Integrated Spatio-temporal Storyline Visualization with Low Crossover

1st Shilpa Batthineni
Masters in Computer Science
Georgia State University
Atlanta, United States
sbatthineni1@student.gsu.edu

2nd Ying Zhu
Creative Media Industries Institute
Georgia State University
Atlanta, United States
yzhu@gsu.edu

***Abstract*—The abstract details the significance of achieving a balance in integrating temporal and spatial dimensions for effective storyline visualization. Storylines represent a chain of events, these events are complex, with multiple dimensions such as time, location, characters, actions, and context.**
The challenge lies in integrating temporal (time-related) and spatial (location-related) dimensions within a unified visualization, given the inherent differences between 2D spatial data and 1D temporal data. The proposed technique involves the development of a timeline-based visualization method that preserves the orderly arrangement of both temporal and spatial dimensions. Additionally, a heuristic method has been devised to mitigate the issue of excessive line crossovers, particularly when dealing with substantial datasets.

***Index Terms*—storyline visualization, spatio-temporal visualization, and story graph, minimum line crossover.**

I. Introduction

In the realm of Storyline visualization, the integration of spatial and temporal dimensions poses a compelling challenge. Stories, comprising a tapestry of events, demand a sophisticated approach to convey the intricacies of both time and location seamlessly. This paper delves into the domain of Integrated Spatio-temporal Storyline Visualization, a novel endeavor that strives not only to capture the essence of diverse dimensions within a narrative but also to mitigate the intricate issue of line crossovers.

A. Context of Multidimensional Storyline:

- Stories unfold as multifaceted data entities, each event encapsulating dimensions such as characters, time, location, and actions.
- The endeavor of storyline visualizations is to artistically present the sequential events and the intricate relationships woven within these dimensions.

B. Desire for Integration:

- In storytelling, the convergence of time and location in a single visual domain is inherently desirable, allowing users to swiftly discern correlations between temporal and spatial information.
- The paper sets its focus on the formidable challenge of integrated spatial-temporal storyline visualization, acknowledging the complexities inherent in reconciling the 2D nature of spatial data with the 1D nature of temporal data.

C. Categorization of Existing Approaches:

- The literature review underscores existing spatial-temporal storyline visualizations, classified into map-based, timeline-based, unconventional, and hybrid approaches.

D. Proposed Solution: Integrated Spatio-temporal Storyline Visualization with Low Crossover:

- We introduce an innovative hybrid approach, a synergy of refined timeline-based visualization and a synchronized map, poised to address the persistent issue of excessive line crossovers.
- The refinement extends beyond the timeline-based method, incorporating a novel heuristic algorithm expressly designed to reduce line crossovers and enhance the overall visual coherence.

E. Objective and Significance:

- The primary objective is to demonstrate the practicality and effectiveness of the proposed integrated approach in offering a comprehensive visualization of multidimensional stories.
- We aims to validate the method's prowess in not only maintaining regular ordering for temporal and spatial dimensions but also in presenting a visually cohesive narrative with minimal line crossovers.

II. Storygraph: An innovative spatio-temporal visualization

The Storygraph, a visualization paradigm meticulously developed for spatio-temporal data within the context of a storytelling engine, represents a significant leap in narrative visualization. This incorporates two parallel vertical axes, akin to those found in parallel coordinates, and a horizontal axis. The vertical axes serve as a representation of location pairs, such as latitude and longitude, while the horizontal axis signifies time. The fundamental unit in the Storygraph is the location line, a line segment connecting location values on the vertical axes. Events occurring at the same location but at

different times are depicted as markers along these location lines. An illustrative example of a Storygraph is presented in Fig. 1, where circles on the line symbolize events transpiring at the same geographical point but at distinct temporal instances.

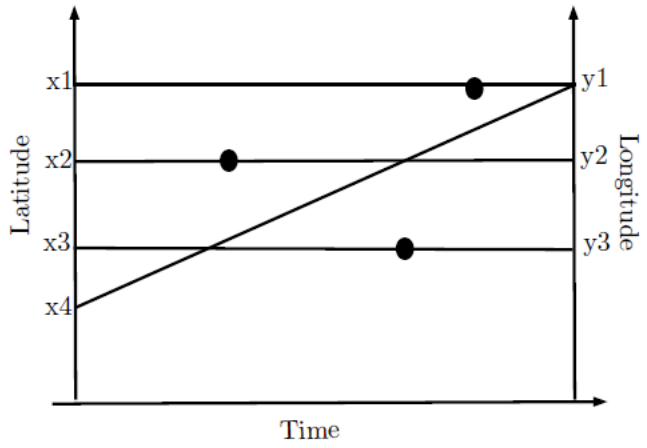

Fig. 1. Storygraph Visualization

A. Coordinate System:

- The core concept driving the Storygraph is the visualization of spatio-temporal or 3D data within a 2D coordinate system.
- Each location, denoted by coordinates (x, y) or (latitude, longitude), is transformed into a distinctive location line within the paradigm.

B. Addressing Visual Clutter Challenges:

- A primary challenge in constructing Storygraphs and other data visualization frameworks is the potential visual clutter arising from plotting large datasets within a confined display area.
- This visual clutter can impede the efficacy of data visualization, hindering the seamless transmission of information.

C. Subproblems of Visual Clutter:

- Point Clutter Problem: Arises when numerous data points converge, hindering the clarity of information.
- Line Crossing Problem: Occurs when multiple location lines intersect, impeding the interpretability of the visualization.

D. Solutions to Visual Clutter:

- Point Clutter Solution: Strategies and techniques are employed to alleviate point clutter, ensuring that individual data points remain distinguishable and do not compromise clarity.
- Line Crossing Solution: The challenge of line crossings is systematically addressed, introducing measures to minimize the intersections of lines and enhance the overall visual coherence of the Storygraph.

In the subsequent sections, we delve into a detailed exploration of these subproblems and their corresponding solutions, shedding light on how the Storygraph effectively manages the intricate balance between information density and visual clarity in the representation of spatio-temporal data.

III. Implementation

This Python code utilizes the Matplotlib and Pandas libraries to create a spatial-temporal visualization based on a given CSV file. Let's break down the code step by step:

- The code calculates the intersection points between lines formed by connecting (0, Latitude*2) and (len(df), Longitude) for each row in the DataFrame.
- It plots lines connecting (0, Latitude*2) and (len(df), Longitude) for each row in light blue color.
- It then sets the ticks and labels for the x and y axes
- It also creates a twin axes sharing the x-axis but with a different y-axis for Longitudes.
- It displays the final spatial-temporal visualization.

```
intersection_points_x=[]
intersection_points_y=[]
for index, row in df.iterrows():
    x1,y1 = 0,row['Latitude']*2
    x2,y2 = len(df),row['Longitude']
    m = (y2 - y1) / (x2 - x1)
    c = y1 - m * x1
    x_min, x_max = plt.xlim()
    y_min = m * x_min + c
    y_max = m * x_max + c
    y = m*index+c
    intersection_points_x.append(index)
    intersection_points_y.append(y)
num_points = len(df)
max_marker_size = 25  # Maximum marker size
min_marker_size = 5  # Minimum marker size
scaled_marker_size = max(min_marker_size, max_marker_size / num_points)
plt.scatter(intersection_points_x, intersection_points_y, color='black',
            label='Intersection', zorder=2, s=scaled_marker_size)

for index, row in df.iterrows():
    plt.plot([0, len(df)], [row['Latitude']*2, row['Longitude']], color='lightblue', zorder=1)

# ax.set_xticks([0,1,150])
ax.set_yticks([-180,-160,-140,-120,-100,-80,-60,-40,-20,0,20,40,60,80,100,120,140,160,180])
ax.set_yticklabels([-90,-80,-70,-60,-50,-40,-30,-20,-10,0,10,20,30,40,50,60,70,80,90])
ax.set_xticks([i for i in range(len(df))])
a=[j['Date'] for i,j in df.iterrows()]
ax.set_xticklabels(a)
ax.set_ylabel('Latitudes')
ax.tick_params(axis='x', labelrotation=90)

ax2 = ax.twinx()
# Set ticks and tick labels for the right y-axis
ax2.set_ylabel('Longitudes')
ax2.tick_params(axis='y', colors='red')
ax2.set_yticks([-180,-160,-140,-120,-100,-80,-60,-40,-20,0,20,40,60,80,100,120,140,160,180])
ax.xaxis.grid(False)
ax.yaxis.grid(False)
plt.show()
```

Fig. 2. Code of Storygraph Implementation

In summary, the code reads a CSV file containing spatial-temporal data, calculates intersection points, and visualizes the data using Matplotlib, representing spatial and temporal information in the same plot. The x-axis represents time (dates), and the y-axis represents both latitude and longitude with different scales and colors.

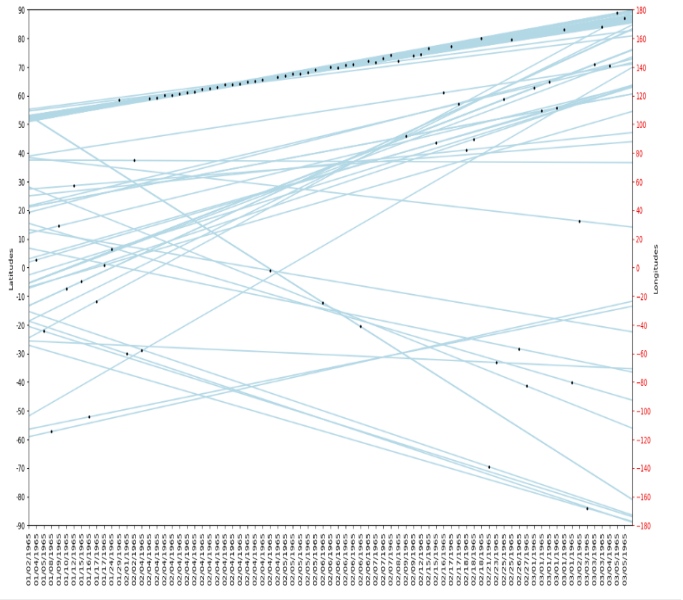

Fig. 3. Storygraph Implementation

IV. Problems of Visual Clutter

A. Point Clutter

The point clutter problem in data visualization arises when there is an overwhelming concentration of data points within the display area, leading to a visual overload that can impede the viewer's ability to discern individual data points and patterns. This phenomenon is particularly pronounced in large datasets where numerous points are plotted in close proximity, resulting in a cluttered and chaotic visual representation.

- As the number of data points increases, the challenge lies in maintaining the distinguishability of each point.
- Point clutter makes it difficult for the viewer to identify and differentiate individual data points, diminishing the interpretability of the visualization.
- Excessive point clutter can lead to information loss, where the nuances and details within the dataset become obscured by the sheer density of plotted points.
- This hinders the effectiveness of the visualization in conveying meaningful insights.

B. Line Crossing

The Line Crossing Problem in data visualization occurs when lines representing different data elements intersect with each other. This can lead to visual confusion and hinder the interpretability of the visualization. The complexity intensifies as the number of intersecting lines increases, making it challenging for viewers to discern the individual paths and relationships within the dataset. Addressing the Line Crossing Problem is crucial for enhancing the overall clarity and coherence of the visual representation.

1) Visual Ambiguity:
   - Intersecting lines create visual ambiguity, making it difficult for viewers to trace the trajectories of individual data elements accurately.
   - The overlapping lines can obscure the underlying patterns and relationships present in the data. Complexity with Increasing Lines:
   - As the number of lines grows, the complexity of the visualization intensifies. Line crossings become more frequent, exacerbating the challenge of untangling and understanding the interconnected paths.
2) Impaired Data Interpretation: The Line Crossing Problem can impair the interpretation of temporal or spatial relationships between data points. Viewers may struggle to distinguish between different events or trends represented by intersecting lines.

V. Solution to Visual Clutter

A. Solution to Point Clutter

The algorithm involves addressing the point clutter problem in a data visualization by adjusting the marker size based on the number of data points at each intersection of lines. The goal is to strike a balance between making the plot visually appealing, ensuring markers are noticeable, and preventing clutter and overlapping in the presence of a large number of data points. It leverages marker size adjustments as a dynamic mechanism to optimize visibility and prevent clutter, catering to both the aesthetic and informative aspects of the plotted data points.

1) Plotting Black Points at Intersections: The solution involves representing the intersections of lines by plotting black points. These points signify locations where lines intersect, and they serve as markers for specific data points in the visualization.
2) Dynamic Marker Size Adjustment:
   - The novelty in your approach lies in dynamically adjusting the size of these markers based on the number of data points at each intersection. This adjustment is crucial for enhancing the visual appeal and informativeness of the plot. Preventing Marker Extremes:
   - The purpose of adjusting the marker size is to avoid extremes—markers that are too large or too small. This adjustment is intended to optimize the visual representation of the data. Enhancing Noticeability with Larger Markers:
   - When there are a small number of data points at an intersection, your solution proposes increasing the marker size. Larger markers are more noticeable, drawing attention to specific intersections and making them stand out in the plot.
3) Scaling Down for Clutter Avoidance:
   Conversely, when there are many data points at an intersection, the solution suggests scaling down the marker size. This is a preventive measure to avoid clutter

and overlapping, ensuring that the plot remains visually coherent and interpretable.

4) Visual Appeal and Informativeness: The overarching goal of the solution is to strike a balance between visual appeal and informativeness. By dynamically adjusting marker sizes based on data density, the plot aims to present a visually pleasing representation of the intersections while conveying meaningful insights from the data.
5) Adaptability to Data Characteristics:
The approach acknowledges the variability in the number of data points across different intersections. By adapting the marker size to the local data context, the solution accounts for the inherent heterogeneity in the dataset.

B. Solution to Line Crossing

The visualization aims to provide insight into areas on the map where rearranging latitude and longitude values could potentially reduce line crossings. By focusing on the segment with the maximum adjacent difference, the script highlights regions where lines are more separated, and the scatter plot of intersection points helps identify key points within that segment. Let's go through the steps and how they contribute to achieving this goal:

1) Line Intersection Points:
   - The algorithm starts by calculating intersection points for all pairs of lines formed by latitude and longitude values.
   - It identifies the x-coordinates of these intersection points within the range of the DataFrame.
2) Maximum Adjacent Difference:
   - It then finds the maximum adjacent difference among these x-coordinates, indicating a segment on the x-axis where lines have the most significant separation.
3) Visualization of Identified Segment:
   - The algorithm sets the x-axis limits to focus on this segment with the maximum adjacent difference.
   - It adjusts latitude and longitude values based on this segment and plots lines on the graph.
4) Scatter Plot of Intersection Points:
   - Intersection points within the identified segment are plotted as black dots on the graph.
5) Customization of Axes:
   - The y-axes are labeled for both latitudes and longitudes.
   - A twin axis is created for longitudes on the right side, with different tick colors.

The algorithm aims to visualize geographical paths and intersection points, with a focus on regions where rearranging latitude and longitude values may reduce line crossings. The segment with the maximum adjacent difference is highlighted for detailed analysis. Overall, it provides a visual representation of potential areas for optimization in geographical data visualization.

Reducing line crossings in geographical visualizations can be valuable for improving clarity and interpretability, especially when dealing with complex datasets or routes. The script attempts to achieve this by identifying and visualizing areas of interest on the map where line crossings can potentially be minimized.

VI. Storyline Visualization with Minimum Line Crossover

Our Heuristic algorithm aims to reduce line crossings in the visualization by identifying a segment with the maximum adjacent difference and focusing on that segment.

```
intersection_points.sort()
max_diff, positions = max_adjacent_difference(intersection_points)
fig, ax = plt.subplots(figsize=(20, 10))
diff = (intersection_points[positions[1]] - intersection_points[positions[0]])/len(df)
plt.xlim(intersection_points[positions[0]]-2*diff, intersection_points[positions[1]]+diff)
plt.ylim(-180, 180)
indexinit =  intersection_points[positions[0]] -diff
intersection_points_x=[]
intersection_points_y=[]
for index, row in df.iterrows():
    x1,y1 = 0,row['Latitude']*2
    x2,y2 = len(df),row['Longitude']
    m = (y2 - y1) / (x2 - x1)
    c = y1 - m * x1
    x_min, x_max = plt.xlim()
    y_min = m * x_min + c
    y_max = m * x_max + c
    y = m*indexinit + c
    plt.plot([x_min, x_max], [y_min, y_max], color='lightblue', zorder=1)
    intersection_points_x.append(indexinit)
    intersection_points_y.append(y)
    indexinit=indexinit+diff
max_marker_size = 25  # Maximum marker size
min_marker_size = 5  # Minimum marker size
scaled_marker_size = max(min_marker_size, max_marker_size / len(df))
plt.scatter(intersection_points_x, intersection_points_y, color='black',
            label='Intersection', zorder=2, s=scaled_marker_size)
ax.set_yticks([-180,-160,-140,-120,-100,-80,-60,-40,-20,0,20,40,60,80,100,120,140,160,180])
ax.set_yticklabels([-90,-80,-70,-60,-50,-40,-30,-20,-10,0,10,20,30,40,50,60,70,80,90])
ax.set_ylabel('Latitudes')
a=[j['Date'] for i,j in df.iterrows()]
ax.set_xticks(intersection_points_x)
ax.set_xticklabels(a)
ax.tick_params(axis='x', labelrotation=90)
ax2 = ax.twinx()

# Set ticks and tick labels for the right y-axis
ax2.set_ylabel('Longitudes')
ax2.tick_params(axis='y', colors='red')
```

Fig. 4. Code of Storygraph Implementation with minimum line crossovers

- Here, the code calculates intersection points for all pairs of lines formed by latitude and longitude values.
- It filters and stores only the intersection points within the valid x-axis range of the DataFrame.
- The intersection points are sorted, and the maximum adjacent difference is found.
- Then the x-axis limits are set to focus specifically on the segment with the maximum adjacent difference.
- The range is extended by a factor of 2 * diff on the left side and by diff on the right side for better visualization.

- The latitude and longitude values are adjusted based on the identified segment with the maximum adjacent difference.
- Finally, lines are plotted on the graph using the adjusted latitude and longitude values.

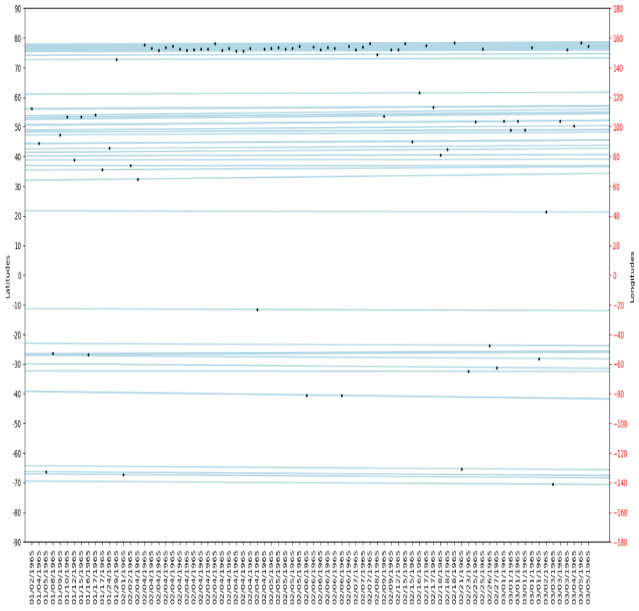

Fig. 5. Storygraph Implementation with minimum line crossovers

By focusing on the segment with the maximum adjacent difference and adjusting latitude and longitude values accordingly, the code aims to visually highlight areas on the map where rearranging latitude and longitude values may reduce line crossings. The lines are spaced based on this identified segment, providing a clearer representation of geographical paths with minimized overlaps.

VII. Dataset Details

A. Significant Earthquakes, 1965-2016

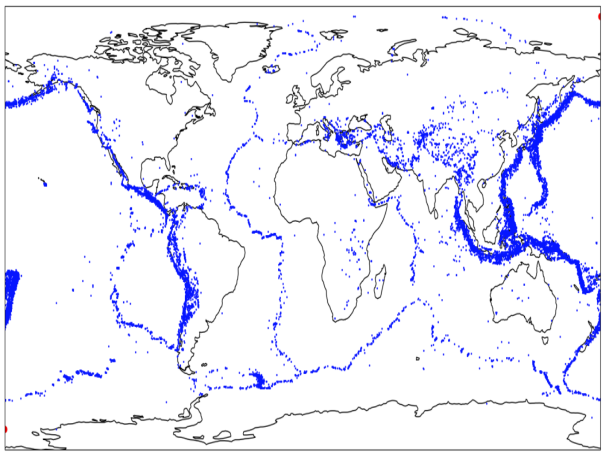

Fig. 6. Interactive Map with Plotted Earthquake Data

This dataset includes a record of the date, time, location, depth, magnitude, and source of every earthquake with a reported magnitude 5.5 or higher since 1965.

Fig. 6 shows an interactive map with each war log plotted on the map as a point. Multiple events may happen at the same location

Conclusion

Our Heuristic approach provides a visual representation of spatio-temporal storylines, with a focus on regions where rearranging latitude and longitude values may reduce line crossings. The visualization strategy aims to enhance the integrated understanding of geographical paths and their temporal aspects, contributing to a more effective and interpretable representation of the spatio-temporal data. Additionally this method does not require time-consuming data preparation and is relatively fast. It works well with a regular spreadsheet. The only requirement is that the spreadsheet has time and location columns.

References

[1] Xi He, Ying Zhu, “Integrated Spatio-temporal Storyline Visualization with Low Crossover.

[2] E. Segel and J. Heer, “Narrative visualization: Telling stories with data, IEEE Transactions on Visualization and Computer Graphics, vol. 16, no. 6, pp. 1139–1148, 2010.

[3] J. Hullman and N. Diakopoulos, “Visualization rhetoric: Framing effects in narrative visualization, IEEE Transactions on Visualization and Computer Graphics, vol. 17, no. 12, pp. 2231–2240, 2011..

[4] J. Zhao, F. Chevalier, C. Collins, and R. Balakrishnan,“Facilitating discourse analysis with interactive visualization, IEEE Transactions on Visualization and Computer Graphics, vol. 18, no. 12, pp. 2639–2648, 2012.

[5] M. Ogawa and K.-L. Ma, “Software evolution storylines in Proceedings of the 5th international symposium on Software visualization. ACM, 2010, pp. 35–42.

[6] N. W. Kim, S. K. Card, and J. Heer, “racing genealogical data with timenets,” in Proceedings of the International Conference on Advanced Visual Interfaces. ACM, 2010, pp. 241–248..

[7] P. H. Nguyen, K. Xu, R. Walker, and B. Wong, “Schemaline: Timeline visualization for sensemaking,” in 18th International Conference on Information Visualisation (IV). IEEE, 2014, pp. 225–233.