Efficient 3-D Scene Prefetching From Learning User Access Patterns

Zhong Zhou, *Member, IEEE*, Ke Chen, and Jingchang Zhang

***Abstract*—Rendering large-scale 3-D scenes on a thin client is attracting increasing attention with the development of the mobile Internet. Efficient scene prefetching to provide timely data with a limited cache is one of the most critical issues for remote 3-D data scheduling in networked virtual environment applications. Existing prefetching schemes predict the future positions of each individual user based on user traces. In this paper, we investigate scene content sequences accessed by various users instead of user viewpoint traces and propose a user access pattern-based 3-D scene prefetching scheme. We make a relationship graph-based clustering to partition history user access sequences into several clusters and choose representative sequences from among these clusters as user access patterns. Then, these user access patterns are prioritized by their popularity and users' personal preference. Based on these access patterns, the proposed prefetching scheme predicts the scene contents that will most likely be visited in the future and delivers them to the client in advance. The experiment results demonstrate that our user access pattern-based prefetching approach achieves a high hit ratio and outperforms the prevailing prefetching schemes in terms of access latency and cache capacity.**

***Index Terms*—3-D scenes, networked virtual environment, prefetching, user access patterns.**

I. Introduction

In RECENT years, 3-D virtual environments have received considerable attentions. Traditional networked virtual environments apply the complete replication of all scene data to maintain interactivity. Some pioneering companies, such as Onlive and Gaikai, have proposed video streaming based remote rendering instead of local data management and rendering. However, their approaches suffer from display quality degradation and interaction lag, creating difficulties. Recent developments in mobile Internet technologies and GPU chips lead us to believe that in the near future, thin devices will be able to run PC-style virtual environments by fetching scene data on demand or perhaps even before demand. Local rendering will then be performed to guarantee rapid responses to interactions. As 3-D virtual environments become increasingly realistic looking, extensive and complex, the amount of scene data to be prefetched will become quite large, which will burden both the network and client cache. Therefore, an efficient prefetching scheme must accurately predict the scene contents that will be visible to the user and then fetch them in advance.

Existing prefetching schemes for 3-D scenes typically predict the future position of a user avatar using extrapolation models based on the user's trace history. However, different avatars in the same position will have different visibility sets because of their different viewing directions, speeds and fields of view. The current trace based prediction schemes define a zone within a specific distance from the user avatar as an area of interest (AOI) and request all scene data within the AOI as the visibility superset. These AOI derived schemes construct a much larger visibility set than a user actually uses. In fact, the scene contents that are visited are strongly related to user access behaviors, user interests and scene content popularities. Thus, incorporating these factors into prefetching schemes is a promising task.

In this paper, we investigate scene content sequences accessed by various users instead of user viewpoint traces and propose a user access pattern based 3-D scene prefetching scheme. We make an offline clustering analysis on user access histories to discover access patterns and then prioritize these patterns by their popularities and users' personal preferences. The proposed prefetching approach combines the access patterns and current access chunk set to make a prediction about the scene contents that are most likely to be visited in the future. Then, the predicted scene contents are delivered to the client in advance when the network is idle. With a limited cache, the proposed prefetching approach can significantly improve the hit ratio and reduce the access latency. The main contributions of this paper are as follows:

1) introduce a novel 3-D scene prefetching scheme, which can accurately predict the scene contents that are most likely to be visited in the future based on user access pattern mining instead of the extrapolation of user traces;
2) propose a user access pattern mining method, which extracts access sequences from user access histories, builds a relationship graph based on the proposed access sequence similarity, and then makes a relationship graph based clustering to discover user access patterns, i.e., the representative sequences of the clusters;
3) present the priorities for user access patterns, which prioritize user access patterns for each individual user by incorporating the popularities of user access patterns and the user's personal preference.

Manuscript received December 15, 2014; revised April 02, 2015; accepted April 28, 2015. Date of publication May 07, 2015; date of current version June 13, 2015. This work was supported by the National 863 Program of China under Grant 2015AA016403 and by the Natural Science Foundation of China under Grant 61170188. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Yonggang Wen.

The authors are with the State Key Lab of Virtual Reality Technology and Systems, Beihang University, Beijing 100191, China (e-mail: zz@buaa.edu.cn; chenke19850113@gmail.com; jczhang@buaa.edu.cn).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors. This includes a demo video that displays the framework of the proposed prefetching scheme and examples of the prefetching results. This material is 16.3 MB in size.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2015.2430817

The remainder of this paper is organized as follows. Section II reviews previous work. Section III presents our motivations and an overview of the proposed user access pattern based 3-D scene prefetching scheme. Section IV describes the details of user access pattern mining. Section V introduces the 3-D scene prefetching approach with user access patterns and the cache replacement policy. The experiment results are described in Section VI. Finally, Section VII draws the conclusions.

II. Related Work

When walking through a 3-D scene, a user explores the scene by browsing and interacting with the objects inside the scene. To support real-time interaction, traditional systems, such as DIVE [1], SIMNET [2], CALVIN [3] and VLNET [4], replicate the scene data at the clients before launching. They employ a complete replication and download all resource data to the clients beforehand. As the data size of the scene increases, the transmission and client storage overhead for complete replication becomes extremely large. This will negatively impact the startup time, storage requirements and performance of thin clients. Moreover, in a 3-D scene, a user visits only a small portion of the scene at any given time.

The task of determining the objects that are visible from a user's viewpoint has been widely studied. Many culling methods aiming to eliminate invisible objects have been proposed. Cohen *et al.* present a comprehensive survey of occlusion culling [5]. Occlusion culling is a computationally complex task, and many online culling methods apply graphics hardware for acceleration [6]–[9]. It determines the visible objects for the current viewpoint and is an important technique for reducing local rendering overhead. However, in networked virtual environments, it is impractical to introduce the round-trip lag incurred by culling for each frame.

AOI based methods represent the prevailing scheme for determining the visible objects in a 3-D scene. The AOI is typically an area in the shape of a circle or circle variant around the viewpoint with the visibility distance as its radius. All 3-D objects inside the AOI are regarded as objects in the potentially visible set. When the viewpoint moves, the newly added visible objects must be incrementally updated and downloaded. AOI was first proposed by Macedonia *et al.* [10] and is widely used for 3-D scene scheduling [11]–[13]. Wang *et al.* [14] consider the view frustum and the distance from the viewpoint to divide the AOI into several sections with different download priorities. Hu *et al.* [15], [16] propose the flow level of detail (FLoD), an AOI based solution for delivering 3-D content between peers in a P2P network. Aljaafreh *et al.* [17] propose a multi-level AOI for streaming 3-D scenes to mobile devices. They set the radii of the multi-level AOI according to the mobile devices' computing capability.

AOI data are updated based on distance, which will lead to large amounts of data transmission over short times. Prefetching has therefore been proposed to reduce the data transmission volume and improve system performance. It predicts objects that are likely to be visible in the future and delivers them to the client in advance. Many prefetching approaches predict the next position of a viewpoint depending on its current location, direction and velocity [18]–[20]. Then, the predicted position is used to compute download priorities or to prefetch 3-D objects. Varadhan and Manocha [21] incorporate level-of-detail (LOD) switching and visibility culling and propose a prioritized scheme for scene prefetching. Chim *et al.* [22] develop the CyberWalk system, which employs an exponentially weighted moving average (EWMA) to predict the next location of a user. Chan *et al.* [23] propose a hybrid motion model for predicting mouse motion and then map the predicted mouse motion into the 3-D scene to calculate a user's motion. Li and Hsu [24] also study mouse motion and propose a most likelihood movement (MLM) prefetch model for scene scheduling. Zheng *et al.* [25] use a first-order dead-reckoning algorithm to predict the position and direction of the viewpoint for upcoming frames. Li *et al.* [26] propose a game-on-demand engine that uses a prioritized content delivery scheme to transmit scene contents to a client in advance.

Most existing prefetching approaches focus on predicting the future viewpoint of a user and exploiting the spatial relationship based on the distance between the user and the objects in the scene to deliver the relevant scene contents in advance. Only a few approaches consider users' activities and personal interests. Park *et al.* [27] combine the spatial distance with a user's interests to calculate the access priority for caching and prefetching. In their scheme, the user's interest is simply defined as the number of accesses to a given object. Hung *et al.* [28] mine the correlations among 3-D objects and use the association rules thus obtained to reconstruct the placement order of 3-D objects in the storage system to improve disk I/O performance. Rahimi *et al.* [29] propose a context-aware prioritized 3-D streaming algorithm that uses a hidden Markov model (HMM) to predict a user's future activities in a 3-D game. Vani *et al.* [30] make an offline analysis on user interactions (the rates of key presses and the sequences of keys pressed) to construct a predictive model for 3-D mesh streaming.

In addition to predicting future viewpoints and analyzing users' activities and interests, user access patterns also assist in predicting the scene contents that are most likely to be visited in the future. Therefore, it is more reasonable to mine user access patterns for 3-D scene prefetching. User access pattern mining has been adopted in web navigation [31], [32] and web recommendation [33], [34] systems to discover web page based user behavior patterns. However, to date, few efficient mining methods have been proposed for the identification of user access patterns in 3-D scene navigation. In this paper, we investigate user access pattern mining and propose a novel 3-D scene prefetching scheme. Our prefetching scheme is based on the patterns mined from user access histories instead of user viewpoint traces. The use of such access patterns is demonstrated to significantly improve the hit ratio and reduce the access latency, especially in the case of a limited cache.

III. Motivation and Overview

According to the research of Song *et al.* [35], human mobility has high predictability, and therefore, the same is also likely to be true for the mobility of user avatars in 3-D scenes. When moving in a 3-D environment, many users may move along similar traces and are likely to visit popular sites in the scene, such

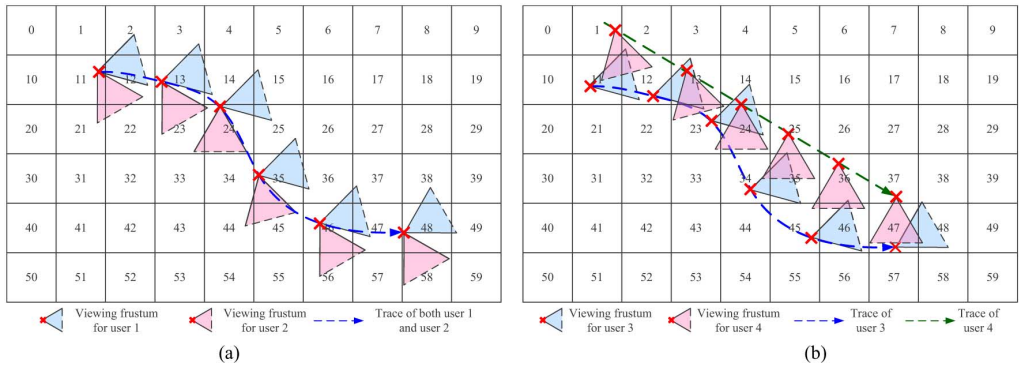


Fig. 1. Relationships between user viewpoint traces and visibility sets: (a) the same trace corresponds to different visibility sets, and (b) different traces correspond to similar visibility sets.

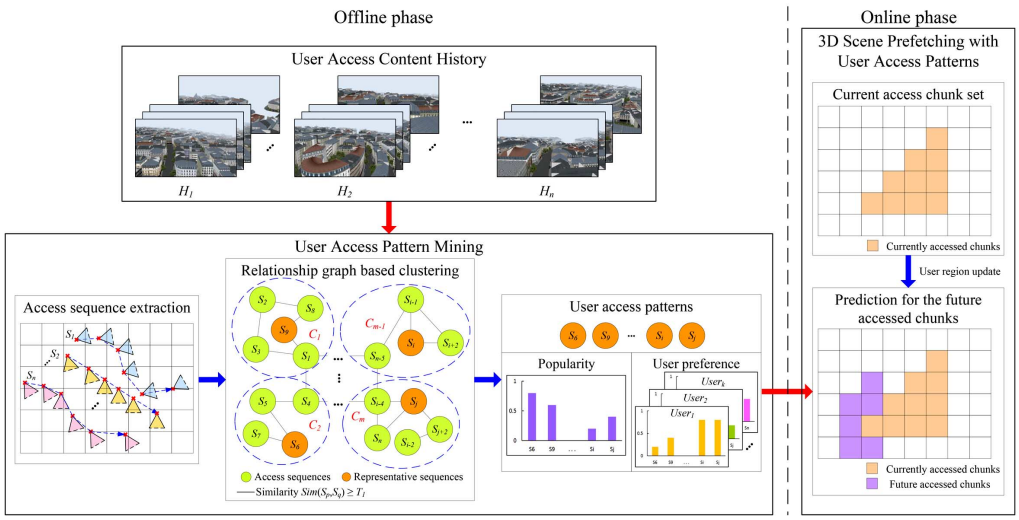


Fig. 2. Overview of the proposed user access pattern-based scene prefetching scheme. In the offline phase, access sequences are extracted from user access content histories and a relationship graph is constructed for these sequences. In the relationship graph, each node represents an access sequence and an edge (link) between two nodes indicates that the two nodes (seque S_p and S_q) have a similarity $Sim(S_p, S_q) \geq T_1$. Then, a relationship graph-based clustering analysis is applied to partition the access sequences into multiple clusters, and representative sequences from among these clusters are identified as the user access patterns. These patterns are prioritized by their popularities and users' personal preferences. In the online phase, both the access patterns and current access chunk set are used to make predictions of the future accessed chunks.

as historical and cultural sites. These user access behaviors will affect the actual scene contents to be visited. However, most existing prefetching approaches still focus on predicting the future viewpoints of users, and these viewpoints are insufficient to represent the users' access characteristics. Because of different viewing directions, speeds and fields of view, the same viewpoint trace may correspond to different visibility sets for two different users, whereas different viewpoint traces may be related to similar visibility sets, as in the case of observing a landmark from different distances. Two examples comparing user viewpoint traces and visibility sets are illustrated in Fig. 1. The 3-D scene is partitioned into $6*10$ uniform chunks. The chunks that are overlapped by a user's viewing frustum are considered to be the visible chunks.

Although users 1 and 2 move along the same trace, the visibility sets of user 1 are different from the visibility sets of user 2. By contrast, users 3 and 4 move along different traces, but their visibility sets are highly similar. We denote the visibility sets of users 3 and 4 by V_3 and V_4, respectively, and record them in time order as follows:

$$V_3 = \{\{2, 11, 12\}, \{12, 13, 23\}, \{14, 23, 24\}, \{25, 34, 35\}, \{36, 45, 46\}, \{47, 48\}\}$$
$$V_4 = \{\{1, 2, 11, 12\}, \{13, 23, 24\}, \{14, 23, 24\}, \{25, 34, 35\}, \{35, 36, 45, 46\}, \{37, 47, 48\}\}.$$

The elements of V_3 are highly similar to the elements of V_4, and they also follow the same order. The scene contents visited by the users are correlated with certain sequential patterns. These patterns, as well as user interests and the popularities of scene contents, can help to predict the chunks that are most likely to be visited in the future. Inspired by this observation, we propose a user access pattern based 3-D scene prefetching scheme. An overview of the scheme is illustrated in Fig. 2.

The proposed prefetching scheme has two components: offline and online phases. In the offline phase, access sequences

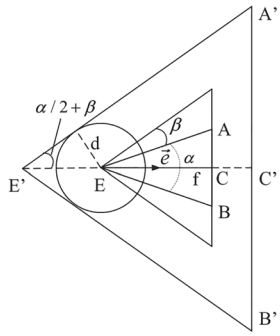


Fig. 3. Shape of the conservative region.

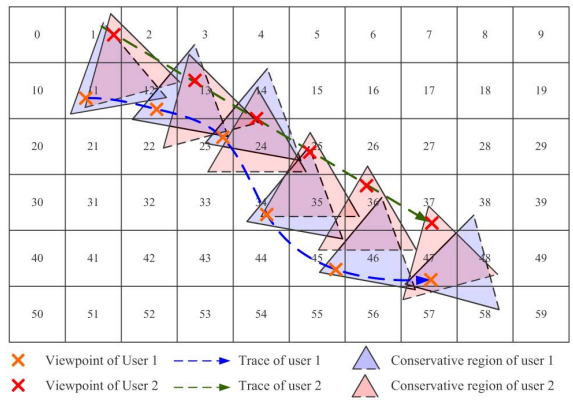


Fig. 4. Conservative regions for two users along different traces. The access sequences S_1 and S_2 of users 1 and 2, respectively, are as follows: S_1 ={{1, 2, 11 , 12}, {3, 12, 13, 22, 23}, {14, 23, 24, 25}, {25, 34, 35, 45}, {36, 45, 46, 47, 56, 57}, {38, 47, 48, 57, 58}}, S_2 ={{1, 2, 11, 12}, {3, 12, 13, 14, 22, 23, 24}, {14, 23, 24, 25}, {25, 34, 35, 36}, {36, 45, 46, 47}, {37, 47, 48, 57}}.

are extracted from user access content histories and a relationship graph is constructed, in which each node represents an access sequence and each pair of nodes (sequences) S_p and S_q with $Sim(S_p, S_q) \geq T_1$ is connected by an edge (link). Then, the access sequences are separated into several clusters through a relationship graph based clustering. Representative sequences from among these clusters are identified as the user access patterns. These access patterns are prioritized by their popularities and users' personal preferences. In the online phase, when the user region is updated, the proposed prefetching scheme uses the current access chunk set and the identified access patterns to predict the chunks that are most likely to be accessed in the future. During the idle time of the network, the predicted chunk data are delivered in advance.

IV. User Access Pattern Mining

A. Conservative Region and User Access Sequence

We define the conservative region of a user as the ε-neighborhood proposed by Wonka *et al.* [36]. The conservative region is a certain neighborhood around the viewpoint, and the data of the chunks it overlaps are all delivered to the client. The conservative region is valid for multiple frames, depending on the user's maximum movement speed and angular velocity. When the viewpoint moves beyond a distance threshold d or the viewing direction turns beyond a rotation threshold β, the conservative region requires updating, and the updated scene data are delivered to the client. The shape of the conservative region is illustrated in Fig. 3, where E is the viewpoint and $E'A'B'$ represents the conservative region of E.

As shown in Fig. 3, given a viewing frustum EAB with the field of view α and visible distance f, the conservative region of E is determined by the position E', distance $E'C'$ and field of conservative region $\alpha/2 + \beta$. The position E' and distance $E'C'$ are calculated as follows:

$$E' = E - \frac{d}{\sin(\alpha/2+\beta)}\vec{e}$$
$$|E'C'| = d + \frac{d}{\sin(\alpha/2+\beta)} + f \quad (1)$$

where $\vec{e}$ denotes the viewing direction of the viewing frustum EAB. The conservative region is a conservative approximation of the union of all viewing frustums that can be reached by a user within a moving distance threshold d and a viewing direction rotation threshold β. When the movement or rotation of the user passes beyond the thresholds, the conservative region must be updated.

When a user performs a walkthrough in a 3-D scene, we record the conservative regions and the chunks they overlap in time order. Then, we extract the useful information from the access histories in the form of access sequences. An access sequence is an ordered list of the chunks in the conservative regions, and its formal definition is as follows:

Definition 1: (access sequence) Let $\mathbb{C} = \{c_1, c_2 \ldots c_n\}$ be the set of n chunks comprising a 3-D scene. An access sequence S is defined as $S = \{s_1, s_2 \ldots s_m\}$, where each element $s_k (k = 1, 2 \ldots m)$ is a nonempty subset of $\mathbb{C}$.

By the definition of an access sequence, a chunk $c_i (i = 1, 2 \ldots n)$ can occur only once in any given element of S, but it can occur multiple times in different elements. For generality, the chunks in an element of S are arranged in lexicographical order. The access sequences can be easily acquired by recording the chunks overlapped by the conservative regions of a walkthrough and removing the continuous duplicates. These conservative regions and their corresponding access sequences are illustrated by an example in Fig. 4. The 3-D scene consists of 6∗10 chunks, and two users perform two separate walkthroughs along different traces.

As shown in Fig. 4, we calculate the chunks overlapped by the conservative regions of users 1 and 2, and obtain two access sequences S_1 and S_2, respectively. Although users 1 and 2 move along different traces and have different viewing directions, access sequences S_1 and S_2 are highly similar. To measure the access sequence similarity, we define the distance metric between two sequence elements and further construct the distance between two sequences.

Definition 2: (element distance) Let A and B be two access sequences, and let a_i and b_j denote the i-th element of A and the

j-th element of B, respectively. The element distance between a_i and b_j is defined by $d(a_i, b_j)$ as follows:

$$d(a_i, b_j) = \frac{|a_i \cup b_j| - |a_i \cap b_j|}{|a_i \cup b_j|}. \tag{2}$$

The element distance metric is also known as the Jaccard distance [37], which is used to measure the similarity between two sets. The distance between two sequence elements is calculated by normalizing the difference between the cardinality of their union and the cardinality of their intersection. Based on the element distance, we define the sequence distance metric, which measures the similarity between two access sequences. The definition of the sequence distance is inspired by the Levenshtein distance [38], which is a metric for measuring the difference between two strings. By incorporating the element distance and Levenshtein distance, we construct the distance metric between two access sequences as follows:

Definition 3: (sequence distance) Let A and B be two access sequences, and let $|A|$ and $|B|$ denote the lengths of A and B, respectively. The sequence distance between A and B is $D_{A,B}(|A|, |B|)$, which is defined recursively as

$$D_{A,B}(i,j) = \begin{cases} \max(i,j) & (\min(i,j) = 0) \\ \min \begin{cases} D_{A,B}(i-1,j) + 1 \\ D_{A,B}(i,j-1) + 1 \\ D_{A,B}(i-1,j-1) + d(a_i, b_j) \end{cases} & (\min(i,j) \neq 0) \end{cases} \tag{3}$$

where i and j denote the current lengths of A and B, respectively, and $d(a_i, b_j)$ is the distance between the i-th element of A and the j-th element of B.

The sequence distance measures the difference between two access sequences. It represents the minimum single-element changes required to modify one sequence into the other. Based on this distance, the similarity between two access sequences A and B is calculated by

$$Sim(A, B) = 1 - \frac{D_{A,B}(|A|, |B|)}{\max(|A|, |B|)}. \tag{4}$$

Equation (4) considers both the distance between two access sequences and their lengths. It measures the similarity of A and B by normalizing the sequence distance rather than using the raw distance directly. This normalization unifies the similarity measurements for access sequences of different lengths.

B. Access Sequence Clustering and Access Patterns

The access sequence clustering analysis partitions a set of access sequences $\Gamma = \{S_1, S_2, \ldots S_p\}$ into a set of sequence clusters $\Lambda = \{C_1, C_2, \ldots C_q\}$ and generates a representative sequence for each cluster C_i. Based on the access sequence similarity, similar sequences are classified into the same cluster. We calculate the similarity for each pair of sequences in Γ and construct the relationship graph for the access sequences. Each access sequence is denoted by an individual node in the relationship graph. If $Sim(S_i, S_j)$ is larger than a certain threshold T_1, then there is an edge between S_i and S_j. The relationship graph is described by an adjacency matrix M. A term m_{ij} of M is defined by

$$m_{ij} = \begin{cases} 1 & (Sim(S_i, S_j) \geq T_1, i \neq j) \\ 0 & (Sim(S_i, S_j) < T_1, i \neq j) \\ 0 & (i = j) \end{cases} \tag{5}$$

and m_{ij} relies on the similarity between sequences S_i and S_j. It denotes whether there is an edge between sequences S_i and S_j. For each term m_{ii} on the primary diagonal of M, m_{ii} is set to zero because there are no edges between sequence S_i and itself in the relationship graph.

Because the similarity relations between access sequences are described by the relationship graph, the task of access sequence clustering is transformed into relationship graph based community structure discovery. The community structure refers to groups of nodes with a high density of within-group edges and a low density of between-group edges. We choose the Newman algorithm [39] as our strategy for community structure discovery because of its effectiveness and efficiency. The Newman algorithm is a bottom-up method that aggregates the nodes of the relationship graph into tightly connected groups. This aggregation is based on a quality function or modularity Q as follows:

$$Q = \sum_i \left(e_{ii} - a_i^2\right) \tag{6}$$

where e_{ii} denotes the fraction of edges that fall within group i, e_{ij} is one-half of the fraction of edges that connect nodes in group i to nodes in group j, and $a_i = \sum_j e_{ij}$.

The Newman algorithm starts with an initial state in which each node (access sequence) is regarded as an individual community. Then, it repeatedly merges communities together in pairs by choosing the merger that results in the greatest increase (or smallest decrease) in Q. The maximal value of Q corresponds to the final community structure, which partitions the access sequences into multiple clusters. From each cluster, we choose a representative access sequence. The representative sequence of a cluster is a member of the cluster and generates the minimum average distance within the cluster. Let C be an access sequence cluster and R be the representative of C. The selection of R is defined as the identification of the sequence that satisfies the following minimization:

$$\min_{R \in C} \sum_{S \in C} \frac{w_S}{\sum_{T \in C} w_T} D_{R,S}(|R|, |S|) \tag{7}$$

in which w_S and w_T denote the frequency of access sequences S and T, respectively.

We calculate the frequency of each cluster, i.e., the sum of the frequencies of access sequences within the cluster. A cluster whose frequency exceeds a threshold T_2 is designated as a key cluster. Then, the representative sequences of the key clusters are regarded as the user access patterns. We summarize the proposed process of user access pattern discovery in Algorithm 1,

which constructs the relationship graph of the access sequences, uses the Newman algorithm to find access sequence clusters, and then discovers the user access patterns from the representative sequences of those clusters. Let n denote the number of access sequences, m denote the number of edges in the relationship graph, and k denote the number of clusters. The time complexity of the user access pattern discovery algorithm is $O(n^2+(n+m)n+k)$, where $O(n^2)$ is the complexity of the relationship graph construction, $O((n+m)n)$ is the complexity of the Newman clustering, and $O(k)$ is the complexity of the representative sequence and user access pattern selection.

ALGORITHM 1. User Access Pattern Discovery

Input: $\Gamma = \{S_1, S_2, \ldots S_p\}$: the set of access sequences.
Output: $\Upsilon = \{R_1, R_2 \ldots R_l\}$: the set of access patterns.
$\Lambda = \emptyset$ // The set of access sequence clusters
$M = [m_{ij}]_{p\times p}$ // The adjacency matrix of the relationship graph
// Build the relationship graph for the access sequences
for each access sequence S_i in Γ **do**
  for each access sequence S_j in Γ **do**
    if $Sim(S_i, S_j) \geq T_1$ and i $\neq$ j **then**
      $m_{ij} = 1$
    else
      $m_{ij} = 0$
    end
  end
end
// Newman clustering
$\Lambda =$ *NewmanClustering(M)*
// Select representative sequences
for each cluster C_k in Λ **do**
  $R_k =$ *SelectRepresentativeSequence*(C_k)
end
// Discover user access patterns
$\Upsilon = \emptyset$
for each cluster C_k in Λ **do**
  if *Frequency*$(C_k) \geq T_2$ **then**
    $\Upsilon = \Upsilon \cup R_k$
  end
end

C. User Access Pattern Priority

We leverage real world heuristics for the user access pattern based scene prefetching. First, each individual user has his or her own preference regarding the mined access patterns and tends to follow some of the patterns more frequently than others. The access patterns must be therefore prioritized by personal preference on a per-user basis. Second, users typically walk through a 3-D scene following certain specific access patterns, and as a result, some patterns are more popular than others. A high popularity means that the access pattern has been followed by many users. Considering both user individuality and commonality, we incorporate both users' personal preferences and pattern popularities into our access pattern priorities.

Let $\Upsilon = \{R_1, R_2 \ldots R_l\}$ be the set of access patterns acquired from user access content histories, and let N_{Ri} denote the frequency of R_i, i.e., the frequency of the cluster to which R_i belongs. The popularity value of R_i is calculated as

$$PV(R_i) = \begin{cases} \frac{N_{Ri}-N_{\min}}{N_{\max}-N_{\min}} & (N_{\max} \neq N_{\min}) \\ 0 & (N_{\max} = N_{\min}) \end{cases} \quad (8)$$

where $N_{\max} = \text{Max}\{N_{R1}, N_{R2} \ldots N_{Rl}\}$ denotes the maximum frequency among the access patterns and $N_{\min} = \text{Min}\{N_{R1}, N_{R2} \ldots N_{Rl}\}$ denotes their minimum frequency. The popularity value represents the public preference regarding the access patterns. Popularity is an important factor for 3-D scene prefetching, especially when a user's personal preference is not available, which commonly occurs when the user is a newcomer to a 3-D scene or is navigating an unfamiliar portion of the scene.

Personal preference depends on a user's individual access sequences. Different users are likely to have different personal preferences. For each individual user, we calculate his or her personal preference regarding the access patterns. Let $\Theta = \{C_{R1}, C_{R2} \ldots C_{Rl}\}$ denote the clusters to which R_i belongs, and let $N_{Ri,u}$ denote the frequency of R_i for user u, i.e., the frequency of user u's access sequences in cluster C_{Ri}. The personal preference value of R_i for user u is calculated by

$$PP(R_i, u) = \begin{cases} \frac{N_{Ri,u}-N_{\min,u}}{N_{\max,u}-N_{\min,u}} & (N_{\max,u} \neq N_{\min,u}) \\ 0 & (N_{\max,u} = N_{\min,u}) \end{cases} \quad (9)$$

where $N_{\max,u} = \text{Max}\{N_{R1,u}, N_{R2,u} \ldots N_{Rl,u}\}$ denotes the maximum frequency among $R_1, R_2 \ldots R_l$ for user u and $N_{\min,u} = \text{Min}\{N_{R1,u}, N_{R2,u} \ldots N_{Rl,u}\}$ denotes the minimum frequency among $R_1, R_2 \ldots R_l$ for user u. The personal preference value represents the user's own interest in the access patterns. For a given user, certain access patterns typically occur more frequently than others, and therefore, the personal preference values of these patterns are higher.

Based on the popularity of access patterns and the personal preferences of users, we combine $PV(R_i)$ and $PP(R_i, u)$ to define the interest priority for each pattern. The interest priority $IP(R_i, u)$ of access pattern R_i for user u is given by

$$IP(R_i, u) = \lambda_{R_i,u} \cdot PP(R_i, u) + (1 - \lambda_{R_i,u}) \cdot PV(R_i) \quad (10)$$

where $\lambda_{Ri,u}$ indicates which factor (popularity or personal preference) contributes more to the priority.

The determination of $\lambda_{Ri,u}$ is related to the correlation between popularity and personal preference. When personal preference is insufficient, the popularity value dominates the interest priority because we must estimate the user's interest based on the public preference. Conversely, the value of $\lambda_{Ri,u}$ is set higher when personal preference sufficiently reflects the user's

interest. Accordingly, the value of $\lambda_{Ri,u}$ for access pattern R_i and user u is defined by

$$\lambda_{R_i,u} = \frac{N_{R_i,u}}{N_{R_i,\max}} \tag{11}$$

where $N_{Ri,u}$ denotes the frequency of R_i for user u and $N_{Ri,max}$ denotes the maximum frequency of R_i for all individual users. Equation (11) indicates that, if user u follows access pattern R_i more frequently, then the personal preference of u accounts for more of the interest priority $IP(R_i, u)$.

V. 3-D Scene Prefetching and Caching

A. 3-D Scene Prefetching With User Access Patterns

Based on the identified access patterns, the proposed online prefetching algorithm uses the current access chunk set, i.e., the set of chunks in the current conservative region, to predict the chunks that are most likely to be visited in the future. We prioritize these predicted chunks according to the interest priority values of their corresponding access patterns. Then, the chunks with high priorities are first delivered to the client when the network is idle.

According to the current access chunk set A, our prefetching algorithm dynamically selects the matched patterns from user access pattern set $\Upsilon = \{R_1, R_2 \dots R_l\}$. We choose the minimum distance between A and the elements of R_i as the metric to measure the matching degree between R_i and A. The matching distance $Md(R_i, A)$ is calculated as

$$Md(R_i, A) = \min_{r_{ij} \in R_i} \frac{|r_{ij} \cup A| - |r_{ij} \cap A|}{|r_{ij} \cup A|} \tag{12}$$

where r_{ij} is the j-th element of R_i and A is the set of chunks in the current conservative region. If $Md(R_i, A)$ is less than a certain threshold T_3, then R_i is regarded as a matched pattern, and the element that corresponds to the minimum distance is considered to be the matched element of R_i.

Then, our prefetching algorithm prioritizes the matched patterns by their interest priority values $IP(R_i, u)$. Let $M_1, M_2 \dots M_w$ be the prioritized matched patterns, and let $m_{1,j_1}, m_{2,j_2} \dots m_{w,j_w}$ denote the corresponding matched elements. The successors $m_{1,j_1+1}, m_{2,j_2+1} \dots m_{w,j_w+1}$ of the matched elements represent the scene contents that are most likely to be visited in the future. To avoid delivering duplicate data, the chunks to be prefetched include only the chunks that belong to the successor elements but do not reside in the client cache. We use a priority queue to guarantee that the chunks in the high-priority successor elements are prefetched to the client first. In the case that none of the access patterns satisfies the requirement of a matching distance of less than T_3, our prefetching algorithm employs the dead-reckoning based approach proposed by Zheng *et al.* [25] as an alternative strategy to predict the future conservative region and the chunks to be prefetched. The details of our online prefetching algorithm are described in Algorithm 2. Let n denote the number of access patterns and m denote the number of matched patterns. Our prefetching algorithm runs in $O(n + m\log m + m)$ time, where the determination of the matched access patterns requires $O(n)$ time, the sorting of the matched access patterns requires $O(m\log m)$ time, and the prediction of the future accessed chunks requires $O(m)$ time.

ALGORITHM 2. User Access Pattern Based Prefetching

Input: $\Upsilon = \{R_1, R_2 \dots R_l\}$: the access patterns,
A: the current access chunk set.
Output: $Q_{prefetch}$: the priority queue of the chunks to be prefetched.

```
Υ_match = ∅
// Determine the matched access patterns
for each pattern R_i in Υ do
    if Md(R_i, A) ≤ T_3 then
        Υ_match = Υ_match ∪ {R_i}
    end
end
if Υ_match ≠ ∅ then
    // Sort in descending order
    SortByInterestPriority(Υ_match)
    // Predict the future accessed chunks
    while Υ_match ! = ∅ do
        M_1st = the first pattern in Υ_match
        m_1st,j_1st = MatchedElement(M_1st)
        m_1st,j_1st+1 = SuccessorElement(m_1st,j_1st)
        for each chunk c_i in m_1st,j_1st+1 do
            if ! InClientCache(c_i) and c_i ∉ Q_prefetch then
                Pushback(c_i, Q_prefetch)
            end
        end
        Υ_match = Υ_match − M_1st
    end
else
    Q_prefetch = PredictbyDeadReckoning()
end
```

Our prefetching algorithm uses the user access patterns to predict the chunks that are most likely to be visited by the client in the future. The predicted chunks are placed into the priority queue according to the interest priority values of their corresponding patterns. When the network is idle, the server delivers the chunks with the highest priorities to the client first. Suppose that the client cache can contain a maximum of n_{cache} chunks, of which n_{region} are occupied by the current conservative region. Then, during the idle time of the network, the server will deliver at most $n_{cache} - n_{region}$ chunks in the priority queue to the client.

B. 3-D Scene Cache Management

The conservative region update and user access pattern based prefetching will cause the server to deliver chunk data to the client. When the client cache cannot provide sufficient free space for the newly delivered chunks, the client must remove and replace certain older chunks in its cache following some cache replacement policy. The most straightforward approach is the least recently used (LRU) policy. In the LRU policy, the chunks in the cache are prioritized by their last access time. If a chunk has not been accessed for a long time, then that chunk

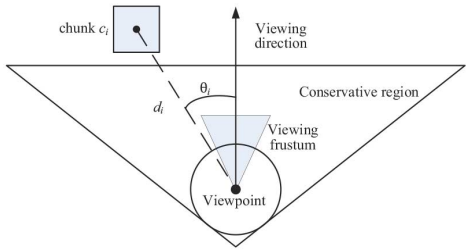


Fig. 5. Distance and angle factors of the cache priority.

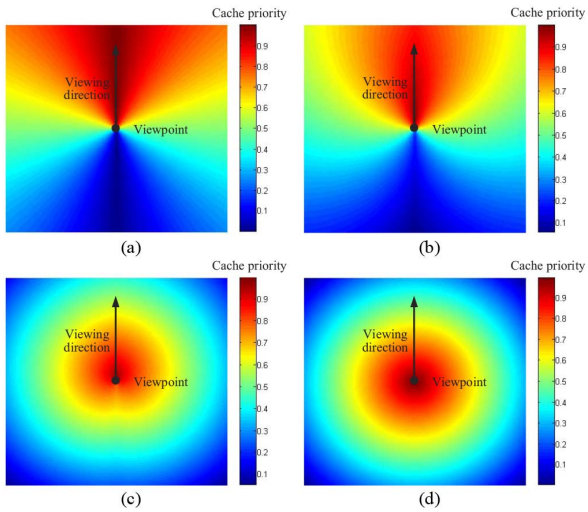


Fig. 6. Distributions of the cache priority: (a) $\omega = 0$, (b) $\omega = 0.2$, (c) $\omega = 0.8$, and (d) $\omega = 1.0$.

will have a lower priority and will thus be more likely to be replaced. However, it has been shown that the LRU policy is not suitable for 3-D scene cache replacement [40] because the 3-D objects accessed by a client may change frequently over time. 3-D scene cache replacement must consider the distance of an object and the viewing direction rather than simply its access time. A larger distance between a chunk and the viewpoint leads to a lower access probability. Similarly, a larger angle between a chunk and the viewing direction also leads to a lower access probability. As Fig. 5 shows, the cache priority of a chunk c_i is determined by two factors: (1) the distance d_i between the chunk and the viewpoint of the conservative region and (2) the angular deviation $\theta_i (0 \leq \theta_i \leq \pi)$ from the chunk to the viewing direction of the conservative region. We define the cache priority of chunk c_i as

$$CP(c_i) = \omega \cdot \left(1 - \frac{d_i}{d_{\max}}\right) + (1-\omega) \cdot \left(1 - \frac{\theta_i}{\pi}\right) \quad (13)$$

where $d_{\max}$ is the maximum distance of chunks in the cache and $0 \leq \omega \leq 1$ is the weighting coefficient.

The weighting coefficient indicates which factor (distance or angular deviation) contributes more to the cache priority. Let us consider the impact of ω on the cache priority, and Fig. 6 shows the priority distributions for $\omega = 0$, $\omega = 0.2$, $\omega = 0.8$ and $\omega = 1.0$. The cache priority with $\omega = 0$ [Fig. 6(a)] is related only to the angle factor, whereas the cache priority with $\omega = 1.0$ [Fig. 6(d)] considers only the distance factor.

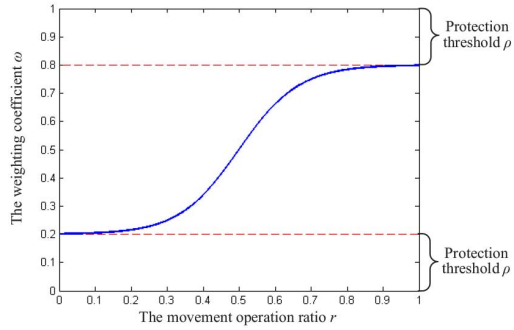


Fig. 7. Weighting coefficient ω versus movement operation ratio r.

TABLE I
PARAMETER SETTINGS FOR THE TEST SCENES

Parameters	Test Scenes		
	Paris	Palace	Venice
Scene Dimensions (units)	2500*2000	1500*3000	2500*4000
Total Vertices	704,957	144,870	4,050,072
Total Faces	712,119	230,206	5,099,408
Chunk Size (units)	100*100	100*100	100*100
Total Chunks	500	450	1000
Visible Distance f (units)	600	600	600
Field of View α (degrees)	60	60	60
Distance Threshold d (units)	60	60	60
Rotation Threshold β (degrees)	±15	±15	±15

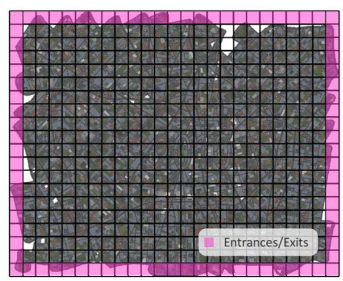


Fig. 8. Entrances/exits (boundary chunks) of the Paris Scene.

Fig. 6 illustrates that ω significantly affects the distribution of the cache priority. More importantly, when viewing frustum rotation is more frequent than viewpoint movement, ω should be set to a low value, and vice versa. Unlike the previous works [22], [40], which set ω to a fixed value, we dynamically adjust ω depending on the frequencies of viewing frustum rotation and viewpoint movement. Between two conservative region updates, the client counts the number of user viewing frustum rotation operations f_R and the number of user viewpoint movement operations f_T. The weighting coefficient ω is related to the movement operation ratio $r = f_T/(f_T + f_R)$. Inspired by the S-shaped function [41], which exhibits the highest sensitivity to changes around the average value, we define ω as

$$\omega = \frac{1-2\rho}{1+e^{-(r-0.5)*\sigma}} + \rho \quad (14)$$

where σ is the scale factor of r and ρ is the protection threshold of ω. Because of the nature of the standard logistic function $1/(1+e^{-x})$, its value is sufficiently close to 0 when $x = -6$ and to 1 when $x = 6$, and therefore, we set σ to 12, i.e., $\sigma = 6 + 6$. Fig. 7 shows the curve of the weighting coefficient ω versus the movement operation ratio r : ω grows slowly when r is

TABLE II
PERFORMANCE OF OUR PREFETCHING METHOD UNDER VARIOUS MATCHING DISTANCE THRESHOLDS IN THE PARIS SCENE

Matching Distance Threshold (T_3)	Cache Capacity (chunks)											
	90		120		150		180		210		240	
	Hit Ratio	Average Prefetched Chunks	Hit Ratio	Average Prefetched Chunks	Hit Ratio	Average Prefetched Chunks	Hit Ratio	Average Prefetched Chunks	Hit Ratio	Average Prefetched Chunks	Hit Ratio	Average Prefetched Chunks
0.2	85.7%	1.6	88.1%	1.6	89.2%	1.6	89.6%	1.5	90.1%	1.5	90.2%	1.5
0.35	88.8%	3	92.3%	2.6	92.5%	2.5	92.7%	2.4	92.7%	2.3	93.1%	2.3
0.5	91.4%	5.1	95.4%	5.8	95.5%	5.2	95.8%	5	95.8%	4.9	96%	4.8
0.65	91.6%	5.3	96%	8.6	96%	8.3	96.1%	7.9	96.5%	7.8	96.7%	7.8
0.8	91.9%	5.4	96.2%	8.7	96.5%	9.9	96.7%	9.2	97%	9.1	97.1%	9

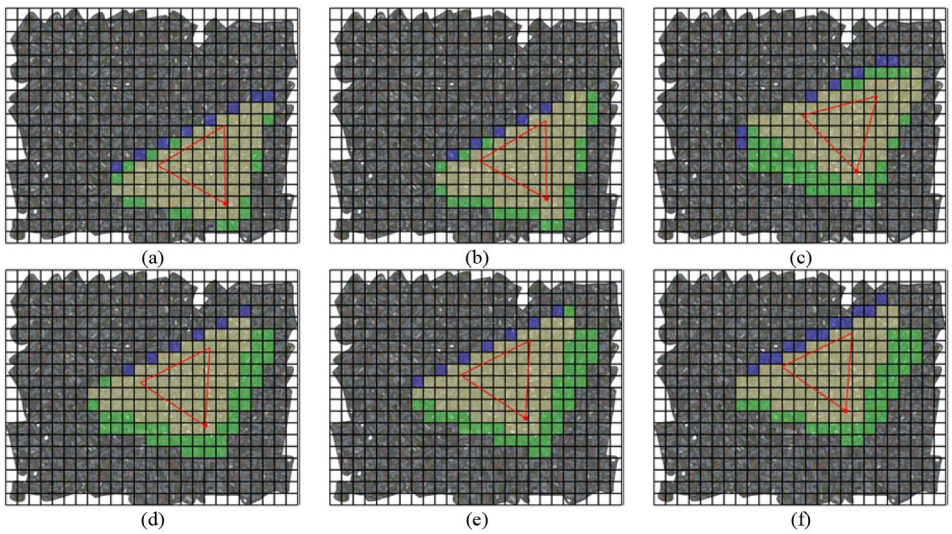


Fig. 9. Prefetching results of the user access pattern-based scene prefetching method: (a) the 124th frame, (b) the 167th frame, (c) the 488th frame, (d) the 575th frame, (e) the 658th frame, and (f) the 695th frame (blue: the prefetched chunks; yellow: the chunks in the current conservative region; green: the chunks in the client cache).

relatively small or large, and the value of ρ limits the range of ω, i.e., $\omega \in (\rho, 1-\rho)$. According to the research of Song *et al.* [35], the lower bound on the predictability of user mobility is 80%. Thus, even when a user continually performs one operation, i.e., $r = 0$ or $r = 1$, there is only an 80% certainty that the user will continue the same operation. Considering the uncertainty of users' future operations, we set the protection threshold ρ to 0.2, i.e., $\rho = 1 - 0.8$.

The client treats a chunk as a single cache unit and manages its cache according to the cache priority as defined above. When the client cache cannot provide a sufficient amount of free space for the newly delivered chunks (arriving as a result of the conservative region update and user access pattern based prefetching), some low-priority chunks inside the cache are removed and replaced. Among the cached chunks, the client first selects chunks that are outside the current conservative region as candidates for replacement. Then, the candidates are sorted by their cache priorities, and those with the lowest priority are removed and replaced by the newly delivered chunks. We summarize the client cache replacement policy and present its pseudo code in Algorithm 3. The selection of the candidate chunks for replacement requires $\mathrm{O}(n)$ time, where n is the number of cached chunks, and the sorting of the candidates runs in $\mathrm{O}(m\log m)$ time, where m is the number of candidates. Thus, the time complexity of the cache replacement policy is $\mathrm{O}(n + m\log m)$.

ALGORITHM 3. Cache Replacement

Input: $C_{deliver} = \{d_1, d_2 \ldots d_{|C_{deliver}|}\}$: the newly delivered chunks,
$C_{cache} = \{c_1, c_2 \ldots c_{|C_{cache}|}\}$: the chunks in the cache,
S: the cache size of the client.
Output: $C_{replace}$: the chunks to be replaced.
$C_{replace} = \emptyset$
if $|C_{deliver}| > S - |C_{cache}|$ **then**
  replace_num $= |C_{deliver}| - (S - |C_{cache}|)$
  // Select the candidate chunks for replacement
  $C_{candidate} = \emptyset$
  for each chunk c_i in C_{cache} **do**
    if ! *ConservativeRegionCheck* (c_i) **then**
      $C_{candidate} = C_{candidate} \cup c_i$
    end
  end
  // Sort in descending order
  SortByCachePriority($C_{candidate}$)
  $C_{replace}$ = *LastChunks*($C_{candidate}$, *replace_num*) // The last *replace_num* chunks
    $C_{cache} = C_{cache} - C_{replace}$
else
    return // There are no chunks to be replaced.
end

TABLE III
HIT RATIO COMPARISON IN THE PARIS SCENE

Cache Capacity (chunks)	Methods			
	No Prefetch	Ours	EWMA	DR
90	83.6%	91.4%	80.4%	85.4%
120	86%	95.4%	88.4%	88.9%
150	87%	95.5%	92.1%	89.7%
180	87.5%	95.8%	93.8%	90.6%
210	88%	95.8%	94.5%	91%
240	88%	96%	95.2%	91.3%

TABLE IV
HIT RATIO COMPARISON IN THE PALACE SCENE

Cache Capacity (chunks)	Methods			
	No Prefetch	Ours	EWMA	DR
90	84%	89.4%	80.9%	86.5%
120	86.1%	94.7%	86.3%	89.3%
150	86.8%	95%	90.3%	90.9%
180	87.3%	95.1%	92.5%	91.2%
210	87.5%	95.2%	94.4%	91.4%
240	88%	95.2%	95%	91.7%

TABLE V
HIT RATIO COMPARISON IN THE VENICE SCENE

Cache Capacity (chunks)	Methods			
	No Prefetch	Ours	EWMA	DR
90	81%	89.3%	76.3%	83.1%
120	83.3%	94%	81.8%	87%
150	84.2%	94.3%	87.2%	88.6%
180	85.3%	94.6%	91.9%	89.3%
210	85.3%	94.6%	93.3%	90%
240	85.4%	94.7%	94.2%	90.4%

TABLE VI
ACCESS LATENCY COMPARISON IN THE PARIS SCENE (UNIT: MS)

Cache Capacity (chunks)	Methods			
	No Prefetch	Ours	EWMA	DR
90	48.2	29.6	66.2	43.84
120	46.3	8.2	32.2	36.7
150	46.1	8	17.3	35.52
180	45.6	8	12.9	34.29
210	45	7.9	10.5	33.75
240	44.2	7.8	9.2	33.04

TABLE VII
ACCESS LATENCY COMPARISON IN THE PALACE SCENE (UNIT: MS)

Cache Capacity (chunks)	Methods			
	No Prefetch	Ours	EWMA	DR
90	8.35	4.45	15.7	7.04
120	8.26	2.89	9.3	6.35
150	8.17	2.83	5.5	5.5
180	8.17	2.77	2.8	5.3
210	8.13	2.46	2.5	5.2
240	8.12	2.4	2.4	5.21

TABLE VIII
ACCESS LATENCY COMPARISON IN THE VENICE SCENE (UNIT: MS)

Cache Capacity (chunks)	Methods			
	No Prefetch	Ours	EWMA	DR
90	342.3	179	592	295.4
120	303	77.5	371.5	235.8
150	270.3	73.4	230.3	195
180	252.5	69.6	137.6	180.7
210	251.5	67.5	100.2	172.1
240	251	65.1	77.7	168.2

VI. EXPERIMENT EVALUATION

A. Experimental Scenarios and Parameter Settings

We have implemented a remote walkthrough prototype system to evaluate the performance of the proposed user access pattern based scene prefetching scheme. It allows multiple users to navigate a 3-D scene using a keyboard and mouse. Using the client program, a user connects to the server to obtain an initial scene package, which contains the geometry and texture data of chunks in that user's conservative region. After receiving the package, the user can navigate the 3-D scene. The client requests additional chunks from the server when the user conservative region is updated, i.e., when the user moves beyond the distance threshold or turns beyond the angle threshold. The server employs the proposed prefetching method to predict the chunks that will be visited in the future. During the idle time of the network, the data of the predicted chunks are delivered to the client.

In our experiment, the server runs on a workstation with an Intel Xeon CPU at 2.0 GHz, 4 GB of memory and a 100 Mbps Ethernet connection to the university network. The client runs on a personal computer with a DualCore Intel Core i3 330UM CPU at 1.2 GHz, 2 GB of memory and an ATI Mobility Radeon HD 540v graphics card. Three test scenes are used, including two city scenes (the Paris Scene and Venice Scene) and one historic site scene (the Palace Scene). Each test scene is partitioned into uniform chunks of 100∗100 units in size. The detailed parameters of each test scene are provided in Table I.

For each test scene, all boundary chunks can be chosen as the entrances/exits of the scene. Fig. 8 illustrates the entrances/exits of the Paris Scene. To collect user access sequences, we invite 50 volunteers to perform walkthroughs as users. In the experiment, each volunteer can freely choose one boundary chunk as his or her starting position. Then, the volunteer performs a walkthrough inside the scene and is allowed to freely move to another boundary chunk to complete the walkthrough. During the walkthrough, the navigation route, moving speed and viewing direction are freely controlled by the volunteer. In each of the three scenes, each volunteer performs 50 walkthroughs. Then, we mine these walkthroughs for user access patterns and calculate the corresponding interest priority values. Inspired by research on DNA and protein sequence analysis [42], [43], [44], in which a value of 0.8 is typically used as an empirical cut-off value for sequence similarity, we set the similarity threshold of our user access pattern discovery algorithm as 0.8 (i.e., $T_1 = 0.8$). The cluster frequency threshold T_2 depends on $Max\{\overline{N_U}, \overline{N_C}\}$, where $\overline{N_U}$ denotes the average number of access sequences per user and $\overline{N_C}$ denote the average number of

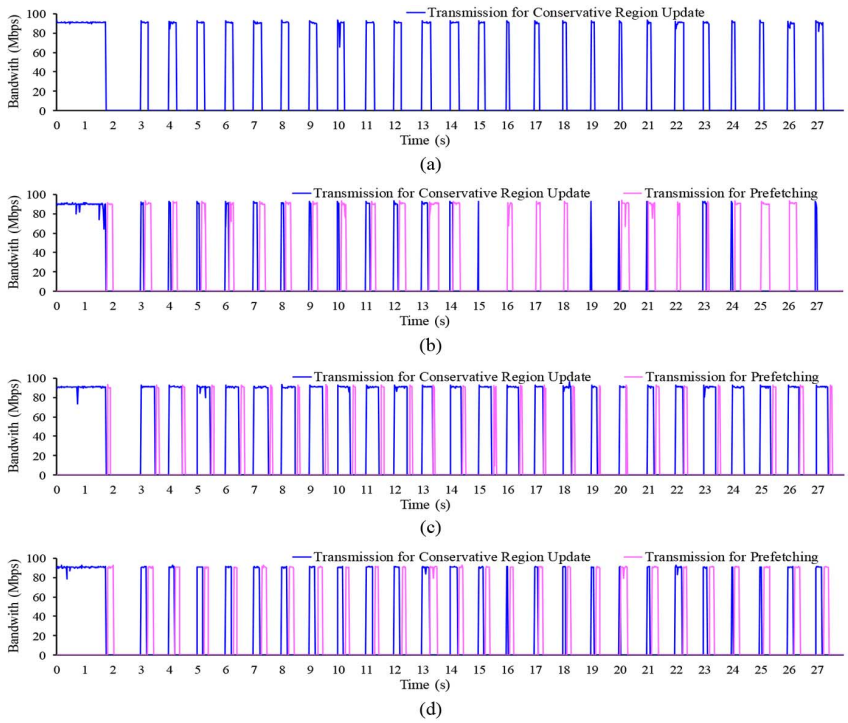


Fig. 10. Bandwidth usage comparison for the walkthrough in the Venice Scene (cache capacity = 120 chunks): (a) no prefetch, (b) our method, (c) EWMA, and (d) DR.

access sequences per cluster. Accordingly, in our experiment, T_2 is set to 50 for the three test scenes.

To evaluate the effectiveness of the proposed scene prefetching method, we invite another 50 volunteers with no knowledge of the test scenes to perform 10 walkthroughs in each of the three scenes. We choose the hit ratio as the metric for performance evaluation and calculate this metric each time the client's conservative region is updated. The hit ratio measures the percentage of required chunks that can be retrieved from the client cache. It is the ratio of the number of chunks that are available in the client cache to the total number of chunks in the conservative region. In this experiment, the hit ratio is calculated as $N_a/(N_a + N_d)$, where N_a denotes the number of chunks that are available in the client cache and N_d denotes the number of chunks that are required to be downloaded from the server.

We test the hit ratio performance of the proposed prefetching method under various matching distance thresholds (i.e., $T_3 = 0.2, 0.35, 0.5, 0.65$ and 0.8). We also calculate the average number of chunks that are prefetched between two consecutive conservative region updates. Table II shows the hit ratios and average prefetched chunk numbers for the Paris Scene. The initial cache capacity depends on the number of chunks overlapped by the conservative region. As some chunks may not be completely covered by the conservative region, we expand the boundary of the conservative region outward by one chunk width and then calculate the initial cache capacity as A_{extend}/A_{chunk}, where A_{extend} is the area of the expanded conservative region and A_{chunk} is the area of one chunk. The cache capacity is initialized at 90 chunks and then grows to 120 (increased by 33%), 150 (66%), 180 (100%), 210 (133%) and 240 (166%) chunks.

As Table II shows, the average number of prefetched chunks increases with increasing T_3, and the hit ratio increases rapidly when T_3 grows from 0.2 to 0.5. However, when T_3 is larger than 0.5, further increase in the hit ratio with increasing T_3 is very slow. T_3 affects the number of matched access patterns and thus affects the chunks to be prefetched and the hit ratio of the proposed method. Based on the trade-off between the hit ratio and the prefetched chunk data, we choose 0.5 as the matching distance threshold (T_3) for the following experiments. Fig. 9 shows the prefetching results for a volunteer's walkthrough in the Paris Scene. Our prefetching method uses the access patterns and current access chunk set to predict the future visited chunks. The chunks prefetched by our method are then visited in subsequent frames.

B. Comparison Results

We compare the proposed user access pattern based scene prefetching method with three other methods, including no prefetching (No Prefetch), exponentially weighted moving average based prefetching (EWMA) and dead-reckoning based prefetching (DR). No Prefetch serve as a base case for the performance comparison. In the No Prefetch scheme, the server delivers chunk data to the client only when the conservative region is updated. EWMA prefetching is a prevailing AOI derived method adopted by various classic and state-of-the-art virtual environment systems [15], [16], [22], [23]. It employs an exponentially weighted moving average to predict the next position of a user. DR prefetching is proposed by Zheng

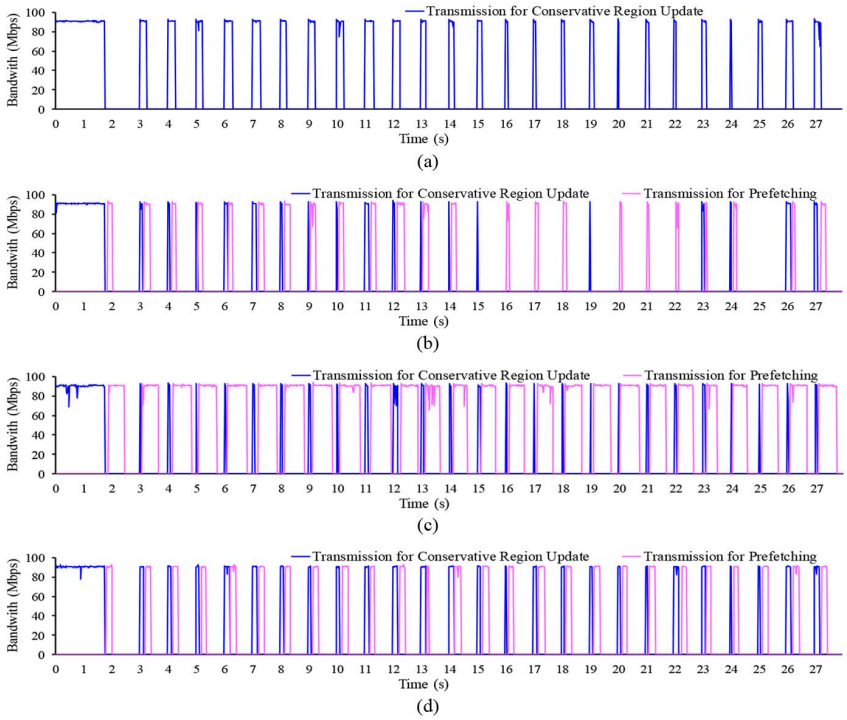


Fig. 11. Bandwidth usage comparison for the walkthrough in the Venice Scene (cache capacity = 240 chunks): (a) No Prefetch, (b) Our Method, (c) EWMA and (d) DR.

TABLE IX
PERCENTAGES OF USER ACCESS PATTERN-BBASED PREDICTIONS VERSUS DEAD-RECKONING-BASED PREDICTIONS FOR OUR PREFETCHING METHOD

Test Scenes	Types of Predictions	
	User Access Pattern based Prediction	Dead-Reckoning based Prediction
Paris Scene	89.3%	10.7%
Palace Scene	90.8%	9.2%
Venice Scene	88.6%	11.4%
Average	89.6%	10.4%

et al. [25] for 3-D data pre-transmission. It uses a first-order dead-reckoning algorithm to predict future viewpoints and viewing directions. Dead-reckoning is critically important in distributed virtual environments and is recommended by IEEE 1278.1 (Distributed Interactive Simulation) [45]. In our experiment, DR is used to predict both the future conservative region and the chunks to be prefetched.

We test the hit ratio performance of the four methods for various cache capacities. As shown in Table III (Paris Scene), Table IV (Palace Scene) and Table V (Venice Scene) show, our prefetching method outperforms the other three methods. When the cache capacity is increased from 90 to 120 chunks, the hit ratio of our method increases considerably, because a larger cache can accommodate more prefetched chunks. When the cache capacity is greater than 120 chunks, the hit ratio of our method is nearly constant, because the cache capacity is sufficient to hold all of the prefetched chunks. Compared with No Prefetch, the hit ratio of our method is approximately 8.5% higher on average. This improvement can be attributed to the use of user access patterns, which enables effective predictions of the chunks that are most likely to be visited in the future.

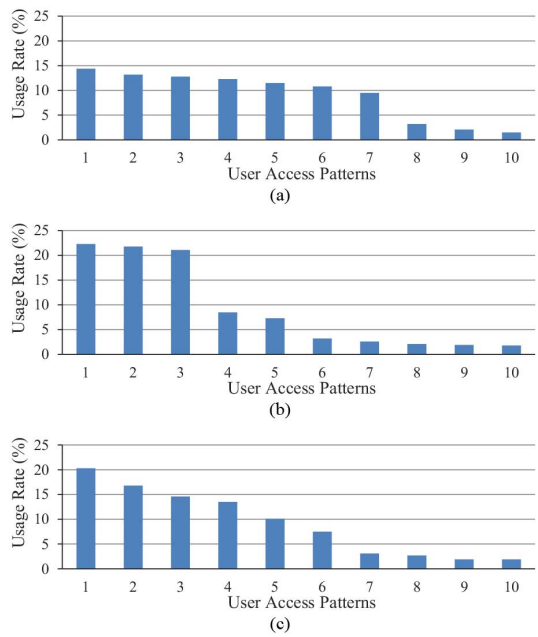


Fig. 12. Usage rates of the top 10 access patterns: (a) Paris scene, (b) Palace scene, and (c) Venice scene.

The DR method also makes a prediction of the future visited chunks, but its hit ratio is clearly lower than our method. DR considers only the previous positions and directions of the conservative region. It neglects the patterns inherent in the history of scene content sequences accessed by various users. As a result, its prediction accuracy and hit ratio inevitably degrade. The hit ratio of EWMA decreases considerably as the cache capacity

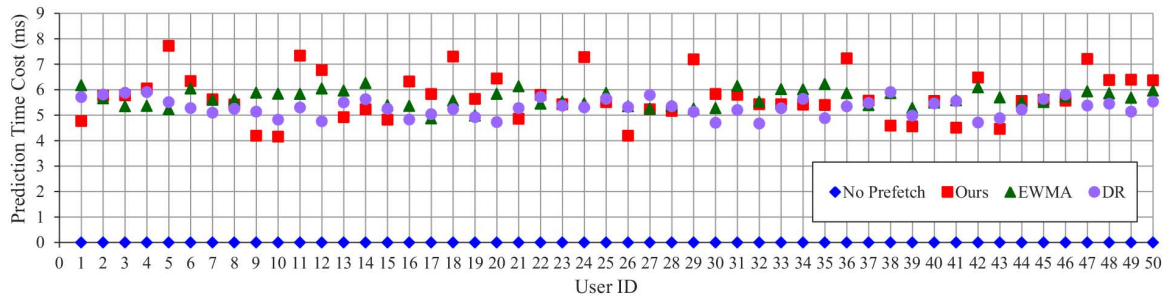


Fig. 13. Prediction time cost for each user in the Paris Scene.

is decreased from 240 to 90 chunks. The cache capacity significantly affects the hit ratio of EWMA. EWMA is an AOI derived method that prefetches chunks around the predicted viewpoint. When the cache capacity is increased, more chunks around the predicted viewpoint can be prefetched, and therefore, the hit ratio of EWMA increases. For the hit ratio to reach approximately 95%, EWMA requires nearly twice the cache capacity required by our method.

We also test the access latency performance of each method. Access latency is the latency between the instant of time when the client requests the chunks in its updated conservative region and the instant of time when the client has received all requested chunks. Only when all chunks in the current conservative region are available in the client cache, the user can perform a meaningful walkthrough inside the scene. Tables VI, VII, and VIII show the experiment results for the Paris Scene, Palace Scene and Venice Scene, respectively. The access latency is strongly correlated with the hit ratio. A higher hit ratio results in fewer chunks being transmitted and thus lowers the access latency. Our method significantly reduces the access latency compared with No Prefetch. In particular, when the cache capacity is larger than 120 chunks, the access latency is reduced by nearly 70-80% on average. Our method also demonstrates superior access latency performance to DR. Through comparison with the DR results, the effectiveness of exploiting user access patterns can be clearly observed. The access latency of EWMA becomes close to the access latency of our method as the cache capacity increases. A larger cache allows EWMA to prefetch more chunks around the predicted viewpoint. As a result, the hit ratio of EWMA increases and its access latency is reduced.

For further comparison, we measure the bandwidth usages of the four methods. In Fig. 10 (cache capacity $=$ 120 chunks) and Fig. 11 (cache capacity $=$ 240 chunks), a walkthrough in the Venice Scene is used as an example for comparison. As shown in Figs. 10 and 11, a finite startup time is required to download the chunks in the initial conservative region. After the startup time (nearly 2 s), the bandwidth usages of the four methods are quite different. Compared with No Prefetch, the chunk data transmission required for updating conservative regions is effectively reduced by our method. The primary reason for this reduction is that the data transmission of our method is amortized during the idle time of the network, and the access latency is hidden in the chunk prefetching. The DR method also prefetches some chunk data to the client when the network is idle. However, DR transmits more chunks during the updating of conservative regions than does our method. The EWMA method clearly reduces the chunk data transmission required for conservative region updates when the cache capacity is 240 chunks. However, as Fig. 11(c) shows, EWMA achieves this reduction at the cost of an extremely large amount of data prefetching, which is why EWMA requires a large cache to maintain both a high hit ratio and a low access latency.

TABLE X
AVERAGE PREDICTION TIME COST COMPARISON (UNIT: MS)

Test Scenes	Methods			
	No Prefetch	Ours	EWMA	DR
Paris Scene	0	5.7	5.6	5.3
Palace Scene	0	4.3	4.8	4.4
Venice Scene	0	7.6	7.9	7.4

Additionally, for our prefetching method, we count the number of predictions using the user access patterns and the remaining number of predictions employing the dead-reckoning based approach. Table IX shows the percentages of these two types of predictions. On average, the percentage of user access pattern based predictions is 89.6%, considerably higher than the corresponding percentage of dead-reckoning based predictions (10.4%). In most cases, our prefetching method uses the user access patterns to predict the chunks that will be visited in the future. Therefore, its improvement in prefetching performance can be predominantly attributed to the use of user access pattern based predictions. We also count the prediction incidence for each access pattern, i.e., the number of predictions which use that access pattern. Then, we calculate the usage rate for each access pattern as its prediction incidence divided by the sum of the prediction incidences of all access patterns. We sort the access patterns by their usage rates, and Fig. 12 shows the usage rates for the top 10 access patterns for the three test scenes. We observe that the sums of the usage rates of the first 7 access patterns for the Paris Scene, the first 5 patterns for the Palace Scene and the first 6 patterns for the Venice Scene exceed 80%. Clearly, certain user access patterns have considerable impact on the predictions.

To evaluate the prediction overhead, we investigate the prediction time cost of our prefetching method, i.e., the average computation time required to make a prediction. We also compare our method with No Prefetch, EWMA and DR. Fig. 13 shows the prediction time costs for each user in the Paris Scene. No Prefetch does not make any predictions of the

chunks to be visited in the future and therefore its prediction time cost is always 0. The prediction time costs of EWMA and DR are relatively stable. However, the prediction time cost of our method varies from 4.1 ms to 7.7 ms. The primary reason for this variation is that different users have different preferences and are interested in different scene contents. User preferences and interests affect the numbers of matched patterns and, consequently, the time required to make predictions. Based on the time costs of the 50 users who participate in this study, we calculate the average prediction time cost for the Paris Scene, and we obtain the results for the Palace Scene and Venice Scene in the same manner. As Table X shows, the average prediction time cost of our method is very close to EWMA and DR. From the results presented above, we observe that our prefetching method demonstrates time performance comparable to EWMA and DR.

VII. Conclusion

Although different users follow different paths with different speeds, viewing directions and fields of view when moving in a 3-D virtual environment, the scene contents accessed by users may follow specific patterns. These access patterns, together with user interests and scene content popularities, will affect the actual data to be visited. Applying these patterns to 3-D scene prefetching is a promising technique for effective scene content prediction.

In this paper, we propose a user access pattern based 3-D scene prefetching scheme, which includes both offline and online phases. In the offline phase, history user access sequences are partitioned into several clusters using the Newman algorithm. Then, user access patterns are selected from among the representative sequences of the clusters. These access patterns are prioritized by their popularities and users' personal preferences. In the online phase, instead of user viewpoint traces, the identified access patterns and the current access chunk set are used to predict the scene contents that are most likely to be visited in the future. When the network is idle, the predicted scene data are delivered to the client in advance.

We conduct comprehensive and comparative experiments to evaluate the performance of our user access pattern based scene prefetching scheme. The experiment results demonstrate that our prefetching method can effectively predict the scene contents to be visited in the future. Our method achieves a significantly improvement in the hit ratio and considerably reduces the access latency. Furthermore, our method outperforms the prevailing prefetching schemes (EWMA and DR), especially in the case of a limited cache.

References

[1] C. Carlsson and O. Hagsand, "DIVE A multi-user virtual reality system," in *Proc. IEEE Virtual Reality Annu. Int. Symp.*, Sep. 1993, pp. 394–400.

[2] J. Calvin *et al.*, "The SIMNET virtual world architecture," in *Proc. IEEE Virtual Reality Annu. Int. Symp.*, Sep. 1993, pp. 450–455.

[3] J. Leigh, A. E. Johnson, C. A. Vasilakis, T. A. Defanti, and R. S. Wurman, "Multi-perspective collaborative design in persistent networked virtual environments," in *Proc. IEEE Virtual Reality Annu. Int. Symp.*, Mar.–Apr. 1996, pp. 253–260.

[4] I. Pandzic, T. Capin, E. Lee, N. Thalmann, and D. Thalmann, "A flexible architecture for virtual humans in networked collaborative virtual environments," in *Proc. Eurograph.*, 1997, pp. 177–188.

[5] D. Cohen-Or, Y. L. Chrysanthou, C. T. Silva, and F. Durand, "A survey of visibility for walkthrough applications," *IEEE Trans. Vis. Comput. Graph.*, vol. 9, no. 3, pp. 412–431, Sep. 2003.

[6] N. K. Govindaraju, A. Sud, S. E. Yoon, and D. Manocha, "Interactive visibility culling in complex environments using occlusion-switches," in *Proc. ACM Interactive 3D Graph.*, 2003, pp. 103–112.

[7] J. Bittner, M. Wimmer, H. Piringer, and W. Purgathofer, "Coherent hierarchical culling: Hardware occlusion queries made useful," *Comput. Graph. Forum*, vol. 23, no. 3, pp. 615–624, Sep. 2004.

[8] M. Guthe, A. Balazs, and R. Klein, "Near optimal hierarchical culling: Performance driven use of hardware occlusion queries," in *Proc. EGSR*, 2006, pp. 207–214.

[9] O. Mattausch, J. Bittner, and M. Wimmer, "CHC++: Coherent hierarchical culling revisited," *Comput. Graph. Forum*, vol. 27, no. 2, pp. 221–230, Apr. 2008.

[10] M. R. Macedonia *et al.*, "NPSNET: A multi-player 3D virtual scenes over the internet," in *Proc. ACM Interactive 3D Graph.*, 1995, pp. 93–ff.

[11] D. Schmalstieg and M. Gervautz, "Demand-driven geometry transmission for distributed virtual environment," *Comput. Graph. Forum*, vol. 15, no. 3, pp. 421–431, Aug. 1996.

[12] J. H. P. Chim, M. Green, R. W. H. Lau, H. V. Leong, and A. Si, "On caching and prefetching of virtual objects in distributed virtual environments," in *Proc. ACM Multimedia*, 1998, pp. 171–180.

[13] F. W. B. Li, R. W. H. Lau, and D. Kilis, "GameOD: An internet based game-on-demand framework," in *Proc. ACM VRST*, 2004, pp. 129–136.

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

[15] S. Y. Hu *et al.*, "FLoD: A framework for peer-to-peer 3D streaming," in *Proc. IEEE INFOCOM*, Apr. 2008, pp. 1373–1381.

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

[17] M. Aljaafreh, H. R. Maamar, and A. Boukerche, "An efficient object discovery and selection protocol in 3D streaming-based systems over thin mobile devices," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Apr. 2013, pp. 2393–2398.

[18] E. Teler and D. Lischinski, "Streaming of complex 3D scenes for remote walkthroughs," *Comput. Graph. Forum*, vol. 20, no. 3, pp. 17–25, Sep. 2001.

[19] C. M. Ng, C. T. Nguyen, D. N. Tran, T. S. Tan, and S. W. Yeow, "Analyzing pre-fetching in large-scale visual simulation," in *Proc. IEEE Comput. Graph. Int.*, Jun. 2005, pp. 100–107.

[20] S. Jiang, B. Sajadi, A. Ihler, and M. Gopi, "Optimizing redundant-data clustering for interactive walkthrough applications," *Vis. Comput.*, vol. 30, no. 6–8, pp. 637–647, Jun. 2014.

[21] G. Varadhan and D. Manocha, "Out-of-core rendering of massive geometric environments," in *Proc. IEEE Vis.*, Nov. 2002, pp. 69–76.

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

[23] A. Chan, R. W. H. Lau, and B. Ng, "Motion prediction for caching and prefetching in mouse-driven DVE navigation," *ACM Trans. Internet Technol.*, vol. 5, no. 1, pp. 70–91, Feb. 2005.

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

[25] Z. Zheng, E. Prakash, and T. K. Y. Chan, "Interactive view-dependent rendering over networks," *IEEE Trans. Vis. Comput. Graph.*, vol. 14, no. 3, pp. 576–589, May 2008.

[26] 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," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 7, no. 3, p. 19, Aug. 2011.

[27] S. Park, D. Lee, M. Lim, and C. Yu, "Scalable data management using user-based caching and prefetching in distributed virtual environments," in *Proc. ACM VRST*, Nov. 2001, pp. 121–126.

[28] S. Hung and D. S. M. Liu, "Using prefetching to improve walkthrough latency," *Comput. Animation Virtual Worlds*, vol. 17, no. 3–4, pp. 469–478, Jul. 2006.

[29] H. Rahimi, A. A. N. Shirehjini, and S. Shirmohammadi, "Context-aware prioritized game streaming," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2011, pp. 1–6.

[30] V. Vani, R. P. Kumar, and S. Mohan, "3D mesh streaming and rendering-an approach based on predictive modeling," *Int. J. Comput. Sci. Issues*, vol. 9, no. 2, pp. 606–613, Mar. 2012.
[31] J. Borges and M. Levene M, "Evaluating variable-length markov chain models for analysis of user web navigation sessions," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 4, pp. 441–452, Apr. 2007.
[32] M. A. Awad and I. Khalil, "Prediction of user's web-browsing behavior: Application of markov model," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 42, no. 4, pp. 1131–1142, Aug. 2012.
[33] T. Iwata, K. Saito, and T. Yamada, "Modeling user behavior in recommender systems based on maximum entropy," in *Proc. ACM WWW*, 2007, pp. 1281–1282.
[34] J. Chen, X. Zhou, and Q. Jin, "Recommendation of optimized information seeking process based on the similarity of user access behavior patterns," *Personal Ubiquitous Comput.*, vol. 17, no. 8, pp. 1671–1681, Dec. 2013.
[35] C. Song, Z. Qu, N. Blumm, and A. L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, Feb. 2010.
[36] P. Wonka, M. Wimmer, and F. X. Sillion, "Instant visibility," *Comput. Graph. Forum*, vol. 20, no. 3, pp. 411–421, Sep. 2001.
[37] P. Jaccard, "The distribution of the flora in the alpine zone," *New Phytologist*, vol. 11, no. 2, pp. 37–50, Feb. 1912.
[38] V. I. Levensthein, "Binary codes capable of correcting deletions, insertions and reversals," *Soviet Physics Doklady*, vol. 10, pp. 707–710, Feb. 1966.
[39] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Phys. Rev. E*, vol. 69, no. 6, p. 066133, Jun. 2004.
[40] J. Chim, R. W. H. Lau, H. V. Leong, and A. Si, "Multi-resolution cache management in digital virtual library," in *Proc. IEEE Adv. Digital Libraries Conf.*, Apr. 1998, pp. 66–75.
[41] D. Kucharavy and R. De Guio, "Application of S-Shaped curves," in *Proc. ETRIA TRIZ Future Conf.*, Nov. 2007, pp. 81–88.
[42] R. Strohal, A. Helmberg, G. Kroemer, and R. Kofler, "MouseVk gene classification by nucleic acid sequence similarity," *Immunogenetics*, vol. 30, no. 6, pp. 475–493, Dec. 1989.
[43] F. Pazos and A. Valencia, "Similarity of phylogenetic trees as indicator of protein–protein interaction," *Protein Eng.*, vol. 14, no. 9, pp. 609–614, Sep. 2001.
[44] K. Cartharius *et al.*, "MatInspector and beyond: Promoter analysis based on transcription factor binding sites," *Bioinformatics*, vol. 21, no. 3, pp. 2933–2942, Apr. 2005.
[45] *IEEE Standard for Distributed Interactive Simulation—Application Protocols*, IEEE Std. 1278.1, 2012.

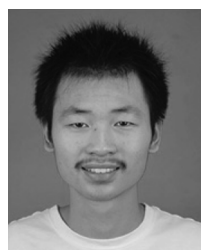
Zhong Zhou (M'10) received the B.S. degree from Nanjing University, Nanjing, China, in 1999, and the Ph.D. degree from Beihang University, Beijing, China, in 2005.

He is an Associate Professor and Ph.D. Adviser with the State Key Lab of Virtual Reality Technology and Systems, Beihang University. His main research interests include augmented virtual environment, natural phenomena simulation, distributed virtual environment, and Internet-based VR technologies.

Dr. Zhou is a member of the ACM.

Ke Chen received the M.S. degree in computer science and technology from Beihang University, Beijing, China, in 2010, and is currently working toward the Ph.D. degree in computer science from the State Key Lab of Virtual Reality Technology and Systems, Beihang University.

His research interests include remote rendering, 3-D visualization, and video coding.



Jingchang Zhang received the B.S. degree in computer science and technology from Shanghai University, Shanghai, China, in 2013, and is currently working toward the M.S. degree in computer science from the State Key Lab of Virtual Reality Technology and Systems, Beihang University, Beijing, China.

His research interests include remote rendering and 3-D visualization.