Prediction-Based Prefetching for Remote Rendering Streaming in Mobile Virtual Environments

Shaimaa Lazem and Marwa Elteir
Department of Computer Science
Virginia Tech, USA
{shlazem, maelteir}@vt.edu

Ayman Abdel-Hamid[1]
College of Computing and Information Technology
Arab Academy for Science and Technology, Egypt
hamid@aast.edu
[1]Adjunct Professor, Department of Computer Science, VT, USA
hamid@cs.vt.edu

Abstract

Remote Image-based rendering (IBR) is the most suitable solution for rendering complex 3D scenes on mobile devices, where the server renders the 3D scene and streams the rendered images to the client. However, sending a large number of images is inefficient due to the possible limitations of wireless connections. In this paper, we propose a prefetching scheme at the server side that predicts client movements and hence prefetches the corresponding images. In addition, an event-driven simulator was designed and implemented to evaluate the performance of the proposed scheme. The simulator was used to compare between prediction-based prefetching and prefetching images based on spatial locality. Several experiments were conducted to study the performance with different movement patterns as well as with different virtual environments (VEs). The results have shown that the hit ratio of the prediction-based scheme is greater than the localization scheme in the case of random and circular walk movement patterns by approximately 35% and 17%, respectively. In addition, for a VE with high level of details, the proposed scheme outperforms the localization scheme by approximately 13%. However, for a VE with low level of details the localization based scheme outperforms the proposed scheme by only 5%.

1 Introduction

Due to the recent advances in wireless networking, deploying distributed virtual reality applications such as virtual guiding malls, and online multiplayer games on mobile devices has become prevalent nowadays. This class of applications require rendering of rich 3D virtual scenes, which is challenging with respect to the restricted capabilities of the graphics card in these devices. Remote rendering is one of the proposed solutions, where the server renders the 3D scene for the client, and then streams the rendered images over the network to the client. This operation leaves few display and rendering tasks for the mobile device graphics card.

In [1], Boukerche et al. proposed a scheduling and buffering technique for remote rendering of virtual walkthrough applications. The client sends periodic updates with its position in the virtual scene to the server, and the server performs a look-ahead strategy to compute images that most likely required by the client in the next movements. Further, the server streams the images scheduled according to their priority to client. In following paper [2], the authors evaluated the performance of their streaming protocol under different network conditions. Moreover, they proposed a rate control scheme that adapts the sender's transmission rate to the observed received rate, and reduces the rate at congestions. Nevertheless, their system in [1] suffered from the following drawbacks:

- Large number of transmitted images per position update. Though the priority of the scheduled images is changed if a new position update is received, the number of images remains high especially with frequent client updates.
- They assumed fixed navigation speed to be able to predict what images are required at any time, which is not true for all movement patterns in virtual environments.
- The system does not handle the handoff situation where the client is disconnected from base station.

Motivated by these deficiencies our project addressed two research questions. First, how to predict images with high prediction accuracy for different movement patterns in virtual environment. Second, could this be integrated with a handoff recovery scheme?

To answer those questions, we proposed a prefetching scheme that predicts client next position form its historical movements. Hence, the server is able to send the predicted image one step ahead to the client. Furthermore, we argue that with high quality prediction for the client movements, the handoff period is better recovered by the predicted images.

We were able to achieve an answer for the first question, and partially answer the second one due to the time limitations. The rest of the report is organized as follows: first, the background and related work are presented. Second, the proposed solution is illustrated. Third, the simulation model is described. Then, the experiments conducted to evaluate the performance of the proposed prefetching technique are discussed. Finally, we conclude our project and present some ideas for future extensions.

2 Background and Related Work

Deploying distributed virtual environments require that the server and the client share the 3D scene over the network. Moreover, a rendering engine is required to render the model efficiently. Several solutions exist for visualizing the remote virtual environments (VE):

Geometry Replication: the entire geometric model is replicated locally at the client, or downloaded from the server before usage. Due to their limited storage, this solution is not suitable for mobile devices.

Progressive Meshes: initial transmission of a low polygon 3D model followed by the progressive streaming of instructions to create complex representations of the model. However, this operation is time consuming, and requires the model to fit entirely in the local memory during rendering, which again might be inappropriate for mobile devices.

Impostors: The sender is responsible for the rendering of complex 3D models and transmitting images that will be used as textures applied on certain simple geometry objects handled by the client.

Image-based rendering: the rendering task is performed by the sender and the resulting images are streamed over the network to the client side, which performs image-based rendering in order to create novel views.

Image-based solutions do not require large memory or disk storage on mobile clients. In addition, the heavy rendering task is performed at server side, which fits with the graphics card capabilities for those devices.

Using image-based rendering for rendering 3D graphics on mobile devices was first introduced by Chang and Ger [3]. In addition, they extended the technique on the client-server framework, where the server constructs the model dynamically and the client runs a 3D wrapping algorithm. Likewise, several client server architectures have been proposed that uses IBR for remote rendering of virtual reality on mobile devices. Yu Lei et al. [6] proposed an image-based walkthrough for mobile devices. They designed a rate control scheme that adapts transmitted images according to bandwidth. Moreover, a cache management mechanism was introduced where the server prefetches images nearest to client position and the most likely navigation path. Furthermore, Thomas et al. [8] presented image based rendering over client server architecture. A camera placement algorithm was introduced to avoid occlusion and exposure problems.

Moreover, Koller et al. [5] discussed the protection of copyrighted 3D models when they are used by remote users. The server holds the complete 3D model and sends images to the client on demand. Further, client renders a low-polygon model of the scene as the user manipulates it. When the user stops, the client sends a request to the server, which will then send a high resolution image of the scene.

Furthermore, [1] used panoramic image-based rendering, which consists of creating a panorama representation of the scene from several images taken by rotating the camera around a fixed point. Figure 1 shows the client server architecture introduced in [1] for rendering and streaming over the wireless network.

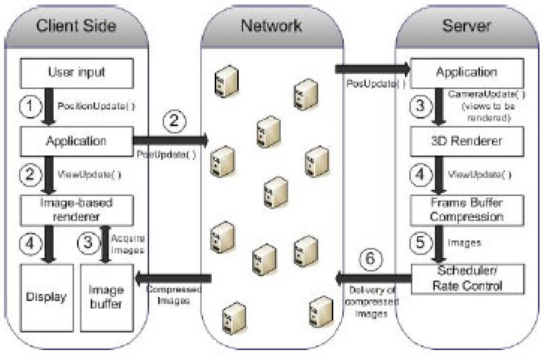


Figure 1: Overview of the streaming system [1]

In addition, they introduced a prefetching scheme that streams n look-ahead images to the client. This scheme considers the number of steps needed by the client to reach the position that requires that image. For example, in 2 step look-ahead strategy, the server sends all the images reachable one, and two steps from the client current position. Moreover, higher priorities are assigned to images reached after 1 step than the 2 steps images. The total number of images in this case is 10 images as shown in the figure below.

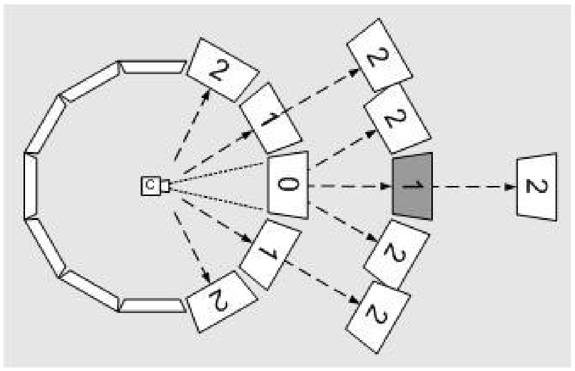


Figure 2:2 look-ahead steps prefetching [1]

Taking into consideration the errors, and disconnections of the wireless environments, in addition to the large image size, 10KB, the large number of streamed images results in a network overhead. Furthermore, this affects the number of clients serviced by one server. We argue that streaming fewer images with high prediction quality will reduce the load over the wireless network, which in turn lead to enhance the system scalability.

This project aims at:

- Proposing a scheme for predicting the needed images from the client movement history (Prediction-based technique).
- Comparing between sending predicted images and images that cover the surrounding environments (Localization-based technique) in order to determine the most influential images that should be sent to client.
- Study the prediction performance with different movement patterns, and with different types of environment.
- Propose a handoff recovery mechanism that incorporates with motion prediction to recover the handoff interval.

3 Prediction-Based Prefetching Framework

In our proposal, the client- server architecture introduced in [1] is used. The localization technique favors the spatial locality and sends the five images covering the field of view surrounding the client as shown in Figure 3 . While the prediction technique aims at anticipating the next movement for the client and sending the appropriate image one step ahead.

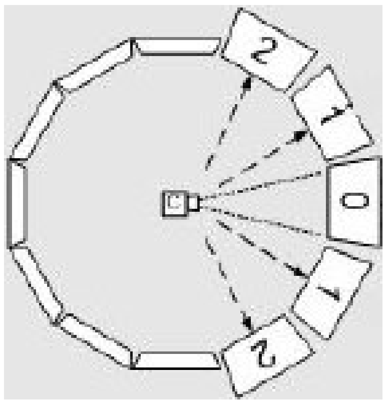
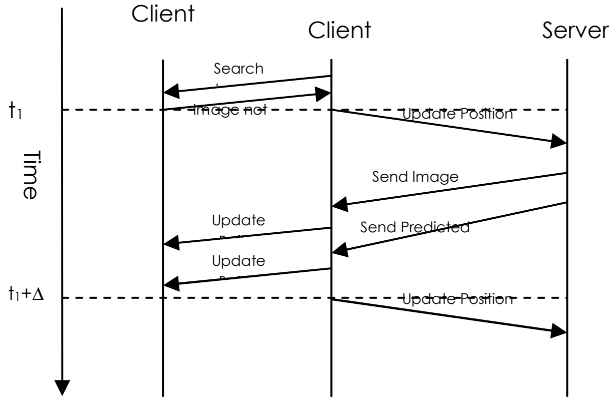


Figure 3: Localization-based prefetching

The proposed techniques work as follows. At regular basis Δt, the client sends a position update message to the server, where the server keeps the client movement history and use it for predicting the next movement in the prediction scheme. Further, it sends the image for the current position as well as the anticipated image.

As for the localization-based technique, the server sends the image corresponding to the client position and the 4 remaining images as in Figure 3.

In the following timing diagram, the proposed prefetching scheme is illustrated. Initially at position p1 the client searches its buffer to check whether the corresponding image exists for rendering. Then the client sends an update position message to the server with its position p1 at time t1. The server sends the corresponding images, and the client updates its buffer as the images are received.



Two predicting schemes, which were used for prefetching objects in distributed virtual environments [4], are adapted; the average window and the weighted average window.

In the average window scheme, the new position is computed as the average of the previous w steps, where w is the window size. The second scheme uses weighted average window, where the recent movement has α weight, and the average of the remaining window movements has (1- α) weight; $0 < \alpha \leq 1$. The following are the equations for both schemes, where P_{n+1} is the estimated movement vector.

Average window scheme:

$$\vec{P}_{n+1} = \vec{p}_n + \frac{1}{w}(\vec{p}_n - \vec{p}_{n-w}) \quad ; \quad n > w$$

Weighted average window:

$$\vec{P}_{n+1} = \vec{p}_n + \alpha(\vec{p}_n - \vec{p}_{n-1}) + (1-\alpha)\frac{1}{w-1}(\vec{p}_{n-1} - \vec{p}_{n-w-1}) \quad ; \quad w>1$$

For both techniques, large w denotes including more movement history in the prediction. Moreover, the two special cases where $\alpha = 1$ or w=1 refer to the situation where the prediction considers the last movement, in which the change in the client position after the next movement equals the previous one as if the client navigates with regular speed in the VE.

The client uses the least recently used (LRU) policy for cache replacement, where every image in the client buffer is stamped with its last access time. When new image is received and the buffer is full, the image with the lowest time stamp is replaced.

Handoff Recovery

During the handoff interval, the client will use the buffered images to navigate through the virtual environment. According to its movements, these images might not be enough to support the complete disconnected duration. A brute force solution is to increase the number of buffered images. However, this is not suitable due to the limited bandwidth of the wireless connection, and the restricted disk storage of the mobile devices. We propose an adaptive technique that overcomes the drawbacks of the straightforward method that integrates with the motion prediction scheme. The solution consists of two main steps:

- The server anticipates the handoff occurrence time
- The server predicts client movements and sends images to recover the incoming handoff interval. The extra images have lower resolution in order to recover larger periods
- To achieve these goals the client update message will be modified to incorporate an indicator for his signal strength, and the server will use this indicator to anticipate the handoff occurrence. Moreover, to decide which images to send, the accuracy of the prediction technique should be studied with larger number of look-ahead steps.

4 Simulation Framework

This section gives the details of the event driven simulator constructed to evaluate the performance of the proposed techniques. First, the simulation model is presented including the virtual environment model, the simulation events, and the simulation parameters. Finally, the simulation model is validated and verified.

Simulation model

Overview

The event driven simulator is used to simulate a system that contains a client and a server, the client communicates with the server in order to perform remote rendering for a virtual environment. As shown in Figure 4, the server is modeled by three components; a prefetching engine, a data structure that keeps track of the client history, and a data structure that represents the virtual environment model. The prefetching

engine receives a position update message from the client, prefetches the corresponding images from the virtual environment model and sends them back to the client. Moreover, the data structure that represents the virtual environment (VE) model is an m x n matrix. Each element in this matrix represents part from the VE and is modeled by 12 panorama images as shown in Figure 4.
The client is modeled by two components; a navigation tool that generates the movement path through the virtual environment, a data structure that represents a map for the virtual environment model, and the client buffer.

Simulation Events

There are six events used by the simulator. Four of them are needed for handling the process of remote rendering streaming. These are client changes position, server prefetches images, server sends image, and client updates buffer. The other two events are concerned with handling the handoff; handoff begin and handoff end. Here are the details of each event:

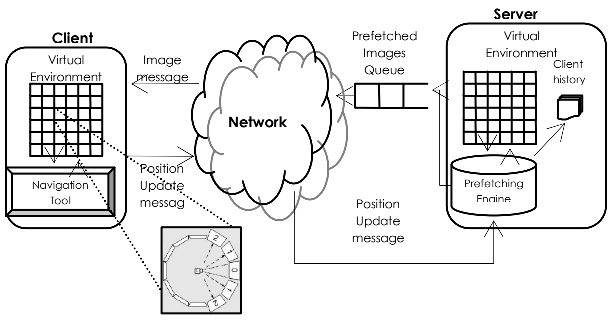


Figure 4: Simulation model the client, the server, and the virtual environment

Client Changes Position event

This event occurs at the client every ΔT. It is responsible for generating the navigation path through the VE. The simulator handles this event by signaling the target client to change its position assuming certain movement pattern. Moreover, the simulator schedules another client changes position event after ΔT, and if the client is connected to the server (no handoff), the simulator creates a new position update message and schedules server prefetches images event for this message after certain network delay.

Server Prefetches Images event

This event occurs at the server in response to the position update message sent by the client. The simulator handles this event by signaling the target server to run the prefetching engine in order to prefetch the corresponding images. Then it schedules server sends image event for each prefetched image in the output images queue. Before inserting new images in the queue, the server deletes any image prefetched for previous position update message.

Server Sends Image event

This event occurs at the server to send an image message to the client. The simulator handles this event by scheduling a client updates buffer event after certain network delay.

Client Updates Buffer event

This event occurs at the client in response to an image message sent from the server. The simulator handles this event by signaling the target client to update its data structure that represent the VE model in order to incorporate the new image.

Handoff Begin event

This event occurs when handoff is detected by the server. The simulator handles this event by simply informing all simulation entities that handoff occurred. At the end of the event, the simulator schedules handoff end event.

Handoff End event

The simulator handles this event by informing all simulation entities that handoff interval is finished.

Simulation Parameters

The simulation includes several parameters that control its behavior. All of these parameters are summarized in the table below along with a description for each parameter.

Parameter	**Description**
Cell size	Used to characterize the virtual environment model. Increasing this value indicates fewer details in the VE.
Step size	Used to characterize the virtual environment model. Using step size smaller than the cell size indicates VE with small level of details and vice versa.
Window size	Used to characterize the motion prediction scheme. Increasing this value indicates that the client history is highly affect the prediction.
Alpha	Used to characterize the motion prediction scheme. Increasing this value indicates that the previous client step has greater effect than the older steps in the predicted images.
Client Buffer	Indicates the number of images that represent the VE model at the client.

Simulation assumptions

- ΔT is greater than the time needed by the client to receive one image from the server; otherwise, the hit ratio will be always zero.
- What is sent from the server is received correctly by the client and vice versa.
- Since we handle only one client in our simulator, we assumed that when the server receives a position update

message from the client, it would directly prefetch the corresponding images from the VE model.

Simulation Model Validation and Verification

Sargent, [7] presents the modeling process, where the *problem entity* is the real system to be modeled; the *conceptual model* is the mathematical, logical or verbal representation of the problem; the *computerized model* is the conceptual model implemented on a computer. The conceptual model is developed through an analysis and modeling phase, the computerized model is developed through a computer programming and implementation phase, and inferences about the problem entity are obtained by conducting computer experiments on the computerized model in the experimental phase.

Based on [7], *computerized model verification* is defined as ensuring that the implementation of the computerized model is correct. *Operational validity* is defined as determining that the model output behavior has sufficient accuracy. *Data validity* is defined as ensuring that the data necessary for the model building, model evaluation and testing, and conducting the model experiments are correct. A discussion about the validations and verifications of our simulation model is given below.

Operational Validity

We ensure that the behavior of our simulator is correct by conducting a simple simulation run that involves small number of movement steps. We printed the random positions generated form the navigation tool in addition to the prefetched images generated from the server's prefetching engine. Then we calculated the corresponding hit ratio, we found that it equals to the value generated from our simulator and hence we ensured that our simulator runs correctly.

Computerized model verification

The design of our simulator follows many of the software engineering concepts, including object-oriented design and program modularity. These concepts facilitated the verification process.

Moreover, we conducted several experiments, which guarantee that our implementation is correct. For example, we conducted an experiment that changes the size of the available buffer at the client. As expected, we noticed that as the buffer size decreases the hit ratio also decreases. Another experiment that study the effects of increasing the number of steps per transition reflects that as the number of steps per transition increases the hit ratio also increases which is also an expected behavior.

Data Validity

The data used to generate our experiments are the movement patterns, the equations used for generating such movements are presented in [4]. Moreover, we included two parameters to further control the behavior of the random walk pattern and conducted two sensitivity analysis experiments for these parameters.

5 Performance Evaluation

For evaluating the performance of the proposed techniques, two sets of experiments were conducted. The first set analyzes the performance of the prediction technique relative to the prediction parameters, window size and Alpha, while the second set aims at comparing the performance of prediction and localization techniques. In both sets, two movement patterns are tested. First, the details of these patterns are given then each set of experiments is discussed in details.

Movement Patterns

We studied two types of movements based on the description provided in [4], the random walk and the changing circular pattern as shown in Figure 5, and Figure 6 respectively.

Random Walk (RW)

In the random walk movement, Figure 5, each step is either a translation with arbitrary length or rotation with arbitrary angle as described by the following equations. The point (x',y') represents the new position, while the point (x,y) is the old position. The translation could be positive or negative. Moreover, in the simulation the rotation angle is restricted from 0 to 180 degrees.

Translation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Rotation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Two parameters were used to characterize the RW movement pattern, the maximum number of steps the client spends in one path (S), and the rotation ratio in the movement (R). Large S represents navigation in straight lines. For example, walking or driving in large environments such as virtual malls or cities. On the other hand, large number of rotations reflects movements in large environments with many objects to see in it.

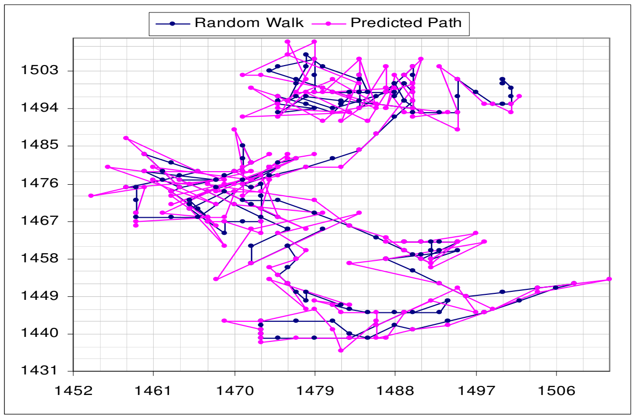


Figure 5: Random walk sample along with the predicted path Cell size=3 step size=5

Changing Circular Pattern (CCP)

As for the changing circular pattern, each step includes a translation with arbitrary length along with rotation with an angle θ. Furthermore, θ changes every four steps by 10 degrees. Circular pattern corresponds to movements around interesting objects or landmarks in the environment. The rotation equation for CCP is the same as in the random walk pattern.

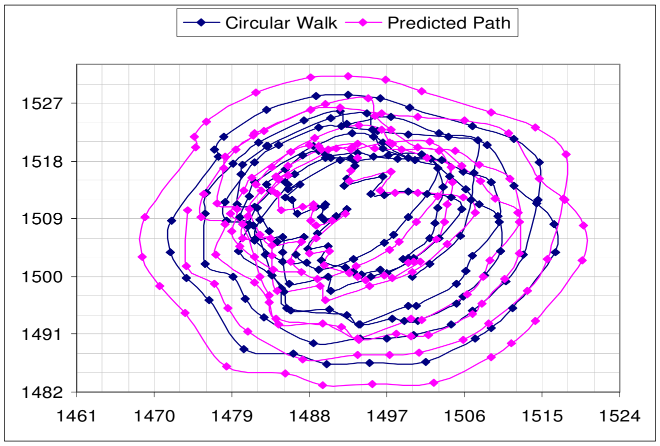


Figure 6: Circular walk sample
Cell size=3 step size=5

Studying the performance of the proposed Prefetching technique

In this section, two experiments are conducted in order to study the performance of different motion prediction schemes using two types of movement patterns. The main performance metric measured in these experiments is the hit ratio. The parameters chosen to reflect the prediction scheme are the Alpha and the window size.

Experiment1: Studying the effect of increasing the window size on the hit ratio

In this experiment the effect of increasing the window size on the hit ration were studied. The next movement vector is estimated as the average of the previous w steps, where w represents the window size. The effect is studied for the random walk as well as the circular movement pattern as shown in Figure 7, Figure 8 in order.

As for the random walk, Figure 7, the best performance is achieved at window size 1 with hit ratio 45%. As the window size increases the performance decreases reaching 14% at window size = 10. This indicates that by including more historical movements in the prediction window, the prediction techniques become less accurate. Moreover, the best performance occurred at w=1, where the prediction considers the previous movement only.

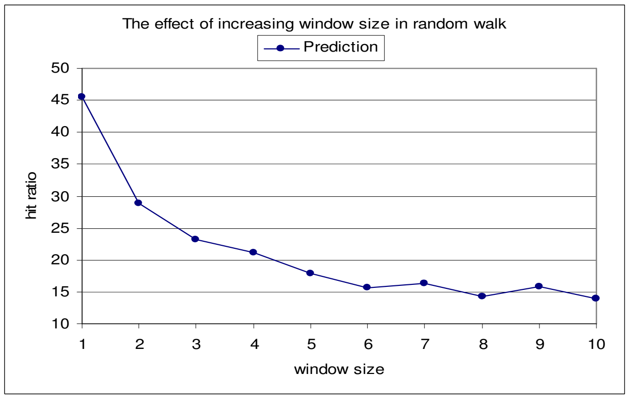


Figure 7: The effect of increasing the window size on the hit ratio for random walk.
Cell size=3 step size=5 simulated number of steps=201
Steps/Transition=3 Buffer size=1000

Mean=21.38, Standard deviation=9.72, 90% Confidence interval (15.65, 26.92)

Figure 8 shows the same experiment for circular movement pattern. The best performance is at w= 1 with hit ratio 26%, and decreases as the window size increases reaching 14% at w=9. Similar to the random walk, the prediction efficiency for the average window scheme decreases as the number of movements in the window increases. The previous sensitivity analysis for the window size suggests using the w=1 for the prediction.

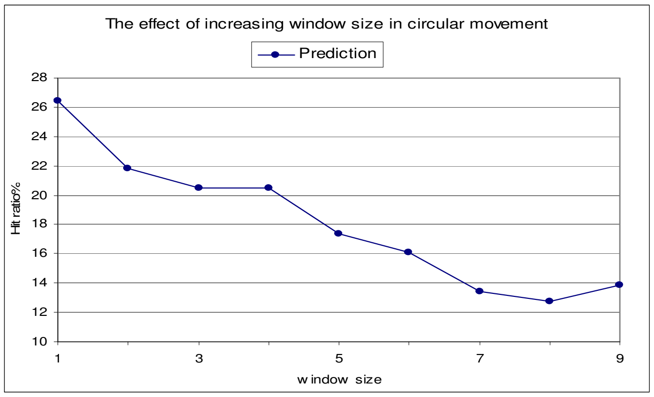


Figure 8: The effect of increasing the window size on the hit ratio for random walk.
Cell size=3 step size=5 simulated number of steps=201
Steps/Transition=3 Buffer size=1000

Mean=18.09, Standard deviation=4.59, 90% Confidence interval (15.24,20.93)

Experiment2: Studying the effect of increasing Alpha on the hit ratio

Experiment 2 studies the effect of increasing Alpha, the recent movement weight, in the weighted average prediction scheme for the random walk and circular movement patterns.

Figure 9 illustrates the change for the random walk movement and window size 2 and maximum number of steps per transition equals 3. At Alpha=0.1 the hit ratio is 27% and it remains almost constant until Alpha= 0.7. Then the hit ratio starts to increase reaching 67% increase with hit ratio 45% at Alpha=1 i.e. the prediction estimates the next translation from the previous movement only.

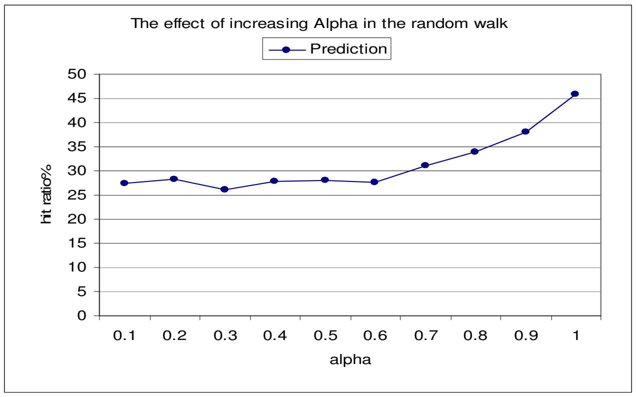


Figure 9: The effect of increasing Alpha on the hit ratio for random walk.
Cell size=3 step size=5 simulated number of steps=201
Steps /Transition=3 Window size=2 Buffer size=1000

Mean=31.4, Standard deviation=6.27, 90% Confidence interval (27.77,35.04)

As for the circular movement, Figure 10 depicts the effect of increasing Alpha for window size= 2, and maximum number of steps per transition= 3. The hit ratio is almost constant with mean =24.88 with 90% confidence interval [24.12, 25.64]. Recalling the circular movement pattern, each step is a translation with a maximum value of the step size, and rotation with an angle that increases by 10 degrees every 4 steps. This suggests that most of the movement time there is approximately constant change in position every movement step. That is why Alpha does not affect the hit ratio.

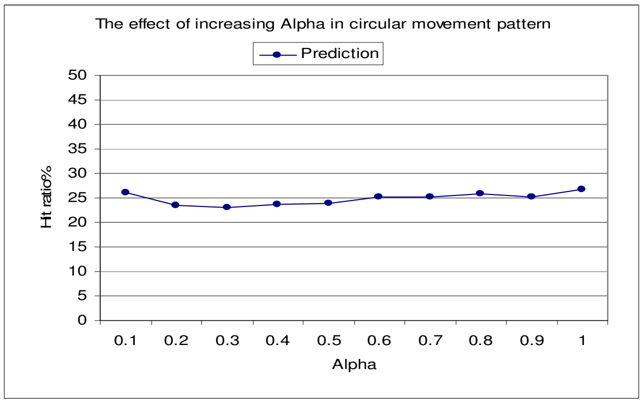


Figure 10: The effect of increasing Alpha on the hit ratio for circular movement pattern.
Cell size=3 step size=5 simulated number of steps=201
Steps/Transition=3 Window size=2 Buffer size=1000
Mean=24.88, Standard deviation=1.25, 90% Confidence interval (24.12, 25.64)

Comparing the performance of the proposed technique with the localization technique

In this section, several experiments are conducted in order to compare the performance of the prediction based Prefetching technique with the localization technique. The main performance metric measured in these experiments is the hit ratio.

Experiment3: Studying the effect of changing the movement patterns on the hit ratio of both techniques

Experiment 3 studies the effect of the parameters characterizing the random walk movement pattern on the hit ratio for both techniques. The first parameter, Figure 11, is the maximum number of steps the client remain in one movement path (S), while the other is the rotation ratio in the movement (R), Figure 12, which indicates how much was the movement was random rotation.

Figure 11 shows the effect of increasing S for both techniques on hit ratio. The movement pattern is random walk with no rotations, and Alpha=1. When S=1 the localization technique performs better than the prediction technique. This is due to the fact that the client position changes every step, which results in reducing the prediction efficiency. As S increases the prediction method outperforms the localization method by average difference 39% in the hit ratio. On average the prediction is better than localization by [22.85, 46.82] at 90% confidence level.

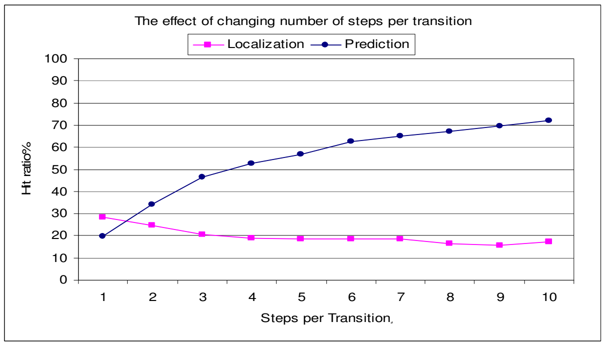


Figure 11: The effect of increasing Steps/Transition on the hit ratio for random walk.
Cell size=3 step size=5 simulated number of steps=201
Rotation ratio=0 Alpha=1 Buffer size=1000 90% Confidence interval (22.85,46.82)

As for the rotation ratio in the movement (R), Figure 12 depicts the effect of increasing the rotation ratio from 0 (the movement consists of translation only) to 50% of the movement with maximum number of steps per transition = 3 and Alpha=1. At R=0, the prediction method is better than the localization method with almost twice the hit ratio. As the R increases in the movement the performance of the prediction technique degrades. This is due to the fact that the rotation movement is not linear, while the prediction equation is linear. However, the prediction technique still outperforms the localization even with higher R. For example, at R=50 the prediction hit ratio was 24% while the localization hit ratio was 15%. On average the prediction is better than the localization by [13.644, 20.89] with 90% confidence level.

The previous experiment shows that the prediction technique outperforms the localization method, specially if the movement was in straight lines and small number of rotations. This improvement is very apparent in the random walk movement pattern. Moreover, the best performance is reached with window size = 1 and Alpha =1. However, the hit ratio for the prediction technique decreases in the circular walk because of the non-linearity of the movement pattern.

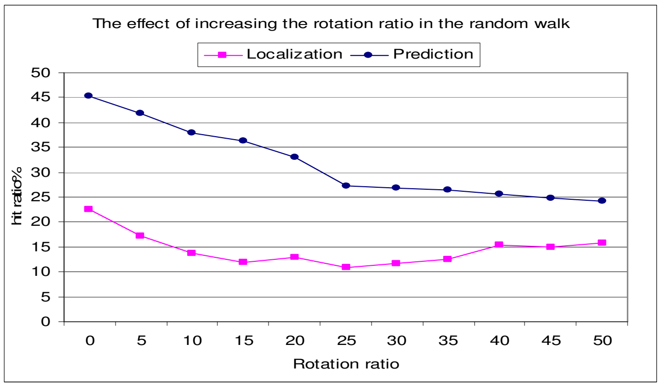


Figure 12: The effect of increasing the rotation ratio on the hit ratio for random walk.
Cell size=3 step size=5 simulated number of steps=201 Steps/Transition=3 Alpha=1 Buffer size=1000
90% Confidence interval (13.644,20.89)

Experiment4: Studying the effect of reducing the size of the client buffer on the hit ratio of both techniques

Since the mobile devices are limited in their storage capacity, it is essential to study the effect of changing the client buffer on the hit ratio. In this experiment, the client buffer is decreased from 1000 images (10 MB) to 1 image (1 MB). 10 MB is a reasonable choice since most of currently available PDAs have memory capacity of at least 32 MB. The same experiment is conducted for two movement patterns; the random walk and the circular walk.

For random walk, Figures 1 has shown that the hit ratio of the localization technique decreases from 25% at 10MB buffer to 10% at 1MB buffer. However, the hit ratio of the prediction based technique is not affected by the buffer size; it remains 35% for all buffer sizes. In other words, the hit ratio in case of prediction based prefetching is better than that of the localization based prefetching by [16.4, 23.22] with 90% confidence level.

For circular walk, Also Figure 2 has shown that the hit ratio of the localization technique decreases from 22% at 10MB buffer to 8% at 1MB buffer. However, the hit ratio of the prediction based technique is not affected by the buffer size and it remains 25% for all buffer sizes. In other words, the hit ratio in case of prediction based prefetching is better than that of the localization based prefetching by [15.1, 20.4] with 90% confidence level.

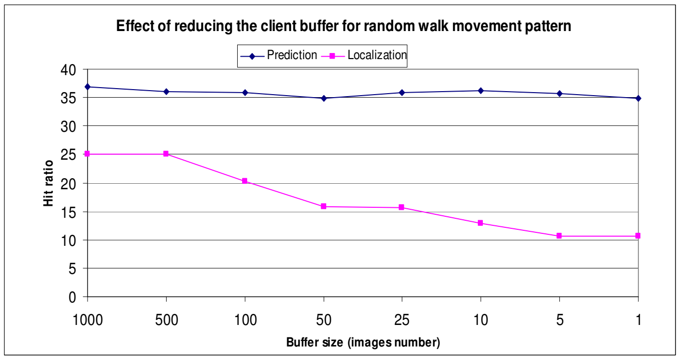


Figure 13: the effect of changing the client buffer size in the hit ratio for random walk
Cell size=3 step size=5 simulated number of steps=335 Steps in one path=2 window size=2 Alpha=0

The obtained results reflect that the hit ratio of the motion prediction technique is based only on the recent prefetched image and hence huge buffer at the client is not necessary. However, the hit ratio of the localization technique is based on the old prefetched images and hence the size of the available buffer is crucial. Generally, we can say that our proposed technique is more resilient to the reduction of the buffer size than the localization technique.

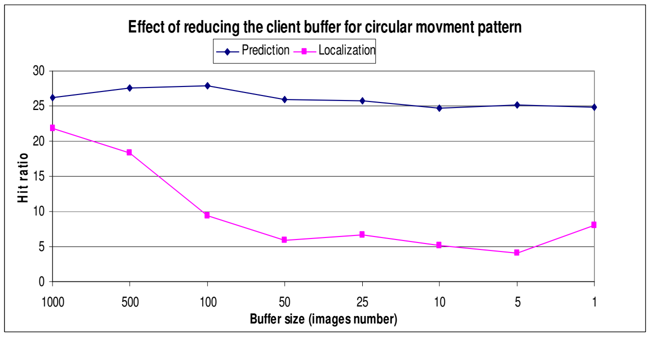


Figure 14: the effect of changing the client buffer size in the hit ratio for circular walk
Cell size=3 step size=5 simulated number of steps=335 Window size=2 Alpha=0

Experiment5: Studying the effect of changing the virtual environment model on the hit ratio of both techniques

Since the level of details of the virtual environments (VE) is different and it is highly affect the hit ratio of any Prefetching technique. It is necessary to study the performance of the proposed technique under different VE models. In this experiment, the cell size and the step size of the target VE is changed in order to reflect the level of details of the VE. We begin with cell size larger than the step size, which indicates VE model with small level of details, and keep increasing the step size until it becomes larger than the cell size, which indicates VE model with high level of details. The same experiment is conducted for two movement patterns, the random walk and the circular walk.

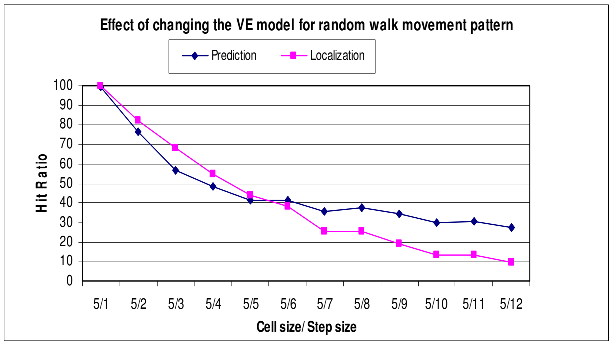


Figure 15: the effect of changing the VE model in the hit ratio for random walk Simulated number of steps=200 buffer size=1000 Window size=2 Alpha=0

For random walk, Figure 3 has shown that for both prefetching techniques as the step size becomes large enough to reflect VE model with high level of details (i.e., In every step the client requests images that most probably do not exist in its buffer), the hit ratio decreases. For localization prefetching, the hit ratio decrease from 100% at cell size=5 and step size=1 (low level of details) to 10% at cell size=12 and step size=5 (high level of details). For motion prediction based prefetching, the hit ratio decrease from 100% at cell size=5 and step size=1 to 30% at cell size=12 and step size=5.

Moreover, we noticed that for VE models that have low level of details (represented by the first five points of each curve) the localization prefetching behaves better than the prediction based prefetching by approximately 5% this is because the former technique sends 5 images per every client update, there is a high probability that the client requests any of theses images in the subsequent steps. However, for VE models that have high level of details (represented by the last seven points of each curve) the prediction based technique behaves better than the localization technique by approximately 13% this is because the localization technique prefetches images based on the spatial locality which in this cases are no longer needed because at every step the client becomes in a different cell. In other words, the hit ratio in case of prediction based prefetching is better than that of the localization based prefetching by [9.6, 1.26] with 90% confidence level.

For circular walk, Figure 4 has shown that the performance of both techniques is approximately the same. The hit ratio decreases from 100% at cell size 5 and step size 1 to 20% at cell size 5 and step size 12. Moreover, for the VE models with low level of details; localization technique behaves better than prediction based technique by 2% on the average and for the VE models with high level of details; prediction technique behaves better than localization technique by 3% on the average. In other words, the hit ratio in case of prediction based prefetching is better than that of the localization based prefetching by [2.07, 0.04] with 90% confidence level.

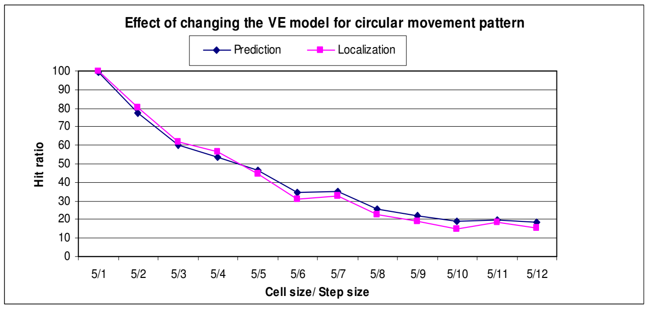


Figure 16: the effect of changing the VE model in the hit ratio for circular walk Simulated number of steps=200 buffer size=1000 Window size=2 Alpha=0

Finally, It is worth mentioning that all of the above experiments are conducted assuming that the number of steps per transition is 2 for the random walk movement, furthermore the prefetching based technique only sends two images for every client update. So for the cases where the localization based technique behaves better than our proposed technique, it is expected that our behavior becomes better if we send more images per update or if the number of steps per transition is increased which is not far from the real movement.

Comparing the performance of the proposed technique with the localization technique in the handoff case

Experiment 6: Studying the hit ratio during the handoff period for both techniques

There are two types of handoff; the first type is the soft handoff where the mobile device migrates from one coverage area to another one without being disconnected from the network. The other type is the hard handoff where the mobile device is completely disconnected from the network during the migration. To study the soft handoff we need to implement some details of the lower layers, so we leave this part to another study and focus instead in studying the hard handoff.

In this experiment, the performance of the localization technique is compared with the prediction based technique. Four versions of the prediction based technique are studied; Prediction 2, Prediction 3, Prediction 4, and Prediction 5, where Prediction n means motion prediction technique that sends n images per each client update. Moreover, we increased the handoff interval period from 5 to 50 client steps.

As shown in Figure 17, as the handoff interval increase the hit ratio during the disconnection interval is decreased, this is a normal behavior since more images are requested which are not available in the client buffer. Furthermore, as more images are sent in the prediction based technique, the performance gets better. Comparing these techniques with the localization technique, we can state that Prediction 5 behaves better than the localization technique by (0.55, 4.3) with 95% confidence level.

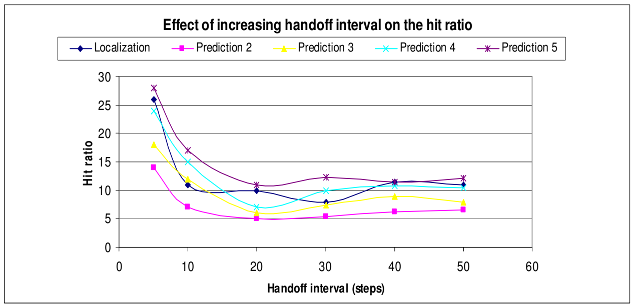


Figure 17: the effect of increasing the handoff interval on the hit ratio for random walk
Simulated number of steps=200 buffer size=1000

Steps in one path=2 window size=2 Alpha=1

6 Conclusions and Future Work

In this project, we proposed a prefetching technique for remote rendering streaming of mobile virtual environments. The introduced technique employs motion prediction at the server side to predict future movements based on the client history. Moreover, several experiments were conducted to compare between prediction, and spatial locality based prefetching for different types of movement patterns as well as different virtual environments. Further, a solution for recovering the handoff interval was proposed and partially implemented. The following is a summarization for our results:

- The best motion prediction scheme for all movement patterns is window scheme with window size=1 or the weighted average model with Alpha=1. i.e., consider the previous movement only to predict the next movement.
- The motion prediction-based prefetching outperforms the spatial locality-based prediction in the random walk movement as the number of steps per transition increases and even when increasing the rotation ratio in the movement.
- The motion prediction-based prefetching is more resilient with the client buffer size than the localization technique.
- For the VE with low level of details and small movements (i.e., cell size/ step size >1), the localization technique performs better than the prediction technique. However, for the VE with high level of details and large movements (cell size/ step size <1) the prediction technique performs better than the localization technique.
- Combining both techniques results in better performance in case of handoff for all handoff intervals.

Future work includes a complete implementation and analysis of our technique during a handoff interval. We will study the scalability of the proposed technique relative to the number of clients. More complex motion prediction techniques will be developed that dynamically adjust the type and number of the prefetched images according to the current VE and movement pattern.

References

[1] A.Boukerche, and R.W.Pazzi, Scheduling and Buffering Mechanisms for Remote Rendering Streaming in Virtual Walkthrough Class of Applications, WMuNeP'06, October 6, 2006, Torremolinos, Malaga, Spain.

[2] A.Boukerche, and R.W.Pazzi, Performance Evaluation of a Streaming Based Protocol for 3D Virtual Environment Exploration on Mobile Devices, MSWiM'06, October 2–6, 2006, Torremolinos, Malaga, Spain.

[3] C. Chang, and S. Ger. Enhancing 3D Graphics on Mobile Devices by Image-Based Rendering, In Proceedings of the 3rd IEEE Pacific Rim Conference on Multimedia: Advances in Multimedia Information Processing, 1105-1111, 2002.

[4] J. Chim, M. Green, R. W. H. Lau, H. Leong, and A. Si, On caching and prefetching of virtual objects in distributed virtual environments. ACM Multimedia, 1998, New York, 171–180.

[5] D. Koller, M. Turitzin, M. Levoy, M. Tarini, G. Croccia, P. Cignoni, and R. Scopigno, 2004. Protected interactive 3D graphics via remote rendering. ACM Trans. Graph. 23, 3 (Aug.2004), 695-703.

[6] Y. Lei, Z. Jiang, D. Chen, and H. Bao. Image-Based Walkthrough over Internet on Mobile Devices, GCC Workshops, LNCS 3252, 728–735, 2004.

[7] R. G. Sargent; "*Verification and validation of Simulation Models*", Proceeding of the *Winter Simulation Conference*, 1998.

[8] G. Thomas, G. Point, K. Bouatouch, A client-server approach to image-based rendering on mobile terminals. Technical Report, ISSN 0249-6399, France, January 2005.