Project Report: Measurement and Analysis of Object Detection with Parallel Streaming on Mobile Devices

吴家行
2020213991

Abstract

With the rapid development of computer science, Augmented Reality (AR) applications become more and more popular in our daily life. AR technology enhances our understanding of the real world by superimposing computer-generated virtual images on real-time captured video frames. However, due to the limited computing power of mobile devices, it's difficult to deploy computation intensive tasks such as object detection on AR devices. Therefore, it is necessary to hand over the computation intensive tasks to the edge server. This kind of system based on edge computing becomes mainstream object detection method on AR devices. This work focus on the performance analysis of object detection with parallel streaming on mobile devices. The contribution of this work includes the following three parts. First, we measure the compression efficiency of different encoding methods, and the video quality corresponding to different compression rates. Second, we implement an object detection system based on parallel streaming. Third, we measure the detection accuracy and end-to-end delay with different sliding methods.

随着计算机科学的飞速发展，增强现实（AR）在我们的日常生活中越来越受欢迎。AR 技术将计算机生成的虚拟图像叠加在实时捕获的视频帧上，增强了我们对现实世界的理解。但是，由于移动设备的计算能力有限，难以在 AR 设备上部署目标检测之类的计算密集型任务。因此需要将计算密集型任务迁移到边缘服务器。这种基于边端协同的目标检测系统成为在移动 AR 设备上主流的目标检测方法。该课程项目对移动设备上的目标检测进行测量分析。这项工作的贡献包括以下三个部分，首先，我们测量不同编码方法的压缩效率，以及对应不同压缩率的视频质量。其次，我们实现了一个基于并行流处理的目标检测系统。最后，我们使用不同的 sliding 方法来测量目标检测的精度和端到端的延迟。实验结果表明，sliding 方式为 6x1 时，我们的目标检测系统可以达到最佳的检测精度和延迟。

1 Introduction

With the rapid development of computer technology, applications such as Augmented Reality (AR) and Virtual Reality (VR) are becoming more and more popular in daily lives. As shown in Fig. 1, AR technology augments the real-time captured video images by superimposing virtual images generated by computers on people's understanding of the real world.

Due to the limited computing power of mobile devices, it is difficult to deploy computationally intensive tasks such as object detection on AR devices, so the computationally intensive tasks need to be handed over to edge servers for processing. The collaboration mode has become the mainstream object detection method in AR scenarios. However, the existing end-to-end object detection systems are not suitable for processing high-resolution videos with many target objects. The reason is that the files of high-resolution images are large, which affects network transmission, and the tracking accuracy and real-time performance of multiple objects are poor. In case of limited mobile computing resources, by designing a reasonable end-to-end collaboration strategy, the video stream processing is parallelized to reduce the end-to-end delay time, thereby improving the performance of mobile end HD video object detection and greatly improving the user experience.

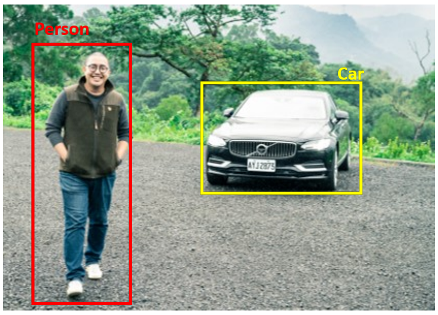

Figure 1: Object Detection on AR applications.

2 SYSTEM DESIGN

Our system architecture of this work is shown in Fig. 2, which is divided into mobile device part and edge server part. The mobile device part is mainly composed of a video stream processor (Streamer), a video encoder (Encoder), an object tracker (Tracker), a socket (Socket) and a controller (Controller). The edge server is mainly composed of a video decoder (Decoder), an object detector (Detector) and a socket (Socket).

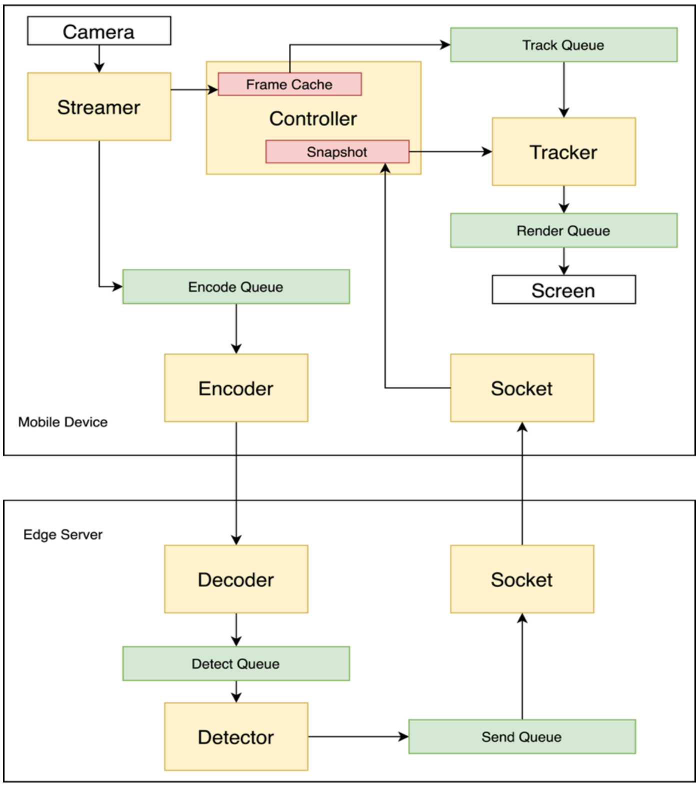

Figure 2: System Overview.

In the mobile device part, the Streamer reads the video stream captured by the camera in real time, and caches the parsed image information and video attributes (such as video length, video width, frame rate, etc.) of each video frame into the Controller. At the same time, the video frame is put into the encoding queue and waits for being encoded. The Encoder obtains video frames from the encoding queue. If the current encoder is idle, the obtained video frames are used as key frames, and the encoded frames are sent to the edge server. It is addressed that the frame is divided into multiple slides. When one slide is encoded completely, it is directly sent to the edge server. At the same time, next slide is being encoded. So, to a certain extent, the transmit process and the encoding process is simultaneous.

The Encoder of the mobile device and the Decoder of the edge server communicate through the sockets. The Encoder is developed based on the kvazaar (https://github.com/ultravideo/kvazaar.) open source library, and the Decoder is developed based on the openHEVC (https://github.com/OpenHEVC/openHEVC) open source library. After the Decoder of the edge server receives the compressed frame, it decodes the frame, and puts the decoded frame into the detection queue to wait for object detection. The Detector obtains video frames from the detection queue, uses the Faster-RCNN detection model to detect all objects in the video frame, and saves the detected bounding box and classification information to the send queue. The Socket is used to return the test results in the sending queue to the mobile device in order.

The Socket of the mobile device is used to receive the detection results from the edge server and save it to the temporary snapshot (Snapshot) in the Controller. The Tracker will monitor the temporary snapshot. If the temporary snapshot has not changed, it means that the new snapshot has not been received. The Tracker will continue to track based on the tracking result of the previous frame. If there is a change in the temporary snapshot, it means that a new target detection result has been received. The Tracker component will perform object tracking based on the new detection result. In this tracking update process, the object tracking model of KCF is used in a parallel manner to effectively improve the accuracy and speed of tracking update. Tracker will save the tracked target bounding box and classification information of each frame to the render queue, and finally the results in the render queue are rendered in real time on the currently playing video frame, show it to users and clean up the video frame buffer in time.

3 PARALLEL STREAMING

To reduce the streaming latency, we propose to use a multithreaded streaming technique to encode the key frame in multiple encoding threads. This is because almost all GPUs support more than one video encoding session and each encoding session generates its own encoding stream independently.

Fig. 3 is an example of frame sliding with 6-slide mode. The key frame is divided into six slides vertically. The total six slides are encoded into six video streams using six encoders respectively. Accordingly, the edge server uses six decoders to decode the six video streams, composites the six image slides into a full frame, and then start to run object detection on the whole frame.

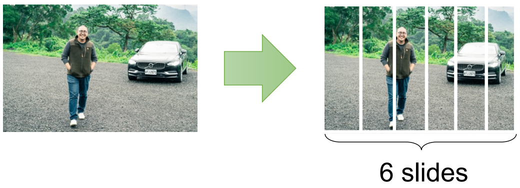

Figure 3: Frame Sliding with 6-slide mode.

Fig. 4 illustrates how the parallel streaming mechanism can reduce the streaming latency, in comparison to a baseline approach. Four main tasks (encoding, transmission, decoding, and inference) are represented with rectangles in different colors. The length of each rectangle is the rough execution time of the corresponding task. In the baseline approach, the four tasks execute sequentially. The streaming latency, as shown in Figure 4, is the total execution time of encoding, transmitting and decoding the whole frame.

With simultaneous transmitting and encoding (second in Figure 4), the transmitting of the first slide starts immediately after it is encoded and in parallel with the encoding of the second slide. As a result, this two-way parallel approach reduces the streaming latency, i.e., the extra streaming latency after the whole frame is encoded, by 1/2.

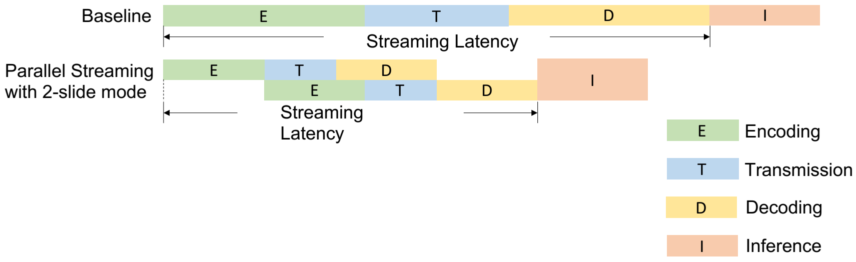

Figure 4: Parallel Streaming.

The parallel streaming technique may be further extended to 8-way or even 16-way for more parallelisms. However, we do not recommend doing so because 1) it requires more simultaneous encoding sessions that may not be possible on many GPUs as we will show later, 2) it reduces the performance of motion estimation in H.264 and thus leads to a lower compression rate, and 3) it makes the implementation more complex.

4 MEASUREMENT AND ANALYSIS

This work focus on measuring and analyzing the relationship between different sliding mode and object detection performance on mobile devices. Because the streaming has a deep relationship with the encode and decode part. So, we also compare the compress efficiency with two different encode methods, HEVC and JPEG. After choosing a suitable encode method, we have an evaluation on different sliding mode.

3.1 Hardware Configuration

As shown in Fig. 5, We implement the edge-server modules on a CentOS 7.0 server. It is equipped with two 8-core Intel Xeon CPU E5-2560 v4 CPUs, two GTX 2080ti GPUs and 256GB memory. The edge-server modules consist of a video decoder and a remote object detector. We implement the mobile-device modules on an iPhone 11, with the A13 Bionic chip embedded with a four-core GPU. The mobile-device modules consist of a video streamer, a video encoder, and an object tracker. Most modules on the device side are implemented in C++ 17 for easy deployment on different platforms such as iOS Frameworks and Android NDK.

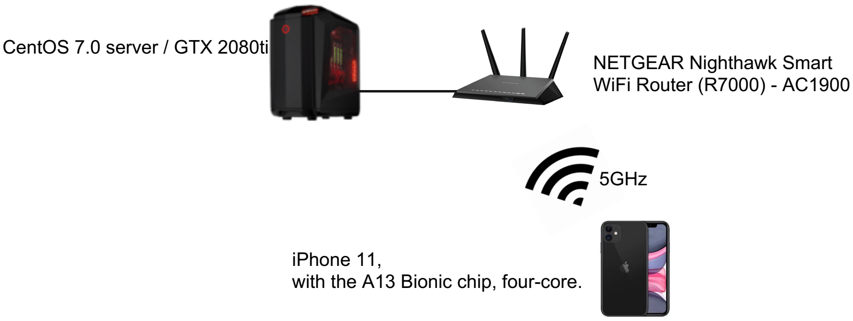
CentOS 7.0 server / GTX 2080ti
NETGEAR Nighthawk Smart
WiFi Router (R7000) - AC1900
5GHz
iPhone 11,
with the A13 Bionic chip, four-core.

Figure 5: Hardware Configuration

3.2 SSIM Analysis

First of all, we want to choose one of the two encoding methods of HEVC and JPEG by comparing the compression efficiency. HEVC is a video encoding method, which uses the "quantization parameter (qp)" to control the encoding quality. The range of the qp is [0,50]. The smaller the qp, the higher the encoding quality. JPEG is an image encoding method, which uses the "quality" to control the encoding quality. The range of the quality is [0,100]. The greater the quality, the higher the encoding quality.

In order to control the variables and ensure the same quality of the picture, we choose structural similarity (SSIM) as the metrics to measure the video quality. Just as the reference paper claims, SSIM>0.98 can be considered as high-revolution video, so we measure different quality for the two encoding methods. As shown in Fig. 6, the SSIM value corresponding to the parameter is found to be similar with the qp=10 of HEVC and the Quality=85 of JPEG, and both of them exceed 0.98. Therefore, these two parameters are used to compare the compression efficiency.

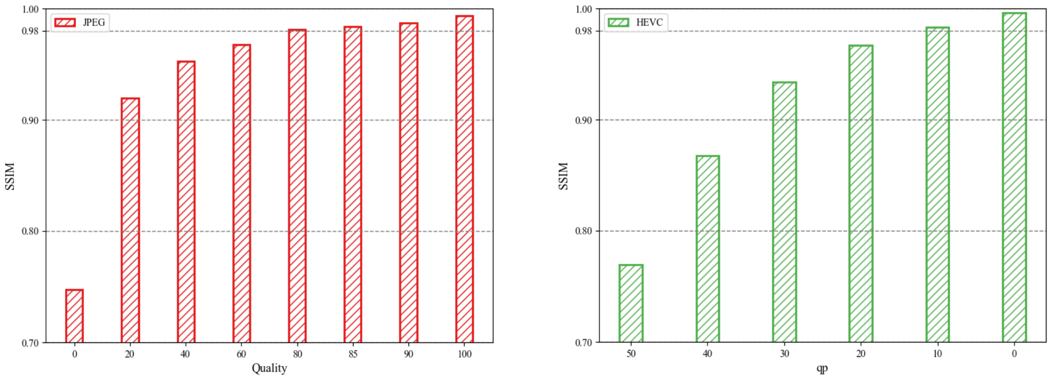

Figure 6: SSIM with different compressed rate in JPEG and HEVC.

3.3 Encoding with HEVC and JPEG

After the quality parameters are given, the compressed file size and compression time of the two encoding methods are compared. As shown in Fig. 7, it can be found that the compressed file size of HEVC is smaller than JPEG, whose P frames are smaller than smaller, and the compression time of JPEG is shorter than JPEG.

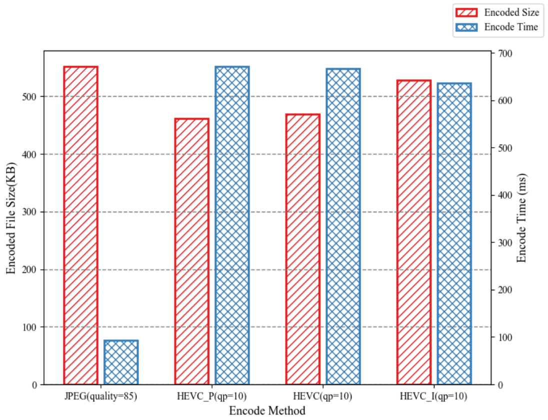

Figure 7: Encoded file size and encode time with JPEG and HEVC.

Each method has its own advantages and disadvantages, but because the frames do not need to be uploaded for each frame, which means only the key frames need to be uploaded, the encoding time is not the main factor that affects the system performance. Besides, the experiment uses the soft-encoding method. If we use the hard-encoding methods, the encoding speed will be faster. Therefore, in order to save bandwidth, we choose HEVC encoding as our encoding method.

3.4 Performance with Different Sliding Mode

After determining the encoding method, experiments are carried out. On a video sequence, different slides (1,2,4,6,8,10-slide) are used to measure the accuracy of object detection and end-to-end latency. The experiment results are shown in Fig. 8. At the beginning, as the number of slides increases, the accuracy is obviously improving and the end-to-end latency drops significantly.

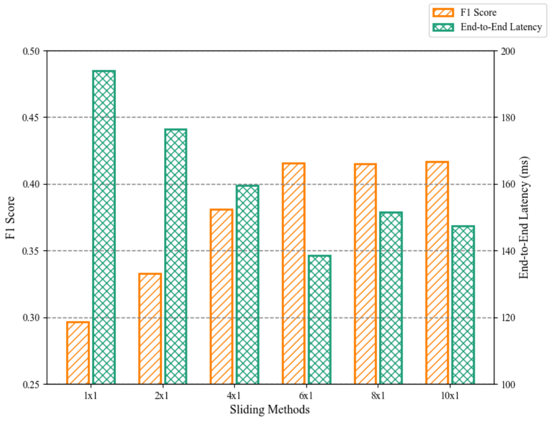

Figure 8: Accuracy and end-to-end latency with different sliding mode.

However, after 6x1 sliding mode, the accuracy no longer changes, and the end-to-end latency also begins to fluctuate. This situation also happens to verify that the reference paper does not say that the more slides, the better. This is because parallel resources on the mobile device are limited, too many slides will also affect the performance of the encoder. So the end-to-end latency and the accuracy will also be affected.

4 CONCLUTION

In this work, an object detection system is proposed to achieve both low-latency and high-quality requirements over a wireless link. First, we measure the compression efficiency of different encoding methods, and the video quality corresponding to different compression rates. Second, we implement an object detection system based on parallel streaming. Third, we measure the detection accuracy and end-to-end delay with different sliding methods. The system has the best performance with 6x1 sliding mode.