1 Introduction

Object segmentation is a technique developed for medical imaging and self-driving industries. We are using this technique for our video special effects where the segmented object areas are treated as alpha channel for video editing. The latest deep learning-based method provides real-time high precision to segmentation, which just fits to our applications. However, some engineering issues arise after our initial experiments on existing DL models: 1) some cases cause segmentation to fail; 2) how to deploy existing models into our existing platform.

2 Objectives

The two objectives of this project is to solve the 2 issues raised in section 1: 1) consistently segment objects in special cases without misinterpretation; if misinterpretation occurs, there must be a mitigation solution; 2) deploy the model under our platform for our specific application. Each of the two objectives can be fulfilled by one group of students.

3 Segmentation & Alpha Channel

A typical segmentation of an object is shown in Figure 3-1 where a well-known model DeepLabV3 is used to label a car as an alpha channel. After over-lapping or masking with the alpha channel, we can hide all backgrounds.

Car label or alpha overlap

Figure 3-1: segmentation of car from a picture

However, when the car is surrounded by white smoke as shown in Figure 3-2, the car can't be recognized at all.

Car label or alpha overlap

Figure 3-2: segmentation of car surrounded by smoke

The resultant label or alpha channel in Figure 3-1 is quite useful in video editing system. However, unlike self-driving applications, if a video object fails to be continuously segmented, the video editing system will produce quite different results, very bad effects to human visual enjoyment. Therefore, we need to find out either 1) better deep learning models or 2) re-training of the existing models or fine tuning of the existing models.

4 Deployment

A segmentation model must be deployed under our platform built as shown in Figure 4-1 where the frontend hardware receives various video signals and sent to CPU/GPU for processing, and then transmits the processed video signals out. The CPU module controls both frontend and backend, it can also receive and distribute video signals from/to both frontend and backend.

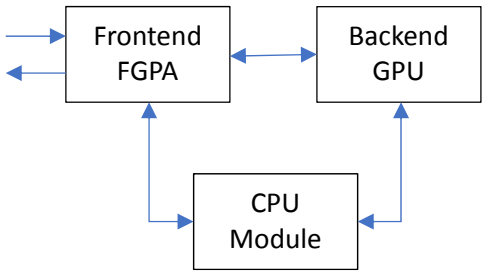

Figure 4-1: System platform

Our platform runs under c++/cuda system. Any Pytorch (Python torch) model must be converted into TensorRT or directly rewritten into libtorch (c++ torch) so that the model can seamlessly integrate into our c++/cuda system.

Project supervisors:

Dr. Yu Liu, Architect on video/audio processing algorithm and modelling in Ross Switcher Department, Ross Video Limited.

Dr. Jiying Zhao, School of Electrical Engineering and Computer Science, University of Ottawa.