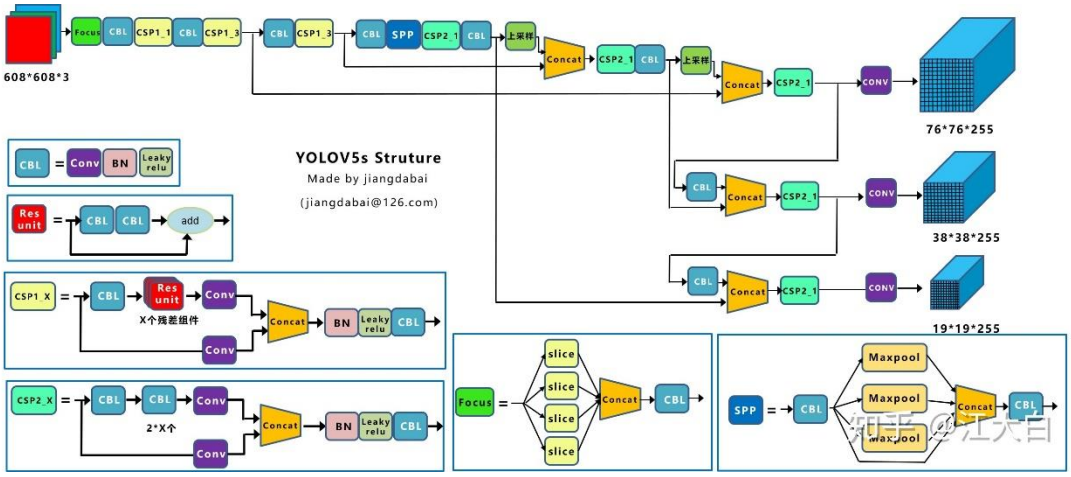
手势识别说明文档

手势处理流程主要为①boundbox 手势框检测，②关键点识别，③3d 手势优化；其中 1 和 2 通过 deeplearning 模型完成，3 通过非线性优化完成。此外，部署时还涉及到 TensorRT 适配。

1.boundbox 检测

当前版本的 boundbox 检测主要基于 yolov5（https://github.com/ultralytics/yolov5）来完成，网络结构为：



训练使用 pytorch 环境，项目工程为 gitlab 上 handtracking 项目，训练环境布置在 docker 中。

2. run docker on docker image "box_pytorch"

3.set data path in file: models/yolo/config/config.py --- config.TRAIN_PATH

4.use following command to train:

```
python models/yolo/net.py --device 0 -b {batch num} --weights models/yolo/config/yolov5s.pt
```

2、keypoints 检测

当前版本的 keypoints 检测的设计思路为基础 backbone 接 regression 层输出结果，其中基础 backbone 可用 resnet、shufflenet、mobileNet，当前稳定版本为 resnet18；regression 所做操作为将 feature map 进项 softmax 操作，转化为 heatmap，然后输出均值作为回归结果。

相关训练代码同样在 gitlab 上的 handtracking 项目，训练环境与流程 yolo 相同。

3、3d 手势生成

（1）模型参数定义

我们采用优化的方法求解表示手势的 21 个关键点位置$\mathrm{K}\sim R^{21\times 3}$，如图 1 所示，21 个关键点包括手腕点 Palm，以及每根手指上的四个关节点，颜色对应分别为：Thumb，Index, Middle, Ring, Pinky。

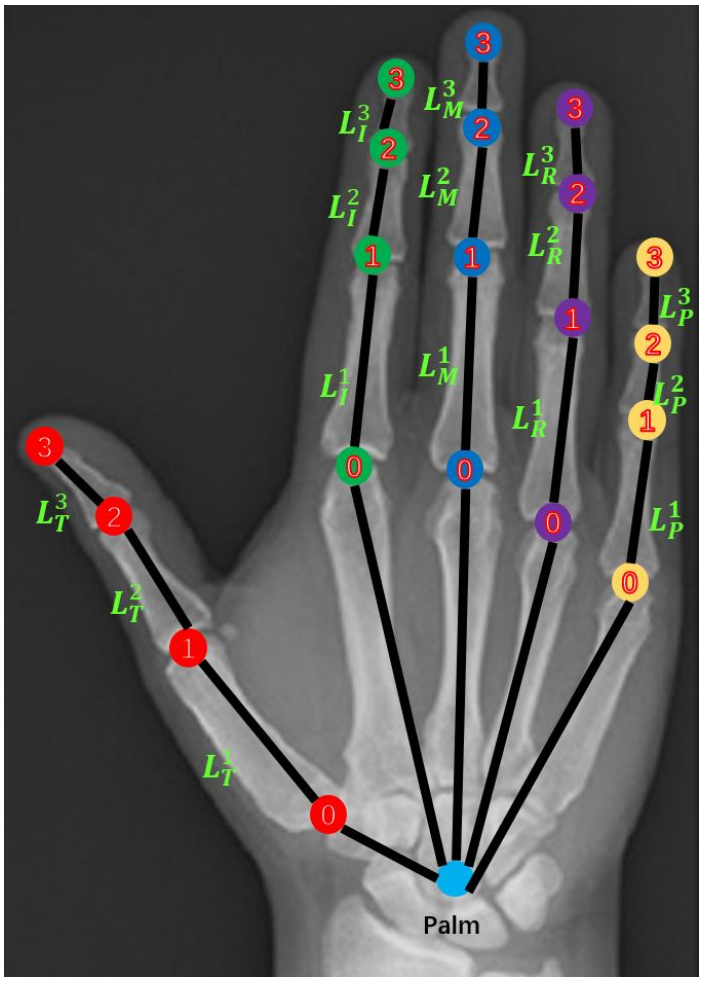


图 1、参数定义

由于观测数量有限（检测模型提供的每个关节点在图像上的 2D 位置），并考虑到各关节点之间的联系，我们通过限定部分参数的方法降低了求解变量的自由度。所限定的参数包括：(1)手掌关节点 Palm 和每个手指的第一个关节点

（T_0, I_0, M_0, R_0, P_0）的相对位置不变，即通过手掌位置K_{Palm}和手掌的旋转角度 R，可求得以上六个点的 3D 位置， 以大拇指为例：

$$K'_{P0} = RK_{P0} + K_{Palm}$$

(2)限定了每根手指相邻两关节之间的骨骼长度 L， 例如大拇指T_0和T_1之间的骨骼长度为L_T^1，限定骨骼长度之后，每根手指可通过四个角度值计算每个关节点的 3D 位置。

手指长度等参数在求解前已通过统计的方法在已有数据中获得，采用这种表示方法，可用手掌位置K_{Palm}，手掌的旋转角度 R 和 20 个（每根手指 4 个）角度值表示21个关节点的3D位置，问题求解变量的自由度由21x3降低为3+3+20。下面将介绍角度参数和 3D 位置之间的计算方法。

（2）模型参数表示方法

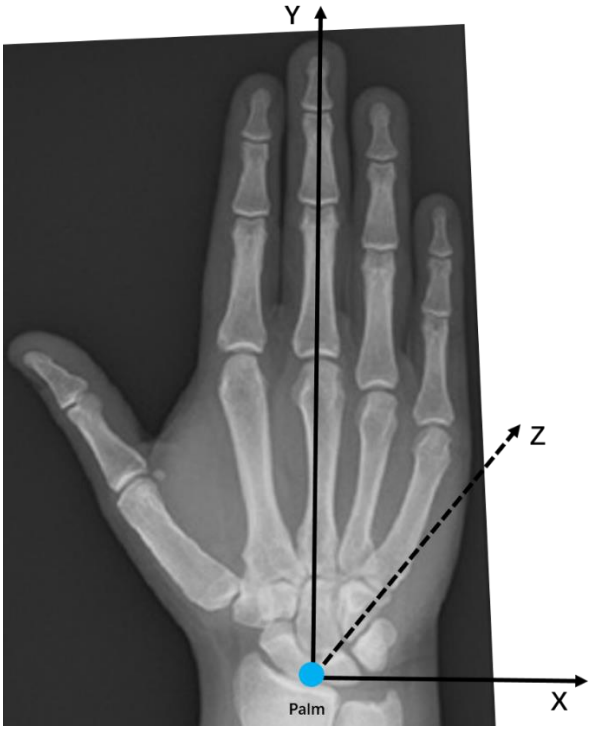


图 2、手部坐标系示意图

我们首先在手部坐标系中求解所以关节点的 3D 位置，然后通过手掌位置K_{Palm}和手掌的旋转角度R转化到世界坐标系中。手部坐标系的定义如图2所示。

在右手五指张开时，定义手掌所在平面为 XOY 平面，手腕到中指的方向为 Y 轴方向，与之垂直的小拇指侧为 X 轴方向，手心朝向为 Z 轴方向。

每根手指上包含四个关节点，第一个关节点和手掌的相对位置不变，只需求解剩余三个关节点。由于手指运动约束，排除部分个人差异，可近似认为无论手指姿态如何，一根手指的四个关节点始终在一个平面内，因此我们通过变量β定义该平面，通过变量$\theta_1,\theta_2,\theta_3$表示手指弯曲程度，以食指为例：

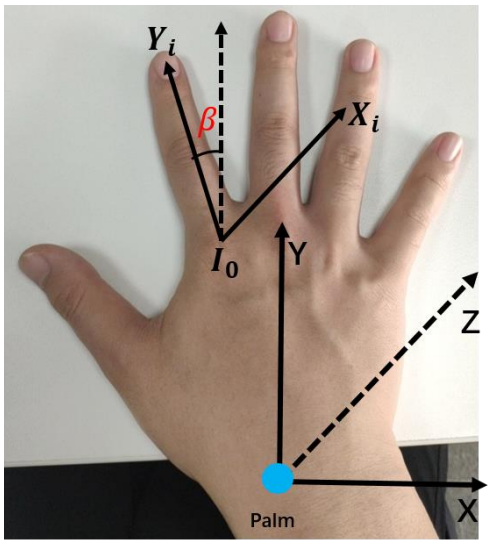


图 3、食指平面示意图

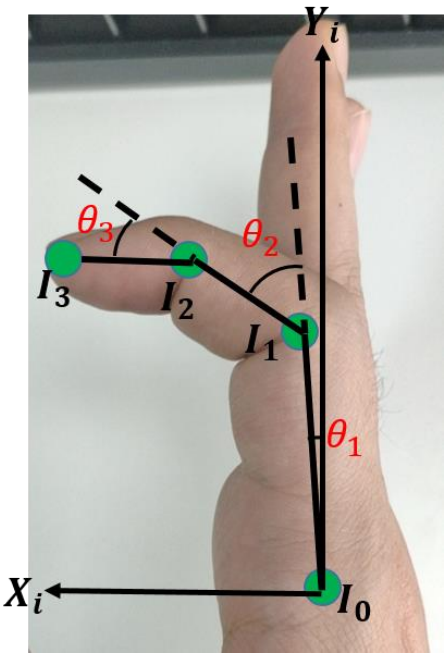


图 4、食指弯曲示意图

如图 3 所示，定义五指张开时食指伸直方向为平面内Y_i轴方向，定义手部坐标系中 Z 轴方向为平面内X_i轴方向，则定义$\boldsymbol{Y_i}$与 YOZ 平面夹角为β，则平面内两个坐标轴在手部坐标系中的定义为：

$$X_i = (0,0,1) \quad Y_i = (\cos\beta, sin\beta, 0)$$

所以平面内任意一点(x, y)，在手部坐标系内的 3D 位置为$xX_i + yY_i$。

而剩余三个关节点在平面内的 2D 坐标，可通过骨骼长度 L 和弯曲角度$\theta_1,\theta_2,\theta_3$计算获得：

$$I_1 = I_0 + L_i^1(sin\theta_1, cos\theta_1)$$

$$I_2 = I_1 + L_i^2(\sin(\theta_1 + \theta_2), cos(\theta_1 + \theta_2))$$
$$I_3 = I_2 + L_i^3(\sin(\theta_1 + \theta_2 + \theta_3), cos(\theta_1 + \theta_2 + \theta_3))$$

带入公式$xX_i + yY_i$即可获得以上三个关节点在手部坐标系中的 3D 位置。

（3）优化求解方法

我们通过优化的方法求解该问题，目标函数由三部分组成，分别为重投影误差f_p，角度范围误差f_r，和手指运动平滑误差f_s，即

$$F = f_p + f_r + f_s$$

下面分别介绍每部分误差的定义：

i.重投影误差表示求解参数与检测模型给出的结果的差异程度。上文介绍了通过手掌位置K_{Palm}，手掌的旋转角度 R 和 20 个（每根手指 4 个）角度值表示 21 个关节点 3D 位置的方法，定义所求的 3D 位置为$\mathrm{K}(K_{Palm}, R, \boldsymbol{\beta}, \boldsymbol{\theta})$，则投影到图片上的 2D 像素位置为$p(\mathrm{K}(K_{Palm}, R, \boldsymbol{\beta}, \boldsymbol{\theta}), \mathrm{C})$，C 为相机的内参和外参，$p$ 为投影函数。定义检测模型输出的 2D 像素位置为p_{model}，则f_p有如下定义：

$$f_p = \|p(\mathrm{K}(K_{Palm}, R, \boldsymbol{\beta}, \boldsymbol{\theta}), \mathrm{C}) - p_{model}\|$$

ii.角度范围误差只和 20 个角度值相关，表示求解结果的合理程度。由于手指运动范围有限，因此我们统计了所有角度的平均值，定义以下约束函数：

$$f_r = \|\boldsymbol{\beta} - \boldsymbol{\beta}_{mean}\| + \|\boldsymbol{\theta} - \boldsymbol{\theta}_{mean}\|$$

iii.手指运动平滑误差基于手指的运动速度，程序运行时连续的两帧图像之间的间隔时间极短，手指形状变换不会过快，因此定义了手指运动平滑误差f_s：

$$f_s = \|\boldsymbol{\beta} - \boldsymbol{\beta}_{last}\| + \|\boldsymbol{\theta} - \boldsymbol{\theta}_{last}\|$$

该优化问题采用 Levenberg–Marquardt algorithm 方法求解目标函数，迭代次数会结合运行平台计算能力调节。

4、模型部署

在 pc 端和 jetson 上基于 TensorRT，主要工作为将第一步框检测模型及第二步关键点模型转化为 TensorRT 支持的 engine 文件。生成 engine 文件可通过两种方式：①c++重写所有层， ②通过 trtexec 程序转化 onnx。

yolo 模型采用方法一实现，由于 yolo 中 head 部分比较复杂，以及后续需要添加阈值筛选处理，所以无法直接转为 onnx，因此采用实现更复杂的 c++重写的方法。重写方法在 adularia 工程中的 generate.cpp 实现。

Keypoints 模型采用常规网络，并无特殊，因此直接转化 onnx 然后使用 tensorrt 提供的 trtexec 程序转化即可。

详细布置见文档二。