Project VCR - Repurposing the Mobile Phone

Asim Fauzi[*]
asim7@kaist.ac.kr
KAIST
School of Computing
Daejeon, Republic of Korea

Daehyeon Nam
ndh8392@kaist.ac.kr
KAIST
School of Computing
Daejeon, Republic of Korea

ABSTRACT

Most VR applications nowadays require some sort of remote controller device paired with an expensive headset. Those applications relying on mobile devices are usually limited to a simple visual experience with no controls. This research targets this problem by enabling controllability without external devices besides a head-mount and the mobile device itself. We provide an innovative solution of capturing hand-landmark data after several stages of processing and use this data to render hand information in the virtual environment. The hand-landmarking is computed using a hand-pose model on streamed images from the mobile device itself. To-https://www.overleaf.com/project/5dfc1c4a7979160001cd0ece tackle the issues of thermal throttling as well as mobile CPU limitations, our main innovation comes from off-loading as much of the workload as possible on designated servers with an image partitioning algorithm. By using a simple server request on future VR applications, hand data can be sent back to the client to mimic real-life hand controllability. Our results proved satisfying with variable sub-second latency considering the whole roundtrip time from client to server to client. In conclusion, we believe this work could pave way for more use cases in mobile-offloading – specifically, for heavy virtual reality computing.

CCS CONCEPTS

• **Human-centered computing → Virtual reality**; *Software and its engineering*; • **Computing methodologies →** Parallel server computing.

KEYWORDS

mobile phone, virtual reality, off-loading, multi-server

[*]Both authors contributed equally to this research.

Eoeun '19, December 19–23, 2019, Eoeun, DJ
© 2019
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.0000/970612.990314

ACM Reference Format:
Asim Fauzi and Daehyeon Nam. 2019. Project VCR - Repurposing the Mobile Phone. In *Eoeun '19: KAIST Symposium on mobile computing, December 19–23, 2019, Eoeun, DJ.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.0000/970612.990314

1 INTRODUCTION

Current limitations of virtual reality include the requirement of certain hardware such as a powerful GPU and CPU. In addition, certain platforms require specific headsets and controllers which are expensive and usually become outdated within a few years. A solution to this second problem could be solved by utilizing what the common person already owns – a mobile device. However, mobile devices are unable to perform such heavy computations with their current limitations. The most notable limitations on modern mobile devices derives mainly from two things; networking and processing power to size ratio. Our second degree solution includes the combination of 5G networking as well as computational server-offloading.

Why are we interested on making VR more universal? We believe VR can be a strong substitute for current daily drivers like laptops and desktops while increasing ease of use and portability. Similar to how the current iPad pro is slowly substituting traditional note-taking methods, we believe by creating virtual workspace environments – we can find a strong alternative for any kind of desktop work whether that be browsing, editing, presenting, etc. The basic idea is that someone would no longer have to carry around their laptop, laptop peripherals, nor note-taking devices. Instead, they can now carry a simple head mount along with their mobile-device to perform the same tasks.

Our contribution is in 2 parts. First, we tried to do off-loading for VR environment. The off-loading used to be hard problem in VR since most of computation should be done in real-time and networking usually have big latency. But as the 5G networking is already here, we tried to do off-loading for most of computation which needs to be in real-time. Even if we did not actually use 5G, we showed the possibility that VR off-loading in real-time can be done. Furthermore, since we used server, we could use heavy models to compute which mean we can overcome the resource of the mobile phone. And paradoxically we could make it in real-time computation

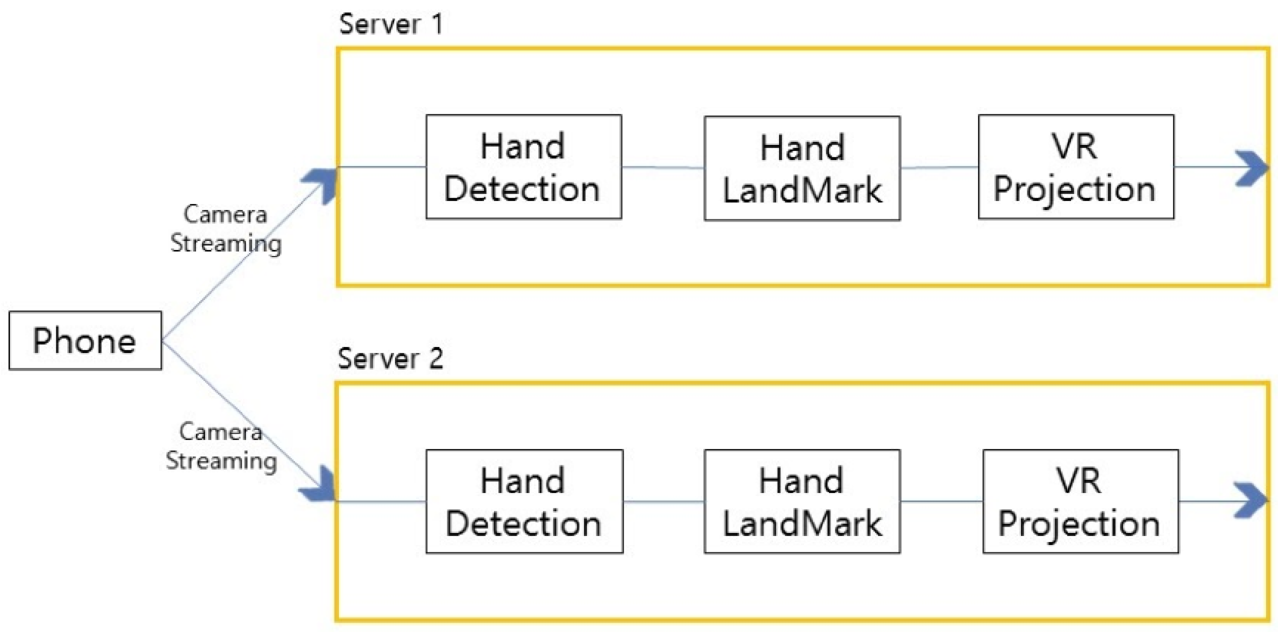

Figure 1: VCR architecture: The phone streams the real-time images to each of server, then each server detects hands and decide which hand to process. So each of server process each of hand. After the decision, servers skeletonize the hands using hand landmark algorithm and project the landmarks on the VR environment.

because of off-loading even if we need heavy computation. Second, multi-server computation for image processing. As we will see, we split one big task into multiple small tasks and we assigned the multiple small tasks into each different servers. Specifically, we had a picture with 2 hands. Instead of processing one server processes one whole picture, we divided the picture into 2 pictures according to the bounding boxes of hands and assigned each of hand to each of server. By using this, we could implement external level of parallel computing. And in the future when this process be flexible, depend on the amount of computation which needs to be done in real-time we can control the number of servers that we use.

2 BACKGROUND AND RELATED WORK

There were many background works we had to look into in regards to virtual reality computing and off-loading as the project itself is quite inter-disciplinary. Off-loading of heavy GPU processing is still a relatively new scene considering recent advancements in GPU architecture. And virtual reality is no different as it is a relatively new concept in general with no real mainstream audience. If we were to divide this project into separate fields of computer science, we would end up with computer visions, computer graphics, networking, and virtual reality. Computer visions is vital to the image partitioning as well as hand-landmark computing. Computer graphics is important for understanding the relationship of hand data on the given image to the resulting world space. Networking is necessary for communications between client and server with 5G being an important player for the future success of this research. Virtual reality is the main motivation of this research and one must understand the depths of its processing requirements as well as its interactivity using specific hardware and peripherals.

3 VCR ARCHITECTURE AND IMPLEMENTATION

Figure 1 presents the overall architecture of VCR. We describe the high level of flow through this architecture, and present the internal details later. We begin with a mobile phone user wearing the VR holder with their phone. The phone takes a real-time image flow from the rear camera and streams it to each server. Then each server that receives the real-time image flow will detect the hands in the image. Servers can detect up to two hands, and each server selects and processes one of them. The area where the hand is detected will be cropped and each server will parse coordinates of landmarks in the each of cropped hand image by using hand landmark algorithm. Finally the server projects the coordinates of landmarks on the VR environment. We used both of windows10 and ubuntu18.04 as our main operating system, and python3.6 for implementation.

Camera streaming

In this stage, we tried to stream the real-time image from phone to the servers. We used the IP webcam android application to stream the real-time image. Basically this application builds the local web-server on the phone and streams on that server. And client can access to the video being streamed by accessing the local web-server that application built. So each of server accessed to that web-server to get real-time image from the phone. Additionally, We set the video resolution 320 x 480 to reduce the latency in streaming. And we noticed that because of narrow angle of camera view, it could not detect the hands very well. So we used external wide-angle lens to make the camera view wide.

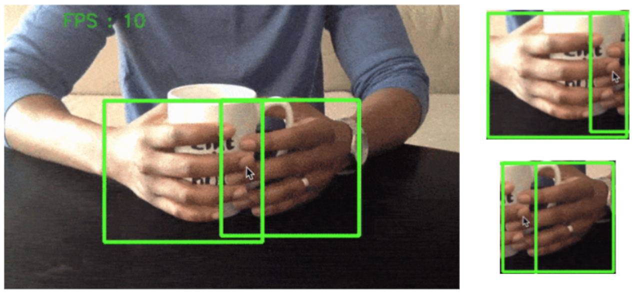

Figure 2: Sample detected hand area: From each of real-time image, servers detect the hands and crop the hand detected area from the image.

Hand detection

As figure 2 shows, in this stage, each server detects the hands from the real-time image and crops the hand detected area. We used pre-trained mobilenet model to detect the hands. The pre-trained model was trained using Egohands dataset. We set the maximum number of hands which can be detected as 2. And we assigned the each of cropped hand image to each of server to process further. The assignment was based on the probability of the hand image, so the image with highest probability of hand is always processed in the server 1, and the second highest probability of hand is always processed in the server2. Since the we could get only 7 frames per second with a thread, we used multi-thread to decrease the latency and we could get about 17 frames per second using 4 threads. From the hand detected area, we expanded the area by 20 pixels on top, bottom, left and right to make the detected area more flexible.

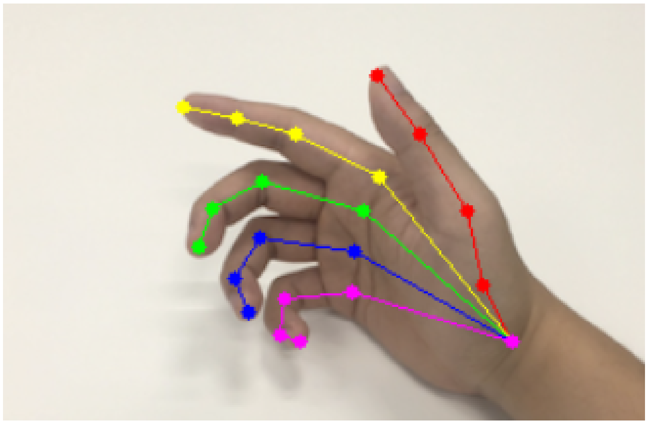

Figure 3: Sample hand landmark detection: From the cropped hand image, servers detect the landmarks of a hand by using the hand landmark model.

Hand landmark

In this stage, each server detects the hand landmarks from the cropped hand image. We used pre-trained vgg-net model to detect the landmarks. There are 4 landmarks in each of finger and one more landmark in the palm. So the model detects the 21 landmark points from the hand image as shown in figure 3. Calculating this model can not be done in real-time without graphic card computation. We used RTX2070 for each of server to compute this model and the computation could be done in almost real-time but with about 0.3s latency.

VR projection

In order for proper interactivity in the virtual environment using one's hands, we had to derive the world space information directly from the processed image using VR projection. This type of projection is a type of perspective projection in which we calculate world space based on the user's perspective. By curating a fake 'eye' behind a 2D plane (the phone's processed image in this case), we can shoot a ray from the eye through the plane and capture that resulting pixel information in world space. The given information on the 2D plane includes both an X and Y coordinate which can be scaled to the field of view of the capturing lens. However, another important piece to this calculation is depth. Depth is important for calculating a proper z-coordinate value in world space, otherwise the ray that is shot out from the 'eye' would keep on going resulting in improper 3D coordinates. For the current scope of this research, we omitted the option of calculating depth as the current methods would add too much to the computing stack. The other naive solution would be to calculate depth using bounding boxes or distancing of landmarks, but this would result in a higher inaccuracy considering these methods are not rotation or translation invariant to the capturing lens.

Individual Contribution and challenges we faced

Daehyeon focused on hand detection and streaming. We spent long time to do streaming from the phone to server. We first tried to make run python codes on android using the python application called pydroid. But after we implemented, we noticed that the python in android does not support online networking. Since we spent too much time for something not main concept idea, we just tried to use already built android application which is IP webcam. We could also try to make an android application that works as a client and streams to the server, but we thought we should not spend more time for this stage. For the hand detection, we notice that sometimes the bounding box of a detected hand was smaller than the an actual hand image. So we added 20pixels of pads on every edge to make it more flexible.
Asim focused on projection on VR environment, and hand landmark.

4 PROTOTYPE

We implemented as we described in previous section and we tested the result with our phone and 2 servers. Since the accuracy of hand detection was not good enough, we used controlled slicing of two hands and each server processes each hand. Controlled slicing means instead of cropping using bounding boxes from the hand detection, slicing the each of image into half vertically.

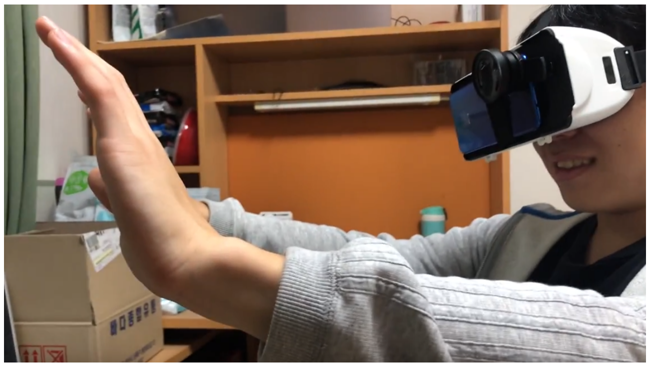

Figure 4: VCR with VR holder: Daehyeon is wearing the VCR with VR holder. The phone in VCR is streaming real-time hands image flow to the server.

Figure 4 shows how our VCR can be used. Daehyeon is wearing the VCR which is VR holder with the phone, the phone is capturing the hand image flow from the rear camera and streaming it to the server. Then, as shown in figure 5, each of server detects the hand landmarks from each half

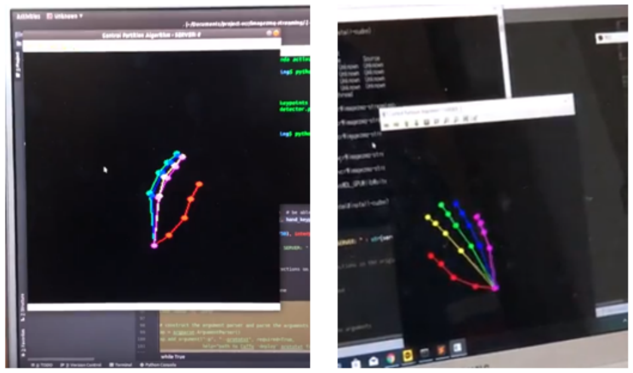

Figure 5: Skeletonized left hand is shown in left, and right hand is shown in right.: We used two servers to process each of hand. And in each server detected landmarks from each of hand and skeletonized it by connecting landmarks.

of image which means each hand. And we connected the landmarks to make a hand skeleton.

5 LIMITATIONS AND FUTURE WORK

Throughout the course of this research, we ran into many road-blocks and redirects with just how much this field is untouched. Our biggest limitation and concern in this research would be the latency issue of RTT. The time it takes to send a captured phone image and stream it to multiple servers to process for landmark data is quite strenuous with how many steps the input takes before reaching the final output stage back in the client. In VR latency is quite important since interactivity within 3D scenes must be relatively real-time or else a lag-effect would be present which would completely undermine the purpose of using a virtual reality environment. The second biggest concern of ours would be actual capturing of one's hands from the device's lens. In traditional VR scenes, one's interactivity is not limited to the vision of the headset. If one is holding an object outside their field of view, that object is still considered as held because of the remote controller's ability to track location without field of view. Whereas in our research, we are limited to the field of view vision if we want to render the most current hand-information. We believe that the first of these problems can be somewhat improved with the aid of a proper 5G network and redevelopment using low-level languages like C++. The second problem can be solved with the similar UE concept of a 'monitors' edge'. We can leave the last computed hand-location as the current hand location if the hand is not in the headset's field of view.

Limitation

There are some limitations we found. First the hand detection and hand landmark algorithm does not work well when

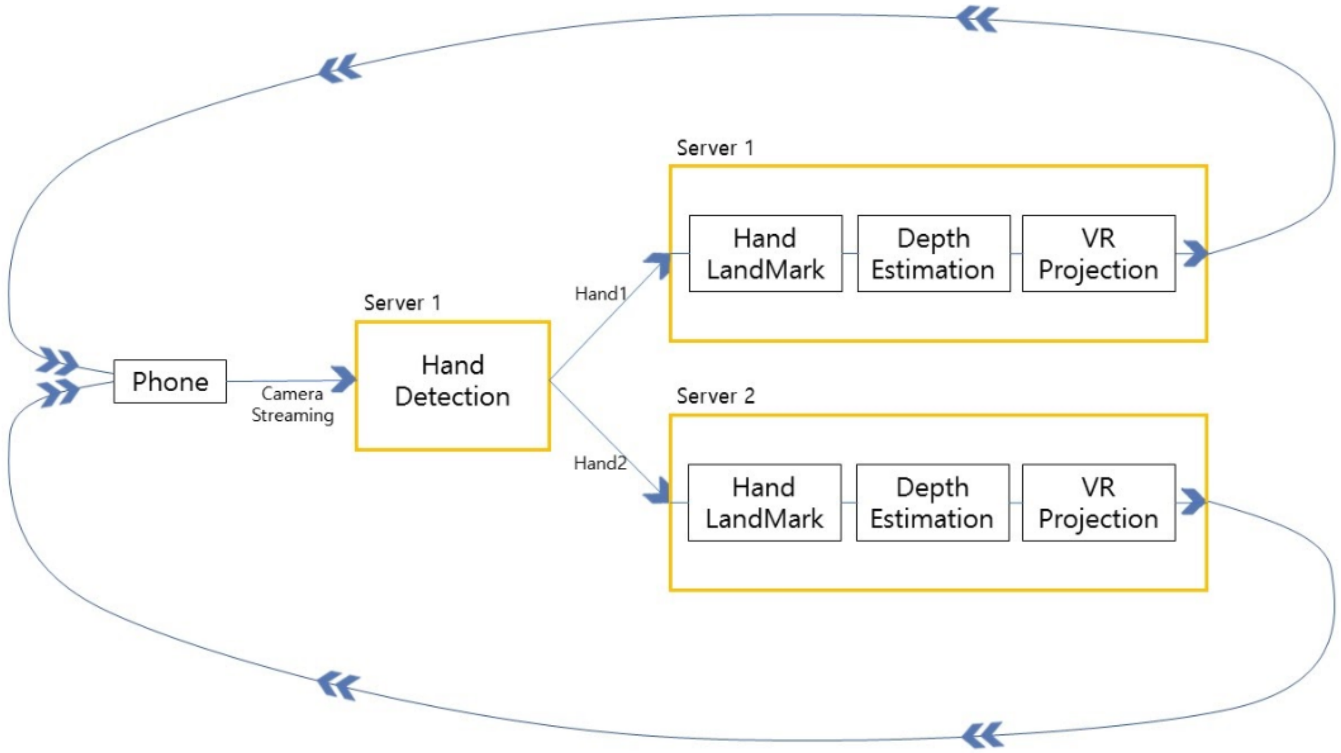

Figure 6: Target VCR architecture: Instead of using hand detection in each server, we will use one hand detection in a server and assign the tasks to servers. In each of server we will add depth estimation function which will estimate the z coordinate. Finally, we will stream the coordinates of landmarks back to phone to really interact with objects in VR environment

the room was dark, in other words, it recognize the hand properly without sufficient light. The one possible solution would be training with the hand in dark room. But if still accuracy is not enough in dark room, since TOF sensor in smart phone is getting popular we can try to use TOF sensor data in phone as well. Furthermore, if we use TOF sensor, the depth estimation also can be more accurate. The second limitation we found is the camera angle of view. Even if we used the external wide-angle lens, it was hard to recognize the hands when I didn't lift my hands. Third limitation is the location of camera in the phone. Usually the camera of phone are located top of the phone. For this reason, if we put it into the VR holder, the left view and right view are not balanced. One possible solution for this limitation would be using the multiple mirrors to make the camera look like it's in the middle of the phone.

Future VCR architecture

We will continue this project, and the VCR architecture that we pursue is shown in figure 6. Basically the architecture looks similar but we made the algorithm efficiently and applicably. Even though we implemented hand detection code, we could not use this code for the demonstration since the accuracy was not good enough. We will train our model by ourselves to get enough hand detection accuracy. Using good hand detection model we will divide the big one task into small multiple tasks and assign to each of server. After that we will use hand landmark and depth estimation. Since we implemented only hand landmark until now, we cannot use it in the VR environment. Because to use in the VR environment we need z coordinates but we did not estimate it. We will research about depth estimation from image and apply it on this algorithm. Finally it will project the coordinates on the VR environment and send the coordinates of each landmark back to the phone so that users can interact with objects in VR environment.

Another potential future work revolves around the use of image-partitioning and off-loading, but rather for capturing vr scenes themselves. VR Streaming is becoming more popular with the increasing popularity in mobile devices to watch 360 videos. However, due to processing and network limitations these experiences are left to very low resolutions. Our solution could partition different images of super-resolution quality using powerful cameras and compute their stitching in real-time by offloading to servers.

6 CONCLUSION

There are many VR applications nowadays but they need additional controllers, sensors, or devices to use them. We have started this research with the thought that we probably are able to make VR application using only a mobile phone. Not to use additional sensors, and controllers, we needed more computation which need to be done to interact in VR environment. So we tried to implement off-loading and multi-server computation for handling this heavy computation. Until now, our VCR consists of streaming from the phone to server and ,in server, detecting, skeletonizing and projection. In streaming we streamed the real-time image flow from the phone to 2 servers and in each server processed each hand. And each server detected hands in an image, skeletonized hands and projected on the VR environment.

While we were implementing it, we noticed some limitations. One of the limitations we found is the problem of the amount of light. Since our application uses the camera, when the surrounding environment was too dark, it could not detect the hands very well. The solution we are thinking about currently is using TOF sensor in dark environment.

Even if we noticed some limitations, we will not stop here. In near future, we will increase the accuracy of detection and add depth estimation function in our application so that it can really interact with objects in VR environment. And we will try to apply it to VR streaming as well.

what we have learned

We really learned how the research is different with the studying or business. At first time when professor talked about this project, we were focused on the useful idea which is more like business idea. And after some meetings with TA, we noticed that what we really need to do is novel technology which is real research. Since we have not concerned about the research idea, it was difficult for us to think in that way. In our research, we tried to develop new technology like multi-server computation but we guess the depth of our research is not enough. In the projects we will continue, we will try to approach research in more various ways, including image merging method of VR streaming.

Since we know we barely have a chance to experience real research, this coarse was a great opportunity to really do it by getting our hand dirty. We appreciate all participants of this coarse including professor, TA and the active students for making amazing coarse together.