Avatar Path Clustering in Network Virtual Environments

Jehn-Ruey Jiang, Ching-Chuan Huang, and Chung-Hsien Tsai
Department of Computer Science and Information Engineering
National Central University,
Jhongli City, Taiwan, R.O.C.
jrjiang@csie.ncu.edu.tw, qulau0412@gmail.com, chtsai@csie.ncu.edu.tw

Abstract—**With the increase of network bandwidth and the advance of 3D graphics technology, networked virtual environments (NVEs) have become popular recently. Early SIMNET and currently booming massively multiplayer online games (MMOGs), such as Second Life (SE) and World of Warcraft (WoW), are examples of NVEs. Because NVE users' interests or habits may be similar, avatars, or the representative of NVE users, may have similar behavior patterns, which leads to similar motion paths in the NVE. This paper proposes two NVE avatar path clustering algorithms, namely, Average Distance of Corresponding Points-Density Clustering (ADCP-DC) and Longest Common Subsequence-Density Clustering (LCSS-DC). Given avatar paths, both algorithms will produce a collection of path clusters and their representative paths (RPs), which can be used to analyze avatar behaviors for improving NVE design. We take SE user trace data as input of the algorithms to demonstrate their applicability. We also show how to adjust algorithm parameters to obtain high-quality path clustering in terms of silhouette coefficient and cluster coverage.**

Keywords-networked virtual environments, massively multiplyer online games, path clustering, density clustering

I. Introduction

Networked virtual environments (*NVEs*) are virtual worlds full of numerous virtual objects created by computer graphics to simulate various real world scenes for multiple geographically distributed users to assume virtual representatives (or *avatars*) to concurrently interact with each other via network connections. Early SIMNET [1] and currently booming *massively multiplayer online games* (*MMOGs*), such as Second Life (SE) [2] and World of Warcraft (WoW) [3], are examples of NVEs. Some NVEs like SE are constructed for the purpose of social intercourse. Within such NVEs, users may exhibit diverse behaviors of navigating the virtual world and interacting with other avatars. Because of similar personalities, interests, or habits, users may possess similar behavior patterns, which in turn lead to similar motion paths within the virtual world. For example, SE allows users to create fantastic objects and sell them in particular areas. Users with similar interests usually move towards similar areas for shopping similar products and thus may have similar avatar paths.

In this paper, two path clustering schemes, namely *Average Distance of Corresponding Points-Density Clustering* (*ADCP-DC*) and *Longest Common Subsequence-Density Clustering* (*LCSS-DC*), are proposed to group similar avatar paths and find *representative paths* (*RPs*) for them. RPs have many applications; for example, an NVE developer can use them to analyze avatar behavior and to find out popular paths for improving the virtual world design, etc. The two algorithms first divide an avatar path into several *path segments* between two hotspots, where a *hotspot* is defined to be an area that has attracted a large portion of avatars to long stay. The algorithms employ respectively the concept of the average distance of corresponding points (ADCP) [4] and the concept of the longest common subsequence (LCSS) [5] to compute the similarity of path segments, and then use the density-based clustering scheme to cluster similar path segments. To demonstrate the applicability of the algorithms, the avatar location trace data in SE reported in [6] are used as inputs of the two algorithms to output RPs of avatar paths. Furthermore, the outputs are measured in terms of the *silhouette coefficient* (or simply *silhouette*) [7-8] and the *cluster coverage* (or simply *coverage*) to show the output quality. Suggestions about how to achieve better quality of path clustering by adjusting parameters are also given.

The rest of this paper is organized as follows. Section 2 provides an overview of relevant background knowledge, and Section 3 describes the proposed path clustering algorithms. Section 4 shows the experimental results and Section 5 summaries this paper and outlines some future work.

II. Related Work

In this section, we introduce some related research results about motion path similarity, data clustering and path clustering in spite that they are not intended for NVEs. Some path similarity measurement methods are examined, followed by three classes of data clustering algorithms and relevant studies on path clustering.

2.1 Path Similarity

In this subsection we describe two methods of measuring the similarity between motion paths: Average Distance of Corresponding Points (ADCP) and Longest Common Subsequence (LCSS) methods. ADCP [4] is for measuring pairwise similarity of vehicle motion paths (or trajectories) in real traffic video of a cross road scene. It is suitable for paths of similar beginnings and stops; it is thus fit for the paths in traffic video since most paths appear and disappear around the video boundaries. Before path similarity is measured, paths are pre-processed and resampled and indexed at equal space intervals. Two paths with similar beginnings and stops are then represented as two equal-sized sequences of points.

A point from the first path and a point from the second path are regarded as *corresponding points* if they have the same index. The distance of every pair of corresponding points is measured and accumulated for calculating the average distance of the two paths to measure their similarity. ADCP is sensitive to noise (e.g., zigzag segments in a path), so some pre-processes are needed to eliminate the noise by merging data points that are too close. Furthermore, since ADCP requires that two paths for measuring have close beginnings and stops, paths should be aligned by padding additional points to path heads and tails until they reach the boundary.

The other path similarity measuring method is the Longest Common Subsequence (LCSS) [5], which regards a path as a sequence of points for matching. Given two sequences S_1 and S_2, the LCSS method can return the longest subsequence L of matched points of the two sequences, where matched points are points that are close enough (within ε distance) and a subsequence of a sequence S is derived by deleting from S any number of points that are not necessary consecutive (e.g., sequences <A, C, E> and <B, C, D> are both subsequences of the sequence <A, B, C, D, E>). The path similarity is then measured by the ratio $|L| / \min(|S_1|, |S_2|)$. Compared with the ADCP method, LCSS can cope with the problems caused by noises and path un-alignment by allowing some points not to be included in the returned subsequence.

2.2 Data Clustering

In this subsection, we describe some research work about data clustering which may be used to cluster motion paths. As described in [9], data clustering partitions numerous data objects into clusters (or groups) according to their similarity. A *cluster* is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters. There are many data clustering methods proposed in the literature. Below, we introduce three categories of them: partitioning, hierarchical, and density-based methods.

2.2.1 Partitioning Methods

Given a set of n data objects, a partitioning method constructs k partitions of the data, where each partition represents a cluster and $k \leq n$. To be more precise, the method classifies the data into k clusters satisfying the following requirements: (1) each cluster must contain at least one object, and (2) each object must belong to exactly one cluster. After partitioning data objects into k initial clusters, the method uses an iterative relocation technique attempting to improve the partitioning by moving objects from one cluster to another. The general criterion of a good partitioning is that data objects in the same cluster are similar to each other, whereas data objects of different clusters are dissimilar.

The famous *k-means* algorithm [11] is one of the partitioning methods. It first randomly selects k of the data objects, each of which initially represents a *cluster mean*. Each of the remaining data objects is then assigned to the cluster to which it is the most similar on the basis of the distance between the data object and the cluster mean. Afterwards, the new mean for each cluster is re-computed and data objects are re-assigned to clusters. This process iterates until the criterion function, typically defined as the sum of square errors, converges. The *k-medoids* algorithm [8] is analogous to the *k*-means algorithm; however, it uses the *medoid*, the data object that is closest to the mean of the cluster, instead of the mean as the reference of the cluster. Furthermore, it uses the sum of absolute errors as the criterion function. The *k*-medoids algorithm is thus more robust to noise and outliers (i.e., the data objects that do not comply with the general object behavior) than the *k*-means algorithm.

2.2.2 Hierarchical Methods

Hierarchical methods seek to build a hierarchy of clusters of data objects, and they are either *agglomerative* ("bottom-up") or *divisive* ("top-down"). Agglomerative methods, such as BIRCH (Balanced Iterative Reducing and Clustering) algorithm [10], begin by taking each element as a separate cluster and then merge them into successively larger clusters. They are convenient to find and handle the outlier values. On the contrary, the divisive methods begin with the whole data objects in a cluster and proceed to split it into successively smaller clusters. They need more memory than its counterparts to process the clustering. Since a pure hierarchical clustering method cannot perform adjustment once a merging or splitting operation has been executed, its performance may suffers. Some studies try to embed the data relocation concept into hierarchical agglomerative clustering methods to ease the suffering.

2.2.3 Density-based Methods

Density-based methods typically regard clusters as dense regions of data objects in the data space that are separated by regions of low density. In these methods, a cluster is regarded as a region satisfying a certain criterion (e.g., the density of data objects exceeds a threshold). DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [12] and OPTICS (Ordering Points to Identify the Clustering Structure) [13] are two typical methods of this kind. DBSCAN [12] processes data objects one by one and regards an object as a core object to be grown into a cluster of arbitrary shape in a spatial database with noise if the number of the object's nearby objects within a specified radius R exceeds a threshold T. After processing all data objects, those not belonging to any cluster are regarded as noises. A disadvantage of DBSCAN is that it is hard to find proper parameters, such as R and T, for clustering. To eliminate the drawback, OPTICS [13] computes the cluster ordering, instead of the explicit data clustering, for automatically and interactively extracting clustering information like cluster centers, etc.

2.3 Path Clustering

Some papers address the problem of path clustering for paths in different applications, such as hurricane (or typhoon) tracks [14], animal movements [14] and vehicular trajectories [15]. The paper [14] proposes a method called TRACLUS using the *partition-and-group* concept for clustering paths. The method first divides paths into smaller segments with the *minimum description length* (*MDL*) principle and then clusters similar segments into a cluster, in which the similarity of paths is measured by the weighted distance of the perpendicular, the parallel and the angle distances. Each cluster has a *representative path* calculated by averaging characteristic parameters of paths in the cluster.

The paper [15] proposes an online path clustering method to build path clusters for a video surveillance system as path data are acquired by the system. Paths are clustered online and clusters are organized in a tree-like structure augmented with probability information. The paper defines the distance of a path to a cluster to be the mean of the normalized distances of very point on the path to the nearest point of a path in the cluster found inside a temporal window. When a path is detected, a new cluster is constructed by creating a tree consisting of a single node if the path is not sufficiently near (or with short enough distance) to any known clusters. Otherwise, the path is added into the nearest cluster by updating the corresponding tree structure of the cluster. As the path advances, the distance from it to the associated cluster is monitored to see if the path is moving away from the cluster. If the distance keeps growing over an amount of time, the cluster is divided into two subclusters by a cluster tree splitting procedure.

III. Proposed Algorithms

3.1 Pre-processing

In NVEs, two avatars can see each other if they are within each other's *AOI (area of interest)*, which is usually assumed to be a circle of radius r centered at the avatar. Furthermore, two avatars have overlap in their AOIs or have some common views if they are within $2r$ distance. Therefore, the portions of two paths are regarded as similar if they are of distances less than the AOI diameter (i.e., $2r$). In the proposed algorithms, the avatars are assumed to have the same AOI radius r and the whole NVE virtual world is divided into several numbered *cells*, which are squares of the side length $2r$. It is likely that two avatars within the same cell can see each other or have overlap in their AOIs. Note that cells are said to be adjacent if they share common edges or points in the boundary. Therefore, a cell may have up to 8 adjacent cells.

Before being clustered, avatar paths need some preprocessing since they usually have different length and different start and end points. In practice, paths are divided into *path segments* of which endpoints are hotspots, where a *hotspot* is a cell whose accumulated period of avatars staying is larger than those of all its adjacent cells. Note that a *loop*, which is a path with the same beginning and stopping points, is not regarded as a path segment. For example, in Figure 1, the solid avatar path is divided into 5 path segments and the dotted path is a loop and should not be regarded as a path segment. After dividing paths into path segments, the proposed algorithms can cluster them. Below, a path segment is simply called a *path* and the set of all path segments is simply called the *path set*.

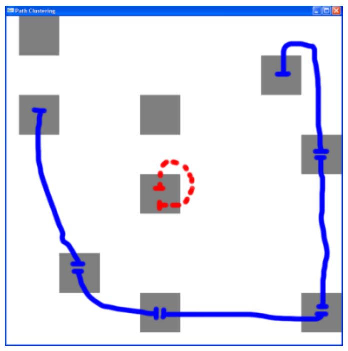


Figure 1. Illustration of dividing paths into path segments by hotspots

3.2 ADCP-DC Algorithm

ADCP-DC algorithm shown in Figure 2 is based on the average distance of corresponding points on two paths to measure the similarity of the paths. Since avatars may move at random directions and random speeds, and path data are usually sampled per time ticks (e.g., per 10 seconds in [6]), even similar paths may have different number of sample points, leading to the difficulty of point data comparison. To overcome the difficulty, ADCP-DC specifies the *number of corresponding points* for paths. For a certain path P of sample points $v_0, v_1, \ldots, v_n$, the *path length* of P is defined to be $\sum_{i=1}^{n} d_i$, where d_i is the distance from v_{i-1} to v_i. Similarly, the length of sample point v_j is defined to be $\sum_{i=1}^{j} d_i$ (note that the length of v_0 is defined to be 0). By dividing the path length and the number of corresponding points, the distance d between two consecutive corresponding points on the path is derived. By the distance d, the length of the corresponding point of index k is defined to be $k \times d$ (note that the first corresponding point is indexed by 0).

Let the length of a corresponding point v be L_v. The coordinate (or location) of v can be calculated by interpolation of the coordinates of the two consecutive sample points u and w which enclose v, where u and v are respectively of length L_u and L_w, and $L_u \le L_v \le L_w$. Let the coordinates of u and w be (x_u, y_u) and (x_w, y_w), respectively. The coordinate (x_v, y_v) of v can be calculated by Eqs. (1) and (2). Note that $v=v_0$ (resp., $v=v_n$) if v is the first (resp., last) corresponding point.

$$x_v = x_u + \frac{L_v - L_u}{L_w - L_u} \times (x_w - x_u) \tag{1}$$

$$y_v = y_u + \frac{L_v - L_u}{L_w - L_u} \times (y_w - y_u) \tag{2}$$

The average distance of corresponding points (ADCP) of two paths can be calculated by first deriving the distance of each pair of corresponding points and then averaging all derived distances. ADCP is used to measure the similarity of two paths for clustering paths. In light of the concept of DBSCAN, a density-based clustering method is employed to cluster paths by specifying the *similar path radius R* and the *minimum number M of paths in a cluster*. In ADCP-DC algorithm, R is set to be the AOI radius for the following reasons: (1) any pair of paths in the same cluster will have ADCP value less than $2R$; (2) two avatars with AOI overlap are of distance less than 2*R* will have some similar view and can be regarded as similar paths.

In ADCP-DC algorithm, two paths are regarded as similar paths if their ADCP value is less than *R*, which is set to be the AOI radius. If the total number of the similar paths of a path exceeds M, the path is regarded as a *core path* and is associated with a *similar path set* (*SPS*). The core path, say p_1, with the most similar paths is first elected as the first representative path, and SPS_1 is regarded as the first cluster, where SPS_1 is the set containing p_1 and all its similar paths. All members of SPS_1 are then excluded from the *SPS* of every remaining core path. Afterwards, the remaining core path with the most similar paths is selected as the second representative path, and its associated *SPS* is the second cluster. The procedure proceeds until no core path exits.

```
Algorithm: ADCP-DC
Input: path set PS, similar path radius R (set to be AOI radius),
minimum number M of paths in a cluster
Output: path cluster collection PCC and representative paths
Var:
$CPS=\varnothing$;  //CPS: core path set
$RPS=\varnothing$;  //RPS: representative path set
$SPS_i=\{P_i\}$, for $1\le i\le n$;  //$SPS_i$: similar path set of path $P_i$
FOR each unordered pair of distinct paths $P_i$ and $P_j$ in PS
  Compute $ADCP_{ij}$ of $P_i$ and $P_j$
   IF $ADCP_{ij} < R$ THEN
      $SPS_i=SPS_i\cup\{P_j\}$;
      $SPS_j=SPS_j\cup\{P_i\}$;
FOR each path $P_i$
   IF $|SPS_i| \ge M$ THEN $CPS=CPS\cup\{P_i\}$;
WHILE $CPS\neq\varnothing$
   Select from CPS path $P_i$ with the largest $SPS_i$
   $CPS=CPS-\{P_i\}$;
   IF $|SPS_i| \ge M$ THEN
      //$SPS_i$ is a path cluster and $P_i$ is a representative path
      $PCC=PCC\cup\{SPS_i\}$;
      FOR each path $P_j$ in CPS
         $SPS_j = SPS_j - SPS_i$;
RETURN PCC and associated representative paths
```

Figure 2. ADCP-DC Algorithm

3.3 LCSS-DC

In LCSS-DC algorithm shown in Figure 3, the entire virtual world is first divided into numbered square cells whose length is of the AOI diameter. According to the cell numbers that the sample points of a path resides in, the path is represented by a sequence of cell numbers. Note that consecutive identical cell numbers in the sequence will be merged to be one number. For example, in Figure 3(a), path A is represented as <60, 61, 62, 63, 55, 47, 39, 31, 32>, and path B, <60, 61, 62, 54, 62, 63, 64>.

The similarity of two paths is measured by the longest common subsequence (LCSS) of the cell number sequences of the paths. Each cell number in the sequence is associated with the location of the sample data residing within the cell; a merged cell number is associated with the centroid of the locations of the sample data residing within the cell. Two cell numbers are regarded common if they are identical or their associated cells are adjacent in the virtual world and their associated locations are with the distance less than the AOI radius. For example, paths A and B in Figure 3(a) have the LCSS of <60, 61, 62, 63>.

After calculating the LCSS of two paths, clustering is then performed. LCSS-DC uses a pair of *similar path thresholds* TH_a *and* TH_b, instead of the similar path radius *R* used in ADCP-DC, as parameters of clustering. Path P_i takes path P_j as its similar path if Eqs. (3) and (4) are satisfied.

$$\frac{|LCSS_{ij}|}{|Seq_i|} \ge TH_a \tag{3}$$

$$\frac{|LCSS_{ij}|}{|SSeq_{ji}|} \ge TH_b \tag{4}$$

In Eqs. (3) and (4), Seq_i and Seq_j are the cell number sequences of P_i and P_j, respectively, $LCSS_{ij}$ is the longest common subsequence of Seq_i and Seq_j, and $SSeq_{ji}$ is the shorted subsequence of path P_j containing the whole $LCSS_{ij}$.

```
Algorithm: LCSS-DC
Input: path set PS, similar path thresholds $TH_a$ and $TH_b$,
minimum number M of paths in a cluster
Output: path cluster collection PCC and representative paths
Var:
   $CPS=\varnothing$;  //CPS: core path set
   $SPS_i=\{P_i\}$, for $1\le i\le n$;  //$SPS_i$: similar path set of path $P_i$
FOR each ordered pair of distinct paths $P_i$ and $P_j$ in PS
   Compute $LCSS_{ij}$ of $P_i$ and $P_j$
   IF $(|LCSS_{ij}|/|Seq_i|) \ge TH_a$ and $(|LCSS_{ij}|/|SSeq_{ji}|) \ge TH_b$
      $SPS_i = SPS_i \cup\{P_j\}$;
FOR each path $P_i$
   IF $|SPS_i| \ge M$ THEN $CPS=CPS\cup\{P_i\}$;
WHILE $CPS\neq\varnothing$
   Select from CPS path $P_i$ with the largest $SPS_i$
   $CPS=CPS-\{P_i\}$;
   IF $|SPS_i|\ge M$ THEN
      //$SPS_i$ is a path cluster and $P_i$ is a representative path
      $PCC=PCC\cup\{SPS_i\}$;
      FOR each path $P_j$ in CPS
         $SPS_j = SPS_j - SPS_i$;
RETURN PCC and associated representative paths
```

Figure 3. LCSS-DC Algorithm

Unlike ADCP-DC, LCSS-DC checks the similarity for each ordered pair of paths. Like ADCP-DC, LCSS-DC also adopts the density-based clustering mechanism for clustering paths. When a path has many enough similar paths, it is regarded as a core path and becomes a candidate of representative paths. Note that the similarity relationship between two paths is asymmetric. To take paths in Figure 3(a) as examples, path B may take path A as a similar path, but not vise versa. The asymmetry is to avoid the case that dissimilar paths appear in the same cluster. For example, if path A in Figure 3(b) takes both paths B and C as its similar paths, then path A is likely to be the representative path of the cluster covering paths A, B and C. It is obvious that such a cluster contains two very dissimilar paths, B and C.

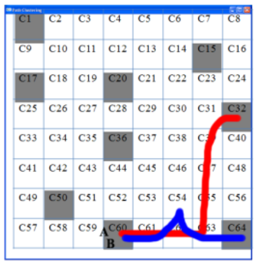
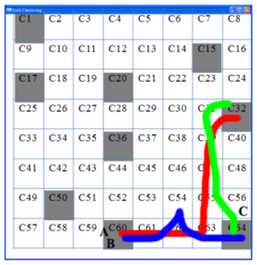
(a)  (b) 

Figure 4. Examples of paths represented as cell sequences

IV. Performance Evaluation

4.1 Setting and Metrics

To demonstrate the applicability and performance of the algorithms, the avatar location trace data collected in SE hours [6], as shown in Figure 5, are used as inputs of the two proposed algorithms. The avatar locations are sampled per 10 seconds for four regions, each of which is a 256x256 (unit^2 or m^2) area of the virtual world with the avatar AOI radius of 16 (unit or m). Each data entry has the following fields: date, time, avatar ID, and avatar location. This paper adopts the trace data for Freebies region in SE for performance evaluation, and adopts the parameter setting shown in Tables I and II. However, several parameters are variables. Below, we will show how to tune those parameters to achieve better clustering quality.

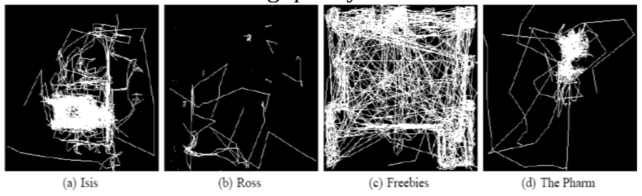


Figure 5. The trace data of avatar paths in different regions of SE

TABLE I. THE PARAMETER SETTING OF ADCP-DC

Algorithm	ADCP-DC
similar path radius *R*	16 (AOI radius)
number of corresponding points	variable
minimum number *M* of paths in a cluster	variable

TABLE II. THE PARAMETER SETTING OF LCSS-DC

Algorithm	LCSS-DC
cell diameter	32 (AOI diameter)
similar path threshold TH_a	variable
similar path threshold TH_b	variable
minimum number *M* of paths in a cluster	variable

A good clustering should have the contrast property that intra-cluster similarity is high and inter-cluster similarity is low. We can apply Eq. (5) of *silhouette* [7] to measure the quality of the clustering.

$$S_i = 1 - \left(\frac{b_i - a_i}{\max(a_i, b_i)}\right) \quad (5)$$

In Eq. (5), S_i is the silhouette value of a path P_i, a_i is the average dissimilarity of P_i and all other paths in the cluster of P_i, and b_i is computed as follows. For each cluster in which P_i is not a member, compute the average dissimilarity between P_i and all paths in the cluster. Then find the cluster generating the largest average dissimilarity, which is b_i. We can observe that the value of silhouette is between 1 and -1. If the silhouette of P_i is greater, then P_i is more similar to the paths in the cluster of P_i, and is more dissimilar to the paths in other clusters. By Eq. (6), we can calculate silhouette values of all paths and average them to obtain the overall silhouette value. As shown in [8], silhouette can be used to measure the accuracy of a clustering and the overall silhouette value over 0.7 can be regarded as the achievement of high accuracy.

In addition to silhouette, another metric called *coverage* is used for assessing the cluster results. This metric concerns the ratio of paths that belong to clusters. The coverage value can be computed according to Eq. (6).

$$\text{coverage} = \frac{\text{the number of paths in clusters}}{\text{the total number of paths}} \quad (6)$$

4.2 ADCP-DC Performance

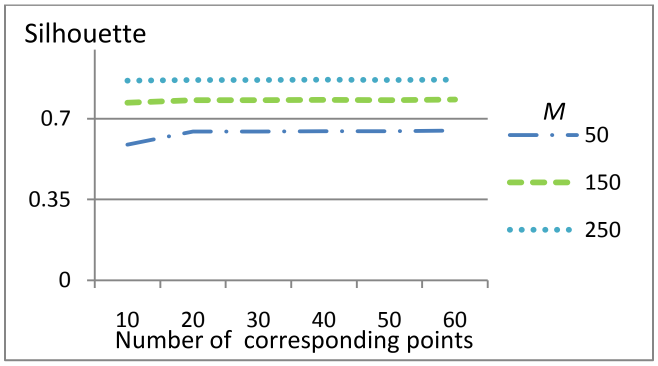


Figure 6. The silhouette of ADCP-DC algorithm

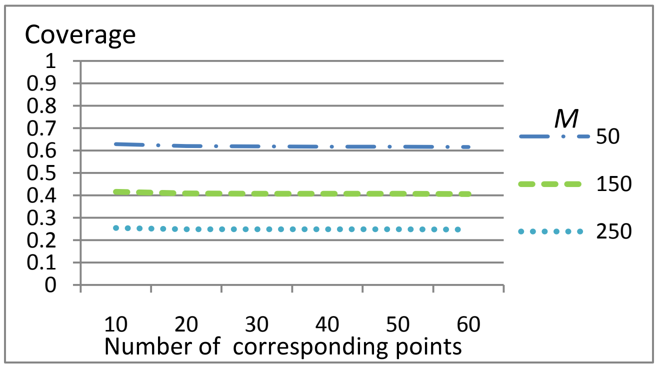


Figure 7. The coverage of ADCP-DC algorithm

As shown in Figures 6 and 7, when the number of corresponding points increases, the silhouette of ADCP-DC algorithm slightly increases and the coverage slightly decreases. In addition, when the minimum number *M* of paths in a cluster increases, the silhouette increases but the coverage decreases. This is because when *M* grows, it requires more paths to form clusters and some paths thus cannot be included in any clusters, leading to the higher silhouette and lower coverage. Based on the experiment results, when the minimum number of paths in a cluster is 150, the silhouette is greater than 0.7, the suggested value mentioned in [8], and the coverage is about 0.41.

4.3 LCSS-DC Performance

Experiment results reveals that when the minimum number M of paths in a cluster is less than 200, the silhouette of LCSS-DC algorithm is below the suggested value 0.7. Therefore, only the results for *M*=200 and *M*=300 are shown in the paper. By Figures 8 and 9, it can be observed that when similar path thresholds TH_a and TH_b grow, the silhouette increases but the coverage decreases. As shown in Figure 8, when the minimum number *M* of paths in a cluster is set to be 200, and TH_a is set to be over 0.74, and TH_b is set to be over 0.50, the silhouette is over the suggested value 0.7 and the coverage is about 0.54 or less. As shown in Figure 9, when *M* is set to be 300, and TH_a is set to be over 0.68, and TH_b is set to be over 0.65, the silhouette is over 0.7 and the coverage is about 0.41 or less.

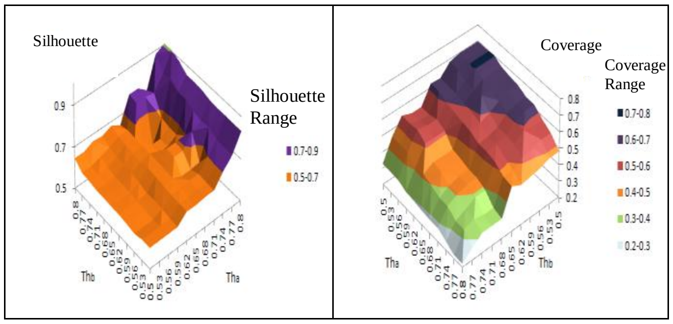


Figure 8. The silhouette and coverage of LCSS-DC algorithm when the minimum number *M* of paths in a cluster is 200

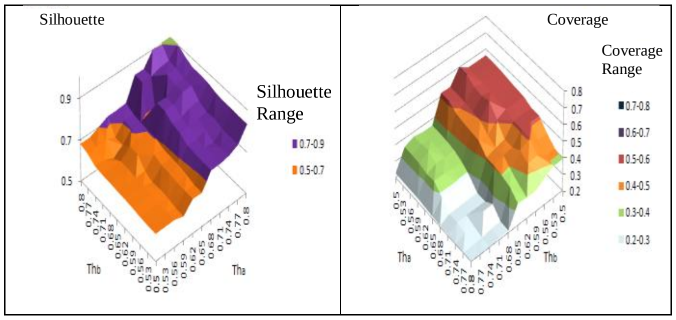


Figure 9. The silhouette and coverage of LCSS-DC algorithm when the minimum number *M* of paths in a cluster is 300

V. CONCLUSION

This paper proposed two algorithms, namely Average Distance of Corresponding Points-Density Clustering (ADCP-DC) and Longest Common Subsequence-Density Clustering (LCSS-DC), to cluster avatar motion paths in NVEs. Avatar paths are first divided into segments by hotspots and then processed by the proposed algorithms to derive a collection of path clusters and associated representative paths, which can be used to observe avatar behavior or to improve the NVE design. The proposed algorithms are applied to process the avatar path data of SE to demonstrate their applicability and quality in terms of silhouette and coverage. At present, we are planning to apply them to avatar path data of other NVEs like WoW. We are also planning to design other clustering algorithms to cluster avatar paths in NVEs.

REFERENCES

[1] D. C. Miller, and J. A. Thorpe, "SIMNET: the Advent of Simulator Networking," Proc. IEEE, pp. 1114-1123, 1995

[2] Secnod Life, http://secondlife.com/.

[3] World of Warcraft, http://www.worldofwarcraft.com/.

[4] Z. Fu, W. Hu, and T. Tan, "Similarity Based Vehicle Trajectory Clustering and Anomaly Detection," Proc. 5th ICIP, pp. 602-605, 2005.

[5] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering Similar Multidimensional Trajectories," Proc. 18th Intl. Conf. on Data Engineering (ICDE'02), pp. 673-685, 2002.

[6] H. Liang, I. Tay, M. F. Neo, W. T. Ooi, and M. Motani, "Avatar Mobility in Networked Virtual Environments: Measurements, Analysis, and Implications," Multimedia Tools and Applications, Vol. 45, pp. 163-190, 2009.

[7] R. Ng, and J. Han, "Efficient and Effective Clustering Method for Spatial Data Mining," Proc. 20th VLDB Conference, pp. 144-155, 1994.

[8] L. Kaufman, and P. J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, Wiley, 1990.

[9] J. Han, and M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann, 2000.

[10] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases," Proc. ACM SIGMOD Conference on Management of Data, pp. 103-114, 1996.

[11] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," Proc. 5th Berkeley Symp, pp. 281-297, 1967.

[12] M. Ester, H.P. Kriegel, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," Proc. 2th International Conference on Knowledge Discovery and Data Mining (KDD 96), pp. 226-231, 1996.

[13] M. Ankerst, M. Breunig, H.P. Kriegel, and J. Sander, "Optics: Ordering Points to Identify the Clustering Structure," Proc. ACM SIGMOD International Conference on Management of Data, pp. 49-60, 1990.

[14] J. Lee, J. Han, and K. Whang, "Trajectory Clustering: A Partition-and-Group Framework," Proc. SIGMOD International Conference on Management of Data, pp. 593-604, 2007.

[15] C. Piciarelli, and G. L. Foresti, "On-line Trajectory Clustering for Anomalous Events Detection," Pattern Recognit. Lett., Vol. 27, No. 15, pp. 1835-1842, 2006.