Project Proposal - Linked Multi-View Multivariate Trajectory Visualization

Devin Lange
devinscottlange@gmail.com
UID: u1265145

Repo: https://github.com/Dev-Lan/Dev-Lan.github.io/tree/master/research/MTAT

Background and Motivation

Motion visualization is relevant to many different scientific domains. Robotics, fluid dynamics, traffic flow, biomechanics are a few examples. I have worked in the past both generating motion data to simulate the movement of crowds of pedestrians and have created tools to analyze that motion as well as other trajectory data. Multivariate trajectory data is an extension of trajectory data where other data attributes are included at each point for your path. An example of a possible derived attribute would be speed.

Project Objectives

I will create a tool for analyzing multivariate trajectory data. I will develop it to work with different datasets, but I will provide a few example datasets. The potential benefit would a tool that could be applied across different domains.

Data

The application will support the loading of CSV files where each row of the file is a single point in a trajectory. The headers of the CSV will be assumed to include at least the following, **t**, **x**, **y**, **id**. In addition, any number of other values can be included for each point. I also have measured trajectories of groups of people moving in different scenarios. In addition, these trajectories include information as to how well different simulation methods can reproduce characteristics of the real data.

Data Processing

Minimal data processing should be required. However, writing code that is complete robust to different datasets can be a time sink. I will draw the line for this project that well formatted data should work. In an ideal world poorly formatted data would return helpful errors so I may have some amount of this, but it will not be the best it possibly could for the purposes of this class.

Visualization Design

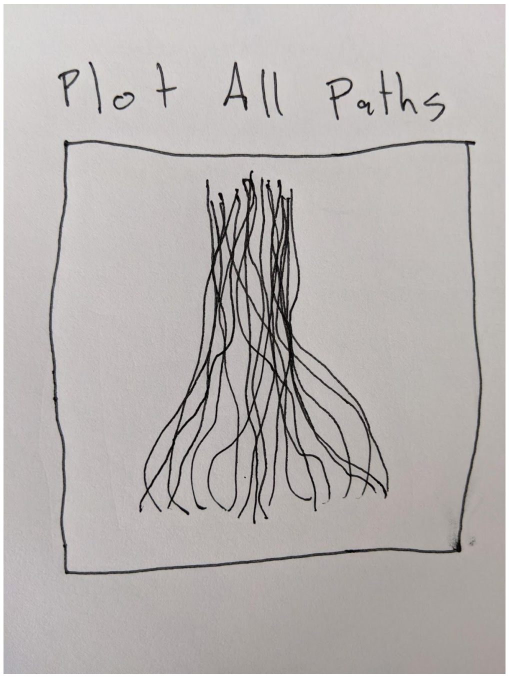
I will create an application that utilizes multiple different views of multivariate trajectory data. The initial thrust of the project will lay the technical framework for adding different views to a single layout as well as an initial number of widgets. The framework will aim to be generalizable to any multivariate trajectory data set to start. To demonstrate the framework I will programmatically create a layout with different widgets and demonstrate it with a few datasets that I already have.

Widget Sketches

I have included initial sketches of all the sketches before laying out the project requirements at the end since I reference some of these widgets.

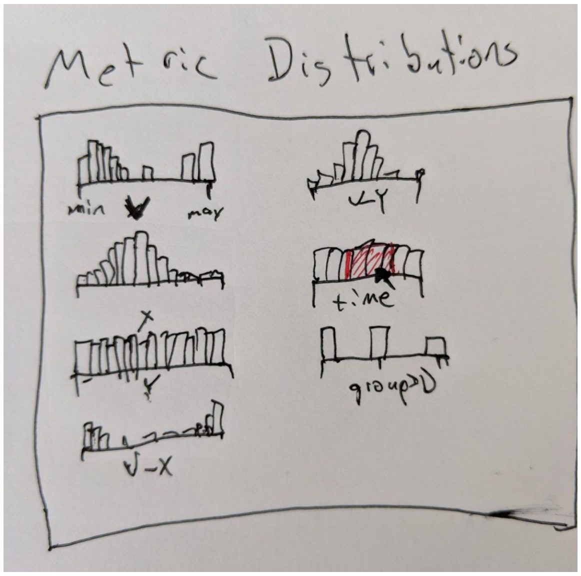
Plot of all Paths

One of the simplest ways to visualize trajectories is to simply draw the paths traced by all trajectories in your data set. While this approach suffers from occlusion, or what I like to call the ball of spaghetti problem, it is a good starting point. In this sketch, I've drawn the example of people walking into a more narrow hallway.



Metric Distributions

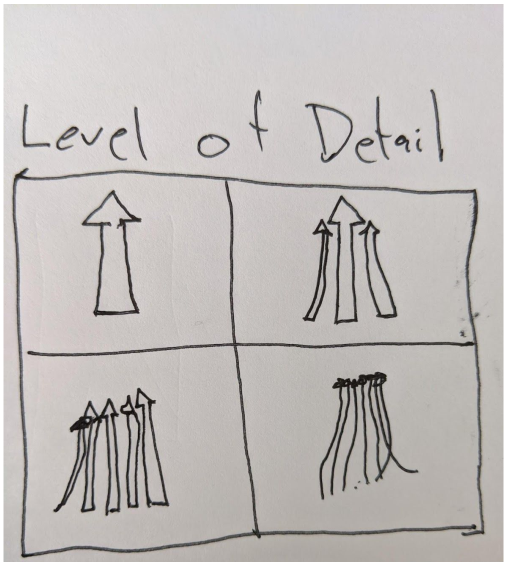
For every data set, you will have at least an id, x, y, t variable for every point. In addition, your dataset could have more information like speed, acceleration, or more abstract concepts like how closely a simulation technique may match measured data. This visualization gives a quick overview of the distribution of these metrics. In addition, this will support brushing, so you can brush on any of the metrics. In the sketch, time is brushed. If the data exist, this widget could also be extended for metrics on each trajectory, or for groups of trajectories,



Level of Detail

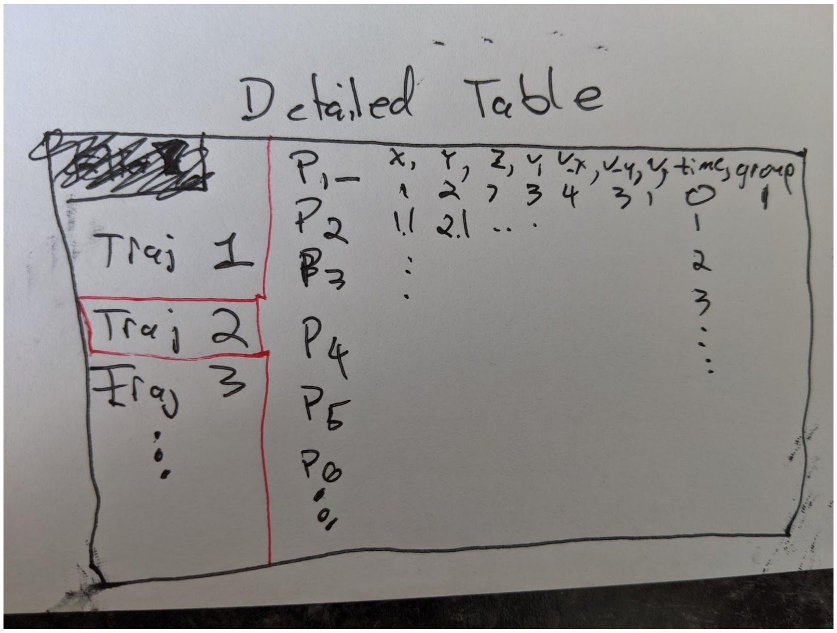
Using k-means clustering a number of plots with different levels of detail can be constructed. A similar approach has been done by D. Sacha et al[1]. In this widget, selecting a trajectory in a lesser level of detail view will result in brushing the group of trajectories that are in the selected trajectory.

* Another possibility is to use the hierarchical structure approach we discussed in class.



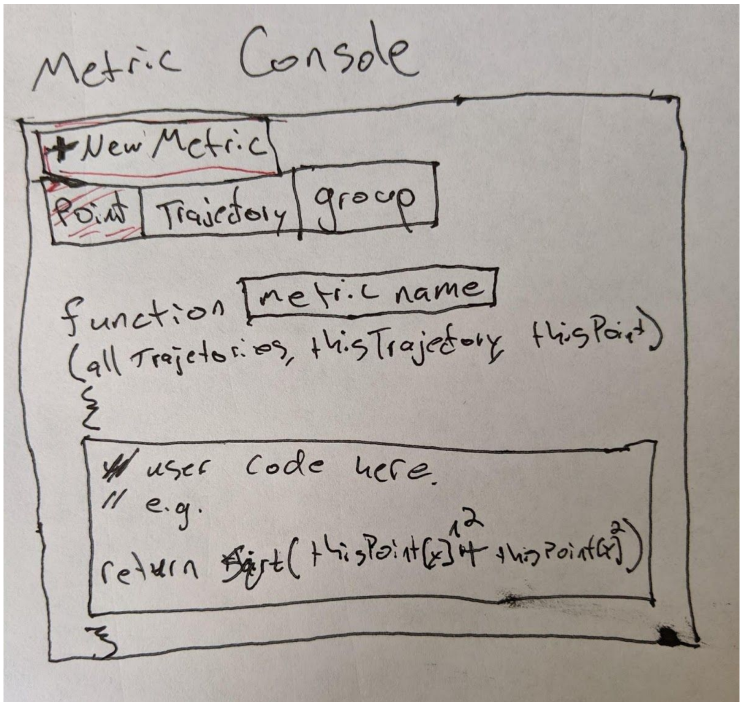
Detailed Table

I think sometimes data visualization designers underestimate how much scientists just love tables of numbers. This sketch shows one possible layout where trajectories are listed, and selecting a trajectory gives a full list of its points. In addition, this selection of trajectories and points could be linked to the other widgets. In addition to its merit as a visualization tool, I suspect this would be useful as a debugging aid while developing new widgets.



Metric Console

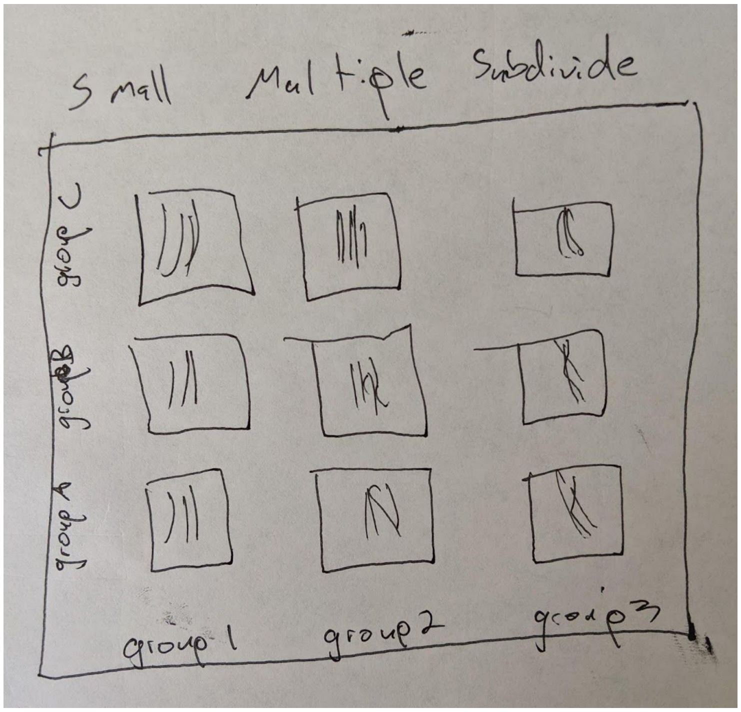
While this framework is already quite robust to different ways of looking at data, it is limited to the data that is initially loaded. Let’s say that while investigating your dataset you have the thought for example that looking at the distance of each point from the origin may help in the understanding of the data. Now, you would have to leave the visualization tool, write code to calculate this new value, run it, then upload the new dataset to the tool, then redo any filters or modifications in the tool. Instead, I would like to provide the ability to define new “metrics” on the fly. In this sketch, you can add a new metric for points, metrics, or groups. In the example shown, the user could define the bodies of new functions in Javascript.



```
function metric name
(allTrajectories, thisTrajectory thisPoint)
{
  // user code here.
  // e.g.
  return Sqrt( thisPoint[x]^2 + thisPoint[y]^2)
}
```

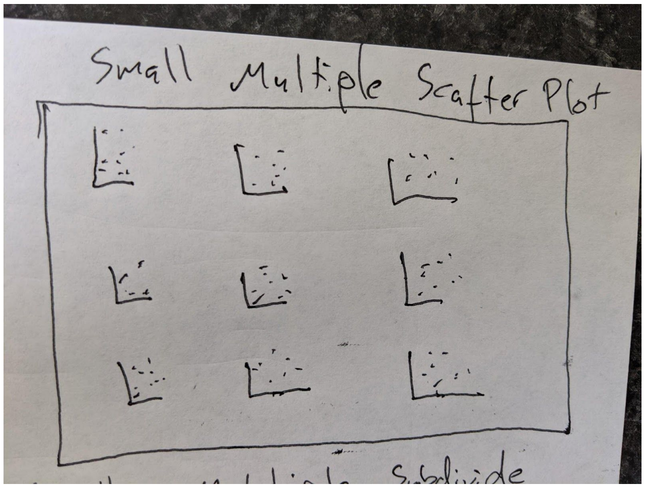
Small Multiples Subdivision

Another idea for a widget would be to support splitting the plot of all paths into different groups. This could aid in a detailed comparison of different groups, or even splitting randomly to help reduce overlap of trajectories.



Small Multiples scatter plot

While similar to the metric distributions widget, this would aid in the analysis of the interaction of multiple metrics. In addition, it would provide a way to brush along two dimensions with one action. Instead of just including a scatterplot matrix I will include a matrix widget that allows selection of different combinations.



Must Have Features

- The base programming framework must be laid so that programmatically adding new widgets in the future is easy.
- 2 linked widgets must be completed
    - A plot of paths traced by all trajectories
    - Metric distributions of points with brushing capability

Optional Features

Stretch Goal - Tier 1 - My goal is to finish these, but will cut if they cannot be finished in time.

- The plot of all paths will also support animation of Trajectories
- 2 additional widgets
    - Level of detail plot with brushing capability
    - Detailed table

Stretch Goals - Tier 2 - I might get started on these, but likely will not get far into this list before project submission.

- Metric console
- Small multiple path subdivision
- Small multiple scatterplots
- Support metrics on trajectories and groups
- Add some number of prebuilt metrics (e.g. speed, acceleration, curve depth)

Future Work - definitely will not get to for this class, but ways this could be extended

- Add a user interface so different combinations of widgets can be made without any programming necessary
  - Should be able to save and share view configurations
- Add support for widget plugins so that many different developers/teams can create their own widgets and others can combine them into a single view.
- Add support so metric functions can be shared easily.
- Level of detail plots can also display aggregates of other metrics.

Project Schedule

Intermediate due dates (meeting these would complete the Stretch Goal - Tier 1 list):

- Nov 1st - Data loader / structure complete
- Nov 8th - Base Layout Framework implemented. 1st widget started, and can be displayed with the layout framework..
- Nov 15th - Must have features complete
- Nov 22nd - Animation added to plot of all plots, Level of detail Plot complete
- Nov 27th - Detailed table widget done

References:

1. Sacha, D. , Al‐Masoudi, F. , Stein, M. , Schreck, T. , Keim, D. A., Andrienko, G. and Janetzko, H. (2017), Dynamic Visual Abstraction of Soccer Movement. Computer Graphics Forum, 36: 305-315. doi:10.1111/cgf.13189