Networking 2013 1569702431

Balance Visual Saliency, Reusability and Potential Relevance for Caching P2P 3D Streaming Contents

Wei Wang†, Jinyuan Jia†, Xiaojun Hei‡

†School of Electronics and Information Engineering, Tongji University, China
‡Dept. of Electronics & Information Engineering, Huazhong University of Science & Technology, China
ww.cs.tj@gmail.com, jiyuan@gmail.com, heixj@hust.edu.cn

***Abstract*—Recent technical progress on the Internet and virtual reality has enabled the proliferation of distributed virtual environments (DVEs). In a DVE, high-resolution 3D contents may generate huge data, and the peer-to-peer (P2P) streaming takes advantages to carry these huge traffic in a cost-effective manner. In this P2P paradigm, peers can cache and share DVE data cooperatively to reduce server workload and improve streaming quality. Nevertheless, it is critical to maintain and update the cached contents in each peer efficiently. In this paper, we propose an efficient caching algorithm for a P2P 3D content streaming framework. The proposed caching algorithm is based on a new preservation metric that is defined for balancing visual saliency, reusability and potential relevance of cached 3D objects. Then these cached 3D objects in each peer are updated adaptively with the ascendant order in importance quantified using this new metric. We implement the proposed caching algorithm in a simulated DVE platform for P2P-based 3D streaming. We conducted a comprehensive simulation study and our experimental results demonstrate that the proposed peer-to-peer streaming method outperforms the state-of-the-art 3D streaming methods (including FLoD and MRM) in the terms of fill ratio, base latency, requests by nodes and requests to the server.**

I. Introduction

With the advance of the Internet and virtual reality technologies, distributed virtual environments (DVEs) have become popular in recent years. By joining together in a virtual environment, users who are geographically dispersed are able to communicate and interact with each other on the Internet [1]. For a high-resolution 3D visual environment system, the 3D content data are huge (e.g., more than 34 TB data in *Second Life* and 70 TB data in *Google Earth*. Nevertheless, it is impossible to preload all these data into a client before a user starts to walkthrough in a virtual scene. Note that a user can only observe a small portion of the whole virtual scene due to the occlusions among 3D objects, a practical way is to download and render the data of the limited visible scene at the current viewpoint in a client. This strategy is particularly useful for thin clients, such as personal digital assistants (PDAs) and other mobile devices.

Various 3D content streaming techniques have been proposed to transfer DVE data, including client-server [2] and peer-to-peer [3], [4], [5], [6], [7], [8] architectures. It is not unusual that millions of clients join a DVE simultaneously. If they are all connected to the server that stores the whole copy of the virtual environment data, the server is likely to be the bottleneck due to its limited bandwidth. On the other hand, in a large-scale DVE system virtual components are redundant for each user. Each user caches these virtual components and transfers them cooperatively in a naive P2P manner. When a user navigates in a virtual environment, her/his viewpoints exhibit strong time and temporal correlations. In a finer granularity, 3D virtual scenes can be transferred using P2P mechanisms for the clients having similar viewpoints in the virtual environment. These clients are clustered as neighbors and a client can send requests to its neighbors for data transmission of the shared visible scenes. By distributing and reusing 3D objects among clients, P2P DVE systems are advantageous in alleviating the burden of the server and enhancing quality-of-experience of users.

Given the limited cache capacity of individual clients, only the visible portion of 3D scene can be loaded. When a user navigates a 3D virtual environment, certain objects must be removed and updated in the cache due to the changes of viewpoints. In this paper, we propose a progressive caching method for real-time navigation in large-scale P2P DVEs. This proposed caching method is based on three factors: in addition to the visual attention that is widely used in cache updating in client/server DVEs, we also propose two new factors of reusability and potential relevance. Our experimental results show that a weighted combination of these three factors provides a satisfied performance measure in updating cached objects.

The rest of this paper is organized as follows. In Section II, we summarize the related work on progressive transmission of DVEs and caching methods. Then, we proposed a P2P 3D content streaming framework in Section III. Within this framework, an efficient caching algorithm is also proposed in Section IV and the experimental results and performance analysis are discussed in Section V. Finally, we present the conclusion remarks and future work in Section VI.

II. Related Work

For 3D content streaming in a large-scale DVE, three key ingredients, area-of-interest (AOI) of 3D scene, progressive transmission and cache updating, are involved and their related work is summarized below.

A. AOI of complex 3D scenes

Given a limited cache capacity in a client, only a small portion of a complex 3D scene can be preloaded. Nevertheless, a virtual viewer can only see a small area of the whole virtual environment, the concept of *area of interest* (AOI) has been

proposed and used widely in DVE systems [2], [9], [10]. AOI is a circular area whose center is coincided with the current viewpoint and whose radius is proportional to the visible distance. All 3D objects enclosed in AOI can be reasonably thought as objects in the currently visible scene. When the viewpoint moves, the new added visible scenes have to be incrementally updated and downloaded from the server or other clients. An efficient scene culling algorithm is proposed in [11], which partitions the entire scene using an axis-aligned mesh. Visible and invisible grids are dynamically maintained according to the temporal coherence between two consecutive AOIs. An improved scalable multi-layer AOI scheduling algorithm is proposed in [12]. Two scheduling mechanisms, area-based and cell-based scheduling, are studied in [13], showing that a hybrid scheduling achieves the best performance.

B. Progressive transmission of 3D contents

Given a fixed viewpoint, AOI can greatly reduce the necessary 3D contents for browsing. The 3D objects contained in an AOI, however, could still be of huge data size if they are presented in a very high resolution. Level-of-details (LOD) techniques [14] have been developed, which models a 3D object O by a base shape $\mathcal{B}$ and a set of hierarchical details $\mathcal{D} = \{D^1, D^2, \cdots\}$. Then the object $O = \mathcal{B} \cup \mathcal{D}$ can be represented hierarchically by $O^1 \subset O^2 \cdots \subset O$, where $O^i = \mathcal{B} \oplus D^1 \oplus D^2 \oplus \cdots \oplus D^i$, $\oplus$ is a synthesis operator. Usually $\mathcal{B}$ is an overall simplified shape that can be transformed very rapidly. When a user watches and manipulates (e.g., translates and rotates) 3D objects, more and more details in $\mathcal{D}$ can be received by the client and smoothly synthesized into the model $\mathcal{B}$. Notably, two classes exist for generating a LOD representation: surface simplification [15] and progressive mesh [16]. In this paper, based on progressive representation [16] of 3D objects, we use the method in [2] to compute an optimal resolution for each 3D object located in an AOI that are suitable for the current viewpoint. Given the object progressive representation, a general problem formulation of P2P transmission for 3D contents are conducted in [17].

C. Caching methods of massive 3D contents

When the limited local cache in a client has been crammed by constantly downloaded scene (made up of 3D objects), some previously cached 3D objects have to be updated and replaced by new downloaded 3D objects (to constitute a new scene). Data replacement in the cache is an important issue that not only exists in 3D content streaming, but also in other well-studied applications as we summarize below.

Caching based on temporal factor. Page-based data replacement strategies, such as least recently used (LRU) and most recently used (MRU) have been widely used in database applications [18]. The merits of page-based replacement strategies relies heavily on the *principle of locality*, i.e., the higher degree of locality, the better performance can be achieved. However, it is shown in [19] that these strategies are not suitable for 3D content replacements since 3D objects that are accessed by a client might change frequently over time. Video streaming [20] is another popular application in which caching is widely studied. In P2P video streaming, video contents are regarded as one-dimensional (frames change in the dimension of discrete time) and thus can be accessed sequentially. A general caching strategy used in many P2P video streaming systems is that a receiver caches the recently played contents and supplies them to the requesters. However, P2P 3D content streaming is much more complex since the 3D objects are indexed in a 3D space and accessed according to viewer's interactive navigation behaviors.

Caching based on spatial factor. A 3D content caching method is proposed in [6] in which 3D objects are removed in a descending order indexed by their distances to the viewpoint. This caching method, however, does not consider the effect of objects' deviating angles from the viewpoint. The strategy of most required movement (MRM) in [2] streams each 3D object using a progressive representation and assigns to each 3D object an access score measured by a weighted combination of its distance and deviation angle from the viewpoint. To take the advantages of both temporal and spatial factors, a hybrid caching method is proposed in [21] which determines a replacement order for each 3D object using both temporal and spatial coherence. All these work determines a 3D object replacement order in the cache mainly based on the spatial relation between objects and viewpoint: they are better fitted into a framework of C/S DVE [18] but not for 3D content streaming in P2P DVEs. When determining an order to remove 3D objects in the cache in a P2P DVE, in addition to the spatial relations, the characteristics of the P2P mechanism such as the potential influence of a cached 3D object on the client's neighbors, should be also taken into account.

III. A Framework of P2P DVE

A number of C/S DVE systems and applications have been developed and deployed [22], [23], [24], but P2P-based DVEs are still few. We first propose a novel framework of a P2P DVE. Then we present an efficient caching algorithm suitable for this proposed P2P DVE. The framework of our P2P DVE is illustrated in Fig. 1: from bottom to top, the network architecture consists of three layers, i.e., physical network layer, Voronoi overlay layer and virtual scene layer. The physical network layer consists of one server (or sever clusters) and geographically dispersed clients. The Voronoi overlay layer is used to highlight awareness of avatars in 3D virtual environments and the virtual scene layer is the place where the avatars live. Below the terms $node$, $viewer$, $client$ and $peer$ are used interchangeably.

Given the spatial relation of viewers in the virtual environment, if a viewer v_i is located in the AOI of another viewer v_j, v_i is regarded as the neighbor of v_j. For example, in Fig. 1, v_3 and v_6 are both neighbors of v_4. Three types of participators involved in the proposed P2P DVE:

- *Server*: The server (or a server cluster) store all 3D contents of the DVE. It also manages clients such as registration in the system and recording their locations in the virtual environment.
- *Requester*: A client that requires pieces of 3D content data for virtual scene rendering.
- *Provider*: The server or a client that holds the required pieces of data and can provide them to a requester.

In the proposed P2P DVE, the roles of nodes are interchangeable, i.e., a requester can also be a provider for another

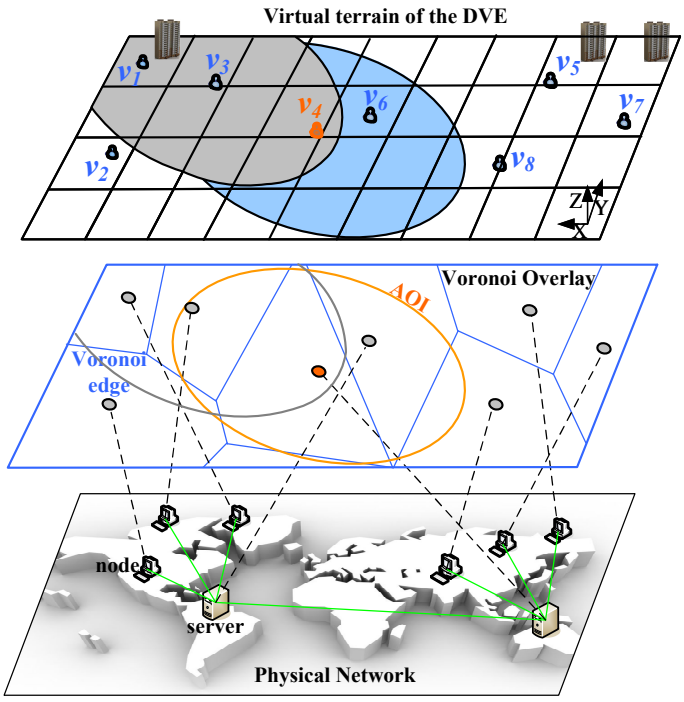

Fig. 1: Network architecture of a P2P-DVE.

node. Note that the server is only considered as a candidate content provider.

Given a requester v_k, we use a mesh-based pull scheme for P2P transmission of necessary 3D contents when v_k roams in the virtual environment [6]. Our proposed P2P framework utilizes a provider selection mechanism by considering peer heterogeneity, traffic locality and peer stability and thus construct a better transmission overlay than the one in [6].

- Step 1: Join the DVE. v_k joins the P2P DVE by registering to the server. Then v_k downloads the scene description file of the entire virtual environment. By applying the greedy forwarding mechanism in [5], v_k connects to its initial AOI neighbors.
- Step 2: Determine the visible scenes. When the viewpoint is initialized or moved, v_k determines its visible scenes by applying the incremental AOI culling method [11]. This step also produces a requesting queue for pieces of necessary 3D contents.
- Step 3: Update AOI neighbors. v_k sends the message of its position updating in the virtual scene layer to its current AOI neighbors. Updating AOI neighbors are found based on the methods utilizing the dynamic Voronoi diagram [25], [5].
- Step 4: Locate candidate providers. After updating AOI neighbors, v_k sends queries of required data pieces to its AOI neighbors. Those neighbors which response positively for a special piece are considered as candidate providers for this piece. A candidate provider also returns its current resource information, e.g., upload capacity and its Internet service provider (ISP).
- Step 5: Select providers. For each piece of required data, from the list of candidate providers, v_k selects a provider which can provide the best service (more details are presented below).
- Step 6: Transfer 3D contents. v_k receives data pieces from providers in parallel and renders the visible scene of current viewpoint for browsing. If the viewpoint moves during the data receiving process, the process is terminated and goto Step 2.
- Step 7: Caching by 3D scene replacement. If the limited cache has been fully occupied by 3D scene data, v_k invokes the proposed *progressive scene replacement mechanism*, as presented in Section IV.
- Step 8: Leave the DVE system. When v_k leaves the DVE system, it logouts from the server and disconnects from its AOI neighbors. For any abnormal leaving such as power failure, the server can diagnose the case by receiving messages from v_k's AOI neighbors.

In [6], only AOI neighbors in the Voronoi overlay layer are considered for P2P streaming and this may lead to a long-distance traffic in peers that are topologically close (in the virtual scene layer) but physically far away (in the physical network layer). In our P2P framework, we consider some more factors in a realistic networking environment. In Step 4, denote the upload capacity of each candidate provider i by u_i, $i = 1, 2, \ldots, n$, and u_0 is the upload capacity of the server in bit-per-second. Given the observation [26] that if a peer stays in a channel longer, this peer tends to keep staying in the channel, the stability s_i of each candidate provider i is measured by its staying time t_i in the DVE system. Let $t_{max} = \max\{t_1, t_2, \ldots, t_n\}$. We define a normalized stability $s_i = t_i/t_{max} \in [0, 1]$: if $s_i > s_j$, then peer i is more stable than peer j. We also define a general distance metric $D(r, i) \in [0, 1]$ between the requester r and a candidate provider i: any customized metric that measures the physical distance between r and i can be used and currently we define $D(r, i) = 0.1$ if r and i are in the same ISP and otherwise $D(r, i) = 1$. This distance definition penalizes cross-ISP traffic by a long physical distance. Let

$$w_i = s_i \cdot D(r, i) \tag{1}$$

be a weighted distance from r to the candidate provider i.

Given the information of w_i and u_i of each candidate provider i returned by Step 4, the provider selection with the best service in Step 5 uses the following strategy. We normalize the weights by $w'_i = w_i/w_{max}$, where $w_{max} = \max\{w_1, w_2, \ldots, w_n\}$. Set $w'_0 = 1$ for the sever of upload capacity u_0. We sort the upload capacities $\{u_0, u_1, \ldots, u_n\}$ of the sever and all candidate providers using the weight w'_i in a descending order and denote the sorted set by $U' = (u'_0, u'_1, \ldots, u'_n)$. Let h be the required bit rate (in bps) of the visible scene movement that is estimated by the viewpoint's velocity at the current position due to user behaviors. We choose the first k peers in U' as the providers, where

$$k = \min\{j : \sum_{i=0}^{j} w'_i u_i \geq h\}.$$

If all the weights are the same (as assumed in [27]), our provider selection strategy satisfies the stochastic fluid theory [27], since

$$\sum_{i=0}^{j} w_i' u_i \leq \sum_{i=0}^{n} \frac{w_i}{w_{max}} u_i \leq \frac{u_0 + \sum_{i=1}^{n-1} u_i}{n-1}.$$

IV. The Caching Algorithm

If the cache of a client is fully occupied by 3D scene data (i.e., 3D objects that constitute of the visible scene), the client has to update and replace some 3D scene data progressively according to the movement of viewpoint. Assume that the set of currently cached 3D objects is $\mathcal{O} = \{O_1, O_2, \ldots, O_m\}$. We define a preservation metric (Section IV-D) for each object O_i, which indicates the importance of O_i by referring to the current viewpoint and its AOI neighbors. This preservation metric is based on three factors: (1) the visual saliency degree (Section IV-A); (2) the reusability degree (Section IV-B); (3) the potential relevance degree for AOI neighbors (Section IV-C). The client determines the order of 3D objects in the cache to be removed by indexing the values output from the preservation metric.

A. Visual saliency degree of 3D objects

Note that 3D objects which can provide higher visual quality should be cached for a longer time. Recall that we use a progressive representation of a 3D object $O = \mathcal{B} \cup \mathcal{D}$, where $\mathcal{B}$ is the base shape for quick transmission and $\mathcal{D}$ is a set of geometric details for progressive transmission. We define the visual saliency degree of 3D object O by considering the base shape $\mathcal{B}$ and the detailed part $\mathcal{D}$ separately.

The base shape $\mathcal{B}$ of O is used for quick transmission such that a viewer can have an overlook of the object O instantly. However, if we oversimplify the object shape, the base shape may have a poor appearance and thus a poor visual saliency. Hence, we define the visual saliency degree of the base shape by the ratio

$$V_{\mathcal{B}} = \frac{C(\mathcal{B})}{C(O)},$$

where $C(\mathcal{B})$ and $C(O)$ are the complexities of the base shape and the full object, respectively. We measure the complexity of a mesh model by its surface area.

The detailed part $\mathcal{D}$ of O will progressively improve the visual quality of the object. Its visual saliency is determined by two factors: (1) the distance d from the object to the viewpoint; (2) the angle θ ($0 \leq \theta \leq \pi$) deviated from O to the viewing direction. See Fig. 2 for an illustration of these parameters.

We define the visual saliency degree of the detailed part $\mathcal{D}$ as

$$V_{\mathcal{D}} = \lambda(1 - \frac{d}{R}) + (1-\lambda)(1 - \frac{2\theta}{\pi}),$$

where R is the radius of AOI and $0 \leq \lambda \leq 1$ is a weighting coefficient. Finally, we define the visual saliency degree of an object $O_i \in \mathcal{O}$ as

$$V(O_i) = \omega V_{\mathcal{B}}(O_i) + (1-\omega) V_{\mathcal{D}}(O_i), \quad (2)$$

where $0 \leq \omega \leq 1$ is a weight that balances the contributions of the base shape and the detailed part. In our experiments, we set $\lambda = \omega = 0.5$. The value output from V is normalized in $[-1, 1]$.

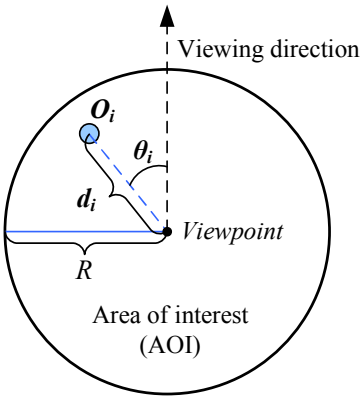

Fig. 2: The parameters in the definition of the visual saliency degree.

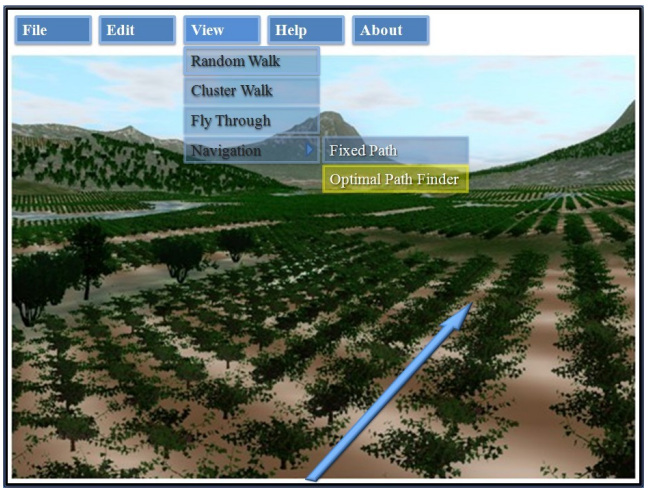

Fig. 3: A typical scene in the virtual environment with similar 3D tree models.

B. Reusability degree of 3D objects

In a large-scale DVE, repeated patterns frequently appear. For example, the trees aligned along the roadside (Fig. 3) and the buildings in commercial and resident areas. It is cost-effective to keep only one copy of the repeated or similar 3D objects in the cache. In addition, if a 3D object has many similar objects in the whole DVE scene, it is more preferable to cache this object for a long time. Based on this intuition, we define the following reusability degree of 3D objects.

Let $\widetilde{\mathcal{O}}$ be the set of all 3D objects in the DVE and $S(\cdot)$ is a metric that returns the similarity of two 3D objects in the range $[0, 1]$, where 1 means identical. Usually the 3D objects in $\widetilde{\mathcal{O}}$ have color information and the similarity metric S in [24] can be applied. We maintain a matrix in the scene description file that encodes the similarities between all 3D objects in the virtual environment, which is downloaded from the server when a client is registered. For each object $O_i \in \widetilde{\mathcal{O}}$, we compute the number $n(O_i)$ of 3D objects in $\widetilde{\mathcal{O}}$ that have

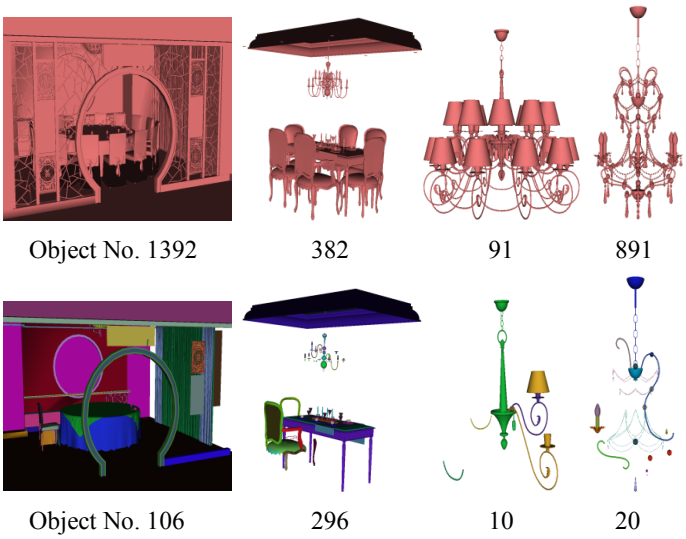

Fig. 4: 3D scenes before (top row) and after (bottom) applying the model reusability. Repeated objects are only shown by one instance in the bottom row. The complexity of 3D scenes is shown by the numbers of 3D objects used to render the scenes.

the similarity value larger than 0.9 with respect to O_i. We use $n(O_i)$ as a reusability measure of the object O_i and the reusability degree of O_i is defined as

$$R(O_i) = \frac{n(O_i)}{n_{max}}, \tag{3}$$

where $n_{max} = \max\{n(O_j)\}$, $\forall O_j \in \widetilde{\mathcal{O}}$. The value output from R is normalized in $[0, 1]$. Four examples of 3D scene renderings with model reusability are illustrated in Fig. 4.

C. Potential relevance degree of 3D objects

Each cached 3D object in a client is used not only for rendering the visible scene, but also for providing a piece of data that can be transmitted to its AOI neighbors. So the priority of removing 3D objects in the cache must consider the potential impact on its AOI neighbors.

For each 3D object $O_i \in \mathcal{O}$, let $ND(O_i)$ be the number of downloading O_i by the AOI neighbors. We use $ND(O_i)$ to estimate how many copies of O_i are maintained in the AOI neighbors. The more copies of a 3D object are maintained in neighbors, the higher priority of this 3D object is to be removed from the cache. We define the potential relevance degree of a 3D object $O_i \in \mathcal{O}$ as

$$P(O_i) = 1 - \frac{ND(O_i)}{ND_{max}}, \tag{4}$$

where

$$ND_{max} = \max\{ND(O_j), \forall O_i \in \mathcal{O}\}.$$

The value output from P is normalized in $[0, 1]$.

D. Combination of three degrees as the preservation caching criteria

We combine the three degrees defined as the weighted average of Eqs. (2), (3) and (4) for a preservation criteria to update cache:

$$M(O_i) = \alpha V(O_i) + \beta R(O_i) + \gamma P(O_i), \tag{5}$$

where $0 \leq \alpha, \beta, \gamma \leq 1$ and $\alpha + \beta + \gamma = 1$. In our experiments, we select $\alpha = \beta = \gamma = \frac{1}{3}$. We use the preservation criteria (5) to build replacement priorities for cached 3D objects: if a 3D object has a lower value of this preservation degree, it is removed from the cache with a higher priority.

Denote the cache capacity of a client as C_{full}, the cached data volume as $C_{occupied}$ and the data volume of the next downloading request as $Data_{request}$. Given the preservation criteria (5), we sort the 3D objects in $\mathcal{O}$ using the value $M(O_i)$, with the first object having the lowest value. Without confusion, $\mathcal{O}$ below denotes the ordered set of 3D objects in the cache.

Our caching algorithm can be outlined in two major steps:

- Step 1. If $C_{full} - C_{occupied} < Data_{request}$, then start the object replacement procedure.
- Step 2. Download the requested data.

The object replacement procedure in Step 1 consists of three sub-steps:

- Step 1.1. Access the 3D objects in $\mathcal{O}$ sequentially and if any 3D object $O_i \in \mathcal{O}$ has redundant details $(D_i^{j+1}, D_i^{j+2}, \cdots)$, then remove these details until the optimal resolution of the object $O_i^j = \mathcal{B} \oplus D_i^1 \oplus D_i^2 \oplus \cdots \oplus D_i^j$ is reached. The object's optimal resolution is determined by using the method in [2]. If $C_{full} - C_{occupied} \geq Data_{request}$, goto Step 2.
- Step 1.2. Access the 3D objects in $\mathcal{O}$ sequentially and remove the detailed part $\mathcal{D}$ of the accessed 3D object O_i (i.e., only keep the base shape $\mathcal{B}$ of O_i). If $C_{full} - C_{occupied} \geq Data_{request}$, goto Step 2.
- Step 1.3. Access the 3D objects in $\mathcal{O}$ sequentially and remove the base shape $\mathcal{B}$ of the accessed 3D object O_i. If $C_{full} - C_{occupied} \geq Data_{request}$, goto Step 2.

V. Performance Analysis

To evaluate the performance of the proposed caching algorithm in a large-scale DVE, we design and implement a P2P DVE simulator in which the following two aspects are considered in-depth:

1) To evaluate whether the combination in the metric (5) is optimal, we compare it with an alternative metric:
$$M(O_i) = 0.5V(O_i) + 0.5P(O_i). \tag{6}$$
Since the visual saliency is a critical measure in the navigation of 3D virtual environment and potential relevance is a critical measure in P2P transmission of 3D contents, we keep items $V(O_i)$ and $P(O_i)$ in both metrics (5) and (6). So the comparison between the metrics (5) and (6) is to evaluate the significance of the reusability measure $R(O_i)$ in Eq. (3).
2) We compare our caching algorithm with two other caching algorithms [2], [6]. The most required movement (MRM) strategy in [2] is a classic one in 3D

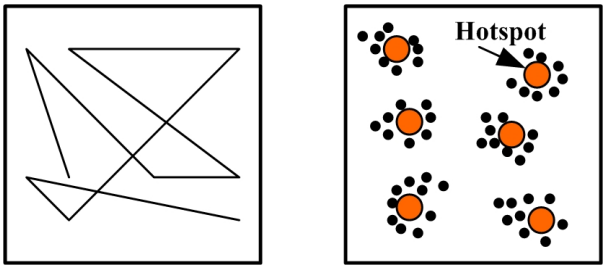

Fig. 5: Two navigation behaviors. Left: A random walk. Right: A clustering movement, in which the point size indicates the possibility that a viewer is likely moved to this position.

content streaming that progressively transmits the base shapes and detailed parts of 3D objects, and thus is closely related to our work. Among state-of-the-art P2P DVEs [3], [4], [5], [6], [7], [8], we use a similar P2P content streaming mechanism as in [6]. Hence, we compare our caching algorithm to these two works. Denote PSRM (progressive scene replacement mechanism) as the abbreviation for our method, P2P-MRM in [2] and FLoD in [6], respectively. Furthermore, we distinguish our method using the metrics (5) and (6), respectively, by PSRM (using the metric (5)) and PSRM-N (using the metric (6)).

A. Experimental settings

We have implemented a prototype system of the proposed P2P DVE in a LAN environment with multiple users. Since the system scale plays an important role in the evaluation of our caching algorithm, we also implement a simulated large-scale simulator that can capture the real behaviors of large-scale P2P DVE using a set of open-source programming libraries. The simulated large-scale DVE platform is stored in a server. The number of 3D objects in the simulated virtual environment is of the order of magnitude of 10^3 to 10^4. The file size and the position of each 3D object is set randomly and the progressive representation is used in all 3D objects. We generate two special object types called *trees* and *buildings*. The similarity value of two 3D objects in each of special object types is set to be 0.99 and the similarity value in the remaining 3D objects is randomly drawn from a poisson distribution in $[0, 1]$.

Our discrete-time simulator uses a time step of 100 ms and then 3000 steps means a simulation process of 300 seconds. To perform a faithful comparison with MRM [2] and FLoD [6], in our simulation we implement two navigation behaviors of a viewer in P2P DVE (Fig. 5):

- A random walk [2]: the orientation and position of a viewer at each step of movement are generated randomly.
- A clustering movement [6]: a viewer has a higher possibility for moving to nearby special hotspots and has a lower possibility of moving to remaining sites. We randomly generate $1.5 \ln n$ hotspots in the 2D map of a virtual environment, where n is the number of 3D objects in the map.

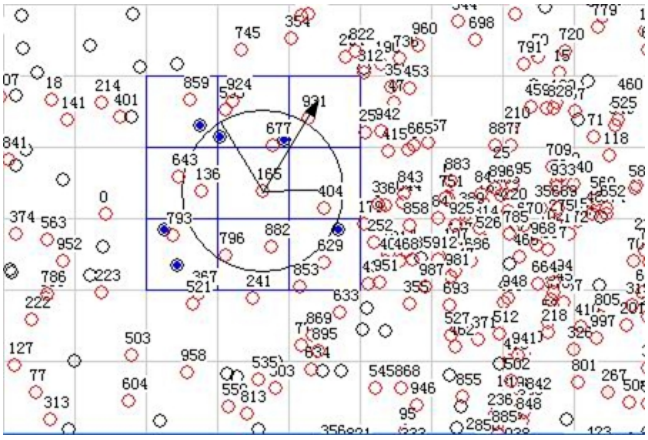

Fig. 6: A part of a screenshot of the P2P DVE simulator.

Since peer heterogeneity, traffic locality and peer stability are not considered in [2], [6], we set all the nodes with the same upload capacity and located in the same ISP; i.e., we only use peer stability to determine the weight w_i in Eq. (1) for each candidate provider. Note that FLoD [6] did not consider the network latency among different nodes in the P2P overlay. Actually two close viewers in the virtual scene layer may be far from each other geographically in the physical network layer (See Fig. 1). To simulate the real-world applications, network latency among different viewers in the virtual scene layer is randomly assigned from 100 ms to 1500 ms in our simulator. Furthermore, FLoD [6] only allowed viewers to start a navigation when 99% data of the visible scene within initial AOI has been downloaded. However, in realistic situations, a user may not have enough patience to wait for a long downloading time. Hence, we remove this strong restriction in our simulation.

A part of a screenshot of the simulated P2P DVE system is shown in Fig. 6 in which the numbers indicate the IDs of nodes in the P2P DVE. Similar to the FLoD [6], we run all the simulations in 3000 steps (i.e., 300 seconds). To evaluate the stable state behavior, the performance analysis is based on the statistic data collected in the last 2000 steps for each simulation. The settings of the detailed parameters used in the simulator are summarized in Table I. Note that in Table I, we have two settings of the number of nodes, i.e., 500 and 1000.

TABLE I: Parameter settings in the simulator

Parameter	Value
DVE dimensions (units)	5000×5000
Cell width (units)	50×50
3D Object file size (KB)	$20 - 50$
3D Object number in DVE	10000
Complexity ratio of base shape over the full object	10%
Increment size in detailed part (Bytes)	50
AOI radius	75
Connection limit of AOI neighbors	20
Number of nodes in DVE	500, 1000
Server upload bandwidth	10 Mbps
Peer download bandwidth	66 KB/s
Peer upload bandwidth	33 KB/s

B. Performance metrics

We use the following metrics to evaluate the performance of the caching algorithms, PSRM, P2P-MRM and FLoD, in the simulated P2P DVE.

- Fill ratio (FR). When a viewer in the virtual scene layer moves to a new position in the virtual environment, FR is the size ratio of the available data for the visible scene in the client (already downloaded) divided by all the data required for rendering the visible scene (should be downloaded). Note that in our definition of FR, we use the optimal resolution of each 3D object as in MRM [2], while FLoD [6] used the full resolution of 3D objects.
- Base latency (BL). BL is the latency between the time instant, when a client requests the data of all 3D objects in the visible scene, and the time instant, when it receives all the base shapes of these 3D objects. Once the base shapes are available in a client, a viewer can start a meaningful navigation in the virtual environment.
- Requests by nodes (RN). RN is the totally number of requests delivered by all the nodes in the DVE at each simulating step. All the data communicated in the physical network layer (See Fig. 1) is composed of two parts: the transmission of 3D contents and the requests delivered by clients and the server. A good caching algorithm should reduce the requests by the nodes (corresponding to clients in the physical network layer) as well as the delivery of 3D contents.
- Requests to the server (RS). RS is the total number of requests received by the server at each simulated step. When the AOI neighbors do not have the 3D content requested by a client, it has to send requests to the server for downloading the data. A good caching algorithm should reduce the RS as well as the RN.

C. Experimental results

Denote that M is the total data size of the DVE, R is the radius of a viewer's AOI, H and W are the height and width of the whole area of the virtual environment, respectively. The the average data size of an AOI can be defined as

$$\overline{Data}_{AOI} = \frac{\pi R^2}{HW} M.$$

We define the cache ratio as the ratio of a client's cache capacity over the $\overline{Data}_{AOI}$. It is clear that the effectiveness of any caching algorithm is dependent of the cache ratio. In our study, we use five settings of cache ratios: 0.25, 0.5, 1, 2, 3.

1) Fill ratios: Fig. 7 shows the experimental results of fill ratios of four methods with respect to different cache ratios in a 500-node DVE system with two navigation behaviors (i.e., random walk and clustering movement). The first observation from the results in Fig. 7 is that the fill ratio does not increase to 1 when the the cache ratio is increasing to 2 and 3. This can be explained by that even if a client has more cache capacity than the required 3D visible scene data, it may still not have the full data of the visible scene at each step, due to the large displacement of viewpoint movement and the limited rendering ability of that client (the rendering time is increased dramatically when the 3D scene data is increased above the optimal resolutions [28]).

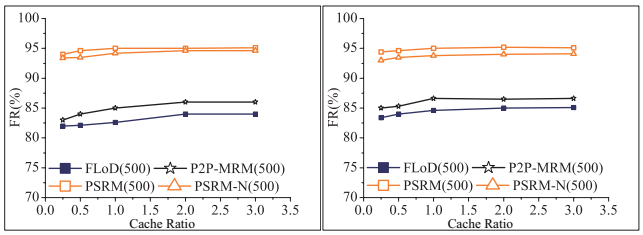

Fig. 7: The fill ratios in a 500-node DVE system with respect to different cache ratios. Left: the behavior of a random walk is used. Right: the behavior of a clustering movement.

As shown in Fig. 7, in a 500-node DVE system, when the cache ratio decreases from 2 to 1, the fill ratio of FLoD is reduced from 82% to 81% in a random walking, and from 84% to 82% in a clustering movement, respectively. When the cache ratio reaches 0.25, the fill ratio of FLoD is reduced to 80% in a random walk and to 82% in a clustering movement. PSRM, PSRM-N and P2P-MRM (MRM in a P2P DVE) generally have the higher fill ratio than FLoD. This demonstrates that using the optimal resolution with respect to the current viewpoint based on a progressive representation of 3D objects, it is possible to maintain a relative high fill ratio even if the cache ratio is very small. A similar conclusion can be drawn from a test on 1000-node DVE system as shown in Fig. 8. In both tests on 500-node and 1000-node DVE system, the PSRM has the highest fill ratios over the four methods: this can be explained by that PSRM takes object reusability and potential relevance into consideration and thus has the higher fill ratios than other methods. The property of maintaining high fill ratios makes PSRM particularly suitable in a small-cache-capacity condition of a client such as mobile phones and PDAs.

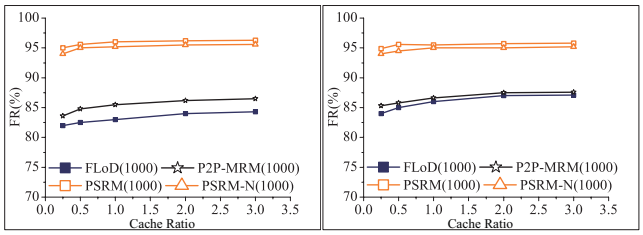

Fig. 8: The fill ratios in a 1000-node DVE system with respect to different cache ratios. Left: the behavior of a random walk is used. Right: the behavior of a clustering movement.

2) Base latency: The base latency measures the delay of receiving all necessary base shapes of visible scene under the current viewpoint. Since P2P-MRM and PSRM apply a caching mechanism that always postpone to remove base shapes from the cache as later as possible, it is expected that P2P-MRM, PSRM-N and PSRM should have the less base latency than FLoD. This expectation is demonstrated by the experimental results shown in Fig. 9 and 10, in which two DVE systems (500-node and 1000-node) are tested.

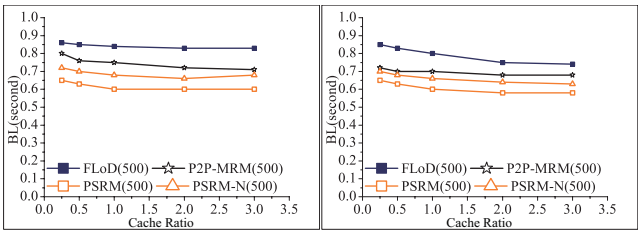

Fig. 9: The base latency in a 500-node DVE system with respect to different cache ratios. Left: the behavior of a random walk is used. Right: the behavior of a clustering movement.

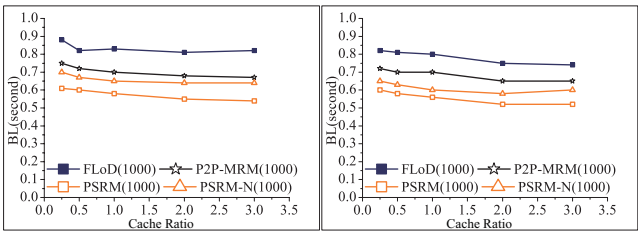

Fig. 10: The base latency in a 1000-node DVE system with respect to different cache ratios. Top: the behavior of a random walk is used. Bottom: the behavior of a clustering movement.

As summarized in Fig. 9 and 10, in either navigation behavior of random walking or cluster movement, the base latency of FLoD is the highest (always larger than 0.7 seconds). The base latency of PSRM-N is less than P2P-MRM, and PSRM is less than PSRM-N in turn. PSRM has the lowest base latency (always less than 0.65 seconds). There is one more reason to explain the best base latency performance of PSRM over FLoD: in peer selection, PSRM selects a peer from its AOI neighbors based on the candidate providers' available resource (i.e., peer stability), while FLoD randomly selects a peer from its AOI neighbors.

3) Requests by nodes: We examined the requests by nodes as a measure of network dataflow in the P2P DVE. PSRM and PSRM-N have much less requests by nodes than P2P-MRM and FLoD. This demonstrates that the caching algorithm that keeps the base shapes as long as possible and uses the potential relevance degree has a good performance in reducing the data requests from a client to its AOI neighbors in the proposed P2P DVE. PSRM generally has less requests by nodes than PSRM-N. This demonstrates that the reusability degree not only enhances the navigation experience in virtual environment (evaluated by fill ratios and base latency), but also can reduce the data from clients sent into the P2P network. The data of requests by nodes of four methods is stable with respect to different cache ratios. This is due to the fact that when the cache capacity is increased, the bottleneck of 3D scene rendering in client's CPU plays a more and more important role in the navigation of virtual environment.

4) Requests to the server: We also examined the requests to the server as another measure of network dataflow in the P2P DVE. Our results shows that PSRM and PSRM-N have much less requests to the server than P2P-MRM and FLoD. This demonstrates that our caching algorithm is particularly suitable in the proposed P2P DVE, since very few data requests are sent back directly to the server. We regard that this suitability comes from the following two aspects: i) Our caching algorithm takes 3D object usability into account, which makes a client always downloads the frequently used 3D objects first and maintains them as long as possible in the cache, such that very few data requests are sent to the server when the viewpoint in virtual environment is changed. ii) Our caching algorithm takes potential relevance of 3D objects into account, which makes the AOI neighbors of a client have as much data as possible that may be requested by that client, such that few data requests have to be sent to the server.

D. Scalability of PSRM

The experimental results have shown that by evaluating with fill ratios, base latency, requests by nodes and requests to the servers, PSRM outperforms PSRM-N, P2P-MRM and FLoD in a large-scale DVE (measured in 500-node and 1000-node systems). It is interesting to ask whether the PSRM is still suitable for small- or medium-scale P2P DVE? We thus test the four methods in P2P DVE systems of scales of 100, 200, 300, 400 and 500 nodes, respectively, with a fixed cache ratio 0.25.

Fig. 11 shows the fill ratio of four methods with two navigation behaviors. The results clearly show that with the decrease of nodes in the DVE to a small scale, the fill ratio decreases gradually in all the four methods, but PSRM is still generally has the highest fill ratios due to its usage of the progress representation and the base shape removal delaying mechanism in the caching algorithm. When there is very few nodes in the DVE, the Voronoi overlays become smaller and each node can find fewer AOI neighbors; thus, the performance of a P2P DVE in a small scale will closely behave like a C/S DVE. The tendency to C/S DVE is demonstrated by the results summarized in Table II, in which the index of requests to the server is increased when the node number is decreased. Also revealed by the data in Table II, the difference between the two navigation styles of random walking and clustering movement becomes very small since there are few nodes in the DVE. In all the cases, the performance of PSRM is slightly better than PSRM-N: this demonstrates that the reusability of 3D objects can still improve the performance in a small-scale P2P DVE.

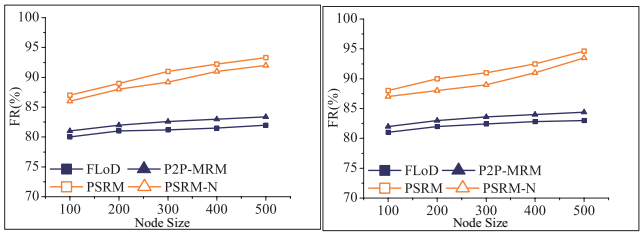

Fig. 11: The fill ratios in the DVE system with a fixed cache ratio 0.25. Left: the behavior of a random walk is used. Right: the behavior of a clustering movement.

VI. Conclusion

In this paper we propose a P2P DVE framework that utilizes a progressive representation of 3D objects. An efficient caching algorithm is proposed for this P2P DVE framework.

TABLE II: Requests to the server per step in a P2P DVE with fixed cache ratio 0.25

	A random walk		A clustering movement	
Node size	PSRM-N	PSRM	PSRM-N	PSRM
100	335	324	339	327
200	292	279	295	271
300	223	215	216	210
400	163	152	157	146
500	123	118	129	115

Three distinct features are designed in the proposed caching algorithm.

1) Consider more realistic factors in the P2P DVE framework including peer heterogeneity, traffic locality and peer stability.
2) Utilize a cache scheduler that postpones the base shapes of 3D objects as later as possible by always removing the detailed parts of 3D objects first.
3) In addition to the traditional visual factor measurement (similar to the one defined in Section IV-A) for ranking the importance of 3D objects in the cache, two more new measures based on object reusability and potential relevance in AOI neighbors are introduced to capture the traffic patterns of 3D DVE streaming applications.

We conducted a comprehensive simulation study. Our experiment results demonstrate that our proposed caching algorithm outperforms two classic methods [2], [6], in terms of fill ratio, base latency, requests by nodes and requests to the server. The proposed caching algorithm performs well on clients having small cache capacity in both large-scale and small-scale P2P DVEs. Nevertheless, in our experiments, we use a fixed setting of weights $\alpha = \beta = \gamma = \frac{1}{3}$ in the preservation metric (5). In addition, user behaviors may impact the performance of the caching algorithm heavily [29]. We plan to improve the caching performance by finding an optimal weight settings in our algorithm depending on customized user behaviors as our future work.

ACKNOWLEDGEMENT

This work has been partially supported by the NSFC (60972014,61272276), the Fundamental Research Funds for the Central Universities (HUST:2012TS018) and the Technology Support Plan of the National "Twelfth Five-Year-Plan" of China (2011BAK08B00,2012BAC11B00-04-03).

REFERENCES

[1] A. Steed and M. F. Oliveira, *Networked Graphics: Building Networked Games and Virtual Environments*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009.

[2] J. Chim, R. Lau, H. Leong, and A. Si, "CyberWalk: a web-based distributed virtual walkthrough environments," *IEEE Trans. Multimedia*, vol. 5, no. 4, pp. 503–515, 2003.

[3] J. Botev, A. Hohfeld, H. Schloss, I. Scholtes, P. Sturm, and M. Esch, "The HyperVerse: concepts for a federated and torrent-based '3D web'," *International Journal of Advanced Media and Communication*, vol. 2, no. 4, pp. 122–128, 2008.

[4] R. Cavagna, J. R. andPatrick Gioia, C. Bouville, M. Abdallah, and E. Buyukkay, "Peer-to-peer visualization of very large 3D landscape and city models using MPEG-4," *Signal Processing: Image Communication*, vol. 24, no. 1-2, pp. 115–121, 2009.

[5] S.-Y. Hu, J.-F. Chen, and T.-H. Chen, "VON: a scalable peer-to-peer network for virtual environments," *IEEE Network*, vol. 20, no. 4, pp. 22–31, 2006.

[6] S.-Y. Hu, T.-H. Huang, S.-C. Chang, W.-L. Sung, J.-R. Jiang, and B.-Y. Chen, "FLoD: a framework for peer-to-peer 3D streaming," in *IEEE INFOCOM*, 2008, pp. 1373–1381.

[7] J. Royan, P. Gioia, R. Cavagna, and C. Bouville, "Network-based visualization of 3D landscapes and city models," *IEEE Computer Graphics and Applications*, vol. 27, no. 6, pp. 70–79, 2007.

[8] M. Zhu, S. Mondet, G. Morin, W. T. Ooi, and W. Cheng, "Towards peer-assisted rendering in networked virtual environments," in *ACM Multimedia*, 2011, pp. 183–192.

[9] F. W. Li, R. W. Lau, D. Kilis, and L. W. Li, "Game-on-demand: An online game engine based on geometry streaming," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 7, no. 3, 2011.

[10] G. Popescu and C. Codella, "An architecture for QoS data replication in network virtual environments," in *Proceedings of Virtual Reality, IEEE*, 2002, pp. 41–48.

[11] J. Jia, P. Wang, S. Wang, and Y. Wang, "An integer incremental AOI algorithm for progressive downloading of large scale VRML environments," in *Edutainment*, 2007, pp. 711–722.

[12] W. Wang and J. Jia, "An incremental SMLAOI algorithm for progressive downloading large scale WebVR scenes," in *ACM Web3D*, 2009, pp. 55–60.

[13] K. Pan, W. Cai, X. Tang, S. Zhou, and S. Turner, "A hybrid interest management mechanism for peer-to-peer networked virtual environments," in *IEEE Parallel & Distributed Processing*, 2010, pp. 1–12.

[14] D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson, and R. Huebner, *Level of Detail for 3D Graphics*. Morgan Kaufmann Pub., 2002.

[15] M. Garland and P. Heckbert, "Surface simplification using quadric error metrics," in *ACM SIGGRAPH*, 1997, pp. 209–216.

[16] H. Hoppe, "Progressive mesh," in *ACM SIGGRAPH*, 1996, pp. 99–108.

[17] S.-Y. Hu, J.-R. Jiang, and B.-Y. Chen, "Peer-to-peer 3D streaming," *IEEE Internet Computing*, vol. 14, no. 2, pp. 54–61, 2010.

[18] M. Franklin, M. Carey, and M. Livny, "Global memory management in client-server DBMS architectures," in *Proceedings of VLDB*, 1992, pp. 596–609.

[19] A. Si and H.-V. Leong, "Adaptive caching and refreshing in mobile databases," *Personal Technologies*, vol. 1, no. 3, pp. 156–170, 1997.

[20] J. Apostolopoulos, W.-T. Tan, and S. Wee, *Video Streaming: Concepts, Algorithms, and Systems*. Technical Report HPL-2002-260, Mobile and Media Systems Laboratory, Hewlett-Packard Company, 2002.

[21] T.-Y. Li and W.-H. Hsu, "A data management scheme for effective walkthrough in large-scale virtual environments," *The Visual Computer*, vol. 20, no. 10, pp. 624–634, 2004.

[22] S. Singhal and M. Zyda, *Networked Virtual Environments: Design and Implementation*. Addison-Wesley Professional, 1999.

[23] Y.-J. Liu, Z. Chen, and K. Tang, "Construction of iso-contours, bisectors, and Voronoi diagrams on triangulated surfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1502 –1517, Aug. 2011.

[24] Y.-J. Liu, Y.-F. Zheng, L. Lv, Y.-M. Xuan, and X.-L. Fu, "3D model retrieval based on color+geometry signatures," *The Visual Computer*, vol. 28, no. 1, pp. 75–86, 2012.

[25] M. Albano, R. Baraglia, M. Mordacchini, and L. Ricci, "Efficient broadcast on area of interest in Voronoi overlays," in *IEEE CSE*, 2009, pp. 224–231.

[26] M. Bishop, S. Rao, and K. Sripanidkulchai, "Considering priority in overlay multicast protocols under heterogeneous environments," in *IEEE INFOCOM*, 2006, pp. 1–13.

[27] R. Kumar, Y. Liu, and K. W. Ross, "Stochastic fluid theory for P2P streaming systems," in *IEEE INFOCOM*, 2007, pp. 919–927.

[28] T. Akenine-Moller, E. Haines, and N. Hoffman, *Real-Time Rendering*. 3rd ed., A.K. Peters Ltd.,, 2008.

[29] P. Morillo, S. Rueda, J. M. Orduña, and J. Duato, "Ensuring the performance and scalability of peer-to-peer distributed virtual environments," *Future Gener. Comput. Syst.*, vol. 26, no. 7, pp. 905–915, July 2010.