A Novel Hyper-cast Approach to Enable Cloud-based Virtual Classroom

Xueshi Hou, Yao Lu, Sujit Dey
Mobile Systems Design Lab, Dept. of Electrical and Computer Engineering
University of California, San Diego
{*x7hou, luyao*}*@ucsd.edu, dey@ece.ucsd.edu*

***Abstract*—In this paper, we explore the possibility of enabling cloud-based virtual space applications providing the advantages of computational scalability and access from any end device, including future light-weight wireless head mounted devices (HMD). In particular, we investigate a virtual classroom application in which the classroom including teacher, students and activities are rendered on the cloud, with each student view captured and streamed to students' end devices. A key challenge is the high bitrate needed to stream all the student views, leading to high cloud cost and wireless network bandwidth. By identifying that multiple student views may share common pixels, we propose a novel hyper-cast approach where common pixels can be transmitted by a single broadcast stream while the residual pixels for individual student views can be transmitted by unicast streams. We formulate the problem of minimizing the total bitrate needed to transmit the student views using hyper-casting and describe our approach. Simulation results show that the hyper-cast approach can significantly reduce total bitrate needed, compared to traditional cloud-based approach of transmitting all the views as individual unicast streams, hence addressing the challenges of cloud cost and network bandwidth.**

***Keywords*-Cloud-based Virtual Space; Virtual Classroom; Hyper-cast;**

I. Introduction

In recent years, Virtual Reality (VR) applications have become popular. New head mounted devices (HMD) together with 3D display technology are unlocking new applications not only in gaming, but also in education, medicine, travel, training and entertainment. However, since VR applications have high computational requirement, current HMDs are either tethered to a PC (like Oculus [1]) or use a smartphone (like Samsung Gear VR [2]) making the latter heavy to wear and hence not portable. In order to enable a truly portable and mobile VR experience, we propose performing the VR rendering on the cloud, and encoding and transmitting the rendered views as video streams over the wireless network to HMDs. In this case, the HMDs will only need video decoding ability, leading to the design of lightweight wireless HMDs, not tethered to PCs or smartphones.

In this paper, we explore enabling a particular virtual reality application, virtual classroom, in which a teacher and students from different geographic locations can participate and communicate in the same classroom session, rendered as avatars in the same virtual classroom. A key challenge in enabling cloud-based virtual classroom is the high bitrate needed to stream all the individual student views rendered on the cloud which have to transmitted to the student HMDs, leading to high cloud cost and wireless network bandwidth. Hence, we address the problem of reducing the sum of bitrates of the cloud-rendered student video streams, without compromising video quality, the latter particularly important for HMDs.

Approaches like asymmetric video encoding [3][4] can reduce overall bitrate needed for all users by encoding left and right views of each user with different quality. However, by potentially compromising video quality, they cannot be used for HMDs. Other approaches like [5] have studied multiple players gaming situations, taking advantage of peer-to-peer sharing between multiple players in the same game scene. However, their analysis of correlation of different user views is limited because they proposed the correlation model only for third-person games, where players watch the entire game scene in a bird-view. Consequently, such methods cannot apply to virtual space applications like virtual classroom.

As opposed to general cloud-based streaming applications where each users view is unique, in virtual classroom, because the students sit close to each other mostly viewing the teacher, the multiple user views can contain large parts very similar to each other. In Figure 1, view A and view B are captured from two virtual cameras corresponding to the views of two students in the virtual classroom world. We can observe that view A and view B share a large portion of common pixels. Taking advantage of this observation, in this paper, we propose a novel hyper-cast system design in which not every pixel of each students view rendered on the cloud needs to be encoded and streamed from the cloud to the student device. Instead, we will transmit the common pixels (*common view*) using broadcast, and transmit the rest of the pixels (we call residual pixels or *residual views*) for each individual user using unicast.

The rest of the paper is structured as follows. In Section II, we provide an overview of the virtual classroom application and our proposed hybrid-cast approach. In Section III, we describe the methodology for common view extraction, and provide the problem statement and our solution. In Section IV, we describe our experimental setup and results. Finally, we conclude in Section V.

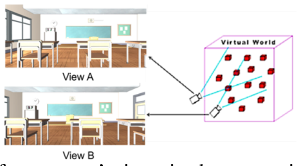

Figure 1. Different users' views in the same virtual classroom.

II. Overview

In this section, we provide an overview of virtual classroom and our proposed hybrid-cast approach. Figure 2 illustrates the virtual classroom application, which we have developed using Unity [6] and Oculus [1]. The classroom consists of several students, each of which has a unique *view* of the classroom.

Figure 2. Virtual classroom.

Figure 3 illustrates the architecture of the proposed hyper-cast approach. On the cloud, graphics are rendered in response to commands from the user. Then they need to be captured, encoded and transmitted to the users. We define two kinds of views: *primary view* and *secondary view*. *Primary view* will be selected among a group of views that contains most common parts with other views. Only one view in the group will be selected as a *primary view* and others will be treated as *secondary views*. In specific, we show an example in Figure 4. Figure 4(a) represents a *primary view* and Figure 4(b) represents a *secondary view*. *Secondary view* consists of two parts, *common view* and *residual view*. The *common view* is part of the view that shares the common area between *primary view* and *secondary view*. Figure 4(d) is the *common view* of the *secondary view* and Figure 4(c) is the corresponding part from *primary view*. The calculation of how to extract *common view* from *primary view* will be explained in the next chapter. *Residual view* is defined as the group of pixels where the *common view* is missing from the *secondary view*. And Figure 4(e) is the *residual view*, which can be used to recover *secondary view* (Figure 4(f)) by combining with *common view* (Figure 4(d)).

Thus, as can be seen from Figure 3, only *primary view* and *residual view* need to be transmitted. They will be sent from the cloud through the core network to the gateways of service providers. Next, the *primary view* will be broadcast by cellular base station and the *residual view* will be unicast. Note that besides using cellular, the hyper-cast approach can be implemented using the broadcast and unicast mechanisms available with other access technologies like Wi-Fi and satellite. Finally, on the end devices, all users will receive the *primary view*. The secondary users will also receive their *residual views*. The primary users will decode and display the video directly while secondary users will decode both *primary view* and *residual view*, transform *primary view* to *common view* and combine *common view* with *residual view* to get his *secondary view*.

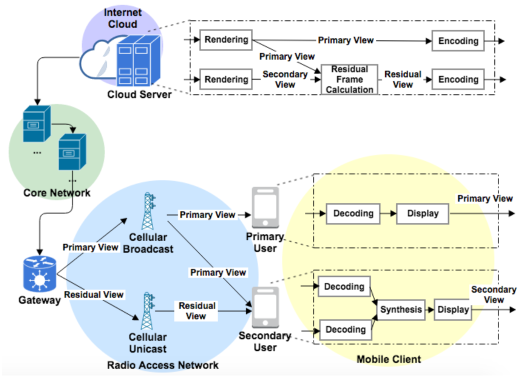

Figure 3. Hyper-cast approach.

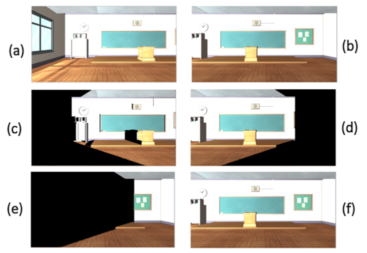

Figure 4. Illustration of *common view* extraction: (a) *primary view* A; (b) *secondary view* B; (c) common pixels from view A; (d) common pixels from view B; (e) *residual view* in B; (f) generated view B from *common view* and *residual view*.

III. Our Approach

In this section, we first describe the methodology for common view extraction and give the problem statement. Next, we present our proposed approach as well as solution to the problem.

A. Common View Extraction

Figure 5 illustrates the *common view* extraction flow. We assume view of user A is a *primary view* and view of user B is a *secondary view*. To extract *common view* between user A and user B, we perform a series of transformation from window space of A to the object space, and further to the window space of B. A *common view* consists of a set of pixels from A that fall within window space of B after transformation. The transformation from object space to window space goes through four steps, as described in [7]. We first transform the object space to eye space using ModelView matrix; we then use projection matrix to transform eye space to clip space; next, clip space is converted to Normalized Device Coordinate (NDC) [8] space using perspective dividing; and we perform viewport transform to the window space. For the transformation from (user A's) windows space to object space, as shown in Figure 5, we perform inverse transformation to each of stated procedures above using inverted transformation matrices.

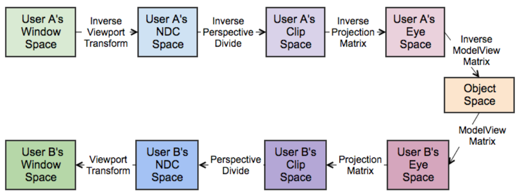

Figure 5. Architecture of *common view* extraction.

B. Problem Statement

Given: All views in the virtual classroom; Dimensions of the virtual classroom, including the width and length.
Find: An optimal strategy to minimize the sum of number of pixels across all views.

$$\begin{aligned}\min \sum_{i\in S_1,S_2} T(i) &= \min(\sum_{i\in S_2} P(i_{residual}) + \sum_{i\in S_1} P(i)) \\ &= \max(\sum_{i\in S_2} P(i_{common}) - \sum_{i\in S_1} P(i)) \\ &= \max(\sum_{i\in S_2} R(i_{common}) - |S_1|) \qquad (1)\end{aligned}$$

Equation 1 demonstrates our objective to minimize the sum of total number of pixels transmitted $T(i)$, across all views i. Note that having a single primary view may lead to many secondary views having little or no common views, and thus large residual views. Therefore, we need to divide the class students into one or more groups, each group having a primary view and zero or more secondary views. S_1 and S_2 represent the *primary view* and *secondary view*, respectively. $P(i_{residual})$ and $P(i_{common})$ represent the residual and common pixels of view i, respectively. $P(i)$ denotes the pixels of view i. We define the common pixel ratio of a *secondary view* i as $R(i_{common}) = \frac{P(i_{common})}{P}$, where P is the number of pixels per frame. $|S_1|$ indicates the number of *primary views*. We note that for each *secondary view*, only one *primary view* is selected to calculate $P(i_{common})$ and $P(i_{residual})$.

C. Proposed Approach

In our approach, in order to assign multiple users to different groups and minimize the sum of bitrate across multiple users, we want to develop a strategy to group the views which have a lot of common pixels together. In order to make the grouping technique fast, we need to avoid the time-consuming process of conducting *common view* extraction between all pairs of different views. Moreover, calculating $R(i_{common})$ between views can be prohibitively expensive. Hence, we develop and use an easy to calculate metric to approximately represent how much is common between two views; we term this metric *common normalized VLength (cnVL)*, which we define next.

$$VLength = xVL + yVL \qquad (2)$$

We define *VLength* as the sum of *xVL* and *yVL*, as shown in Equation 2. In this definition, *xVL* and *yVL* denote the length of the front wall and side wall within the current view respectively. Figure 6 illustrates from the top of the classroom an example of *xVL* and *yVL* with the view of student in seat #9.

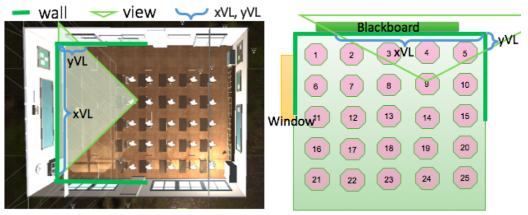

Figure 6. The model of the virtual classroom and camera view when looking from the top. It presents the boundary of the classroom, the seat positions as well as a demonstration of users' view. The lengths corresponding to xVL and yVL are shown.

Table I
FOUR DIFFERENT CASES.

Case	a	b	c	d
Left wall visible	✓			✓
Front wall visible	✓	✓	✓	✓
Right wall visible			✓	✓

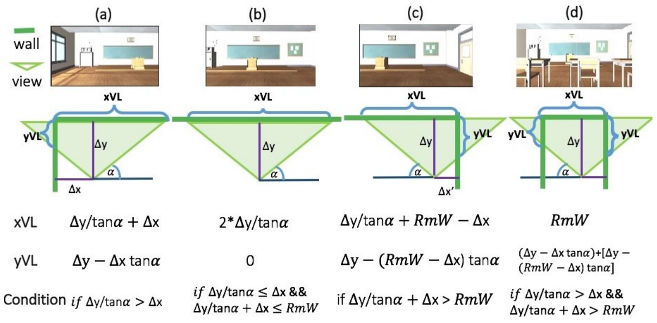

Figure 7. (a)-(d) show four types of relative positions between classroom boundary and view boundary. *RmW* denotes the width of the classroom in x-axis.

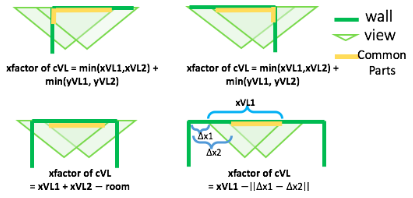
wall
view
Common Parts
xfactor of cVL = min(xVL1,xVL2) + min(yVL1, yVL2)
xfactor of cVL = min(xVL1,xVL2) + min(yVL1, yVL2)
xVL1
Δx1
Δx2
xfactor of cVL = xVL1 + xVL2 − room
xfactor of cVL = xVL1 −||Δx1 − Δx2||

Figure 8. Several typical situations of the xfactor of *cVL*.

To calculate *VLength*, we summarize four difference types of relative positions between classroom boundary and view boundary, as shown in Table I. Figure 7 also describes the methodology to calculate *VLength* given these four types of relative positions. Especially, we note that for case (d), VLength is obtained separately according to *yVL* from the left wall (yVL_{left}) and right wall (yVL_{right}), respectively, as shown in Equation 3.

$$VLength = xVL + yVL_{left} + yVL_{right} \qquad (3)$$

We define the *Common VLength* (cVL) as a metric to evaluate the common ratio between two views p and q. Equation 4 describes cVL, with *xfactor* and *yfactor* defined in Equation 5 and 6 respectively. *xfactor* is defined as the length of the front and side wall within both views, which is shown in Figure 8. Condition 1 indicates that two views are of case (a) and (c), respectively; condition 2 indicates that both views are of case (b); and condition 3 indicates all other case combinations. *yfactor* is defined as the squared ratio of the distance to the front wall for both views.

$$cVL = xfactor \times yfactor \tag{4}$$

where

$$xfactor = \begin{cases} xVL(i) + xVL(j) - room & \text{if condition 1;} \\ xVL(i) - |\Delta x_i - \Delta x_j| & \text{if condition 2;} \\ \min(xVL(i), xVL(j)) + \\ \min(yVL_{left}(i), yVL_{left}(j)) + \\ \min(yVL_{right}(i), yVL_{right}(j)) \\ & \text{otherwise (condition 3);} \end{cases} \tag{5}$$

$$yfactor = \begin{cases} (\Delta y_i / \Delta y_j)^2, & \text{if } \Delta y_i < \Delta y_j \\ (\Delta y_j / \Delta y_i)^2, & \text{if } \Delta y_i \geq \Delta y_j \end{cases} \tag{6}$$

To evaluate our proposed metric, we compare *cVL* (Equation 4) to the common pixel ratio for every pair of views in a virtual classroom with a seat pattern of 2x5 (10 views, 100 pairs of views). Each view has a resolution of 690x400 pixels. Figure 9(a) illustrates the correlation between common *VLength (cVL)* and the common pixel ratio values for the 100 pairs of views, with an overall correlation of 0.8828. To increase the correlation with common pixel ratio, we define *common normalized VLength (cnVL)* as follows:

$$cnVL(p,q) = \frac{cVL(p,q)}{cVL(p,p)} \tag{7}$$

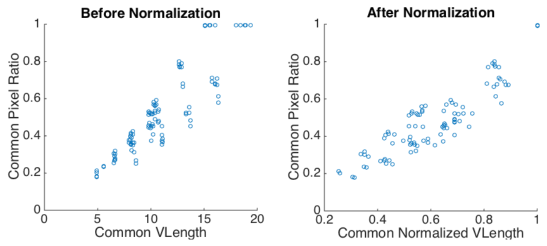

Figure 9. Validation of model parameters, showing the relationship between common pixels and *cVL* as well as *cnVL*.

Figure 9(b) illustrates a higher correlation, 0.9207, between *cnVL* (Equation 7) and the common pixel ratio. The calculation of $cnVL$ values is much simpler and faster compared to obtaining the actual *Common Pixel Ratio* between every two views using graphic rendering. Therefore, due to the high correlation between $cnVL$ and common pixel ratio, we can approximate the common pixel ratio with the metric $cnVL$, which greatly saves runtime. Subsequently, we update our problem formulation (Equation 1) as follows:

$$\min \sum_{i \in S_1, S_2} T(i) = \max(\sum_{i \in S_2} R(i_{common}) - |S_1|)$$
$$\approx max(\sum_{p,q} cnVL(p,q) - n) \tag{8}$$

We implement the grouping algorithm that will consider explicitly every possible grouping choice, considering each view in a group as primary view. With Equation 8, we can do the grouping fast and effectively by using our proposed metric to avoid prohibitively expensive calculation of $R(i_{common})$.

IV. Experimental Results

We have performed experiments to evaluate the performance of our proposed hyper-cast approach on virtual classrooms of different sizes. Our approach is implemented using MATLAB, and runs on a Windows 10 operating system with an Intel Core i7-5930K Haswell-E 6-Core 3.5GHz processor and 32G memory.

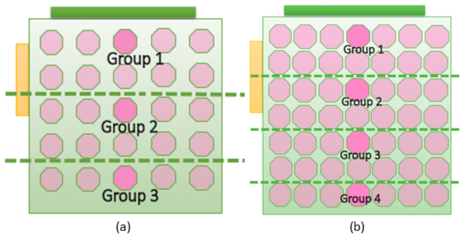

Figure 10. Groups obtained for (a) 5x5, and (b) 7x7 classrooms. Every deep pink circle represents the seat chosen as *primary view* while the light pink circle corresponds to the seats with *secondary views*.

Figure 10 shows the result of applying our proposed approach on virtual classrooms of size 5x5 and 7x7 seat patterns. Our proposed hyper-cast approach reduces the bitrate needed by 44.56% for 5x5 classroom, and 48.4% for 7x7 classroom, compared to the conventional approach of transmitting all the views as individual unicast streams.

V. Conclusion

In this paper, we propose a multi-user hyper-cast approach which can significantly reduce high-quality video encoding bitrate needed across multiple users. We plan to develop more efficient ways of grouping students, and also explore other virtual space applications in the future.

References

[1] [Online]. Available: https://www.oculus.com/

[2] [Online]. Available: http://www.samsung.com/global/galaxy/gear-vr/

[3] G. Saygili, C. G. Cihat and A. M. Tekalp, "Evaluation of asymmetric stereo video coding and rate scaling for adaptive 3D video streaming," *IEEE Transactions on Broadcasting*, vol. 57, no. 2, pp. 593–601, June 2011.

[4] S. Valizadeh, A. Maryam and N. Panos, "Bitrate reduction in asymmetric stereoscopic video with low-pass filtered slices", in *Proc. of 2012 IEEE International Conference on Consumer Electronics (ICCE)* (Las Vegas, NV), Jan. 13–Jan. 16, 2012, pp. 170–171.

[5] W. Cai and V. C. Leung, "Multiplayer cloud gaming system with cooperative video sharing," in *Cloud Computing Technology and Science (CloudCom), IEEE 4th Intl. Conf.*(Taipei), Dec. 3–6, 2012, pp. 640–645.

[6] [Online]. Available: https://unity3d.com/

[7] [Online]. Available: http://www.songho.ca/opengl/gl_trans form.html/

[8] C. Everitt, "Projective texture mapping," in *White paper*, NVidia Corporation, 2001, 4.