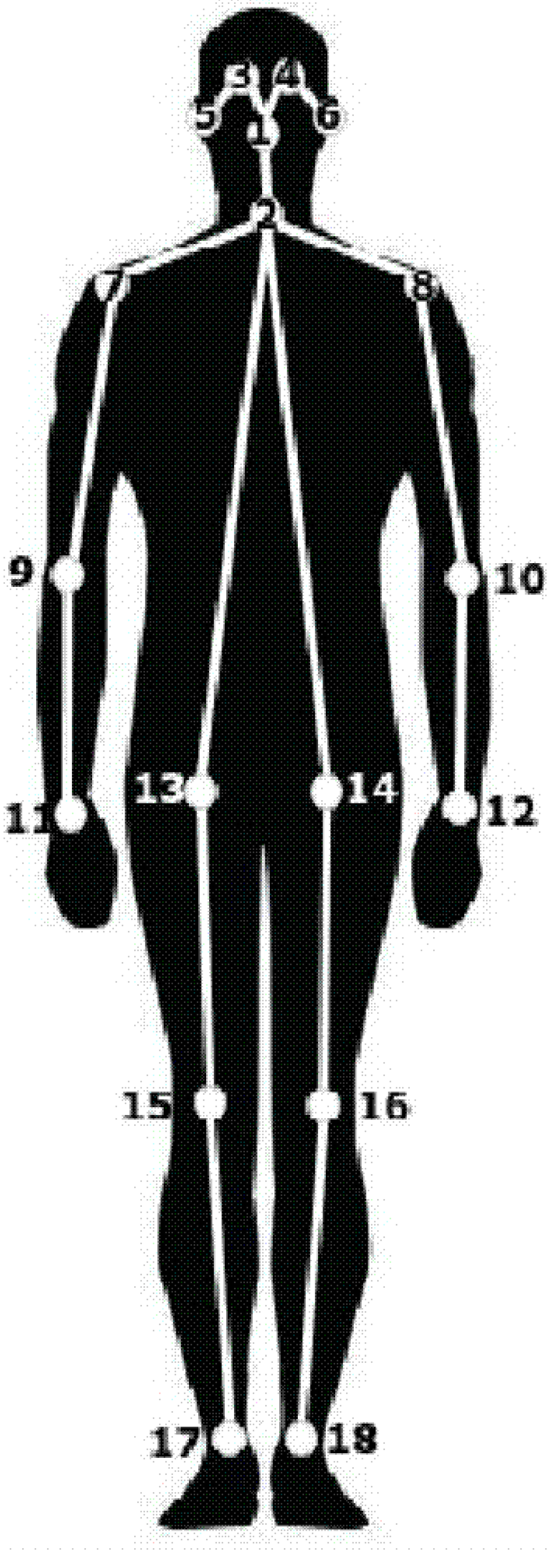
Orin board users can directly run the program by opening the terminal and entering the tutorial command. Jetson-Nano motherboard users need to first enter a Docker container and then enter the tutorial command in the Docker container to start the program.

Source code path:

```
#Jetson-Nano users need to enter the Docker container to view this.
~/dofbot_pro_ws/src/dofbot_pro_vision/dofbot_pro_vision
```

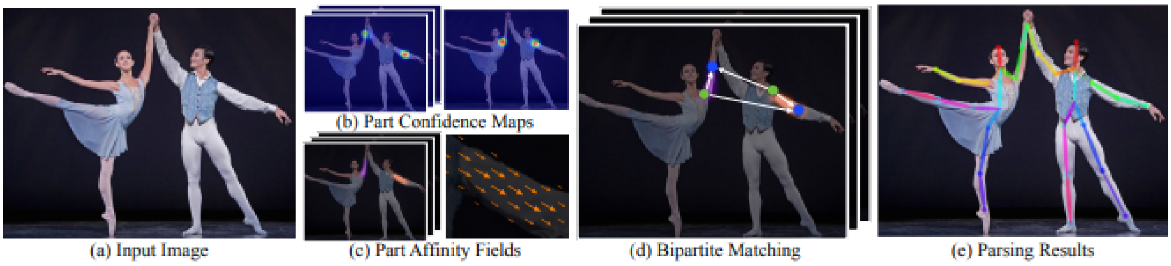
2.1 Overview

Human Posture Estimation estimates human posture by correctly connecting the detected key points of the human body in the picture. Key points of the human body usually correspond to joints with a certain degree of freedom on the human body, such as the neck, shoulders, elbows, wrists, waist, knees, ankles,



1	鼻子
2	脖子
3	左眼
4	右眼
5	左耳
6	右耳
7	左肩
8	右肩
9	左肘
10	右肘
11	左腕
12	右腕
13	左髋
14	右髋
15	左膝
16	右膝
17	左踝
18	右踝

2.2 Principle

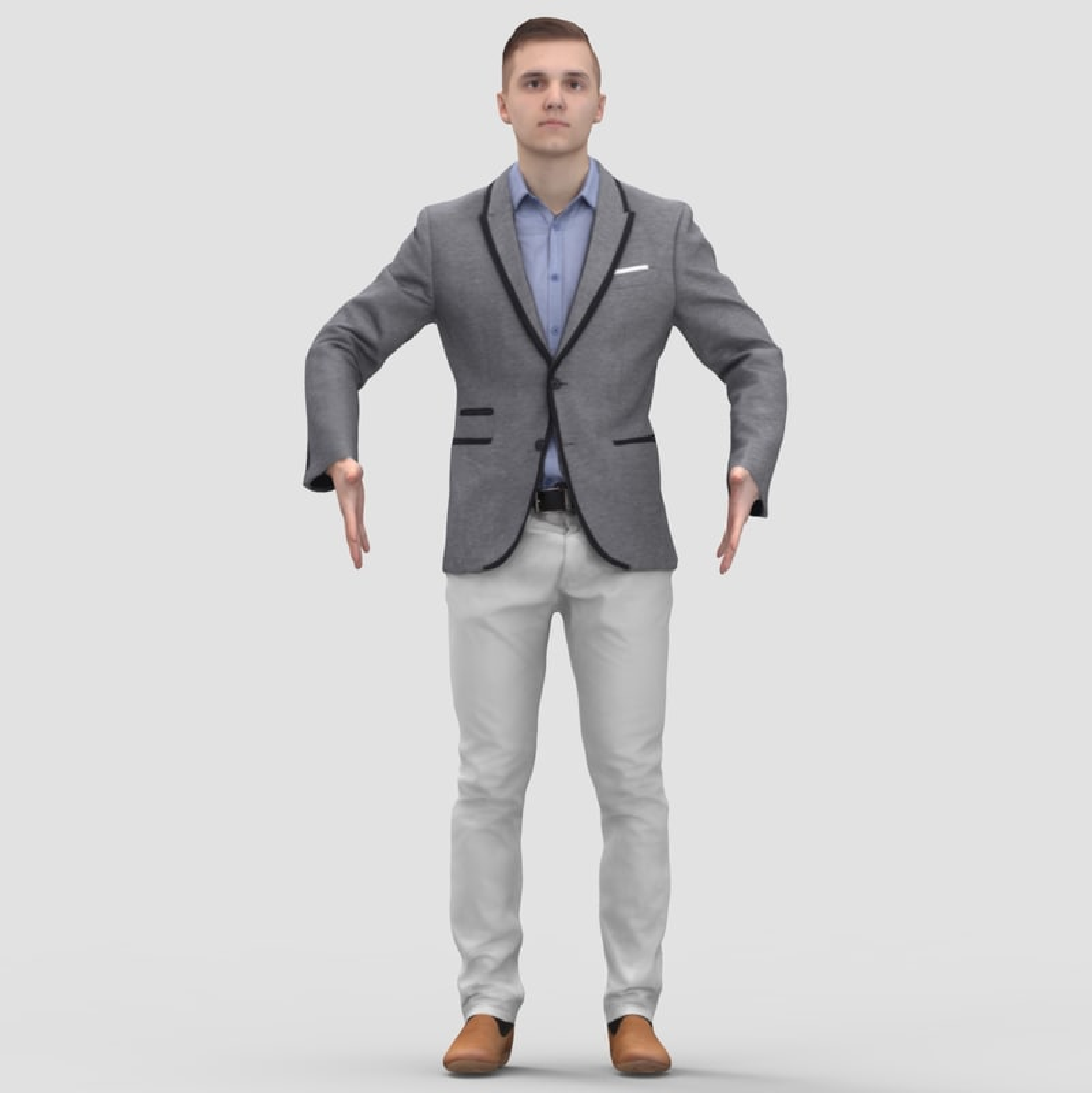


Input an image, extract features through the convolutional network, and obtain a set of feature maps, which are then divided into two branches, and the CNN network is used to extract Part Confidence Maps and Part Affinity Fields respectively;After obtaining these two pieces of information, we use Bipartite Matching in graph theory to find the Part Association and connect the joint points of the same person.Due to the vector nature of PAF itself, the generated pair matching is very accurate and is eventually merged into the overall skeleton of a person;Finally, find Multi-Person Parsing based on PAFs—>convert the Multi-person parsing problem into a graphs problem—>Hungarian Algorithm (Hungarian algorithm)
(The Hungarian algorithm is the most common algorithm for partial graph matching. The core of this algorithm is to find the augmenting path. It is an algorithm that uses the augmenting path to find the maximum matching of a bipartite graph.)

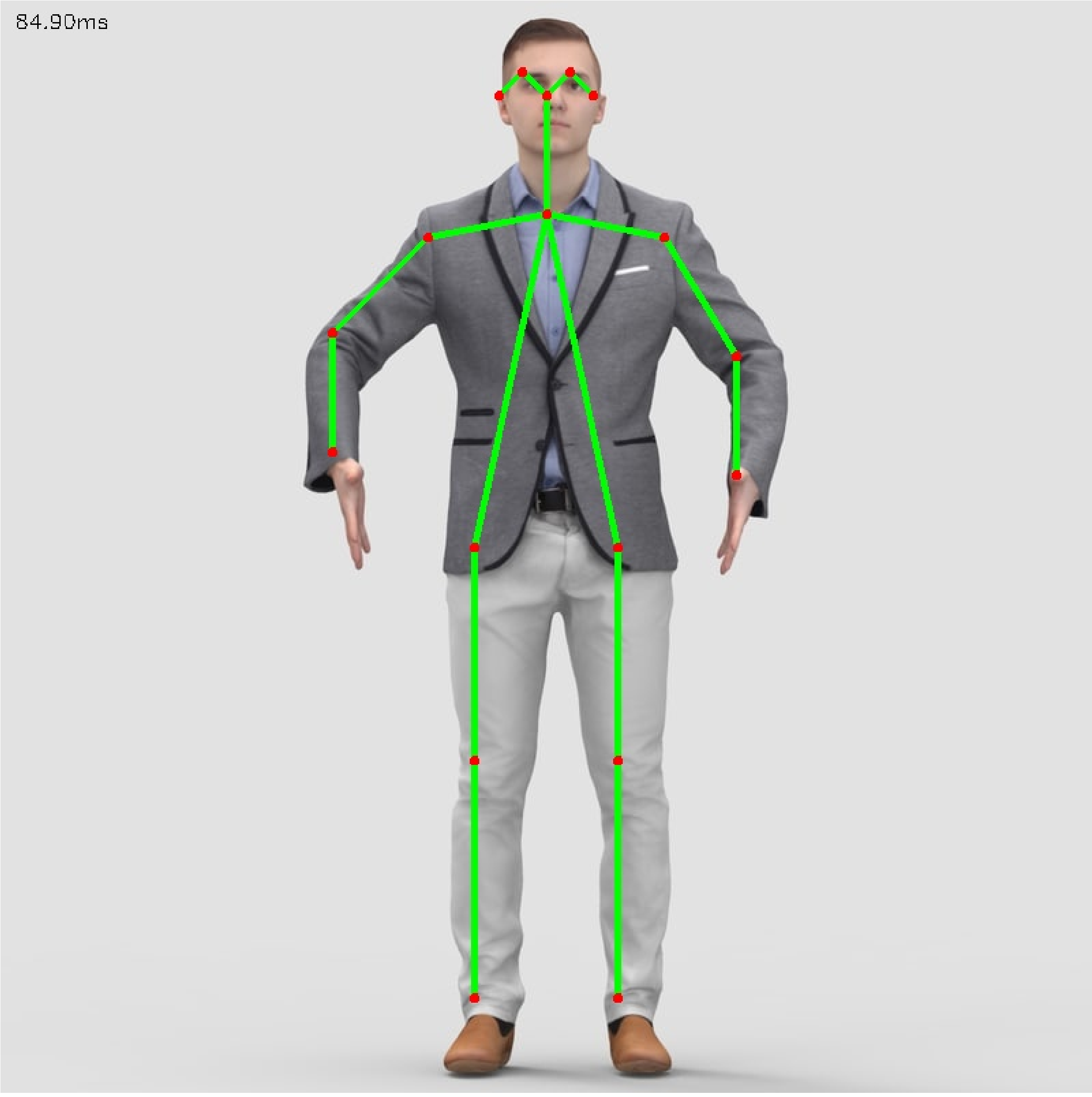
2.3 Start up

```
#If you are a Jetson-nano motherboard user, you need to start the camera node
within Docker. The startup command is `ros2 launch orbbec_camera
dabai_dcw2.launch.py`
ros2 run dofbot_pro_vision detect_pose
```

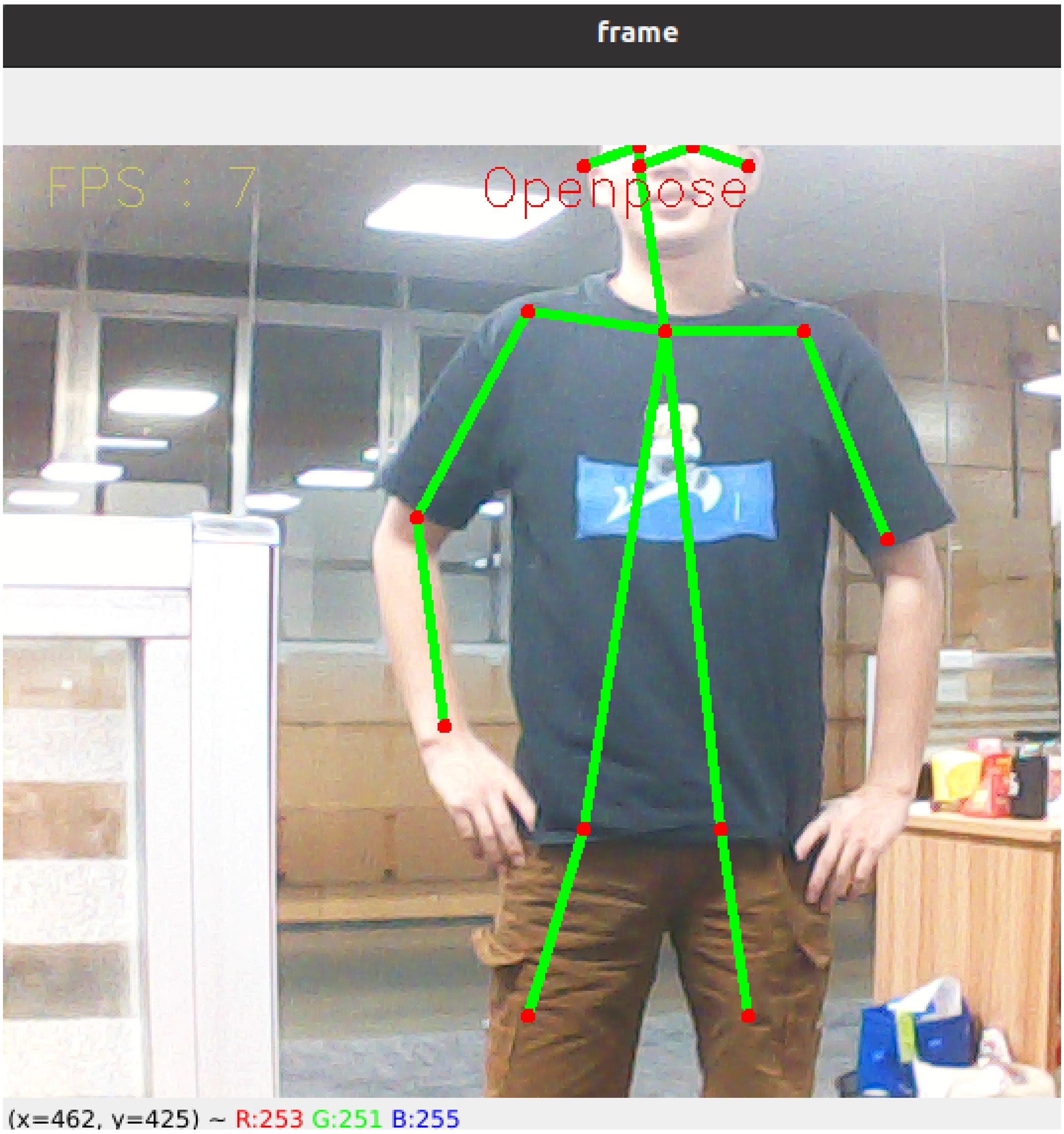
Enter image



Output picture



You can see the actual camera footage.


frame
FPS : 7
Openpose
(x=462, y=425) ~ R:253 G:251 B:255