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

Christian Bauckhage*†, Rafet Sifa†, Anders Drachen‡, Christian Thurau§ and Fabian Hadiji§
*B-IT, University of Bonn, Bonn Germany
†Fraunhofer IAIS, St. Augustin, Germany
‡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 for analyzing the design of maps. Although the idea of analyzing the spatio-temporal behavior of players attracts growing attention [1], [10]–[13] and all major developers currently use some form of spatial analytics [3], publicly available information on spatio-temporal gameplay is surprisingly scarce [14]. This may be due to the novelty of the idea but also related to the fact that telemetry data from commercial games is proprietary. Practical examples in this paper therefore pertain to well known FPS games whose source code or API definitions are publicly available.

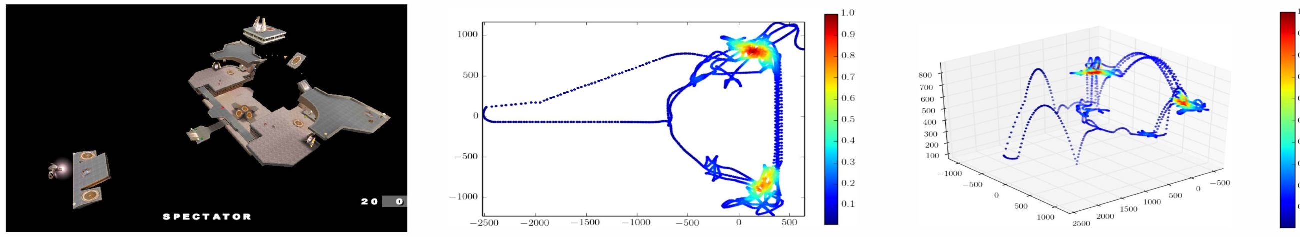
In practice, evaluating and tuning the design of maps, from simple open area maps to complex multi-layered maps with triggers, mission conditions etc., operates at a highly granular level [3], [4]. For example, a 3D heatmap visualizing event frequencies on a map can reveal that a specific area sees too high or too low a concentration of the feature being investigated (death events, kill events, purchases, . . .). 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, trajectory data were reported to be valuable [2], [4], [12], [14].

Addressing the need for a meaningful partition of maps based on player behavior, an alternative to heatmaps is to apply unsupervised machine learning techniques and spatial clustering to trajectory data so as to identify movement preferences [14]. Spatial clustering attempts to group 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 move between these, allows for identifying areas of a map which may require closer analysis. This is notably useful when the map geometry is complex or when large scale behavioral data sets are available. Indeed, in these situations, map partitioning is not only useful but a necessity [3], [4], [12]. However, any such partition should be based on objective measurements rather than on subjective decisions and the chosen approach should reflect patterns of spatial behavior on the map.

II. Contribution

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

Two of these techniques (DEDICOM and DESICOM) have never before been applied in games research. We show that they allow for localizing hotspots 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 our knowledge, spatial clustering



(a) aerial view of the *Quake III* map q3dm17 (b) 2D heatmap (c) 3D heatmap

Fig. 1: Recordings of in-game activities allow for visualizing simple spatial statistics as to crucial events and movement behaviors of players. The techniques discussed in the text enable a more in-depth analysis of the spatio-temporal behavior of players.

techniques that retain information about movements 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 intricate maps and provide mechanisms for recording a player's in-game activities in so called *demo* files.

We parse recorded demo files for player positions and obtain trajectories $\boldsymbol{x}_1 \rightarrow \boldsymbol{x}_2 \rightarrow \ldots \rightarrow \boldsymbol{x}_T$ where $\boldsymbol{x}_t \in \mathbb{R}^3$ represents a player's location at time t. Figure 2b shows 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 game world. 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 trajectory data and obtain sets of n prototypical *waypoints* $W = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n\}$ where $\boldsymbol{w}_i \in \mathbb{R}^3$ and $n \in O(100)$.

Having extracted waypoints from a 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 first assign each $\boldsymbol{x}_t$ to its closest waypoint $\boldsymbol{w}_i = \boldsymbol{w}(\boldsymbol{x}_t)$ and then determine every move $\boldsymbol{x}_t \rightarrow \boldsymbol{x}_{t+1}$ where $\boldsymbol{w}(\boldsymbol{x}_t) \neq \boldsymbol{w}(\boldsymbol{x}_{t+1})$. 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. Hence, if $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t+1}$ reside in different meshes represented by $\boldsymbol{w}_i$ and $\boldsymbol{w}_j$, the pair $(\boldsymbol{w}_i, \boldsymbol{w}_j)$ is added to the edges of G.

Waypoint transition graphs provide an informative data structure for game analytics as they characterize how 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 from 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 graph may be weighted to encode further details as to player behaviors or relations among waypoints.

For instance, in this paper, we consider two kinds of weighting schemes. On the one hand, we compute the shortest Euclidean path length l_{ij} between $\boldsymbol{w}_i$ and $\boldsymbol{w}_j$ along the edges in the waypoint graph. We then normalize the l_{ij} to $d_{ij} \in [0, 1]$ and, by setting $s_{ij} = 1 - d_{ij}$, obtain a measure of the *spatial similarity* between any two points. On the other hand, we count the number of transitions $\boldsymbol{w}_i \rightarrow \boldsymbol{w}_j$ observed from a set of trajectory data and simply set s_{ij} to this number. This way, we obtain a *temporal similarity* between any two waypoints. Note that spatial similarity does not imply temporal similarity and vice versa. For instance, two waypoints may be spatially close but temporally far apart, simply because the observed player just rarely transited between them.

Next, we review algorithms for player trajectory analysis and discuss benefits of methods that allow for considering asymmetric relationships encoded in a waypoint graph.

IV. Clustering Player Trajectory Data

Clustering features 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 techniques tailored to asymmetric similarity data. For these we provide reference implementations so that interested readers can replicate our results.

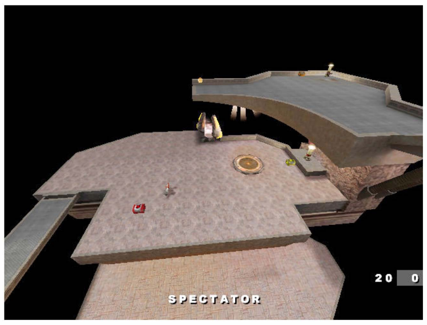
A. *k-means Clustering*

The k-means algorithm is a simple and therefore popular clustering technique. However, it operates on implicit assumptions which do not apply to every kind of data [17], [18] and we briefly expose its limitations in learning coherent map partitions from player trajectories.

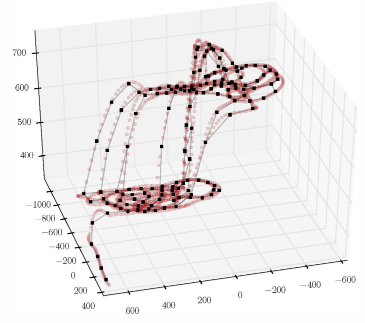
Recall that we are concerned with finding latent spatial structures within a sample $W = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n\}$ of waypoints. Using k-means, we tacitly assume structures to correspond to spatial densities, since the algorithm represents clusters C_i by centroids $\boldsymbol{m}_i$ and groups data w.r.t. distances to centroids. This reduces k-means clustering to the problem of finding appropriate centroids which is accomplished by minimizing

$$E(k) = \sum_{i=1}^{k} \sum_{\boldsymbol{w}_j \in C_i} \left\| \boldsymbol{w}_j - \boldsymbol{m}_i \right\|^2 \tag{1}$$

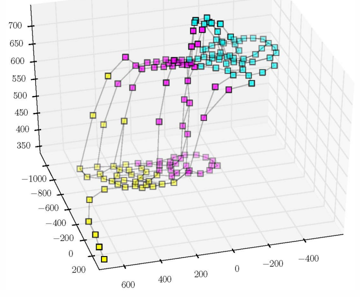
over the $\boldsymbol{m}_i$. To this end, k-means resorts to greedy optimization. When started, it randomly initializes $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_k$. Given these initial guesses, the algorithm determines k clusters, then updates its estimates of the cluster centroids, and repeats both steps until clusters stabilize.



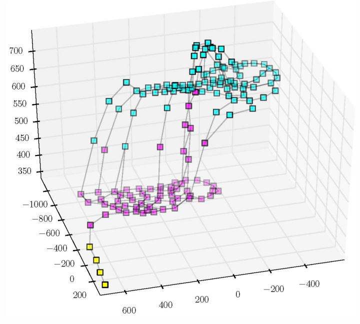
(a) part of q3dm17



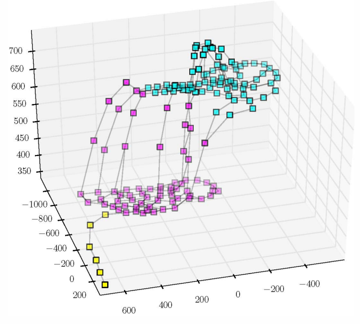
(b) player trajectories and waypoints



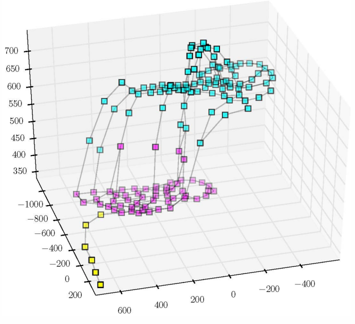
(c) k-means clustering



(d) spectral clustering



(e) DEDICOM



(f) DESICOM

Fig. 2: Outcomes of different clustering methods on trajectory data. (a) screenshot of a part of the *Quake III* map q3dm17. (b) plot of a player trajectory on this map and an automatically determined waypoint graph. (c)–(f) clusters found in the waypoint graph. Spectral clustering, DEDICOM, and DESICOM identify structures that reflect the topology or architecture of the map.

Note that this procedure implicitly fits a Gaussian mixture model. To expose this behavior, we consider an arbitrary cluster C_i which contributes a term of

$$\sum_{\boldsymbol{w}_j \in C_i} \left\|\boldsymbol{w}_j - \boldsymbol{m}_i\right\|^2 = \left\|\boldsymbol{X}_i\right\|^2 \tag{2}$$

to (1) where the columns of matrix $\boldsymbol{X}_i$ are given by $\boldsymbol{w}_j - \boldsymbol{m}_i$. Now, recall that $\left\|\boldsymbol{X}_i\right\|^2 = \text{tr}[\boldsymbol{X}_i\boldsymbol{X}_i^T]$. Since $\boldsymbol{\Sigma} = \boldsymbol{X}_i\boldsymbol{X}_i^T$ is a covariance matrix, we may apply the spectral decomposition

$$\boldsymbol{\Sigma} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T \tag{3}$$

where $\boldsymbol{U}$ is an orthogonal matrix of eigenvectors of $\boldsymbol{\Sigma}$ and the eigenvalues in the diagonal matrix $\boldsymbol{\Lambda}$ indicate variances along the principal axes of cluster C_i. Finally, recall that traces are invariant under similarity transformations, i.e. $\text{tr}[\boldsymbol{\Sigma}] = \text{tr}[\boldsymbol{\Lambda}]$. By minimizing (1), the k-means algorithm therefore minimizes traces of covariance matrices and thus produces compact convex clusters of low internal variance. This can be seen in Fig. 2c where k-means clustering clustered the waypoints into blob-like structures some of which consist of points from geometrically distinct parts of the map.

B. Spectral Clustering

While k-means clustering is guided by local properties of data, spectral clustering assumes a more global view. Given a sample $W = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n\}$, it considers pairwise similarities of elements of W which are gathered in a matrix $\boldsymbol{S} \in \mathbb{R}^{n \times n}$ where large entries $S_{ij} = s(\boldsymbol{w}_i, \boldsymbol{w}_j)$ indicate high affinities.

Spectral clustering is thus closely related to the problem of graph partitioning, because $\boldsymbol{S}$ can be seen as a weighted graph adjacency matrix. In [19], Fiedler showed that spectral decompositions as in (3) allow for graph partitioning and that clusters can be determined from looking at the eigenvectors belonging to the k smallest eigenvalues of the graph Laplacian $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{S}$ where $\boldsymbol{D}$ 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

$$\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{S}\boldsymbol{D}^{-\frac{1}{2}} \tag{4}$$

of $\boldsymbol{S}$, we compute its decomposition, determine the k smallest eigenvalues, collect the corresponding eigenvectors in a matrix $\boldsymbol{U} \in \mathbb{R}^{n \times k}$, apply k-means to the n rows of $\boldsymbol{U}$, and thus group the waypoints in $W = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n\}$. 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 shortcomings w.r.t. our problem. First of

Algorithm 1 python code for spectral clustering

```
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, evcts = eig(L)

# extract "smallest" k eigenvectors
sortedevals = argsort(evals)
U = evcts[:,sortedevals[:k]]

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

all, as it relies on the spectral theorem, it requires symmetric matrices $\boldsymbol{S} = \boldsymbol{S}^T$ in order to work properly. Yet, game physics and player behaviors may cause asymmetric spatio-temporal relations. Consider a *QuakeIII* player stepping into a teleporter. If $\boldsymbol{w}_i$ and $\boldsymbol{w}_j$ are the closest waypoints before and after teleporting, then $s(\boldsymbol{w}_i, \boldsymbol{w}_j) \neq s(\boldsymbol{w}_j, \boldsymbol{w}_i)$. Therefore, waypoint graphs determined from player trajectories are usually directed and for similarity matrices derived therefrom we have $\boldsymbol{S} \neq \boldsymbol{S}^T$.

Second of all, spectral clustering of similarity matrices does not provide a characterization of affinities among clusters. Given a similarity matrix $\boldsymbol{S} \in \mathbb{R}^{n \times n}$, spectral clustering groups a set of n objects into k clusters but does not produce a matrix $\boldsymbol{R} \in \mathbb{R}^{k \times k}$ which relates these clusters. Next, we discuss two approaches that 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 the best of our 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 $\boldsymbol{S} \in \mathbb{R}^{n \times n}$ of asymmetric relations among n objects, DEDICOM considers the factorization problem

$$\boldsymbol{S} \approx \boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^T \tag{5}$$

where $\boldsymbol{A} \in \mathbb{R}^{n \times k}$ and $\boldsymbol{R} \in \mathbb{R}^{k \times k}$. This resembles the spectral decomposition in (3), but matrix $\boldsymbol{A}$ is of rank $k \ll n$ and not necessarily orthogonal and $\boldsymbol{R}$ is dense rather than diagonal. Once determined, the columns of $\boldsymbol{A}$ can be understood as k "basis" vectors or latent factors behind the relationships encoded in $\boldsymbol{S}$ and matrix $\boldsymbol{R}$ 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 $\boldsymbol{R}$.

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

$$\min_{\boldsymbol{A},\boldsymbol{R}} \left\| \boldsymbol{S} - \boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^T \right\|^2 \tag{6}$$

which is convex in $\boldsymbol{R}$ but neither in $\boldsymbol{A}$ nor in the product $\boldsymbol{A}\boldsymbol{R}$. Known algorithms [27], [28] therefore randomly initialize $\boldsymbol{A}$ and $\boldsymbol{R}$ 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 $\boldsymbol{A}$ by considering the situation after stacking matrix $\boldsymbol{S}$ and matrix $\boldsymbol{S}^T$ side by side. This yields

$$\left[\boldsymbol{S}\ \boldsymbol{S}^T\right] = \left[\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^T\ \boldsymbol{A}\boldsymbol{R}^T\boldsymbol{A}^T\right] = \boldsymbol{A}\left[\boldsymbol{R}\boldsymbol{A}^T\ \boldsymbol{R}^T\boldsymbol{A}^T\right] \tag{7}$$

and allows for solving for $\boldsymbol{A}$ if $\boldsymbol{A}^T$ is held fixed [25]. To see how, we substitute $\boldsymbol{T} = [\boldsymbol{S}\ \boldsymbol{S}^T]$ and $\boldsymbol{B} = [\boldsymbol{R}\boldsymbol{A}^T\ \boldsymbol{R}^T\boldsymbol{A}^T]$ and write (7) as $\boldsymbol{T} = \boldsymbol{A}\boldsymbol{B}$. Resorting to the pseudo inverse $\boldsymbol{B}^\dagger$ of $\boldsymbol{B}$, we find

$$\boldsymbol{A} = \boldsymbol{T}\boldsymbol{B}^\dagger = \boldsymbol{T}\boldsymbol{B}^T\left(\boldsymbol{B}\boldsymbol{B}^T\right)^{-1} \tag{8}$$

which, after reversing our substitution, provides the following update for matrix $\boldsymbol{A}$

$$\boldsymbol{A} \leftarrow \left(\boldsymbol{S}\boldsymbol{A}\boldsymbol{R}^T + \boldsymbol{S}^T\boldsymbol{A}\boldsymbol{R}\right)\left(\boldsymbol{R}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{R}^T + \boldsymbol{R}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{R}\right)^{-1}. \tag{9}$$

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

$$\boldsymbol{R} \leftarrow \boldsymbol{A}^\dagger \boldsymbol{R} \boldsymbol{A}^{T\dagger} = \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\boldsymbol{S}\,\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1} \tag{10}$$

and steps (9) and (10) have to be repeated until convergence.

Looking at (9) and (10), we note that matters simplify, if we constrain the solution for $\boldsymbol{A}$. Requiring its columns to be orthogonal unit vectors in $\mathbb{R}^n$, we have $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}$ and (9) becomes

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

As the updated $\boldsymbol{A}$ may not be orthogonal, we determine $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{T}$ where $\boldsymbol{Q}$ is orthogonal and $\boldsymbol{T}$ is upper triangular.[1] We then set $\boldsymbol{A} \leftarrow \boldsymbol{Q}$ and compute

$$\boldsymbol{R} \leftarrow \boldsymbol{A}^T\boldsymbol{S}\,\boldsymbol{A}. \tag{12}$$

Upon convergence of this algorithm, we apply k-means clustering to the n rows of $\boldsymbol{A}$, and thus obtain cluster labels for the waypoints in $W = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n\}$. Algorithm 2 shows a python implementation that summarizes our derivation.

Constraining $\boldsymbol{A}$ 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 $\boldsymbol{A}$ 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

[1]Note that this is the well known QR decomposition but, to avoid confusion with matrix $\boldsymbol{R}$ from DEDICOM, we write $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{T}$.

Algorithm 2 python code for orthogonal DEDICOM

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

R = random.rand(k, k)
A = random.rand(n, k)
A, T = qr(A)

errold = inf
for t in range(100):
    RRt  = dot(R, R.T)
    RtR  = dot(R.T, R)
    SARt = dot(S, dot(A, R.T))
    StAR = dot(S.T, dot(A, R))

    # update matrix A
    A = dot(SARt + StAR, inv(RRt + RtR))
    A, T = qr(A)

    # update matrix R
    R = dot(A.T, dot(S, A))

    errnew = norm(S - dot(A, dot(R, A.T)))
    if abs(errold-errnew) < 0.00001:
        break
    else:
        errold = errnew

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

anyway [29], [30]. Indeed, experiments with unconstrained and 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 $\boldsymbol{R}$ is given by

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

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

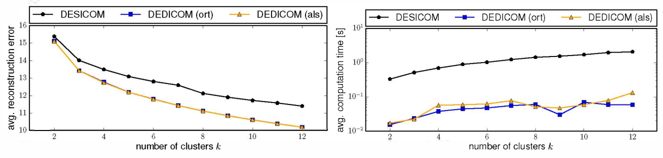
In [31], Kiers presented DESICOM, a decomposition into *simple* directional components. His approach computes a sparse factor matrix $\boldsymbol{A}$ such that each of its n rows contains only one entry different from zero. This is accomplished using a procedure that updates $\boldsymbol{A}$ one row at a time, then updates $\boldsymbol{R}$ using (10), 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.

Algorithm 3 python code for DESICOM

```
from numpy import *
from numpy.linalg import *

R = random.rand(k, k)
A = random.rand(n, k)

errold = inf
for t in range(100):
    # row-wise update of matrix A
    for i in range(n):
        ci = delete(S[:,i], i).reshape(n-1,1)
        ri = delete(S[i,:], i).reshape(n-1,1)
        v  = vstack((ci, ri))
        Ai = delete(A, (i), axis=0)
        M  = vstack((dot(Ai, R), dot(Ai, R.T)))

        hvals = zeros(k)
        avals = zeros(k)
        # in row i, iterate over all columns
        for l in range(k):
            f1 = dot(M[:,l], M[:,l])
            f2 = dot(v.T, M[:,l])[0]
            c4 = R[l,l]**2
            c2 = f1-2*S[i,i]*R[l,l]
            c1 = -2*f2
            roots = roots([4*c4, 0, 2*c2, c1])
            h = ones(3) * inf
            for r in range(len(roots)):
                rt = roots[r]
                h[r] = c4*rt**4 + c2*rt**2 + c1*rt
            r = argmin(h)
            hvals[l] = h[r]
            avals[l] = roots[r]
        minl = argmin(hvals)

        # compute candidate for new row i of A
        anew = zeros(k)
        anew[minl] = avals[minl]
        # replace old row with new row only if this
        # does not cause any column of A to become 0
        s = sum(Ai, axis=0)
        if min(abs(s + anew)) > 0.000001:
            A[i,:] = anew

    # update matrix R
    AtAi = inv(dot(A.T, A))
    AtSA = dot(A.T, dot(S, A))
    R = dot(AtAi, dot(AtSA, AtAi))

    errnew = norm(S - dot(A, dot(R, A.T)))
    if abs(errold-errnew) < 0.00001:
        break
    else:
        errold = errnew

# cluster rows of A
labels = argmax(A, axis=1)
```

DESICOM is slower than any of the algorithm discussed so far (see Fig. 3) but has several intriguing features. First of all, clusters can be read off matrix $\boldsymbol{A}$ directly. Since for each row index j there is a single column index i such that $A_{ji} > 0$, we simply assign waypoint $\boldsymbol{w}_j$ 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 $\boldsymbol{S}$. This is desirable since similarity matrices are typically non-negative themselves. Third of all, since $\boldsymbol{A}$ is non-negative we may 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 $\boldsymbol{R}$ and thus obtain a compressed similarity matrix whose entries resemble probabilities.

Figure 2f shows that DESICOM performs similar to the two previous methods. However, (normalized) cluster affinity



(a) reconstruction error (b) computation time

Fig. 3: Quantitative performance evaluation of DEDICOM and DESICOM. The graphs show averages over computations with 10 similarity matrices of size 250×250. Regarding reconstruction accuracy, orthogonal and unconstrained DEDICOM are indistinguishable but the orthogonal version is slightly faster.

matrices resulting from DESICOM are more easy to interpret. Here, we found

$$\boldsymbol{R} = \begin{bmatrix} 1.00 & 0.00 & 0.00 \\ 0.00 & 0.92 & 0.08 \\ 0.00 & 0.03 & 0.97 \end{bmatrix}$$

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 Partitionings

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 apartment. 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

$$\boldsymbol{R} = \begin{array}{r|cccc} & \text{yellow} & \text{red} & \text{blue} & \text{green} \\ \hline \text{yellow} & 0.36 & 0.14 & 0.10 & 0.40 \\ \text{red} & 0.12 & 0.32 & 0.11 & 0.46 \\ \text{blue} & 0.08 & 0.16 & 0.47 & 0.29 \\ \text{green} & 0.11 & 0.08 & 0.12 & 0.69 \end{array}$$

from which we read, among others, that it is unlikely for the player to transit from the red to the blue cluster but that the player did indeed frequently jump from waypoints in the red cluster down to waypoints in the green cluster. We also recognize the green cluster to be of high self-affinity; this indeed corresponds to the hotspot in the recording player's locations visible on the bottom of the heatmap in Fig. 4.

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 $k = 5$ 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 blue cluster which only interacts with the transitional yellow cluster. We also note that, according to the unnormalized matrix $\boldsymbol{R}$, affinities between the 5 clusters found are generally rather low.

The second player was an experienced 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 identified by DESICOM and corresponds to the green cluster. In contrast to the "camper", we observe that cluster affinities found for the second player are higher which indicates his propensity to move between 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 passed through more frequently (the blue, green, and orange 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

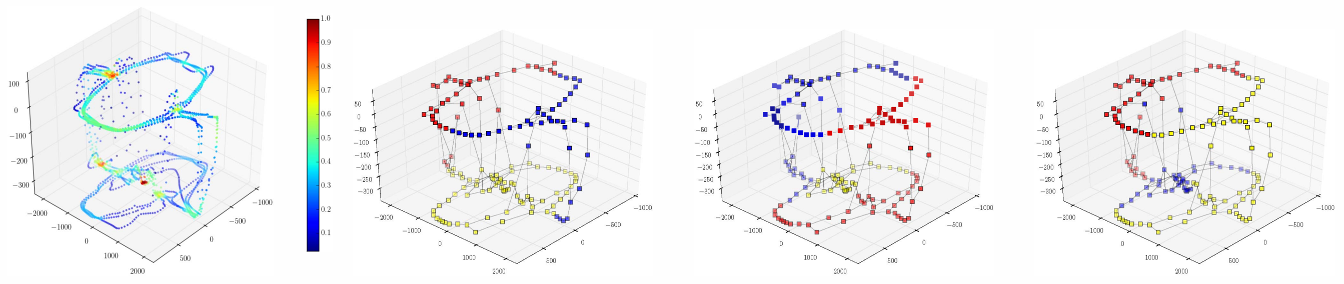

(a) trajectory heat map (b) spectral clustering (c) DEDICOM (d) DESICOM

Fig. 4: Player trajectories on the *UT 2003* map Gael. Given a symmetric waypoint transition graph, spectral clustering recognizes physically coherent structures of the map. Yet, as they operate on asymmetric transition data, DEDICOM and DESICOM, uncover clusters that more faithfully represent the recording player's behavior. Readers familiar with the map will recognize the location where the player had to jump to get a health kit while avoiding a lethal pit. Both methods also reveal that the player traversed the lower part of the map in a non-circular manner.

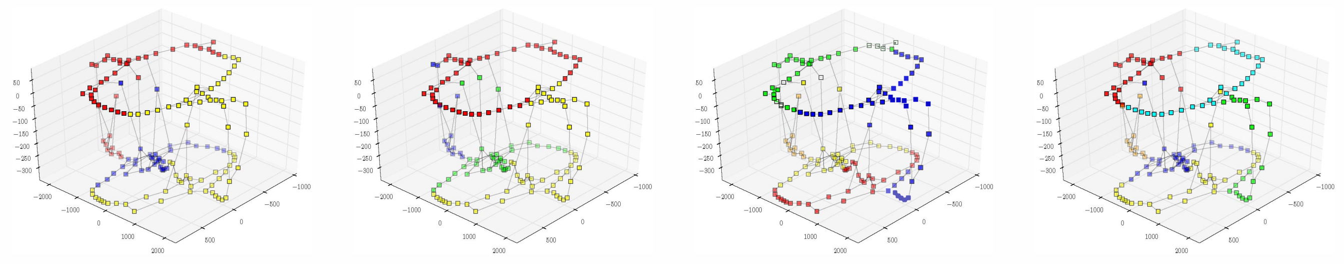

(a) $k = 3$ (b) $k = 4$ (c) $k = 5$ (d) $k = 6$

Fig. 5: DESICOM applied to waypoint similarity data automatically identifies hotspots in a waypoint transition graph. For instance, the blue cluster in (a), the green in (b), the yellow in (c), and the blue in (d) correspond to the preferred player location visible on the bottom of the heatmap in Fig. 4. This effect persists, if the number k of clusters to be determined is increased.

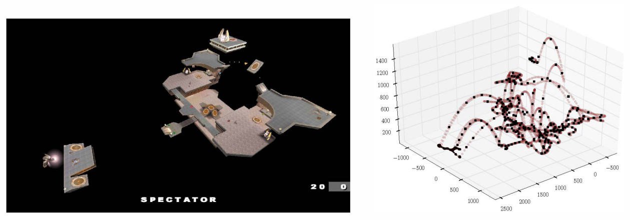


Fig. 6: Behavior-based waypoints for *Quake III* q3dm17.

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 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 weighted adjacency matrices of waypoint 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

[1] C. Thompson, "Halo 3: How Microsoft Labs Invented a New Science of Play," Wired Magazine, September 2007.

[2] B. Medler, "Play with Data – An Exploration of Play Analytics and Its Effect on Player Experiences," Ph.D. dissertation, Georgia Institute of Technology, 2012.

[3] M. Seif El-Nasr, A. Drachen, and A. Canossa, Eds., *Game Analytics – Maximizing the Value of Player Data*. Springer, 2013.

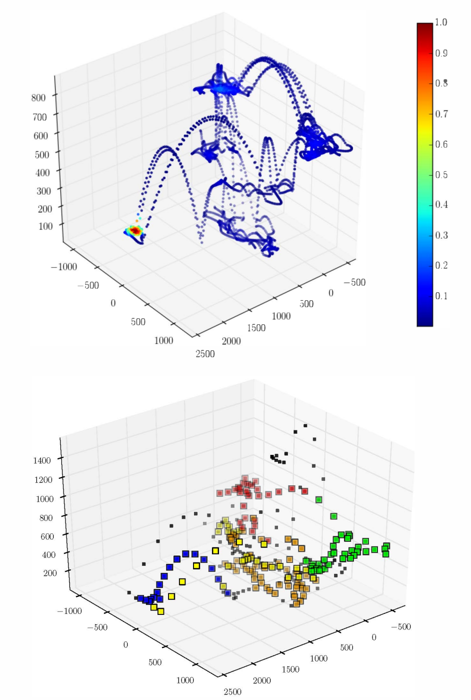
[4] B. Harrison and D. Roberts, "Arrrgghh!!!: Blending Quantitative and Qualitative Methods to Detect Player Frustration," in *Proc. FDG*, 2011.

[5] C. Bauckhage and C. Thurau, "Towards a Fair 'n Square Aimbot – Using Mixtures of Experts to Learn Context Aware Weapon Handling," in *Proc. GAME-ON*, 2004.

[6] C. Thurau, C. Bauckhage, and G. Sagerer, "Learning Human-Like Movement Behavior for Computer Games," in *Proc. SAB*, 2004.

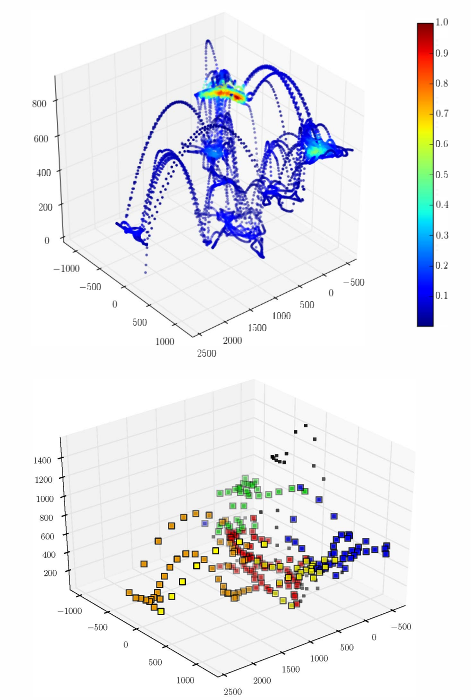
[7] D. Hale, G. Youngblood, and P. Dixit, "Automatically Generated Convex Region Decomposition for Real-time Spatial Agent Navigation in Virtual Worlds," in *Proc. AIIDE*, 2008.

[8] S. Hladky and V. Bulitko, "An Evaluation of Models for Predicting Opponent Positions in First-Person Shooter Video Games," in *Proc. CIG*, 2008.



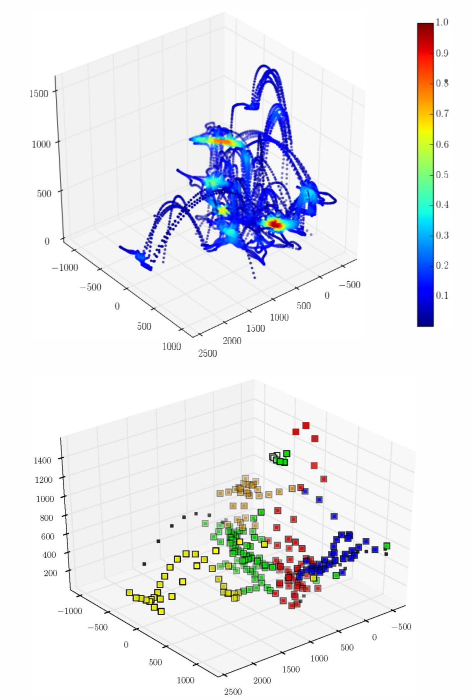
	yellw	red	blue	green	orng
yellw	0.39	0.00	0.03	0.02	0.11
red	0.00	0.07	0.00	0.00	0.00
blue	0.01	0.00	11.43	0.00	0.00
green	0.01	0.01	0.00	8.55	0.00
orng	0.01	0.00	0.00	0.08	38.35

(a) data and results for a “camper”



	yellw	red	blue	green	orng
yellw	29.98	0.20	0.25	0.00	0.01
red	0.03	15.91	0.04	0.10	0.03
blue	0.18	0.01	1.79	0.08	0.00
green	0.00	0.10	0.02	12.30	0.00
orng	0.13	0.04	0.00	0.05	1.02

(b) data and results for an experience player



	yellw	red	blue	green	orng
yellw	4.42	0.00	0.11	0.03	0.00
red	0.00	0.10	0.07	0.11	0.00
blue	0.07	0.06	80.74	2.62	0.27
green	0.26	0.12	0.21	69.28	0.58
orng	0.01	0.02	0.38	0.31	57.57

(c) data and results for a semi professional player

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

[9] R. Sifa and C. Bauckhage, “Archetypical Motion: Supervised Game Behavior Learning with Archetypal Analysis,” in *Proc. CIG*, 2013.

[10] N. Hoobler, G. Humphreys, and M. Agrawala, “Visualizing Competitive Behaviors in Multi-User Virtual Environments,” in *Proc. VIS*, 2004.

[11] J. Miller and J. Crowcroft, “Group Movement in World of Warcraft Battlegrounds,” *Int. J. of Advanced Media and Communication*, vol. 4, no. 4, 2010.

[12] A. Drachen and A. Canossa, “Evaluating Motion: Spatial User Behavior in Virtual Environments,” *Int. J. of Arts and Technology*, vol. 4, no. 3, 2011.

[13] B. Harrison and D. Roberts, “Using Sequential Observations to Model and Predict Player Behavior,” in *Proc. FDG*, 2011.

[14] A. Drachen and M. Schubert, “Spatial Game Analytics and Visualization,” in *Proc. CIG*, 2013.

[15] A. Drachen, R. Sifa, C. Bauckhage, and C. Thurau, “Guns, Swords, and Data: Clustering of Player Behavior in Computer Games in the Wild,” in *Proc. CIG*, 2012.

[16] C. Bauckhage, M. Roth, and V. Hafner, “Where am I? – On Providing Gamebots with a Sense of Location Using Spectral Clustering of Waypoints,” in *Proc. SAB Workshop on Player Satisfaction*, 2006.

[17] D. Arthur and S. Vassilvitskii, “k-means++: The Advantages of Careful Seeding,” in *Proc. SODA*, 2007.

[18] D. Aloise, A. Deshapande, P. Hansen, and P. Popat, “NP-Hardness of Euclidean Sum-of-Squares Clustering,” *Machine Learning*, vol. 75, no. 2, 2009.

[19] M. Fiedler, “A Property of Eigenvectors of Nonnegative Symmetric Matrices and its Application to Graph Theory,” *Czechoslovak Mathematical J.*, vol. 25, no. 4, 1975.

[20] A. Ng, M. Jordan, and Y. Weiss, “On Spectral Clustering: Analysis and an Algorithm,” in *Proc. NIPS*, 2001.

[21] I. Dhillon, Y. Guan, and B. Kulis, “Kernel k-means, Spectral Clustering and Normalized Cuts,” in *Proc. KDD*, 2004.

[22] U. von Luxburg, “A Tutorial on Spectral Clustering,” *arXiv*, 2007.

[23] K. Rohe, S. Chatterjee, and B. Yu, “Spectral Clustering and the High-dimensional Stochastic Block Model,” *The Annals of Statistics*, vol. 39, no. 4, 2011.

[24] R. Harshman, “Models for Analysis of Asymmetrical Relationships among N Objects or Stimuli,” in *Proc. Joint Meeting of the Psychometric Society and the Society for Mathematical Psychology*, 1978.

[25] B. Bader, R. Harshman, and T. Kolda, “Temporal Analysis of Semantic Graphs using ASALSAN,” in *Proc. ICDM*, 2007.

[26] J. Kunegis and J. Fliege, “Predicting Directed Links Using Nondiagonal Matrix Decompositions ,” in *Proc. ICDM*, 2012.

[27] H. Kiers, J. ten Berge, Y. Takane, and J. de Leeuw, “A Generalization of Takane’s Algorithm for DEDICOM,” *Psychometrika*, vol. 55, no. 1, 1990.

[28] Y. Takane and Z. Zhang, “Algorithms for DEDICOM: Acceleration Deceleration or Neither?” *J. of Chemometrics*, vol. 23, no. 7–8, 2009.

[29] D. Donoho and J. Tanner, “Neighborliness of Randomly Projected Simplices in High Dimensions,” *PNAS*, vol. 102, no. 27, 2005.

[30] P. Hall, J. Marron, and A. Neeman, “Geometric Representations of High Dimension, Low Sample Size Data,” *J. Royal Statistical Society B*, vol. 67, no. 3, 2005.

[31] H. Kiers, “DESICOM: Decomposition of Asymmetric Relationships Data into Simple Components,” *Behaviormetrika*, vol. 24, no. 2, 1997.