User Motion Prediction in Large Virtual Environments

Jaroslav Přibyl
Department of Computer Graphics, FIT
Brno University of Technology
Božetěchova 2
612 66, Brno, Czech Republic
pribyl@fit.vutbr.cz

Pavel Zemčík
Department of Computer Graphics, FIT
Brno University of Technology
Božetěchova 2
612 66, Brno, Czech Republic
zemcik@fit.vutbr.cz

ABSTRACT

Motion prediction of various objects is important for work of many people. We have to distinguish between near and distant time prediction queries. The trajectory of an object represented by mathematical functions can be used for near time prediction. These formulas are often called motion functions and they use recent movements to predict future locations of the objects. It is impossible to use simple mathematical formulas to evaluate distant time queries, because the movement trajectory between current and distant future time can alter widely. Trajectory pattern of an object is suitable prediction method to take into account for both near and distant time queries. Consequently, data mining methods can mine trajectory patterns from historic movements and these patterns can be used to predict the future objects movement. The best contemporary methods exploit combination of trajectory pattern method and motion function. This means that in case no trajectory pattern is found, the motion function is used to determine object near location. Using the trajectory pattern prediction principle a new approach to optimize communication between client and server in large virtual environments is introduced. Both short time and long time prediction queries are used to minimize the overall amount of downloaded data from network server and to obtain the probably requested parts of the scene in advance.

Keywords

Motion prediction, large virtual environment, user profile, trajectory pattern, movement history

1. INTRODUCTION

User (or object) motion prediction in large virtual environments will be described in this paper. The large virtual environment can be a terrain model or a model of a building. Whole scene is stored in a network database. The client application transmits to the server information about recent movements of object and its current position. The server predicts the future object position and sends necessary data (e.g. geometry, textures and other application dependent data) to the client application. The near time query prediction is sufficient (e.g. rendering optimization, data preprocessing, etc.) in many cases, but a need of long time prediction (e.g. mobile networks, deliver network services, traffic information etc.) is sometimes necessary to uncover the future location of the object.

Current location and current movement of the object are needed to perform a short time prediction. Several techniques were invented to predict near time location of the object. These methods work fine for near time prediction queries but fail when they'll be used to predict location far away from the current position of the object.

Motion functions can be used to predict the near locations. These functions describe motion of an object by simple mathematical formulas. They can predict trajectory of the object reasonably well for near time locations. The motion function cannot be used to predict locations far away from the current object position because the motion of object might be affected by various circumstances (terrain obstacles, road networks, regular traffic jams, etc.). Let us consider an example (see Figure 1.). Some linear motion functions [Tao03a, Pat04, Jen04, Sal00 and Tao03b] can be used to predict the future object location during several consequent days. The prediction may fail e.g. on Friday in our example. To avoid this, nonlinear functions can be used [Tao04a]. They can capture the regular path of the object on Friday, but they may fail too (e.g. predict the “wrong place” in our example). Instead of the motion function, the movement pattern principle can be used to predict future location of the object accurately.

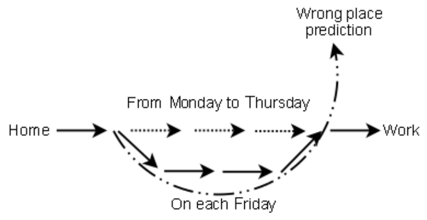

Figure 1. User go from home to work each week.

Trajectory pattern can be used to examine that on Friday the user goes to work with some probability and during other week days with another probability [Yav05, Yan06].

The main interest of the above mentioned technical articles is to discover trajectory patterns. They assume that the prediction can be easily done through the discovered trajectory patterns. But they don't take into account that the prediction query result can consist of many of patterns. The total amount of discovered patterns grows with every new object in the virtual environment. Therefore we need a method to organize the large amount of trajectory patterns and an efficient data structure to answer the predictive queries fast and accurately. Only several technical articles address the problem of large amount of discovered patterns [Jeu08]. Some of them address the problem partially [Mam04] and they develop pattern indexing techniques. Very few works [Jeu08] solve the case when no pattern is found. They use trajectory patterns to answer the predictive query and in case when no pattern is found call motion function to predict future location of the object.

In this paper we present a new approach to predict object motion in large virtual environment. To the best of our knowledge no one has solved the object motion prediction in context of 3D virtual environment. We use the prediction for optimize the rendering process and the data flow between client application and network server.

2. RELATED WORK

In this section we introduce state of the art of object motion prediction methods. Generally the object motion prediction methods can be categorized into two main classes.

Motion function based prediction

Motion functions result from vector representation of object motion, position and direction. Motion functions can be categorized into linear and nonlinear types. Linear models [Sal00, Tao03b] assume only linear movement of an object. Moreover, nonlinear models assume nonlinear movement [Tao03c].

At time t_0, the object location equals l_0 and velocity equals v_0. Object future location can be predicted for the linear model at time t_f through this following equation:

$$l(t_f) = l_0 + v_0 \times (t_f - t_0),$$

where l and v are 3-dimensional vectors.

The nonlinear prediction methods are generally more accurate in comparison with the linear methods. Today most accurate prediction mathematical method is recursive motion function (RMF) [Tao03c]. Object location l at time t is defined as follows:

$$l_t = \sum_{t=1}^{f} c_i \cdot l_{t-i},$$

where c_i is a constant matrix and f is the minimum number of the most recent timestamps which are needed to compute the elements of all c_i. The RMF method can be used only for near future prediction. The prediction accuracy can drop severely if we would like to know where the object is located in some distant time. Generally the motion functions predict wrong locations when movement of the object contains sudden changes in its direction or velocities. It is due to the motion function is strongly dependent on current previous locations.

Pattern based prediction

A lot of methods solve a pattern based prediction problem. One of them is neural network pattern based prediction. The classical back propagation network joined with a self organizing feature map (SOFM) can be used.

Further useful method is discrete Markov model [Rab89]. Some works derive Markov transition probabilities from one or multiple cells to another.

Association rules can be used to predict future locations of an object as well. Work [Yav05, Tao04b] address spatio-temporal association rules of the form $(r_i, t_1, c) \rightarrow (r_j, t_2)$, with confidence c, where r_i and r_j are regions at time t_1 a t_2 respectively ($t_2 > t_1$). In other words an object in location r_i at time t_1 is likely appear in location r_j at time t_2 with probability c. Study [Yan06] mine sequential patterns from object trajectories.

All above mentioned techniques can discover large amounts of patterns and amounts of patterns discovered as a result of a predictive query can be large as well. Only [Jeu08] method organizes the discovered patterns and can quickly answer to the predictive query.

No previous article solves the case if large amount of objects in virtual environment store and use their own motion history. Otherwise no study describes an opportunity to use the pattern based prediction techniques in computer graphics especially in distributed 3D virtual environment. The main purpose of this paper is to optimize the rendering process and communication between client and network server.

3. PREDICTION MODEL FOR LARGE VIRTUAL ENVIRONMENT

The proposed approach is intended for large distributed virtual environments. The prediction system is formed as a part of a server including representation of all models. The client applications request data from the server and render the scene.

The virtual environment is divided into a regular grid. This approach was chosen for several reasons. The first reason is that the virtual environment consists of large amount of textures. In order to save free memory, the quad tree data structure was employed to handle texture level of detail. Furthermore fast rendering, distributed character of the scene and prediction related to distributed parts of the virtual environment were good reasons for involving the regular grid. Each cell of the grid represent one distributed piece of the scene and has to be managed in the network server. There can be a large amount of cells to be managed on the server. Each cell of the grid can be a possible object location on his path through the virtual environment. Therefore the prediction granularity should be one cell.

The trajectory pattern of an object is represented by sequence $\{(l_0, l_{1, \ldots,} l_{i, \ldots,} l_{n-1})\}$, where l_i denotes object location l_0 at time i. Location coordinates are [x, y, ..., d]$\in R^d$, where d is used dimension. Object trajectory pattern can be discovered from its historical movement [Mam04]. A given period T is defined by a number of timestamps such that a trajectory pattern may reappear. An object trajectory can be decomposed into $\left\lfloor \frac{n}{T} \right\rfloor$ sub-trajectories. The period T is data dependent. To identify all locations with the same time offset the term group is used. In other words each group G_t represents all locations visited by object at time offset t. Consequently, the groups are clustered to dense clusters R_t. We call R_t a frequent region at time t. The R_t^j symbol is used to distinguish between frequent regions with the same time offset. It represents j-th frequent region at time offset t.

These definitions can be adapted to our model of virtual environment. We need to predict the future cells which will be necessary to be downloaded from network server in near and far future. For our virtual environment, the time period T is a sequence of visited cells during one client session. A group G corresponds to frequent region R. At this point the dense cell and the dense region terms have to be defined.

Definition 1: Dense cell c is a cell with more than one pass through of an object along the movement history of the object in virtual environment.

Definition 2: The size of a dense cell *Size(c)* is defined by amount of visitors of the cell c along the movement history of an object in virtual environment.

Definition 3: Dense region is a group of adjacent dense cells with the same dense cell size. All dense cells associated with dense region R_j^m have the same dense cell size j. The variable m is dense cell identifier and $m \in <1, n>$, where n is number of discovered dense regions.

Let us consider an example. Each red and green cell is a dense cell in Figure 2. The green groups are dense regions at level 2 and the red groups of cells are dense regions at level 3. Overall we have five dense regions counted from left to right and from bottom to top in this example. The bottom-leftmost red region is dense region R_3^1 and the top-rightmost dense region is marked as R_2^5.

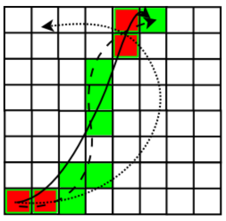

Figure 2. Virtual environment is divided into regular grid. The colored groups of cells are dense regions.

A density based clustering algorithm DBSCAN [Est96] was adapted to obtain dense regions. We modify the algorithm to work as follows. For every dense cell c from dense region R a dense cell q from dense region R exists, so that dense cell c is inside the Eps-neighborhood of dense cell q and $N_{Eps}(q)$ contains at least *minPts* dense cells.

Definition 4: The Eps-neighborhood of a dense cell c marked as $N_{Eps}(c)$ is defined by:

$$N_{Eps(c)} = \{q \in D \mid size(c) = size(q)\},$$

where D is input set of dense cells.

We have introduced a method to obtain dense regions R from a sequence of object locations. Dense regions allow us to describe object motion by trajectory pattern and to predict next movement of the object.

Trajectory pattern concept

The concept of a trajectory pattern was defined in earlier work [Jeu08]. Now our modification of the trajectory pattern concept is proposed.

Definition 5: The trajectory pattern *P* is an association rule of form $R_j^1 \wedge R_j^2 \wedge ... \wedge R_j^m \xrightarrow{conf} R_j^n$. The form $R_j^1 \wedge R_j^2 \wedge ... \wedge R_j^m$ is sequence of dense regions visited by object. We call this sequence a *premise*. The form R_j^n is a dense region called a *consequence* and this dense region might be visited by the object with probability *conf.* The term *conf* stands for confidence.

Thanks to current visited dense regions we are able to predict the object future location (dense region). The prediction is done by a prediction query. The prediction query is defined by number of dense regions which will be predicted. Two types of prediction queries are defined.

Definition 6: The distant time prediction query is a spatio-temporal query satisfying the condition that the number of predicted dense regions is greater than *d*, where *d* is user defined threshold.

The boundary between distant time query and non-distant time query is strictly application-dependent.

Trajectory pattern retrieval

The trajectory pattern discovery process can be divided into two parts. First of all we have to detect the dense regions. This task can be completed by modified DBSCAN algorithm which was described previously. The second task in trajectory pattern discovery process is to extract trajectory patterns (it corresponds to association rules) from input dense regions. The apriori algorithm [Agr94] can be used to extract association rules from user movement history described by sequences of dense regions.

Apriori algorithm

We briefly describe the apriori algorithm used to obtain the association rules from dense regions. Inputs of the apriori algorithm are sequences of dense regions. This set of sequences represents movement history of the object. Outputs of the apriori algorithm are sets of association rules with various length and probabilities.

3.1.1 Obtain frequent 1-patterns

We can obtain frequent 1-patterns from dense regions discovered by DBSCAN algorithm. Each frequent 1-pattern corresponds to one dense region R_j.

3.1.2 Obtain frequent k-patterns

The time period T is a sequence of visited dense cells during one client session.

Trajectory ID	Sequence of visited cells
TID1	1, 2, 10, 11, 19, 27, 28, 36, 44, 45, 53, 61, 62
TID2	1, 2, 3, 11, 12, 20, 28, 36, 44, 52, 53, 61, 62
TID3	1, 2, 3, 4, 12, 13, 21, 22, 30, 38, 46, 53, 54, 58, 59, 60, 61

Table 1. Database of visited grid cells.

The TIDx identifiers represent sequence of visited dense cells during one client session. The colored blocks represent the dense regions. The numbers in the table correspond to dense region identifiers.

Candidate item set can be generated from input dense regions. Please follow the work [Agr94] because of candidate item set definition. For each candidate item set the trajectory ID has to be kept together with the candidates. This defines the TID modification of the apriori algorithm.

The apriori algorithm has also three input parameters. The first parameter is set of frequent 1-patterns, second parameter is the trajectory database represented by sequences of dense regions and the last parameter is candidate item set. The key idea of the apriori algorithm it to generate candidate item set of length *k* from candidate item sets of length *k-1*. Outputs of the apriori algorithm are sets of frequent k-patterns with property parameter called a *support*. We can compute probability for each association rule using the *support* property. The Table 2. shows discovered k-patterns for data from Table 1.

k	Discovered sets of k-patterns
1	{1}, {2}, {3}, {4}, {5}
2	{{1,2}, {1,3}, {1,4}, {1,5}}, {{2,3}, {2,4}, {2,5}}, {{3,4}, {3,5}}, {{4,5}}
3	{{1,2,3}, {1,2,4}, {1,2,5}}, {{1,3,4}, {1,3,5}},{{1,4,5}}, {{2,3,4}, {2,3,5}}, {{2,4,5}}, {{3,4,5}}
4	{{1,2,3,4},{1,2,3,5}}, {{1,3,4,5}},{{2,3,4,5}}
5	{{1,2,3,4,5}}

Table 2. The discovered sets of k-patterns. Each subset corresponds to one association rule.

For example, the first association rule from 3-pattern set is $R_3^1 R_2^2 \rightarrow R_2^3$. The corresponding *support* for each association rule is defined in the next table.

k	Probabilities discovered for association rules
1	{0.23}, {0.23}, {0.15}, {0.23}, {0.15}
2	{{0.30}, {0.20}, {0.30}, {0.20}}, {{0.29}, {0.42}, {0.29}}, {{0.5}, {0.5}}, {{1.0}}
3	{{0.29}, {0.42}, {0.29}},{ {0.5}, {0.5}},

	{{1.0}}, {{0.5}, {0.5}}, {{1.0}}, {{1.0}}
4	{{0.5},{0.5}},{{1.0}},{{1.0}}
5	{1.0}

Table 3. Each association rules has defined a support property obtained from the apriori algorithm. The probabilities are computed from the support properties.

For our example, the first association rule from the 3-pattern set is $R_3^1 R_2^2 \xrightarrow{0.29} R_2^3$.

At this point the association rules and corresponding probabilities are known. These rules can be used to predict the object future location. If overall amount of the trajectory patterns grows, then the prediction process gradually becomes inefficient. This is why the principle of trajectory pattern tree [Jeu08] has to be adapted. Now, the main ideas of the trajectory pattern tree have to be briefly pointed out.

Trajectory pattern tree

The trajectory pattern tree is a variant of signature tree [Mam03] which is dynamically balanced tree designed for signature bitmaps.

The key difference between the signature tree and the trajectory pattern tree is how the construction algorithm of the trees encodes signatures. The trajectory pattern tree encodes a trajectory pattern into a pattern key. The pattern key is designed to efficiently process the prediction query using trajectory pattern tree. The pattern key consists of two parts. The first part is a premise key and the second part is a consequence key. The premise part of the pattern key covers current trajectory pattern of an actual object and the consequence part of the pattern key covers destination dense region.

3.1.3 Premise key

Each dense region R_j^n has its own dense region id which is defined by *n*. We encode each dense region id using a hash function 2^{n-1}. The bit length of each premise key is equal to the number of dense regions.

The premise key of a trajectory pattern includes several dense regions. We use a bitwise operation OR for all current visited dense regions. Every set bit in the premise key represents actual dense region.

3.1.4 Consequence key

The consequence key is constructed as follows. We collect all destinations dense regions from all association rules and assign them an identifier with the same hash function which was used for premise key. The length of the consequence key is equal to the number of destination dense regions. The premise and consequence keys can be generated from association rules from Table 2. If we join the premise and consequence keys together then we get a pattern key.

3.1.5 Pattern key

The pattern key represents a trajectory pattern, which is encoded using premise and consequence keys.

Pattern	Pattern key
{1,2,3,4}	01000 00111
{1,2,3,5}	10000 00111
{1,3,4,5}	10000 01101
{2,3,4,5}	10000 01110

Table 4 Pattern keys created from 4-patterns association rules from Table 2. Numbers in column Pattern correspond to dense region identifiers.

Similarly to the existing approach [Jeu08], we place the consequence key before the premise key. Some generated pattern keys are shown in Table 4. where the red part of the formula is consequence key and the green part is premise key. The premise key is constructed using bitwise operation OR for the first three items of each association rule. The last item in each association rule is destination dense region.

According to [Jeu08] we define some operations over pattern keys, where *pk* is pattern key, *ck* is consequence key and *rk* is premise key:

Union(pk_1, pk_2,..., pk_n): the function return a new pattern key formed as $pk_1|pk_2|...|pk_n$, where the operator | is bitwise OR.

Size(pk): the function return the number of set bits in pk.

Contain(pk_1, pk_2): the function return true when $pk_1\&pk_2 = pk_2$.

Difference(pk_1, pk_2): the function return Size($pk_1 \oplus (pk_1\&pk_2)$).

Intersect(pk_1, pk_2): let $ck_1(ck_2)$ denote the consequence key and $rk_1(rk_2)$ denote the premise key of $pk_1(pk_2)$. If Size($ck_1\&ck_2$) > 0 and Size($rk_1\&rk_2$) > 0 then return true else return false.

3.1.6 Search in trajectory pattern tree

We have to create a predictive key to search future locations of an object in trajectory pattern tree. The predictive pattern key has specific format in our case of virtual environment.

The predictive key is composed from premise and consequence keys as well as the pattern key. The premise key is composed from recent movement of the object. Further the consequence key is composed from dense regions which might we visited in future. The distance of such regions from current object location is less or equal than prediction length.

Definition 7: The prediction length of a predictive key is number of dense regions between current object location and future object location.

Let us consider an example. The currently visited dense regions are identified by sequence $R_3^1 R_2^2$. The prediction length is e.g. two. Now we construct a prediction key for this example. We have two current dense regions $R_3^1 R_2^2$ and we have to predict two dense regions ahead. We can use pattern keys from Table 4.

The bitwise operation OR is applied onto result of functions 2^0 and 2^1 for dense regions id R_3^1 and R_2^2. The result of the operation OR is premise key 00011.

The consequence key can be computed from the pattern key Table 4. First of all we select the pattern keys starting with sequence of dense regions $R_3^1 R_2^2$. This fits in with two first pattern keys from the Table 4. Second we perform OR between consequence keys from the first two pattern keys from Table 4. As a result we get the consequence key 11000. If we put the consequence and premise key together then the prediction key is 1100000011.

At this point the prediction key can be selected as an input of the trajectory pattern tree. The trajectory pattern tree returns all the trajectories (sequences of dense regions) satisfying a condition Intersect(pk, q), where *pk* is the trajectory pattern from a node of the tree and *q* is the input predictive key. The depth first search method is used to find all the intersecting trajectory patterns. We have to select one trajectory pattern with the highest confidence from returned set of candidates.

3.1.7 *Premise similarity*

The trajectory pattern search process gives us number of trajectory patterns. We have to select a trajectory pattern with highest confidence. A method for measure premise similarity is introduced now.

Let us consider the result from trajectory pattern tree is trajectory patterns described by dense regions {1, 2, 3, 4} and {1, 2, 3, 5}. The corresponding trajectory patterns are 0100000111 and 1000000111 (values from Table 4.) and the prediction key is 1100000011 (from previous example). The current object position is R_2^2. The more set bits close to the current object position exist, the more important the dense region is. A weight function assigns a weight to each set bit in the premise key based on its position in the premise key. For example, the weighted function can be linear as follows:

$$\omega_i = \frac{i}{\sum_{j=1}^{Size(rk)} j},$$

where *rk* is the premise key of trajectory pattern. The linear function result is (1/6, 1/3, 1/2, 0, 0) for both trajectory patterns and (1/3, 2/3, 0, 0, 0) for the prediction key. We compare the premise key *rk* of the trajectory pattern *pk* with the premise key *rkq* of the prediction key *q*. The premise similarity is computed as follows:

$$S_r = \sum_{i=1}^{Size(rk\&rkq)} \omega_i \quad (0 \le S_r \le 1)$$

The sum is performed over all set bits in premise key *rk* which is also set in premise key rkq. For our example the premise similarity between *rk* and *rkq* equals $S_r = 1/6 + 1/3 = 1/2$.

The earlier work [Jeu08] defines two algorithms to search the most probable regions. The first one is forward query processing algorithm which is used for near location prediction. We have to take into account the confidence of the association rules. Thus we merge the premise similarity and the confidence together:

$$S_p(pk, q) = S_r \text{ x } c \quad (0 <= S_r <= 1),$$

where *pk* is trajectory pattern key discovered by the apriori algorithm, *q* is prediction key and *c* is confidence for the trajectory pattern key. This equation can be applied to all the candidates from the trajectory pattern search result set. The candidate with the highest S_p will be selected. For our example the S_p for both trajectory patterns key is as follows:

$$S_p(00111, 00011) = 1/2 \text{ x } 0.5 = 1/4$$

The second algorithm to search most probable location is backward query processing. This algorithm is used to process distant time queries. The algorithm is different from the forward query processing, because the recent movements aren't as important as in the short time prediction. This algorithm operates only with intersection between two premise keys and no weights are used. For the consequence key the algorithm works with time relaxation length. In our case this parameter stands for number of dense regions close to the prediction length.

4. USER PROFILES AND TRAJECTORY PATTERNS

We propose a concept of user profile to optimize the process of finding trajectory patterns in large virtual environment in order to improve the efficiency of the above presented trajectory pattern concept.

Definition 8: The user profile is a data structure containing information about each unique user. The content of the user profile is highly application-dependent. Every user profile contains unique user

ID, trajectory pattern tree and application dependent information.

The information from user profile can be used to perform accurately prediction in a shorter time with significant less memory consumption. All movement history represented by trajectory patterns can be categorized into several groups (e.g. student or teacher groups etc.).

In order to achieve the minimal memory consumption of the prediction system, we can create and store one single trajectory pattern tree for each single group. The prediction might be more accurate because the probabilities of the association rules are not affected by other users with significantly different interests.

Let us consider an example of two groups with different count of trajectories. The first group of users has significantly more number of trajectory patterns than the second group. The prediction categorized into second groups is considerably affected by trajectory patterns from the first group without giving any reason.

Further we propose a new approach to solve the case when no pattern is found. If we didn't found any trajectory pattern for object categorized into some of defined groups, we join two similar groups together. Then we perform the search query again. If there are no groups which can be merged together, the algorithm uses a recursive motion function to predict the next object location.

5. RESULTS

No study known to the authors investigates user motion prediction in large distributed virtual environment. Some papers describe process of preloading data to cache memory. This process can accelerate rendering. The earlier work [Chm98] proposes hot regions which affect motion vector of given object. Further work [Var02] uses two view frustums to preload data to cache memory. No work solves the case when the prediction of distant location is needed for optimization the download process.

The prediction can optimize the amount of downloaded data and speed up the process of setting up appropriate level of detail for given object.

For testing purposes we have a terrain scene with relatively small number of cells and we use a synthetic pattern generator to generate dense regions.

The prediction mechanism is used to download textures from network server. Each texture is represented by several files with different resolution. These resolution files are used for texture level of detail purposes.

The two different approaches to download textures from network server were compared. The first method download textures included in view frustum of a given object. Level of detail metric for this approach is defined as distance between object and each cell of the grid. If all textures from current view frustum are downloaded then the system cannot exactly decides which texture at which level of detail should be downloaded further.

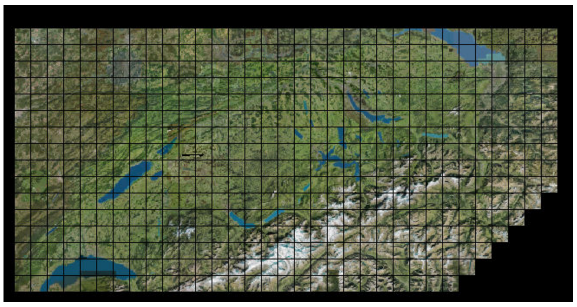
Figure 3 Virtual environment with 16x33 cells (3D triangle model represents large part of Switzerland).

In such case the prediction method can be utilized successfully. We have examined that the prediction accuracy for our synthetic set of trajectory patterns depends on the parameters of the DBSCAN algorithm Eps and minPts. We use Eps=4 and minPts=1. The number of generated trajectory patterns by our synthetic generator is 256 and the number of dense regions discovered by our DBSCAN algorithm is 147. We limit the size of each dense region to five cells because of scaling up the prediction process.

Table 5. show comparison of downloaded textures at different level of detail between the pattern and between the view frustum prediction models.

The number of downloaded textures depends on the object's behavior in the virtual environment. If the user goes faster then the prediction is more useful for lower resolution. It is because the higher resolutions won't be downloaded at time because the scene changes significantly (se Figure 4.).

	Number of downloaded textures				
Texture resolution	256	512	1024	2048	4096
with prediction	276	79	40	19	9
view frustum	394	147	63	29	17

Table 5. Table show number of downloaded textures during the virtual scene walkthrough.

If the user often stops, then the prediction is useful for higher resolution. Let us consider an example.

The user stays on the same place for a while and he gets the full resolution of the visible part of scene. The download process can continue to the nearest not yet available predicted textures.

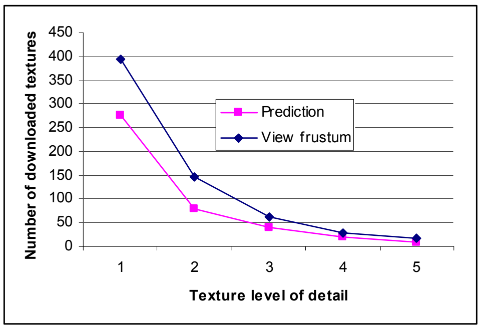

Figure 4 Graph representing the fast walk through the virtual environment. The prediction is most useful for lower resolutions.

6. CONCLUSION

We adapt method for user motion prediction in large virtual environments. We demonstrate that the prediction is meaningful and can increase efficiency of texture selection to download. This is the very contribution of this paper as well as this paper make an attempt to use prediction in 3D virtual environment. The system, however, can still be improved. The further work will be concentrated to improve the process of mining association rules and to examine other method of motion prediction and their adaption to network virtual environments. Furthermore we have to propose an algorithm closely connecting motion function principle with trajectory pattern selection. Further work will be focused on out-of-core rendering algorithms.

7. ACKNOWLEDGEMENTS

This work has is supported by the Ministry of Education, Youth and Sports of the Czech Republic under the research program LC-06008 (Center for Computer Graphics), and by the research project Security-Oriented Research in Information Technology, MSM0021630528.

Special thanks also go to CadWork Informatik AG and CadWork development team in Brno for professional support of this project.

8. REFERENCES

[Agr94] Agrawal, R. and Srikant, R. Fast algorithms for imnining association rules, in VLDB, pp. 487-499, 1994.

[Est96] Ester, M., Kriegel, H.-P., Sander, J. and Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise, in SIGKDD, pp. 226-231, 1996.

[Chm98] Chim, J., Green, M., Lau, R. W. H. Leong, H. and Si, A. On caching and prefetching of virtual objects in distributed virtual environments, ACM Multimedia, pp.171–180, 1998

[Jen04] Jensen, C. S., Lin, D. and Ooi, B. C. Query and update efficient b+-tree based indexing of moving objects. in VLDB, pp. 768-779, 2004.

[Jeu08] Jeung, H., Liu, Q., Shen, H. T., Zhou, X. A hybrid prediction model for moving objects, in ICDE, pp. 236-245, 2008.

[Mam03] Mamoulis, N., Cheung, D. W and Lian, W. Similarity search in sets and categorical data using the signature tree, in ICDE, pp. 75-86, 2003.

[Mam04] Mamoulis, N., Cao, H., Kollios, G., Hadjieleftheriou, M., Tao, Y. and Cheung, D. W. Mining, indexing, and querying historical spatiotemporal data, in SIGKDD, pp. 236-245, 2004.

[Pat04] Patel, J. M., Chen Y, and Chakka, V. P. Stripes: an efficient index for predicted trajectories, in SIGMOD, pp. 635-646, 2004.

[Rab89] Rabiner, L. A tutorial on hidden Markov models and selected applications in speech recognition, Proceedings of the IEEE, vol. 77, pp. 257-286, 1989.

[Sal00] Saltenis, S., Jensen, C. S., Leutenegger, S. T. and Lopez, M. A. Indexing the positions of continuously moving objects, in SIGMOD, pp. 331-342, 2000.

[Tao03a] Tao Y. and Papadias, D. Spatial queries in dynamic environments, TODS, vol. 28, no. 2, pp. 101-139, 2003.

[Tao03b] Tao, Y., Papadias, D. and Sun, J. The TPR*-Tree: An optimized spatiotemporal access method for predictive queries. in VLDB, pp. 790-801, 2003.

[Tao04a] Tao, Y., Faloutsos, C., Papadias, D. and Liu, B. Prediction and indexing of moving objects with unknown motion patterns, in SIGMOD, pp. 611-622, 2004.

[Tao04b] Tao, Y, Kollios, G., Considine, J., Li, F. and Papadias, D. Spatio-temporal aggregation using sketches, in ICDE, p. 214, 2004.

[Var02] Varadhan, G. and Manocha, D. Out-of-core rendering of massive geometric environments,

IEEE Computer Society, pp. 69-76, 2002.

[Yan06] Yang, J. and Hu, M. Trajpattern: Mining sequential patterns from imprecise trajectories of mobile objects. in EDBT, pp. 664-681, 2006.

[Yav05] Yavas, G., Katsaros, D., Ulusoy, 0. and Manolopoulos, Y. A data mining approach for location prediction in mobile environments, Data & Knowledge Engineering, vol. 54, no. 2, pp. 121-146, 2005.