Dynamic Adaptive Streaming for Augmented Reality Applications

Stefano Petrangeli, Gwendal Simon, Haoliang Wang, Vishy Swaminathan

Dynamic Adaptive Streaming for Augmented Reality Applications

Stefano Petrangeli
Adobe Research
petrange@adobe.com

Gwendal Simon*
IMT Atlantique
gwendal.simon@imt-atlantique.fr

Haoliang Wang
Adobe Research
hawang@adobe.com

Vishy Swaminathan
Adobe Research
vishy@adobe.com

***Abstract*—Augmented Reality (AR) superimposes digital content on top of the real world, to enhance it and provide a new generation of media experiences. To provide a realistic AR experience, objects in the scene should be delivered with both high photorealism and low latency. Current AR experiences are mostly delivered with a download-and-play strategy, where the whole scene is considered a monolithic entity for delivery. This approach results in high start-up latencies and therefore a poor user experience. A similar problem in the video domain has already been tackled with the HTTP Adaptive Streaming (HAS) principle, where the video is split into segments, and a rate adaptation heuristic dynamically adapts the video quality based on the available network resources. In this paper, we apply the adaptive streaming principle from the video to the AR domain, and propose a streaming framework for AR applications. In our proposed framework, the AR objects are available at different Levels-Of-Detail (LODs) and can be streamed independently from each other. An LOD adaptation heuristic is in charge of dynamically deciding what object should be fetched from the server and at what LOD level. Our proposed heuristic prioritizes content that is more likely to be viewed by the user and selects the best LOD to maximize the object's perceived visual quality. Moreover, the adaptation takes into account the available bandwidth resources to ensure a timely delivery of the AR objects. Experiments carried out over the Internet using an AR streaming prototype developed on an iOS device allow us to show the gains brought by the proposed framework. Particularly, our approach can decrease start-up latency up to 90% with respect to a download-and-play baseline, and decrease the amount of data needed to deliver the AR experience up to 79%, without sacrificing on the visual quality of the AR objects.**

***Index Terms*—Augmented Reality; HTTP Adaptive Streaming; LOD Adaptation Heuristic; AR Streaming**

I. Introduction

Augmented Reality (AR) represents the blending of digital content with the real world. The release of software development kits such as ARKit and ARCore [1,2] has enabled more than 500 million devices to feature AR capabilities as of 2019 [3]. As headsets become cheaper and more widespread, business studies forecast an increased demand for AR content in the near future [3].

To provide an immersive experience that resembles (as closely as possible) our interaction with the real world, AR applications should ideally feature both the highest possible visual quality and the lowest possible start-up latency. A visual quality up to *photorealism* is extremely bandwidth-consuming, with data sizes ranging from tens[1] to hundreds[2] of megabytes, for a single AR object. To put things in perspective, this roughly corresponds to the amount of data needed to stream 20 minutes of Netflix content at standard resolution [4]. Current AR experiences are mostly delivered with a simple *download-and-play* strategy, which results in high start-up latency. High latency is well known for having a negative impact on user experience for both web-browsing and video streaming applications. Even though such an impact has yet to be investigated in AR, it is plausible to anticipate a similar negative influence in this domain as well.

Fig. 1: An AR object at two different Levels-Of-Detail (LOD); on the left, a high level; on the right, a low level.

A similar challenge has been successfully tackled for delivering videos over the Internet, which has evolved from simple *download-and-play* to *streaming*. Particularly, HTTP Adaptive Streaming (HAS) has emerged as one of the main principles to deliver videos over the best-effort Internet. In adaptive video streaming, the video quality can dynamically change to accommodate for the varying bandwidth conditions of the network, which minimizes re-buffering events. The client is equipped with a rate adaptation heuristic, which decides the optimal quality to download based on information such as the available bandwidth, the status of the video buffer, and the available video bitrates, among others.

Our contributions. We address the latency issues that currently affect AR applications by leveraging similar adaptation principles as those used in the video domain. We deal with an AR scene composed of multiple AR objects, which are available at the server at multiple Levels-Of-Detail (LODs),

*Worked performed while the author was in Adobe Research

[1]https://tinyurl.com/y29nwt6w
[2]https://tinyurl.com/y5kj5dx7

each of them providing a representation of the AR object at different resolutions[3] (see Figure 1). Instead of downloading the whole AR scene before displaying it, we propose the client to stream the scene by fetching each object independently. We design an *LOD adaptation heuristic*, which dynamically decides which object and LOD level should be downloaded to minimize the user waiting time, while still guaranteeing a good visual quality. Particularly, the goal of the heuristic is to prioritize the delivery of the objects that are the most likely to be viewed by the user, and to select the best LOD level with respect to the available bandwidth and the relative position of the object. Intuitively, objects further from the viewer can be retrieved at a lower LOD, since the viewer cannot appreciate fine-grained details. As in regular adaptive video streaming, the objects are retrieved from the server using the HTTP protocol.

The main contributions of this article are therefore twofold. First, we present a client-based framework for the low-latency streaming of AR scenes. The streaming client periodically computes the *priority* of each object in the scene, based on the size and the position of the object. Moreover, for each object LOD, a *utility* value is computed, representing the contribution of the analyzed LOD to the visual quality of the scene. Second, we present an LOD adaptation heuristic for AR scenes, whose goal is to schedule the download of the different objects in the scene and decide on the particular LOD level to request. This decision is based on the available bandwidth estimated at the client and the aforementioned priority and utility values. Detailed experimental results collected using a streaming prototype developed on an iOS device showcase the gains of the proposed system, which is able to provide a low-latency delivery of the AR scene without sacrificing visual quality.

The remainder of this paper is organized as follows. Section II presents related work on delivery optimizations for 3D and immersive media. Section III details the proposed streaming framework and LOD adaptation algorithm, while Section IV reports the results obtained using the developed streaming prototype. The main conclusions and an outline for future improvements and research opportunities in the AR streaming domain are described in Section V.

II. Related Work

A large body of literature addresses the problem of streaming large-scale 3D/AR environments. One of the main solutions is based on a view-dependent approach; a representative example is provided by popular services such as Google Earth and Google Maps. Most of the works in this area have focused on streaming for large-terrain and 3D remote walkthrough. Rusinkiewicz et al. propose the QSplat rendering system for streaming dense polygon meshes [5]. The authors propose an optimized file format structure, which allows progressive streaming and refinement of object LOD. The object is divided into a hierarchical set of nodes, each node providing a more fine-grained representation of the object. A prioritization mechanism aims at deciding what node should be retrieved, based on the distance of the user from the object. A hierarchical representation for 3D objects has also been proposed by Hoppe et al. [6], which enables the reduction of visual artifacts (the so-called *popping* effect) during LOD changes. This is achieved by anticipating the viewer position in the 3D environment and adjusting the visual quality of the objects accordingly. Several other works have focused on novel adaptation and prefetching strategies for 3D walkthrough systems. In Meng at al. [7], the LOD refinement process is driven by the gaze position of the user. Particularly, the portion of the object visualized in the fovea region of the human eye is fetched at a higher quality. This mechanism prioritizes portions of the object whose quality can be better appreciated by the viewer. To optimize the streaming of 3D terrains data, 3D tiles can also be employed [8]. An importance score is computed for each tile based on its distance from the viewpoint. A tile management algorithm maintains the largest squared area of content around the user, subject to the available memory on the device. A priority mechanism that decides the quality of 3D objects in remote walkthrough applications is proposed by Chim et al. and Zach et al. [9,10]. The priority of an object is computed based on its euclidean and angular distance from the viewer. To limit the amount of memory used by the client, a cache is employed whose replacement strategy strives to maintain content that has the highest probability of being viewed. Zhou et al. propose a prefetching system that leverages a prediction algorithm to anticipate the users' movements in the 3D environment [11]. The prediction algorithm is generated by exploiting the similarities between users' access patterns for the same scene. The prefetching mechanism can also be based on the position of the object in the scene [12]. The authors divide the scene in different zones, with the zone in front of the viewer associated to the highest priority. Based on the zone they belong to, objects are placed in different request queues, with objects in high priority queues being prefetched before the others. These works usually rely on real-time streaming protocols to ensure a timely delivery of the 3D content, which are less scalable and more expensive that the streaming systems based on HTTP adaptive streaming. These techniques have been successfully applied to the streaming of immersive video content, as 360-degree and volumetric videos [13,14]. A tiling mechanism is employed to divide the video content, being it 2D or 3D, into smaller regions, which can be individually streamed. This mechanism allocates more quality to content that is more likely to be watched by the user and, therefore, saves bandwidth. Particularly, Chou et al. introduce a utility metric to drive the tile adaptation algorithm with respect to the visual quality contribution of a particular quality level and the viewing probability of a tile [13].

The literature presented in this section shares similar objectives and principles with the framework proposed in this paper. Particularly, we propose to expand the successful techniques adopted in 3D remote walkthrough systems and in adaptive

[3]In this paper, we extend the LOD terminology to refer to both the geometry and the texture component of an AR object.

video streaming domain to the context of AR streaming. In our framework, the LOD heuristic leverages an adaptation strategy formulated as a rate adaptation heuristic from HAS, while being based on metrics as the priority of the objects in the scene and the visual quality contribution of the LODs, which are typical from 3D scene delivery systems. By combining these different techniques we showcase how it is possible to enable low-latency and yet high-quality AR experiences, using a prototype implemented on a mobile device.

III. AR Streaming Framework

We detail in this section the proposed framework for the low-latency streaming of AR experiences. Rather than considering the whole AR scene as a monolithic entity to deliver, we consider each object as an atomic component, which can be streamed independently from the others. Using this approach, it is possible to lower the scene start-up latency by first fetching content that is most likely to be viewed by the user. Moreover, adapting the LOD level of each object provides an additional lever to prioritize more relevant content and guarantee a timely delivery despite varying bandwidth conditions. This task is accomplished by an LOD adaptation heuristic, which decides the object and LOD to request from the server. This adaptation is performed every time an object at the chosen LOD has been download, similar to the adaptation carried out at video segment boundaries in HAS. The inputs of the heuristic are threefold:

- The available bandwidth estimated by the client during the download of an object;
- A priority value indicating the importance of the object in the scene, which depends on factors such as the distance from the viewer, the position in the scene, and the size of the object;
- The utility of a specific LOD level of a given object, to account for the fact that increasing the quality of an object might result in marginal gains in terms of visual quality, especially when the object is far from the viewer.

The design choice to keep the adaptation heuristic agnostic of the underlying AR scene allows potential re-use of the existing heuristics developed for the video domain in an AR context. The remainder of this section details the object priority (Section III-A), the LOD utility computation module (Section III-B), and the LOD adaptation heuristic (Section III-C).

A. Object Priority

An AR scene can be composed of a multitude of different objects, each positioned at different locations and with different characteristics in terms of physical size and appearance. Given this information on the scene structure and the position in six degrees-of-freedom of the user, which unequivocally defines the Field-Of-View (FOV), the first module of the proposed framework determines which object is more important for the viewer. This information is crucial to correctly prioritize relevant objects and to allocate more bandwidth to objects that are more likely to be viewed by the user. Particularly, the priority value of object i at time t (omitted for clarity) is given in the following equation, which is an extension of the priority calculation from Chim et al. [9]:

$$P_i = (1 + A_i) \times \tilde{A}_i \times e^{-K\theta_i} \tag{1}$$

A_i represents the area of the user's FOV occupied by the object i, equal to 1 when the object occupies the entire FOV. This value depends both on the distance and the physical size of the object and is computed as the area of the convex hull of the object bounding box, as also proposed by van der Hooft et al. [15]. Despite being an approximation of the real percentage of FOV occupied by the object, A_i nevertheless indicates which objects are bigger and/or closer to the user, and therefore sets priority on objects with regards to their impact on user's visual perception. $\tilde{A}_i$ is the *potential* area of object i in the FOV, i.e., the area that would result if the user would turn and point directly towards i. This term is introduced to take into account that rotational changes of the FOV can be much faster than translational ones. The viewer can indeed explore a large portion of the (closer) scene by simply rotating her head, which may happen in less than a second [16]. $\tilde{A}_i$ increases priority for objects that could be relevant for the user in the near future, despite being currently outside the FOV. Finally, the third term of Equation (1) takes into account the angular distance between the user's viewing direction and the center of the object's bounding box, represented by θ_i. Intuitively, the more a user would need to rotate to have a specific object in FOV, the lesser the importance of the object, as its probability of being in view diminishes. In the remainder of the paper, we set $K = 1.309$ (i.e., 75°).

It is worth mentioning that the priority value for each object is updated periodically by the client as the user moves and interacts with the AR scene, and is fed to the adaptation heuristic together with the LOD utility values.

B. LOD Utility

In our AR streaming system, each object is available at the server at multiple LODs. Depending on the complexity of the object and distance to the viewer, a higher LOD might only marginally contribute to an increase in visual quality with respect to a lower one. This well-known concept in computer graphics has been successfully exploited for the remote rendering of 3D worlds and video games. We propose in this paper to extend this concept to the context of AR streaming. Particularly, we introduce an LOD utility value, which represents the quality improvement the viewer would gain by using a specific LOD, weighted by the time necessary to download it. While the priority value presented in the previous section is an *inter-object* metric, which indicates the relative importance between objects in the scene, the utility value is an *intra-object* metric indicating the utility of the available LODs for a specific object.

A widely used metric in computer graphics to define the objective visual contribution of a particular LOD is the Screen-Space Error (SSE) [17], which is defined as the difference

in pixels ρ between rendering on screen a lower LOD rather than a higher one. The SSE ρ depends on factors such as the distance to the object, the resolution and field-of-view of the screen, and the LOD geometric error, which expresses the absolute distance between the vertices of the original high-quality object and its LOD representation (obtained through a process of decimation). Lower values of the SSE indicate higher-quality LODs. In this paper, we compute ρ as proposed by Cozzi et al. [17]. We first compute the quality contribution q_{ij} of each LOD j of object i at time t (omitted for clarity), as in Equation (2):

$$q_{ij} = \begin{cases} [1 + (\rho_{ij} - \rho^*)]^{-1} & \text{if} \quad \rho_{ij} \geq \rho^* \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

where ρ_{ij} is the SSE of the analyzed LOD j, and ρ^* is a configurable target SSE value, which indicates the minimum screen space error below which any LOD refinement would not be perceivable by the user. The quality contribution decreases when the LOD improves, until it reaches a value where the SSE of the analyzed LOD is equal or smaller than the target SSE. Next, the utility value is computed as in Equation (3):

$$U_{ij} = \begin{cases} 0 & \text{if} \quad (\rho_i \leq \rho^* \quad \text{or} \quad \rho_i \leq \rho_{ij}) \\ q_{ij} / \left(1 + \delta_{ij}\right) & \text{otherwise} \end{cases} \tag{3}$$

where ρ_i is the SSE of the LOD currently in view for object i, and δ_{ij} is the estimated downloading time in seconds, computed as the ratio between the size of the LOD and the estimated available bandwidth. If the LOD in view already satisfies the target SSE, it is not necessary to further upgrade the quality of the object at the current time, and the utility value of all LODs is set to zero. We also set the utility value of LOD j to zero if the SSE of the LOD in view is smaller than that of LOD j, i.e., downloading LOD j would reduce the object visual quality. Otherwise, the quality contribution q_{ij} is weighted by the downloading time to fetch the specific LOD, which incorporates the trade-off between increasing quality and ensuring a timely delivery.

It is worth stressing that all the values in Equation (2) and Equation (3), at the exception of ρ^*, are dynamic. Both ρ_{ij} and ρ_i can change due to user movements in the AR scene, while δ_{ij} depends on the available network resources.

C. LOD Adaptation Heuristic

In the adaptive streaming context, several heuristics have been proposed to address the dynamic video adaptation problem [18]. The fundamental rationale behind these heuristics is to balance the trade-off between increasing video quality and avoiding re-buffering events. Particularly, the video buffer status at the time of the request provides an explicit deadline for the download of the next segment. The goal of the heuristic is to request the segment at the highest possible video quality, such that the probability of meeting the next and future deadlines is maximized. Different from video, such a temporal aspect is missing in the AR domain, as an AR scene is composed of objects that, albeit potentially animated, are static. Despite this, a virtual delivery deadline can be introduced in the AR domain as well. Indeed, the decision taken at a certain instant on what object and LOD to request is only going to be relevant for a limited amount of time, because the user can freely move in the AR scene. We define this quantity as the *minimum reaction time to changes* in the AR scene. Any LOD delivered after this deadline has high chances to not be optimal anymore because the user likely changed her position in the scene. From this perspective, the goal of an LOD adaptation heuristic is to request one or multiple objects at the highest possible LODs, such that the probability of downloading the objects within the reaction time deadline is maximized, which is indeed similar to the optimization goal of standard rate adaptation heuristic for video.

Algorithm 1: Proposed LOD adaptation heuristic

Inputs : P_i, objects' priority values
U_{ij}, objects' LOD utility values
Output: Q, queue of LODs requests

$L \leftarrow$ LODSelection(P, U) (Equation (4))
if L *is empty* **then**
  $Q \leftarrow$ LOD with $\max P_i U_{ij}$
else
  $Q \leftarrow L.sort()$

In light of the above, we formulate the LOD adaptation problem as a multi-choice knapsack problem, presented in Equation (4):

$$\begin{aligned} \max_{\alpha} \sum_i \sum_j \alpha_{ij} V_{ij} \quad \text{with} \quad V_{ij} = P_i U_{ij} \\ \sum_j \alpha_{ij} \leq 1 \quad \forall i \\ \sum_i \sum_j \alpha_{ij} \delta_{ij} \leq \Delta^* \\ \alpha_{ij} \in \{0; 1\} \quad \forall i, j \end{aligned} \tag{4}$$

where α_{ij} represents the decision variable of the problem, equal to 1 when LOD j of object i is selected for download, and 0 otherwise. The objective is to maximize the cumulative value of the LODs to request, where the value is given by the product of the priority and utility values. The first constraint restricts the selection of only one LOD per object to download. The second constraint indicates that the download of the selected LODs should be completed in Δ^* seconds, with Δ^* representing the aforementioned minimum reaction time to changes, which has a similar role as the buffer level in adaptive video streaming.

The complete LOD adaptation heuristic is presented in Algorithm 1. We first obtain a list of LODs to request based on the formulation in Equation (4). The list L can be empty when

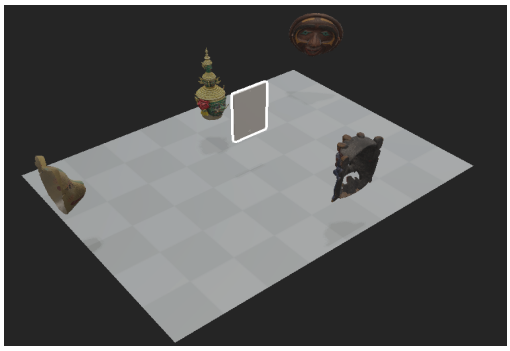

Fig. 2: The AR scene used for the experiments is composed of four different objects.

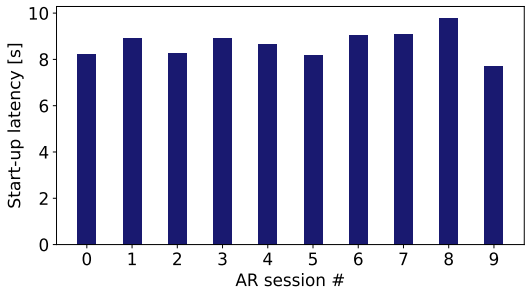

Fig. 3: The start-up latency for the download-and-play approach ranges from 7.70 s to 9.79 s.

the optimization problem has no solution. This can happen if no LODs can be retrieved within Δ^* seconds (e.g, because the estimated available bandwidth is low). In this corner case, we simply request the LOD of the object with the highest value V. Otherwise, we sort the list of selected LODs based on decreasing value of V. Given the produced sequence of LOD requests Q, the AR client can request the first element in the queue only, or use any existing prefetching mechanism as HTTP/2 push or multiplexing to request multiple LODs at the same time [14]. In the remainder of the paper, we set ρ^* and Δ^* to 1 pixel and 2 seconds, respectively.

IV. Performance Evaluation

A. Experimental Setup

We present in this section the experimental setup that is used to assess the performance of the proposed AR streaming framework, which was implemented on an iOS device using ARKit.

We designed a simple AR scene composed of four different objects located at different positions with respect to the initial position of the viewer (Figure 2). Each object is available at four different LODs. A full description of the objects and the scene is provided in Table I. The geometric error of lower LODs is computed with respect to the highest available LOD, which therefore has geometric error equal to 0. The width, height, and depth of the object's bounding box are used as proxies for the object's dimensions. All position coordinates are expressed with respect to a right hand coordinate system whose positive z-axis points towards the viewer. As an example, Object #3 is positioned three meters on the left with respect to the viewer and is rotated by 90° around the y-axis to face the viewer, at the start of the AR session. It is worth noting that we do not consider any semantic relationship among the objects in the scene (e.g., two objects interacting among each others). This simplification makes the gains and drawbacks of the proposed solution easier to get. We discuss in Section V how animations, triggers, and interactions between objects can play an important role in the context of AR streaming.

The web server hosting the AR content is located on the US east coast (North Virginia), while the streaming client is located on the US west coast (North California), and is connected to the Internet through a corporate Wi-Fi network. This configuration allowed us to test the performance of the proposed approach under realistic conditions.

We compare the proposed streaming framework with a download-and-play scenario, where the whole scene is downloaded at the highest LOD before being displayed to the user. Even though this approach results in a high start-up latency, it provides the scene with the highest level of photorealism. As far as the performance metrics are concerned, we compute:

- The *scene start-up latency*, calculated as the time elapsed between the start of the AR session and the moment the first object is displayed on screen;
- The *screen-space error* of the object in view, as a measure of visual quality;
- The *missing object events*, which occur when a user misses an opportunity to watch an object because it has not been delivered yet, and is therefore not present in the scene;
- The *amount of data* used to deliver the AR scene.

For the streaming scenario, ten user traces have been collected using the developed prototype to ensure a realistic interaction with the AR scene.

B. Results

We first report the results for the start-up latency for both the download-and-play (Figure 3) and the proposed streaming approach (Figure 4, top graph). On average, across experiments, the start-up latency for the download approach is (8.68 ± 0.56) s, which decreases to only (0.78 ± 0.03) s in the streaming case. This behavior is expected, as our streaming client starts fetching the content that is most likely to be watched by the user, and upgrades the quality of the objects over time. The improvement in start-up latency comes with minor negative effects on the overall user experience. Particularly, we report the missing object events time (middle graph in Figure 4), i.e., the amount of time the user is not able to experience an object in the scene because it has yet to be delivered. This value should be as close as possible to zero to ensure the user is not missing any important part of the AR scene. Next, we report the amount of time an object in view is watched at an acceptable visual quality (bottom graph in Figure 4), i.e., when the SSE of the object in view is

	Object #1 [19]	Object #2 [20]	Object #3 [21]	Object #4 [22]
Geometric error	0.00119 - 0	0.00153 - 0	0.001179 - 0	0.002271 - 0
Size [MBytes]	0.5 - 11	0.3 - 10	0.2 - 9	0.3 - 8
Dimension [m]	(0.35, 0.84, 0.34)	(1.34, 1.36, 0.67)	(1.49, 2.34, 1.10)	(1.08, 1.35, 0.48)
Position [m]	(0, 0, -1)	(2.0, -0.5, -2.0)	(-3.0, 0.0, 0.0)	(0.0, 0.0, 2.0)
Rotation [deg]	(0, 0, 0)	(0, -45, 0)	(0, 90, 0)	(0, 180, 0)

TABLE I: The characteristics of the four objects used for the experiments.

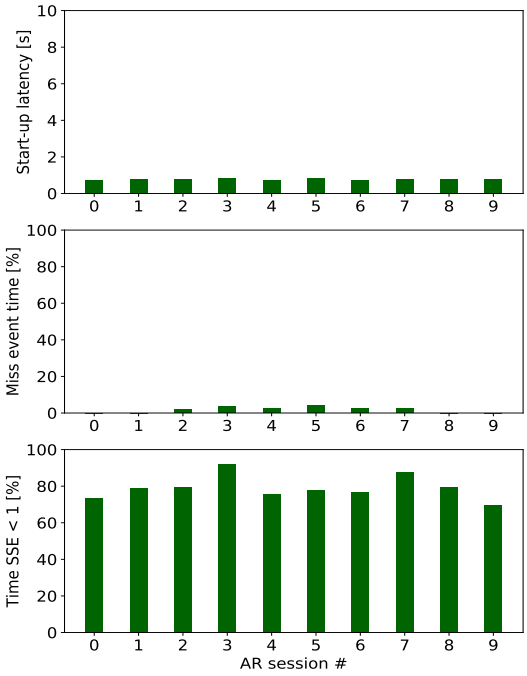

Fig. 4: Our streaming approach reduces start-up latency to 0.78 s on average (top graph). This improvement only marginally affects the overall user experience, in terms of missing object events (middle graph) and time spent watching an object at sub-optimal quality (bottom graph).

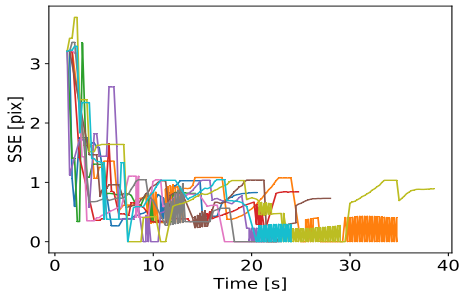

Fig. 5: Using the proposed LOD adaptation heuristic, the SSE of the object(s) in view quickly decreases below the target value.

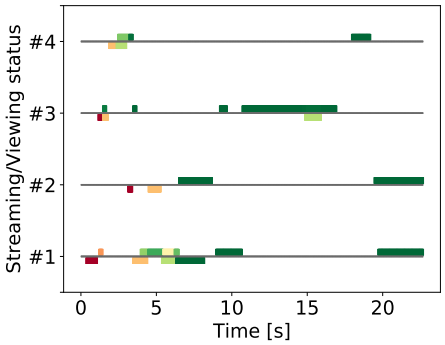

Fig. 6: For each object, the graph depicts the evolution of the SSE in view (bars above gray line) and LOD streaming events (bars below gray line). Green colors represent lower SSE values in view and higher quality LODs streamed from the server, respectively.

below the target SSE. As discussed in Section III-C, we use the SSE metric to quantify the visual quality of an AR object, and chose the target SSE equal to 1 pixel. Both metrics are expressed as a percentage of the total AR session duration. We report the results for the streaming scenario only, as in the download-and-play case all the objects are available in the scene (once they have been retrieved by the server), and at the highest possible LOD. In the streaming case, missing object events occur for less than 4 % of the AR session viewing time in the worst case scenario, while the time spent at an optimal visual quality level is (79 ± 6) % of the total session time.

Figure 5 reports the evolution over time of the SSE, for all the AR sessions. The SSE of the object in view tends to decrease quickly thanks to the adaptation proposed in Section III-C, and to fall below the target SSE threshold. This comes with a consistent advantage in terms of start-up latency, as previously discussed. The proposed client tends to retrieve lower LODs first to guarantee a timely start-up of the experience and avoid missing object events, and to refine the quality of the objects later to improve user experience. Figure 6 provides an example of such adaptation, for a selected AR session. For each object, bars above the gray line indicate the evolution of the SSE over time. A dark green color corresponds to an SSE equal or below the target SSE of 1 pixel. Bars below the gray line indicate the streaming of a particular LOD level, with the length of the bar indicating the amount of time needed to fetch the LOD from the server. A red color indicates a low quality level, while a dark green color represents a high quality LOD streamed from the server. The streaming session starts by downloading Object #1, which is positioned in front of the viewer at the beginning of the session (see Figure 2 and Table I), at the lowest available LOD. Object #1 is viewed for a short period of time before the user moves to Objects #3 and

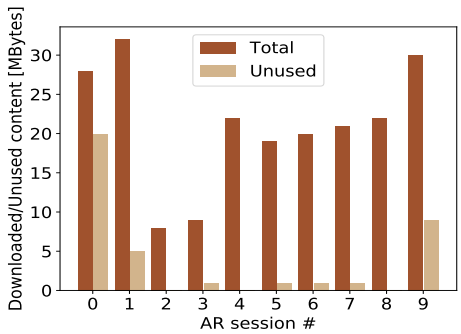

Fig. 7: Streaming the AR scene can reduce the amount of data retrieved from the server by 43 % on average, with respect to a download-and-play approach.

#4. The adaptation heuristic follows the movements of the user by downloading the lowest and second lowest LOD for Objects #3 and #4. This actions prevents a missing object event to happen, while still providing the objects at good visual quality, as the green bars indicate. Next, the user shifts focus back to Object #1, by getting closer to it around second 5 (indicated by the SSE bar going from green to yellow, which indicates a decrease in quality). This triggers the download of the highest available LOD for Object #1. After this event, the client does not request any additional content, as the LODs of the objects in the scene are high enough to guarantee an acceptable visual quality to the user. An additional LOD increase is triggered at second 15, once the user gets closer to Object #3.

To conclude this section, we report the amount of data used by the streaming client during the AR session (Figure 7). In the download-and-play approach, all the objects in the scene are downloaded at the highest LOD level, for a total of 38 MBytes of data downloaded from the server. Streaming the AR scene can consistently reduce this amount, from a minimum of 14 % to a maximum of 79 %. The utility value proposed in Section III-B effectively captures when the quality of an object would need to be increased. Therefore, a higher LOD is requested only if it would result in an actual increase in perceived quality. Sessions with fewer data fetched from the server correspond to cases where the user did not get close enough to an object to trigger an LOD increase. We also measure the amount of unused data, e.g., LODs that have been downloaded but never displayed to the user. This can happen because the user can quickly switch her focus to a different object, making the old LOD request not relevant anymore. As we discuss in Section V, FOV prediction algorithms can be used to reduce this behavior and also to ensure the user can always visualize the AR content at the highest possible visual quality, by anticipating the user's movements.

V. Conclusion

A. Summary of our Contributions

We presented in this paper a novel framework for the low-latency delivery of AR applications, where we propose to apply the adaptation principles that are successfully used in the adaptive video streaming domain to the AR delivery domain. The main component of this framework is an LOD adaptation heuristic with the goal of dynamically deciding which AR object to request and at which LOD level. This decision is based on the priority (i.e., importance) of the object in the AR scene and the utility of the object's LODs. Both metrics are computed based on the relative position of the object with respect to the viewer and the perceived visual quality of the LODs, calculated via the screen-space error. Our proposed heuristic prioritizes objects that are more likely to be in the FOV of the user and LODs that provide the highest gain in terms of visual quality. We model this adaptation as in the HAS domain by replacing the playout deadline, naturally provided for videos by the buffer level status, with the concept of *minimum reaction time to changes* in the AR scene. To ensure the relevance of the adaptation decision for the viewer in terms of selected object and LOD, the heuristic aims to fetch content that can be delivered within this time window. We showcased the benefits of the proposed approach using a prototype implemented on an iOS device, which streams an AR scene composed of four objects over the Internet. Compared to a download-and-play approach, our framework can greatly reduce the start-up latency without sacrificing on the visual quality of the AR objects. The work presented in this paper represents the first step in the broader domain of delivery optimization strategies for large-scale AR experiences. To conclude, here we identify a number of research challenges and areas for improvements that will be pursued as future work in this area.

B. Open Research Questions

6-DOF prediction. Being able to accurately predict the position of the user is a crucial task to provide the viewer with the most relevant content at all times. This problem has been extensively investigated in the 360° video and 3D graphics domains [23,24]. In the AR domain, this problem is further complicated because the user can move in six degrees-of-freedom (both head rotations and position translation in the scene). Moreover, an AR scene is composed of a set of discrete, separated objects, which are super-imposed to and could potentially interact with the real world.

Animations and triggers. The AR scene used for the experiments in this paper was composed of four static objects. Albeit simple, this configuration allowed us to clearly assess the performance of the proposed framework. In reality though, an AR scene can be much more complex, as the AR objects can be animated and therefore move and interact among each other. Moreover, objects can be associated with *triggers*, e.g., a particular set of behaviors that occur when the user interacts with the scene. As an example, a proximity trigger would cause an object to appear (or disappear) when the user would come close enough to it. Understanding the complex dynamics that can arise in an AR scene is an important task to anticipate the content the user would need in the near future.

Minimum reaction time to changes. The reaction time

Δ^* is an important parameter of the proposed framework, and it has to be carefully selected. A conservative choice can unnecessarily constrain the LOD adaptation heuristic, while an overestimation can result in a sub-optimal user experience. We plan to investigate in future work how to optimally set this value based on the scene content and the exploration behavior of the user.

Texture compression and streaming. An AR object is usually composed of multiple types of textures, each encoding a given characteristic of the object (color, reflection to light, roughness, etc.). Textures can account for a large portion of the AR object's size and their efficient compression and delivery is an important research area to guarantee an effective AR delivery strategy [25,26].

File format and compatibility. In the 3D/AR domain, a large amount of file format specifications have been proposed to represent and store 3D content, with no solution clearly outperforming the others, which clearly complicates interoperability. As an example, Sketchfab supports over fifty different 3D file formats[4]. The glTF[5] and USDZ[6] formats are gaining traction as industry standards, but how well these formats are suitable for streaming is still an open research question.

Visual quality assessment. While the screen-space error provides an objective way to asses the visual quality of an object, it is only computed based on the geometry of the object. It is therefore required to develop new visual quality metrics for the 3D domain that can take into account the trade-off between texture quality and mesh LOD.

Caching and novel LOD adaptation heuristics. In this paper, we proposed to model the LOD adaptation problem similar to a video rate adaptation problem, which opens up the possibility to re-factor existing adaptive video heuristics for the AR domain. Differently from video though, an AR client can be equipped with a local, limited memory cache where the objects are temporarily stored [11]. This caching policy should be designed in conjunction with the LOD adaptation heuristic to minimize expensive re-transmissions of the same content.

REFERENCES

[1] Apple, "Arkit software development kit," https://developer.apple.com/augmented-reality/, 2019.

[2] Google, "Arcore software development kit," https://developers.google.com/ar/, 2019.

[3] A. Insider, "762 million ar-compatible smartphones in the wild," https://arinsider.co/2018/07/25/762-million-ar-compatible-smartphones-in-the-wild/, 2018.

[4] Netflix, "Streaming data usage," https://help.netflix.com/en/node/87.

[5] S. Rusinkiewicz and M. Levoy, "Streaming qsplat: A viewer for networked visualization of large, dense models," in *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, ser. I3D '01. New York, NY, USA: ACM, 2001, pp. 63–68.

[6] H. Hoppe, "Smooth view-dependent level-of-detail control and its application to terrain rendering," in *Proceedings Visualization '98 (Cat. No.98CB36276)*, Oct 1998, pp. 35–42.

[7] F. Meng and H. Zha, "Streaming transmission of point-sampled geometry based on view-dependent level-of-detail," in *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, Oct 2003, pp. 466–473.

[8] J. Pouderoux and J.-E. Marvie, "Adaptive streaming and rendering of large terrains using strip masks," in *Proceedings of the 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, ser. GRAPHITE '05. New York, NY, USA: ACM, 2005, pp. 299–306.

[9] J. Chim, R. W. H. Lau, H. V. Leong, and A. Si, "Cyberwalk: a web-based distributed virtual walkthrough environment," *IEEE Transactions on Multimedia*, vol. 5, no. 4, pp. 503–515, Dec 2003.

[10] C. Zach, "Prefetching policies for remote walkthroughs," in *10th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2002*. ., 2002, pp. 153–159.

[11] Z. Zhou, K. Chen, and J. Zhang, "Efficient 3-d scene prefetching from learning user access patterns," *IEEE Transactions on Multimedia*, vol. 17, no. 7, pp. 1081–1095, July 2015.

[12] F. W. B. Li, R. W. H. Lau, D. Kilis, and L. W. F. Li, "Game-on-demand: : An online game engine based on geometry streaming," *TOMCCAP*, vol. 7, no. 3, pp. 19:1–19:22, 2011.

[13] J. Park, P. A. Chou, and J. Hwang, "Volumetric media streaming for augmented reality," in *2018 IEEE Global Communications Conference (GLOBECOM)*, Dec 2018, pp. 1–6.

[14] S. Petrangeli, V. Swaminathan, M. Hosseini, and F. De Turck, "An http/2-based adaptive streaming framework for 360-degree virtual reality videos," in *Proceedings of the 2017 ACM on Multimedia Conference*, ser. MM '17. New York, NY, USA: ACM, 2017, pp. 306–314.

[15] J. van der Hooft, T. Wauters, F. De Turck, C. Timmerer, and H. Hellwagner, "Towards 6dof http adaptive streaming through point cloud compression," in *Proceedings of the 27th ACM International Conference on Multimedia*, ser. MM '19. New York, NY, USA: ACM, 2019, pp. 2405–2413.

[16] U. Rijezon, M. Djupsjbacka, M. Bjrklund, C. Hger-Ross, H. Grip, and D. G. Liebermann, "Kinematics of fast cervical rotations in persons with chronic neck pain: a cross-sectional and reliability study," *BMC Musculoskeletal Disorders*, vol. 11, no. 1, 2010.

[17] P. Cozzi and K. Ring, *3D Engine Design for Virtual Globes*, 1st ed. Natick, MA, USA: A. K. Peters, Ltd., 2011.

[18] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, and R. Zimmermann, "A survey on bitrate adaptation schemes for streaming media over http," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 562–585, Firstquarter 2019.

[19] NovaVR3D, "Khon mask," https://sketchfab.com/3d-models/khon-mask-62f199ca65c04b3f960ba2224f7fa087, License: https://creativecommons.org/licenses/by/4.0/.

[20] alban, "Bella coola wooden mask," https://sketchfab.com/3d-models/bella-coola-wooden-mask-7ea34e01230d499a85778e986425fdff, License: https://creativecommons.org/licenses/by/4.0/.

[21] Rigsters, "Nepalese tribal mask," https://sketchfab.com/3d-models/nepalese-tribal-mask-916f5ac8df4849ed910d8b3f5ec455ef, License: https://creativecommons.org/licenses/by/4.0/.

[22] jkredzvr, "Kitsune festival mask," https://sketchfab.com/3d-models/kitsune-festival-mask-b52b35b4689842499a879f4b80ba25b0, License: https://creativecommons.org/licenses/by/4.0/.

[23] C. Li, W. Zhang, Y. Liu, and Y. Wang, "Very long term field of view prediction for 360-degree video streaming," in *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, March 2019, pp. 297–302.

[24] K. Lee, D. Chu, E. Cuervo, J. Kopf, Y. Degtyarev, S. Grizan, A. Wolman, and J. Flinn, "Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '15. New York, NY, USA: ACM, 2015, pp. 151–165.

[25] G. Simon, S. Petrangeli, N. Carr, and V. Swaminathan, "Streaming a sequence of textures for adaptive 3d scene delivery," in *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, March 2019, pp. 1159–1160.

[26] P. Krajcevski, S. Pratapa, and D. Manocha, "Gst: Gpu-decodable supercompressed textures," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 230:1–230:10, Nov. 2016.

[4]https://tinyurl.com/y3qlrggo

[5]https://www.khronos.org/gltf/

[6]https://graphics.pixar.com/usd/docs/Usdz-File-Format-Specification.html