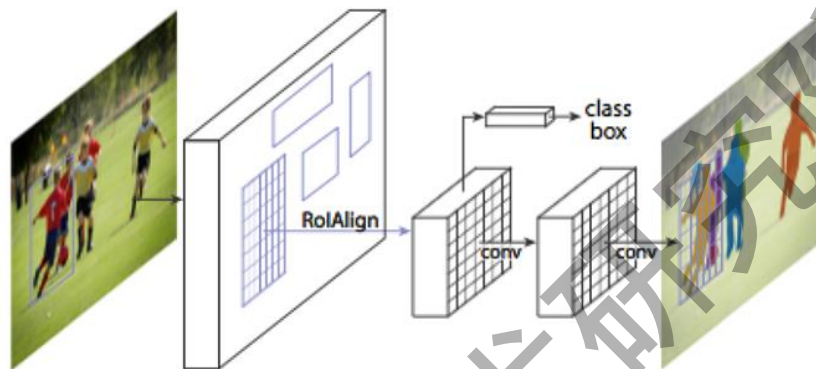


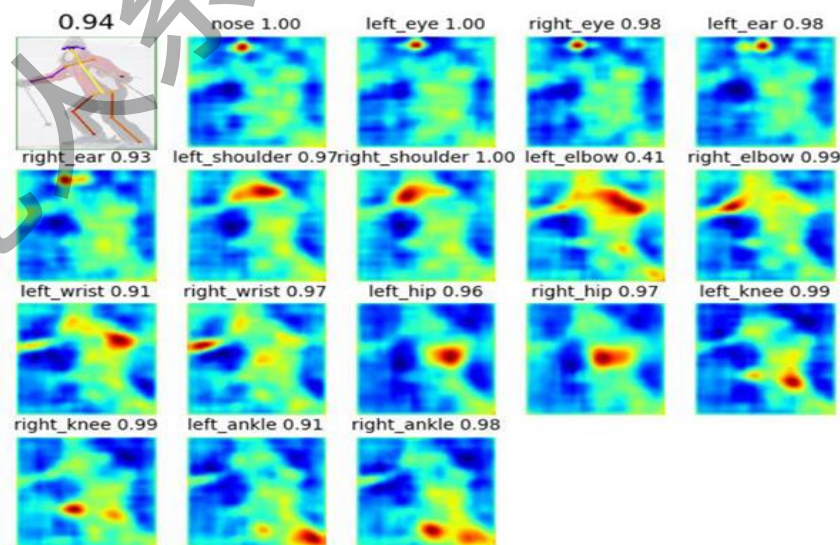
手势识别控制

本次实验手势识别基于 MediaPipe 框架实现。MediaPipe 是一款由 Google Research 开发并发布的多媒体机器学习模型应用框架，主要用于构建用于处理不同形式的感知数据的通道。MediaPipe 在手势识别技术上使用了由多个模型协同工作的机器学习通道：手掌识别模型 BlazePalm，用于识别手的整体框架和方向；Landmark 模型，作用于手掌探测器返回的裁剪图像区域，并返回 21 个手部关键点。BlazePose，用于人体姿态估计。



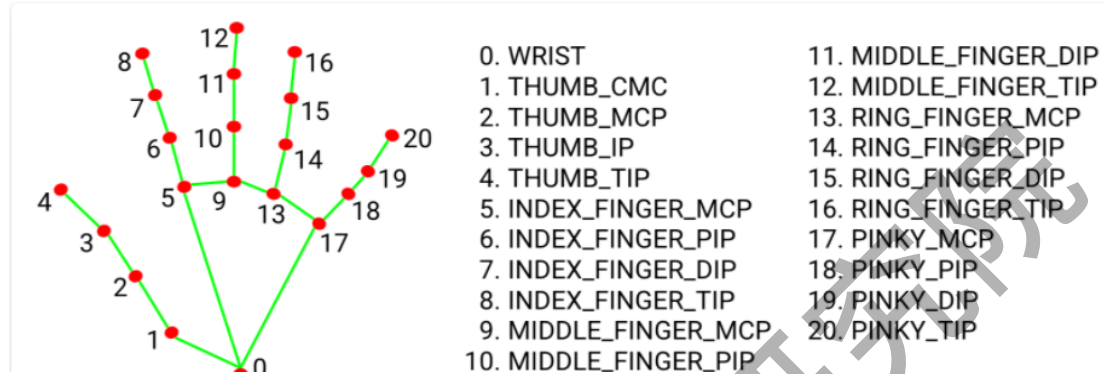
Mask R-CNN. He et al. 2017.

基于关键点检测的方法是在目标检测的框架上进行拓展实现的，上图是基于 Mask R-CNN 的人体关键点检测方法。首先由主干特征提取网络得到特征图。主干网络可以根据精度和速度的权衡选用不同的卷积神经网络结构，如 AlexNet 网络或 MobileNet 网络等。接着利用 One-stage 目标检测算法或 Two-stage 目标检测算法，提取感兴趣区域（ROI）特征图，其后，该特征图分类以及回归分支用于目标检测输出目标二维矩形框，或逐像素进行分类完成实例分割任务。如下图所示，增加的关键点检测分支最终输出 $28 \times 28 \times 17$ 的热力图（heat map），每一层热力图均预测一个关键点。



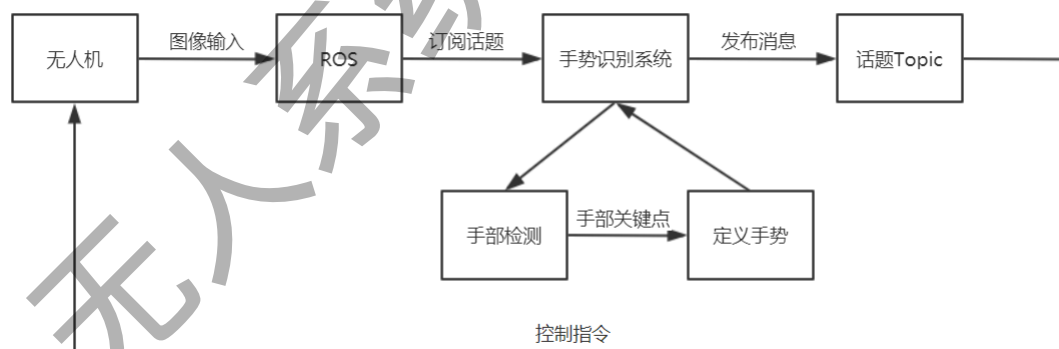
MediaPipe 中的手掌识别模型 BlazePalm，用于检测识别手部的初始位置，其主要是基于 MobileNet 主干网络结构实现，MobileNet 网络结构是一款轻量级的深度神经网络，其特点是基于深度可分离卷积来构建神经网络。传统卷积分成两步，每个卷积核与每张特征图进行

按位相成然后进行相加。深度可分离卷积可以将标准卷积分解成一个深度卷积和一个逐点卷积（ 1×1 卷积核），首先按照通道进行计算按位相乘的计算，此时通道数不改变；然后将第一步的结果，使用 1×1 的卷积核进行传统的卷积运算，此时通道数可以进行改变。深度可分离卷积跟标准卷积的区别就是精度基本不变的情况下，参数与计算量都明显减少。所以使用 **BlazePalm** 进行手势检测，其运行速度会非常快。此外该模型还可识别多种不同手掌大小，具备较大的缩放范围，还能识别被遮挡的手部，并且能通过对手臂、躯干等人体特征的检测来准确识别定位手部，大大增强了手部识别的准确率。



在对整个图像进行手掌检测后，我们随后使用手部关键点（Landmark）模型，在检测到的手部区域内通过回归对 21 个 3D 手关节坐标执行精确的关键点定位，即直接预测坐标。该模型学习一致的内部手势表征，即使对部分可见或者被遮挡的手部，也能表现出稳定的性能。为获取真实数据，手动标注了约 3 万张包含 21 个 3D 坐标的真实图像，如果存在对应的坐标，则从图像深度图中提取深度值。为涵盖更多可能出现的手势并对手部几何性质提供更多监督，谷歌还在各种背景下渲染出手部的优质合成模型，并将其映射为相应的 3D 坐标。随后，我们便可以通过使用关键点坐标之间的位置信息来定义手势。

本次实验的基本流程为：



1、安装 mediapipe 库：

```
pip install mediapipe
```

2、连接无人机到计算机，在新的终端启动无人机驱动：

```
roslaunch rmtt_driver rmtt_bringup.launch
```

3、在新的终端中，运行手势控制节点：

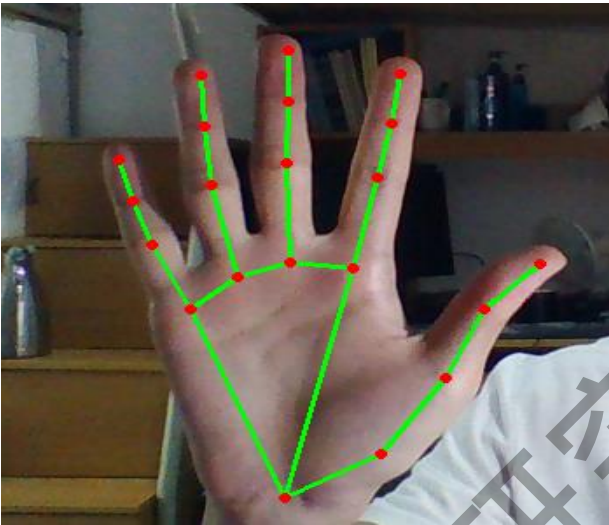
```
roslaunch rmtt_tracker gesture_control.py
```

4、在新的终端启动遥控程序

roslaunch rmtt_teleop rmtt_teleop_key.launch

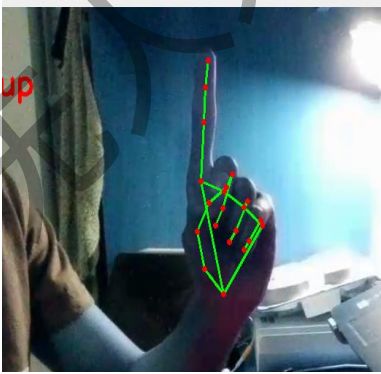
首先需要从无人机上获取所需要的视频信息。启动驱动后，无人机会将摄像头捕获的图像信息发布到话题/image_raw。

函数检测并获取手部的 21 个特征点并进行输出。

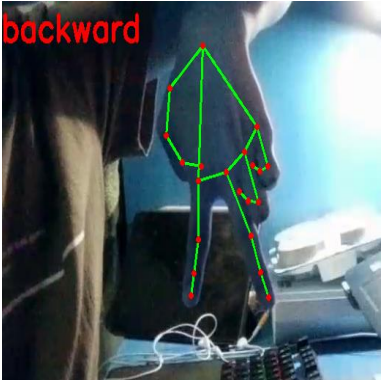

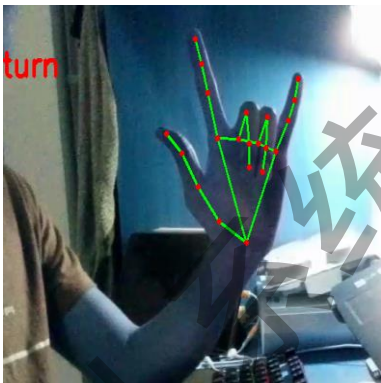
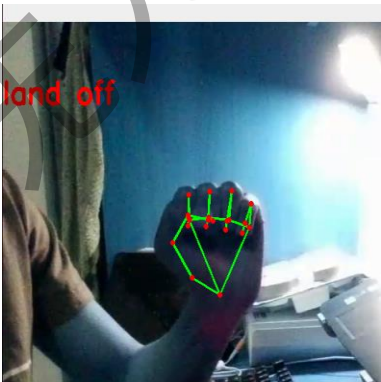


有了 21 手部关键点的坐标，我们便可以根据特征点之间的位置关系，定义手势信息。如：我们将大拇指的关键点 0 到关键点 2 连成一个向量，关键点 3 到关键点 4 连成一个向量，通过计算两个向量之间的夹角，来定义大拇指的弯曲与伸展。我们给定一个阈值，当向量夹角大于阈值时，我们认为大拇指是弯曲的，反之，则认为其实伸直的。随后，我们可以再次通过 x, y 坐标的大小关系，判断手指的指向，最终我们定义了以下的手势信息。

手势指令信息

手势	命令	判断条件
	向上飞行	大拇指弯曲 食指伸直 中指弯曲 无名指弯曲 小拇指弯曲 关键点 8 的 y 值大于 关键点 6 的 y 值

	<p>向下飞行</p>	<p>大拇指弯曲 食指伸直 中指弯曲 无名指弯曲 小拇指弯曲 关键点 8 的 y 值小于 关键点 6 的 y 值</p>
	<p>向左飞行</p>	<p>大拇指伸直 食指伸直 中指弯曲 无名指弯曲 小拇指弯曲 关键点 8 的 x 值小于 关键点 6 的 x 值</p>
	<p>向右飞行</p>	<p>大拇指伸直 食指伸直 中指弯曲 无名指弯曲 小拇指弯曲 关键点 8 的 x 值大于 关键点 6 的 x 值</p>
	<p>向前飞行</p>	<p>大拇指弯曲 食指伸直 中指伸直 无名指弯曲 小拇指弯曲 关键点 8 的 y 值大于 关键点 6 的 y 值</p>

	<p>向后飞行</p>	<p>大拇指弯曲 食指伸直 中指伸直 无名指弯曲 小拇指弯曲 关键点 8 的 y 值小于 关键点 6 的 y 值</p>
	<p>保持静止</p>	<p>大拇指伸直 食指伸直 中指伸直 无名指伸直 小拇指伸直</p>
	<p>转圈</p>	<p>大拇指伸直 食指伸直 中指弯曲 无名指弯曲 小拇指伸直</p>
	<p>降落</p>	<p>大拇指弯曲 食指弯曲 中指弯曲 无名指弯曲 小拇指弯曲</p>

检测并识别到手势，根据定义的速度指令，将其发送给无人机。控制 TT 无人机运动和降落的话题分别为/cmd_vel 和/land，同样地，我们还需要知道话题使用的消息类型，这样我们才能正确发布消息给话题。

可以通过 `rostopic info` 查询话题消息类型：

```
tianbot@ros2go:~/tello$ rostopic info /land
Type: std_msgs/Empty

Publishers:
* /rmtt_teleop_key (http://192.168.10.2:44129/)
* /gesture_6064_1622017091341 (http://192.168.10.2:36127/)

Subscribers:
* /rmtt_driver (http://192.168.10.2:45797/)
```

获取需要的信息后，根据识别到的手势信息，将控制指令以消息发布给对应的话题，进而使无人机响应动作。至此，我们完成了无人机的手势控制飞行的任务。使用 `rqt_graph` 查当前节点与话题之间的联系。

