Multimed Tools Appl
DOI 10.1007/s11042-012-1052-x

Automatic player behavior analysis system using trajectory data in a massive multiplayer online game

Shin-Jin Kang · Young Bin Kim · Taejung Park · Chang-Hun Kim

Abstract This paper presents a new automated behavior analysis system using a trajectory clustering method for massive multiplayer online games (MMOGs). The description of a player's behavior is useful information in MMOG development, but the monitoring and evaluation cost of player behavior is expensive. In this paper, we suggest an automated behavior analysis system using simple trajectory data with few monitoring and evaluation costs. We used hierarchical classification first, then applied an extended density based clustering algorithm for behavior analysis. We show the usefulness of our system using trajectory data from the commercial MMOG World of Warcraft (WOW). The results show that the proposed system can analyze player behavior and automatically generate insights on players' experience from simple trajectory data.

Keywords Trajectory clustering · Behavior analysis · World of Warcraft · MMORPG · MMOG

1 Introduction

Recently, the number of MMOGs has increased exponentially on various platforms, but their development process has not kept pace. Many parts of the development process are still performed manually. Especially in the field of game content development, game designers have difficulty making correct decisions without reliable player feedback.

S.-J. Kang
School of Games, Hongik University, Korea/NCsoft, Samsung-dong, Seoul, Korea
e-mail: directx@hongik.ac.kr

Y. B. Kim · T. Park · C.-H. Kim (✉)
Department of Computer Science, Korea University, Anam-dong Seongbuk-gu, Seoul, Korea
e-mail: chkim@korea.ac.kr

Published online: 14 March 2012

In a typical MMOG design process, game designers specify expectations in player behavior on each game map, and then proceed with subsequent work based on such expectations. Terrain and entity data are generated based on the expectation of player behavior. If these expectations were correct, no modification would be necessary. Unfortunately, such a design process based on human hunches—even when they come from highly experienced human game designers—often causes discrepancies between the designs and reality. Figure 1 illustrates the problematic cases caused by wrong assumptions in MMOG design processes. To address the discrepancy between expectations and reality, game developers should modify game content even after release; this modification is highly labor-intensive and inevitably increases costs. To prevent such problems, it is common for many testers to participate in feedback processes relying on manual monitoring and time-consuming game analysis.

For last ten years, MMOGs have achieved major technical improvements in rendering, autonomous agent and network load balancing. Most of these techniques focused on real-time interaction of MMOG, therefore MMOG could show high quality visual content in real-time. However, game design in MMOG still remains subjective and intuitive. To increase efficiency of game design, several monitoring and visualization techniques are proposed from HCI and computer graphics society [18]. These techniques support useful visualization but they have limitations that require high evaluation cost. Various on-line player evaluation models are proposed in the social science society [4], but these models also require intensive manual monitoring and evaluation. Besides, these are technically difficult to apply for commercial games, because they assume that there is no limitation in acquiring parameters for MMOG. In practice, accessing parameters in MMOG is very restricted because of degeneration in system performance, anti-hacking and business issues, especially for server-side parameters of commercial MMOG.

The purpose of this paper is to present a behavioral analysis system for MMOG development. We believe that the value of proposed system is in this framework for behavioral analysis satisfying following three conditions. (1) The system should support an automatic monitoring system to reduce manual monitoring cost. (2) The system should support automatic behavior evaluation which gives reliable

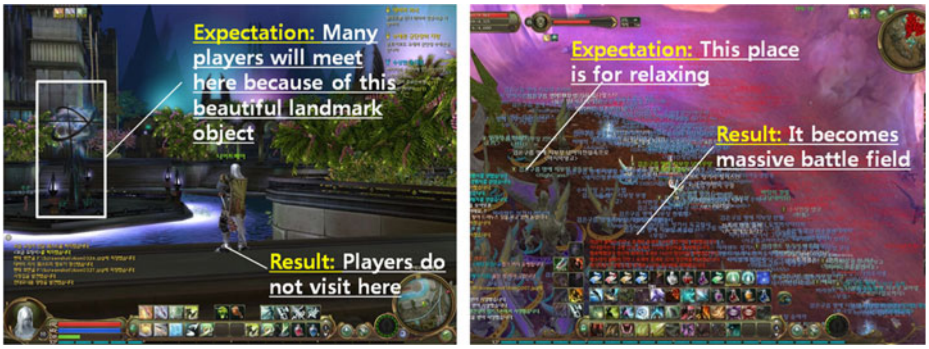

Expectation: Many players will meet here because of this beautiful landmark object
Result: Players do not visit here
Expectation: This place is for relaxing
Result: It becomes massive battle field

Fig. 1 Discrepancy cases in the map of a commercial MMOG (images from NCsoft AION)

and objective information. (3) The system should use few numbers of parameters and they are easily accessible with little performance burdensome.

These conditions are not fully satisfied in previous works from multidisciplinary research fields. To the best of our knowledge, no work has been done on trajectory research with respect to behavior analysis in MMOGs. To this end, we suggest a new concept, "segment of interest" (SOI). To find out the SOI, we combine two techniques which are newly designed; (1) a partitioning algorithm for the SOI classification and (2) an extended DBSCAN algorithm for flexible clustering. Experimentally, our system was able to analyze players' behavior automatically. The gathered information from our system can be utilized as reference data to improve the automatic process of MMOG content development.

2 Related work

2.1 Game play visualization and evaluation

Player feedback data play an important role in the development of MMOGs. It is necessary for a game designer to know player behavior in order to check out the current player experience and determine player demand. Visualization techniques have recently been applied to discover player behavior in detail. Most visualization work focuses on player trajectory or time series data. Borner et al. [5] visualized the evolution of communities in a multiuser online world. Their system records spatial and temporal player interactions such as navigation, object manipulation, and communication. Hoobler et al. [16] focused on playing strategies in a combat game. Their system traces players' trajectories, fire actions, and statuses. These data are visualized with highlighted player positions, paths, fire traces, the occupancy map, and fields of view. The result is drawn on screen-aligned icons and 2D overlays that are updated with every frame. Chittaro et al. [7] suggested visual analysis of the navigation patterns of moving entities, such as players and nonplayer characters (NPCs). Their system provides a set of interactive visualizations that highlight the interesting navigational behavior of single or groups of moving entities. It helps to improve the design of the virtual world and to study player navigational behavior. Thawonmas et al. [24, 25] focused on visualizing player behavior based on locating clusters of similar movement patterns. DeRosa [9] described BioWare's system for monitoring player activity by using passive tracking systems to measure and improve player experience. Recently, Kim et al. [18] combined the collection and analysis of behavioral instrumentation with HCI methods to develop a system for tracking real-time player experience.

Once these visualized data are acquired, evaluating how people play a game is another important process in MMOG development. Various player behavior evaluation models have been suggested. Bartle's taxonomy [4] is a player-representative evaluation model for MMOGs. He suggested that online game players can typically be identified based on their behavior as achievers, explorers, killers, or socializers. Matsumoto et al. [20] suggested the classification of players in MMOGs using hidden Markov models based on player action sequences. Ducheneaut et al. [10] tracked World of Warcraft (WOW) players to analyze their gaming experience: how long they played and the classes and races they preferred. Cowley et al. [8] evaluated the player's experience using a model derived from flow theory. Results for dynamic player modeling are also recorded in other work [6, 17].

In general, previous behavior evaluation methods trace high-dimensional data for evaluating player behavior. These data are difficult for computers to interpret automatically. Therefore, additional intensive analysis is required to discover knowledge about player behavior. This paper suggests an automatic player behavior analysis system based on the trajectory clustering method. Trajectory clustering identifies the sparse and crowded places in the game world, and so discovers the overall distribution patterns of the data set. Additionally, the derived clusters can be visualized more effectively with various techniques.

2.2 Clustering algorithms

A number of clustering algorithms have been suggested in the data mining field. Generally, they can be classified into four categories: (1) hierarchical algorithms, (2) density-based algorithms, (3) grid-based algorithms, and (4) partitioning algorithms. Hierarchical clustering algorithms work by grouping trajectory points into a hierarchical data structure according to similarity. The results of hierarchical clustering are usually represented as a binary tree. A number of hierarchical clustering techniques have been proposed. Typical examples include CURE [14], ROCK [13], CLIQUE [2], and BIRCH [26]. CURE and ROCK are both centroid-based hierarchical clustering algorithms, and as such are unable to identify arbitrary cluster shapes. By contrast, BIRCH and CLIQUE do not predetermine the number of clusters but rather choose the appropriate cluster centers in an initial step and employ a suitable similarity measure according to the cluster shapes present in the data.

Density-based clustering algorithms group neighboring data objects into clusters based on local conditions. These algorithms allow clustering to be performed in one scan of the data set. DBSCAN [11] is an algorithm representative of this group. It regards clusters as dense regions of objects in the input space that are separated by regions of low density. DBSCAN's basic idea is that the density of points in a radius around each point in a cluster has to be above a certain threshold. WaveCluster [23] is a density- and grid-based clustering method that uses a wavelet transform of the feature space. It is computationally effective but can only be applied to low-dimensional data. OPTICS [3] is a clustering algorithm that computes an augmented cluster ordering for automatic and iterative clustering analysis.

Grid-based clustering algorithms divide the input space into rectangular grids and then combine adjacent high-density cells to form clusters. These methods do not suffer from the nearest-neighbor problem in high-dimensional spaces. Examples include STING [12], DENCLUE [15], and MAFIA [21]. These have advantages for rapidly discovering clusters of any shape. In general, grid-based clustering algorithms are effective with low-dimensional input spaces. As the dimensionality of the space increases, these algorithms face exponentially high computational costs for finding adjacent high-density cells to form clusters.

Partitioning algorithms partition the objects into a certain number of clusters. The clusters are formed in order to optimize an objective criterion such as distance.

Each object is assigned to the closest cluster. A cluster is typically represented either by the mean of the objects assigned to the cluster or by one representative object of the cluster. CLARANS [22] is a partitioning clustering algorithm developed for large data sets that uses a randomized and bounded search strategy to improve the scalability of the k-medoid approach.

Our purpose is to identify the behavioral meaning of player trajectories. Previous clustering algorithms used only the trajectory points to analyze movement. Therefore positional information was the only available parameter for clustering. However in MMOG, even a single trajectory can imply various behavioral meanings. In this regard, we need to consider various parameters related to player behavior for better and richer behavioral analysis. For this purpose, we suggest a modified DBSCAN algorithm that uses a more sophisticated clustering approach.

3 Trajectory data for behavior evaluation

3.1 Behavior-based segmentation

MMOG players have various purposes in MMOG world. They show many behaviors based on their preferences and interests. Table 1 shows sequential behaviors of MMOG players and their evaluations examples. If developers can understand the every player's style, they will be able to make more player-oriented content in the MMOG world. The proposed system focuses on the behavioral meaning of a trajectory for behavior analysis of MMOG players. Thus, we define the *segment of interest* (SOI) in the trajectory. Previous trajectory clustering methods have assumed that the sampler performs only movement in all trajectories. Therefore, all other meaningful behavior in the trajectory is ignored. This paper suggests a behavior-based segment approach. The MMOG allows for a wide range of possible behaviors, including combat, gathering, item production, goods trading, and social behavior, to offer an immersive feeling for multiple players. Accordingly, each trajectory includes detailed subtrajectories with diverse internal meanings in terms of behavior.

Table 1 Sequential behaviors and player evaluation examples *(time)*

Player ID	B_0	B_1	B_2	B_3	$B_{...}$	Player evaluation
000001	Log-in *(12:00)*	Monster hunting at zone No. 48 *(12:00–13:17)*	Moving *(13:17–13:30)*	Monster hunting at zone No. 45 *(13:30–15:00)*	....	Combat-oriented player
000002	Log-in *(20:45)*	Chatting at zone No. 15 *(20:45–21:15)*	Log-out *(21:15)*	.....	....	Social-oriented player
....	....	....	....	....	....	....
010213	Log-in *(15:30)*	Gathering at zone No. 21 *(15:30–14:15)*	Moving *(14:15–14:30)*	Selling at zone No. 22 *(14:30–15:30)*	....	Commerce-oriented player

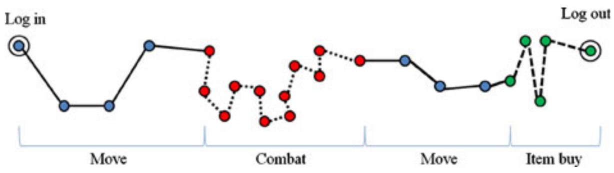


Fig. 2 Behavior-based segment of interest (SOI) concept

Our system uses these trajectory subsegments to analyze player behavior. Figure 2 illustrates our behavior-based segmentation concept.

3.2 Trajectory pattern observation

Figure 3 shows three observed trajectory patterns in WOW. The blue line is the trajectory, and the red line is the player's view direction for a period of time. In Fig. 3a, all adjacent trajectories have similar directions, indicating a very high directional similarity. The view direction is highly correlated with the trajectory direction, and the rate of direction change of both the view and the trajectory is low. By observation, we could classify these trajectories as mainly used for movement behavior. In Fig. 3b, the rate of direction change of both the view and the trajectory is very high, and the directional correlation between the two is very low. Additionally, there are many intersections of the trajectories. By observation, these trajectories generally appear during combat or social behavior. In Fig. 3c, there are a small number of trajectories at low density. These trajectories are difficult to group, and they decrease over time. We could assume that these trajectories arise when players encounter content that they dislike.

The research goal of this paper is to identify meaningful behavioral information from raw trajectory data. To achieve this, every subtrajectory needs to be classified into a certain SOI. The MMOG world is relatively static compared with the real world. Most of environment objects are stationary in the MMOG world because of the high computational cost for dynamic agent pathfinding. Players usually perform similar behaviors under these environmental conditions. Therefore, some correlated play patterns can be found as a grouped trajectory in a specific region of the game world. Based on this approach, the proposed system classifies every segment by region first, to determine their behavioral meaning approximately.

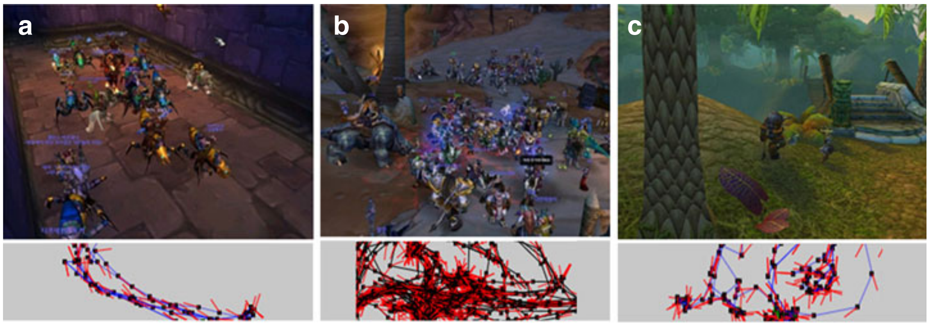


Fig. 3 Trajectory observation patterns in the commercial MMOG WOW

3.3 Sampling features for behavior evaluation

Our purpose is to have the SOIs automatically extracted from the trajectory for behavior evaluation. Behavior evaluation is a high-level abstraction process based on individual player experiences in the MMOG. Usually, this type of analysis requires high-dimensional features and expert knowledge about MMOG play. In this paper, we try to minimize the complexity of sampling data for evaluation automation. For the trajectory sampling data, our system traces three basic trajectory parameters: (1) position P, (2) view direction θ, and (3) initial starting time t_0. Because the trajectory is sampled at static time steps, the system identifies the player's position and view direction at specific times. Every player has different initial starting time because they log-in to the MMOG at different times.

In this paper, player behavior is categorized into four behaviors: (1) social, (2) combat, (3) movement, and (4) idling. Compared with previous behavior analysis systems, our system has advantages in using only a few sample parameters for analyzing player behavior. This reduces the dimensional complexity in the analysis and increases the system's scalability for various MMOG frameworks.

3.4 Feature selection for SOI

Our system uses artificial neural networks (ANNs) as classifiers. Determination of the input parameters for an ANN depends entirely on the judgment of the modeler. As the number of input parameters increases, the training time for the resulting ANN increases exponentially. Moreover, including a larger number of input parameters does not guarantee better performance. In this paper, we employ a genetic algorithm (GA) to determine optimal input parameters that yield the best training performance of the ANN. We adopt a GA because of its ability to search large spaces, about which little is known a priori, efficiently. It is also relatively insensitive to noise, enabling a more robust feature selection strategy [1]. We design the GA with the following rules for the ANN classifier.

– **Encoding:** Encoding is a way of encoding problem solutions into bit strings called the 'chromosomes'. We map each candidate parameter directly to a gene in a chromosome, with the gene value set to 0 or 1 depending on whether the particular parameter is used as input to the ANN. Table 2 shows the candidate parameters derived from three sampling parameters: position, view direction, and initial starting time.

Table 2 Candidate input parameters derived from trajectory data

Category	Variables
Density	1) position density, 2) crossing position density, 3) neighbor position density, 4) position density inside view frustum
Time	5) stalling time with interaction, 6) stalling time without interaction, 7) interaction time, 8) view direction duration, 9) trajectory direction duration, 10) overlapping position time
Direction	11) view direction, 12) trajectory direction, 13) directional homogeneity, 14) variance of trajectory direction, 15) variance of view direction, 16) variance of trajectory–view correlation

- **Crossover and mutation:** Crossover is a way of exchanging genes between parents during reproduction. We adopted the uniform crossover method. This exchanges genes independently of each other with a given probability. After crossover, there can be a mutation that flips the gene from ‘0’ to ‘1’, or vice versa. Mutation is necessary to prevent the algorithm from becoming stuck in a local minimum. Our system uses a mutation rate of 0.1%.

- **Selection:** Our goal is to obtain ANN input parameters that can accurately predict the SOI given trajectory data. At the selection stage, the system needs to select the chromosome (i.e., the particular combination of input parameters) that results in the best classification performance by the ANN. For that, we used (1) as the fitness function. Here, x = (number of input variables *before* selection)/(number of input variables *after* selection), y is *training error* (%), and z is the *generation epoch*. In this equation, the lower the number of input variables and the training error, the greater the evaluation value becomes; and the larger the generation number, the greater the differences between chromosomes become.

$$f = \left(1.0 - e^{-(x-1)^z}\right) \times e^{-0.01y^z} \tag{1}$$

With these GA rules, the system can obtain an optimal parameter subset within 10,000 epochs. The size of the chromosome is 10 in every generation, and training data were constructed with 500 samples from the training map. Table 3 shows the ANN input parameters that produced highly accurate behavior classification. We adopted position density, overlapping position time, directional homogeneity, variance of trajectory direction, variance of view direction, and variance of trajectory–view direction correlation. With this process, we could filter out the most discriminating parameters for SOI classification from among all available sampling parameters.

Table 3 Derived parameters from trajectory data

Parameter	Description
Position density	The number of sampled positions within the grid in a given time. It can be used to estimate clustering density.
Overlapping position time	The total time of trajectory overlapping within a grid in a time. It can be used to estimate the temporal coincidence of trajectories.
Directional homogeneity	Cosine similarity of the regression line of each trajectory in a grid. This parameter shows the extent to which the average directions of the trajectories in each region are similar.
Variance of trajectory direction	Variance in the trajectory direction, which means the degree of change in the trajectory direction.
Variance of view direction	Variance in the view direction, which means the rate of change in the player’s target of interest.
Variance of trajectory–view difference	Variance of the view differences. This value shows the discrepancy between player movement and viewing.

4 System

Our system consists of four stages. The first stage is a training stage. The trajectory of expert players is sampled via a plug-in of the game, then the data with behavior marks are sent to our analysis system via network automatically to train the system. The second one is a sampling stage. MMOG clients trace the player's trajectory via a plug-in and send collected data to our analysis system for analysis. The third one is an analysis stage. In this stage, the collected target trajectories are hierarchically divided for optimal classification and are then classified by the ANN. Feature positions are detected. Then the extended DBSCAN algorithm is applied for trajectory clustering. With the above three stages, a trajectory segmented into behavioral SOIs is acquired. Finally, the visualization and data analysis stage takes place.

Figure 4 shows the system process flow. The following sections are a detailed description of every stage in the system. Table 4 shows pseudocode for an MMOG development process with the proposed system.

4.1 Hierarchical spatial division for classification

The proposed system uses a quad tree-based hierarchical spatial division for classification. This method is motivated by the split-and-merge segmentation method in the image processing field. We adopted this method for three reasons. The first reason is for classification efficiency. MMOGs usually include large and complicated worlds, and player visitation rates in the various subzones usually differ greatly. Players tend to play with others, and the amount of highly interesting content is limited; this leads to specific regions becoming either very popular or completely ignored. By adopting a hierarchical spatial division method, we tried to reduce the amount of unnecessary classification and increase classification correctness. The second reason is that such visitation rate discrepancies between subzones produce

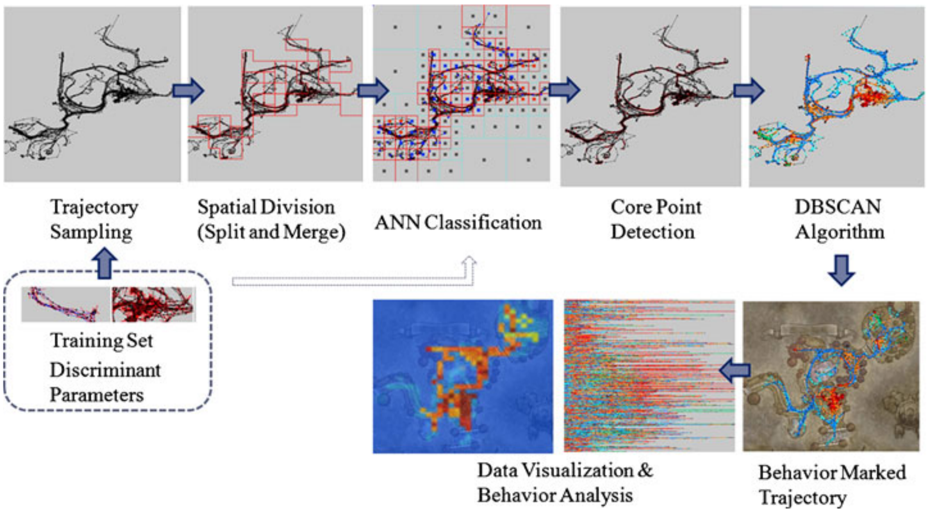


Fig. 4 System process flow

Table 4 Pseudocode for an MMOG development process with the proposed system

```
Algorithm
1. At training step
    $traing\_data_i$ = Select_target_sampling_region()
    For every training data: $traing\_data_i$
        Classifier_training( $traing\_data_i$ )
2. At sampling step
    for every tester: $P_i$
        $R\_list$ = trajectory_sampling($P_i$)
3. On the proposed behavior analysis system
    while (all $R\_list_i$ are homogeneous)
        Split_inside_trajectory ( $R\_list_i$ )
        Merge_adjacent_trajectory ( $R\_list_i$ )
    Behavior_classification ( $R\_list_i$ )
    For every group trajectory: $R\_list_i$
        $feature\_pos$ = Find_featurepos_inside_trajectory
                    ( $R\_list_i$ )
    For every group trajectory: $feature\_pos$
        DBSCAN_clustering ( $feature\_pos$ )
4. At comparison step
    For every group trajectory: $R\_list_i$
        diff = Estimated behavior of $R\_list_i$ −
                behavior of $R\_list_i$
        If ( diff > $threshold\_value$ )
            Fix the related content
        Else
            Confirm behavior of $R\_list_i$
```

level-of-detail problems for classification. For example, in the city, there are various interactive objects and NPCs. The density of content is very high, with a multitude of properties. If the system applies a static region size for classification, various behaviors in dense regions will be overlooked. To analyze player behavior correctly, the grid size should be adaptively changed according to behavior density. The third reason is a practical implementation issue. Most MMOGs use a hierarchical regional data structure for server-side load balancing and client-side visibility culling. For easy integration with existing MMOG data structures in the data sampling stage, we choose a hierarchical spatial method.

4.1.1 Region splitting

Each quad can be divided into four smaller subquads recursively until the splitting condition is no longer satisfied. Our proposed system uses three splitting conditions: 1) density uniformity D_d, 2) temporal coincidence T_d, and 3) directional homogeneity R_d. We use these conditions because our trajectory parameters are based on three features—density, time, and direction—and we use the homogeneity of these features for classification. Equation 2 shows our splitting conditions.

$$f = D_d \cup T_d \cup R_d \tag{2}$$

The density uniformity condition D_d guarantees the uniform distribution of trajectories in a quad. Our system adopts region-based classification, and the training set acquired by users of our analysis system is of grid type. Therefore, for correct classification, every trajectory needs to be distributed uniformly in a quad. To achieve this, we use the *center of mass* value in a quad. If the distance between the centroid of the parent quad M_d and the mean centroid of the child quads C_m is greater than the threshold value θ, then the grid is split. D_d is normalized by the quad size S_d.

$$D_d = \mid \frac{M_d - C_m}{S_d} \mid > \theta \tag{3}$$

Temporal coincidence T_d is a measure of the extent to which individual trajectories overlap in time. If the number of positions at the same time per unit grid size is smaller than a threshold value, the system can assume that the quad is not homogeneous in time.

The mean of the cosine similarity of the regression line of each trajectory U_i is used for calculating the directional homogeneity R_d. U_m is the median degree of regression line in the grid. If R_d is greater than a threshold value, the quad is not directionally homogenous and needs to be split.

$$R_d = \sum_{i=0}^{N} \mid U_i \mid\mid U_t \mid cos(\theta) > \theta \tag{4}$$

The purpose of spatial division is to acquire a homogeneous grid for optimizing classification accuracy. We proceed with spatial division if one of the splitting conditions is satisfied. If the density is zero or the region is homogeneous, splitting is stopped. If the region is not homogeneous, the system splits it into four child regions. In previous trajectory clustering methods, a hierarchical spatial division method is applied to optimize clustering. By contrast, our method is used for maximizing the classification correctness.

4.1.2 Region merging

After the splitting process, the trajectory is segmented by small grids. Because of limitations in the region-based classification, segments in a trajectory are divided by the static shapes of grid cells. To eliminate this problem, a merge process is applied. The homogeneity of each group with the adjacent groups is checked after the splitting has been completed. If the homogeneity of two groups meets a certain level, the two are considered to be the same group and are merged. We have observed some tendencies from two simple region-merging conditions:

- **Size priority:** Most small regions are likely to be merged into larger ones.
- **Variance priority:** Regions of low variance are often merged into regions of higher variance.

The two conditions often happen at the same time. Let θ be a threshold value to prevent dissimilar regions from being merged, and let S_i, μ_i, and σ_i^2 denote the area, statistical mean, and variance of R_i, respectively. Then, a region R_j can be a valid merge candidate if either one of the following conditions is satisfied.

$$\mu_i > \mu_j, S_i > S_j, and \mid \mu_i - \mu_j \mid < \theta \tag{5}$$

In region merging, the merge order is an important factor to consider, because merged results can be quite different depending on the merge order, even under the same merge criterion. In this paper, we regard the priority of a region as its variance value, and regions with lower priority will be merged first. This ordering can be implemented by a loop operation that begins from 1 and proceeds to a sufficiently large number. Then, only regions having a lower standard deviation than the argument in the loop will be considered for merging.

4.2 Behavior classification with supervised trajectory data

Our proposed system uses supervised classification for player behavior analysis. For the training set, 500 training records from 10 expert players were used. The size ranged from 32×32, 64×64, 128×128, and 256×256. The supervised data for combat behavior trajectories were sampled during the fights with monsters in the normal hunting field. The social behavior trajectories were acquired during chat with other players or purchasing of items in the town centers. The movement behavior trajectories were acquired during relatively long journeys between the towns and the hunting fields. To prevent over-fitting in the classification, the sample data for learning were extracted from different training maps. Figure 5 shows our training data sets for supervised learning.

We chose an ANN as a classifier. Because it is a nonlinear model, it is flexible in modeling complex behavioral relationships, compared with conventional classifiers. Also, ANNs are data-driven self-adaptive methods in that they can adjust themselves to the data without any explicit specification of a functional or distributional form for target behavior. This makes the system easily adaptable to various types of target behavior. Our ANN was constructed based on multilayer perceptrons (MLPs). The estimation algorithm stopped when it reached the maximum number of epochs. The input values of the ANN are the six parameters described in Table 3, which are

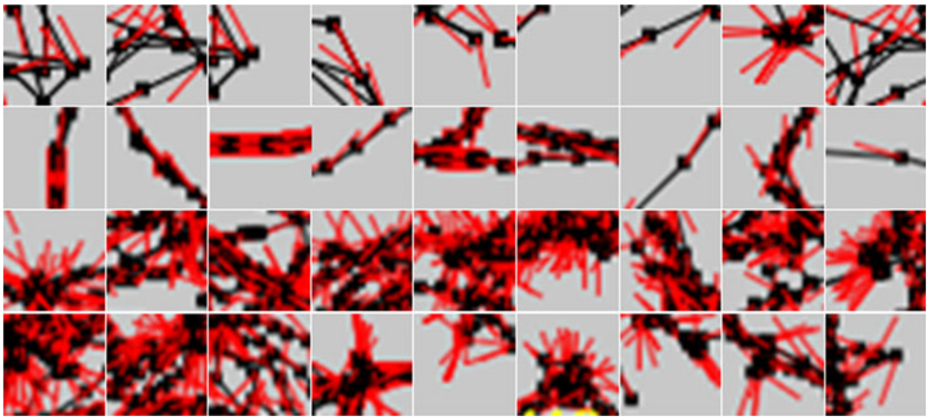


Fig. 5 Training data for classification. From the *top* to the *bottom* row: *1* idling behavior, *2* movement behavior, *3* social behavior, and *4* combat behavior training sets

normalized. Three hidden layers are used. The hyperbolic tangent is used for the hidden layer activation function, and the output layer activation function is softmax. Output values from the ANN are the four target behaviors assigned by the user of the system. Back-propagation is used for training the ANN, and a generalized delta rule is used for updating the weight value in the network. It took 32 seconds to run the training with 500 training records. The classification accuracy of the ANN using threefold cross-validation was about 89.5%.

4.3 Local clustering: DBSCAN algorithm

After classification, the system can have homogeneous behavior values in every grid cell. We could assume that every grid cell has behavior features inside it, but it is difficult to define the cell's area precisely. Hierarchical classification results show a grid type classification boundary along with static segmentation of the trajectories. To solve this problem, we additionally use a density-based trajectory clustering algorithm: DBSCAN. DBSCAN's definition of a cluster is based on the notion of density reachability. A point q is directly density-reachable from a point p if it is not farther away than a given distance ϵ, and if p is surrounded by sufficiently many points such that one may consider p and q to be part of the same cluster. Figure 6 shows a connectivity example of DBSCAN.

Work by Lee et al. [19] shows that the DBSCAN algorithm can be used effectively for line clustering. In this paper, we extended the DBSCAN algorithm to handle clustering of behavior SOIs. We modified the definition of points originally specified for the DBSCAN algorithm. Because DBSCAN is based on density values, other features of trajectories can be eliminated during clustering. We define four more core point conditions and use parameter similarity distance instead of the Euclidian distance ϵ. Following are the definitions of additional core point conditions.

- **End point condition:** An end point starts or ends a trajectory. Play starts or ends at this point. End of a play may include the death of the character.
- **Directional feature point condition:** This is where the direction of a trajectory significantly changes. This occurs when there is a cause in terms of behavior that brings abrupt change to the target.

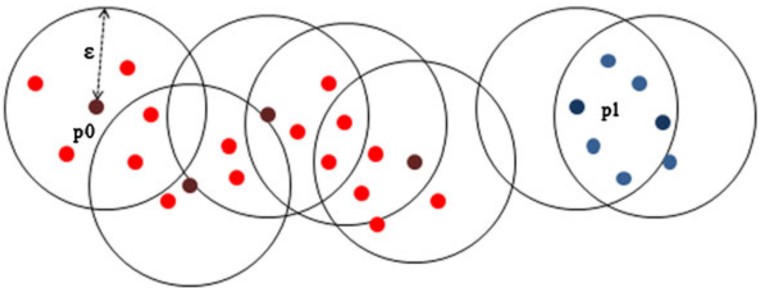


Fig. 6 Density reachability and density connectivity in DBSCAN

- **Cross point condition:** This is where two trajectories cross each other. This occurs when there is a cause in terms of meeting or some active behavior in the region.

The distance between line segments is the main tool for trajectory clustering. The original DBSCAN algorithm uses Euclidian distance for similarity. The purpose of clustering is local expansion from a discriminant feature point. The expansion standard should consider the parameter features. Our trajectory parameters belong to three factors: position, direction, and time. Connectivity should be related to these factors for preservation of the feature for classification. Our distance function in this system is composed of three components: 1) density distance (d_p) (Euclidian distance), 2) temporal distance $d_t = \mid T_i - T_j \mid$, where T_i is the time at the sampled position i, and 3) the angle distance $d_\theta = \mid L_i \mid\mid L_j \mid \times \cos(\theta)$, where L_i is the unit vector at position i. We finally define the distance between two line segments as follows: $dist(L_i, L_j) = w_\perp\ d_p(L_i, L_j) + w_\parallel\ d_t(L_i, L_j) + w_\theta\ d_\theta(L_i, L_j)$. The weights $w_\perp$, $w_\parallel$, and w_θ are determined depending on the application. Our proposed system gives high weight to d_p for social and combat behavior, to d_θ for movement, and to d_t for idling behavior, by considering its discriminating factor. If the distance is under the threshold value, two points are clustered. The results of DBSCAN depend on the parameter settings. The classification value for the determination of the core point of DBSCAN was set at 4, and the clustering distance ϵ was determined in proportion to the area in the grid to which each feature point belonged. Accordingly, a larger merged grid with the same attribute had a larger coverage than the adjacent small grids. Based on these behavioral feature points, clustering based on DBSCAN can generate detailed behavioral segments.

We tried to keep the classification errors at least below 15% during the parameter tuning process by fixing the classifier architecture and parameters. It took around 6 hours for us to tune these parameters in this research. Because four of the selected features are related to trajectory direction, threshold values for directional homogeneity give the highest impact on classification results. Higher directional threshold values are dominant in the movement behaviors. Threshold values for density and time are relatively stable. DBSCAN clustering distances are the second most influential parameters in our system. Those values a determine the clustering areas in core SOI points, inappropriate values make short or overlapped segments of every behavior in a trajectory.

5 Experimental results

The proposed system was implemented using C/C++ and DirectX 9.0 API. Experiments were performed on a computer equipped with an Intel 1.83 GHz processor, 2 GB of memory, and an NVIDIA 8400 GT graphics card. To verify the usability of our proposed system, the experiments for behavior analysis were conducted on two maps from the commercial MMOG WOW.

The first map is Ogrima, the most crowded city in the WOW world. 10,240 positions were sampled from 50 players for 3 days. Ogrima consists of five subzones.

In the south zone, various functional buildings including banks, housing, and auction places exist. This zone is the most popular area for various social behaviors. In the northeast zone, there are many NPCs, which are the gateways to ‘instance fields’. Usually, many players meet at this area to go into the instance fields. There is an underground area in the central zone. In this area, there is an instance dungeon for newbie players, which provides abundant data for a long period. Ogrima has a characteristic movement environment. Most roads are located in long, narrow tunnels. Therefore, long periods of movement are required for navigation through the city.

Figure 7 shows the manual monitoring result. Manual monitoring results were acquired from five monitoring players who had reached maximum level of WOW. They followed the test players using an automatic script and marked the behaviors manually. Each colored circle indicates the dominant behavior of the test players. Social behavior is marked in red; movement behavior, in blue; idling behavior, in green; and combat behavior, in purple. Figure 8 shows the analyzed results with our proposed system. It took 12 seconds to run the analysis. The system correctly detects the dense social area around the south zone. Most trajectories are located in banks and auction places, and they spread out like the spokes of a wheel. Idling spots are detected in the northeast zone even with low trajectory density. It shows that there are periodic meetings in these zones. Many end core points are found because of players entering into instance dungeons. Relatively few specific behaviors are found in the center and northwest area. Because most players studied had already reached a high level, there was little need for them to visit these places, which consist of low-level content. Because of the map’s geometric features, long, narrow trajectories are found. The system detects these trajectories as movement trajectories. Especially many core points for movement SOIs are found near specific zones where gateways are located.

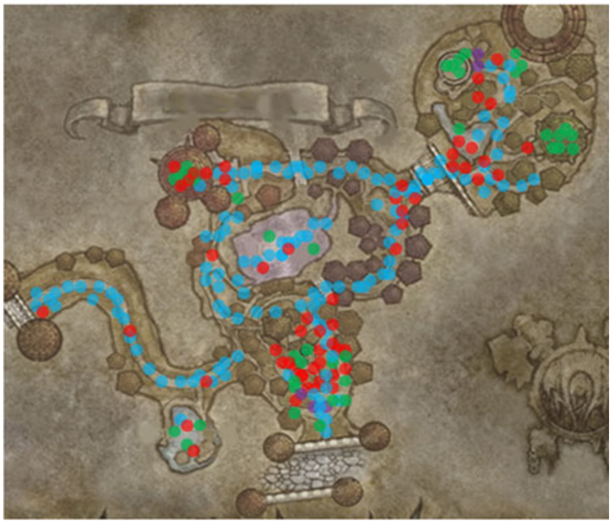


Fig. 7 Behavior monitoring result (manual monitoring) for the Ogrima map (*red*: social, *blue*: movement, *purple*: combat, *green*: idling behavior)

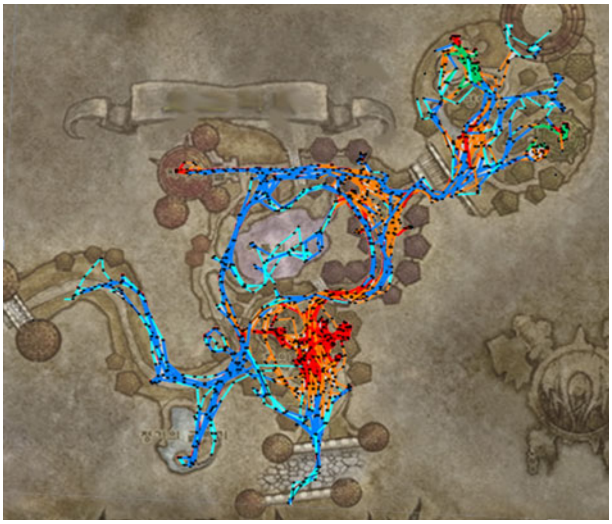


Fig. 8 Behavior analysis result (our system) for the Ogrima map

The second map is Nagrand, which is one of the popular hunting fields in the WOW world. 22,240 positions were sampled from 50 players for 3 days. Nagrand has a broad range of quests at many levels, suitable for high-level players; therefore, it is frequently visited by max-level players. Usually, Nagrand requires a continuous-combat play pattern compared with other levels. Because most of the hunting fields are close to main roads, much combat here happens around the

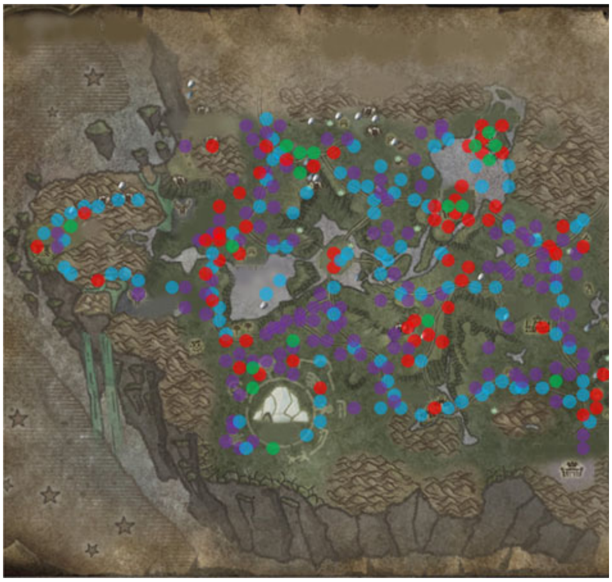


Fig. 9 Behavior monitoring results (from manual monitoring) for the Nagrand map

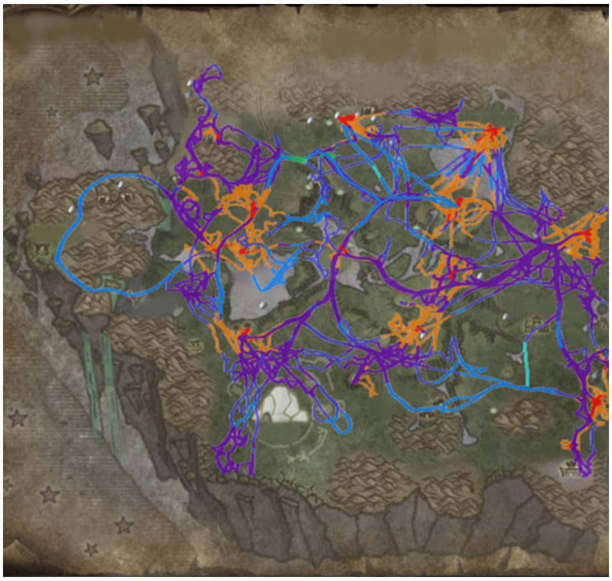


Fig. 10 Behavior analysis results (from our system) for the Nagrand map

roads. The main starting point is located at the northeast town. Figure 9 shows the behavior monitoring results acquired by manual monitoring work. Figure 10 shows the experimental results. It took 21 seconds to run the analysis. Our proposed system correctly detects nine distributed social areas in the map. These groups are located around towns. It means that the town is used as a social hub, as the game design intended. Few meetings were identified around the massive combat content region. The system also detected long, narrow trajectories of movement behavior along the roads. Because this behavior is mixed with combat behavior

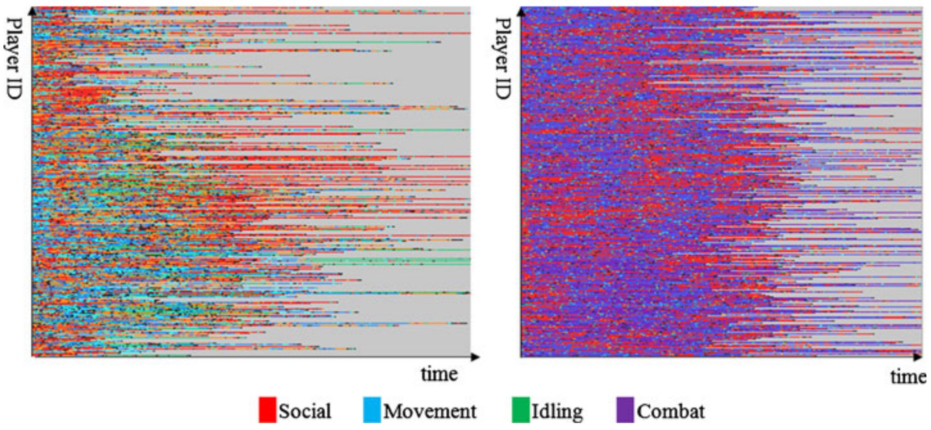


Fig. 11 SBB comparison between the Ogrima and Nagrand maps

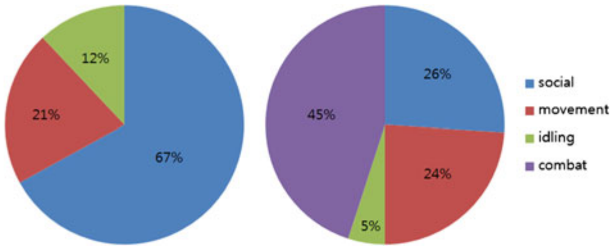


Fig. 12 Behavior ratio pie chart for the Ogrima and Nagrand maps

trajectories, there are several overlapping trajectories. Our system gives higher priority to combat behaviors; therefore, these areas were grouped as combat behavior areas.

Based on these results, additional useful information can be acquired through the knowledge discovery process. Because the trajectory data are sequential data with static time intervals, if the data lie in a single dimension, player experience can be easily serialized. We visualize this 1D sequential data as a *sequential behavior bar (SBB)*. These data are useful for analyzing the players' personal experiences, including playing time, behavior sequence, behavior duration, and behavior frequency. Figure 11 shows the SBB of two maps. In the SBB of Ogrima, many short social behavior patterns exist. This means that many players visited Ogrima only for social behavior, then left the map within a short time. These trajectories are concentrated in the south zones. The average playing time in Ogrima was 15 minutes, and the standard deviation was 435.34. In the SBB of Nagrand, many long-duration combat behavior patterns were found. This means that many players visited Nagrand for combat behavior and remained on the map for a long time. These trajectories were distributed over all the zones. The average playing time in Nagrand was 30.45 minutes, and the standard deviation was 219.21.

By counting the frequencies of every action per player, we can produce a *behavior ratio pie-chart (BRP)* for the target map. With this data, a user can find out the player behavior ratio during a specific time on a map. This gives key statistics for classifying personalized play patterns in MMOGs. It is especially useful for map usage analysis. Figure 12 shows a BRP result. From the BRP of Ogrima, we see that Ogrima is used mainly for social activities with 67% in social SOIs. Movement comprises 21% of the entire content, implying that much playing time is consumed in moving around Ogrima. Compared with other maps, this value is relatively high. From the BRP of Nagrand, we see that Nagrand is used mainly for combat with 45% in combat behavior. Social behavior comprises 26% of the total behavior, implying that much playing time is consumed during combat in Nagrand.

6 Discussion

Our result can be affected by test player gaming styles because we trained the classifier based on the players' sampling data. However, our results are relatively stable against the various game play patterns in the topics discussed in this study. Basically, our training data consists of an aggregation of the test players' trajectories. In this regard, in aggregating and classifying the four major target behaviors (social, combat, movement, and idling), the deviations caused by personal gaming styles are averaged out. As a result, the trajectory patterns show several coincidences. If we tried to classify more detailed behaviors such as combat styles, such differences in gaming styles might give higher impact on the results. To avoid such deviations and provide more reliable results, we focus on the four proposed behaviors.

The accuracy in classification would increase if the system could log actual data from a game server. For instance, if a game server recorded specific interactions (e.g. usage information on specific combat skill), we could analyze target behaviors more correctly. However, such real-time operations are practically difficult because tracking and storing such information requires huge computational resources on a server that should handle thousands of concurrent players without any defects in the modern commercial MMOG architecture.

Our method is based on client-side data sampling which gives no workload to a game server. Among the usable parameters in the client of MMOG, trajectory data is acquired by basic sampling parameters to check basic player movements in the map. We show that possibility of usage of trajectory can be extended to analysis of behavior and knowledge discovery. We think that our method can be used as a complementary system for behavior analysis with low sampling costs. In addition, its performance and detail can be increased with the sampling of system interaction parameters. Furthermore, this approach demonstrates an alternative research technique for MMOG researchers who cannot access internal server data of the game company for security reasons.

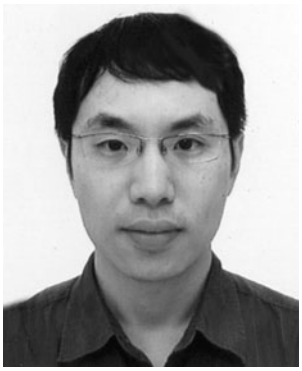
7 Conclusion

For a long time, player behavior analysis depended on human experimental feedback. This work is expensive in terms of interviewing time, and it can be easily affected by subjective judgment. If accurate feedback is enabled by an automated system, the development process can be further optimized. In this paper, we proposed an automated behavior analysis system using trajectory data. The proposed system combines hierarchical classification and a trajectory clustering method to identify the behavioral meaning of trajectory segments. Our system demonstrated the capability to link trajectory data to player behavior automatically. Additionally, it can be used to create key statistics for further knowledge discovery. Our system has been consistently updated for applying to next generation commercial MMOG from NCsoft. The results acquired from our system are useful for MMOG designers; the behavioral analysis data can help reduce costs greatly by eliminating manual works required in reworking level content and in detecting unexpected and unplanned actions (i.e. outliers). Our approach has an advantage in that sophisticated behavior analysis can be implemented with only simple trajectory data even without any expertise.

Acknowledgement This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2011-0017595).

References

1. Abhinav S, Ashraf S (2006) Genetic algorithms for artificial neural net-based condition monitoring system design for rotating mechanical systems. Adv Soft Comput 34:135–149
2. Agrawal R, Gehrke J, Gunopulos D, Raghavan P (1998) Automatic subspace clustering of high dimensional data for data mining applications. In: Proceedings of SIGMOD, pp 94–105
3. Ankerst M, Breunig MM, Kriegel H, Sander J (1999) OPTICS: ordering points to identify the clustering structure. In: Proceedings of SIGMOD, pp 49–60
4. Bartle R (1996) Hearts, clubs, diamonds, spades: players who suit MUDs. Accessed at http://www.mud.co.uk/richard/hcds.htm. Jan 1 2012
5. Borner K, Penumarthy S (2003) Social diffusion patterns in three-dimensional virtual worlds. Information Visualization 2(3):182–198
6. Charles D, Black M (2004) Dynamic player modelling: a framework for player-centric digital games. In: Proceedings of the international conference on computer games: artificial intelligence, design and education, pp 29–35
7. Chittaro L, Ranon R, Leronutti L (2006) VU-flow: a visualization tool for analyzing navigation in virtual environments. IEEE Trans Vis Comput Graph 12(6):1475–1485
8. Cowley B, Charles D, Black M, Hickey R (2008) Toward an understanding of flow in video games. Comput Entertain 6:1–27
9. DeRosa P (2007) Tracking player feedback to improve game design. Gamasutra
10. Ducheneaut N, Yee N, Nickell E, Moore RJ (2006) Building an MMO with mass appeal: a look at gameplay in world of warcraft. Game Developer Magazine
11. Ester M, Kriegel H-P, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases. In: Proceedings of the knowledge discovery and data mining conference, pp 226–231
12. Franklin SE, Wilson BA (1991) Spatial and spectral classification of remote-sensing imagery. Comput Geosci 17(8):1151–1172
13. Guha S (2000) ROCK: a robust clustering algorithm for categorical attributes. In: Proceedings of ICDE, pp 345–352
14. Guha S (1998) CURE: an efficient clustering algorithm for large databases. In: Proceedings of SIGMOD, pp 1–10
15. Hinneburg A, Gabriel H (2007) DENCLUE 2.0: fast clustering based on kernel density estimation. LNCS, vol 4723, pp 70–80
16. Hoobler N, Humphreys G, Agrawala M (2004) Visualizing competitive behaviors in multi-user virtual environments. In: Proceedings of the 15th IEEE visualization conference, pp 163–170
17. Houlette R (2004) Player modeling for adaptive games. In: AI game programming wisdom II. Charles River Media, pp 557–566
18. Kim JH, Gunn DV, Schuh E, Phillips B, Pagulayan RJ, Wixon D (2008) Tracking real-time user experience (TRUE): a comprehensive instrumentation solution for complex systems. In: Proceeding of SIGCHI, pp 443–452
19. Lee JG, Han J (2007) Trajectory clustering: a partition and group framework. In: Proceedings of SIGMOD, pp 593–604
20. Matsumoto Y, Thawonmas R (2004) MMOG player classification using hidden Markov models. LNCS, vol 3166, pp 429–434
21. Nagesh H, Goil S, Choudhary A (1999) MAFIA: eficient and scalable subspace clustering for very large data sets. Technical report 9906-010, Northwestern University
22. Ng RT, Han J (2002) CLARANS: a method for clustering objects for spatial data mining. IEEE Trans Knowl Data Eng 14(5):1003–1016
23. Sheikholeslami G, Chatterjee S, Zhang A (1998) WaveCluster: a multiresolution clustering approach for very large spatial databases. In: Proceedings of very large data bases, pp 428–439
24. Thawonmas R, Hirano M, Kurashige M (2006) Cellular automata and hilditch thinning for extraction of user paths in online games. In: Proceedings of the 5th ACM SIGCOMM workshop on network and system support for games
25. Thawonmas R, Kurashige M, Chen KT (2007) Detection of landmarks for clustering of online-game players. Int J Virtual Real 6(3):11–16
26. Zhang T, Ramakhrisnan R, Livny M (1996) BIRCH: an efficient data clustering method for very large databases. In: Proceedings of SIGMOD, pp 103–114



Shin-Jin Kang received an MS degree from the Department of Computer Science and Engineering from Korea University in 2003. After graduation, he joined Sony Computer Entertainment Korea (SCEK) as a video game programmer. From 2006, He has worked at NCsoft Korea as a lead game designer in AAA MMORPG projects including AION and Lineage series. He received PhD degree in Computer Science & Engineering at Korea University in 2011. And he is now a professor at the School of Games in Hongik University and also a senior manager in research and development division of NCsoft Korea.

Young Bin Kim received a B.A. degree in the Department of Computer Science and Engineering at Korea University in 2010. He is a M.S. course student in the Department of Computer Science and Engineering at Korea University. His current research interests include image processing and video analysis.

Taejung Park has worked as a research professor in the BK21 program at Korea University since 2006. He received B.A. and M.S. degrees in electrical engineering from Seoul National University in 1997 and 1999, respectively. He designed a data structure for a direct matrix solver for FEM simulation for semiconductor devices. After working with two small start-up technology businesses in Korea in 1999 and 2002, he received a PhD from Seoul National University for 3D mesh compression in 2006. His current research interests include parallel numerical simulation techniques and nonlinear interpolation methods from the viewpoint of information technology.

Chang-Hun Kim is a Professor in the Department of Computer Science and Engineering at Korea University. He received his B.A. degree in Economics from Korea University in 1979. After graduation, he joined the Korea Advanced Institute of Science and Technology (KAIST) as a Research Scientist, where he was involved in many national research projects in the area of Computer Aided Design and Geometric Modeling. He received his PhD from the Department of Electronics and Information Science, Tsukuba University, Japan in 1993. During 1993–1995, he headed the Human Interface and Graphics Laboratory for System Engineering Research Institute (SERI). His current research interests include fluid animation and mesh processing. He is also a member of IEEE Computer Society and ACM.