Voxel-Based Spatio-Temporal Visualization of Gameplay Traces with Anomaly Detection

Ling Liu[1], Colan Biemer[1], Günter Wallner[2,3] and Seth Cooper[1]

[1]Northeastern University
[2]Johannes Kepler Universität Linz
[3]Eindhoven University of Technology

Abstract

As video games take place across space and time, space-time visualizations are promising for conveying gameplay data, as they allow all the data to be shown at once and avoid, e.g., timeline scrubbing. However, space-time visualizations can suffer from clutter and occlusion of data. To address these issues, we explore AI techniques to visualize space-time gameplay traces. Clutter is reduced by aggregating the data of individual traces into voxels. To highlight areas of interest and reduce their occlusion, anomaly detection using isolation forests is used to visually emphasize voxels of interest, changing color and opacity. We present two case studies, based on *StarCraft: Brood War* and a custom platformer game, demonstrating the impact of the anomaly detection approach.

Keywords

gameplay visualization, voxels, anomaly detection

1. Introduction

Modern single- and multiplayer video games feature a rich set of player behaviors, often contained in playtraces encompassing movement data. However, movement data is complex to analyze and visualize as it unfolds over space and time and becomes very large very quickly (cf. [1]). As such, traditional visual analysis for 2D games, when faced with massive amounts of player data, often relies on multiple views to present temporal and spatial data separately or resorts to animations. Such solutions (e.g., [2]) often require manual adjustments to the viewport and timelines to analyze the detailed information. Multiple views, on the other hand, come with additional cognitive costs such as increased load on working memory and effort for making comparisons [3]. Combining space and time in a single view — such as in a space-time cube — is, however, prone to visual clutter and occlusions [4, 5], and thus only feasible for the analysis of a small amount of movement traces.

That, however, means that such solutions are not feasible for games such as multiplayer real-time strategy (RTS) games like *StarCraft: Brood War* [6] which involve large numbers of units moving simultaneously. Furthermore, if movement data is augmented with further contextual data such as key combat events these often get obscured. Even in singleplayer games, such as platform games, multiple attempts can be considered in aggregate as a large number of players needs to be visualized to understand common patterns. Given the large-scale nature, prior work has thus often resorted to static or time-agnostic visualizations [7, 8], which — due to neglecting the time dimension — struggle to convey dynamic player interactions.

To address these limitations, we propose an AI-assisted 3-dimensional spatio-temporal visualization framework that: (1) aggregates raw trajectories into voxels to reduce visual clutter, and (2) applies anomaly detection to highlight regions of interest. We built a voxel-based aggregation method for spatio-temporal game data that preserves the essential details of the movement and reduces the clutter.

Joint AIIDE Workshop on Experimental Artificial Intelligence in Games and Intelligent Narrative Technologies, November 10-11, 2025, Edmonton, Canada.

liu.ling1@northeastern.edu (L. Liu); biemer.c@northeastern.edu (C. Biemer); guenter.wallner@jku.at (G. Wallner); se.cooper@northeastern.edu (S. Cooper)
0000-0003-4993-9548 (C. Biemer); 0000-0002-0815-5985 (G. Wallner); 0000-0003-4504-0877 (S. Cooper)

We took a framework for anomaly detection — isolation forests [9] — and applied it to create a visualization that emphasizes rare movements: by increasing the transparency of voxels containing more common movements, rarer voxels are emphasized. Prior work in game analytics [10, 8] has highlighted the importance of outliers that deviate from common movement behavior as those are often more informative (e.g., for detecting unintended paths).

To demonstrate our approach, we conducted case studies across two different game genres: a real-time strategy game, *StarCraft: Brood War* [6], and a custom-built platformer game, *Recformer*. By applying our model to these two types of game data, we visually filtered out common movements and retained the interesting patterns that game researchers are more likely to be interested in. The case studies provide a subjective visual comparison of the isolation forest-based emphasis versus two simpler baselines: a sliding-window emphasis and an emphasis based solely on speed.

2. Related Work

Gameplay Analytics. As video games become increasingly complex, the field of game analytics has emerged as a valuable asset for understanding various aspects of player behavior [11]. Drachen and Schubert [12] proposed that game analytics can be based on four types of information: the attributes of the character, the involved events, the spatial information of the character, and the changes in all these types of information over time. Our work particularly focuses on the latter two, specifically movement across space and time simultaneously. Movement – given its complexity – is, however, challenging to analyze and visualize and a variety of solutions have been proposed within the field of information visualization [13].

Movement Visualization for Games. The field of game analytics has drawn upon this large heritage in geovisualization and cartography. Early work in visual game analysis has, for instance, advocated the use of geographic information systems (GIS) software [14]. Since then several methods have been proposed in the games literature ranging from visualizing individual paths (e.g., [5, 15]) to different forms of aggregation (e.g., [16, 8]), and covering both 2D (e.g., [8, 16]) as well as 3D (e.g., [5]) representations. For the sake of the completeness it should also be noted that solutions that abstract from the actual geographic coordinates such as by focusing on semantic locations (e.g., [16] have been presented as well. Our focus is, however, on the actual spatio-temporal trajectories. In doing so, our work particularly relates to the one of Sufliarsky et al. [5] in the sense that we adapt the concept of a space-time cube. However, we utilize aggregation to overcome issues of clutter when representing larger amounts of trajectories. In doing so, we draw upon work by Wallner and Drachen [7] who used alpha shapes to construct boundary polygons enclosing the points of the trajectories. However, their work focused on a 2D dimensional snapshot, rather than conveying movement over time.

Volume Visualization. In other words, we view movement as volumetric data. In this sense, our approach also relates to work in volume rendering [17] which has been heavily utilized in medical imaging. However, its application in the context of gameplay analysis has not yet been fully explored to the best of our knowledge, although some solutions such as the 3D heatmaps offered by Unity Analytics [18] touch upon it to a slight degree. One of the major challenges in volume rendering naturally lies in the ability to handle occlusions and reveal the inner structures of the volume (see also [17]) for which semi-transparency is essential (e.g., [19]) and which we also utilize in our approach.

Anomaly Detection in Game Data. To detect anomalies (rare) movements, we used Isolation Forest [9], an anomaly detection algorithm that finds more efficient isolated data through random features and recursive algorithms. Its adaptability to high-dimensional data makes it suitable for analyzing numerous player trajectories. The effectiveness of the algorithm in game analysis has been demonstrated in previous work, such as for collusion detection in multiplayer games [20].

3. System Overview

In this section we describe the general data processing pipeline that converts the raw data into the voxel format used for visualization and anomaly detection. The system takes as input the playtrace trajectories for the individual units, along with an associated team (in this work, either team 0 or team 1), in the form of sampled x, y, t coordinates, where x, y are the spatial and t the time coordinate. It produces a voxel description with a base color and emphasis value (used to control transparency and color brightness) for each voxel.

Our system aims to detect less common, anomalous player movements and emphasizing those in spatio-temporal game data by combining voxel-based visualization and isolation forests. As isolation forests are an unsupervised learning approach, they enable automated analysis without manual labeling.

A game's original 2D coordinates are discretized into a rectangular 3D grid. The granularity of the voxel structure is determined by the discretization factor f. A larger f results in coarser XY resolution, fewer grid cells, and smaller data files. The grid sizes are as below :

$$\begin{aligned} n_x &= \lfloor (x_{\max} - x_{\min})/f \rfloor + 1 \\ n_y &= \lfloor (y_{\max} - y_{\min})/f \rfloor + 1 \\ n_z &\approx \tfrac{n_x + n_y}{2} \text{ (we use Python round, ties-to-even)} \end{aligned}$$

As result, the voxels are approximately isotropic in three directions and visually symmetrical.

This gives a 3D grid of $n_x \times n_y \times n_z$ voxels. For each voxel, let D be the sum of step distances (that is, the distance between samples for units in the XY dimension) within the voxel and M the number of samples with nonzero step distance. For voxel (i, j, k), let the neighborhood window size be odd integers w_x, w_y, w_z with half-widths $r_x = (w_x - 1)/2$, $r_y = (w_y - 1)/2$, $r_z = (w_z - 1)/2$. We define the window around (i, j, k) as

$$\Omega(i,j,k) = \left\{ (u,v,w) \in [1,n_x] \times [1,n_y] \times [1,n_z] : \ |u-i| \le r_x,\ |v-j| \le r_y,\ |w-k| \le r_z \right\}$$

Each voxel containing samples then has three parameters associated with it:

$$\begin{aligned} \textit{voxel\_speed} &= D/M \\ \textit{window\_speed} &= \frac{\sum_\Omega D}{\sum_\Omega M} \\ \textit{move\_count} &= M \end{aligned}$$

In cases where the denominator is zero, the associated value is set to zero. Voxels also store which teams had samples in them. Voxels with no samples are set to null.

For isolation forest anomaly detection, we used NumPy's [21] `sklearn` [22] module. Voxel data is flattened into a $3 \cdot n_x \cdot n_y \cdot n_z$ array, and the `sklearn.preprocessing.StandardScaler` is used to set the mean to 0 and variance to 1 for normalization. Then unsupervised anomaly detection is performed at the voxel level using `sklearn.ensemble.IsolationForest` with hyperparameters:

$$\begin{aligned} \texttt{n\_estimators} &= 100 \\ \texttt{max\_samples} = \text{"auto"} &= \min(256, N) \\ \texttt{max\_features} &= 1.0 \\ \texttt{random\_state} &= 42 \end{aligned}$$

This is used to compute an anomaly score for each voxel-level feature vector, assigning higher anomaly scores to voxels that deviate from the learned distribution. The final output is a structured VTR (Visualization Toolkit [23] Rectilinear Grid) file for interactive visualization in ParaView [24] – an open source visualization platform. We use the anomaly score and team to visualize the voxels. More anomalous voxels are emphasized with brighter colors and more opacity, while less anomalous

voxels are de-emphasized with duller colors and less opacity. Different colors represent different team compositions within a voxel: red represents only team 0, blue represents only team 1, and green represents both team 0 and team 1. All color scales have the same opacity curve control points. We also visualize an image of the level or map along with the voxels to provide environmental context for interpretation of the movement data.

4. Visualization Application Comparisons: Two Case Studies

Here we illustrate the proposed voxel-based anomaly detection approach by means of two games from two different genres: *StarCraft: Brood War*, a popular multiplayer RTS game, and *Recformer*, a basic custom-built platforming game.

We present visual comparisons of the isolation forest with two baseline methods: First, using only *voxel speed* directly, in which the emphasis value is just the voxel speed normalized across all voxels. Second, the emphasis value is based on a *sliding window* for local pattern detection. For each voxel, we checked a $3_x \times 3_y \times 5_t$ neighborhood and calculated a rarity score based on the spatial distribution pattern in the sliding window.

Furthermore, we used *isolation forest* as the unsupervised anomaly detection method itself to assign emphasis values. Each voxel is represented by its local statistical features, and the isolation forest estimates how outlier-like that voxel is compared to the overall distribution. Voxels deemed more anomalous receive higher emphasis values.

The resulting visualizations (voxel speed, sliding window, and isolation forest) are shown in Figures 1 and 2. We used the same settings for all visualizations. All values are normalized to the range of 0–1. Red and blue represent voxels with data from only one out of the two teams, and green represents voxels containing both teams. Colors are mapped by fixing the hue while varying lightness and saturation, producing a perceptually monotonic gradient. The transparency curve control points (value, opacity) are:

$$\{(0, 0.1),\ (0.55, 0.4),\ (1, 1)\},$$

Here, value refers to the scalar data value of the field used for opacity mapping in ParaView, which is the same variable as for coloring. ParaView uses it for linear interpolation between adjacent control points.

StarCraft: Brood War *StarCraft: Brood War* data was obtained from the STARDATA dataset [25]. We used a *single replay* (one complete match), parsed with TorchCraft [26]. From this replay we extracted unit trajectories sampled at 8 Hz (8 fps), voxelized the coordinates into a 3D grid (X and Y from map coordinates; Z from time), and aggregated motion statistics per voxel. We used the open-source library Screp[1] to parse the game origin replay file to obtain basic game information, including maps. We then filtered out defensive structures to focus on the analysis of mobile combat units (including aircraft units). Since *StarCraft: Brood War* has a large map area and higher event density, $f = 15$ is used to control the data scale and the stability of the rendering, producing a grid size of $32_x \times 32_y \times 32_t$.

Resulting visualizations are shown in Figure 1. The time within each visualization progresses from the bottom to the top.

The voxel speed image shows that almost all voxels are transparent, with the green portion almost entirely hidden within the red and blue voxels, making it hard to observe them. The direction of movement of the two teams can be inferred by following the x, y axis along the z axis, but it is difficult to infer any players' decisions from these results.

The sliding window image reveals the vast majority of the coloring is concentrated at the extremes of the color gradient, making it difficult to discern the focus of the game.

In the isolation forest image, the overall color transition of all voxels is smoother. The intensity of the red and blue voxels increases towards the end of the game (top of the cube), and there are also many green, non-transparent voxels near the two colors representing the individual teams, likely representing

[1]Code available on GitHub: https://github.com/icza/screp

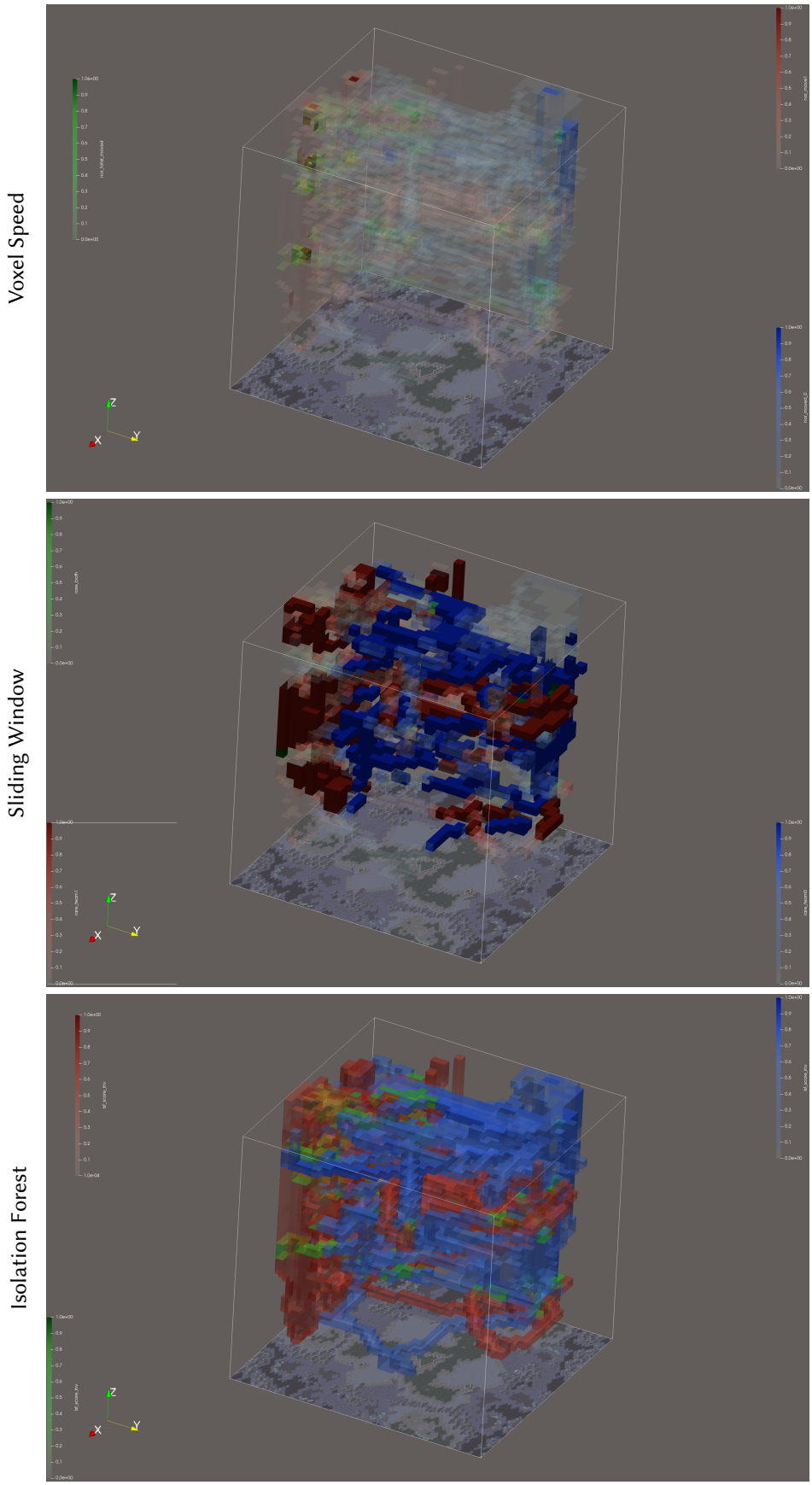

Figure 1: Comparison of different approaches to voxel emphasis in *StarCraft: Brood War*.

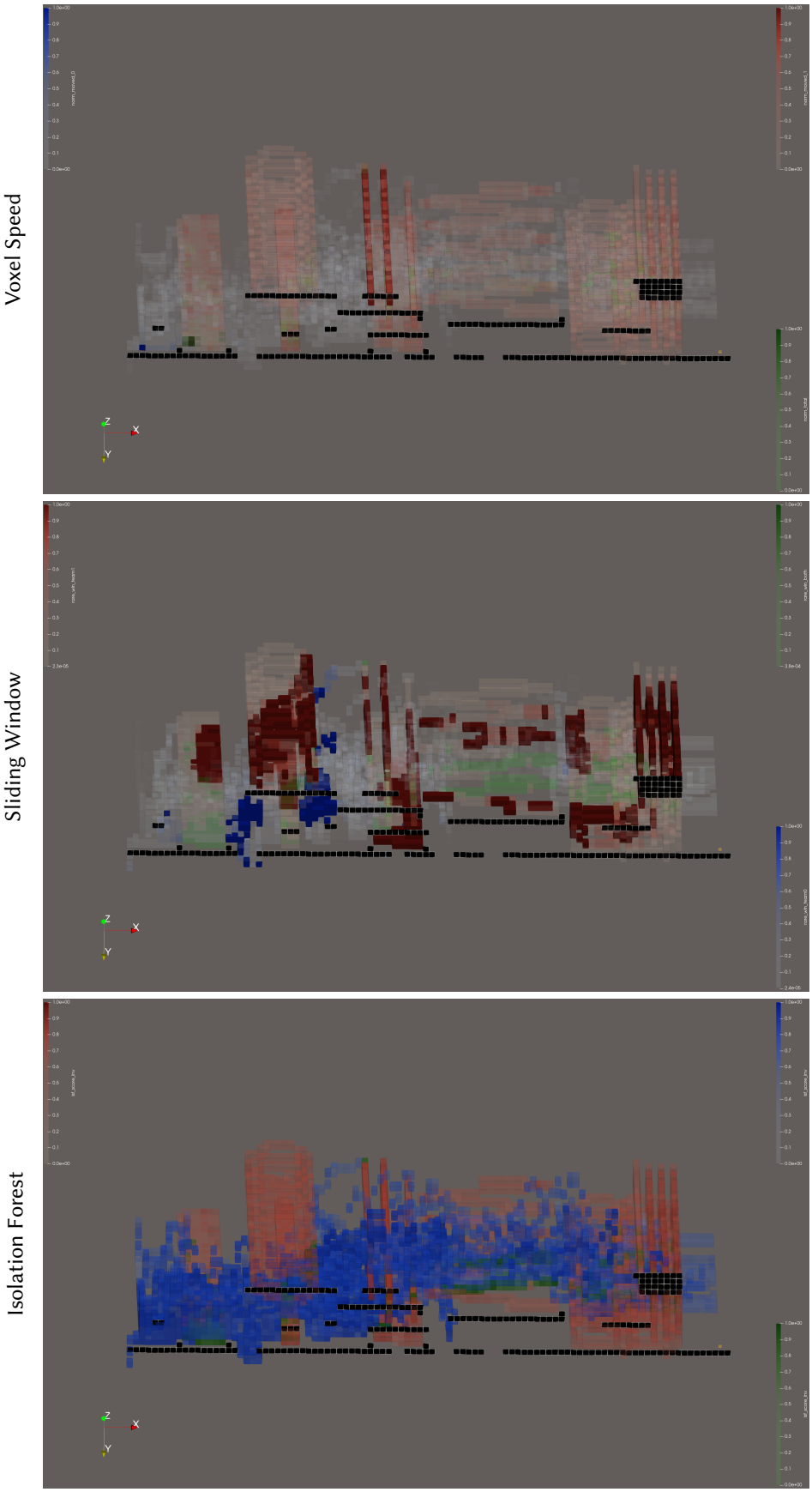

Figure 2: Comparison of different approaches to voxel emphasis in *Recformer*.

the players' final battle at the end of the game. With a slight rotation and adjustment of the viewing angle, the green voxels can be seen through the red and blue voxels, likely representing the area of intense fighting between the two teams. At the same time, some paths become transparent and filtered out. There are probably some combat units moving in non-combat areas, such as advancing or mining. For both teams, more abnormalities (more solid color) are shown at the top of the z-axis, i.e., the end of the game. This is also in line with the gameplay of *StarCraft: Brood War*, which involves collection, construction, and development in the early stages rather than direct combat.

Recformer [27] is a custom-built platforming game with simple geometric graphics, where the player must run and jump to avoid enemies and pits, while collecting all the coins to complete a level. We used a level with one coin at the end.

Recformer data was gathered specifically for this work from players recruited on the Prolific crowdsourcing platform. We recruited 20 participants who were paid $1 to play the game online and logged their gameplay data. The median time spent was 3 minutes and 17 seconds, resulting in a median pay rate of $18.26 per hour. The methods were approved by the authors' IRB, and participants consented to participate in the research before playing. As it was possible for each participant to play multiple times, we gathered 180 playtraces in total.

For singleplayer games, we consider all players' game data as team 0, and all enemies as team 1. As all the enemies in the *Recformer* level move along a fixed trajectory at a constant speed, we used a single playtrace for them. For this platform game and its smaller maps, we used a finer $f = 1$ to preserve more geometric detail. This resulted in a voxel grid of approximately $96_x \times 22_y \times 30_t$ voxels.

Note that, if viewed axis-aligned (xy-plane), the visualization in this case study resembles a heatmap. However, unlike a traditional heatmap which does not convey unfolding of the data over time, rotating the view in our visualization — since the z-axis encodes the time — allows to observe the dynamics of movement patterns and anomalies.

Resulting visualizations are shown in Figure 2. The player starts from the left and moves towards the right, and the time progresses from the front to the back.

The voxel speed image shows that most voxels are semi-transparent, with only a small number of voxels at the beginning of the game showing a prominent intensity, likely representing players starting the game collectively from the starting point.

In the sliding window results, some enemy paths coloring faded, and again only the extremes of the color gradient stand out. In particular, the players' voxels exhibit almost exclusively extreme transparency, making it difficult to explore the changes in player behavior.

In the isolation forest view, the voxel color distribution for both teams is smoother, with most of the enemy team's red becoming uniformly opaque. As in this platform game, all enemies follow the same path and speed. In addition to a few highlighted voxels at the beginning of the game, the blue team's voxels also show some brightness towards the middle of the game. Green voxels, representing the simultaneous presence of both players and enemies within the area, are visible at the intersection of red and blue voxels. Visually, these regions appear to correspond to areas where players employ identical evasive strategies after approaching enemies. We can also notice that some bright blue voxels are far away from the enemy. These are likely players who are wandering, waiting, or even pausing the game. On some platforms where players might fall or need to avoid obstacles, we can observe along the z-axis that the intensity of the blue or green voxels gradually decreases. This is likely because the players who are still playing the game have found a relatively fixed way to avoid the enemies, making the voxels more opaque than at the beginning of the game. Considering that this is a relatively simple platform game, this performance is understandable.

The visual result also shows that most of the enemies' voxels are translucent and similar. This also verifies our idea of the anomaly detection pipeline, as enemies that always repeat the same action patterns may be less interesting to game analysts in most cases.

In summary of these two case studies, the voxel velocity visualization is uniformly transparent, making it difficult to discern differences. The sliding window highlights some end paths, but most intermediate values still appear similarly colored. The isolation forest balances local prominence with global data variation, but its results are more sensitive to visualization parameter choices.

5. Conclusion

This paper proposes a general voxelization visualization pipeline for transforming game playtrace trajectory data from 2D into 3D voxel volumes for visual analysis, and using anomaly detection to emphasize voxels of interest. We use the same steps across two distinct games; spatio-temporal information is aggregated into a regular grid using a voxelization parameter; then an isolation forest is applied to the normalized 3D feature vectors for unsupervised anomaly detection. All results are written to a VTR file for overlay display in ParaView using color and transparency mapping. Overall, this pipeline strikes a good balance between consistency across games and interpretable results, while maintaining simplicity in implementation and flexibility in expansion. Our voxel-based anomaly visualization provides a visual 3D approach to analyze complex intertwined game paths that is envisioned to potentially be useful for tasks such as level design diagnosis, player behavior analysis, and detecting unusual strategies.

We still face the limitations that results can be affected by the discretization and parameter selection, which can lead to unstable rarity and outlier scores in sparse regions; alignment of map textures and voxel coordinates currently requires minimal manual fine-tuning; the isolation forest hyperparameters are fixed for both datasets, and the feature dimensionality is currently low. However, being able to adjust the discretization is not only a limitation but also a level-of-detail parameter. A rougher discretization could reveal high-level issues, while a smaller one would enable a focus on more fine-grained movement. Future plans include: introducing automated map registration and cross-map standardization; incorporating more unsupervised methods to enhance adaptability to a broader range of games to promote the reproduction and reuse of this system. We would also like to perform a user study to better understand the usefulness and ease-of use of these visualizations for game designers.

Declaration on Generative AI

During the preparation of this work, the author(s) used Grammarly in order to: Grammar and spelling check, Paraphrase and reword, Improve writing style; and ChatGPT in order to: Citation management. After using these tools/services, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] G. Andrienko, N. Andrienko, P. Bak, D. Keim, S. Wrobel, Visual analytics of movement, Springer, 2013.

[2] Y.-T. Kuan, Y.-S. Wang, J.-H. Chuang, Visualizing real-time strategy games: The example of StarCraft II, in: 2017 IEEE Conference on Visual Analytics Science and Technology (VAST), 2017, pp. 71–80. doi:10.1109/VAST.2017.8585594.

[3] E. Jun, S. Landry, G. Salvendy, Exploring the cognitive costs and benefits of using multiple-view visualisations, Behaviour & Information Technology 32 (2013) 824–835. doi:10.1080/0144929X.2011.630420.

[4] J. A. W. Filho, W. Stuerzlinger, L. Nedel, Evaluating an immersive space-time cube geovisualization for intuitive trajectory data exploration, IEEE Transactions on Visualization and Computer Graphics 26 (2020) 514–524. doi:10.1109/TVCG.2019.2934415.

[5] A. Šufliarsky, G. Walllner, S. Kriglstein, Through space and time: Spatio-temporal visualization of MOBA matches, in: J. Abdelnour Nocera, M. Kristín Lárusdóttir, H. Petrie, A. Piccinno, M. Winckler (Eds.), Human-Computer Interaction – INTERACT 2023, Springer, Cha, Switzerland, 2023, pp. 167–189.

[6] Blizzard Entertainment, StarCraft: Brood War, Game [PC], 1998.

[7] G. Wallner, A. Drachen, Visualization of player movement patterns with line integral convolution and alpha shapes, in: Proceedings of the 19th International Conference on the Foundations of

Digital Games, FDG '24, Association for Computing Machinery, New York, NY, USA, 2024. URL: https://doi.org/10.1145/3649921.3649997. doi:10.1145/3649921.3649997.

[8] G. Wallner, N. Halabi, P. Mirza-Babaei, Aggregated visualization of playtesting data, in: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, CHI '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 1–12. doi:10.1145/3290605.3300593.

[9] F. T. Liu, K. M. Ting, Z.-H. Zhou, Isolation forest, 2008 Eighth IEEE International Conference on Data Mining (2008) 413–422. URL: https://api.semanticscholar.org/CorpusID:6505449.

[10] J. Dankoff, Game telemetry with DNA tracking on Assassin's Creed, 2014. https://www.gamedeveloper.com/design/game-telemetry-with-dna-tracking-on-assassin-s-creed Accessed: August, 2025.

[11] M. S. El-Nasr, A. Drachen, A. Canossa, Game analytics, Springer, 2016.

[12] A. Drachen, M. Schubert, Spatial game analytics and visualization, in: 2013 IEEE Conference on Computational Inteligence in Games (CIG), 2013, pp. 1–8. URL: https://ieeexplore.ieee.org/document/6633629. doi:10.1109/CIG.2013.6633629, iSSN: 2325-4289.

[13] N. Andrienko, G. Andrienko, P. Gatalsky, Exploratory spatio-temporal visualization: an analytical review, Journal of Visual Languages & Computing 14 (2003) 503–541. URL: https://www.sciencedirect.com/science/article/pii/S1045926X03000466. doi:10.1016/S1045-926X(03)00046-6.

[14] A. Drachen, A. Canossa, Analyzing spatial user behavior in computer games using geographic information systems, in: Proceedings of the 13th International MindTrek Conference: Everyday Life in the Ubiquitous Era, MindTrek '09, Association for Computing Machinery, New York, NY, USA, 2009, pp. 182–189. doi:10.1145/1621841.1621875.

[15] J. L. Miller, J. Crowcroft, Avatar movement in World of Warcraft battlegrounds, in: 2009 8th Annual Workshop on Network and Systems Support for Games (NetGames), 2009, pp. 1–6. doi:10.1109/NETGAMES.2009.5446226.

[16] D. Moura, M. S. el Nasr, C. D. Shaw, Visualizing and understanding players' behavior in video games: discovering patterns and supporting aggregation and comparison, in: ACM SIGGRAPH 2011 Game Papers, SIGGRAPH '11, Association for Computing Machinery, New York, NY, USA, 2011. doi:10.1145/2037692.2037695.

[17] A. Kaufman, K. Mueller, Overview of volume rendering., in: C. D. Hansen, C. R. Johnson (Eds.), The Visualization Handbook, Elsevier, Burlington, USA, 2005, pp. 127–174.

[18] Unity Technologies, Unity Analytics overview, 2021. URL: https://docs.unity3d.com/2020.1/Documentation/Manual/UnityAnalyticsOverview.html.

[19] I. Viola, A. Kanitsar, M. Groller, Importance-driven volume rendering, in: IEEE Visualization 2004, 2004, pp. 139–145. doi:10.1109/VISUAL.2004.48.

[20] L. Greige, F. D. M. Silva, M. Trotter, C. Lawrence, P. Chin, D. Varadarajan, Collusion detection in team-based multiplayer games, 2022. URL: https://arxiv.org/abs/2203.05121. arXiv:2203.05121.

[21] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, T. E. Oliphant, Array programming with NumPy, Nature 585 (2020) 357–362. URL: https://doi.org/10.1038/s41586-020-2649-2. doi:10.1038/s41586-020-2649-2.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[23] W. Schroeder, K. Martin, B. Lorensen, The Visualization Toolkit (4th ed.), Kitware, 2006.

[24] J. Ahrens, B. Geveci, C. Law, Visualization Handbook, Elsevier Inc., Burlington, MA, USA, 2005, pp. 717–731. URL: https://www.sciencedirect.com/book/9780123875822/visualization-handbook.

[25] Z. Lin, J. Gehring, V. Khalidov, G. Synnaeve, Stardata: A starcraft ai research dataset, 2017. URL: https://arxiv.org/abs/1708.02139. arXiv:1708.02139.

[26] G. Synnaeve, N. Nardelli, A. Auvolat, S. Chintala, T. Lacroix, Z. Lin, F. Richoux, N. Usunier,

Torchcraft: a library for machine learning research on real-time strategy games, arXiv preprint arXiv:1611.00625 (2016).

[27] C. F. Biemer, Recformer, 2025. URL: https://github.com/bi3mer/recformer.