Towards Augmented Reality Training

by

Lu Sun

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Ph.D. degree in
Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

In this thesis proposal, we aim to develop an Augmented Reality (AR) framework for training scenarios. Compared with existing augmented reality applications, our proposed framework has three advantages: 1) Existing methods typically guide the trainee through a fixed series of steps, while the proposed method acts as an interactive system between the trainer and the trainee; 2) it does not need markers to estimate the environment; 3) the proposed approach is compatible with commercial equipments that have less powerful graphics capacity and various network conditions.

Three sub-goals constitute our final goal. The first sub-goal is a video object segmentation algorithm. We present a method for extracting foreground objects from video and its application to content-aware video compression. Our method uses trimaps inferred from background subtraction to represent the foreground-background relationship. The appearance of foreground and background are modeled with Radial Basis Functions initialized from the background substraction step. Finally, Graph Cuts are used to compute a binary mask. Our method is fully automatic, fast, and does not make restrictive assumptions about object motions. In experiments on standard data sets, the proposed approach achieves comparable results to state-of-the-art video object segmentation methods but our method is much faster. We also demonstrate an application of the proposed method to content-aware video compression. With this algorithm, we are able to leverage an existing 3D object pose estimation algorithm to perform a markerless environment estimation.

Given the rapid evolution of mobile devices in terms of both hardware and software, most of recent AR applications target in mobile devices, such as mobile phones and AR glasses. The second sub-goal is a hybrid remote rendering method optimized for mobile devices, which minimizes bandwidth requirement and interaction latency. To incorporate it into the AR proposed framework that targets in training scenarios, the hybrid remote rendering method is also able to support multi-client cooperation. It adopts a client-server architecture and maintains two versions of models: low-fidelity models and high-fidelity models. Low and high fidelity models differ in the number of polygons and in rendering quality. On the one hand, the low-fidelity models are stored on the client side and the local rendering capacity is leveraged to produce lower quality rendering results. On the other hand, the high-fidelity rendering results of key models are sent from the server to the client and overlaid upon the locally rendered frames. We define key models as those that are important to the application. The models that the user is interacting with can be identified from interaction information sent to the server, while the models that are important to the application are specified in advance by the application developers. We also conducted a user study to identify the effects that the proposed method has on user object recognition and quality of experience.

Finally, the last sub-goal is to combine the former two to develop an augmented reality framework. It acts as a bridge between the trainer and the trainee when they are in different geometrical locations and is able to leverage the recently emerged augmented reality glasses or mobile devices. Client-server pattern is used in this system. There are two types of clients in our system: a) trainer, b) trainee. Both of the trainer and the trainee use AR glasses or mobile devices to capture videos of the environment. Then the

videos captured are sent to the server that uses the proposed video object segmentation algorithm combined with a markerless 3D object pose estimation method to calculate the pose of all the objects of interest. Moving objects are rendered and the frames are sent to the clients and overlaid on the real scenes, i.e. any objects moved by the trainer will be rendered and sent to the trainee, and, similarly, any objects moved by the trainee are rendered and sent to the trainer. During this course, the remote trainer guide the trainee through the real time operations and collect their operations, then the trainer is able to give customized feedbacks to each trainee, which is how the learning actually takes place.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This introduction chapter is to briefly introduce the goal of my Ph.D. study: an augmented reality framework for training scenarios. First, we depict the application scenario and unveil the roadmap to the final goal, namely the sub-goals. Next, this chapter also contains the problem statement of each sub-goal. It unveils the statuses of the corresponding research fields and what we have done or will do in each research field. Then we list the main contributions that support the scientific novelty of this thesis proposal. Finanly, we conclude with an outline of the manuscript structure.

1.1 Motivation of the Problem

Augmented Reality (AR) is attracting more attention than ever before. The use of AR in numerous fields has been explored by researchers, such as games, communication, medicine, training and etc [**?**, **?**, **?**, **?**]. In this article, we discuss the use of AR in training and propose a new framework that facilitates remote training.

Currently, traditional training methods (e.g. book, audio and videos) are still the most popular methods in remote training. However, the AR system is now widely regarded as a promising platform for complex and highly demanding tasks. Gavish et. al. developed an experiment to evaluate the performance difference between AR training system and traditional training system [**?**]. By using AR system and traditional training system in a real industrial maintenance and assembly task and They argue that trainees using AR training system achieve fewer unsolved errors.

However, as pointed out by Crescenzio et al., many factors-such as cumbersome hardware, the need to put markers on the aircraft, and the need to quickly create digital content-seem to hinder its effective implementation in industry [**?**]. With the rapid progress of AR glasses and mobile devices in years, the hardware used by augmented reality applications have been simplified. Many of them use only AR glasses or mobile devices [**?**, **?**]. The development of markerless 3D object recognition approaches [**?**] reduced the use of markers in augmented reality applications. However, many augmented reality applications still do not meet the need to quickly create digital content, especially for those applications

that involve real equipment. One of the contributions of the proposed work is to enable creating digital content in real-time for trainees when they are interacting with real objects and need guidelines.

We are going to develop a training system with augmented reality hardware. Envision a scenario: A trainer is directing the trainees how to operate a machine that consists of multiple movable components. The trainer and trainees are in different locations and each of them wears augmented reality glasses. When the trainer is operating a component of the machine, the trainees will see that change in their views as a virtual component overlaid on the real scene. Vice versa, when the trainees are operating a component of the machine on their side, the trainer will see the changes from each trainee in his or her view. The change made by each trainee is demonstrated as a virtual object overlaid on the real scene. With this system, we are able to deliver better experience for remote training, since the trainees will obtain intuitive instructions, and the trainer can see the performance of the trainees in real time and give feedback immediately. This is different from traditional remote training approaches, e.g. books, audios, videos, and etc.

To achieve the training framework, we propose three building blocks:

- An on-line and fast video object segmentation algorithm to segment the components used in the training scenario
- A remote rendering framework that adapts to various network conditions and hardware capacities
- An augmented reality framework that provides real time and remote training services

The problem of the first building block can be formulated as: Given a sequence of monocular optical observations, segment the objects of interest from the background, without knowing the entire sequence. The problem of the second building block can be formulated as: Knowing the 3D models of the objects of interest, and their transformations over time, render the 3D models remotely on a high-end workstation and apply the rendering results on the clients that have less powerful graphic capacity. The problem of the third building block can be formulated as: Giving the video sequences of both the trainer's and the trainee's workplace, render the 3D models operated by the trainer on the trainee's view.

1.2 Problem Statement

1.2.1 Object Segmentation

Object segmentation is an essential part of augmented reality. It is the process of separating foreground objects from the background in a video [**?**]. A wide range of applications benefit from the progress of video object segmentation, e.g. robot-object interaction, recognition, video compression etc. A variety of methods have been proposed to address the task [**?**, **?**,

?, ?, ?]. Most methods use motion cues to initialize the object segmentation. A common method for motion detection is background subtraction with mixture of Gaussians [?, ?]. This technique models each pixel indepedently as a mixture of Gaussians but as it detects motion in each frame independently, the results lack completeness and temporal persistence.

1.2.2 Remote Rendering

Mobile applications for gaming, training, healthcase among others rely on the rapid evolution of mobile devices in terms of both hardware and software. Mobile devices offer a more intuitive interaction experience through gestures compared to PCs that mostly use traditional keyboard and mouse interfaces. However, complex 3D models demand high-end computing hardware. Compared to PCs, mobile devices have much lower processing power, limited storage and less capable rendering hardware, even in most recent high-end mobile devices [?, ?, ?, ?]. Developing mobile applications, especially 3D graphics applications, often requires simplification of the 3D models leading to a degraded rendering quality. Computationally intensive 3D graphics rendering tasks further burden the onboard battery.

One way to address the resource limitations of mobile devices is through Cloud Mobile Rendering (CMR). CMR offloads computationally intensive rendering to cloud servers: the server initializes a rendering engine and encoder to service every connected client. The models are rendered on the server side and encoded in video frames streamed from the server to the client [?, ?, ?, ?, ?].

However, CMR systems often require very high network bandwidth which is seldom available and suffer from interaction latency [?]. However some attempts have been made to improve such systems. Boukerche and Pazzi [?] use environment maps rendered on a server and sent to the client to reduce bandwidth. The client is able to respond to user interactions resulting in panning and tilting without latency based on the environment map. Shi et al. [?] propose a framework that leverages depth maps to reduce interaction latency. They take advantage of 3D image warping to synthesize the mobile display from the depth images generated on the server. However, theses two image-based rendering techniques assume static scenes and only support limited user interactions. For example, the use of environment maps by Boukerche and Pazzi [?] accelerates panning interactions, but a new environment map needs to be generated for translation, which increases interaction delay and bandwidth requirements. Similarly, at scene changes, the environment maps or the depth maps also need to be regenerated in the approach proposed by Shi et al. [?].

1.2.3 Augmented Reality Training Framework

As the augmented reality technology is progressing rapidly, it not only opens the doors for unlimited creativity and innovation, but also enables enterprises to speed up the training process and make it more beneficial to employees. Augmented reality is taking training to a higher level, as it allows trainees to gain hands-on experience without the costs and risks

of real hands-on. Another advantage that AR can provide is taking the remote training further than existing tools can. A trainer can immerse the remote trainees in a project that in-house teams are working on, bringing trainees together from across the globe in a way that feels more real then ever before.

In addition, VR systems can provide extra cues, not available in the real world [**?**], that can facilitate the learning of the task and they allow simulating the task in a flexible way to adapt it to users' needs and training goals.

Researchers have made great efforts to apply augmented reality in training fields, especially in medicine, maintenance and repair, training, machine setup, etc.

In augmented reality applications, the virtual objects are overlaid on top of the real scenes, via monitors, HMD (Head-Mounted Display) or even holographic projection. However, in different applications, the interactive objectives are different. For example, Webel et al. [**?**] proposed a framework for assembly and maintenance training. The trainees operate on a real machine, while the instructions are shown by overlaying a virtual machine on the real one. In this work, the interactive targets are the real objects, while the virtual ones are shown as the instruction. Gonzalez-Franco et al. [**?**] developed an approach for training in complex manufacturing scenarios, where the virtual objects are projected into the real scenes without their real counterparts. The trainers operate on the virtual objects with a wand to teach the trainees. In this use case, the interactive targets are the virtual objects.

We categorize the augmented reality applications into two categories according to the interactive objectives: real objects and virtual objects. Our proposed framework falls into the first category, namely real objects. As mentioned in Sec. 1.1, the trainees need to operate the components on their side according to the virtual components.

Currently, the augmented reality applications of the first category mostly involve virtual objects as the instruction. The virtual objects are prepared before-hand. The difference between our proposed method and the existing methods is that the instruction in our method is generated by a trainer in real time, therefore it enables the trainers to give instructions in urgent situations, such as safety events.

In typical augmented reality applications, camera registration needs to be done related to the object of interest. However, in our framework, camera registration alone is not enough since there may exist multiple objects of interest. Give the 3D structure of OOI (Object of Interest), Tjaden et al. [**?**] proposed a method to estimate object pose according to object segmentation of a video sequence. This method works with multiple objects. It is an appropriate approach for our framework. First, the speed of this method is fast, it runs at 50-100 Hz on a commodity laptop. Second, Our previous work on video object segmentation can be a preprocessing step for 3D object post estimation, since it produces silhouette of each component.

After the camera, the AR glasses and all the components are registered, the server in between trainer side and trainee side renders the moving components according to the trainee's view, in high quality. However, not all AR glasses or mobile devices are equipped with a powerful graphic capacity, e.g. Google Glasses or low-end mobile phones. So

the camera and the AR devices send a video sequence to the server respectively. The registration and rendering are both done on the server side, which relieve devices on both side from heavy computation workload.

1.3 Contributions

The main contributions of the proposed framework are:

- an on-line object segmentation method that empowers pose estimation algorithms to calculate faster and more accurately
- a remote rendering framework delivering high quality rendering to mobile devices while minimizing bandwidth usage
- an augmented reality training framework that enables the trainers to provide instructions to the trainees remotely and collect feedback, in real-time
- an augmented reality training framework that works with devices without powerful graphics capacity.

1.4 Outline

The reminder of this thesis is structured as follows. Chap. 2 reviews the previous literature of each research fields. Chap. 3 shows our work on video object segmentation and its usage in video content-aware compression. Chap. 4 demonstrates our on-going work of hybrid remote rendering. Chap. 5 unveils our plan to accomplish the final goal (i.e. realtime remote training with augmented reality). Chap. 6 concludes this thesis proposal and outlines the schedule of the remaining work.

Chapter 2

Related Work

2.1 Object Segmentation

Video object segmentation is the process of separating foreground objects from the background in a video [**?**]. A wide range of applications benefit from the progress of video object segmentation, e.g. robot-object interaction, recognition, video compression etc.

A variety of methods have been proposed to address the task [**?**, **?**, **?**, **?**, **?**]. Most methods use motion cues to initialize the object segmentation. A common method for motion detection is background subtraction with mixture of Gaussians [**?**, **?**]. This technique models each pixel indepedently as a mixture of Gaussians but as it detects motion in each frame independently, the results lack completeness and temporal persistence. In this paper, we address this shortcoming by modeling the appearance of objects from motion.

Our proposed method treats moving pixels as a partial segmentation of objects and builds the appearance model for them, see Section 3.1.2. With the appearance model, those pixels that are not moving can successfully be classified as foreground or background to form a more complete segmentation. We model appearnce with Radial Basis Functions (RBF) as described in Section 3.1.2, an approach commonly used in interactive editing for selection propagation [**?**, **?**]. We adapt the technique of Tao and Krishnaswamy [**?**] but use motion detection instead of user selection to initialize the models.

Moving pixels obtained with background subtraction approaches may be not just incomplete, but also contain pixels from the background. This is mainly caused by shadows and noise. The consequence is that the computed segmentations are likely to cover regions that are not part of the foreground. We address this mis-classification with modeling background appearance and optimizing the classification. We use Graph Cuts [**?**, **?**] to optimize the foreground and background classification based on the appearance model. To accelerate the computation, we divide frames into regions that contain one object each, see Section 3.1.1, and oversegment those regions with a fast superpixel method [**?**] before classification, see Section 3.1.3.

We evaluate our video object segmentation approach on a common dataset [**?**] in Section **??**. Our evaluation demonstrates that our approach achieves comparable results to

two state-of-the-art approaches but our approach is on-line and is $10 - 200\times$ faster than these methods.

The major contribution of our work is a fast on-line video object segmentation method that takes both motion and appearance of objects into account. We propose a novel integration of RBF appearance modeling and background subtraction through a Graph Cuts optimization on superpixels. We use local modeling to accelerate segmentation by dividing frames into regions that contain one foreground object each which greatly reduces the number of pixels to be processed. In Section **??**, we propose an application of our on-line approach for compressing videos with compression quality adapted to foreground and background. The video object segmentation allows us to reduce the quality of the background by pre-processing it with a bilateral filter before compression while foreground objects are compressed as is by the compression scheme. We use H.264 coding.

A large number of methods have been proposed for extracting moving objects from an image sequence, using motion, depth, appearance or a combination of these cues. Among those, motion is used most frequently as cues for object extraction [**?**] and many approaches use background subtraction to this end [**?**, **?**]. Classic background subtraction methods model the appearance of the background at each pixel and label the pixel that change rapidly to be foreground [**?**, **?**, **?**]. These method typically assume a stationary or slowly panning camera. More recently, optical flow is also used [**?**, **?**, **?**, **?**]. The motion estimation algorithms typically provide pixel-wise labeling and errors are inevitable even using state-of-art algorithms. To obtain robuster masks with semantic meaning, energy minimization is often used [**?**]. Optical flow usually provides more motion cues than background subtraction (e.g. pixel correspondence accoss frames), but in general is much slower than background subtraction methods. Our approach is closely related to tracking. We infer trimaps from detected motion to coarsely separate foreground from background, which is different from other methods based on motion cues [**?**, **?**].

With the advances in computational power of modern PCs and the progress of fast superpixel methods, the use of superpixels as units of object extraction are increasingly popular [**?**, **?**, **?**]. Instead of naive superpixel voting schemes, researchers often use some optimization approaches to decide if a superpixel belongs to the foreground or the background [**?**, **?**, **?**]. However, most of the superpixel methods are computationally intensive which prevents their use in real-time video object extraction [**?**]. A fast superpixel method was proposed by Siva and Wang [**?**]. It is based on seam carving and dynamic programming and it achieves real-time performance in low resolution videos. We adapt this method and further accelerate the computation of superpixels by only segmenting the regions within the bounding boxes of foreground objects into superpixels.

Several video object extraction methods track points over the image sequences and then cluster the resulting point trajectories pairwise or in triplets [**?**, **?**, **?**]. The advantage of employing point tracking is the capacity of handling videos where moving objects are stationary in a number of frames. The underlying assumption induced is that the objects are rigid so that all object points move according to a single translation [**?**, **?**], while the work of Ochs et al. [**?**] assumes a single similarity transformation. This assumption makes these methods not applicable to non-rigid objects. The methods produce a sparse

labeling in each frame, and superpixels are taken into account when turning the point trajectories into dense regions. These methods are also able to handle partial occlusion but the drawback is that they are very slow (in the order of minutes per frame).

The work by Lee et al. [?] shows the potential of using shape matching in object extraction. The method first identifies object-like regions in any frame [?], followed by computing a series of binary partitions among those candidate regions to discover groups of shapes with persistent appearance and motion. The drawbacks of this method include that it requires pre-processing identifying all object-like regions beforehand, it is very slow (in the order of minutes per frame), and it is not applicable to stationary or occluded (in some frames) foreground objects.

Depth information has been demonstrated to be robust to environment changes such as illumination change, dynamic backgrounds and camera motion [?, ?]. The work of Taylor et al. [?] infers depth layers from occlusion information. This gives it the capacity to be used outdoors and ensures it to be robust to occlusion and disocclusion. However, the method takes around 30 seconds for a VGA image on a standard desktop.

The works of Papazoglou and Ferrari [?] and Wang et al. [?] are closely related to ours. Papazoglou and Ferrari [?] use optical flow as the motion cues. After generating a coarse segmentation, Graph Cuts are used to minimize an energy function containing an appearance term and smoothness terms. In this paper, we propose a novel algorithm based on Radial Basis Functions to model the appearance and estimate how close it is from a pixel to foreground or background, consindering the neighboring frames. We also use Graph Cuts to obtain the final masks.

Many modern approaches offer offline processing to recognize objects in videos [?, ?, ?, ?]. The requirement of the availability of the entire video limits those methods in processing long sequences. In contrast, our method is online and offers processing in "streaming", which gives it the capacity to handle longer videos and integrate with other online applications.

2.2 Remote Rendering

Nowadays, with the rapid progress of mobile devices on hardwares and softwares, products in many categories have leveraged the advantages of them. Those areas include games, training, medicine and many other applications in everyday life. Moreover, the mobile device has wider availability than PCs, and it offers more intuitive interaction experience than PCs. Instead of keyboards and mice, users use gestures to interact with applications. However, complex 3D models demand high-end computing hardwares. Compared with PCs, mobile devices have much lower processing capacity, limited storages and less powerful rendering hardware, even with most recent high-end ones. Developing mobile applications, especially 3D graphics applications, often requires simplification of the 3D models, so that it degrades rendering quality. Computation intensive 3D graphics rendering tasks also impose severe challenges on the limited battery capacity.

An emerging direction to address this issues is Cloud Mobile Rendering (CMR). It offloads computation intensive rendering to the cloud servers, so that mobile devices are relieved of the burden of rendering. Basically, it when a mobile client connects to the server, the server will initialize a rendering engine and an encoder for the mobile client. The models are rendered on the server side and rendered frames are encoded and streamed from the server to the client as a raw video stream. Lamberti and Sanna [**?**] and Lu et al. [**?**] are examples of such CMR systems.

However, the CMR systems often suffer from very high network bandwidth requirement and interaction latency. For decades, a lot of attempts have been made to reduce bandwidth requirement and interaction. Boukerche and Pazzi [**?**] uses environment map in CMR. The server renders an environment map and sends it to the client. With the environment map, the client is able to respond to the user interaction in terms of pan and tilt without latency. Shi et al. [**?**] proposed a framework that leverages depth maps to reduce user interaction latency. It takes advantage of 3D image warping, to synthesize the mobile display from the depth images generated on the server. However, the image-based rendering techniques have a disadvantage that they assume static scenes and only support limited user interactions. For example Boukerche and Pazzi [**?**] uses environment map to accelerate panning interaction, but for translation, a new environment map needs to be generated, which increases interaction delay and bandwidth requirement. Moreover, when scene change, environment maps and depth maps also need to be re-generated.

This paper proposes a remote rendering system that aims at minimizing the network bandwidth requirement for remote rendering and the network latency. It has a client-server architecture and maintains two versions of models: low-fidelity models and high-fidelity models. The differences between two versions of models are in three aspects: a) number of polygons, b) resolution of textures and c) quality of rendering effects.

On the client side, the mobile devices runs with low-fidelity models that have less polygons, lower resolution of textures and lower quality of rendering effects, while on the server side, the work station runs with high-fidelity models that have more polygons, higher resolution of textures and higher quality of rendering effects. Depending on which models that need to be rendered in high-fidelity (key models), the server renders them and send the frames as a video stream. We define key models in two-fold: models that the user is interacting with are key models, and models that have semantical importance are key models. The models that user is interacting with can be identified by interaction information sent to the server, while the models that have semantical importance are specified by developers.

This architecture is able to reduce network bandwidth requirement and user interaction latency. Because only the regions of interest of the entire frame are encoded and streamed to the clients, while the changes on the rest part are discarded, it will reduce the bit rate of encoded video stream. The user interaction latency is composed of three parts: rendering, encoding and network transmission. Our proposed method aims at reducing the rendering time by only rendering and encoding the models of interest as high-fidelity mode and rendering the rest as low-fidelity mode without encoding. We call this rendering process as two-pass rendering and will describe it in the following sections.

Most rendering systems use a simple image-based approach that renders all 3D graphics on the server and sends the compressed image stream to the client. The server receives user interaction feedback from the client. This approach requires no 3D graphics capacity on the client and is able to use advanced encoding, such as H.264/AVC, to adjust the image quality according to the bandwidth availability. Lamberti and Sanna [?] and Lu et al. [?] are good examples. The largest disadvantage of this type of approach is latency, including user interaction transmission time, rendering and encoding time at the server side, image transmission time, decoding time at the client side [?]. The cloud gaming system OnLive sets the target latency to be less than 80ms, so as not to affect user experience even in motion-intensive games [?]. However, this target is not easy to meet considering the high uncertainty of the network transmission. Using a geographically proximate server helps greatly reduce the latency because every 1000 miles of physical distance adds 25 ms round-trip delay to the overall latency.

Chen [?] proposed a method widely known as QuichTime VR to display virtual environments without rendering 3D models. It uses 360-degree cylindrical panoramic images to compose a virtual environment. Panning and zooming are simulated by digital warping of computer rendering, specialized panoramic images or by "stitching" together overlapping photographs taken with a regular camera. Movement in a space is accomplished by "hopping" to different panoramic points. The framework proposed by Boukerche and Pazzi [?] also uses environment map to accelerate panning and zooming interaction. But their approach no longer simulates movement by "hopping" to another panoramic image, instead, they use a remote server to render panoramic images in real-time. To relieve the long delay of rendering and transmitting panoramic images, they design a cache system to buffer the visited views. A common disadvantage of methods that use environment map is that they only handle static environments. If the objects in the scene can move, re-rendering and re-transmission of environment maps can lead to great latency.

Chang and Ger [?] proposed building Layered Depth Images (LDIs) on the server. The mobile devices use 3D warping algorithm to synthesize the frame from a new view. At the time when this paper is published, the mobile devices almost has not 3D capacity. The original purpose of this study is a workaround for mobile devices to display users 3D contents. Such studies may seem ?outdated? now since current mobile phones and tablets have relatively powerful 3D graphics processors, although still fall behind desktops. Shi et al. [?] use a similar technique but aim at reducing interaction latency of the cloud mobile rendering system. It takes advantage of 3D image warping, to synthesize the mobile display from the depth images generated on the server. The approach generates the depth maps at carefully selected viewpoints and developed a viewpoint selection algorithm to achieve the balance of high warping quality and large motion coverage. The method is successful in reducing user interaction latency. But it is not designed to handle highly dynamic scenes, since a set of new reference depth maps need to be generated and delivered to the client whenever the scene changes. So this method is very suitable for environment walkthrough applications but not suitable for games and training applications.

Levoy [?] proposed a method that renders simplified models on clients to reduce bandwidth requirement. In this method, the server first render both complete and simplified models and calculate a difference image between the two. Then the simplified model and

the difference image are transmitted to the client. Finally, the client renders the simplified models and applies the difference image to produce a high-quality rendering. The simplification of models can be performed in terms of the textures and the number of polygons. This method reduces the bandwidth requirement for obtaining a high-quality rendering. Compared with our method, this work shares the same idea of using simplified models. But the difference between them is that Levoy [?] uses simplified models to achieve a compression effect on image transmission, while our method uses simplified models to display users the less important models.

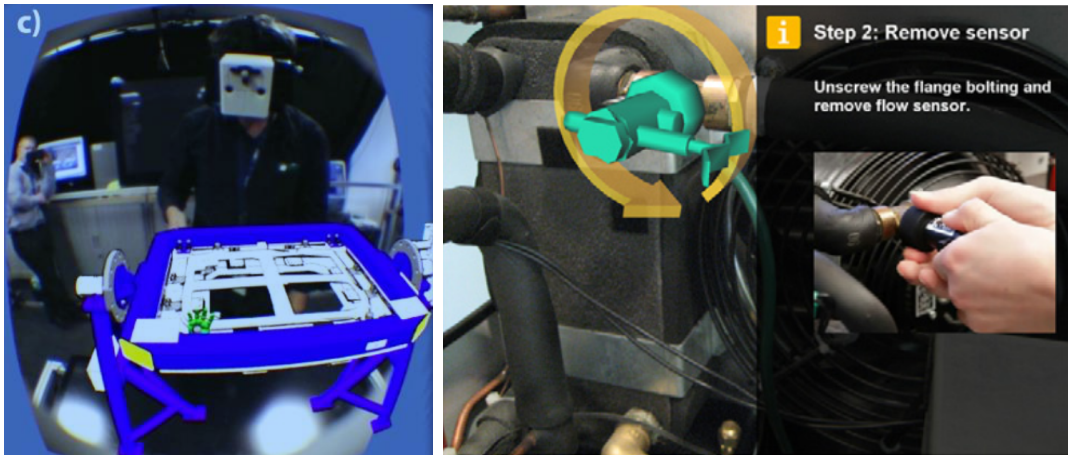
Liu et al. [?] extend the image streaming approaches and develop an automatic adaption algorithm that changes the rendering quality according to the network bandwidth. They use H.264 video encoding with fixed bit rate mode while adapting the rendering factors (e.g. view distance, realistic effect and texture detail) to improve the user experience. The goal of this method is to improve the visual quality of encoded frames with insufficient network bandwidth. However, the rendering quality of all models is adapted at the same level since H.264 video encoding does not distinguish the importance of different models in a scene. Moreover, it does not address one of the most severe issues in remote rendering, interaction latency.

Our approach does not assume any priori knowledge about the applications. It is able to handle dynamic scenes and all types of interaction (e.g. motion, panning and zooming). We leverage the advantage of graphics hardwares on modern mobile devices, so that not all models need to be rendered, encoded and transmitted. This reduces the bandwidth requirement and interaction latency.

2.3 Augmented Reality Training

The first augmented reality system can be dated back to 1962 [?, ?]. It employed a see-throught Head-Mounted Display (HMD). The HMD was tracked by a mechanical tracker and an ultrasonic tracker. However, only simple wireframes could be drawn and overlaid on the user's view due to the lack of computing capacity at that time [?]. In 1992, Boeing [?] had already leveraged AR technology to assist their workers in assembling wire bundles.

The research and development for AR have gone on over the past five decades. The use of augmented reality technology has been explored in various fields, e.g. education, training, tourism, gaming, health, safty, etc [?, ?, ?]. Freitas and Campos [?] developed an educational system using AR technology. The system teaches 2nd grade-level concepts, such as the means of transportation and types of animals. The system tracks a marker and superimposes the 3D models on the real-time video feed shown to the whole class. Experiments show that the nature of game-base learning of this system helps increase motivation among students [?]. Schrier [?] developed a historic role-play game at a real site. The participants are assigned to a specific historic role and interact with virtual roles with a GPS-enabled mobile device. Billings Hurst et al. [?] implemented a more interactive and realistic way of reading books. If the users look at the pages through a handheld AR display, they see 3D models appearing out of the pages.



(a) Interact with real objects (b) Interact with virtual objects

Figure 2.1: Two categories of augmented reality training scenarios according to which objects people interact with.

There exist many reviews of implementations and progress in augmented reality applications [?, ?, ?, ?]. Billinghurst and Kato [?] focused on collaborative AR applications and demonstrated examples that allow people to view or change the same virtual models. Regenbrecht et al. [?] showed ten AR applications in various fields which make cutting edge progress at that time. Zhou et al. [?] summarize the papers published on ISMAR from 1997 to 2007 in an statistical way. They focus on analyzing the frequency and percentage of citations of each topic involved in augmented reality applications. Lee [?] discussed the areas that augmented reality technology has been used in and identify how it can be used in each area. We present a new methodology to categorize existing applications of augmented reality in the remaining part of this section.

In this section, we focus on the use of augmented reality technology in training. After reviewing recent progress in this area, we propose a new methodology of categorizing the AR applications. We categorize them into two categories in terms of interactive targets: real objects and virtual objects. Fig. 2.1 shows two examples, where the left one shows that a trainee is interacting with a real object, and the right one depict that a trainee is interacting with a virtual object. When interacting with a real object, the overlaid virtual ones act as an instruction that guides the trainee through the process.

2.3.1 Virtual Objects

In the framework depicted in [?], the camera captures the images of a participant standing in front of a blue screen. The participant uses a "wand" to interact with a virtual environment, where the "wand" can be a finger, a fire extinguisher or other things we have seen in normal life. There is no marker needed to recognize the position and orientation of the "wand". Luo et al. [?] developed an AR-based therapy application that was designed for post-stroke finger extension rehabilitation. The application involves both therapists and

the patients. During the rehabilitation, the patients need to wear a head-mounted display and an orthosis. The virtual objects prepared by the therapist are mixed with the patient's hand and displayed on the head-mounted display. When the patient is trying to grasp or release the virtual object, the on-site therapist adjusts the assistance provided by the orthosis. Leblanc et al. [**?**] compared an AR training method with the traditional methods in the field of straight laparoscopic colorectal skills acquisition training. The traditional methods cost much higher than the the AR training method since it uses human cadavers. They conclude that simulator training followed by cadaver training can appropriately integrate simulators into the learning curve and maintain the benefits of both training methodologies. Nishino et al. present a Japanese calligraphy training system to teach learners how to write better characters [**?**]. It allows the learners to watch and feel the writing techniques of an instructor. More specifically, the training system displays a virtual brush and enables the learners to intuitively master instructor's motor skills through the sense of touch with a haptic device. Gonzalez-Franco et al. [**?**] developed an approach for training in complex manufacturing scenarios, where the virtual objects are projected into real scenes without their real counterparts. The trainers operate on the virtual objects with a wand to teach the trainees. In this use case, the interactive targets are the virtual objects.

The methods in this category are usually applied on the fields where the cost or risk of real hands-on experience is too high. For example, a surgeon could use AR to learn how to perform open-heart surgery without risking patients' lives.

2.3.2 Real Objects

In engineering, especially in the field of machine maintenance and repairing, the cost of real hands-on experience is relatively low, but the complexity of the target is very high. In this kind of scenarios, people often use augmented reality technology to show virtual objects as a guide.

Boeing proposed the first industrial augmented reality application that helps its workers with assembling aircraft wire bundles [**?**]. Henderson and Feiner developed a framework to assist conducting military routine maintenance tasks inside an armored vehicle turret [**?**]. The experiment shows that use of AR increases the precision of component locating by 56% compared with the use of traditional untracked head-up displays (HUDs) and speeds up the task by 47% compared with standard computer monitors. Crescenzio et al. [**?**] implemented an AR-based maintenance tool for daily inspections of the Cessna C.172P, an airplane often used by flight schools. It superimposes digital replicas of parts and subparts or graphical symbols to attract the operator's attention and guide the technicians through a task. It uses a markerless, feature-based method for HMD registration. What is more interesting is that they also provide an authoring tool for new training material generation. By leveraging the CAD models and existing scanning tools, a trainer can generate new training materials without the knowledge of programming. However, it still requires a lot of efforts to scan real scenes for locating the virtual objects in the real scenes.

Hincapi et al. [**?**] demonstrated an AR-based training process to perform maintenance on the body of the RV-10 aircraft. The system overlays virtual objects onto real objects

as an instruction. Trainees need to work on the RV-10 aircraft component based on the instructions. Webel et al. [**?**, **?**] proposed a framework for assembly and maintenance training. The trainees operate on a real machine, while the instructions are shown by overlaying a virtual machine on the real one. In this work, the interactive targets are the real objects, while the virtual ones are shown as the instruction.

Most of existing methods generally guide the user through a fixed series of steps and the digital content is prepared beforehand. The topic of augmented reality training material authoring is overlooked in this research area. Zhou et al. [**?**] conclude that only 3.8% papers published on ISMAR from 1997 to 2007 discussed about authoring but made 8.9% of citations. However, we cannot find existing work on real-time authoring tools. Wang et al. [**?**] developed an authoring tool for AR-based examinations. Providing 3D models involved in exam questions, the application generates the AR content and stores it in a separate file that can be shared, edited or imported into an AR interface. Anderson et al. [**?**] proposed a movement training application that is equipped with a human motion recording tool. They record people's movements and present them in an augmented reality mirror, and the users can see their mirror image as well as the targeting movements in the mirror and follow those movements. However, they are still not real-time authoring tools, since the content need to be prepared or captured before the training. Moreover, during the training, the applications collect minimal feedback from the user and cannot adjust according to the user's performance or mistakes. Missing real-time authoring tools and the lack of just-in-time feedbacks place an obstacle between the trainer and the trainee. Our proposed framework bridges the gap between the trainer and the trainee. On the one hand, guides or instructions are not fixed but generated by the trainer in real time instead. On the other hand, the steps taken by trainees are collected and sent to trainers, so that trainers are able to evaluate the trainees' performance and correct mistakes.

Chapter 3

Video Object Segmentation

In this section, we present a method for extracting foreground objects from video and its application to content-aware video compression. Our method uses trimaps inferred from background subtraction to represent the foreground-background relationship. The appearance of foreground and background are modeled with Radial Basis Functions initialized from the background substraction step. Finally, Graph Cuts are used to compute a binary mask. Our method is fully automatic, fast, and does not make restrictive assumptions about object motions. In experiments on standard data sets, the proposed approach achieves comparable results to state-of-the-art video object segmentation methods but our method is much faster. We also demonstrate an application of the proposed method to content-aware video compression.

3.1 Method

Our method consists of three steps: (1) trimap generation, (2) color similarity calculation, (3) foreground-background labelling. Fig. 3.1 illustrates the overview of the proposed pipeline. First, motion is detected with the background subtraction method, followed by blob analysis to generate trimaps. Second, the appearance model is built with Radial Basis Functions, where sampling is done in different regions of the trimap for foreground and background respectively. With knowing how close a pixel is to the foreground or the background according to its color, we produce evidence maps. To enhance the spatial and temporal persistency, we smooth the maps with a bilateral filter kernel. Third, using Graph Cuts, we are able to generate a binary mask by minimizing an energy function. Next, we present the detail of these steps.

3.1.1 Trimap Generation

Background subtraction. We begin by computing background subtraction frame by frame using effective mixture of Gaussians algorithms [**?**, **?**] according to a comparison [**?**] of various background subtraction methods. We employ the implementations of the two

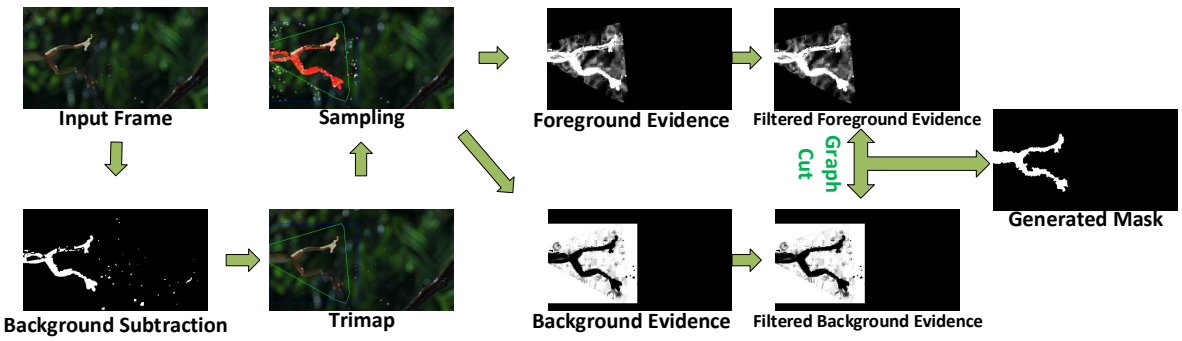


Figure 3.1: Overview of the proposed video object segmentation method. The images are a frame from the dataset SegTrack v2 [**?**].

algorithms MOG and MOG2 in OpenCV. After background subtraction, we perform a median filter of kernel size 3 to reduce noise, see Fig. 3.2(b).

Blob analysis. After obtaining moving pixels in the background subtraction step, we apply closing morphological transformation to group them. We perform a dilation step followed by an erosion in the binary labelling of the image as it is typically used to close small hole inside the foreground objects. The purpose of grouping moving pixels is to reduce isolated moving pixels. Then the convex hulls of blob contours are calculated. The contours are found using Suzuki's algorithm [**?**] while convex hulls are calculated using Sklansky's algorithm [**?**]. The bounding boxes are then computed from the convex hulls, see Fig. 3.2(c).

We base our trimap generation on the contours and bounding boxes of the blobs of moving pixels. A trimap is a partition of images into three regions: a definite foreground, a definite background, and a blended region where pixels are considered as a mixture of foreground and background colors. More specifically, moving pixels obtained using background subtraction serve as the definite foreground, while the pixels inside the bounding box and outside of the convex hull are treated as definite background. Those pixels inside the convex hull but not recognized as moving pixels are the blended region. Such a trimap is established for each blob and segmented individually. Because in some cases, background subtraction generates trimaps smaller than the actual size of objects, we expand the areas of the convex hulls and bounding boxes by a ratio r such that $A^* = r \cdot A$, where A and A^* are the areas before and after expansion respectively, see Fig. 3.2(d). In our implementation, we typically use $r = 2$.

3.1.2 Color Similarity Calculation

The goal of this stage is to estimate the distance of a pixel to the foreground and the background based on a color similarity metric between every pixel in the blended region and the definite regions.

Radial Basis Functions Li et al. described in detail a method using Radial Basis Functions (RBF) to propagate user edits in image matting [**?**]. Similar to [**?**], we use

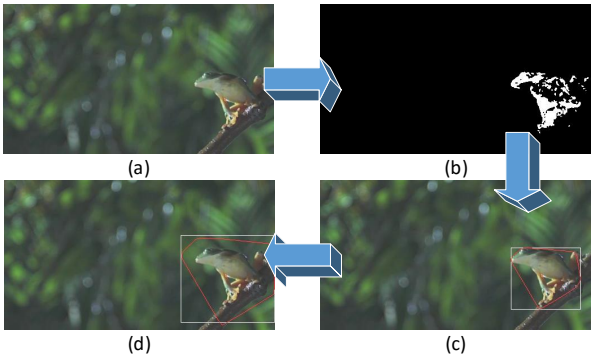


Figure 3.2: Trimap generation. (a) Original frame. (b) Background subtraction result. (c) Convex hull (red) and bounding box (white) of the blob. (d) Convex hull (red) and bounding box (white) after expansion. The images are a frame from the dataset SegTrack v2 [**?**].

RBF to account for all the pixels within the definite foreground or the background regions. However, we define the RBF in the three-dimensional color space where each pixel i is represented by its feature vector $\mathbf{f}_i$. The method samples the foreground pixels and the background pixels, respectively, and computes the coefficients of the Radial Basis Functions subject to interpolation constraints using a linear solver. With G representing all foreground or background pixels, we formulate the interpolation constraints as a least-square energy function,

$$\sum_{i \in E} (1 - h(\mathbf{f}_i))^2 \tag{3.1}$$

where $E \in G$ is the subset of all definite foreground or background pixels, and $\mathbf{f}_i$ is the color vector of pixel i. $h(\cdot)$ is the RBF centered at the pixels in $B \in G$, where B is another subset of G. To allow *soft* interpolation and to gain computational efficiency, we define that $|B| < |E|$, where $|\cdot|$ counts the number of pixels in a set. We choose to sample the set E to have twice the size of B. Let

$$\begin{aligned} h(\mathbf{f}) &= \sum_{i \in B} a_i \phi(||\mathbf{f} - \mathbf{f}_i||) \\ \text{with } \phi(r) &= exp(-\sigma * r^2) \end{aligned} \tag{3.2}$$

where $\mathbf{f}$ is any point in the color space, a_i are the unknown coefficients, $\phi(\cdot)$ is some pre-defined radial basis and σ controls the smoothness of the Gaussian bases [**?**].

The coefficients a_i represent the importance of a color being similar to a color within the set B. Eqn. 3.2 directly estimates the similarity of a color to all the colors in the foreground or the background regions. Pixels with the value of 1 are most similar to the corresponding definite region while pixels with the value of 0 represent dissimilar colors. As subsets are sampled from definite regions and Eqn. 3.1 is over-determined, the values of pixels may be greater than 1 or less than 0 but we clip the values to $[0, 1]$.

(a) Input Image and Strokes (b) Output Image

Figure 3.3: An example of computing color similarity with RBF. (a) The two strokes specifies pixels having similarity 1 (green) and 0 (red). (b) The resulting RBF is applied on every pixel.

The reason that we use a subset $E \in G$ instead of G to construct Eqn. 3.1 is to accelerate the computation because of the use of the linear solver to compute the RBF, each pixel is costly [**?**].

Moreover, we adapt the method used by Tao and Krishnaswamy [**?**] to sample the definite foreground or background regions using importance sampling rather than random sampling. Our goal is to reduce the number of samples needed while maintaining consistent results. In our application, we use 27 clusters, 54 radial basis functions and 108 terms in Eqn. 3.1.

Foreground and background evidence. Assume that F is the set of definite foreground pixels, B is the set of background pixels and D is the set of blended pixels. We define the evidence of the i^{th} pixel to be the foreground $e_f(\mathbf{f}_i)$ as

$$e_f(\mathbf{f}_i) = \begin{cases} 1, & \text{if } i \in F \\ clip(h_f(\mathbf{f}_i)), & \text{if } i \in D \\ 0, & \text{if } i \in B \end{cases} \tag{3.3}$$

where $\mathbf{f}_i$ is the color vector of the i^{th} pixel, $h_f(\mathbf{f}_i)$ is the RBF to the foreground region and $clip(\cdot)$ clips a value to the range $[0, 1]$. Similarly, the preference of the i^{th} pixel to be the background $e_b(\mathbf{f}_i)$ as

$$e_b(\mathbf{f}_i) = \begin{cases} 1, & \text{if } i \in B \\ clip(h_b(\mathbf{f}_i)), & \text{if } i \in D \\ 0, & \text{if } i \in F \end{cases} \tag{3.4}$$

where $h_b(\mathbf{f}_i)$ is the RBF to the background region.

Evidence smoothness For some videos, the temporal persistence of background subtraction methods is poor, therefore the RBF of the same object can vary greatly from frame to frame. To alleviate this problem, we add an extra smoothing step to the evidence computation. We use a filter with a cubic kernel in the spatial and temporal domain, and

smooth the foreground evidence and the background evidence seperately. The smoothed evidence of pixel p is a weighted sum of the set of its neighboring pixels Ω^+, where Ω^+ is the window centered in p. However, when the algorithm processes the t^{th} frame, the frames after the t^{th} frame are not available, therefore, only half of Ω^+ actually have an effect, denoted as Ω. The smoothed foreground evidence e_f^* is represented as

$$\begin{aligned} e_f^*(p) &= \frac{\sum_{q\in\Omega} e_f(q)\cdot w_{p,q}}{\sum_{q\in\Omega} w_{p,q}} \text{ where} \\ w_{p,q} &= exp(-\frac{(p_x-q_x)^2+(p_y-q_y)^2}{2\sigma_d^2} - \frac{||\mathbf{f}_p-\mathbf{f}_q||^2}{2\sigma_r^2}) \end{aligned} \tag{3.5}$$

In Eqn. 3.5, p_x and q_x are x coordinates of pixels p and q, while p_y and q_y are y coordinates of pixels p and q, and $\mathbf{f}_p$ and $\mathbf{f}_q$ are the color vectors of these pixels. The variances σ_d^2 and σ_r^2 control the degree of smoothness. By increasing σ_d^2 and σ_r^2, the evidence becomes smoother. We set $\sigma_d^2 = 1$ and $\sigma_r^2 = 1$ which are the same weights as in a bilateral filter. The filter performs smoothing while preserving dissimilarity between pixels far away or with greatly distinct colors.

3.1.3 Foreground-background Labelling

Superpixels. We oversegment the m^{th} trimap in the t^{th} frame into superpixels $\mathcal{S}_m^t$, which greatly reduces computational and memory costs. We use the method proposed by Siva and Wong [**?**], which is a seam carving approach to superpixel generation and yields results with grid structure. This method is proven to be faster than most of the existing approaches, and can achieve accuracies close to the state-of-the-art superpixel generation algorithms, e.g. SLIC [**?**].

Graph Cuts. The problem of segmentation can be considered as an energy minimization problem.

We use graph cuts in the minimization and represent the superpixels $\mathcal{S}_m^t$ as the nodes $\mathcal{V}$ in a graph with an edge to every neighboring superpixel. A neighboring superpixel is defined as superpixel that borders the superpixel under consideration, either horizontally or vertically. The two terminals in the graph cut represent the labels of foreground and background. Therefore,

$$E(\mathcal{L}) = \sum_{i\in\mathcal{S}} D_i(\mathcal{L}_i) + \sum_{(i,j)\in\mathcal{N}} V_{i,j}(\mathcal{L}_i,\mathcal{L}_j), \tag{3.6}$$

where $\mathcal{L} = \{\mathcal{L}_i | i \in \mathcal{I}\}$ is a labelling of superpixels $\mathcal{S}$. D_i is a data penalty function, $V_{i,j}$ is an interaction potential, and $\mathcal{N}$ is a set of all pairs of neighboring superpixels. For simplicity, we discard the superscript and subscript of the notation $\mathcal{S}_m^t$. D_i denotes individual label-preferences of superpixels to be foreground or background. Interaction potentials $V_{i,j}$ encourages spatial coherence by penalizing segmentation between non-edge superpixel pairs.

D_i is represented by the sum of evidence for pixels to be foreground or background.

$$D_i(\mathcal{L}_i) = \begin{cases} \sum_{p \in \mathcal{S}_i} e_b(\mathbf{f}_p), & \mathcal{L}_i = 1 \\ \sum_{p \in \mathcal{S}_i} e_f(\mathbf{f}_p), & \mathcal{L}_i = 0 \end{cases} \tag{3.7}$$

where $\mathcal{S}_i$ is the i^{th} superpixel, p is a pixel within the superpixel $\mathcal{S}_i$, $\mathbf{f}_p$ is the color vector of the pixel p, and e_f and e_b are define in Eqn. 3.3 and Eqn. 3.4, respectively. $V_{i,j}$ is represented by the Euclidean distance between average color vectors of two superpixels, i.e.,

$$V_{i,j}(\mathcal{L}_i, \mathcal{L}_j) = \begin{cases} dist(i,j), & \mathcal{L}_i \neq \mathcal{L}_j \\ 0, & \mathcal{L}_i = \mathcal{L}_j \end{cases} \tag{3.8}$$

where $dist(i,j)$ computes the Euclidean distance between average color vectors of the i^{th} and j^{th} superpixels. The output segmentation is then the labelling that minimizes

$$\mathcal{L}^* = \arg\min_{\mathcal{L}} E(\mathcal{L}). \tag{3.9}$$

We use the method proposed by Boykov and Kolmogorov [**?**] to solve Eqn. 3.9.

3.2 Evaluation

3.2.1 Dataset

We evaluate our method on the dataset SegTrack v2 [**?**]. SegTrack v2 is a video segmentation dataset with full pixel-level annotations on multiple objects at each frame within each video. It contains 14 videos, including videos captured by either static cameras or moving cameras. We use the five videos with static cameras (birdfall, frog, hummingbird, bird_of_paradise and penguin). The ground truth of the videos hummingbird and penguin are split up, with each object labelled separately. In those cases, we combine the separate labels of the ground truth together to obtain a ground truth with labels for all objects.

We quantify performance with the F-Measure [**?**] based on the amount of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). The F-Measure is defined as the harmonic mean of precision and recall, where $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$, and hence the F-Measure

$$FM = \frac{2 * Precision * Recall}{Precision + Recall}. \tag{3.10}$$

where the F-Measure ranges from 0 to 1, with value 1 representing the ground truth segmentation.

3.2.2 Results

We compare our method with the two recent methods of Papazoglou and Ferrari [?] and Zhang et al. [?], as well as two background subtraction methods [?, ?]. The video object segmentation method of Papazoglou and Ferrari [?] produces a coarse segmentation with optical flow, then refines the segmentation by minimizing an energy function with Graph Cuts. The method of Zhang et al. [?] uses optical flow and layered DAG to generate and score an enhanced set of object proposals. Graph Cuts are used to refine the segmentation. We used the implementations provided by the respective authors[1]. The two background subtraction approaches [?, ?] are used in our method to generate initial moving pixel segmentation. We compare our method with the two approaches to illustrate the obtained improvements. We ran all experiments on the same PC with a 3.6 GHZ Intel i7 CPU.

When evaluating our method, we used the same parameters for all experiments. However, for the videos birdfall and bird_of_paradise, we used MOG background subtraction approach [?] with a learning rate of 0.005 to detect initial moving pixels, while, for the other videos, we used the MOG2 approach [?] with a learning rate 0.005. Because the two subtraction methods produce very different results for those videos, we chose the one giving us better initial moving pixels for each video. As for the approaches we used for comparison, we used the default parameters of the implementations provided by the authors.

The F-Measure obtained by the five methods are shown in Table 3.1, while Table 3.2 shows the running time of the methods [?], [?] and our method. We only show the corresponding results of the background subtraction method used in our approach (MOG or MOG2). For two videos (bird_of_paradise and penguin), our method gives much better results than the initial segmentation produced by the background subtraction methods. For these two videos, the MOG or MOG2 approaches miss large parts of the object, while our method produces better results by integrating motion cues and the appearance models. (See the first and second columns of Fig. 3.4). For the video frog (third column of Fig. 3.4), our method slightly improves the mask quality, around 4%. But it accomplishes the mask of the frog, such as the torso and the legs. For the video hummingbird, the result of our method is worse than MOG2, which is caused by the large regions of mis-detection (the forth column of Fig. 3.4). In general, our method uses background subtraction methods to generate an initial object segmentation and effectively refines the initialization.

The method proposed by Parazoglou and Ferrari [?] produces better results while our method is much faster and produces comparable results. For four of the five videos, the method [?] outperforms our method in F-Measure, but it is around 10 times slower. For the frog video, our method gives very close performance. For the penguin video, the method [?] misses the moving penguins. Moreover, our method uses an online procedure, it reduces the demand for resources by computing the object segmentation frame by frame, which enables our method handle long videos. In contrast, the approach of [?] optimizes the results of the entire video, which limits its use on high-resolution and long videos. The memory demand of method [?] is very large, so that it is impossible for us to run it with

[1] http://www.dromston.com/projects/video_object_segmentation.php
http://groups.inf.ed.ac.uk/calvin/FastVideoSegmentation/

Table 3.1: Comparison with methods [?] and [?] on four videos birdfall, frog, hummingbird and penguin. The results are measured by F-Measure.

	Ours	[?]	[?]	MOG [?]	MOG2 [?]
bird_of_paradise	0.509	0.963	-	0.381	-
birdfall	0.159	0.72	0.832	0.14	-
frog	0.767	0.81	-	-	0.739
hummingbird	0.453	0.61	0.036	-	0.6
penguin	0.615	0.19	0.117	-	0.384

Table 3.2: The execution time of the comparison methods on four the videos: Birdfall, frog, hummingbird and penguin (in seconds).

	Ours	[?]	[?]
bird_of_paradise	97	1890.5	-
birdfall	1	205.52	250
frog	135	2740.1	-
hummingbird	65	435.13	484
penguin	35	225.08	240

the two videos bird_of_paradise (98 frames) and frog (279 frames). So we evaluated the method with the other three videos. Our method outperforms the method in two of the three videos. For the videos hummingbird and penguin, the method [?] misses the objects. Our method is around $10 - 200\times$ faster than the methods [?] and [?]. These methods are implemented in MATLAB while our method is implemented in C++ and OpenCV.

We explored the effects of color spaces and different values of σ in Eqn. 3.2. We perform the tests on the frog video. Fig. 3.5(a) shows the results of three color spaces and different values of σ. The advantage of one color space over others is not obvious. However, the values of σ have an impact on mask quality, reaching a peak at a specific value indicating the appropriate amount of smoothing. In other experiments of this paper, we use the RGB color space and set σ to 0.006.

When constructing the RBF, one important parameter is the number of constraints, i.e. the number of terms in Eqn. 3.1. As shown in Fig. 3.5(b), the number of constraints impacts the mask quality. In this experiment, we set the number of Gaussians to 7, i.e. $|B| = 7$. The number of constraints, i.e. $|E|$, is shown along the X-axis in Fig. 3.5(b). From $|E| = 7$ to $|E| = 35$, the quality of masks increases with the number of constraints. If we continue to increase $|E|$, there is no obvious benefit. Usually, we set $|E|$ as 2 to 5 times larger than $|B|$.

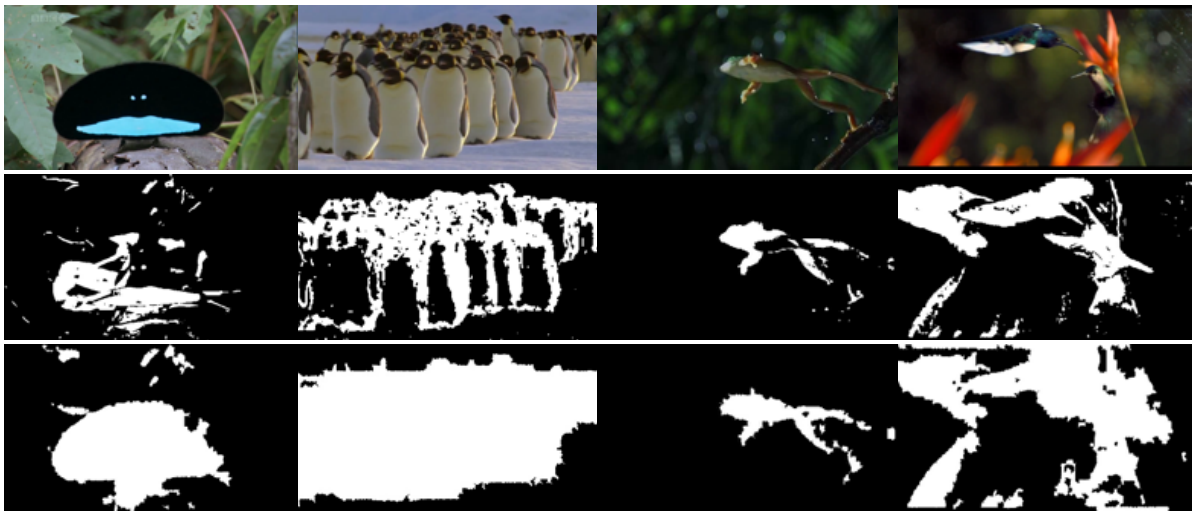
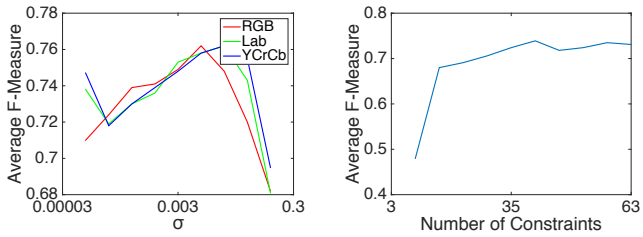


Figure 3.4: Background subtraction masks vs. final masks. The first row demonstrates the original frames, the second row shows the masks generated by background subtraction methods, while the third row is the final masks produced by our method.



(a) (b)

Figure 3.5: Parameter Influence: 3.5(a) σ vs. F-Measure in color spaces RGB, Lab and YCrCb. 3.5(b) Effect of the number of constraints on mask quality. We evaluated the quality of the produced masks using the average F-Measure of the frames with different values of σ and number of constraints.

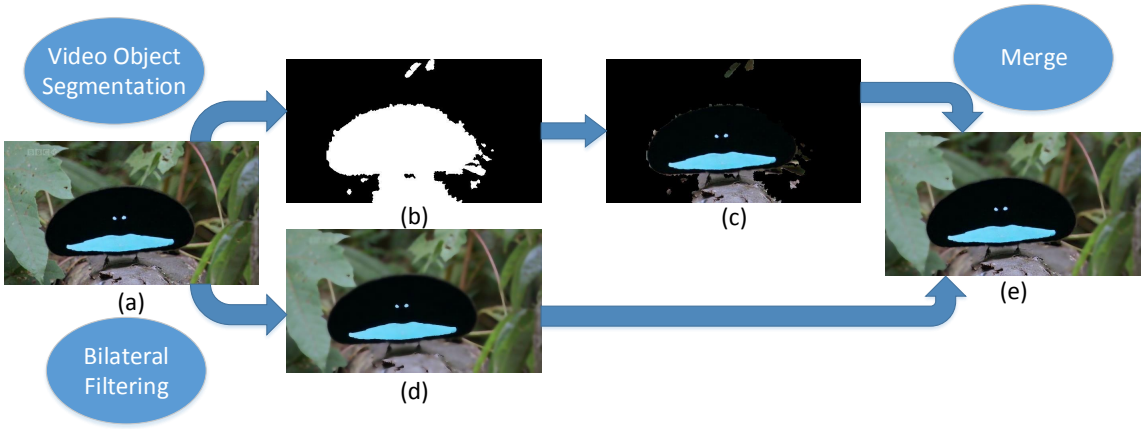


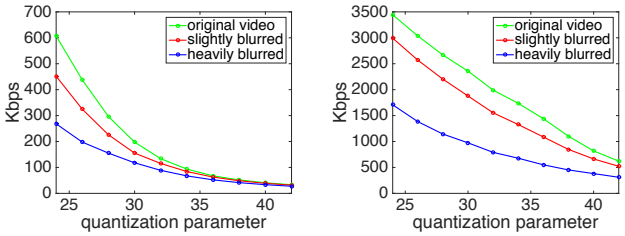
Figure 3.6: Content-Aware Video Compression. (a) Original frame. (b) Mask generated with the video object segmentation method described above. (c) The object pixels covered by the mask. (d) The frame blurred with bilateral filter. (e) The merged frame.

3.3 Content-Aware Video Compression

Video compression standards such as H.264/AVC compress videos as a whole. We applied the approach described above to conduct content-aware compression. We first extract the moving objects and blur the background using a bilateral filter. In this way, we obtain frames with original objects and blurred background. Then we use the obtained video as the input to a H.264 encoder. This method is effective in reducing the bitrate of compressed videos with the same encoding parameters since the blurred background has reduced high frequency components. The proposed content-aware compression shares some ideas with the block-based layered approach by Wang et al. [**?**] and the sparsity decompression of Chen et al. [**?**]. Our approach is also related to seam carving for compression as proposed by Decombas et al. [**?**].

Fig. 3.6 demonstrates the workflow of content-aware video compression. First, a mask (b) of foreground objects is generated with our video object segmentation method, followed by extraction of foreground pixels (c) covered by the mask. The original frame (a) is also blurred with the bilateral filter (d). Finally, the blurred image and the foreground regions are merged together (e). The composite frames are used as the inputs of a typical video encoder such as H.264/AVC. Our results demonstrate that this method can reduce the bitrate of compressed videos.

Fig. 3.7 shows the effectiveness of our video compression method. The Fig. 3.7(a) shows the results with H.264 inter-coding, while the Fig. 3.7(b) demonstrates the results with H.264 intra-coding. The green lines illustrate the bitrate of the original video compressed by H.264, which has the highest bitrate among all three types of videos. The red lines show the composite video blurred with the bilateral filter and smoothing parameter set to 0.1. The composite video achieves a lower bitrate than the original video. Moreover, with lower H.264 quantization parameters, the effectiveness of bitrate reduction becomes



(a) Inter-coding (b) Intra-coding

Figure 3.7: Bitrate of Compressed Videos. The X-axis shows the various quantization parameters, while the Y-axis is the bitrate (Kbps) of the compressed videos. Green lines are the bitrate of the original video, red lines show the bitrate of the composite videos with slight blurring, and blue lines demonstrate the bitrate of the composition videos but with heavy blurring. (a) and (b) shows the results with different coding methods.

more noticeable. We also noticed that the bitrate reduction is higher with inter-coding than intra-coding, since background blurring benefits not only intra-frame compression, but also the prediction between frames. We also tried to increase the smoothing parameter of the bilateral filter, which gives a heavier blurring effect. The blue lines demonstrate the bitrate of the heavily blurred videos. For the inter-coding, the bitrate is slightly reduced, while for the intra-coding, the bitrate reduction is more obvious. We conclude that compression of videos with high-resolution foreground and blurred background can effectively reduce the bitrate, the quantity of reduction depends on the blurring degree of background, and the effectiveness is more noticeable with inter-coding than intra-coding.

Chapter 4

Hybrid Remote Rendering

4.1 Method

We propose a remote rendering system that minimizes the network bandwidth requirements and the network latency in remote rendering. Our approach is a client-server architecture and maintains two versions of models: low-fidelity models and high-fidelity models. Low and high fidelity models differ in the number of polygons and in rendering quality.

On the client side, the mobile device renders low-fidelity models that have less polygons, lower fidelity of textures and lower quality rendering effects, while on the server side, the workstation renders high-fidelity models that have more polygons, higher fidelity of textures and higher quality rendering effects. Hence key models are rendered on the server and captured in images which are sent to the client as a video stream. We define key models as those that are important to the application. The models that the user is interacting with can be identified from interaction information sent to the server, while the models that are important to the application are specified in advance by the application developers.

This architecture is able to reduce required network bandwidth and latency in user interactions. Since only the regions of interest of the entire frame are encoded and streamed to the clients, while the rest is discarded, it reduces the bit rate of the encoded video stream. The user interaction latency is composed of three components: rendering, encoding and network transmission. Our proposed method aims at reducing the rendering time by only rendering and encoding the key models in high-fidelity mode and rendering the rest in low-fidelity mode without encoding.

4.1.1 Client-Server Prototype Design

The proposed system aims at providing high-quality rendering on less powerful mobile devices, while minimizing the amount of data transferred via network. Moreover, it also needs to enable multi-client cooperation to enhance the user experience in training scenarios.

The proposed system is inspired by Levoy [**?**], Lamberti and Sanna [**?**] and Lu et al. [**?**]. It is a hybrid approach of model-based and image-based methods. On one hand, the low-

fidelity models are stored on the client-side and the local rendering capacity is leveraged to produce decent rendering results. On the other hand, the high-fidelity rendering results of specific models that are required are sent from the server to the client and overlaid upon the locally rendered frames. In this way, the data transfered via network is minimized since the existence of local models and the pixels sent via the network are restricted at the models that are required by the client.

There are three challenges in realizing a multi-client cooperation system with the functionalities described above. First, the existence of multiple clients requires that the server must not be blocked by any of the clients. Second, each client has a different view from all the other clients. This can result in performance issue since a scene needs to be rendered multiple times in one iteration. Third, because only a subset of the models are rendered on and rent from the server, the occlusion becomes a problem since the model that is closer to the camera might not be required to render.

We address the first problem by interact with each client independently. More specifically, we enable the server to receive interactions from and send them to every client independently, so that if a client becomes unresponsive, the server is not blocked. The second issue is relieved by implementing the server as "render-on-demand", which means the server renders the scene for a client only when it requires a new frame. We address the third challenge by a two-pass rendering process. In the first pass, the entire scene is rendered. Then the colour buffer is cleared before the second pass. In this way, the second pass is rendered upon the depth information obtained from the first pass.

As depicted in Fig. 4.1, in this system, all clients are connected to the rendering server and send interaction commands to the server. On the server side, the application status changes according to the commands received from all clients. The application status changes are synchronized with all clients. In this way, the clients are able to cooperate with each other. In other words, when a client changes the status of the application, all clients will see the change immediately. To minimize the amount of transferred data, the client decides which models need to be rendered in high-fidelity and the server only sends the pixels of those models. The other models are still rendered locally.

Fig. 4.2 illustrates the workflow of communication between the server and the clients. It shows the workloads during one frame on the server and the clients. If the user interaction sent from any client would influence the simulation states of the server, it synchronizes the state changes among all clients, which ensures all clients have the same states. Moreover, the server renders and encodes the frames for all the clients. Upon receiving the high-fidelity frame, the client encodes and overlays it on the locally rendered frame.

4.1.2 Workflows

Our prototype consists of one server and multiple clients, we will show the workflow of both and various issues in the next sub-sections. Fig. 4.3 shows the workflow of the server. In each loop, the server updates the simulation status of the scene and receive commands from all clients. After that, the server sends commands to all clients in order to keep the simulation statuses of those client synchronized. In the server side, each client has its own

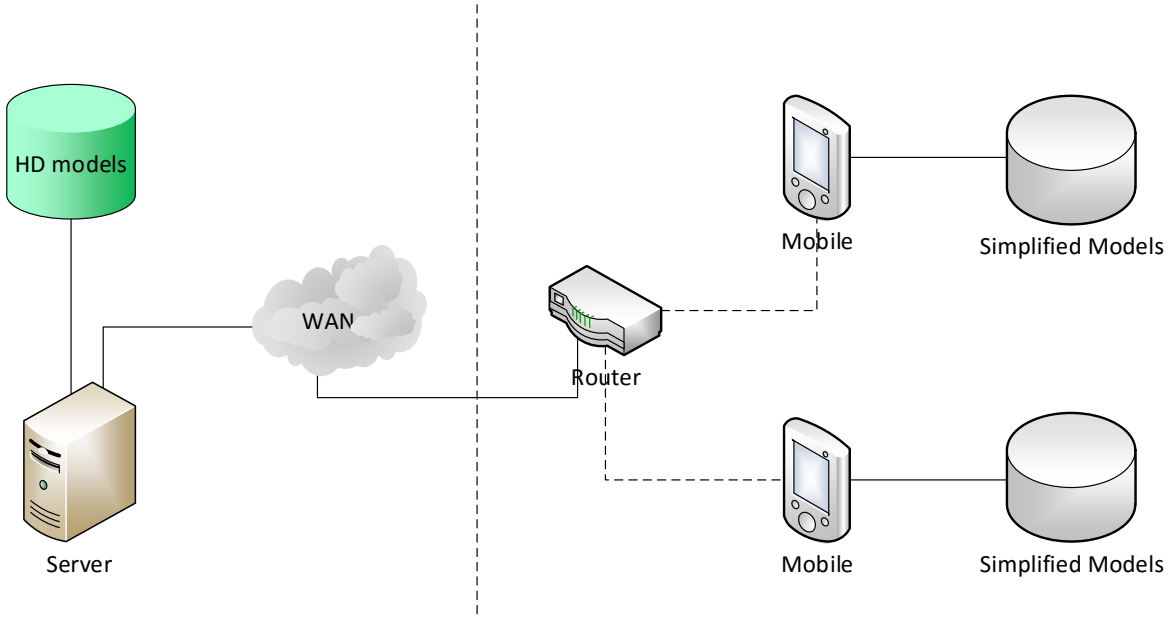


Figure 4.1: System architecture

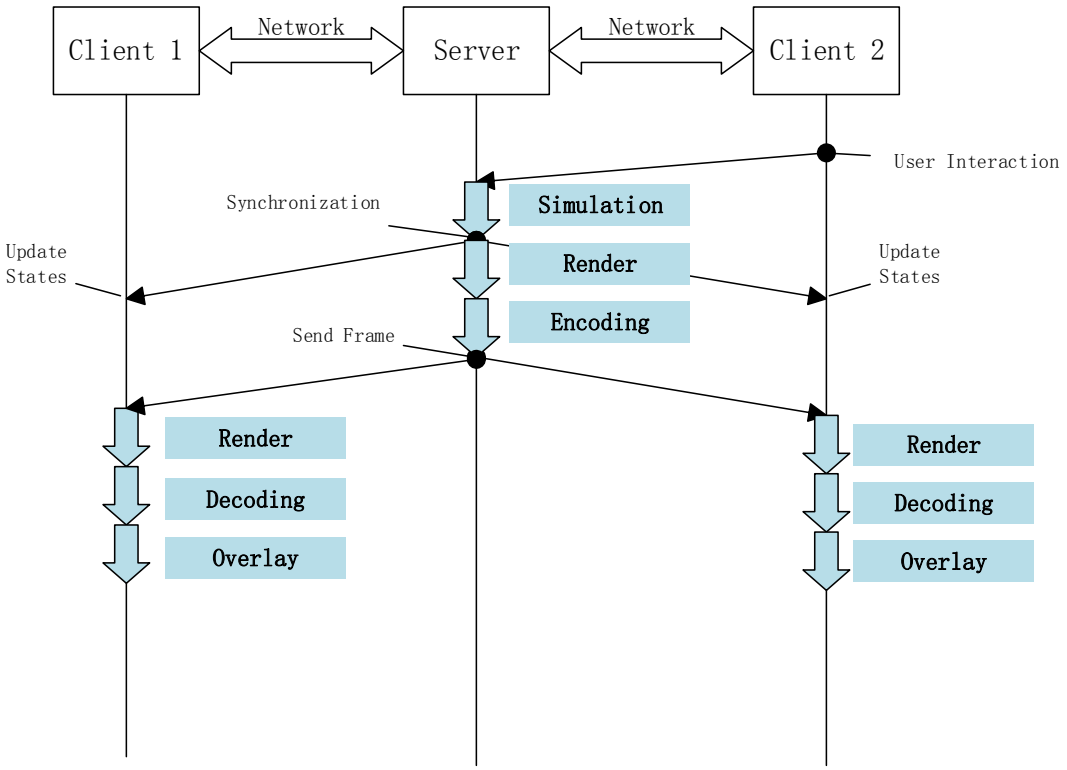


Figure 4.2: Workflow of communication between the server and the clients

view that is independent from all the other clients. When rendering the scene, the server does not render it for each client, instead, it only renders for the clients that have requested a new frame.

Fig. 4.4 indicates the workflow of clients. In each loop, the client gets commands from the server and adjusts the simulation status according to those commands. Then it sends user interaction commands to the server. In each iteration, the client will receive the high-fidelity frame from the server since it has requested in the previous iteration. The client has simplified models stored locally, it renders the scene in every iteration. The high-fidelity frames acquired from the server are overlaid upon the locally rendered frames. Once done, it sends the frame request to the server. In this way, the server does not need to render frames for all clients in every loop, instead it renders a frame when it is needed.

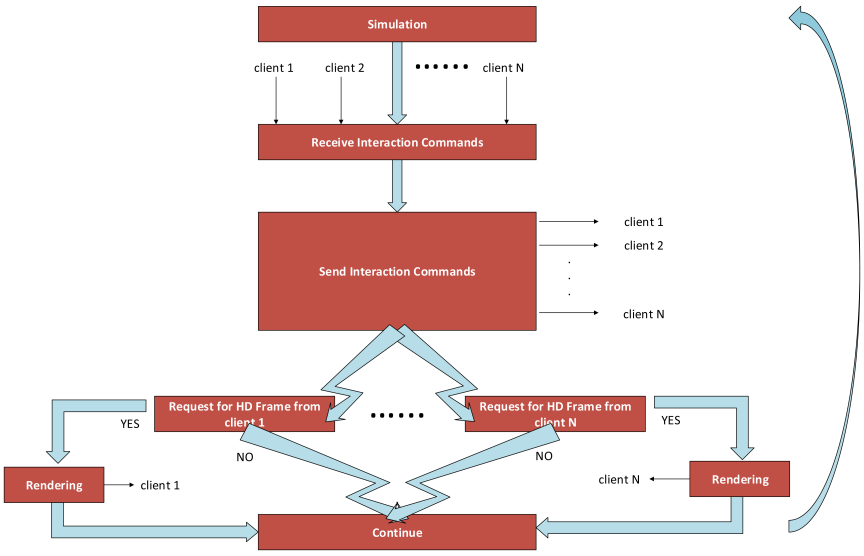


Figure 4.3: Server-side Workflow

4.1.3 Two-pass Rendering

As mentioned in section **??**, we use a two-pass rendering processing on the server side. Because frames sent to the client side contain only the key models, the server renders the high-fidelity version of those key models only. But this is not enough, since the rest models may occlude the key models. We propose a two-pass rendering process to address this issue.

As shown in Fig. 4.5, in the first pass, it renders all models except for the key models using low-fidelity version of models. Then the color buffer is cleaned and only depth information is preserved. In the second pass, only key models of their high-fidelity version are rendered, while considering the depth information obtained from the first pass rendering, therefore the occlusions are preserved in the final rendering result.

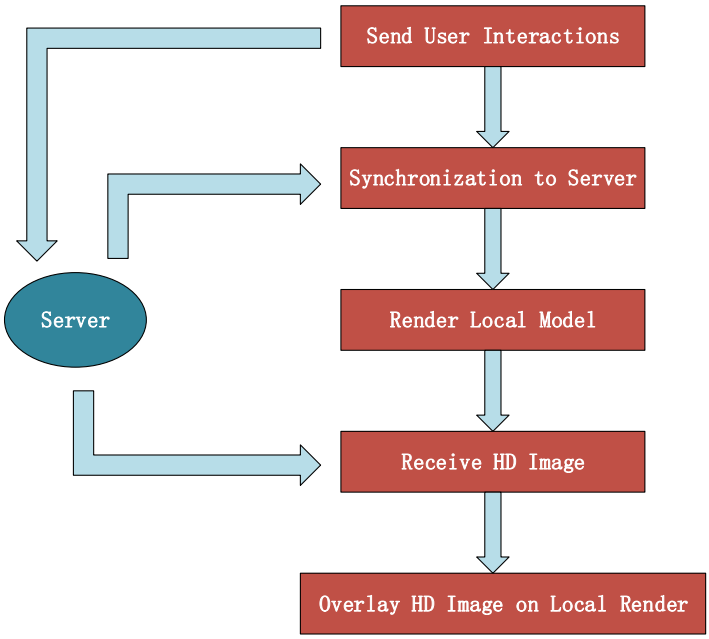


Figure 4.4: Client-side Workflow

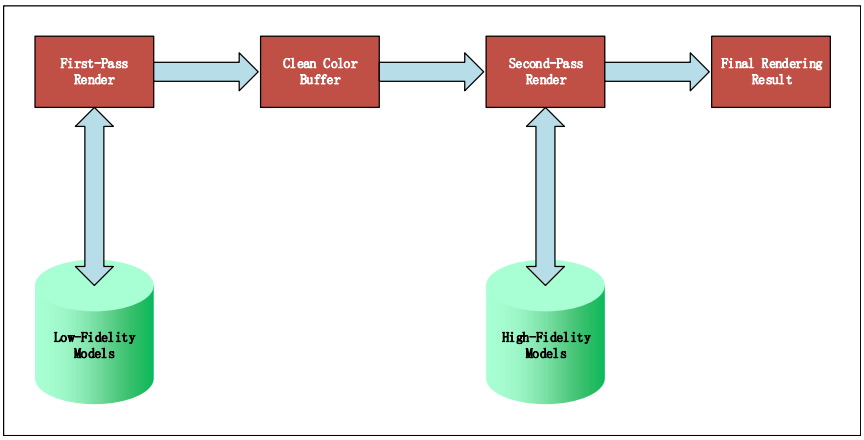


Figure 4.5: Two-pass rendering

4.2 User Study

In order to validate the use case of our proposed hybrid remote rendering technique, we designed a user study. The user study is contemplated into two parts: a) pre-trial study, b) main study. We conduct and analyze the pre-trial study before the main study.

4.2.1 Pre-Trial Study

In the main study, we vary the model resolution and test its effect on users' ability of object recognition. It requires a very careful selection of low-level display of the meshes. We would like to therefore to first run a simplified version of the experimental procedure to determine the correct level. We would like to show the participant the same three types of models but rather than the models appearing randomly at different resolutions, the same model would be shown continuously at increasing level of details where at each level, the participant would select the type of the model (dog, cat or horse). In this way we would be able to determine the level at which participants can reliably detect the type of model.

4.2.2 Main Study

In this user study, we aim at validating the Quality of Experience (QoE) improvements that can be achieved by the adoption of the proposed framework for mobile device 3D graphics applications. We use a 3D mobile application that prompts the user to respond rapidly to a visual stimulus in our evaluation. This is a stand in for a variety of applications that require the user to react swiftly to changing aspects in a 3D environment. Examples of such applications are games, flight simulators, military training systems, etc... We compare our results to conventional remote rendering and local rendering approaches.

The test application we use in our experiments depicts one or more 3D objects appearing randomly within a virtual room and disappearing within a short period. There are 3 types of objects: dogs, cats and horses. For each type, we prepared 5 different 3D models. When the participant sees an object, he/she needs to press on the object and one of three buttons that corresponds to that object, as shown in Fig. 4.6.

We run the test application 20 times for each subject. We vary the model resolution, environment resolution, delay and jitter across these runs. The table below details the test application configuration for the 20 runs. The order of the 20 configurations will be randomized across subjects to avoid biasing the results.

As shown in Tab. 4.1, we prepare four types of scenes: a scene with only low-resolution models and low-resolution environment, a scene with low-resolution models and high-resolution environment, a scene with high-resolution models and low-resolution environment and a scene with high-resolution models and high-resolution environment. We simulate three different network delays: $0ms$, $80ms$ and $120ms$. We also consider network jitter, where network jitter is the sudden fluctuation on the network.

Table 4.1: Configurations of the main user study application.

Name	Model Res	Environment Res	Delay	Jitter
APP_1	low	low	$0ms$	No
APP_2	low	high	$0ms$	No
APP_3	high	low	$0ms$	No
APP_4	high	high	$0ms$	No
APP_5	low	low	$80ms$	No
APP_6	low	high	$80ms$	No
APP_7	high	low	$80ms$	No
APP_8	high	high	$80ms$	No
APP_9	low	low	$80ms$	Yes
APP_10	low	high	$80ms$	Yes
APP_11	high	low	$80ms$	Yes
APP_12	high	high	$80ms$	Yes
APP_13	low	low	$120ms$	No
APP_14	low	high	$120ms$	No
APP_15	high	low	$120ms$	No
APP_16	high	high	$120ms$	No
APP_17	low	low	$120ms$	Yes
APP_18	low	high	$120ms$	Yes
APP_19	high	low	$120ms$	Yes
APP_20	high	high	$120ms$	Yes

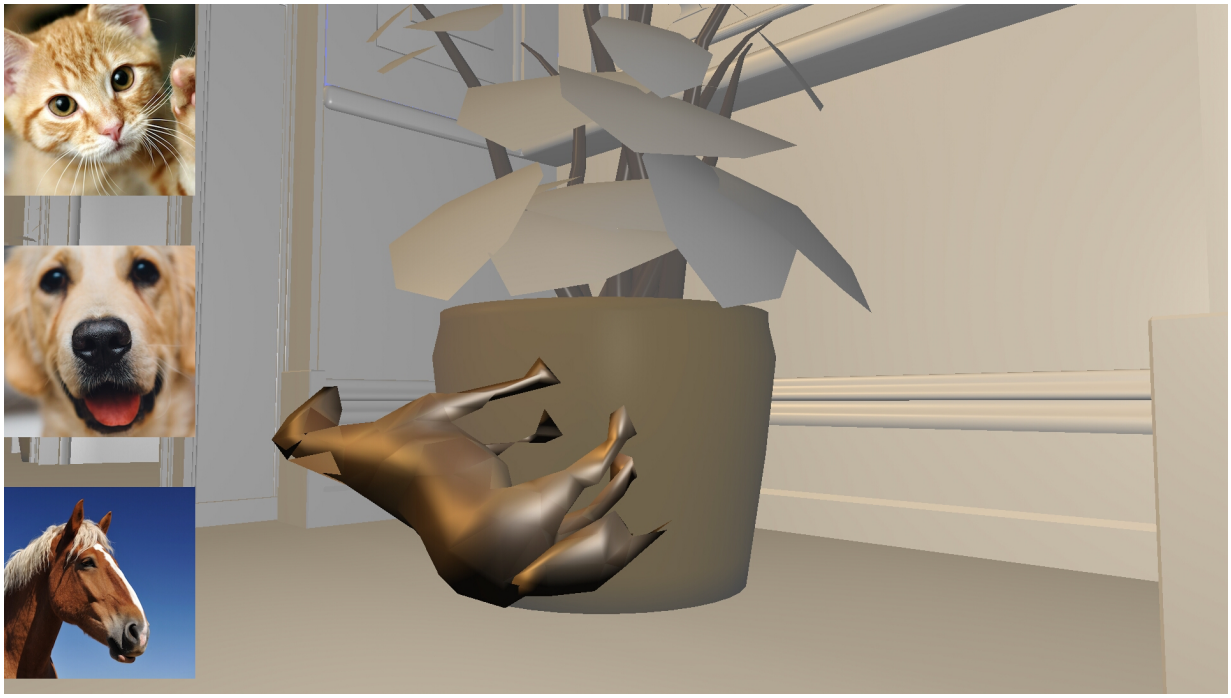


Figure 4.6: Screenshot of the user study application.

As the user interacts with the test application, we collect two types of data: QoE and user performance.

The user performance is measured by the number of targets participants successfully recognize. The QoE is measured by using the Mean Opinion Score (MOS) questionnaire. The questionnaire queries the subject about the quality and level of impairment perceived during their interaction with the test application. A 5 point Likert Scale is used to collect the answers. A score between 1 to 5 is associated with each question. The mean of all scores is calculated to obtain the MOS. The MOS is collected through a form directly integrated within the test application.

4.2.3 Data Analysis Overview

In our experiment configurations, there are four factors:

- model resolution
- environment resolution
- delay
- jitter

And there are two metrics:

Table 4.2: Factors and levels of the ANOVA analysis. The first row demonstrates the four factors, while the second row shows the levels of each factor. In the last two rows, we show whether a level of a factor is considered in the corresponding ANOVA analysis.

	Model Res		Environment Res		Delay			Jitter	
Levels	Low	High	Low	High	$0ms$	$80ms$	$120ms$	Yes	No
Four-way ANOVA	✓	✓	✓	✓	✗	✓	✓	✓	✓
Three-way ANOVA	✓	✓	✓	✓	✓	✓	✓	✗	✗

- MOS
- user performance

In order to understand how each factor affects MOS and user performance, we use analysis of variance (ANOVA) to interpret our experimental data.

We could use a four-way ANOVA test to investigate all the four factors. However the factor combination of delay and jitter is not complete, since jitter does not exist when delay is $0ms$. So we exclude the delay level $0ms$ for the four-way ANOVA test. As shown in Table 4.2, the four-way ANOVA considers all the four factors and their levels, expect for the level $0ms$ of delay. We also analyze the effect of factor delay and its interaction with model resolution and environment resolution. So we resorted to three-way ANOVA analysis for the combination of delay, model resolution and environment resolution.

In Table 4.2, we can see that all levels of factors model resolution, environment resolution and delay are considered in the three-way ANOVA. However, because jitter can not exist when delay is $0ms$, we do not take the factor jitter into consideration in the three-way ANOVA.

Before actually performing the tests, we draw eight hypotheses to be tested, see Table. 4.11

4.2.4 Results and Discussion

Pre-Trial Study

Tab. 4.4 demonstrates the data collected from the pre-trial study. Each row shows the number of faces needed by each participant to recognize the model. Because there might be outliers in the data, we select the second highest as the number of faces of the fidelity models. The results are used to decide the number of faces of low-resolution meshes in the main study.

Main Study

First, we analyze the grand mean of each factor and each level. The grand mean is calculated by summarizing all samples that uses a specific level of a factor. For instance,

Table 4.3: Factors and levels of the ANOVA analysis. The first row demonstrates the four factors, while the second row shows the levels of each factor. In the last two rows, we show whether a level of a factor is considered in the corresponding ANOVA analysis.

H_{01}	Different model resolutions do not affect the user performance of recognizing objects
H_{02}	Different environment resolutions do not affect the user performance of recognizing objects
H_{03}	Different network delays do not affect the user performance of recognizing objects
H_{04}	Existence of jitter does not affect the user performance of recognizing objects
H_{05}	Different model resolutions do not affect the quality of experience
H_{06}	Different environment resolutions do not affect the quality of experience
H_{07}	Different network delays do not affect the quality of experience
H_{08}	Existence of jitter does not affect the quality of experience

Table 4.4: Data collected from the pre-trial study. The rows represent various models used in the study, while columns are data collected from different participants. Integers show the number of faces where the participant recognize the model. However, in one case, one of the participants does not recognize the model after all, it is denoted by the dash.

Participant #1	Participant #2	Participant #3	Participant #4	Participant #5
148	22	57	51	111
162	57	422	22	422
122	383	134	22	238
75	148	-	69	134
162	51	32	22	134
1095	179	464	1940	464
995	1603	1325	32	422
75	162	148	91	148
75	83	216	122	196
464	262	748	216	562
422	1204	422	288	383
47	101	134	22	134
22	47	510	29	238
238	22	348	317	162
69	83	75	75	51

Table 4.5: Grand means of user performance for each factor. Grand means of both ANOVA tests are shown. It is measured by the number of targets that the participants successfully recognized, where higher value represents better performance. The first row represents the factors and their levels. The numbers in the second and third rows are shown with respect to different levels. Numbers of red color show the grand means of factors that are statistically significant.

	Model Res (low/high)	Environment Res (low/high)	Delay ($0ms/80ms/120ms$)	Jitter (yes/no)
Four-way ANOVA	5.4/6.5	6.1/5.8	-/6.0/5.9	6.2/5.8
Three-way ANOVA	5.7/6.5	6.2/6.0	6.0/6.1/6.2	-

every participant repeated the experiment for 20 times with various configurations, in which there are 10 configurations use low-resolution models. To calculate the grand mean of low model resolution, we summarize the data of those 10 experiments of each participant. So the total number of samples is 150.

Table. 4.5 shows the grand means of user performance of the factors. Because we have done a four-way and a three-way ANOVA tests, the grand means obtained from both tests are shown. User performance is measured by the number of targets that the participants successfully recognized, where higher scores represent better performance. The first row represents the factors and their levels. The numbers in the second and third rows are shown with respect to different levels. For instance, the numbers 5.4/6.5 indicates that the average number of targets that the participants correctly recognized is 5.4 for low model resolution, and 6.5 for high model resolution. Since jitter does not exist with delay of $0ms$, the grand means are only available in four-way ANOVA.

It is shown that user performance varies with different levels of each factor. However, not all factors have a significant effect on user performance. We can see that for both four-way and three-way ANOVA, with high model resolution, participants are able to recognize more targets. The grand means for the other three factors do not show enough difference between levels of a factor. The differences are either not large enough or do not agree in the four-way ANOVA and three-way ANOVA tests.

Table. 4.6 shows the grand means with respect to MOS. As same as the Table 4.5, the second row and third row show the results obtained from the four-way ANOVA and three-way ANOVA tests, respectively. The means are measured by a 5 point Likert Scale, where larger values indicate better quality of experience. In each cell, we show multiple means that are calculated for different levels of a factor, e.g. 2.3/2.5 in the first cell indicates that the average MOS of low model resolution in four-way ANOVA is 2.3, and the average MOS of high model resolution is 2.5.

From the table, we can see that reducing network delay clearly improves the quality of experience, since lower delays give us larger MOS scores. Note that delay is not the only factor that improves the quality of experience. With both four-way ANOVA and three-way ANOVA analysis, the participants score higher with high-resolution models over low-resolution models (i.e. 2.5 over 2.3 in four-way ANOVA and 2.8 over 2.6 in three-way ANOVA). However, compared with the differences among levels of network delay, differences between low model resolution and high model resolution are relatively small.

Table 4.6: Grand means of quality of experience for each factor. Results of both ANOVA tests are shown. It is measured by MOS, where higher value represents higher quality of experience. The first row represents the factors and their levels. The numbers in the second and third rows are shown with respect to different levels. Numbers of red color show the grand means of factors that are statistically significant.

	Model Res (low/high)	Environment Res (low/high)	Delay ($0ms/80ms/120ms$)	Jitter (yes/no)
Four-way ANOVA	2.3/2.5	2.3/2.4	-/2.5/2.2	2.5/2.3
Three-way ANOVA	2.6/2.8	2.8/2.7	3.1/2.6/2.4	-

Next, we analyze the variance for both metrics, i.e. user performance and MOS. To determine whether any of the differences between the means are statistically significant, we compare the p-values to a significance level to accept or reject the null hypothesis. We choose a significance level of 0.05 that usually works well. The significance of 0.05 implies a 5% risk of concluding that a factor is significant when it is actually not.

Table. 4.7 demonstrates the results of analysis of variance (ANOVA) with user performance as the metric. The last column is the p-values. As mentioned in Sec. 4.2.3, we also did a three-way ANOVA test with factors model resolution, environment resolution and network delay. Table 4.8 shows the results.

It is obvious that only the factor model resolution has a p-value less than 0.05 in both ANOVA tests. This indicates that only the model resolution is significant in terms of user performance. To explore the interaction between a factor pair or among multiple factors, we also calculate the p-values for those interactions. This kind of calculations are performed for every combination of factors, including combination of two factors, combination of three factors, and combination of four factors if plausible. From Table. 4.7 and Table. 4.8, we can see that there is no combination whose p-value is larger than 0.05. This indicates that these factors are independent from each other in terms of user performance.

Table. 4.9 demonstrates the results of four-way ANOVA with Mean Option Score (MOS) as the metric, and Table. 4.10 shows the results of three-way test. Different from the ANOVA of user performance, with MOS as the metric, model resolution is not significant, instead, the factor delay is statistically significant. Because the p-value of delay is 0.0281 in the four-way ANOVA test and it is 0.0018 in the three-way ANOVA test. Both are less than the significance level of 0.05. It is worth mentioning that although the p-value is not less than 0.05, it is a very small value (i.e. 0.087). It is close to the significance level of 0.05.

Further more, we analyze the interaction for factor combinations. Similar to what we did in the analysis for user performance, we calculate the p-values for all combinations. The results demonstrate that there is no interaction between the four factors.

From the p-values obtained above, we can draw the conclusion that the factor Model Resolution is the only factor that affects the user performance significantly, while the factor Delay is the only factor that affects the quality of experience significantly.

Now we can decide whether accept or reject the null hypotheses proposed in Sec. 4.2.3, according to the analysis results. There are two null hypotheses we reject, i.e. H_{01} and H_{07}, since they do have significant effect on user performance or quality of experience.

Table 4.7: Four-way ANOVA of user performance.

Source	Sum Sq.	d.f.	Mean Sq.	F	p-value
Model Res	73.7	1	73.7	14.9	0.0001
Environment Res	6.34	1	6.34	1.28	0.2589
Delay	0.7	1	0.7042	0.14	0.7063
Jitter	8.44	1	8.44	1.71	1.1929
Model Res * Environment Res	0	1	0	0	0.9769
Model Res * Delay	0.7	1	0.7	0.14	0.7063
Model Res * Jitter	1.2	1	1.2	0.24	0.6223
Environment Res * Delay	1.2	1	1.2	0.24	0.6223
Environment Res * Jitter	5.1	1	5.1	1.03	0.3109
Delay * Jitter	2.6	1	2.6	0.53	0.4689
Model Res * Environment Res * Delay	14.5	1	14.5	2.93	0.0882
Model Res * Environment Res * Jitter	1.84	1	1.84	0.37	0.5429
Model Res * Delay * Jitter	0.2	1	0.2	0.04	0.8392
Environment Res * Delay * Jitter	15.5	1	15.5	3.13	0.0781
Model Res * Environment Res * Delay * Jitter	6.34	1	6.34	1.28	0.2589
Error	1108.27	224	4.95		
Total	1246.66	239			

Table 4.8: Three-way ANOVA of user performance.

Source	Sum Sq.	d.f.	Mean Sq.	F	p-value
Model Res	30.42	1	30.42	6.33	0.0128
Environment Res	0.56	1	0.56	0.12	0.7342
Delay	1.2	2	0.6	0.12	0.8826
Model Res * Environment Res	0.2	1	0.2	004	0.8385
Model Res * Delay	2.711	2	1.36	0.28	0.7544
Environment Res * Delay	4.58	2	2.29	0.48	0.6217
Model Res * Environment Res * Delay	5.73	2	2.87	0.6	0.5517
Error	806.8	168	4.8		
Total	852.2	179			

Table 4.9: Four-way ANOVA of MOS.

Source	Sum Sq.	d.f.	Mean Sq.	F	p-value
Model Res	3.27	1	3.27	2.95	0.0871
Environment Res	1.07	1	1.07	0.96	0.3271
Delay	5.4	1	5.4	4.88	0.0281
Jitter	3.27	1	3.27	2.95	0.0871
Model Res * Environment Res	0.42	1	0.42	0.38	0.54
Model Res * Delay	0.02	1	0.02	0.02	0.9024
Model Res * Jitter	0.42	1	0.42	0.38	0.54
Environment Res * Delay	1.35	1	1.35	1.22	0.2704
Environment Res * Jitter	2.02	1	2.02	1.82	0.1783
Delay * Jitter	0.82	1	0.82	0.74	0.3911
Model Res * Environment Res * Delay	0.27	1	0.27	0.24	0.6239
Model Res * Environment Res * Jitter	0.6	1	0.6	0.54	0.4622
Model Res * Delay * Jitter	0.6	1	0.6	0.54	0.4622
Environment Res * Delay * Jitter	3.27	1	3.27	2.95	0.0871
Model Res * Environment Red * Delay * Jitter	3.75	1	3.75	3.39	0.0669
Error	247.73	224	1.106		
Total	274.25	239			

Table 4.10: Three-way ANOVA of MOS.

Source	Sum Sq.	d.f.	Mean Sq.	F	p-value
Model Res	1.61	1	1.61	1.2	0.2743
Environment Res	0.45	1	0.45	0.34	0.5623
Delay	17.48	2	8.74	6.55	0.0018
Model Res * Environment Res	0.05	1	0.05	0.04	0.8468
Model Res * Delay	1.88	2	0.94	0.7	0.4964
Environment Res * Delay	0.43	2	0.22	0.16	0.8503
Model Res * Environment Res * Delay	1.23	2	0.62	0.46	0.6309
Error	806.8	168	4.8		
Total	852.2	179			

Table 4.11: Factors and levels of the ANOVA analysis. The first row demonstrates the four factors, while the second row shows the levels of each factor. In the last two rows, we show whether a level of a factor is considered in the corresponding ANOVA analysis.

H_{01}	Different model resolutions do not affect the user performance of recognizing objects	reject
H_{02}	Different environment resolutions do not affect the user performance of recognizing objects	accept
H_{03}	Different network delays do not affect the user performance of recognizing objects	accept
H_{04}	Existence of jitter does not affect the user performance of recognizing objects	accept
H_{05}	Different model resolutions do not affect the quality of experience	accept
H_{06}	Different environment resolutions do not affect the quality of experience	accept
H_{07}	Different network delays do not affect the quality of experience	reject
H_{08}	Existence of jitter does not affect the quality of experience	accept

The user study supports the idea of only rendering and sending models of importance in remote rendering applications. On one hand, not rendering the environment in high resolution has no significant effect on users' object recognition. In gaming, training, medical visualization and other areas where remote rendering has been used, low-resolution environment does not prevent users from completing the task. On the other hand, without rendering and sending the environment to clients, it saves the rendering time and network usage, thus it is able to reduce the delay that has a significant effect on the quality of experience.

4.3 Conclusions

We propose a hybrid remote rendering framework for mobile applications. It uses a client-server model, where the server is responsible for rendering high-fidelity models, encoding and sending the rendering frames to the client. The client uses low-fidelity models to render frames locally and overlays the frames received from the server onto its local rendering frames. Only models of interest are rendered on the server and sent to the client. In this way, the client is able to display rendering effects that are not available with mobile graphic devices, with minimum bandwidth requirement. Moreover, since low-resolution models are stored and rendered on the client side, our framework is able to adapt itself to different network conditions, even if the network is totally unavailable.

We conduct a user study on the factors involved in the proposed framework, model resolution, environment resolution, network delay and jitter. We developed an experimental application that requires the participants to recognize objects with different factor configurations. 15 volunteers were recruited in this study and we use ANOVA to analyze and interpret the results. It shows that model resolution affects users' ability to recognize objects and network delay plays an importance role in quality experience. In real production environment, the network condition often does not satisfy the remote rendering applications. Therefore trading off between rendering quality and network delay is essen-

tial. However, the existing remote rendering applications have no such capacity. The most important contribution of the proposed framework is adding such capacity.

Chapter 5

Remaining Work: Realtime Remote Training with Augmented Reality

5.1 Overview

As mentioned in Section. 1.1, we are going to develop a training framework with augmented reality hardware. With this framework, people are able to perform manufacturing or assembly training remotely with augmented reality. A live video is captured by augmented reality glasses for the trainer and each trainee. The video is sent to the server in real time. The server acts as a middle ware between the trainer and trainees. It is responsible for estimating the pose of each component. Then high quality virtual components will be rendered on the augmented reality glasses even the devices have no graphics capacity, since the virtual components are rendered in a high quality on a dedicated server. The framework works in an on-line manner, the users can see the result in real-time, there is no need to wait for the entire video to be captured completely.

Fig. 5.1 shows one of the two parts of the scenario described above. This part depicts how the trainer gives instructions to the trainees. The upper part shows the setup of the trainer side. The grey gear represents the original pose of the component #2, while the black and oblique gear shows the pose of it after some operations. The trainer wears augmented reality glasses that are equipped with a camera. The camera is used to capture videos and send them to the server. The server estimates the pose of each component in every video frame it receives, with the CAD of all components known beforehand. If the AR glasses is not available, a mobile device with a camera also works, e.g. a mobile phone or a tablet.

The lower part of Fig. 5.2 shows the setup of the trainee side. There are same components in the training environment of each trainee as on the trainer side. The trainees also wear augmented reality glasses. Similar to the trainer side, the glasses are equipped with a camera that records videos and sends them to the server. With the video received from the trainee side, the server is able to calculate the pose of components related to each trainee. As mentioned above, the server also estimates how the trainer moved every component on

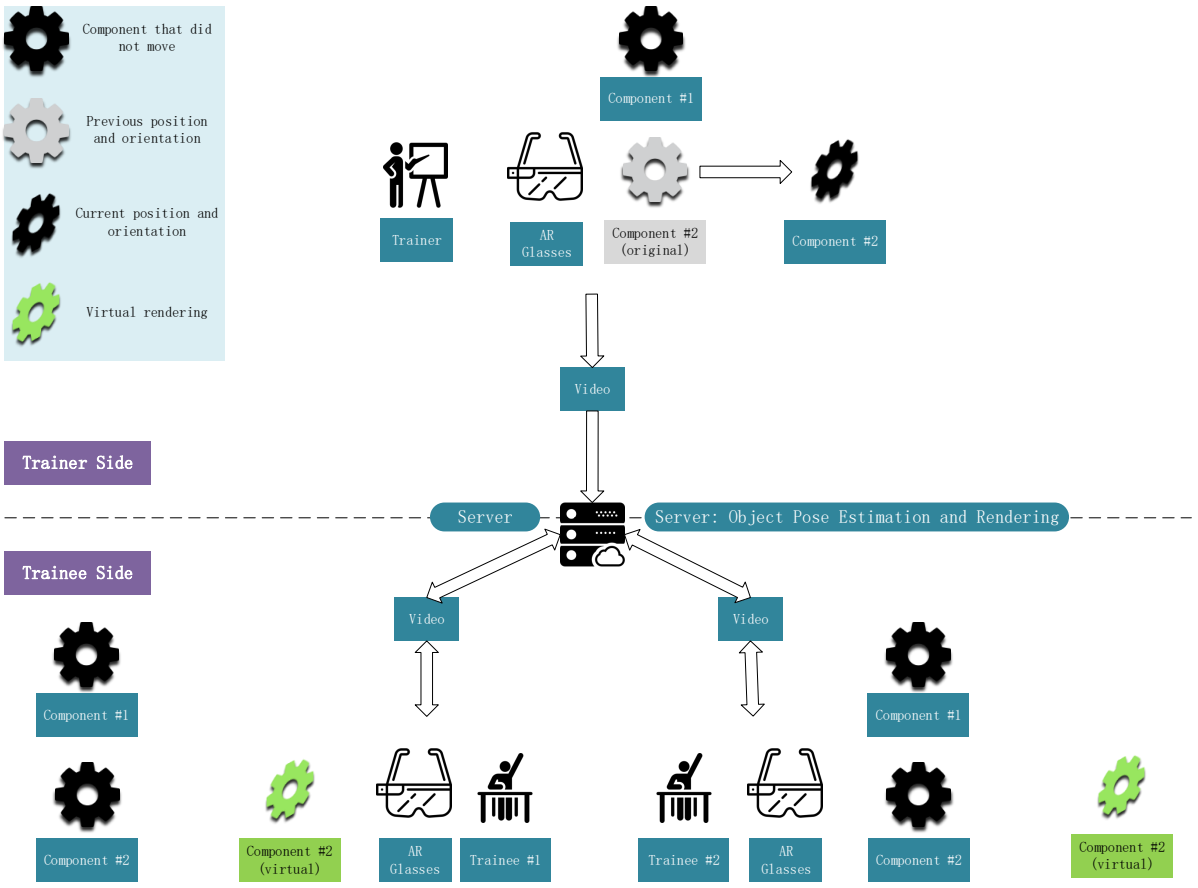


Figure 5.1: Training Scenario Part 1. This figure shows how the operations made by the trainer are rendered on the trainee side. The gears represent the components in the training scenario. The black gears denote the real components, while grey ones are their previous poses and the green gears demonstrate the virtual component rendered by the augmented reality devices. Moreover, the oblique gears represent the current pose after a translation and a rotation.

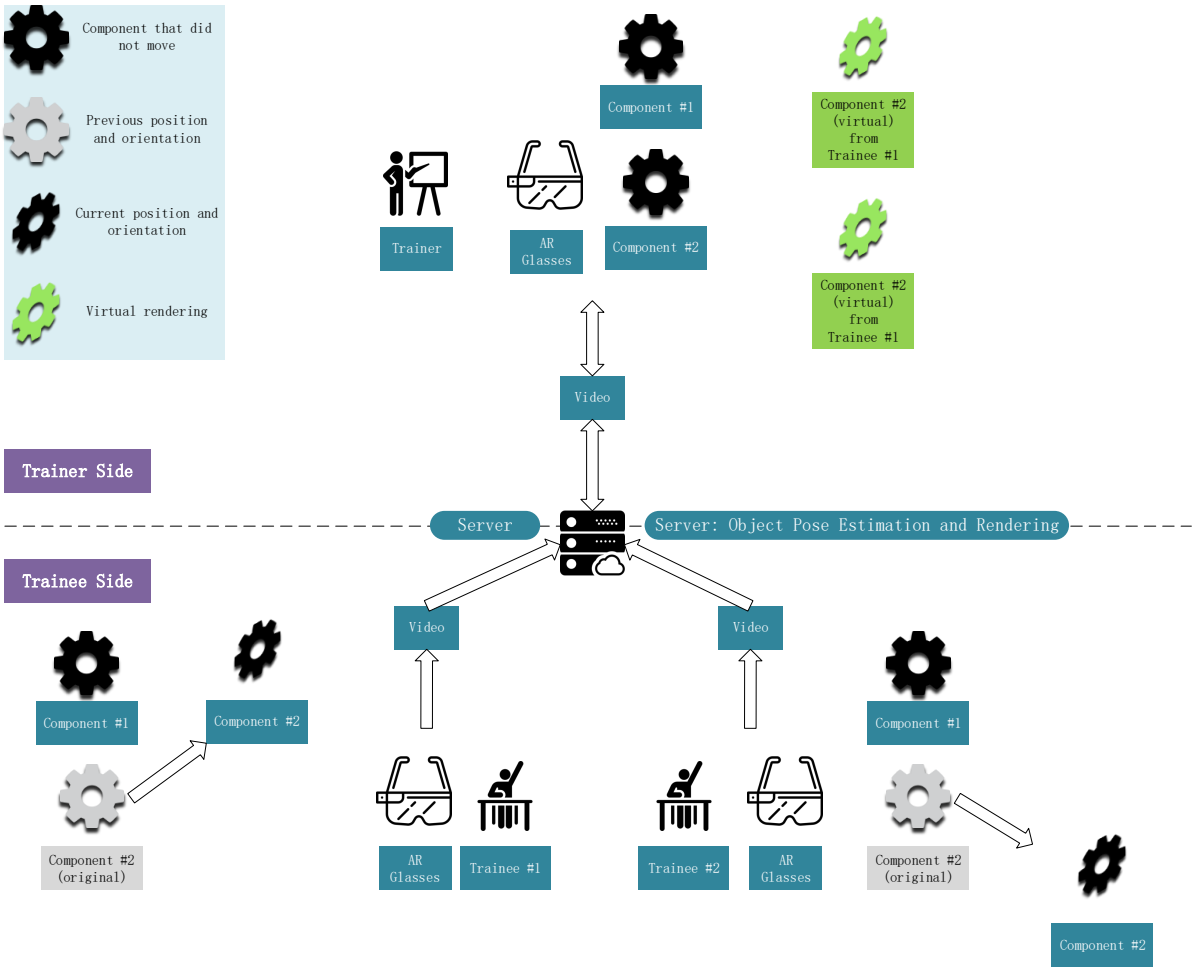


Figure 5.2: Training Scenario Part 2. This figure shows how the operations made by the trainees are rendered on the trainer side. The gears represent the components in the training scenario. The black gears denote the real component, while grey ones are their previous poses and the green gears demonstrate the virtual component rendered by the augmented reality devices. The oblique gears represent the current pose after a translation and a rotation.

his or her side. Then it renders the 3D model of each component that has been moved, according to each trainee's view and sends it to the trainees' glasses as videos. The black gears represent the real components on the trainee side, while the green gears show the changes made by trainer.

Fig. 5.2 demonstrates the second part of the scenario. It shows how the trainer sees the operations made by the trainees. After receiving instructions from the trainer, each trainee perform their own operations. The operations are recorded by the camera on the augmented reality glasses as a video. Similar to the trainer side of the first part, the server estimates how each component is moving. The estimations are performed for each video from the trainees. Moreover, the server also receives a live video from the trainer, and calculates the pose of every component on the trainer side. Then the server renders the components according to the operations made by each trainee, sends the virtual components to the trainer and displays them with the augmented reality glasses. Note that a trainee may behave differently from each other, so several versions of each component are sent to the trainer. As shown in Fig. 5.2, trainee #1 moves component #2 up, while trainee #2 moves component #2 down. Thus there are two versions of the virtual component shown in the trainer's view (marked green).

The components on the trainer side and the trainee side must be the same components, otherwise the training makes no sense. However, they do not need to be at the same location or orientation, since the server is responsible to track each component on each side.

5.2 Communication Schema Design

As shown as Fig. 5.3, the system consists of four services: control service, pose estimation service, rendering service and the encoding service. The control service is responsible for controlling the entire process. First of all, it sends requests to the pose estimation service to perform pose estimation for each client, including the trainer and all the trainees. Note that the pose estimation service is a non-blocking service. It receives the video captured from each client independently and updates it estimation result right after receiving every frame from every client. In this way, it maintains a database of the relative pose of each object for each client. Every time it receives the request from the control service, it sends the current database to the control service.

Once the control service gets the result from the pose estimation service, it sends the pose estimation result and the rendering request to the rendering service. When the rendering service finish rendering the models of interest for all the clients according to their point of view, it sends back the rendering result to the control service. Note that the models of interest include the models of the components whose state (i.e. position and orientation) has been changed by the trainer but not changed accordingly by the trainees. In another word, it means the components which the trainer and the trainees need to pay attention to.

Last, the control service sends the rendering result and encoding request to the encoding

service. Once the encoding service has accomplished encoding, it sends the encoded frames to each client. The encoding service is a non-blocking service. The control service does not need to wait for the response from it to continue the process.

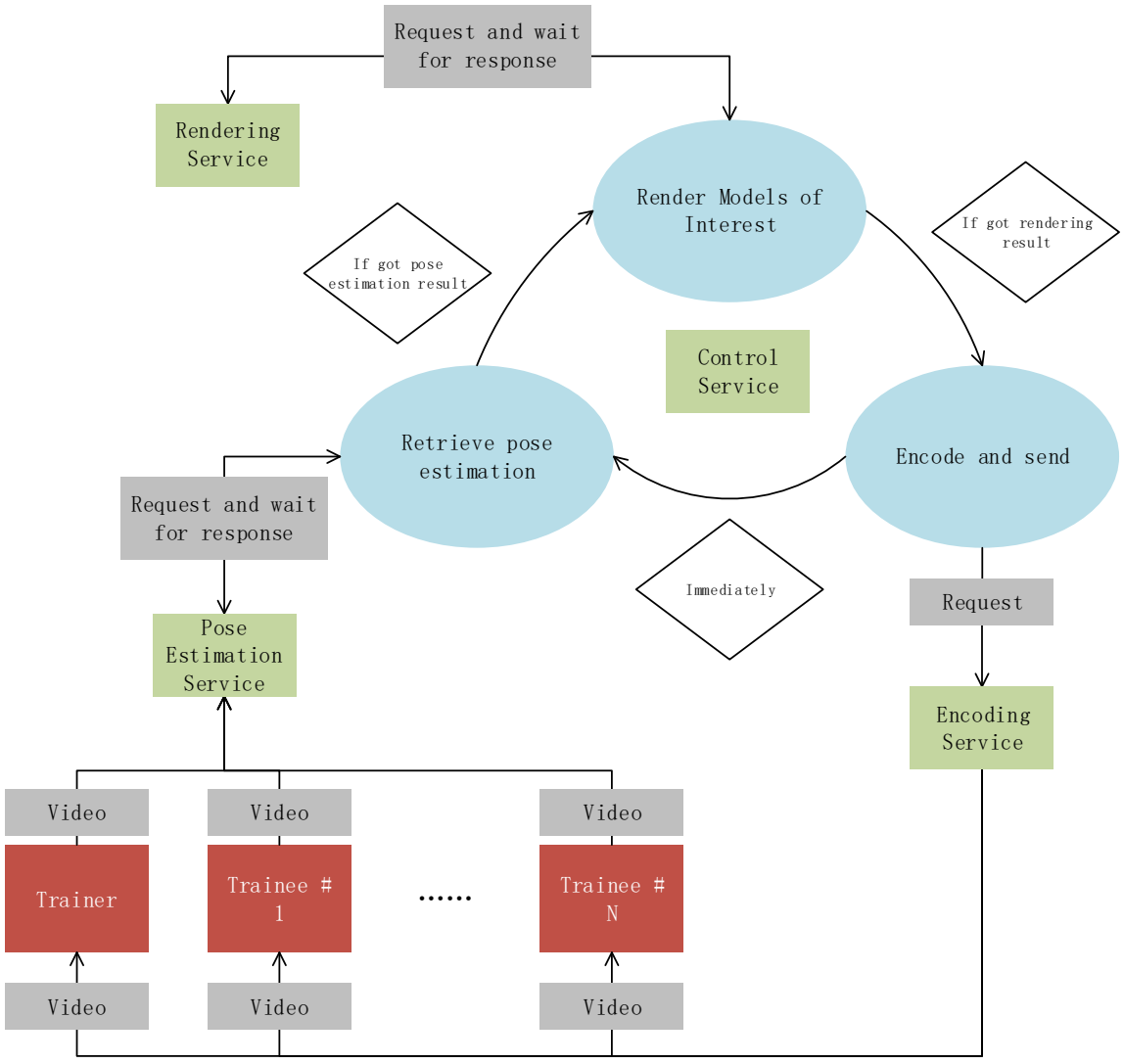


Figure 5.3: Communication Schema

As mentioned in Chap. 4, our remote rendering method includes two types of models: high-fidelity models and low-fidelity models, where the high-fidelity models are stored on the server side and the low-fidelity models are stored on the client side. Thus even without the rendering service, the clients are able to render the models if they have the basic rendering capacity. In our schema, the only blocking service is the rendering service. For real-time performance, we set a time limit of 1/30 second for the rendering service. If it does not accomplish the rendering within the time limit, the control service will send the pose estimation result to the clients and let them do the rendering themselves.

5.3 Pose Estimation of Multiple Objects

The first step towards our proposed framework is to estimate the poses of the objects of interest, namely the position and orientation of each object over time. We use a model-based algorithm to estimate the poses. With model-based pose estimation algorithms, the 3D structure of the objects of interest must be known beforehand. It is often the case in industrial training [**?**].

Some model-based approaches use edge or point features associated with the 3D models for estimating the pose [**?**, **?**, **?**, **?**]. However, there are two major disadvantages with feature-based methods. First, they struggle with motion blur and are prone to local minima especially with cluttered backgrounds. Second, using point-based features also requires the objects' surfaces to be sufficiently textured, which significantly limits the variety of suitable objects.

Recently, region-based pose estimation methods have emerged, which are mainly based on statistical level-set segmentation approaches. The main advantage is that they do not require sufficiently textured objects and only reply on structure of the objects. However, this category of approaches only work in application scenarios where it is undesirable or even impossible to modify the objects. In another word, they require the objects to be rigid. It is often the case in industrial or manufacturing training.

In [**?**] the authors present PWP3D, the first region-based approach that achieves real-time frame rates (20-25 Hz) using GPUs by solving the pose estimation similar to the variational approach suggested in [**?**] but using level-set functions instead of separately integrating over the foreground and background region to simplify computations and make it real-time capable. Recently, another improved PWP3D version was proposed, which runs at 30 Hz on a mobile phone [**?**]. Tjaden et al. built an algorithm based on PWP3D, which improves convergence properties, especially for rotational motion [**?**]. Also, the described implementation that uses the GPU only for rendering purposes, performs the rest computations on the CPU to achieve frame rates of about 50-100 Hz when tracking a single object on a commodity laptop.

Our method uses the work proposed by Tjaden et al. [**?**] in our pose estimation service, where the object segmentation and pose estimation are performed in an interleaved fashion for each camera image. However, the approach uses a level-set segmentation method and requires manual initialization of the segmentation, which limits its usage in real applications. We proposed an automatic video object segmentation method, as described in Chap. 3. In our work, the object segmentations are initialized automatically and it works with multiple objects. Compared with other state-of-the-art automatic video object segmentation methods, our approach has the advantage that it runs in real-time. Moreover, the synthetic silhouette that is generated with the models can be used as a ground truth segmentation, which will be used to improve the foreground and background segmentation results.

5.4 Limitations

However, the proposed framework still has several limitations. First, the 3D models of objects involved in the training must be known beforehand, since the techniques we use in pose estimation is a model-based method. The reason why we use a model-based pose estimation method is that this kind of methods are typically more accurate than those approaches that estimate the 3D structure and pose at the same time. Another advantage of using a model-base pose estimation method is that it does not require the presence of markers. However, our proposed framework aims at manufacturing or assembly scenarios, in which the CAD models are typically known beforehand. In some other training scenarios, extra effort is needed to obtain the 3D structures of the objects involved. Second, the proposed framework does not work with non-rigid bodies. For instance, in surgery training, the organs are deformable, it is not enough to only track the translation and orientation of an organ.

Chapter 6

Conclusion

Augmented reality is an emerging direction in industry and research, which has a broad use in various fields, e.g. games, communication, medicine and training. We focus on the use of it in the area of training in this proposal. Manufactures, such as Boeing and BMW, have already leveraged the advantages of augmented reality in their training department. Great improvements have been made by using AR compared with traditional resorts.

However, there are two main disadvantages with the existing AR training systems. On one hand, most of existing methods generally guide the user through a fixed series of steps and the digital content is prepared beforehand. On the other hand, during the training, the trainers cannot collect feedbacks from the the trainees.

To solve the two disadvantages, we propose a framework for augmented reality training. It contains two essentials: a) an real-time object pose tracking system that tracks the translations and orientations of multiple objects, b) a remote rendering system that delivers high quality 3D rendering in real-time.

In this proposal, we present three works:

- a method for extracting foreground objects from video and its application to content-aware video compression
- a remote rendering method that minimizes the bandwidth and computing power
- an augmented reality training framework that enables the trainers to provide instructions to the trainees remotely and collect feedback, in real-time

The method for extracting foreground objects from videos is presented in Section 3, which is used in the real-time object pose tracking system. The proposed method is fully automatic, fast, and does not make restrictive assumptions about object motions. In experiments on standard data sets, the proposed approach achieves comparable results to state-of-the-art video object segmentation methods but our method is much faster. It is an essential building block in an real-time object pose tracking system. We also demonstrate an application of the proposed method to content-aware video compression.

The second work, hybrid remote rendering method, is resented in Section 4. Although the field of remote rendering has been studied for decades and the implementations have been used in many areas, such as gaming and virtual tour, researchers primarily focused on remote rendering of the entire frame. We pay attention on the importance of models. In another word, we render object with higher importance with high-resolution models, and render objects with lower importance with low-resolution models. This flexibility gives the remote rendering system a better adaptability to various network environments and device capacities, which empowers our AR training system to adapt real production scenarios.

The remaining work is described in Section 5. It depicts the overall structure of the framework, the communication design and the pose estimation of multiple objects. We have two main design objectives: a) trainers should be able to guide trainees in an manufacture or assembly task in real time, b) trainers should be able to collect feedback from trainees in real time. To achieve the design objectives, we use a client-server structure in our framework. There are two types of clients in our system: a) trainer, b) trainee. On the client side, we use AR glasses or mobile devices to capture videos of the environment. The captured videos are sent to the server that is responsible for 3D object pose estimation and rendering. The two design objectives are also main advantages of our proposed framework over existing AR training system.

6.1 Schedule

The following schedule is given in terms of the estimated time required following the completion of this proposal.

Sept. 2017 (1 month) Finish and submit the paper on hybrid remote rendering.

Oct. 2017 to Nov. 2017 (2 months) Finish Pose Estimation of Multiple Objects.

Dec. 2017 to Jan. 2018 (2 months) Finish the designed Communication Schema and the implementation of our proposed system.

Feb. 2018 to Apr. 2018 (3 months) Finish evaluating our proposed system.

May 2018 to Aug. 2018 (4 month) Finishing writing thesis.