Beyond Heatmaps: Spatio-Temporal Clustering using Behavior-Based Partitioning of Game Levels

Christian Baukhave$\dagger$, Rafet Sifa$\ddagger$, Anders Drachen$\ddagger$, Christian Thurau§ and Fabian Hadiji§

$^*$B-IT, University of Bonn, 53113 Bonn Germany
$^\dagger$Fraunhofer IAIS, 53745 St. Augustin, Germany
$^\ddagger$Aalborg University, Aalborg, Denmark
$^\§$GameAnalytics, Berlin, Germany

Abstract—Evaluating the spatial behavior of players allows for comparing design intent with emergent behavior. However, spatial analytics for game development is still in its infancy and current analysis mostly relies on aggregate visualizations such as heatmaps. In this paper, we propose the use of advanced spatial clustering techniques to evaluate player behavior. In particular, we consider the use of DEDICOM and DESICOM, two techniques that operate on asymmetric spatial similarity matrices and can simultaneously uncover preferred locations and likely transitions between them. Our results highlight the ability of asymmetric techniques to partition game maps into meaningful areas and to retain information about player movements between these areas.

I. INTRODUCTION

Many modern games such as first- or third-person games in which players guide avatars through 3D worlds require spatial movements. In the Quake-, Battlefield- or Unreal series, in real-time strategy games such as StarCraft, or in role playing games such as World of Warcraft, spatial navigation is key and spatial behaviors are integral to the gaming experience.

The challenge of understanding spatio-temporal behaviors is thus a major driver behind behavioral telemetry in games and methods to evaluate design intent against emergent player behavior have become vital to game development [1]–[3]. Behavioral analysis enables designers to study dimensions of gameplay as experienced by the player but any analysis that ignores the spatial aspects of play risks misleading results [4]. Historically, simple visualizations of spatial player behavior have been the tool of the trade, often in the form of heatmaps (see Fig. 1). However, heatmaps are limited in that they ignore directional and temporal information. It is thus surprising that research on more advanced spatio-temporal analytics in games has primarily been focused on guiding artificial agents (bots) in FPS or RTS games [5]–[9] rather than on assisting design.

The work presented here aims at informing level design and game layout. We provide new methods for the analysis of spatio-temporal player behavior and aim at a behavior-based partitioning of game levels (a.k.a. maps or playfields) for the specific purpose of improving tools available for analyzing the design of maps. Although the idea of analyzing the spatio-temporal behavior of actual players attracts growing attention [1], [10]–[13] and all major publishers and developers currently use some form of spatial analytics [3], publicly available information on spatial and spatio-temporal gameplay is surprisingly scarce [14]. This is due to the novelty of the idea but also related to the fact that telemetry data from commercial games is proprietary and of business value. Practical examples in this paper therefore pertain to well known FPS games whose source code or API definitions are publicly available.

In practice, the evaluation and tuning of the design of maps, from simple open area maps to complex multi-layered maps with triggers, mission conditions etc., operate at a highly granular level [3], [4]. For example, a 3D heatmap visualizing event frequencies on a game map can reveal that a specific area sees too high or too low a concentration of the feature being investigated (death events, kill events, purchasing events, or preferred player locations). However, the heatmap itself cannot reveal why a specific concentration occurs. In order to resolve this problem, a more detailed investigation must be carried out and, in this context, detailed trajectory data were reported to be highly useful [2], [4], [12], [14].

To address the need for a meaningful way of dissecting maps based on player behavior, an alternative to heatmaps is to apply unsupervised machine learning techniques and spatial clustering to trajectory data in order to identify players movement preferences [14]. Spatial clustering attempts to group spatial objects that are similar with respect to their general characteristics and/or are co-located in 3D space. Using spatial clustering techniques to detect areas where players congregate and how they movement between these, provides a means for identifying areas of a map which may require closer analysis in order to evaluate design. This is notably useful in situations where the map geometry is complex and thus requires partitioning, or where large scale behavioral data sets are available. Indeed, in these situations, dividing maps into sections is not only useful but a necessity [3], [4], [12]. However, any such division should be based on behavioral patterns rather than on subjective decisions and the chosen approach should retain access to the larger patterns of spatial behavior on the map.

II. CONTRIBUTION

In this paper, we address the problem of spatial clustering in 3D games with complex map geometries and multiple Z-level planes. We compare four different techniques for clustering player trajectories and aim at partitioning game level maps into areas of player interest (see Fig. 2). We discuss their abilities to identify structures that are geometrically distinct and form meaningful parts of the map.

Two of these techniques (DEDICOM and DESICOM) have never before been applied in games research. We will show that...
they allow for localizing hotspots (clusters) of player activity and, at the same time, uncover relations between hotspots. Both techniques use affinity matrices to indicate interactions between spatial clusters. They also allow for finding clusters in areas where player activity has a low density such as in transitional regions between hotspots. To the best of our knowledge, spatial clustering techniques that retain information about the movement between clusters have not been presented in game contexts before.

III. DATA ACQUISITION AND PRE-PROCESSING

All practical examples reported in this paper pertain to Quake III and Unreal Tournament 2003. Both games feature many intricate maps and provide mechanisms to record a player’s in-game activities. Data was therefore collected from players of different skill levels.

We then parse the resulting demo files for player positions and obtain trajectories $X = \{x_1, x_2, \ldots, x_T\}$ of consecutive points where each $x_t \in \mathbb{R}^3$ represents a player’s location at time $t$. Figure 2b illustrates that player trajectories determined this way consist of $T \in O(10,000)$ densely spaced points and form three-dimensional structures indicative of architectural features of the underlying game map. Data like these allow for generating various types of heatmaps, however, mere spatial statistics ignore temporal aspects and cannot characterize transitions between parts of a map. Towards more advanced analysis, we therefore apply $k$-means clustering to a trajectory and obtain sets of prototypical waypoints $W = \{w_1, w_2, \ldots, w_n\}$ where $w_i \in \mathbb{R}^3$ and $n \in O(100)$.

Having extracted waypoints from a player trajectory, we construct a waypoint transition graph $G = (V, E)$ where the vertex set $V = W$ and edges $E \subseteq V \times V$ are drawn as follows: we assign each position $x_t$ to its closest waypoint $w_i = w_t(x_t)$ and then determine every move $x_t \rightarrow x_{t+1}$ where $w_t(x_t) \neq w_t(x_{t+1})$. Since a waypoint forms the barycenter of a cell within a Voronoi tessellation of the trajectory, i.e. a navigation mesh, such moves constitute transitions between meshes. Accordingly, if $x_t$ and $x_{t+1}$ belong to two different meshes represented by $w_i$ and $w_j$, the pair $(w_i, w_j)$ is added to the set of edges of $G$.

Waypoint transition graphs provide an informative data structure for game analytics. They characterize how players perceive the layout of a game world and how they navigate a map. First of all, they generally are directed graphs. This is because the game mechanics may prevent certain transitions between waypoints. For instance, a player may jump down a ledge and thus transit from one waypoint to another but the game physics may render it impossible to get back up the ledge immediately so that the relation between the two waypoints is asymmetric. Second of all, edges in a waypoint transition...
Clustering algorithms feature prominently in game analytics and there are numerous variants [15]. Here, we review baseline methods that have been applied to player trajectories before [6], [9], [16] and then focus on more elaborate techniques that do not apply to every kind of data [17], [18]. While $k$-means clustering is guided by local properties of data, spectral clustering incorporates a more global view. Given a sample $W = \{w_1, \ldots, w_n\}$, it relies on pairwise similarities between objects in the set. Similarities are gathered in a matrix $S \in \mathbb{R}^{n \times n}$ where large entries $S_{ij} = s(w_i, w_j)$ indicate high affinities.

Spectral clustering is closely related to the problem of graph partitioning, because $S$ can be seen as a weighted graph adjacency matrix. In a seminal paper [19], Fiedler showed that spectral decompositions as in (5) allow for graph partitioning and proved that clusters can be determined from looking at the eigenvectors belonging to the $k$ smallest eigenvalues of the graph Laplacian $L = D - S$ where $D$ is a diagonal matrix with $D_{ii} = \sum_j S_{ij}$.

There are many variants of spectral clustering [20]–[23]; in this paper, we consider an efficient baseline technique. Given the normalized Laplacian

$$L = I - D^{-\frac{1}{2}} S D^{-\frac{1}{2}}$$

of $S$, we compute its decomposition, determine the $k$ smallest eigenvalues, collect the corresponding eigenvectors in a matrix $U \in \mathbb{R}^{n \times k}$, apply $k$-means to the $n$ rows of $U$, and thus group the waypoints in $W = \{w_1, \ldots, w_n\}$. For reference, we show a python implementation of this approach in algorithm 1.

Looking at Fig. 2d, we observe that spectral clustering partitions the waypoint graph into components that reflect the architecture of the game map. Based on spatial similarity data extracted from trajectories, it identifies the upper and the lower platform as well as the bridge leading to the latter.

Although this corroborates earlier findings [16], spectral clustering has two apparent shortcomings w.r.t. our problem. First of all, as it relies on the spectral theorem, it requires symmetric matrices $S = S^T$ to work properly. Yet, game
Algo\texttt{rithm 1} python code for spectral clustering

```python
from numpy import *
from numpy.linalg import *
from scipy.cluster.vq import *

# compute normalized Laplacian $L$ of $S$
D = inv(sqrt(diag(sum(S, axis=1))))
L = eye(n) - dot(D, dot(S, D))

# compute eigenvectors / eigenvalues of $L$
evals = argsort(evals)
U = evcts[:,sortedevals[:k]]

# cluster rows of $U$ using k-means
clusters, labels = kmeans2(U, k)
```

physics and player behaviors may cause asymmetric spatio-temporal relations. Consider a \textit{Quake III} player stepping into a teleporter. If $w_i$ and $w_j$ are the closest waypoints before and after teleporting, then $s(w_i, w_j) \neq s(w_j, w_i)$. Therefore, waypoint graphs determined from player trajectories are usually directed and for similarity matrices derived therefrom we have $S \neq S^T$.

Second of all, spectral clustering of similarity matrices does not provide a characterization of affinities among clusters. Given a similarity matrix $S \in \mathbb{R}^{n \times n}$, spectral clustering groups a set of $n$ objects into $k$ clusters but does not produce a matrix $R \in \mathbb{R}^{k \times k}$ which relates these clusters. Next, we discuss two approaches which address both these issues.

C. DEDICOM

The idea of decomposition into directional components (DEDICOM) was introduced by Harshamn [24] for community detection from asymmetric social ties. It recently resurfaced in online social network analysis [25], [26] but to our best knowledge-- has not yet been considered in game analytics. We therefore review the underlying concepts and then introduce a novel and efficient algorithm for DEDICOM.

Given a matrix $S \in \mathbb{R}^{n \times n}$ of asymmetric relations among $n$ objects, DEDICOM considers the matrix factorization problem

$$S \approx A R A^T$$  \hspace{1cm} (7)

where $A \in \mathbb{R}^{n \times k}$ and $R \in \mathbb{R}^{k \times k}$. This resembles the spectral decomposition in (5), but matrix $A$ is of rank $k \ll n$ and not necessarily orthogonal and $R$ is dense rather than diagonal. Once determined, the columns of $A$ can be understood as $k$ “basis" vectors or latent factors behind the relationships encoded in $S$ and matrix $R$ will indicate relations among these latent components or clusters.

We must emphasize that DEDICOM’s notion of “directional components" does not allude to spatial reasoning but refers to transitive aspects of preference relations. For instance, if $A$ likes $B$ and $B$ likes $C$, then $A$ may also like $C$ and there is an abstract direction of affinity from $A$ to $C$. However, if $A$ hardly likes $B$ but $B$ likes $C$, then $A$ may or may not be fond of $C$. It are compressed numerical characterizations of affinity transitions like these that are captured in $R$.

Solving DEDICOM can be cast as a problem of minimizing a matrix norm

$$\min_{A,R} \| S - A R A^T \|^2$$ \hspace{1cm} (8)

which is convex in $R$ but neither in $A$ nor in the product $A R$. Known algorithms [27], [28] therefore randomly initialize $A$ and $R$ and then apply iterative schemes similar to the $k$-means procedure discussed above.

Our own approach to DEDICOM is based on an alternating least squares procedure proposed in [25]. It simplifies estimating $A$ by considering the situation after stacking matrix $S$ and matrix $S^T$ side by side. This yields

$$\begin{bmatrix} S & S^T \end{bmatrix} = \begin{bmatrix} A R A^T & A R^T R A^T \end{bmatrix} = A \begin{bmatrix} R A^T & R^T A^T \end{bmatrix}$$ \hspace{1cm} (9)

and allows for solving for $A$ if $A^T$ is held fixed [25]. To see how, we substitute $T = [S \ S^T]$ and $B = [R A^T \ R^T A^T]$ and write (9) as $T = A B$. Resorting to the pseudo inverse $B^+$ of $B$, we find

$$A = T B^+ = T B^+ \left( B B^+ \right)^{-1}$$ \hspace{1cm} (10)

which, after reversing our substitution, provides the following update for matrix $A$

$$A \leftarrow (S A R^T + S^T A R) \left( R A^T R^T + R^T A^T R \right)^{-1}.$$ \hspace{1cm} (11)

Once an update for $A$ is available, the current estimate of $R$ can be improved using

$$R \leftarrow A^T S A^\dagger = \left( A^T A \right)^{-1} A^T S A \left( A^T A \right)^{-1}$$ \hspace{1cm} (12)

and steps (11) and (12) have to be repeated until convergence.

Looking at (11) and (12), we note that matters simplify, if we constrain the solution for $A$. Requiring its columns to be orthogonal unit vectors in $\mathbb{R}^n$, we have $A^T A = I$ and (11) becomes

$$A \leftarrow (S A R^T + S^T A R) \left( R R^T + R^T R \right)^{-1}.$$ \hspace{1cm} (13)

As the updated $A$ may not be orthogonal, we determine $A = Q T$ where $Q$ is orthogonal and $T$ is upper triangular.\footnote{Note that this is the well known QR decomposition but, to avoid confusion with matrix $R$ from DEDICOM, we write $A = Q T$.}

We then set $A \leftarrow Q$ and compute

$$R \leftarrow A^T S A.$$ \hspace{1cm} (14)

Upon convergence of this algorithm, we apply $k$-means clustering to the $n$ rows of $A$, and thus obtain cluster labels for the waypoints in $W = \{w_1, \ldots, w_n\}$. Algorithm 2 shows a python implementation that summarizes our derivation.

Constraining $A$ to orthogonality is beneficial as it fixes a rotational ambiguity [25] and improves runtime (see Fig. 3). A potential drawback is that unconstrained solutions might be more accurate. But since $A$ is an $n \times k$ matrix where $n \gg k$ it contains only few albeit high dimensional columns. Because of the peculiarities of high dimensional data, it is thus likely that unconstrained DEDICOM will result in orthogonal factors.
clusters, labels = kmeans2(A, k)  
# cluster rows of A using k-means  
for t in range(100):  
errold = inf  
A, T = qr(A)  
A = random.rand(n, k)  
R = random.rand(k, k)  
from scipy.cluster.vq import *  

Fig. 2e, the affinity matrix affinities among the resulting clusters. For the example in contrast to spectral clustering, DEDICOM also characterizes produces clusters similar to those from spectral clustering. Yet, waypoint similarities. The example in Fig. 2e shows that this asymmetric matrices whose entries indicate spatio-temporal asymmetric matrices whose entries indicate spatio-temporal  

due to orthogonal DEDICOM revealed no difference (see Fig. 3).  

All DEDICOM results in this paper were obtained from asymmetric matrices whose entries indicate spatio-temporal waypoint similarities. The example in Fig. 2e shows that this produces clusters similar to those from spectral clustering. Yet, in contrast to spectral clustering, DEDICOM also characterizes affinities among the resulting clusters. For the example in Fig. 2e, the affinity matrix \( R \) is given by  

\[
R = \begin{bmatrix}
60.23 & 61.36 & -13.35 \\
41.31 & 43.55 & -8.09 \\
-6.01 & -6.92 & 4.06
\end{bmatrix}
\]

and we recognize one cluster to have a rather low self-affinity and even lower (negative) affinities to the other two. Indeed, this cluster consist of waypoints along the bridge which the recording player only traversed once. Both other clusters are positively related. Observing that the one has a higher affinity to the other than to itself reflects the fact that the recording player had more ways to transit from the upper platform to the lower than vice versa.

Although these results are encouraging, negative affinities may raise concern among practitioners. Next, we discuss an approach that addresses this issue.

**D. DESICOM**

In [31], Kiers presented DESICOM, a decomposition into simple directional components. His approach computes a sparse factor matrix \( A \) such that each of its \( n \) rows contains only one entry different from zero. This is accomplished using a procedure that updates \( A \) one row at a time, then updates \( R \) using (12), and repeats these steps until convergence. Although a mathematical discussion of the is algorithm is beyond our scope, we provide a python implementation in algorithm 3 and refer to [31] for technical details.

```python
from numpy import *
from numpy.linalg import *
from scipy.cluster.vq import *

Algorithm 2 python code for orthogonal DEDICOM
```

```python
from numpy import *
from numpy.linalg import *
from scipy.cluster.vq import *

Algorithm 3 python code for DESICOM
```

DESICOM is slower than any of the algorithm discussed so far (see Fig. 3) but has several intriguing features. First of all, cluster can be read off matrix \( A \) directly. Since for each row index \( j \) there is a single column index \( i \) such that \( A_{ji} > 0 \), we simply assign waypoint \( w_j \) to cluster \( C_i \). Second of all, if DESICOM starts with non-negative initializations of the factor matrices, it will provably find a non-negative decomposition of \( S \). This is desirable since similarity matrices are typically non-negative themselves. Third of all, since \( A \) is non-negative we rescale its columns such that they sum to one which provides us with factors that can be interpreted in terms of probabilities. By the same token, we may scale the rows of \( R \) and thus obtain a compressed similarity matrix whose entries resemble probabilities.

Figure 2f shows that DESICOM clusters similar to the two previous methods. However, (normalized) cluster affinity
matrices resulting from DESICOM are more easy to interpret. Here, we found
\[
R = \begin{bmatrix} 1.00 & 0.00 & 0.00 \\ 0.00 & 0.92 & 0.08 \\ 0.00 & 0.03 & 0.97 \end{bmatrix}
\]
which again reveals that the waypoints in the first cluster (i.e. the bridge) hardly “interact” with the other clusters. Also, we once again observe that it was more likely for the player to move from the upper to the lower platform than vice versa.

V. EXPERIMENTS

In this section, we present experiments with the methods discussed above. Our examples consider the games Quake III and UT 2003 where successful play depends on knowledge of map layouts and which enable research since their source code has been openly released. We collect data for different players and determine behavior-induced partitionings of maps and as well as affinities between the resulting clusters.

A. Behavior-Induced Map Partionings

In a first series of experiments, we applied spectral clustering, DEDICOM, and DESICOM to trajectory data much larger than in the didactic example above. In short, we found that spectral clustering works well for symmetric, spatial similarity data whereas DEDICOM and DESICOM perform most convincingly if applied to asymmetric matrices reflecting spatio-temporal similarities between waypoints.

Figure 4 shows player trajectories on the UT 2003 map Gael and clustering results for \( k = 3 \). All three methods uncover structures that match the map geometry. Yet, DEDICOM and DESICOM recognize the fact that the recording player traversed the lower part of the map in a non-circular fashion and yield clusters that reflect this movement patterns.

Figure 5 displays DESICOM results for different numbers of clusters. We observe that for increasing \( k \), more and more details about constituent parts of the map become apparent. We also note that DESICOM aptly identifies hotspots in player trajectories. Even for small \( k \) some clusters contain much fewer points than others but correspond to parts of the map the player visited more frequently; interestingly, these hotspots persist over a range of settings for \( k \).

Finally, DESICOM provides interpretable cluster affinity values. For example, for Fig. 5b where \( k = 4 \) we obtained the following (normalized) affinity matrix

\[
\begin{array}{cccc}
\text{cyan} & \text{magenta} & \text{yellow} & \text{red} \\
0.44 & 0.04 & 0.46 & 0.06 \\
0.29 & 0.15 & 0.51 & 0.05 \\
0.13 & 0.03 & 0.81 & 0.03 \\
0.39 & 0.09 & 0.23 & 0.29 \\
\end{array}
\]

from which we read, among others, that it is unlikely for the player to transit from the magenta to the red cluster but that the player did indeed more frequently jump from waypoints in the magenta cluster down to waypoints in the cyan cluster. We also recognize the red cluster as a transitional region in which the player was less likely to stay but quickly moved to the cyan and on to the yellow cluster.

B. Comparative Analysis of Play Styles

In a second series of experiments, we applied DESICOM to compare playing styles of different players; Fig. 7 shows an example. To produce these results, we had an experienced player traverse the Quake III map q3dm17 and extracted 250 waypoints from his recordings (see Fig. 6). Using these as a common reference, we computed waypoint transition graphs from data generated by three different players which we then clustered into \( k = 6 \) components.

The heat map in Fig. 7a shows that the first player was a “camper” who spent most time at the remote platform from where he used the railgun to snipe his opponents. DESICOM identifies his preferred location in form of the magenta cluster which only interacts with the transitional red cluster. We also note that, according to the unnormalized matrix \( R \), affinities between the 6 clusters found are generally rather low.

The second player was an experience player who circled the map yet tried to control the match from the upper left platform of the map’s main structure. This preference location was again identified by DESICOM and corresonds to the magenta cluster. In contrast to the “camper” we observe that cluster affinities found for the experience player are higher which indicates his propensity to move between different parts of the map.

The third player was a semi professional who made use of all parts of the map and had no recognizable single preference location. Still, DESICOM correctly identified smaller regions where he was found more frequently (the green and magenta cluster). What is most striking, however, is that on average the affinities between waypoint clusters determined for this player are highest. This indicates that he was constantly moving between different regions of the map and showed no simple pattern of transiting from one part to the other.

VI. CONCLUSION

Knowledge generation from in-game trajectory data is an important problem in the emerging area of game analytics. Analyzing spatio-temporal player behavior can inform the implementation of believable game bots and, more importantly, can help to improve map design during game development. In this paper, we considered advanced clustering algorithms that
can simultaneously uncover a player’s preference locations as well as propensities for transiting between them. In particular, we considered the use of DEDICOM and DESICOM, two matrix factorization methods, that allow for decomposing asymmetric similarity matrices into latent factors.

Using such elaborate techniques is well appropriate for game analytics, since the mechanics of 3D game worlds typically cause asymmetric relations between map locations; for instance, it may be possible to jump down from a ledge but impossible to quickly get back up again. We discussed the theory behind asymmetric matrix factorization, introduced a novel algorithm for accelerated DEDICOM, and provided implementation examples for interested readers to work with.

Our results show that, given adjacency matrices of weighted waypoint similarity graphs, DEDICOM and DESICOM consistently produce meaningful and interpretable clusters from player trajectories. In short our results suggest that, if speed is pivotal, analysts should resort to orthogonal DEDICOM whereas if ease of interpretation is more important, it seems preferable to use DESICOM.

REFERENCES

### Table 1: Comparative Analysis of Movements on the Quake III Map q3dm17

<table>
<thead>
<tr>
<th></th>
<th>red</th>
<th>green</th>
<th>blue</th>
<th>cyan</th>
<th>magnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camper</td>
<td>3.92</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>6.77</td>
</tr>
<tr>
<td>Expert</td>
<td>31.63</td>
<td>0.08</td>
<td>0.10</td>
<td>0.34</td>
<td>0.19</td>
</tr>
<tr>
<td>Semi</td>
<td>88.14</td>
<td>0.52</td>
<td>0.15</td>
<td>0.43</td>
<td>2.07</td>
</tr>
</tbody>
</table>

(a) Data and results for a “camper”

Fig. 7: Comparative analysis of movements of different players on the Quake III map q3dm17; see the text for details.

---

References:


