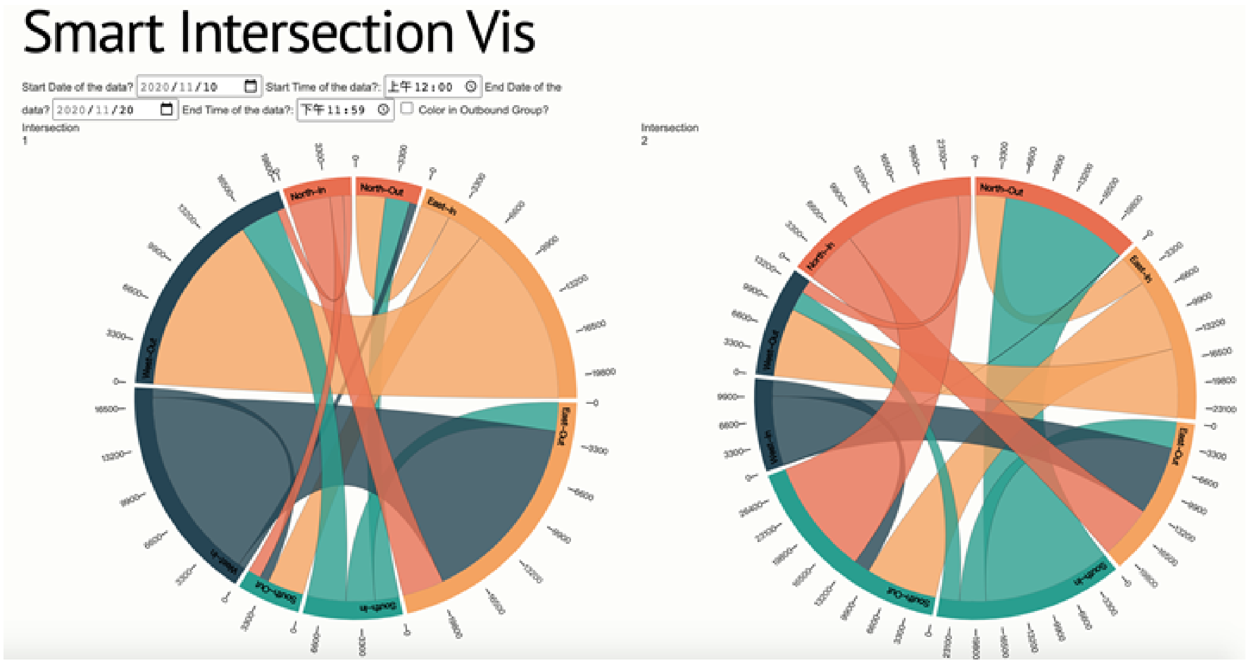
Smart Intersection Visualization

Category: Computer Visualization, Smart Intersections.


Smart Intersection Vis

Start Date of the data? 2020/11/10 Start Time of the data?: 上午12:00 End Date of the data? 2020/11/20 End Time of the data?: 下午11:59 ☐ Color in Outbound Group?

Fig. 1. A view of the Smart Intersection Visualization tool for the two LiDAR instrumented intersections in the City of Kelowna.

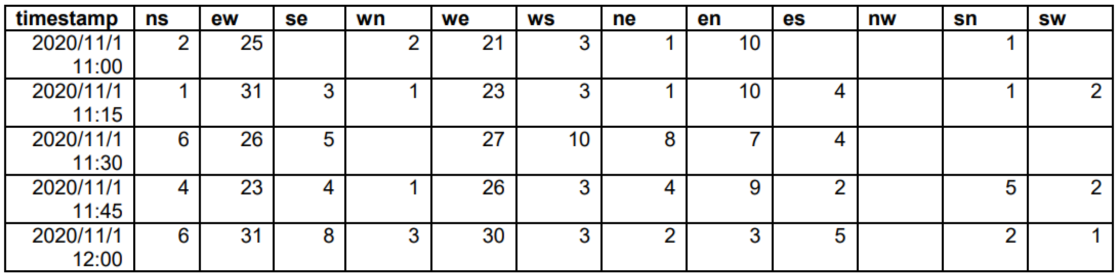
Abstract— The use of different sensors to instrument and gather data on traffic and vehicle flow from intersections is on the rise due to the increasingly connected and smart vehicles and infrastructure that comprise smart cities and vehicular networks of today and the future. There is a need, and gap in the literature and available tools to effectively visualize the intersection flow (of both vehicles and pedestrians) in an interactive manner.
This visualization is needed on a individual-intersection level, as well as a multi-intersection view that would comprise geographic regions or cities. The visualization tool would be primarily used for two critical traffic-flow analysis functions: (i) insight into historical or archived intersection data via comparison and query, (ii) real-time observation of multiple-intersections.

1 Introduction

Traffic congestion, delays, as well as long and sometimes burdensome commutes are an inherent and accepted part of daily life in the majority of urban centers and cities across the globe today. The problem has only increased over the years since the inception of the automobile, and has continued its dramatic progression on urban roads as the world becomes increasingly modernized and the automobile within the reach of many more individuals and households. Despite the exponential increase of automobiles on the road in the past decades – recent investigations have on average predicted a further doubling of automoblies worldwide by 2035-2040[a][b]. New methods of alleviating the increased number of vehicles and resultant strain in dense urban roadways are necessary to sustain the functionality of these networks into the future. A significant step in discovering new ways of dealing with the increased traffic magnitude and resultant incidences and congestion is to instrument the existing roadways to enable exploratory and investigative analysis as well as live monitoring for them. Specifically, it would be beneficial to be able to conduct and visualize scenarios for entire geographic regions (cities, municipalities, highways, and high-collision regions) with cost effective, and 5G-enabled sensors such as LiDARs and cameras.

One of the countermeasures to the problem is Traffic Monitoring and Management using intelligent transportation such as GPSs, cameras and LiDARs. The previous two methods receive their fair share of criticism due to accuracy, deployment cost and privacy concerns. Compared with the other two devices mentioned above, LiDARs are accurate and collect no private or potentially sensitive data. However, the application of LiDARs has been limited due to the cost of the sensor and deployment until recently. With the 5G telecommunication revolution, using LiDAR to monitor traffic conditions becomes a promising solution for many major cities - allowing them to draw insights from traffic flows to develop infrastructure or other more immediate countermeasures and observe potential hazards anomalies in real-time. The benefits are evident when using LiDAR sensors coupled with 5G infrastructure for Traffic Monitoring and Management purposes.

The City of Kelowna, together with Rogers and the UBC Radio Science Lab, plans to install LiDARs on intersections better to understand pedestrians, cyclists, and vehicles' movements. This information can help improve road safety, enable near-miss and conflict analysis. In summer 2020, they installed two LiDARs on two intersections to assess the merits of 5G-enabled LiDAR sensors for traffic monitoring. The current problem that we attempt to solve for our project partners is the visualization of the LiDAR data and traffic flows in a way that is intuitive, interactive and comparable across many different parameters (time of day, certain days of the week, select intersections, evening rush hours on Thursdays and Fridays in the downtown core, intersections along a specific street scheduled for an additional lane). The clients would like to visualize the data so that they can monitor the traffic conditions based on the LiDAR sensors data and make short term traffic regulations or long term traffic construction decisions. Some



timestamp	ns	ew	se	wn	we	ws	ne	en	es	nw	sn	sw
2020/11/1 11:00	2	25		2	21	3	1	10			1	
2020/11/1 11:15	1	31	3	1	23	3	1	10	4		1	2
2020/11/1 11:30	6	26	5		27	10	8	7	4			
2020/11/1 11:45	4	23	4	1	26	3	4	9	2		5	2
2020/11/1 12:00	6	31	8	3	30	3	2	3	5		2	1

Fig. 2. ...

visualization will also be open to public so they can enjoy the benefit of 5G technology as well.

2 Requirements and Goals.

Derived from multiple meetings with the rest of the team for the "Smart Intersections 4.0: Using 5G-enabled Stationary LiDAR to Monitor traffic in Downtown Kelowna" are a summary of the requirements, wants, and future extensibility considerations that this visualization should support:

(i) Effectively visualize the quantity and direction of vehicle flow for a standard 4-way intersection.

(ii) Interactivity over multiple time intervals and locations (for generating insights and comparison).

(iii) Extensible to observe multiple intersections simultaneously (as more LiDAR sensors are installed).

(iv) Extensible to a solution with a Real-Time (RT) data-flow (for observation of multiple city intersections).

3 Data and Task Abstraction

3.1 Chord Diagrams:

The source of the data are the 5G-enabled LiDAR sensors that have been installed at the two intersections. Previous work processes this raw LiDAR data, and using the API of that work we are able to arrive at the data input for this visualization. This data (or the 'what') is in the form of a multi-dimension data table of ordered attributes: 1 column for the timestamp of the observation, and 12 columns for each potential path a vehicle may take at a 4-way intersection (timestamp, NS, EW, SE, WN, WE, WS, NE, EN, ES, NW, SN, and SW). 'SW' is an abbreviation for a vehicle entering the intersection from the South, making a left-turn, and leaving the intersection from the West. Furthermore, as the data is in the form of vehicle counts per potential path for 15-minute intervals; the derived data produced by our tool is the summation of these intervals to produce a single quantitative attribute for each of the 12 potential paths for the time interval specified by the user. Finally, these 12 quantitative attributes are mapped to a specific matrix format that is the input to the chord diagram generation section of the code.

The task (or the 'why) is derived from the four higher-level goals described earlier. The chord diagram represents the part-whole relationship as each of the 12 potential paths (represented by a chord) are a part of entire traffic in and out of an intersection (represented as the 360-degree arc that comprises the chord diagram). Furthermore, the chord diagram represents the abstract task of the visualizing 2D flow of a dataset with multiple inputs and outputs; or more precisely with respect to the domain, the task of visualizing the 2D flow of vehicles entering and leaving an intersection of interest.

3.2 Stacked Bar Charts:

The data, derived data, and sources are the same as subsection above on 'Chord Diagrams' - except that the final input to the stacked bar chart generation section of the code is a multidimensional table of quantitative attributes (the data right before processing into the matrix from the chord diagram section).

The task (or the 'why') is to complement and address limitations of the part-whole relationship of each of the 12 potential paths of the chord diagram. The stacked bar chart retains the part-whole relationship, but allows further comparison of the magnitude of the total input to the system – or net number of vehicles in and out of an intersection in the context of the problem domain. The stacked bar chart no longer displays the 2D flow of vehicles, but complements the chord diagram with the resizing of the entire stacked bar to represent the magnitude of total vehicles entering or leaving an intersection. The alternative of resizing the chord diagram to reflect this important quantitative attribute for traffic flow analysis is unfeasible - as the clarity of the chord diagram greatly reduces when scaled down.

4 Solution

This section details our visualization solution, as well as the rationale for the design choices and idioms used. The smart intersection visualization problem is detailed in Section (x), and the solution has the requirement of satisfying the four high-level project goals as set out by the team conducting the "Smart Intersections 4.0: Using 5G-enabled Stationary LiDAR to Monitor traffic in Downtown Kelowna" interdisciplinary research investigation. The four high-level goals are mention earlier, the solution and design choices are described with respect to these requirements.

The key feature, and most basic unit being visualized in this work is the flow of vehicles (and eventually pedestrians as well) at individual intersection level. This is the flow of vehicles or pedestrians entering and leaving a standard 4-way intersection (with inputs generalized as north (N), south (S), east (E) and west (W)) - creating 12 unique paths an observed vehicle can take (NS, EW, SE, WN, WE, WS, NE, EN, ES, NW, SN, and SW). The visualization choices for effectively depicting this type of flow were identified as (I) Sankey diagrams, (II) chord diagrams, and (III) bundled arc diagrams.

Our initial design choice was the Sankey diagram as the final solution would stitch together multiple individual Sankey diagrams to present traffic flow at an individual intersection, and multi-intersection level that is clear, novel, and non-obvious. Furthermore, there was clear pathway to having this diagram overlay on top of a geographic map of multiple intersections in a city. However, the tool and library limitations for both building a multi-way Sankey diagram (bi-directional, not quad-directional flow is most commonly visualized with this chart type and so only the most common case is supported by the available tools) and to stitch multiple Sankey diagrams together. It was outside the scope of the timeline and goals of this project to make these changes to libraries and tools available. However, this was an interesting step in the design process that yielded a potential novel way to visualize traffic flow for intersections (both a single and multiple intersections) that warrants further investigation and an attempt realize in future work.

4.1 (I) Geometry:

The circular geometry of the chord diagram does not seem to initially map as cleanly to the rectangular shape of the standard 4-way intersection.

Solution: Splitting the chord diagram into 8 arcs (rather than just 4), two for each cardinal direction added great clarity to the resultant visualization. With this, the mapping of real datasets from the intersections show that this geometrical mismatch is not a hinderance for inferring the desired visualized attributes.

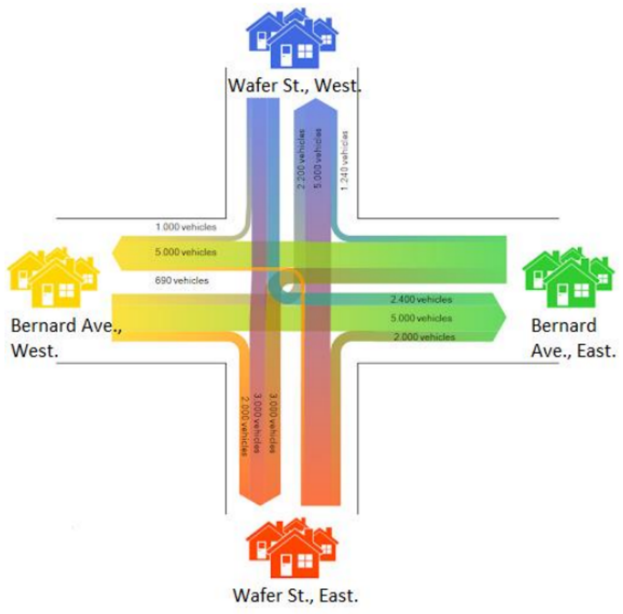


Fig. 3. Results of the top 3 Coins (BTC, ETH, and LTC) for the first set of parameters. These parameters are mainly comparison-adjusted metrics that deal with the distribution of pools and P2P nodes.

4.2 (II) Non-Standard Intersections:

The design choice should not be limited to the standard 4-way intersection.

Solution: Splitting the chord diagram into 2 arcs for each intersection input direction (same as solution to (I) Geometry). With this, the mapping of generated test datasets from the intersections with 3 or 5 input directions show that chord diagram is suitable other, non-standard geometries of intersections as well.

4.3 (III) Arc-Distortion:

If traffic flow is very heavy in one particular direction (due road closures, construction, accidents, etc.) the proportion of certain arcs can grow to distort the geometry and thus interpretation of the diagram.

4.4 Solution:

The arc-distortion acts as a channel in many of the disproportionate real-world cases tested. In some exaggerated or extreme, simulated test-cases the diagrams were distorted - a potential solution to this in the future (if it is a problem) is to code spacers between the arcs that adjust maintain the geometric integrity of the diagram.

4.5 (IV) Absolute Values vs Proportionality:

The chord diagram can not depict the magnitude of the total number of vehicles entering an intersection. This is because there are only 360 degrees in a circle, and adjusting the size of the chord diagram is not feasible due to clarity and visibility reasons.

Solution: Complement the chord diagram with a stacked bar chart that can effectively display the magnitude of the total number of vehicles entering an intersection. Further description of the two diagrams can be found in the 'Data and Task Abstraction' section.

4.6 (V) Cluttering in the Middle of the Chord Diagram:

The chord diagram can lead to cluttering, and resultant lowered visibility and clarity, in the center of the diagram (where all the chords overlap) when extending the number of chords.

Solution: Use non-fully saturated colors and complementary colors from the same color scheme or pallet.

Small multiples was chosen over geographic overlay for multiple-intersection visualization for two primary reasons; the first being that a background map would reduce the clarity of the chord diagram and that small multiples can be ordered and ranked based on queries as an extension (I.e. sort and display intersections from largest vehicle magnitude to the least). ' gram is suitable other, non-standard geometries of intersections as well.

5 Implementation

In this section, we discuss the technical detail of our implementation and what components each author did work on.

5.1 Technical Details

The library we use is D3.js. We choose this library because it is open-source and free, as our tool will be used/edited by others, we do not want to use commercial software that will incur licensing issues and extra costs. Meanwhile, D3 is based on JavaScript, which means we can incorporate it into the webpage easily. Other available libraries are R (none of us have any previous experience with it) and python (do not have a free well-supported library for chord). As none of the group members are familiar with many popular JavaScript libraries such as React, jQuery and Bootstrap, the interface is mainly written in plain HTML, CSS and JavaScript. Our first challenge is data pre-processing. Because the raw data is tabular while the chord diagram takes a matrix as the input, we need to design an algorithm to filter, sum up the raw data and generate a corresponding matrix. We first split the time stamp into dates and times and then filter out the data we want to display based on user input. Afterwards, we sum the numbers in each attribute and get an array with 12 entries. Our chord diagram has eight groups (inbound and outbound for each direction), so we designed an 8x8 matrix that is diagonally symmetric because group bindings such as west-inbound to east-outbound and east-outbound to west-inbound should have the same value. Other matrix cells are left with zero.

The chord diagram [1] and stacked bar plot [2] are provided in a D3 sample. The functionalities in these samples are limited, and one of the significant challenges in this part is that the bar chart and chord diagram is from different D3 versions (v3 for the chord and v5 for the box plot). As D3 underwent extensive updates during these two versions, many functions are depreciated, and much effort is used on writing an upgraded version of the chord diagram.

As the chord diagrams focus on this project, we put much effort into upgrading the chord diagram from a raw sample. We added the ticks around the outer groups with tick size auto adjustment to provide another channel to visualize the traffic flow size. We make the chord diagram interactive as when users move the cursor onto one specific group, only ribbons connected to this group will appear, and details of the data will appear on a label. We arrange our groups to abstractly showing the travelling direction of an intersection viewing from the top, and groups orders are alternating between inbound and outbound. The default chord picks a colour from the group where the ribbons start, and we designed our colouring function because we choose to stick to colouring all ribbons either from inbound or from outbound directions. All the groups and colour choices are written in a separate file, so this tool can be easily adapted to be used on other datasets.

5.2 Work Breakdown

Before the progress report is completed together, the group members evenly write up the proposal and come up with the design choice of a chord diagram together. Huancheng wrote the related work and data/task abstraction in the progress report, while Nikhil wrote the Introduction and Proposed Solutions. After the progress report, the initial task breakdown is that Huancheng will focus on d3 visualization implementation (chord diagram, stacked-box plot) while Nikhil will focus on the data filtering and pre-processing. However, due to Nikhil having a paper due on Monday night during the week of presentation, and he did not manage to get a working version of the data filtering, the presentation demo is made based on work from Huancheng Yang with minimal data filtering. After the presentation, the work is split as Nikhil

[1]https://bl.ocks.org/vikkya/b160c62ac2043251ca658ac462bca7fc

[2]https://bl.ocks.org/LemoNode/21d81ff82e80cbe4acbfee28ff060b11

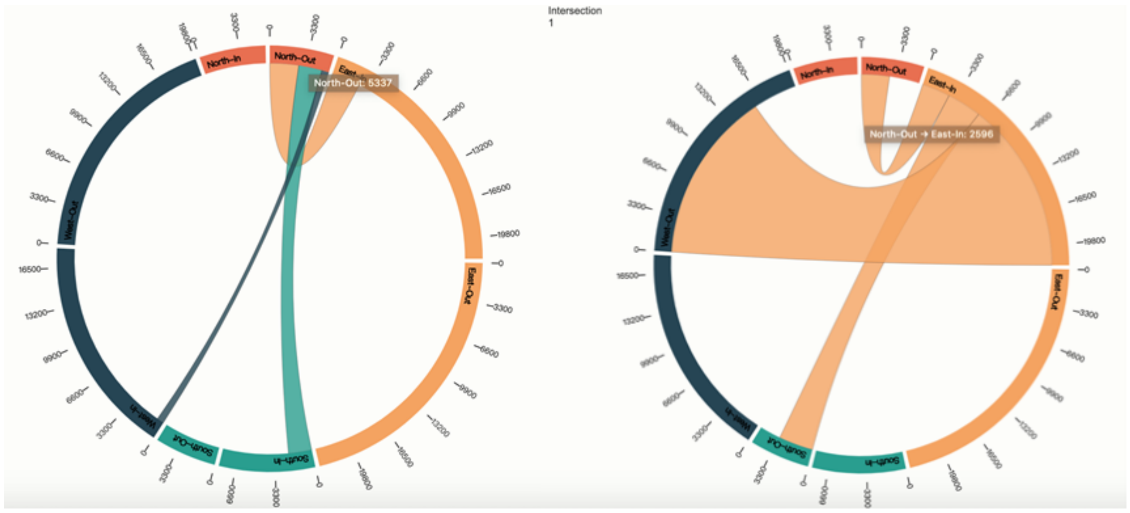


Fig. 4. Users hover the mouse on the chord diagram, left images shows label when hovering on specific direction, right image shows label when hovering on the specific ribbon.

will mainly finish up the introduction, related work, data/tasks abstractions and solutions, while Huancheng will fix the data pre-processing, apply more advanced filters and incorporate data from the other intersection. Meanwhile, Huancheng will write the implementation, result, discussion sections of the report.

6 Results

Scenario 1. User want to compare two intersections' data in the same time interval In this scenario, the user would like to compare the two intersections' data during the same interval. The user can input the desired date and time, and the chord diagram will refresh with the new filter criteria. The two intersections are overlayed side by side. The user can also choose to colour the ribbons using the outbound direction's colour by checking the box beside. Figure 1 is an example output based on data from Nov 10th to Nov 20th :

The overview of traffic flow in each direction, and each intersection is well presented, with the outer tick shows the estimation of traffic flow in each direction. If the user wants to look in a specific direction, it can hover the mouse onto the ribbon or the group. See Figure 4 for demonstration.

In this way, the chord shows a clear proportional while it remains useful in fetching the numerical result. As we can see, because the first intersection is undergoing construction in south/north directions, the traffic flow in these two directions is limited, while the second intersection has a large traffic flow from North, South and East directions. In the first intersection, most vehicle traffics straight without turning, while most traffic from south and north directions is turning to the left and right to avoid the construction.

Scenario 2. User want to compare same intersections' data in the two different time intervals This is when the user chooses to display one intersection at two different time intervals and compare them. We can see the chord diagram shows shrinkage in North Out and South In while expands West Out and East In. This represents two peak hour patterns in the intersection. We noticed that although the traffic flow is doubled during the afternoon compared to the morning, the chord diagram shows very minimum change (only the outer tick), which is why we added a stacked bar chart to complement this disadvantage. See Figure 5 for demonstration

Scenario 3. Users want to compare numerical values of traffic during each time interval in a day When users would like to compare numerical values instead of proportional relationships, the best way is to use the horizontal stacked bar chart. By choosing the day from the bottom, the user can view the traffic flow breakdown in multiple 3-hour time intervals on that day. The bar chart clearly shows the numerical changes, and it uses the same colour code as the bar chart. The user can also toggle sort at the bottom to rank different time intervals. The bar chart's main drawback is that the proportional relationship is hard to visualize, and the user cannot see the detailed breakdown of the traffic flow in each direction (from where to where), which is the primary goal of the chord diagram. See Figure 6 for demonstration

Scenario 4. Users want to add more intersections/real-time data processing Our algorithm is designed to rely on external data files and feeds, and we wrapped up the chord diagram functions to provide an interface for generating the diagram. The user can easily adjust the program to include the online data feed and create a new div space to put other diagrams. The group name and corresponding colour coding are also external files to modify easily without digging through the code.

7 Discussion and Future Work

Our design's strength is that we use a chord diagram and a bar chart to complement each other to fulfill the requirement of traffic flow visualization as our solution takes care of the numerical relationships and the proportional relationships. Meanwhile, we designed our chord diagram to handle basic numerical value visualizations, so the chord diagram is useful to be left on the screen to monitor traffic data in real-time, which is the project's high-level application. Our design's weakness is that we focus too much on the chord diagram and do not put enough effort into developing the stacked bar chart. We initially thought of the bar chart as a limited complement of the chord, and when we realize not everything can be visualized on the chord diagram, and making numerical comparison on a chord diagram is messy, we do not have enough time to create a stacked bar chart that has the same number of features as our chord diagram. We learned from this project that the most powerful tool is not always the most appropriate tool. We use D3 because the chord diagram is a new visual design, and not much software is supporting it, let alone we need some customization on the diagram. D3 is a powerful tool for us to achieve our goal, but given the limited time, we do not utilize the full potential of D3. If we have another term, the project will be much more improved and have more impressive features, but as somebody said, building a visualization in D3, you must also have a deep understanding of SVG, DOM, JavaScript, geometry, colour spaces, data structures. Without a solid foundation, debugging the code you are not familiar with is time-consuming and challenging. The other thing we learned is the limitations of JavaScript. As most computers do not allow JavaScript to write to local files, it is hard to automate the pre-processing process. Therefore, the pre-processing must be done during the webpage loading at the back, and this is not only hard to do but also inefficient. We choose JS because we can do webpage input, data processing and visualization (d3) under the

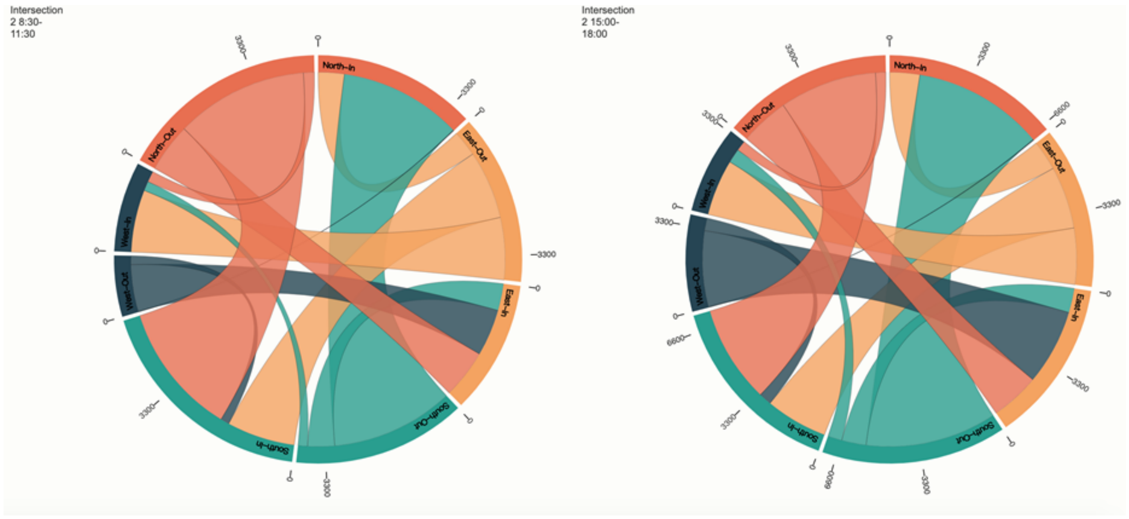


Fig. 5. Overlaying two chord diagrams to compare two different time intervals on the same intersection.

same structure, but if given another chance, we will consider a better alternative. If we were given more time, we will further improve the stacked bar chart and introduce more features. We will make the two graphs interactive, such as clicking on the stacked bar chart to display this chord diagram data. We will also introduce a more advanced filter like selecting only Monday data.

8 Related Work

8.1 Traffic Flow Monitoring using Sensors

Using sensors at intersections for traffic analysis has been prevalent due to technological advances. Although using LiDAR sensor to monitor intersection traffic is a new application, there have been other traffic monitor applications that uses other sensors.

Banerjee [2] uses fish cameras and high-resolution signal data to show trajectory patterns by drawing them directly onto the intersections and crosswalks. This method is straightforward at showing an object's movement but not efficient for aggregate data count and analysis.

In Wang's Paper [3], he presents a visualization on traffic jam analysis based on trajectory data and creates several visualization designs, including the spatial view showing the traffic jam density on each road of the city by color, an embedded road speed views show the speed patterns of four roads, a graph list view shows a list of sorted traffic jam propagation graphs. His visual design is based on the vehicle fleet's GPS data including speed and position information, which enables a much-detailed traffic information visualization. Figure 1 Wang's proposed vis design for traffic jam analysis. (a) represents the spatial view of traffic jam, (b) represents the pixel-based road speed view, (c) represents the sorted propagation graph of traffic jam inside the green rectangle, (d) represents the filter view for (c).

8.1.1 Traffic Condition Visualization Design

Song [4] proposed a visualization for intersection traffic flow data using a radial layout. He uses circles to represent the number of vehicles, and different colours represent vehicles from different directions. He also creatively uses a radical layout to present the vehicle flow in 24 hours for seven days. However, using rings inside the circle makes the data near the center of the circle hard to read. We will see if we have any chance to improve this visualization method.

The current traffic visualization we see in Google Map or other software can only show traffic in each direction. In Box's paper [5], he designed a 3D method to visualize the traffic down to lane level by overlaying traffic-light color encoded cuboid on each lane to show how fast the traffic flows on each lane. This visualization is easy to understand, but does not convey much information in the vis, and is good for traffic users, not traffic monitors.

Roberg-Orenstein's paper [6] studies how to use traffic regulation to break the deadlock in traffic jams in incident-induced traffic jams. What is interesting is that the author uses a grind view to show the traffic jam's impact on the overall traffic network, but it lacks the interactivity that enables the user to zoom to a specific area.

8.2 General Visualization Choices

The idea of the necklace map [7] is placing a necklace around the map region to present statistical data instead of presenting it directly on the map. This method saves precious space inside the map for other visualizations, and the symbols on the necklace are customizable. We utilized the necklace idea in our proposed solution to potentially show other information around the intersection.

Ocupado [8] is a tool for visualizing location-based count over time across buildings. It provides a comprehensive set of tools to compare data under different scenarios, including the zoomable binned time series chart, which helps compare intersection data under different time intervals and show trends. The spatial heat map is useful in locating high-volume intersections for users quickly.

Prior work has taken edge bundling approaches in visualizing routes in urban traffic. Danny Holten and Jarke J. van Wijk use edge bundling to visualize US airline networks and migration graphs. Edge bundling results in a less cluttered figure and preserves the map's geographical shapes while clearly indicating the links between nodes. The hierarchy relationship between nodes in a traffic-visualization is hard to define as many small edges will be merged into inter-cluster edges. Fortunately, their application does not require a hierarchy relationship to bundling the edges. This scenario is useful when we have data from multiple intersections in the future and would like to show connections between them and therefore have an overall view of the traffic flow information.

9 Conclusion

In this paper, we describe our smart intersection vis design for two traffic intersections in Kelowna. Our design features a chord diagram and a stacked bar chart. The chord diagram is chosen to monitor traffic flow in multiple intersections at the same time under small multiple layouts. It is modified to accommodate traffic data with a specially designed data matrix while adding many features to the traditional chord diagram to enhance the user's experience and complement the chord diagram's disadvantage in displaying numerical values. The stacked bar chart is used to make a numerical comparison among traffic data at different time intervals. These visualizations help the traffic data analysts monitor various intersections' traffic data simultaneously and find traffic patterns quickly by comparing data across different date and time intervals, while also enabling future integration of data from new intersections and other traffic data.

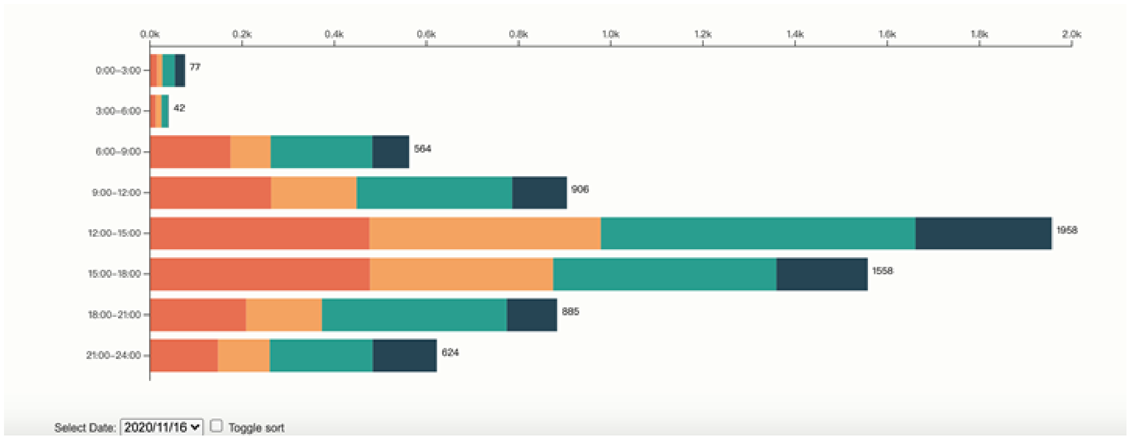


Fig. 6. Bar Chart with traffic flow comparison from different time intervals in the same day

References

[1]D. Schrank, ”2019 Urban Mobility Report - Texas AM University,” 2019 Urban Mobility Report, 2019. [Online]. Available: https://static.tti.tamu.edu/tti.tamu.edu/documents/mobility-report-2019.pdf. [Accessed: 23-Oct-2020].
[2] T. Banerjee, K. Chen, X. Huang, A. Rangarajan, and S. Ranka, “A Multi-sensor System for Traffic Analysis at Smart Intersections,” 2019 Twelfth International Conference on Contemporary Computing (IC3), Aug. 2019.
[3] Z. Wang, M. Lu, X. Yuan, J. Zhang, and H. V. D. Wetering, “Visual Traffic Jam Analysis Based on Trajectory Data,” IEEE Transactions on Visualization and Computer Graphics, vol. 19, no. 12, pp. 2159–2168, 2013.
[4] W. Song, C. Huang, and B. Jiang, “Visual methods for time-varying intersection traffic flow data,” 2017 4th International Conference on Systems and Informatics (ICSAI), 2017.
[5] S. Box, X. Chen, S. Blainey, and S. Munro, “Fine-grained traffic state estimation and visualisation,” Proceedings of the Institution of Civil Engineers - Civil Engineering, vol. 167, no. 5, pp. 9–16, 2014.
[6] P. Roberg-Orenstein, C. Abbess, and C. Wright, “Traffic Jam Simulation,” Journal of Maps, pp. 107–121, 2007.
[7] B. Speckmann and K. Verbeek, “Necklace Maps,” IEEE Transactions on Visualization and Computer Graphics, vol. 16, no. 6, pp. 881–889, 2010.
[8] M. Oppermann and T. Munzner, “Ocupado: Visualizing Location-Based Counts Over Time Across Buildings,” Computer Graphics Forum, vol. 39, (3), pp. 127-138, 2020.
[9] T. Munzner, Visualization analysis and design. Boca Raton: CRC Press, 2015.