**专业综合实践训练**

**第十一周进展报告**

# 第十周前已完成：

* 项目开题报告和相关背景调研，服务器配置，AI-EXPRESS编译环境配置（Linux）。
* 通过现有例程，能从USB摄像头读取图片，识别其中的人体骨架，并在Web端显示。
* 已经确定姿态识别使用的神经网络和训练数据
* 重新梳理了程序的交叉编译和烧录流程，确定AI-EXPRESS例程中能够获得人体17个点的骨架信息，含每个点的坐标和置信度。

# 目前面临的主要问题：

模型部署上板难度较大，整个开发板使用Xstrem和Xproto框架，涉及py，cpp，json，java，等多种编程形式，还需要熟悉.sh，cmakefiles等多种文件。现有程序框架（识别骨架），仅cpp代码就有数千行，想在此基础上进行修改，首先需要花费相当的时间熟悉已有的代码。在熟悉代码的基础上，如果想将模型部署到开发板，还需要单独编写Xstrem Method并修改json文件和cmakefiles，并且将pytorch的神经网络模型转换成onnx模型，再利用模型转换工具将onnx模型转换成二进制文件，工作量远高于预期。

我们没有找到开发板中的纯视频的接口，包括usb相机的启动程序，无法让视频心率功能上版。也没找到js中的接收代码。

想问有没有从0基础开始搭建系统的教程，很难搞清楚这么多代码和文件中哪些是必须的，哪些不是必须的，要从头搭才有可能对这个逻辑有一个很深的认知，比如主线是怎么运行的，整个逻辑是怎么撑起来的，如果只有接口的话就会非常困难。

想知道开发板开发环境中内含的opencv是不是完全的。比如haarcascade等库在有的版本安装opencv时并不原始内含，不知道这件事有没有详细的说明。

基于以上一些原因，我们暂时先不考虑原定的视频心率算法，因为重新编写一个Xstrem Method工作量过大（远高于py编程），且可行性尚不明确。我们将先集中力量熟悉现有框架，并实现姿态识别代码的部署。

# 重要的项目进度规划调整：

在这个阶段我们尝试理解xstream，xproto并编写一定的程序。三维，二维的模型都在尝试训练，二维模型在上板尝试。**整个流程中都有一定的困难，在和丁老师进行交流之后考虑做如下的调整**：

原有的正在做的上板任务，xstream和模型转换等（江柔蓝）继续执行

暂时不考虑xproto的更改，xstream以能命令行输出或以其余方式实现网络输出为节点

先前着重训练二维模型，现在开始训练三维模型，预计一周之内训练完毕

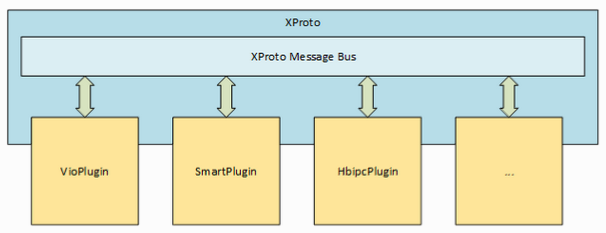
用kinect v2先实现三维的骨架检测，进行实时的姿态识别

用kinect摄像机或者usb摄像机进行视频心率识别的尝试

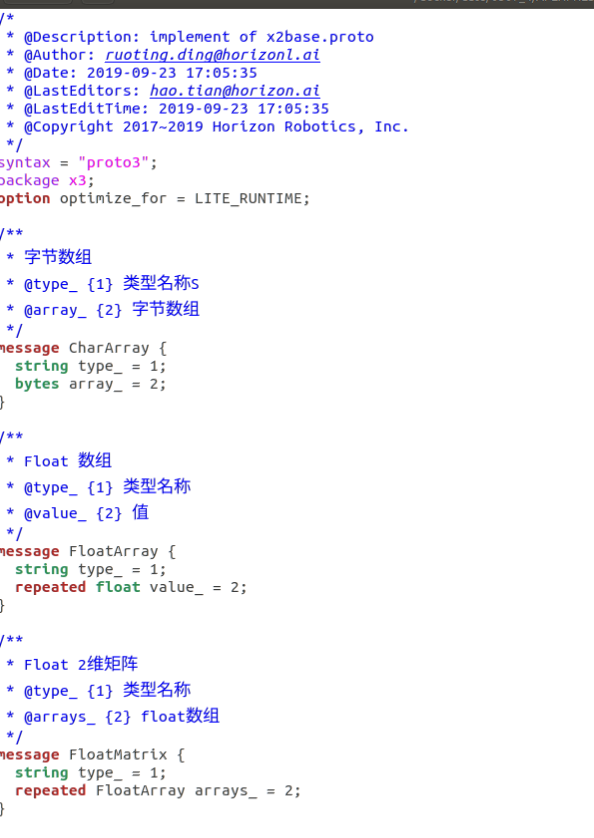
# 第十周-第十一周进度汇报：

## 李浩伟：

了解Xproto的架构。

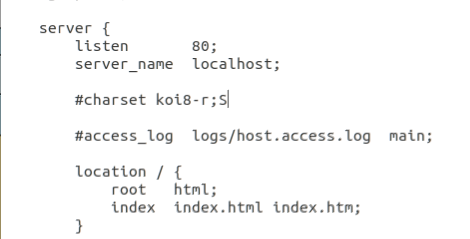


Xproto由Xproto-Framework，msgtype和plugins组。framework构成整个系统的框架，msgtype指定信息类型，这部分在.proto文件中有明确指出，在cpp文件中也有显著定义。Plugins痛过派生原始接口类实现适应于不同消息类型，作用于不同情景的不同接口。信息发送实行订阅制，发送端通过pushmsg方法向主线发送消息，通过async方法循环调用，达成连续的信息发送。通过register对总线消息进行订阅，收获别处发送的消息。中间过程中找到了显著的plugin调用，可以找到显著的结构信息的发送和处理过程。到现在为止没有找到usb摄像机的调用程序，视频信息的上发接口。



X3.proto的配置文件

网络通过nginx执行代理，开放本机80端口



上方代码源自ngnix.conf.default，为nginx代理的配置文件，在80端的ip访问中，引导进主机/html目录中，访问index.html文件。这里的Index.html网页页面为原始的页面，点击后引导进modules文件夹。

Websocketplugin上发网络中的信息内容，但是没有找到js中的接受端，js中的解编码端也没有找到。

开放了开发虚拟机的远程访问的公网端口，便于开发。将开发板做相同的端口转发时遇到了困难，ssh无法登录开发板。

准备了kinect2，准备先用kinect实现三维实时姿态识别。找到了之前准备的kinect的接口代码，准备编写连续发送的高层接口，对端口进行封装。同时考虑标定一次rgb相机和深度相机，其本身提供的映射接口的调用速度慢，三维重建需要对这一部分进行优化。

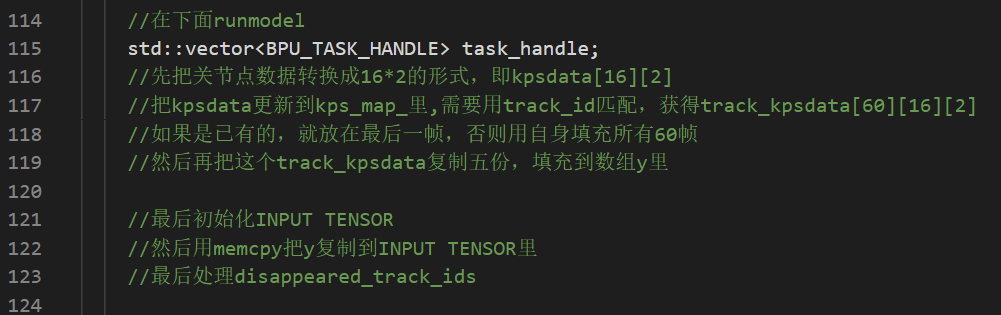
开始着手实时的视频心率检测算法和算法测试工作。

考虑到多任务的性质，要建立多线程的运算框架。

## 江柔蓝：

1. 模型转换工具链环境配置（Linux），由于地平线更新了工具链版本，部分配置需要重新进行。
2. 将onnx模型文件转换为可以部署上板的二进制文件（.bin），受到模型编译工具链的限制，转换成的bin文件可能存在一些问题（**由于不支持部分算子，需要等待地平线公司人员处理**）。转换模型使用hb\_mapper checker命令和hb\_mapper makertbin命令，需要配置.yaml文件，并且准备校准数据集（二进制文件）。
3. 了解Xstream结构，编写cpp代码实现Xstream的method，进行模型加载，关节点数据预处理，模型运行，结果输出（在控制台中）。由于这一部分代码**必须使用Xstream的框架，以及是c++语言，工作量比预想中大很多**。此外，由于预测需要利用时序信息而不是单帧数据，这里还要考虑如何储存历史数据并传输给模型；由于现有示例几乎都使用图片形式作为输入，而我们的模型输入并非图片，目前的代码有多处不确定能否正常运行的地方。

在这一部分中，我用接近四天的时间浏览了整个工程的代码架构，以及尤其是【加载模型-数据预处理-运行模型-输出数据后处理】的示例（source /common/xstream /framework/method），以及Behavior Method 的全部代码。最终决定在Behavior Method的基础上修改，从BehaviorEvent类下派生DetectEvent类，并重写Init函数和RunSingleFrame函数，在Init中添加加载模型和从json文件中读取参数的代码，在RunSingleFrame中添加数据预处理-运行模型-输出数据后处理的代码（见下）



其中初始化input\_tensor和output\_tensor时，必须保证tensor的参数与模型的输入输出匹配。

由于不清楚cmakefile的具体调用关系，这里就直接进行编译，一个一个对着Error找bug。

目前已经能确保代码通过编译，也基本确定这一思路的可行性，不过暂时没有能够正确运行的模型，所以只能拿一个简单的“假模型”进行测试。

1. 考虑到最后要将内容输出到html，本周还测试了在cpp内的读写文件操作。

## 李子涵：

网络训练和模型转换（转换成onnx文件）