Towards Augmented Reality Training

by

Lu Sun

Thesis proposal 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 the existing augmented reality applications, our proposed framework has three advantages: 1) the proposed method acts as an interactive system between the trainer and the trainee, while the existing methods typically guide the trainee through a fixed series of steps; 2) the proposed method does not need markers to estimate the 3D structure of the models and the environment; 3) the proposed method 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 in 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, the algorithm Graph Cuts is 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 the 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 3D structure estimation for the models and the environment in a training scenario.

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 framework for applications on mobile devices. In our remote rendering approach, we adopt a client-server model, where the server is responsible for rendering high-fidelity models, encoding the rendering results and sending them to the client, while the client renders low-fidelity models and overlays the high-fidelity frames received from the server on its rendering results. With this configuration, the client is able to keep functioning regardless of network condition and bandwidth. Moreover, to minimize the bandwidth requirements, only key models are rendered in high-fidelity mode.

Finally, the last sub-goal is to combine the former two sub-goals to develop an AR framework for training scenarios. 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 [57, 28, 35, 92]. 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 [27]. 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 [19]. 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 [35, 92]. The development of markerless 3D object recognition approaches [86] 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 [61]. 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 [61, 54,

90, 9, 85]. Most methods use motion cues to initialize the object segmentation. A common method for motion detection is background subtraction with mixture of Gaussians [39, 100]. 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, healthcare 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 an encoder to serve every connected client. The models are rendered on the server side and encoded in video frames streamed from the server to the client [42, 51, 55, 13, 76].

However, CMR systems often require very high network bandwidth which is seldom available and suffer from interaction latency [15]. Various attempts have been made to improve such systems. Boukerche and Pazzi [7] 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. [74] 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 [7] accelerates panning interactions, but a new environment map needs to be generated for a novel viewpoint, 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. [74].

We propose a remote rendering system that minimizes the network bandwidth requirements 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 the user is interacting with. These models can be identified from interaction information sent to the server. Additional key models may be specified in advance by the developers based on application-specific criteria.

The proposed 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 and encoding time by only rendering and encoding the key models in high-fidelity mode and rendering the rest in low-fidelity mode without encoding.

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 [29], 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. [92] 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. [28] 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. [86] 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 [61]. 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 [61, 54, 90, 9, 85]. Most methods use motion cues to initialize the object segmentation. A common method for motion detection is background subtraction with mixture of Gaussians [39, 100]. 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 [48, 84]. We adapt the technique of Tao and Krishnaswamy [84] 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 [31, 8] 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 [77] before classification, see Section 3.1.3.

We evaluate our video object segmentation approach on a common dataset [47] 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 [30] and many approaches use background subtraction to this end [96, 18]. Classic background subtraction methods model the appearance of the background at each pixel and label the pixel that change rapidly to be foreground [37, 39, 100]. These method typically assume a stationary or slowly panning camera. More recently, optical flow is also used [54, 61, 97, 90]. 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 [61]. 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 [61, 90].

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 [61, 90, 60]. Instead of naive superpixel voting schemes, researchers often use some optimization approaches to decide if a superpixel belongs to the foreground or the background [61, 90, 60]. However, most of the superpixel methods are computationally intensive which prevents their use in real-time video object extraction [1]. A fast superpixel method was proposed by Siva and Wang [77]. 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 [9, 60, 66]. 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 [9, 60], while the work of Ochs et al. [66] 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. [45] shows the potential of using shape matching in object extraction. The method first identifies object-like regions in any frame [24], 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 [21, 85]. The work of Taylor et al. [85] 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 [61] and Wang et al. [90] are closely related to ours. Papazoglou and Ferrari [61] 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 [61, 54, 90, 9]. 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

Games, training, medicine and many other applications in everyday life [70, 17, 41, 72] leverage the advantages of mobile devices with their rapidly progressing hardware and software. The touch-based interaction on mobile devices is arguably more intuitive than the use of keyboard and mice on PCs. Nevertheless, mobile devices have their limitations when it comes to rendering and interacting with complex 3D models, e.g. lower processing power, limited storage and less capable rendering hardware. Computationally intensive 3D graphics rendering tasks also impose severe challenges on the limited battery capacity of mobile devices.

To address these issues, remote rendering can be employed. This technique leverages the computational capacity of a remote server to relieve the rendering burden on the client side. One direction in this field is partitioning and simplifying large and complex

3D meshes on a server and sending a part of simplified meshes to the client. Park and Lee [63] propose a framework to enable users to view large meshes on mobile devices hierarchically. Depending on various scales, meshes with different Level Of Detail (LOD) are generated on the server and transmitted to the clients. However, transmission of meshes with different LOD frequently increases power consumption. Yan et al. [93] propose the algorithm ASEHM that minimizes the transmission frequency in sending 3D meshes. Instead of transmitting a new level of the same model, ASEHM refines a coarser LOD into a higher-fidelity LOD, provided that the relationship between the LODs is available.

Another research direction is offloading the entire rendering task to the server. Basically, when a mobile client connects to the server, the server will initialize a rendering engine and an encoder for the mobile client. All the models are rendered on the server side and rendered frames are encoded and streamed from the server to the client as a video stream. Moreover, 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 [42], Moimark and Cohen-Or [59] and Lu et al. [51] have proposed such CMR systems. Many cloud gaming services and frameworks have leveraged this technology, e.g. OnLive[1], NVIDIA GeForce Now[2], PlayStation Now[3] and Gaming Anywhere[4]. Those services and frameworks use the cloud computing infrastructures to render the games and encode the rendered frames, while the clients decode the rendered frames and send the players' interactions to the cloud. The primary disadvantage of this type of approach is the introduction of additional latency, including user interaction transmission time, rendering and encoding time at the server side, image transmission time, decoding time at the client side [75]. However, minimizing the latency is challenging considering the high uncertainty of the network transmission. Using a geographically proximate server helps to greatly reduce the latency since every 1000 miles of physical distance adds 25ms round-trip delay to the overall latency [65].

To relieve the heavy dependence of remote rendering methods on the network, researchers proposed various methods [62, 49, 94]. One type of methods uses environment maps to accelerate rendering, e.g., Boukerche and Pazzi [7] render environment maps on the server and send them to the client. With the environment map, the client is able to respond to the user interaction in the form of panning, tilting and zooming without latency. Evironment maps have some similarity to panoramas which were originally proposed by Chen [16] in their QuickTime VR system which displayed virtual environments without rendering 3D models. QuickTime VR accomplished moving through the environment by "hopping" to different panoramic points. Boukerche and Pazzi [7] do not simulate movement by "hopping" to another panoramic image, instead, they use a remote server to render panoramic images in real-time. They address the delay in rendering and transmitting panoramic images through a caching design that buffers visited viewpoints. A common disadvantage of methods that use environment maps is that they only handle

[1]http://onlive.com
[2]https://www.nvidia.com/en-us/geforce/products/geforce-now/
[3]https://www.playstation.com/en-ca/explore/playstationnow/
[4]http://gaminganywhere.org

static environments and navigation typically leads to wait times for the user. If the objects in the scene move or the user navigates, re-rendering and re-transmission of environment maps can lead to latency [56, 69].

Some researchers attempted to leverage the advantages of a depth map to synthesize new views on the client. Shi et al. [74] proposed a framework that leverages depth maps to reduce user interaction latency. They take advantage of 3D image warping to synthesize the mobile display from the depth images generated on the server. Chang and Ger [12] proposed building Layered Depth Images (LDIs) on the server. The mobile device uses a 3D warping algorithm to synthesize the frame from a new view point. At the time of the design of LDIs, mobile devices had almost no capacity to render 3D scenes and hence LDIs were designed as a way around displaying 3D content on a mobile device. These image-based methods incorporating scene depth often face issues with visible gaps and holes in the rendered scene. Bao and Gourlay [3, 4] proposed a method that uses a superview to direct the image warping and reduce the gaps and holes. The method is successful in reducing the flaws in the synthetic images. 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. Their method is well suited for environment walkthrough applications but not for games and training applications.

Other approaches support dynamic scenes and all types of interactions. They accelerate the data transmission to adapt to various network environments or reduce interaction delay. Levoy [46] proposed a method that renders simplified models on clients to reduce bandwidth requirement. In this method, the server first renders both complete and simplified models and calculates 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 is that Levoy [46] uses simplified models to achieve a compression effect on image transmission, while our method uses simplified models on the client side for display and interaction. Also, our simplification is selective and achieves a bandwidth reduction by rendering environment models only on the client. Liu et al. [50] extended image streaming and developed 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 key models and environment models in a scene. Their work also does not address one of the most severe issues in remote rendering, interaction latency. Hemmati et al. [33] propose a method for cloud gaming, which selectively renders important objects and reduces video bitrate by not rendering unimportant objects. However, they did not answer the question about how it influences user experience. Inspired by Hemmati et al. [33], Lu et al. [52] proposed an approach that maximizes user experience with limited

network bandwidth for stereo video streaming. In this approach, the left view and right view are rendered asymmetrically in terms of texture detail and certain objects are not rendered in on view. This technique enables intelligent decision of whether or not to render an individual object and how good the corresponding texture detail will be if rendered.

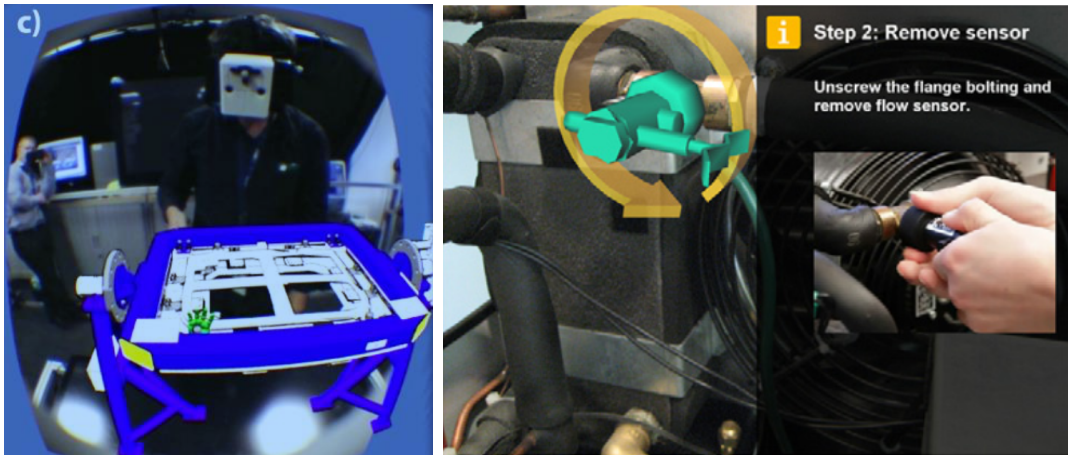
We propose a general-purpose remote rendering framework. First, by only rendering key models remotely, our method minimizes the network bandwidth requirement. Second, our method leverages the graphics capacity of mobile devices to loosen the tight dependence of the clients from the remote server. Applications using our framework are able to adapt to various network environments. We do not assume a priori knowledge about the application, thus our framework is able to handle dynamic scenes and all types of interaction (e.g. motion, panning and zooming). We also benefit from the increasing rendering capabilities of mobile devices while addressing the fact that mobile devices still fall behind PCs. We believe that the idea of offloading the entire rendering task to the server may be outdated given the recent advances in mobile device hardware.

2.3 Augmented Reality Training

The first augmented reality system can be dated back to 1962 [38, 95]. 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 [81]. In 1992, Boeing [10] 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 [25, 73, 5]. Freitas and Campos [25] 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 [25]. Schrier [73] 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. [5] 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.

There exist many reviews of implementations and progress in augmented reality applications [6, 71, 98, 44]. Billinghurst and Kato [6] focused on collaborative AR applications and demonstrated examples that allow people to view or change the same virtual models. Regenbrecht et al. [71] showed ten AR applications in various fields which make cutting edge progress at that time. Zhou et al. [98] 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 [44] discussed the areas that augmented reality technology has been used in and identify how it can be



(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.

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 [57], 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. [53] 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. [43] 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 [58]. 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. [28] 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 [10]. Henderson and Feiner developed a framework to assist conducting military routine maintenance tasks inside an armored vehicle turret [34]. 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. [19] 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. [35] 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. [92, 91] 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. [98] 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. [88] 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. [2] 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 [39, 100] according to a comparison [80] of various background subtraction methods. We employ the implementations

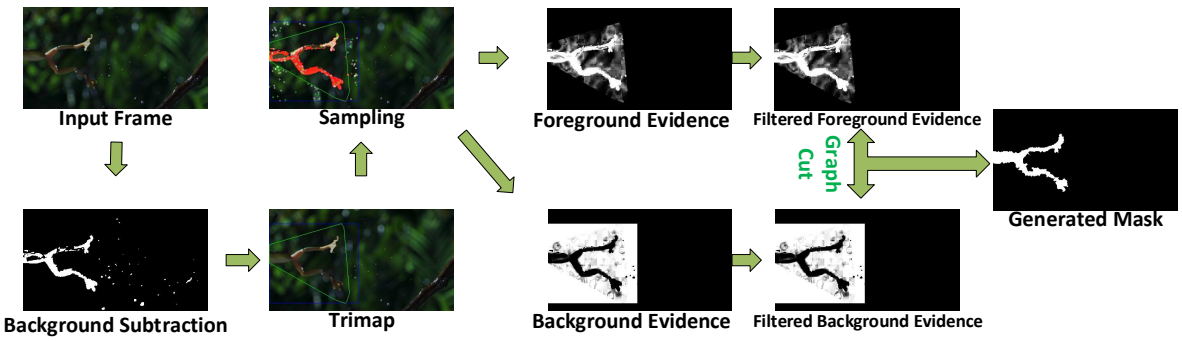


Figure 3.1: Overview of the proposed video object segmentation method. The images are a frame from the dataset SegTrack v2 [47].

of the two 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 [83] while convex hulls are calculated using Sklansky's algorithm [78]. 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 [48]. Similar to [48], we use

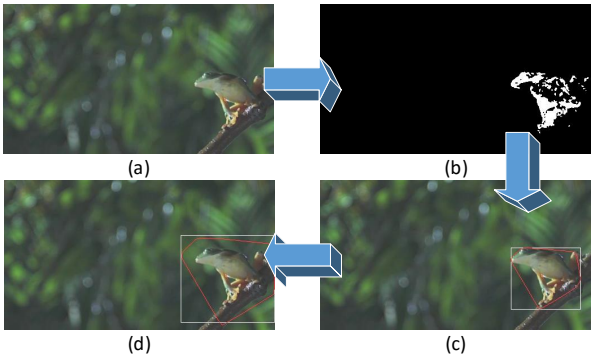


Figure 3.2: Trimap generation. (a) Original frame. (b) Background subtraction result. (c) Convex hull (red) and bounding box (white) of the blob. (d) Convex hull (red) and bounding box (white) after expansion. The images are a frame from the dataset SegTrack v2 [47].

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 [48].

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 [84].

Moreover, we adapt the method used by Tao and Krishnaswamy [84] 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 [77], 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 [1].

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 [8] to solve Eqn. 3.9.

3.2 Evaluation

3.2.1 Dataset

We evaluate our method on the dataset SegTrack v2 [47]. 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 [80] 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 [61] and Zhang et al. [97], as well as two background subtraction methods [39, 100]. The video object segmentation method of Papazoglou and Ferrari [61] 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. [97] 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 [39, 100] 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 [39] with a learning rate of 0.005 to detect initial moving pixels, while, for the other videos, we used the MOG2 approach [100] 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 [61], [97] 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 [61] produces better results while our method is much faster and produces comparable results. For four of the five videos, the method [61] 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 [61] 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 [61] optimizes the results of the entire video, which limits its use on high-resolution and long videos. The

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

Table 3.1: Comparison with methods [61] and [90] on four videos birdfall, frog, hummingbird and penguin. The results are measured by F-Measure.

	Ours	[61]	[97]	MOG [39]	MOG2 [97]
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	[61]	[97]
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

memory demand of method [97] is very large, so that it is impossible for us to run it with 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 [97] misses the objects. Our method is around $10 - 200\times$ faster than the methods [61] and [97]. 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 $|B|$.

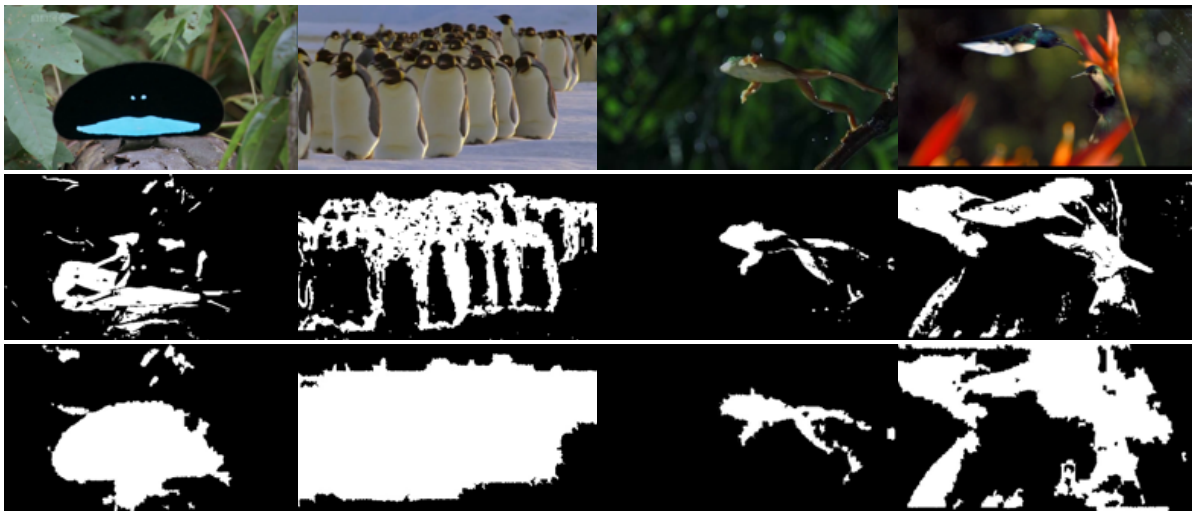
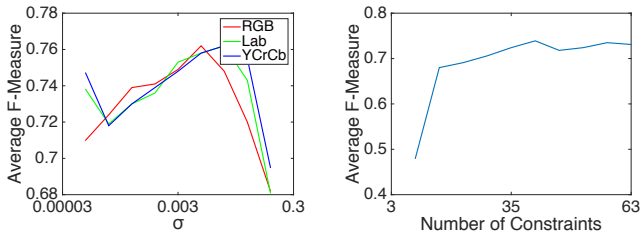


Figure 3.4: Background subtraction masks vs. final masks. The first row demonstrates the original frames, the second row shows the masks generated by background subtraction methods, while the third row is the final masks produced by our method.



(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.

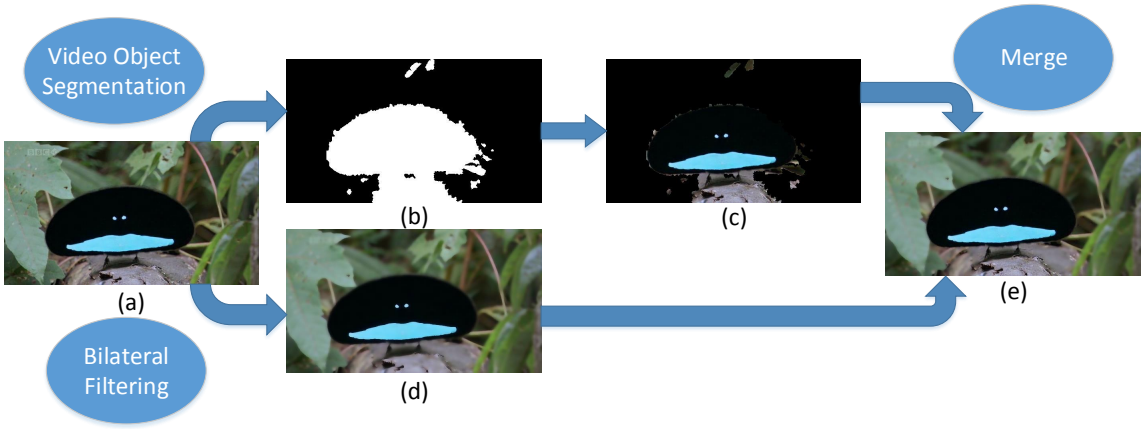


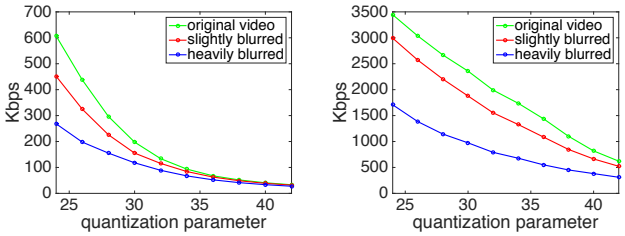
Figure 3.6: Content-Aware Video Compression. (a) Original frame. (b) Mask generated with the video object segmentation method described above. (c) The object pixels covered by the mask. (d) The frame blurred with bilateral filter. (e) The merged frame.

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. [89] and the sparsity decompression of Chen et al. [14]. Our approach is also related to seam carving for compression as proposed by Decombas et al. [23].

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

In this section, we present a hybrid remote rendering framework for applications on mobile devices. In our remote rendering approach, we adopt a client-server model, where the server is responsible for rendering high-fidelity models, encoding the rendering results and sending them to the client, while the client renders low-fidelity models and overlays the high-fidelity frames received from the server on its rendering results. With this configuration, the client is able to keep functioning regardless of network condition and bandwidth. Moreover, to minimize the bandwidth requirements, only key models are rendered in high-fidelity mode. We conduct a user study on factors that may impact the objective and subjective difficulty of an interactive task in a 3D mobile application. The results show that model fidelity has a significant influence on the objective difficulty, while network delay plays an important role in the subjective difficulty. The results of the user study explain how our method is able to benefit the users.

4.1 Method

4.1.1 Client-Server Prototype Design

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

Our system is inspired by Levoy [46], Lamberti and Sanna [42] and Lu et al. [51] and uses a hybrid approach incorporating both local and remote rendering. Hence, the low-fidelity key models are stored on the client-side and the local rendering capacity is leveraged to produce lower quality rendering results. Conversely, the high-fidelity renderings of key models are sent from the server to the client and overlaid upon the locally rendered frames. Accordingly, the data transferred via the network is minimized due to the small amount of pixels that are sent via the network compared to a CMR solution.

There are three challenges in realizing multi-client support within the context of the above described system. First, it is required that the server must not be blocked by any of

the clients. Second, each client has its own view that may be different from all the other clients, this can result in a performance issue as a scene needs to be rendered multiple times in a single animation loop. Third, because only key models are rendered and sent by the server, occlusion becomes a problem since the environment models that are closer to the viewpoint may not be rendered on the server but just locally.

We address the first problem by mandating that the server interacts with each client independently. More specifically, we enable the server to receive commands from and send to every client separately, so that if a client becomes unresponsive, the server is not blocked. The second issue is alleviated by implementing a "render-on-demand" function on the server, which means that the server renders the scene for a client only when the client requires a new frame. We address the third challenge through a two-pass rendering process. In the first pass, the entire scene is rendered in low fidelity on the server. Then the colour buffer is cleared before the second pass where only the key models are rendered in high fidelity. In this way, the second pass is rendered with the depth information obtained from the first pass.

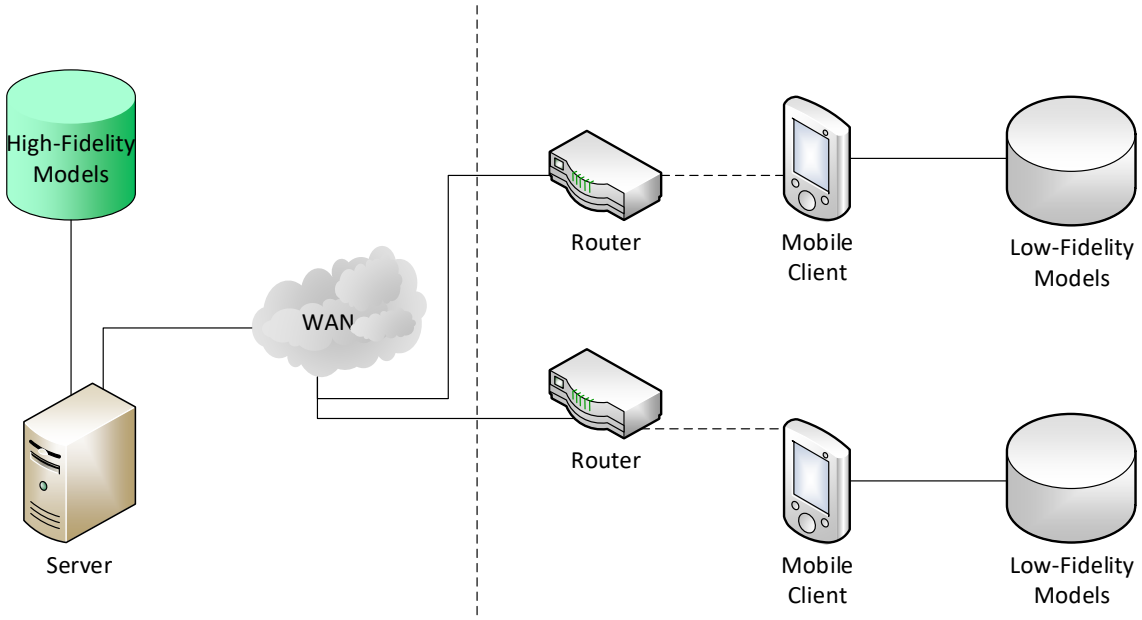


Figure 4.1: System architecture

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 after synchronization. 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 depicting those key models. The other models are still rendered locally.

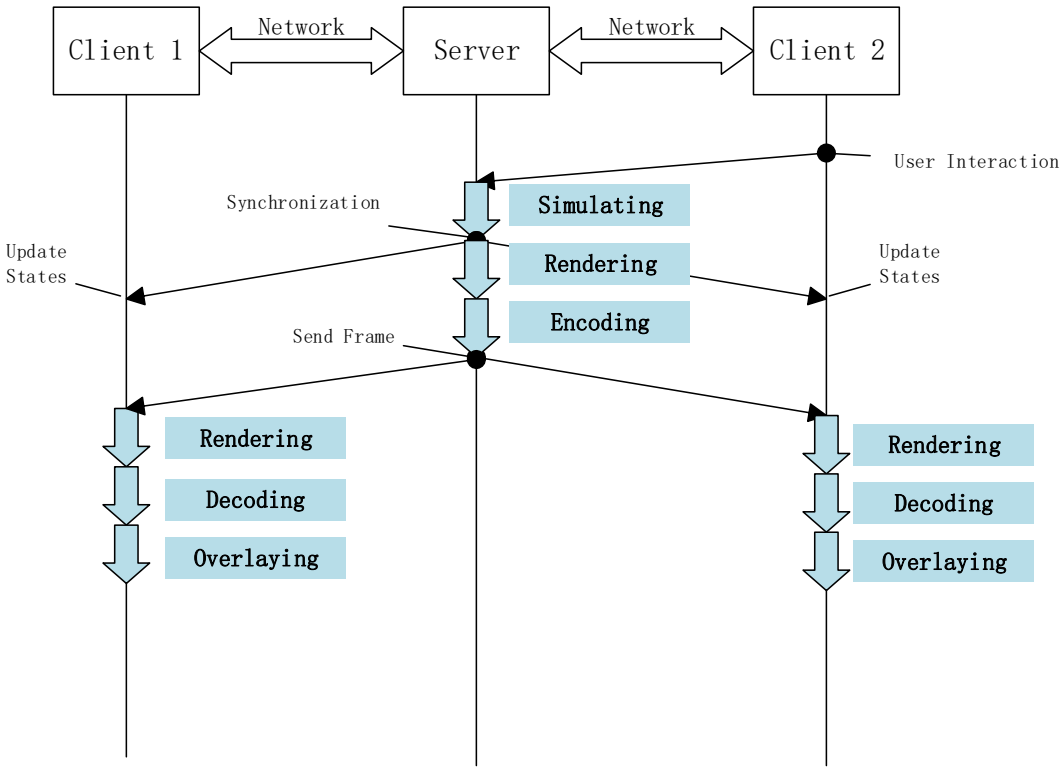


Figure 4.2: Workflow of communication between the server and the clients

Fig. 4.2 illustrates the sequence of messages communicated between the server and the client processes. It shows the tasks that are performed by the server and clients to process one frame. If the commands sent from any client influence the simulation states of the server, the latter synchronizes the state changes among all clients to ensure that all of them maintain a consistent state. Moreover, the server renders and encodes the frames for all the clients. Upon receiving the high-fidelity frame, the client decodes and overlays it on the locally rendered frame.

4.1.2 Process Behavior

The proposed system consists of one server and multiple clients. Fig. 4.6 shows the behavior of the server process. In each loop, the server updates the simulation status of the scene and receives commands from all clients. Next, the server sends the received commands to all clients to ensure synchronization between them. On the server side, each client has its own view that is independent from those of the other clients. The server only renders a new frame upon request from the client.

Fig. 4.7 shows the behavior of the client processes. In each loop, the client receives commands from the server and adjusts the simulation status accordingly. Then, it sends its user interaction commands to the server. In each iteration, the client will receive the high-fidelity frame from the server that 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.

4.1.3 Two-pass Rendering

As mentioned in Section 4.1.1, we use a two-pass rendering process on the server side. The frames sent to the client depict only high-fidelity key models. But this is not enough for valid view of the 3D scene since other scene content may occlude parts of the key models. We propose a two-pass rendering process to address this issue.

As shown in Fig. 4.8, in the first pass, all models are rendered in low-fidelity except for the key models. Then the color buffer is cleared and only the depth buffer is preserved. In the second pass, only the high-fidelity key models are rendered, while considering the depth information obtained from the first pass rendering, therefore the occlusions are preserved in the final rendered scene.

4.1.4 Process Behavior

The proposed system consists of one server and multiple clients. Fig. 4.6 shows the behavior of the server process. In each loop, the server updates the simulation status of the scene and receives commands from all clients. Next, the server sends the received commands to

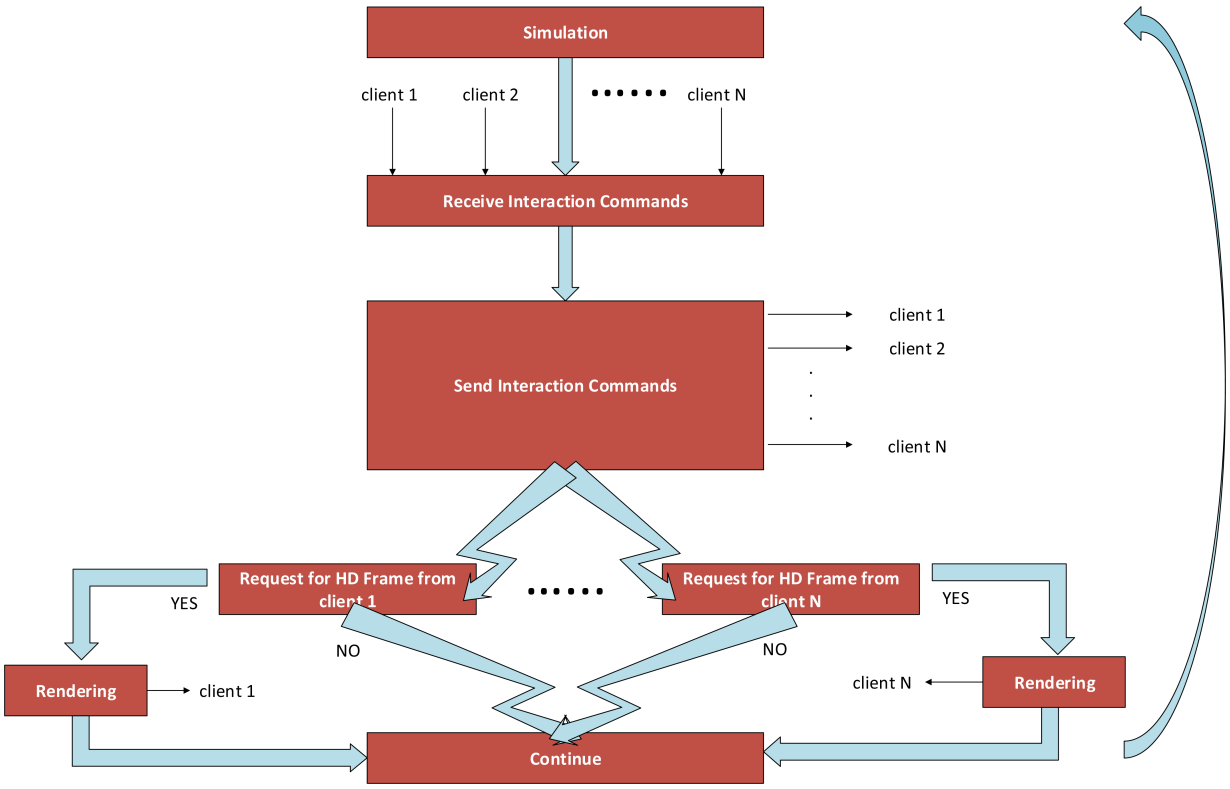


Figure 4.3: Server-side Workflow

all clients to ensure synchronization between them. On the server side, each client has its own view that is independent from those of the other clients. The server only renders a new frame upon request from the client.

Fig. 4.7 shows the behavior of the client processes. In each loop, the client receives commands from the server and adjusts the simulation status accordingly. Then, it sends its user interaction commands to the server. In each iteration, the client will receive the high-fidelity frame from the server that 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.

4.1.5 Two-pass Rendering

As mentioned in Section 4.1.1, we use a two-pass rendering process on the server side. The frames sent to the client depict only high-fidelity key models. But this is not enough for valid view of the 3D scene since other scene content may occlude parts of the key models. We propose a two-pass rendering process to address this issue.

As shown in Fig. 4.8, in the first pass, all models are rendered in low-fidelity except for the key models. Then the color buffer is cleared and only the depth buffer is preserved. In

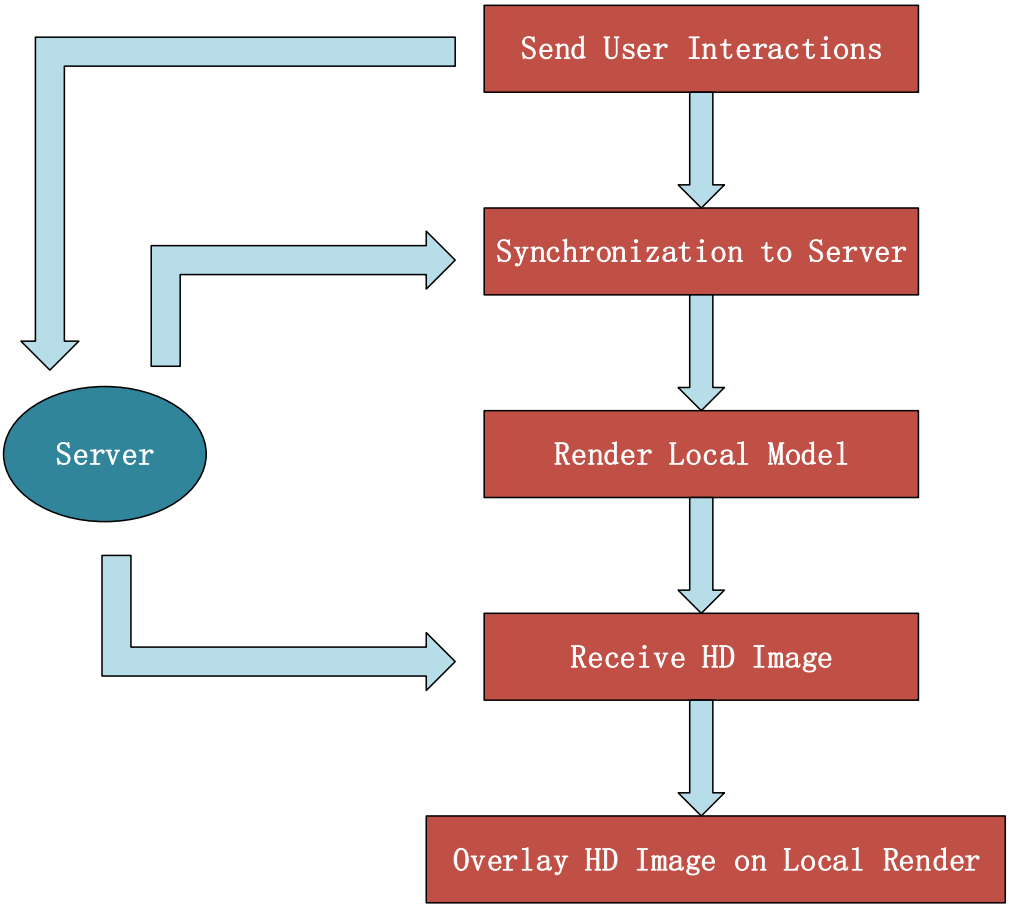


Figure 4.4: Client-side Workflow

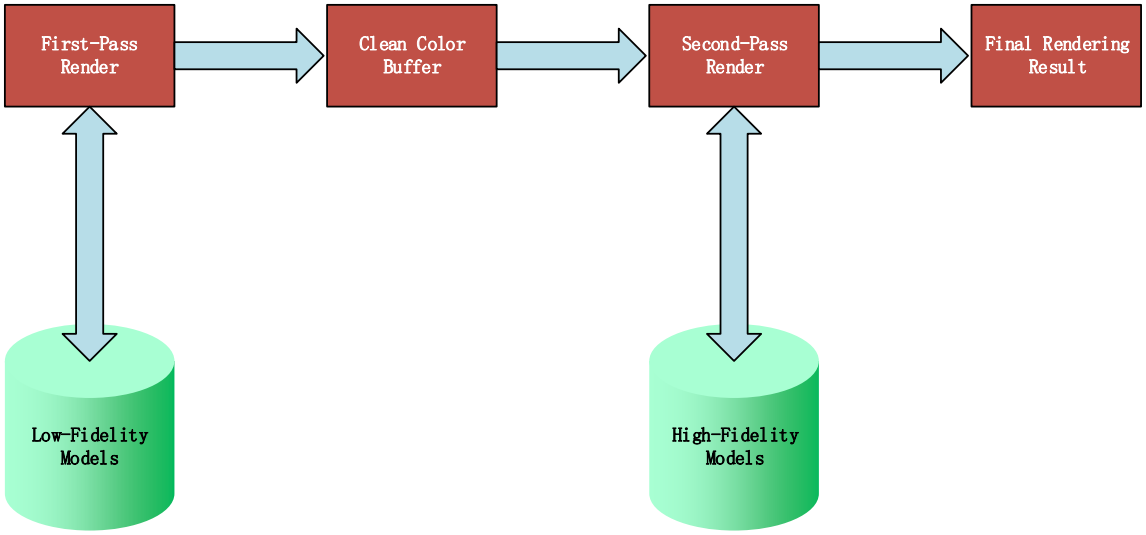


Figure 4.5: Two-pass rendering

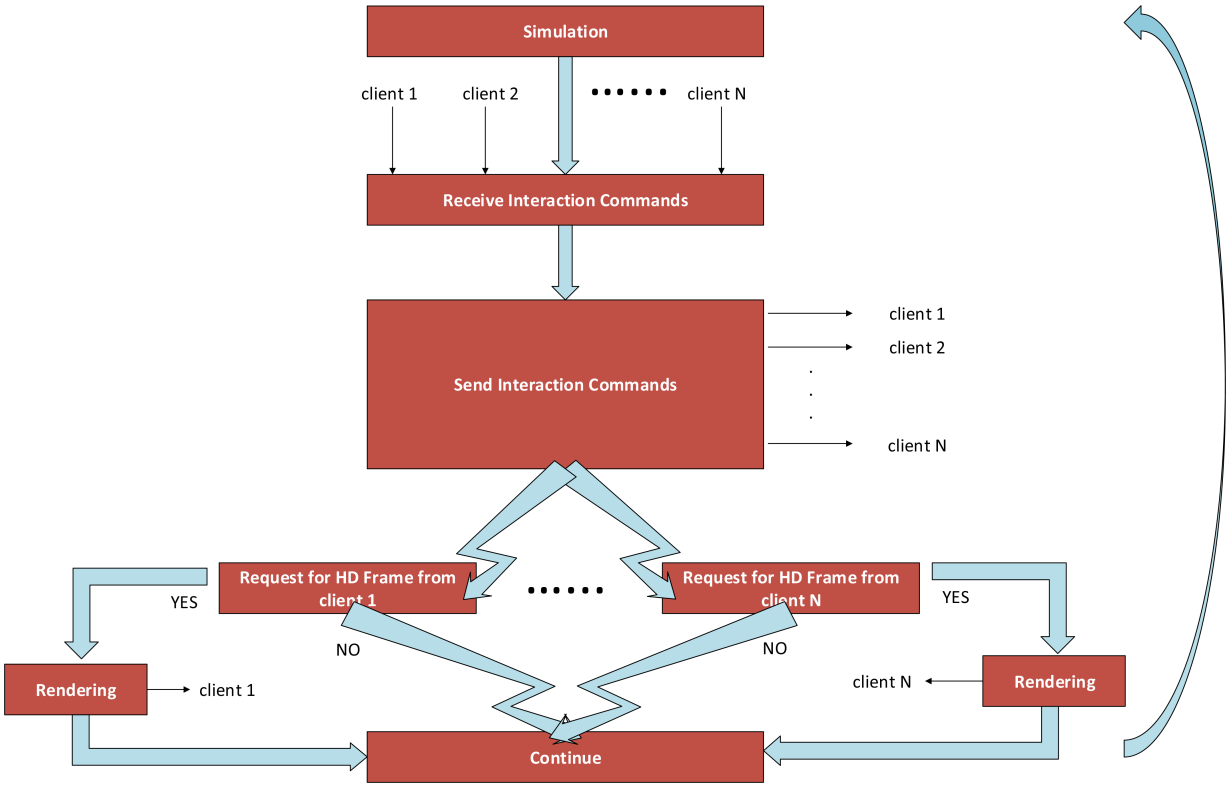


Figure 4.6: Server-side Workflow

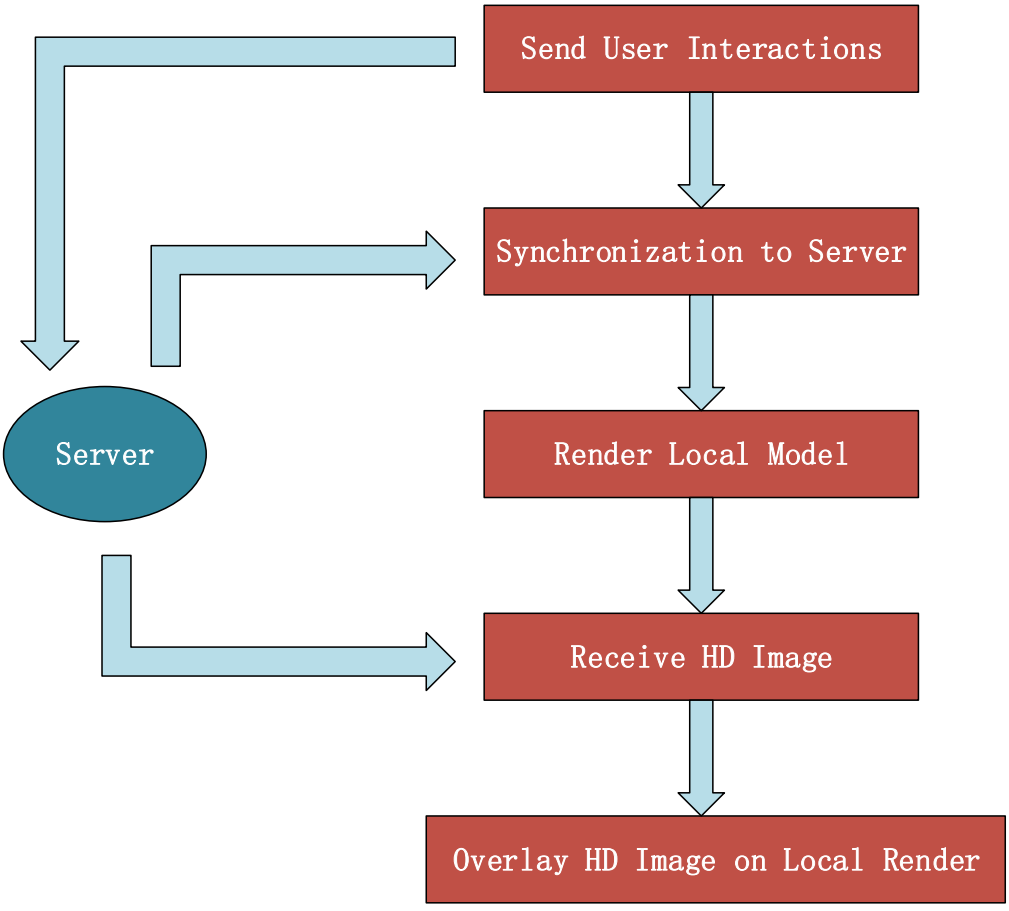


Figure 4.7: Client-side Workflow

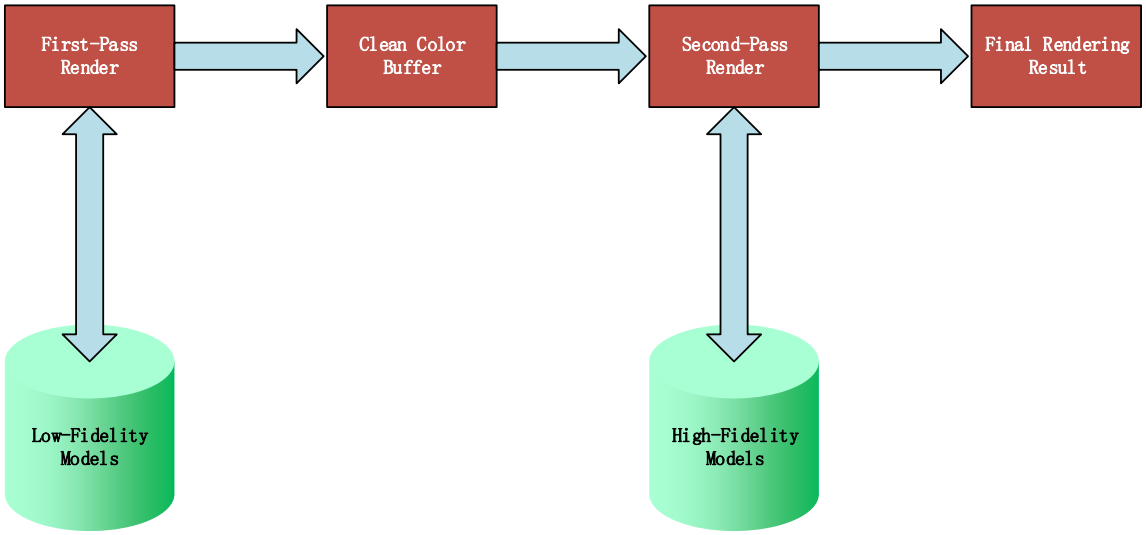


Figure 4.8: Two-pass rendering

the second pass, only the high-fidelity key models are rendered, while considering the depth information obtained from the first pass rendering, therefore the occlusions are preserved in the final rendered scene.

4.2 User Study

In Section 4.1, we propose a hybrid remote rendering framework that offloads a part of the rendering task to the remote server. More specifically, only key models are rendered remotely to minimize network bandwidth requirement and network delay.

To overcome the limitations of mobile devices in rendering high-fidelity models, designers often use one of the two approaches:

1. Local-only rendering: Reduce the fidelity of the models and render them locally on mobile devices.

2. Server-only rendering: Render the scene remotely on a server and transfer to the mobile phone client as a video stream.

However, both approaches risk reducing the user's Quality of Experience (QoE). For the local-only rendering approach, the user is interacting with models that are not rendered at their optimal quality. Fine details in the model might be missing, which reduces the aesthetics of the scene. Also, for some applications, these details might carry important information. For instance, it is important for users in many gaming applications to quickly distinguish between similar objects in a scene. Objects rendered as low-fidelity models

might not be immediately discernible. For the server-only approach, QoE might be affected by a long delay between the user's actions and application's response. This would be mostly caused by the network delay between client and server. Furthermore, bandwidth limitations can cause interruptions in the video stream. The proposed approach strikes a balance between local-only rendering and server-only rendering. It renders only key models remotely at high-fidelity while other models are rendered locally at low-fidelity.

Previous work has explored how various factors affect the user experience. Hong et al. [36] conducted a subjective user study to quantify the effect that video bitrate and frame rate have on QoE in cloud gaming. Slivar et al. [79] use Steam In-home streaming platform[1] as a case to study how the frame rate and bitrate influence the QoE. They model the QoE as a combination of two aspects: perceived graphics quality and perceived game fluidity. In the work by Suznjevic et al. [82], five factors (i.e. latency, jitter, packet loss, frame rate and frame resolution) are studied in terms of their effects on the QoE of NVIDIA GeForce Now platform. Hemmati et al. [33] developed an algorithm that minimizes bitrate by not rendering unimportant models. Our method leverages a similar idea by rendering key models in high fidelity and environment models in low fidelity. But the difference is that our method renders the environment models locally instead of not rendering them at all.

Our method targets not only gaming, but also other types of applications, such as training applications and virtual tours. The expectations of users in terms of graphics quality and application fluidity may depend on the different types of applications. For instance, the players of a game may focus on the quality of special effects, while the users of a surgery application may focus on how clearly anatomical features can be distinguished. Thus, we do not measure QoE but how difficult it is to accomplish a given task. Thus, we are interested in the effects of the key model fidelity and environment fidelity on the perceived diffculty of task completion. We name this measure the Difficulty of Task (DoT). We measure DoT in two ways: Objective DoT (ODoT) and Subjective DoT (SDoT). The ODoT is a measure of how well the users can accomplish a task. It might be measured by a score or a level calculated by the application. The SDoT is a measure of the users' opinion on the task difficulty and can be answered by a subjective questionnaire.

Our user study was approved by the Research Ethics Board at the University of Ottawa (File Number: H12-16-05). The main user study is informed by a pre-trial study discussed next.

4.2.1 Pre-Trial Study

A major goal of the main study is to analyze the effect of 3D model fidelity on the DoT. However, in the proposed framework, there are two types of models, high-fidelity models and low-fidelity models. Those models will be used in the main study (see Section 4.2.2). We have prepared 15 high-fidelity models and simplified them to obtain low-fidelity models using QSlim [26].

[1] http://store.steampowered.com/streaming/

Objective

The objective of the pre-trial study is to decide the number of polygons for the low-fidelity models. If it is too low, the participants will not be able to recognize the object that the model represents. In the pre-trial study, we want to find out at which point the object represented by the model becomes discernible to the majority or all subjects.

Participants

We recruited 5 participants, 4 males and 1 female. None of the subjects were visually impaired.

Independent Variables and Dependent Measures

The experiment was conducted with one independent variable: number of polygons of the surface mesh in the 3D model. The independent variable has 50 levels. For the first level, the model is composed of 20 polygons. For each of the 49 subsequent levels, the number of polygons is increased by 10%. We measure the number of polygons necessary for a subject to recognize the object for each 3D model.

Procedure

The experiment was conducted using a program that shows 15 models at different levels of fidelity. We create 50 levels of fidelity for each model by starting with 20 polygons and increasing its polygon count by 10% for every level using QSlim [26]. Our program shows the subjects a model on the screen and asks them to select the object it represents from a side menu by choosing among dog, cat, horse, or unknown. Subjects were instructed to choose the unknown option if they were unsure. We start with a model with very low polygon count (20 polygons). Every time the subject selects an answer, we increase the polygon count to the next level adding more details to the model. The order through which the models are shown to the subjects is randomized to avoid biasing the results.

Results

Table 4.1 contains the data collected from the pre-trial study. The number of polygons above which a participant ceases to make mistakes (i.e. the participant choses the correct answer consistently at all further levels of fidelity). The data are used to decide the number of polygons for the low-fidelity models in the main study. To eliminate the outliers, we select the second highest number among all participants for each model.

Table 4.1: Data collected from the pre-trial study, which represents the number of polygons above which a participant ceases to make mistakes are shown. However, in one case, one of the participants does not recognize the model even at full resolution (marked by a dash). The numbers in bold are those selected as the final number of polygons of the low-fidelity models for the main study.

	Participant #1	Participant #2	Participant #3	Participant #4	Participant #5
Model #1	148	22	57	51	**111**
Model #2	162	57	**422**	22	422
Model #3	122	383	134	22	**238**
Model #4	75	**148**	-	69	134
Model #5	162	51	32	22	**134**
Model #6	**1095**	179	464	1940	464
Model #7	995	1603	**1325**	32	422
Model #8	75	162	148	91	**148**
Model #9	75	83	216	122	**196**
Model #10	464	262	748	216	**562**
Model #11	422	1204	**422**	288	383
Model #12	47	101	**134**	22	134
Model #13	22	47	510	29	**238**
Model #14	238	22	348	**317**	162
Model #15	69	83	**75**	75	51

4.2.2 Main Study

Our main user study is designed to address the following research questions:

1. What is the effect of key model fidelity on the DoT?
2. What is the effect of environment model fidelity on the Dot?
3. What is the effect of network delay on the DoT?
4. What is the effect of network jitter on the DoT

We ask the subjects to interact with a 3D mobile application that prompts the user to respond rapidly to visual stimuli. 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 server-only and local-only rendering approaches. In our user study, we do not consider any network-aware adaption strategies of the video stream performed by the remote server.

The test application we use in our experiment depicts a 3D environment representing a virtual room. Guided by a compass, the user must navigate (pan and tilt) the environment to reach a destination at which an object will appear and disappear within a short period of

time. A menu prompting the user to specify the type of that object is brought up after an object appears, as shown in Fig. 4.9. This test application is designed as a simplification of a large variety of 3D applications. It consists of two sub-tasks: navigation and recognition. For instance, in a 3D shooting game, the players move around in the environment and look for his/her targets. After finding a target, the players need further recognize the targets to decide whether it is an enemy or a friend. Another example is surgery training applications [11], in which the users need to navigate to change the view point and recognize different components of the body before performing an action.

Hypothesis

We devise eight null hypotheses to test in the experiment, see Table 4.2. The null hypotheses are defined for each measurement factor. In those null hypotheses, we assume all four factors do not have any effect on the ODoT nor the SDoT. Rejecting a null hypothesis indicates that there is no evidence proving that the corresponding factor has no effect on that dependent measure.

Table 4.2: Hypotheses.

H_{01}	Different model fidelities do not affect the ODoT
H_{02}	Different environment fidelities do not affect the ODoT
H_{03}	Different network delays do not affect the ODoT
H_{04}	Existence of jitter does not affect the ODoT
H_{05}	Different model fidelities do not affect the SDoT
H_{06}	Different environment fidelities do not affect the SDoT
H_{07}	Different network delays do not affect the SDoT
H_{08}	Existence of jitter does not affect the SDoT

Participants

15 volunteers participated in the main study, 12 males and 3 females. None of the subjects were visually impaired.

Independent Variables and Dependent Measures

The experiment was conducted with four independent variables: key model fidelity, environment model fidelity, delay and jitter. Key model fidelity and environment model fidelity have two levels: high and low, as described in Section 4.2.1. The variable delay is set to introduce network delay. In the variable network delay, we summarize the effects of network transmission, remote rendering and encoding. In this experiment, the variable delay was simulated locally. Similar to the work by Zhu et al. [99], we use the normal distribution of (4.1) for jitter in settings with non-zero delay.

$$p(j|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}^{-\frac{(j-\mu)^2}{2\sigma^2}} \tag{4.1}$$

where j is the value of jitter, $\mu = 77$ms and $\sigma^2 = 1$.

Combining the four variables results in 20 configurations. Table 4.3 details the test application configurations. The order of the 20 configurations is randomized across subjects to avoid biasing the results.

Table 4.3: Test application configurations.

Configuration	Key Model Fidelity	Environment Model Fidelity	Delay	Jitter
1	low	low	0ms	not applicable
2	low	high	0ms	not applicable
3	high	low	0ms	not applicable
4	high	high	0ms	not applicable
5	low	low	80ms	not applied
6	low	high	80ms	not applied
7	high	low	80ms	not applied
8	high	high	80ms	not applied
9	low	low	80ms	applied
10	low	high	80ms	applied
11	high	low	80ms	applied
12	high	high	80ms	applied
13	low	low	120ms	not applied
14	low	high	120ms	not applied
15	high	low	120ms	not applied
16	high	high	120ms	not applied
17	low	low	120ms	applied
18	low	high	120ms	applied
19	high	low	120ms	applied
20	high	high	120ms	applied

In this experiment, we collect two types of data: ODoT and SDoT. The ODoT is measured by the number of objects that participants incorrectly recognize. The SDoT is measured using a questionnaire directly integrated within the test application. The questionnaire contains one question: "Please score how difficult it was to complete the task". A 5 point Likert Scale is used to collect the answers. A score between 1 to 5 is associated with each question, where 1 represents the option "Very Easy" and 5 represents the option "Very Difficult".

Procedure

We run the test application 20 times for each subject. We vary the key model fidelity, environment model fidelity, delay and jitter across these runs. Table 4.3 details the test

(a) Sub-task of navigation (b) Sub-task of recognition

Figure 4.9: Screenshots of the test application. During one run of the test application, the two sub-tasks are repeated ten times with different animal models. (a) Sub-task of navigation. In the test application, users need to pan and tilt to align the camera viewfinder with the orange plate by following the direction of the compass. If there exist network delay and jitter, users will feel them when panning and tilting. (b) Sub-task of recognition. After aligning the camera viewfinder with the plate, an object appears. The object is not static, but rotating and moving around instead. As similar to the sub-task of navigation, the movement and rotation of the object is also influenced by network delay and jitter.

application configurations for the 20 runs. The order of the 20 configurations is randomized across subjects to avoid biasing the results. For each configuration, ten 3D objects appear. So the maximum score of the ODoT for each configuration is 10 and the minimum score is 0. As mentioned in Section 4.2.2, we also collect SDoT during the experiment. After each run of the test application, the participant was asked to fill the SDoT questionnaire.

As shown in Table 4.3, we prepare four types of scenes: a scene with only low-fidelity key models and low-fidelity environment models, a scene with low-fidelity key models and high-fidelity environment models, a scene with high-fidelity key models and low-fidelity environment models and a scene with high-fidelity key models and high-fidelity environment models. We simulate three different network delays: 0ms, 80ms and 120ms and we also consider network jitter.

Results

To understand how each factor affects the SDoT and the ODoT, we use analysis of variance (ANOVA) to interpret our experimental data.

We use a four-way ANOVA test to investigate the effect of all four factors. However the factor combination of delay and jitter is not complete, since jitter is not applicable when delay is 0ms. So we exclude the delay level 0ms in the four-way ANOVA test. As shown in Table 4.4, the four-way ANOVA considers the four factors and their levels, except for 0ms of delay.

We also analyze the effect of the delay factor and its interaction with key model fidelity and environment model fidelity. For this analysis, we resort to a three-way ANOVA for the

combination of delay, key model fidelity and environment model fidelity. Table 4.4 shows the factors of the three way ANOVA.

Table 4.4: Factors and levels of the ANOVA tests.

	Key Model Fidelity		Environment Model Fidelity		Delay			Jitter	
	Low	High	Low	High	0ms	80ms	120ms	applied	not applied
Four-way ANOVA	✓	✓	✓	✓	✗	✓	✓	✓	✓
Three-way ANOVA	✓	✓	✓	✓	✓	✓	✓	✗	✗

Table 4.5 shows the grand mean of the ODoT measure for all factors. The ODoT refers to the number of objects the user did not recognize, hence higher scores represent higher objective difficulty of task. Since we employ two tests of significance (four-way and a three-way ANOVA) we calculate two grand means of the ODoT for each factor level. Four-way ANOVA does not consider the results of configurations 1 to 4 of Table 4.3 (since it does not consider 0ms delay as previously explained). Table 4.5 shows that the ODoT varies with different levels of each factor. However, not all factors have an effect on the ODoT. We can see that in both, the four-way and the three-way ANOVA, participants are able to reduce the number of incorrectly recognized objects with high-fidelity key models and in low-fidelity environment models. The grand means for the factor delay do not agree in the four-way ANOVA and the three-way ANOVA, so we consider the factor delay as not having an effect on ODoT.

Table 4.6 shows the grand means with respect to SDoT. As in Table 4.5, we show the grand means calculated for the four-way ANOVA and three-way ANOVA tests in Table 4.6. The means are measured through a 5 point Likert Scale, where larger values indicate higher subjective difficulty of task.

From Table 4.6, we can see that three factors have effects on the SDoT. The SDoT is reduced for high-fidelity key models and low delay. The grand means for the factor environment model fidelity do not agree in the four-way ANOVA and the three-way ANOVA, so we cannot conclude that the factor environment model fidelity has an effect on SDoT.

Table 4.5: Grand means of the ODoT for each factor.

	Key Model Fidelity		Environment Model Fidelity		Delay			Jitter	
	low	high	low	high	0ms	80ms	120ms	applied	not applied
Four-way ANOVA	4.6	3.5	3.9	4.2	-	4.0	4.1	3.8	4.2
Three-way ANOVA	4.3	3.5	3.8	4.0	4.0	3.9	3.8	-	-

However, the conclusions drawn from Table 4.5 and Table 4.6 may not be significantly different in statistics. Hence, we analyze the variance for both metrics, i.e. ODoT and

Table 4.6: Grand means of the SDoT for each factor.

	Key Model Fidelity		Environment Model Fidelity		Delay			Jitter	
	low	high	low	high	0ms	80ms	120ms	applied	not applied
Four-way ANOVA	2.7	2.5	2.7	2.6	-	2.5	2.8	2.5	2.7
Three-way ANOVA	2.4	2.2	2.2	2.3	1.9	2.4	2.6	-	-

SDoT to determine whether any of the differences between the means are statistically significant, we compare the p-values to a significance level to decide whether the p-value suggests accept or reject on the null hypothesis. We choose a significance level of 0.05.

Table 4.7 and Table 4.8 demonstrate the results of ANOVA in terms of the ODoT, where Table 4.7 shows the results of the four way ANOVA while Table 4.8 shows the results of the three way ANOVA.

Only the key model fidelity factor exhibits a p-value less than 0.05 for both ANOVA tests for the ODoT measure. To explore the interaction between a factor pair or among multiple factors, we also calculate the p-values for those interactions. These 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 smaller than 0.05. This indicates that these factors are independent from each other in terms of the ODoT.

Table 4.7: Four-way ANOVA of the ODoT.

Source	F-statistics	p-value
Key Model Fidelity	14.9	0.0001
Environment Model Fidelity	1.28	0.2589
Delay	0.14	0.7063
Jitter	1.71	0.1929
Key Model Fidelity × Environment Model Fidelity	0	0.9769
Key Model Fidelity × Delay	0.14	0.7063
Key Model Fidelity × Jitter	0.24	0.6223
Environment Model Fidelity × Delay	0.24	0.6223
Environment Model Fidelity × Jitter	1.03	0.3109
Delay × Jitter	0.53	0.4689
Key Model Fidelity × Environment Model Fidelity × Delay	2.93	0.0882
Key Model Fidelity × Environment Model Fidelity × Jitter	0.37	0.5429
Key Model Fidelity × Delay × Jitter	0.04	0.8392
Environment Model Fidelity × Delay × Jitter	3.13	0.0781
Key Model Fidelity × Environment Model Fidelity × Delay × Jitter	1.28	0.2589

Table 4.8: Three-way ANOVA of the ODoT.

Source	F-statistics	p-value
Key Model Fidelity	6.33	0.0128
Environment Model Fidelity	0.12	0.7342
Delay	0.12	0.8826
Key Model Fidelity × Environment Model Fidelity	004	0.8385
Key Model Fidelity × Delay	0.28	0.7544
Environment Model Fidelity × Delay	0.48	0.6217
Key Model Fidelity × Environment Model Fidelity × Delay	0.6	0.5517

Table 4.9 shows the results of four-way ANOVA for the SDoT measure, and Table 4.10 shows the results of three-way ANOVA test. As opposed to the tests of significance for the ODoT measure, for the SDoT, the key model fidelity is not significant. Instead, the delay factor is statistically significant (with a p-value of 0.0281 for the four-way ANOVA test and 0.0018 for the three-way ANOVA test).

Further more, we analyze the interaction for factor combinations (just like what we did for the ODoT measure). The results demonstrate that there is no interaction among the four factors.

Table 4.9: Four-way ANOVA of SDoT.

Source	F-statistics	p-value
Key Model Fidelity	2.95	0.0871
Environment Model Fidelity	0.96	0.3271
Delay	4.88	0.0281
Jitter	2.95	0.0871
Key Model Fidelity × Environment Model Fidelity	0.38	0.54
Key Model Fidelity × Delay	0.02	0.9024
Key Model Fidelity × Jitter	0.38	0.54
Environment Model Fidelity × Delay	1.22	0.2704
Environment Model Fidelity × Jitter	1.82	0.1783
Delay × Jitter	0.74	0.3911
Key Model Fidelity × Environment Model Fidelity × Delay	0.24	0.6239
Key Model Fidelity × Environment Model Fidelity × Jitter	0.54	0.4622
Key Model Fidelity × Delay × Jitter	0.54	0.4622
Environment Model Fidelity × Delay × Jitter	2.95	0.0871
Key Model Fidelity × Environment Model Fidelity × Delay × Jitter	3.39	0.0669

Given the results obtained with the tests of significance, we conclude that the key model fidelity factor affects the ODoT significantly, while the delay factor affects the SDoT significantly. We could not show any influence of other factors beyond what may occur by chance. We summarize our results in Table 4.2. H_{01} and H_{07} are rejected since the key model fidelity and delay factors have significant effect on ODoT and SDoT respectively.

Table 4.10: Three-way ANOVA of SDoT.

Source	F-statistics	p-value
Key Model Fidelity	1.2	0.2743
Environment Model Fidelity	0.34	0.5623
Delay	6.55	0.0018
Key Model Fidelity × Environment Model Fidelity	0.04	0.8468
Key Model Fidelity × Delay	0.7	0.4964
Environment Model Fidelity × Delay	0.16	0.8503
Key Model Fidelity × Environment Model Fidelity × Delay	0.46	0.6309

Table 4.11: Hypotheses. The third column shows if a hypothesis is accepted or rejected according to our ANOVA analysis.

H_{01}	Different model fidelities do not affect the ODoT	reject
H_{02}	Different environment fidelities do not affect the ODoT	accept
H_{03}	Different network delays do not affect the ODoT	accept
H_{04}	Existence of jitter does not affect the ODoT	accept
H_{05}	Different model fidelities do not affect the SDoT	accept
H_{06}	Different environment fidelities do not affect the SDoT	accept
H_{07}	Different network delays do not affect the SDoT	reject
H_{08}	Existence of jitter does not affect the SDoT	accept

Now we can answer the four questions asked at the beginning of this section.

1. The key model fidelity has a significant effect on the ODoT. Greater key model fidelity helps the user to achieve better recognition. But we could not show a significant effect on the SDoT.

2. We could not show a significant effect of environment fidelity on the ODoT nor the SDoT.

3. The delay factor has a significant effect on the SDoT. Longer delay results in higher SDoT. It did not have a significant effect on the ODoT.

4. We could not show a significant effect of jitter on the ODoT nor the SDoT in our study.

In conclusion, our user study supports the idea of only rendering and sending key models in remote rendering applications. Based on an analysis of the variance of both, ODoT and SDoT, we found that key model fidelity is the only factor that has a statistically significant effect on ODoT, and that delay is the only factor that has a statistically significant effect on SDoT. Thus, our application design is well informed to address overall DoT by addressing the factors key model fidelity and delay which are the two factors that have significant effects. To minimize the network delay, we reduce the environment model fidelity. The

user study indicates that this reduction does likely not affect the SDoT. Moreover, since the factor key model fidelity has a significant effect on ODoT, our method maintains the fidelity of the key model.

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 rendered frames to the client. However, in our approach only key models are rendered on the server and sent 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. In this way, the client is able to display rendering effects that are not available with mobile graphic devices, with lower bandwidth requirements than in streaming-only solutions. Moreover, since low-fidelity models are stored and rendered on the client side, our framework is able to adapt to different network conditions, and an application is able to continue to run, even if the network becomes completely unavailable.

We conducted a user study on the factors involved in the proposed framework affecting the ODoT and the SDoT: key model fidelity, environment model fidelity, delay and jitter. We developed an experimental application in which user are asked to recognize objects in different configurations pertaining to the identified factors. Our user study shows that the key model fidelity affects the ODoT and that network delay plays an important role in the SDoT. When streaming over non-dedicated networks, the conditions often do not satisfy the requirements of remote rendering applications. Therefore, trading off between rendering quality and network delay is essential but existing remote rendering applications cannot address this issue in a satisfactory way. The most important contribution of the proposed framework is enabling such a trade-off.

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

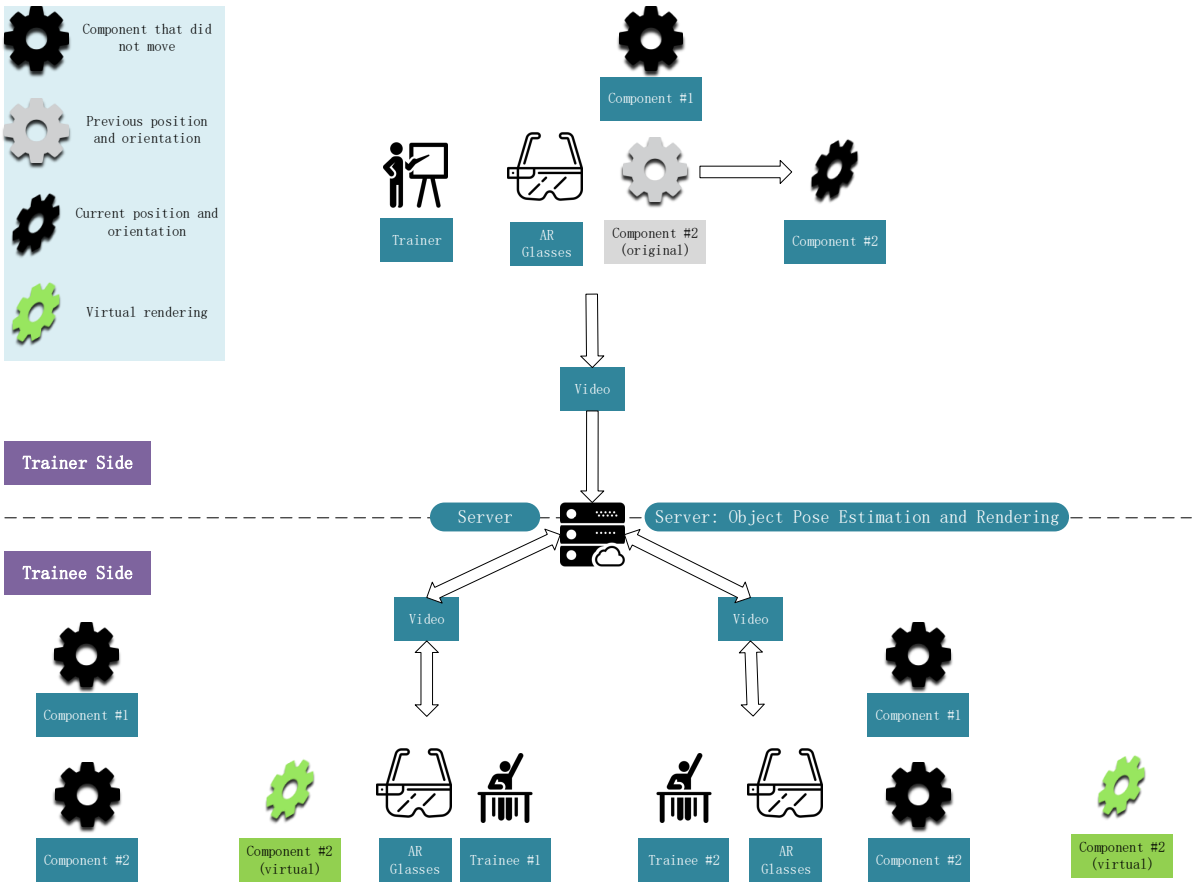


Figure 5.1: Training Scenario Part 1. This figure shows how the operations made by the trainer are rendered on the trainee side. The gears represent the components in the training scenario. The black gears denote the real components, while grey ones are their previous poses and the green gears demonstrate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

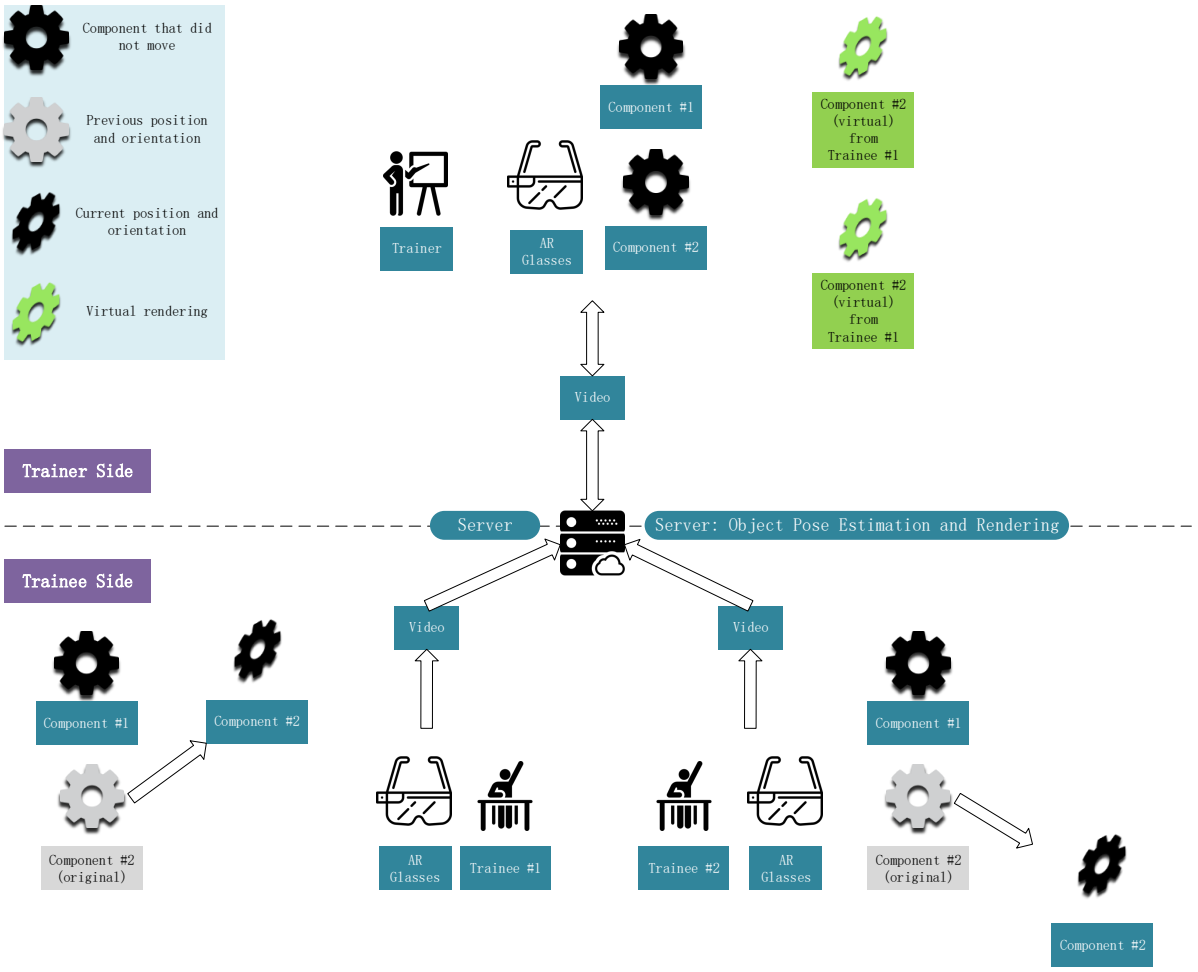


Figure 5.2: Training Scenario Part 2. This figure shows how the operations made by the trainees are rendered on the trainer side. The gears represent the components in the training scenario. The black gears denote the real component, while grey ones are their previous poses and the green gears demonstrate 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.

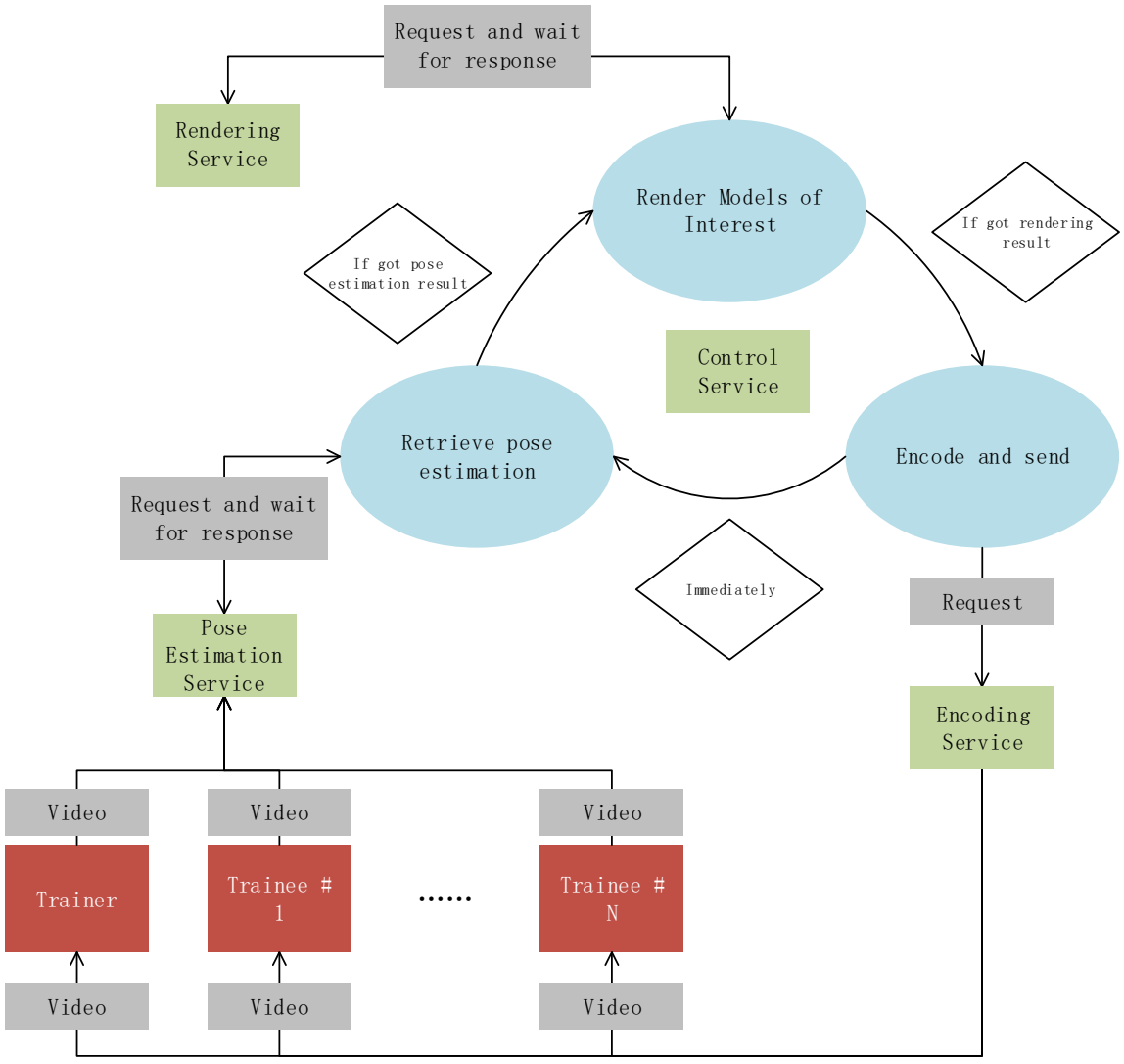


Figure 5.3: Communication Schema

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 [20].

Some model-based approaches use edge or point features associated with the 3D models for estimating the pose [32, 87, 64, 40]. 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 [68] 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 [22] 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 [67]. Tjaden et al. built an algorithm based on PWP3D, which improves convergence properties, especially for rotational motion [86]. 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. [86] 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.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(11):2274–2281, 2012.

[2] Fraser Anderson, Tovi Grossman1, Justin Matejka1, and George Fitzmaurice1. Youmove: Enhancing movement training with an augmented reality mirror. In *ACM User Interface Software and Technology Symposium*, October 2013.

[3] Paul Bao and Douglas Gourlay. A framework for remote rendering of 3-d scenes on limited mobile devices. *IEEE Transactions on Multimedia*, 8(2):382–389, April 2006.

[4] Paul Bao, Douglas Gourlay, and Youfu Li. Deep compression of remotely rendered views. *IEEE Transactions on Multimedia*, 8(3):444–456, June 2006.

[5] M. Billinghurst, H. Kato, and I. Poupyrev. The magicbook - moving seamlessly between reality and virtuality. *IEEE Computer Graphics and Applications*, 21(3):6–8, May 2001.

[6] Mark Billinghurst and Hirokazu Kato. Collaborative augmented reality. *Commun. ACM*, 45(7):64–70, July 2002.

[7] A. Boukerche and R. W. N. Pazzi. Remote rendering and streaming of progressive panoramas for mobile devices. In *Proceedings of the 14th Annual ACM International Conference on Multimedia*, Multimedia '06, pages 691–694, 2006.

[8] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms forenergy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1124–1137, September 2004.

[9] Thomas Brox and Jitendra Malik. Object segmentation by long term analysis of point trajectories. In *Europ. Conf. Comput. Vis.*, pages 282–295, September 2010.

[10] T. P. Caudell and D. W. Mizell. Augmented reality: an application of heads-up display technology to manual manufacturing processes. In *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences*, volume ii, pages 659–669 vol.2, Jan 1992.

[11] J. Cecil, P. Ramanathan, V. Rahneshin, A. Prakash, and M. Pirela-Cruz. Collaborative virtual environments for orthopedic surgery. In *2013 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 133–137, Aug 2013.

[12] C.-F. Chang and S.-H. Ger. Enhancing 3d graphics on mobile devices by image-based rendering. In *In Proceedings of the 3rd IEEE Pacific Rim Conference on Multimedia: Advances in Multimedia Information Processing*, PCM '02, 2002.

[13] Chuo-Ling Chang and B. Girod. Receiver-based rate-distortion optimized interactive streaming for scalable bitstreams of light fields. In *2004 IEEE International Conference on Multimedia and Expo (ICME) (IEEE Cat. No.04TH8763)*, volume 3, pages 1623–1626 Vol.3, June 2004.

[14] C. Chen, J. Cai, W. Lin, and G. Shi. Incremental low-rank and sparse decomposition for compressing videos captured by fixed cameras. *Vis. Comm. and Image Rep.*, 26:338–348, 2015.

[15] K. T. Chen, Y. C. Chang, H. J. Hsu, D. Y. Chen, C. Y. Huang, and C. H. Hsu. On the quality of service of cloud gaming systems. *IEEE Transactions on Multimedia*, 16(2):480–495, Feb 2014.

[16] Shenchange Eric Chen. Quicktime vr: an image-based approach to virtual environment navigation. In *Proceedings of the 22nd annual conference on computer graphics and interactive techniques*, SIGGRAPH '95, 1995.

[17] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. Clonecloud: Elastic execution between mobile device and cloud. In *Proceedings of the Sixth Conference on Computer Systems*, EuroSys '11, pages 301–314, New York, NY, USA, 2011. ACM.

[18] A. Colombari, A. Fusiello, , and V. Murino. Segmentation and tracking of multiple video objects. *Pattern Recognition*, 40(4):1307–1317, 2007.

[19] F. De Crescenzio, M. Fantini, F. Persiani, L. Di Stefano, P. Azzari, and S. Salti. Augmented reality for aircraft maintenance training and operations support. *IEEE Computer Graphics and Applications*, 31(1):96–101, Jan 2011.

[20] R. Deriche D. Cremers, M. Rousson. A review of statistical approaches to level set segmentation. *Integrating color, texture, motion and shape. Int. J. Comput. Vision*, 72:195–215, 2007.

[21] Meir Johnathan Dahan, Nir Chen, Ariel Shamir, and Daniel Cohen-Or. Combining color and depth for enhanced image segmentation and retargeting. *The Vis. Comput.*, 28(12):1181–1193, 2012.

[22] S. Dambreville, R. Sandhu, A. Yezzi, and A. Tannenbaum. A geometric approach to joint 2d region-based segmentation and 3d pose estimation using a 3d shape prior. *SIAM J. Img. Sci.*, 3:110–132, 2010.

[23] M. Decombas, E. Renan, F. Capman, F. Dufaux, and B. Pesquet-Popescu. Closed loop seams approximation for video compression. In *Proc. IEEE Int. Symp. signal, Image, Video, Commun.*, July 2012.

[24] Ian Endres and Derek Hoiem. Category independent object proposals. In *Europ. Conf. Comput. Vis.*, pages 575–588, September 2010.

[25] Rubina Freitas and Pedro Campos. Smart: A system of augmented reality for teaching 2nd grade students. In *Proceedings of the 22Nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction - Volume 2*, BCS-HCI '08, pages 27–30, Swindon, UK, 2008. BCS Learning & Development Ltd.

[26] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 209–216, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[27] Nirit Gavish, Teresa Gutirrez, Sabine Webel, Jorge Rodrguez, Matteo Peveri, Uli Bockholt, and Franco Tecchia. Evaluating virtual reality and augmented reality training for industrial maintenance and assembly tasks. *Interactive Learning Environments*, 23(6):778–798, 2015.

[28] Mar González-Franco, Julio Cermeron, Katie Li, Rodrigo Pizarro, Jacob Thorn, Paul Hannah, Windo Hutabarat, Ashutosh Tiwari, and Pablo Bermell-Garcia. Immersive augmented reality training for complex manufacturing scenarios. *CoRR*, abs/1602.01944, 2016.

[29] Florian Gosselin, Fabien Ferlay, Sylvain Bouchigny, Christine Mégard, and Farid Taha. *Design of a Multimodal VR Platform for the Training of Surgery Skills*, pages 109–116. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[30] Ramrez A. Graciela and Chacn M. Mario I. New trends on dynamic object segmentation in video sequences: A survey. *RIEE&C*, 11(1):29–42, 2013.

[31] D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *J. Royal Statis. Soc. Series B*, 51(2):271–279, 1989.

[32] C. Harris and C. Stennet. Rapid ? a video-rate object tracker. In *British Machine Vision Conference*, September 1990.

[33] Mahdi Hemmati, Abbas Javadtalab, Ali Asghar Nazari Shirehjini, Shervin Shirmohammadi, and Tarik Arici. Game as video: Bit rate reduction through adaptive object encoding. In *Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '13, pages 7–12, New York, NY, USA, 2013. ACM.

[34] S. J. Henderson and S. Feiner. Evaluating the benefits of augmented reality for task localization in maintenance of an armored personnel carrier turret. In *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, pages 135–144, Oct 2009.

[35] Mauricio Hincapi, Andrea Caponio, Horacio Rios, and Eduardo Gonzlez Mendvil. An introduction to augmented reality with applications in aeronautical maintenance. In *Transparent Optical Networks (ICTON), 2011 13th International Conference on*, June 2011.

[36] H. J. Hong, C. F. Hsu, T. H. Tsai, C. Y. Huang, K. T. Chen, and C. H. Hsu. Enabling adaptive cloud gaming in an open-source cloud gaming platform. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(12):2078–2091, Dec 2015.

[37] Ramesh Jain and H.-H. Nagel. On the analysis of accumulative difference pictures from image sequences of real world scenes. *IEEE Trans. Pattern Anal. Mach. Intell.*, PAMI-1(2):206–214, April 1979.

[38] L. Johnson, A. Levine, R. Smith, and S. Stone. Simple augmented reality. *2010 Horizon Report*, 211:250–255, 2010.

[39] P. KaewTraKulPong and R. Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In *Proc. Europ. Workshop Adv. Video Based Surveillance Syst. (AVBS '01)*, pages 135–144, September 2001.

[40] K. Kim, V. Lepetit, and W. Woo. Keyframe-based modeling and tracking of multiple 3d objects. In *ISMAR 2010. 9th IEEE International Symposium on*, pages 193–198, October 2010.

[41] Donghyun Kwon, Seungjun Yang, Yunheung Paek, and Kwangman Ko. Optimization techniques to enable execution offloading for 3d video games. *Multimedia Tools and Applications*, 76(9):11347–11360, May 2017.

[42] F. Lamberti and A. Sanna. A streaming-based solution for remote visualization of 3d graphics on mobile devices. *IEEE Trans. Vis. Comput. Graph*, 13:247–260, 2007.

[43] Fabien Leblanc, Bradley J Champagne, Knut M Augestad, Paul C Neary, Anthony J Senagore, Clyde N Ellis, Conor P Delaney, and the Colorectal Surgery Training Group. A comparison of human cadaver and augmented reality simulator models for straight laparoscopic colorectal skills acquisition training. *J Am Coll Surg*, pages 21–24, 2010.

[44] Kangdon Lee. Augmented reality in education and training. *TechTrends*, 56(2):13–21, Mar 2012.

[45] Y. J. Lee, J. Kim, and K. Grauman. Key-segments for video object segmentation. In *IEEE Int. Conf. Comput. Vis.*, November 2011.

[46] Marc Levoy. Polygon-assisted jpeg and mpeg compression of synthetic images. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 21–28, 1995.

[47] Fuxin Li, Taeyoung Kim, Ahmad Humayun, David Tsai, and James M. Rehg. Video segmentation by tracking many figure-ground segments. In *IEEE Int. Conf. Comput. Vis.*, pages 2192–2199, June 2013.

[48] Y. Li, E. Adelson, and A. Agarwala. Instant propagation of sparse edits on images and videos. *Comput. Graphics Forum*, 29(7):2049–2054, 2010.

[49] Xiaohui Liang, Qinping Zhao, Zhiying He, Ke Xie, and Yubo Liu. A point-based rendering approach for real-time interaction on mobile devices. *Science in China Series F: Information Sciences*, 52(8):1335–1345, Aug 2009.

[50] Yao Liu, Shaoxuan Wang, and Sujit Dey. Content-aware modeling and enhancing user experience in cloud mobile rendering and streaming. *IEEE J. on Emerging and Selected Topics in Circuits and Systems*, 4:43–56, March 2014.

[51] Yan Lu, Shipeng Li, and Huifeng Shen. Virtualized screen: A third element for cloud-mobile convergence. *IEEE Multimedia*, 18:4–11, April 2011.

[52] Yao Lu, Yao Liu, and Sujit Dey. Asymmetric and selective object rendering for optimized cloud mobile 3d display gaming user experience. *Multimedia Tools and Applications*, 76(18):18291–18320, Sep 2017.

[53] Xun Luo, R. V. Kenyon, T. Kline, H. C. Waldinger, and D. G. Kamper. An augmented reality training environment for post-stroke finger extension rehabilitation. In *9th International Conference on Rehabilitation Robotics, 2005. ICORR 2005.*, pages 329–332, June 2005.

[54] Tianyang Ma and L.J. Lateck. Maximum weight cliques with mutex constraints for video object segmentation. In *IEEE Conf. Comput. Vis. Pattern Recogn.*, pages 670–677, June 2012.

[55] Z. Ma, T. Yue, X. Cao, Y. Xu, X. Li, and Y. Wang. Interactive screen video streaming-based pervasive mobile workstyle. *IEEE Transactions on Multimedia*, 19(10):2322–2332, Oct 2017.

[56] B. Macchiavello, C. Dorea, E. M. Hung, G. Cheung, and W. T. Tan. Loss-resilient coding of texture and depth for free-viewpoint video conferencing. *IEEE Transactions on Multimedia*, 16(3):711–725, April 2014.

[57] C. Nakajima and N. Itho. A support system for maintenance training by augmented reality. In *12th International Conference on Image Analysis and Processing, 2003.Proceedings.*, pages 158–163, Sept 2003.

[58] Hiroaki Nishino, Kouta Murayama, Kazuya Shuto, Tsuneo Kagawa, and Kouichi Utsumiya. A calligraphy training system based on skill acquisition through haptization. *Journal of Ambient Intelligence and Humanized Computing*, 2(4):271–284, Dec 2011.

[59] Y. Noimark and D. Cohen-Or. Streaming scenes to mpeg-4 video-enabled devices. *IEEE Computer Graphics and Applications*, 23(1):58–64, Jan 2003.

[60] P. Ochs and T. Brox. Object segmentation in video: a hierarchical variational approach for turning point trajectories into dense regions. In *IEEE Int. Conf. Comput. Vis.*, November 2011.

[61] Anestis Papazoglou and Vittorio Ferrari. Fast object segmentation in unconstrained video. In *IEEE Int. Conf. Comput. Vis.*, pages 1777–1784, 2013.

[62] G. Paravati, C. Celozzi, A. Sanna, and F. Lamberti. A feedback-based control technique for interactive live streaming systems to mobile devices. *IEEE Transactions on Consumer Electronics*, 56(1):190–197, February 2010.

[63] Jiro Park and Haeyoung Lee. A hierarchical framework for large 3d mesh streaming on mobile systems. *Multimedia Tools and Applications*, 75(4):1983–2004, Feb 2016.

[64] Y. Park, V. Lepetit, and W. Woo. Multiple 3d object tracking for augmented reality. in: Mixed and augmented reality. In *ISMAR 2008. 7th IEEE/ACM Interna- tional Symposium on.*, pages 117–120, September 2008.

[65] S.G. Perlman, R. Van der Laan, T. Cotter, S. Furman, R. McCool, and I. Buckley. System and method for multi-stream video compression using multiple encoding formats, July 1 2010. US Patent App. 12/538,092.

[66] P.Ochs and T.Brox. Higher order motion models and spectral clustering. In *IEEE Conf. Comput. Vis. Pattern Recogn.*, pages 614–621, June 2012.

[67] V. Prisacariu, O. Kahler, D. Murray, and I. Reid. Real-time 3d tracking and reconstruction on mobile phones. *IEEE Transactions on Visualization and Computer Graphics*, 21:557–570, 2015.

[68] V.A. Prisacariu and I.D. Reid. Pwp3d: Real-time segmentation and tracking of 3d objects. *Int. J. Comput. Vision*, 98:335–354, 2012.

[69] Peter Quax, Jori Liesenborgs, Arno Barzan, Martijn Croonen, Wim Lamotte, Bert Vankeirsbilck, Bart Dhoedt, Tom Kimpe, Kurt Pattyn, and Matthew McLin. Remote rendering solutions using web technologies. *Multimedia Tools and Applications*, 75(8):4383–4410, Apr 2016.

[70] P. Ramanathan, M. Kalman, and B. Girod. Rate-distortion optimized interactive light field streaming. *IEEE Transactions on Multimedia*, 9(4):813–825, June 2007.

[71] H. Regenbrecht, G. Baratoff, and W. Wilke. Augmented reality projects in the automotive and aerospace industries. *IEEE Computer Graphics and Applications*, 25(6):48–56, Nov 2005.

[72] Luis Rodriguez-Gil, Pablo Orduña, Javier García-Zubia, and Diego López-de Ipiña. Interactive live-streaming technologies and approaches for web-based applications. *Multimedia Tools and Applications*, Mar 2017.

[73] K. L. Schrier. Revolutionizing history education: using augmented reality games to teach histories. Master's thesis, Massachusetts Institute of Technology.

[74] S. Shi, K. Nahrstedt, and R. Campbell. A real-time remote rendering system for interactive mobile graphics. *ACM Trans. Multimedia Comput., Commun., Appl. (TOMCCAP)*, 8(46):43–56, 2012.

[75] Shu Shi and Cheng-Hsin Hsu. A survey of interactive remote rendering systems. *ACM Computing Surveys*, 47:57, July 2015.

[76] Pieter Simoens, Bojan Joveski, Ludovico Gardenghi, Iain James Marshall, Bert Vankeirsbilck, Miha Mitrea, Francoise Prêteux, Filip De Turck, and Bart Dhoedt. Optimized mobile thin clients through a mpeg-4 bifs semantic remote display framework. *Multimedia Tools and Applications*, 61(2):447–470, Nov 2012.

[77] Parthipan Siva and Alexander Wong. Grid seams: A fast superpixel algorithm for real-time applications. In *Canadian Conf. Comput. Robot Vis.*, pages 127–134, 2014.

[78] Jack Sklansky. Finding the convex hull of a simple polygon. *Pattern Recogn. Lett.*, 1(2):79–83, December 1982.

[79] I. Slivar, M. Suznjevic, and L. Skorin-Kapov. The impact of video encoding parameters and game type on qoe for cloud gaming: A case study using the steam platform. In *2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX)*, pages 1–6, May 2015.

[80] Andrews Sobrala and Antoine Vacavant. A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. *Comput. Vis. Image Understanding*, 122:4–21, 2014.

[81] Ivan E. Sutherland. A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, AFIPS '68 (Fall, part I), pages 757–764, New York, NY, USA, 1968. ACM.

[82] M. Suznjevic, I. Slivar, and L. Skorin-Kapov. Analysis and qoe evaluation of cloud gaming service adaptation under different network conditions: The case of nvidia geforce now. In *2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6, June 2016.

[83] Satoshi Suzuki and Keiichi Abe. Topological structural analysis of digitized binary images by border following. *Comput. Vision Graphics Image Process.*, 30(1):32–46, 1985.

[84] M. W. Tao and A. Krishnaswamy. Fast adaptive edge-aware mask generation. In *Proc. Graphics Interface*, pages 77–83, 2012.

[85] B. Taylor, V. Karasev, and S. Soatto. Causal video object segmentation from persistence of occlusions. In *IEEE Conf. Comput. Vis. Pattern Recogn.*, June 2015.

[86] Henning Tjaden, Ulrich Schwanecke, and Elmar Schmer. Real-time monocular segmentation and pose tracking of multiple objects. In *Europ. Conf. Comput. Vis.*, September 2016.

[87] L. Vacchetti, V. Lepetit, and P. Fua. Stable real-time 3d tracking using online and offline information. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26:1385–1391, 2004.

[88] Ming-Jen Wang, Chien-Hao Tseng, and Cherng-Yeu Shen. *An Easy to Use Augmented Reality Authoring Tool for Use in Examination Purpose*, pages 285–288. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[89] S. Wang, J. Fu, Y. Lu, S. Li, and W.M. Gao. Content-aware layered compound video compression. In *Circuits and Systems (ISCAS)*, May 2012.

[90] Wenguan Wang, Jianbing Shen, and Fatih Porikli. Saliency-aware geodesic video object segmentation. In *IEEE Conf. Comput. Vis. Pattern Recogn.*, pages 3395–3402, June 2015.

[91] Sabine Webel, Uli Bockholt, Timo Engelke, Nirit Gavish, Manuel Olbrich, and Carsten Preusche. An augmented reality training platform for assembly and maintenance skills. *Robotics and Autonomous Systems*, 61(4):398 – 403, 2013. Models and Technologies for Multi-modal Skill Training.

[92] Sabine Webel, Ulrich Bockholt, Timo Engelke, Matteo Peveri, Manuel Olbrich, and Carsten Preusche. Augmented reality training for assembly and maintenance skills. In *The International Conference SKILLS 2011*, December 2011.

[93] Yajie Yan, Xiaohui Liang, Ke Xie, and Qinping Zhao. Asehm: a new transmission control mechanism for remote rendering system. *Multimedia Tools and Applications*, 69(3):585–603, Apr 2014.

[94] S. Yang, Y. Kwon, Y. Cho, H. Yi, D. Kwon, J. Youn, and Y. Paek. Fast dynamic execution offloading for efficient mobile cloud computing. In *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 20–28, March 2013.

[95] Steve Chi-Yin Yuen, Gallayanee Yaoyuneyong, and Erik Johnson. Augmented reality: An overview and five directions for ar in education. *Journal of Educational Technology Development & Exchange*, 4:119–140, 2011.

[96] Hui-Chi Zeng and Shang-Hong Lai. Adaptive foreground object extraction for real-time video surveillance with lighting variations. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, pages 1201–1204, April 2007.

[97] Dong Zhang, Omar Javed, and Mubarak Shah. Video object segmentation through spatially accurate and temporally dense extraction of primary object regions. In *IEEE Conf. Comput. Vis. Pattern Recogn.*, pages 628–635, June 2013.

[98] Feng Zhou, Henry Been-Lirn Duh, and Mark Billinghurst. Trends in augmented reality tracking, interaction and display: A review of ten years of ismar. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '08, pages 193–202, Washington, DC, USA, 2008. IEEE Computer Society.

[99] Wenwu Zhu, Y. T. Hou, Yao Wang, and Ya-Qin Zhang. End-to-end modeling and simulation of mpeg-2 transport streams over atm networks with jitter. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(1):9–12, Feb 1998.

[100] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Proc. 17th Int. Conf. Pattern Recogn.*, pages 28–31, August 2004.