Outlier-preserving Focus and Context Visualization for Parallel Coordinates

Vincent Yang, Bradley Wang *University of California, Davis*

1 Goal and Approach

Our goal in this project was to create a parallel coordinates system that would make outliers and primary trends extremely obvious for the viewer. The system should scale well with large datasets and be an effective tool for finding interesting relationships or outliers. Our approach is to implement the same ideas as from the paper. To scale with large datasets, we would simplify large datasets down to a normalized set of trends and outliers, then display these trends/outliers instead. Clustering would enable the user to find unique/interesting trends via. interaction, although this is a supplementary feature.

2 Our Experience

2.1 Positives

This paper was interesting because it has a new approach to parallel coordinates that we haven't seen before. This was also our first time implementing a visualization paper, which gave us insight into how difficult it is to create these systems. Far more effort went into figuring out how to use which tools, which I didn't expect. Next, the paper's approach and pictures also gave us a strong idea on the benefits of their method. I learned alot on how data may be processed before displaying to the user. We also learned about how people in visualization have to think about the reality of their visualization methods – that computing might take a long time or the screen might not be large enough.

2.2 Negatives

We found the paper a little bit difficult to make sense of. The title of the paper implied that it would focus on outlier and focus, but it didnt distinguish the difference between outlier and focus particularly well. This made working on the implementation rather difficult, as we had to read significantly more background literature than previously expected. We also found that the paper was somewhat misleading. It promotes efficiency via calculating bucket frequency to accentuate features with parallel coordinates plots. A large part of the paper focuses on the speed benefits of this approach. They also mention that buckets should be calculated between pairs of axes, implying that you would only need to calculate $n - 1$ axes pairs where n is the number of axes. However, this is not true at all. Later on in the paper, they bring

up the point that if you actually want to be able to reorder the axes (a common feature in parallel coordinates), then you would have to calculate frequencies for every possible combination of axes. This is far more than before! With this, we lose much of the speedup we previously obtained with their method. Initially, we implemented the preprocessing by using JavaScript, but found that using Python to precompute the buckets as opposed to computing on page-load made far more sense for the user. With this, the user could simply load data as opposed to processing then loading. Some challenges I faced with precomputing was figuring out how to use gaussian smoothing. In the paper, they simply mentioned that it was a common technique in visualization then linked to an early paper regarding gaussian smoothing. This helped very little with my understanding for what to do. I ended up finding blog posts teaching me how to do 2D gaussian smoothing. In the current implementation, we had to look at various smoothnesses dataset by dataset to determine which alpha value (how much to smooth) for each dataset. From what we could find, the proper value should be the standard deviation. Yet with many datasets, we would get completely uniform data, making the data too smooth. In their implementation, they likely found a better way to run smoothing for displaying/calculating the focus. Another issue with focus is that they never described how the user would interact with their system to specify which areas to focus on. In traditional parallel coordinates, the user may scrub an axis to indicate that they want to look at a subset of data where the values are within a certain range on a given axis. Yet in this implementation, this functionality doesnt seem to work – rendering the original values for the user to see would somewhat defeat the purpose of displaying trends and outliers for extremely large datasets. Scrubbing for finding trends or outliers doesnt make sense either, since the user should already be able to see this clearly.

3 Paper Information

TBD

4 System Differences

Our implementation allows for the user to specify how many bins they want. The paper doesnt cover details of how the user interacts with their system, so we decided to simply add a command line argument to the precomputation script. This is accessible by running python3 bin.py n, where n is the number of bins desired per axis.

4.1 Datasets Chosen

The papers examples used a CFD simulation dataset (mixture of two fluids) and remote sense data. We were unable to find the original datasets, which made it difficult for us to compare our implementations. We chose the international census data to emphasize the speed improvements with their method. The census data is nearly a gigabyte and has over 100 axes, which makes it interesting for us to observe. This dataset was released by the US Census Bureau in 1950 and has its projections through to 2050. Specifically, the original data used to generate this dataset came from observing cities with over 5000 people. We looked specifically at midyear population age by country. Our other datasets are smaller. As the number of axes increased, we had a drastic increase in time spent computing. However, with adding more rows, we didnt observe a similar slowdown. This is likely because adding values to buckets runs in linear time but adding an axis requires that we calculate more buckets for every existing axis.

5 Case Study/Analysis Results

TBD

6 Strengths and Limitations

A key strength of this method is that it scales well for large data sets. Since we simply represent the count in buckets then normalize these buckets, the original size of the dataset has very little influence on that of the precomputed data. A limitation of this method and parallel coordinates overall is that it is primarily designed for numerical variables. As such, we cant work with many datasets. We also cant represent time series or continuous data well, since we bind them to their closest bucket ranges. This work could be furthered by diving deeper into how buckets should be created. Bucket creation is core to their paper and may drastically affect the visualization. We defaulted to using 10 buckets per axis, but this may result in a loss of detail for the final visualization. Another idea for future work would be to improve the outlier detection. The outlier detection works by scanning buckets that are at least 10% full. Changing this barrier would change the outliers visible for the user.

References

[1] Matej Novotny and Helwig Hauser. Outlier-preserving Focus+Context Visualization in Parallel Coordinates. *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 893-900, Sept.-Oct. 2006.

[2] D. Bauer and R.Peikert. Vortex tracking in scale-space. *Joint Eurographics - IEEE TCVG Symposium on Visualization*, pages 140-147, May 2002.

[3] Fabian Bendix, Robert Kosara, and Helwig Hauser. Parallel sets: Visual analysis of categorical data. *INFOVIS*, page 18, 2005.