实习工作总结

本次实习共七周，主要做了以下方面的工作:主机及TX2的相关环境配置；调试Moonergy及编写其c接口；阅读TX2中tegra\_multimedia\_api中视频解码相关程序并编写程序文档；基于TX2中DeepStream SDK源码编写视频解码程序；基于ffmpeg编写其视频解码程序；在TX2上测试基于tegra\_ multimedia ,opencv, gstreamer, ffmpeg的视频解码性能。

1. **主机及**TX2**的相关环境配置**

主机环境配置参考Sunergy的环境配置。

TX2刷机使用JetPack3.2.1包，下载地址：

<https://developer.nvidia.com/embedded/jetpack>

刷机流程参考地址：

<https://blog.csdn.net/weixin_42749767/article/details/81200537>

配置好的环境包括TensorRT 4.0,cuDNN 7.1.5,CUDA 9.0

安装deepsreamer SDK,参考地址：

<https://mp.weixin.qq.com/s?__biz=MjM5NTE3Nzk4MQ==&mid=2651233122&idx=2&sn=1432ec9259b21f15e4ed2a605663e336&chksm=bd0e4ff08a79c6e6de57fccbb2252b5153616d923ced475601b914499df0ffdfbf32bedef610&mpshare=1&scene=23&srcid=06293P0XHaVQ7nmjCLPfRdGR#rd>

配置测试Sunergy，在TX2上可以直接使用PySunergy实现目标检测，使用YoloV3,网络加载时间为10.7秒，检测一张图片速度为0.9秒

刷机注意事项：在刷机过程中保证网络的通畅，TX2板子必须通过网线连接且与主机处于同一个网段。

1. **调试Moonergy及编写其c接口**

Moonergy是基于ffmpeg实现视频硬解码接口程序，在配置调试中由于所需的库和依赖比较多，在调试中出现了很多问题，主要有boost的安装，boost安装使用源码安装，安装版本为boost\_1\_60\_0；ffmpeg的安装也采用源码安装，修改其Makefile中相关头文件和依赖使其和安装的都对应上；opencv安装版本为3.0；如果还使用IPP加速的话还需要安装IPP。

C接口代码的编写主要修改之前的moonergy.h和moonergy.cpp文件，整理其接口函数，使其简介易懂，然后编写主程序测试其接口函数是否正确。在编写接口函数文档过程中，需要注意接口代码的易读性，使用结构体将相关函数参数放在里面，使得函数参数尽量的简洁些；规范使用函数注释。

1. **阅读TX2中tegra\_multimedia\_api中视频解码相关程序并编写程序文档**

tegra\_multimedia\_api在TX2刷机完成之后已自动下载完成，其主要用于多媒体中的视频的编解码；视频格式转换；视频格式之间的尺度变换；直接从摄像头获取数据进行编码等方面；还使用了cuda进行GPU加速，更好的利用了TX2的GPU性能。这里主要调试其中视频解码的相关源码。tegra\_multimedia\_api解码程序主要基于V4L2底层框架，使用TX2的NVDEC解码芯片对H264或H265编码格式的视频进行硬解码，经过多次测试，目前只能解码解封装后的.h264码流文件，原始MP4文件需要先经过解封装成.h264格式才能读入。相关程序文档见tegra\_multimedia\_decode。

1. **基于TX2中DeepStream SDK源码编写视频解码程序**

NVIDIA新推出的DeepStream SDK借助NVIDIA Tesla GPU的硬件特性（包括卓越的解码性能，通过降低精度进行高速推理以及低功耗），帮助开发人员快速构建高效、高性能的视频分析应用程序。DeepStream SDK提供包含对输入视频流解码、预处理和推理的模块，所有模块都经过精细调整，以提供最大的帧吞吐量。解码模块接受以H.264、H.265和MPEG-4等格式编码的视频，并对它们进行解码以渲染NV12颜色格式的原始帧。预处理阶段将颜色格式从NV12转换为BGR，并将帧的大小调整为神经网络所需的分辨率。色彩空间转换和缩放使用自定义CUDA内核和NPP库函数的组合。推理模块利用一个由TensorRT导入和执行的神经网络批量处理这些帧。TensorRT优化的网络具备高吞吐量和低延迟以及一流的功率效率。DeepStream目前支持基于Caffe的网络。这里主要研究其解码模块，通过研究发现其解码功能实现是基于gstreamer，实现的思路是基于gstreamer建立解码的pipeline,在上面加入解码所需要的各种元件,其中主要解码元件使用omxh264dec可以利用tx2的NVDEC实现硬解码，然后将解码得到的视频数据格式转为opencv的Mat数据格式，并通过opencv测试解码速度和显示。基于gstreamer不光可以实现视频文件的解码，还可以实现usb摄像头，rtsp摄像头，tx2板卡摄像头的解码。相关源码见tx2\_gstreamer\_decoder。

1. **基于ffmpeg编写其视频解码程序并在tx2上测试**

为了测试tx2是否可以使用ffmpeg解码的gpu硬件加速功能，编写了相关程序。FFMPEG解码流程：

1. 注册所有容器格式和CODEC: av\_register\_all()；
2. 打开文件: av\_open\_input\_file()；
3. 从文件中提取流信息: av\_find\_stream\_info()；
4. 穷举所有的流，查找其中种类为CODEC\_TYPE\_VIDEO；
5. 查找对应的解码器: avcodec\_find\_decoder()；
6. 打开编解码器: avcodec\_open()；
7. 为解码帧分配内存: avcodec\_alloc\_frame()；
8. 不停地从码流中提取出帧数据: av\_read\_frame()；
9. 判断帧的类型，对于视频帧调用: avcodec\_decode\_video()；
10. 解码完后，释放解码器: avcodec\_close()；
11. 关闭输入文件: avformat\_close\_input\_file()

测试表明，tx2不支持ffmpeg的gpu加速模块，在没有gpu加速模块的情况下，虽然一路解码速度可以达到230，但cpu使用率达70左右了。相关程序源码见tx2\_ffmpeg\_decoder。

为了测试通过ffmpeg将.mp4文件解封装成.h264裸流文件，然后再通过tegra\_ multimedia硬解码，编写了基于ffmpeg的解封装程序，测试了解封装时间，很快，一帧大约为0.40ms,间接的实现了视频硬解码。为了更好的利用其tx2性能，还需要将解封装与解码融合。相应的解封装程序见tx2\_ffmpeg\_mp4toh264

1. **在TX2上测试基于opencv, tegra\_multimedia\_api，gstreamer, ffmpeg的视频解码性能。**

为了测试不同解码方式在tx2的不同表现，分别测试了四种不同的解码方式，根据测试结果，tegra\_multimedia\_api的解码速度最快。在四种方式中tegra\_multimedia\_api和gstreamer都使用了tx2的NVDEC解码芯片，实现了硬解码，通过比较，虽然gstreamer解码中使用了omxh264dec的硬件加速模块，在测试中也显示利用了TX2的NVDEC解码芯片，但其解码速度并不高且在四路时cpu基本已满；在tx2上tegra\_multimedia\_api的解码性能最好，但现在只能间接实现普通mp4格式的视频文件解码，后期还需要将mp4解封装与H264码流文件解码融合，才能实现普通视频的快速解码，详细测试结果见test\_report。

1. **一点实习心得**

一个多月的时间很快，虽然很短，但还是学到了不少东西。我的工作主要是基于tx2研究相关解码算法，使其可以更高效率的利用tx2的硬件资源达到更快的解码速率且更少的使用CPU。之前这方面研究比较少，基本都是直接调用opencv实现视频的读取，没有更深入的研究相关解码方式，这次通过对不同解码方式的研究和测试，对视频解码有了更深的认识。通过这次实习，锻炼了代码阅读能力与编程能力，也认识到了很多不足，工具的使用，编程的规范性，研究问题的方式方法等基本功能力还有待加强。