Using Camera State Transforms for Commuter Network Visualization

Yves Chiricota, Michael J. McGuffin, and Martin Simard

Abstract—We present a novel approach for visualizing commuter networks, i.e., directed graphs whose nodes (cities and towns) each have a geographic location, and whose edges each have a direction and an associated number of commuters moving between two nodes. Our approach involves using *camera state transforms* that map the current camera state (position and orientation) to various rendering parameters to achieve a hybrid 2D-3D visualization. As the camera's angle and distance change, the shapes and transparency levels of nodes and edges morph in response, allowing for a smooth transition between a 2D view (from above) to a 3D view (from the side) that reveals information in different ways.

Index Terms—Interactive network visualization, geographic information systems.

1 Introduction

A commutor network describes the number of workers traveling each day between their city of residence to their city of work. Visualizations of commuter networks allow geographers to better understand patterns of human labor, identify poles and sub-poles, determine areas of influence, and construct models related to economics, sociology, and territorial organization. For instance, analyzing commuter networks helps geographers understand how cities are organized in terms of transportation and communication.

Each node in a commuter network represents a city and has a meaningful geographic location. Hence, unlike the visualization of many other graphs, the positions of nodes in a commuter network are fixed. Furthermore, the geographic distribution of nodes can be highly non-uniform, and their can be a large number of edges, leading to edge occlusion problems. New methods are required to improve the visualization of such networks.

We have developed a prototype system for visualizing such networks (Figure 1). The data shown has been previously visualized in [3], but we have since developed new visualization features. The ground plane is displayed as a texture map of a traditional 2D map, and the commuter network is displayed on top of this ground plane. The user may freely move a 3D camera around the visualization. Clicking on a city triggers a smooth camera movement from the current point of interest to the new one. For any given city, the user may also optionally display only incoming or outgoing edges.

An edge filtering feature allows the user to hide all edges whose number of commuters falls outside a desired range, allowing the user to declutter their view and only see edges with large or small numbers of commuters. To select the range of visible edges, we developed a range slider widget to select an interval of values. Slider positions within the widget are mapped to the inverse of the cumulative distribution of metric values, so that the number of elements filtered is linearly proportional to slider position. This makes the widget "auto-adaptive" to the metric values present.

The most novel feature of our system is how the rendering of the visualization depends on the camera's current state. This dependency is determined by what we call *camera state transforms*, enabling us to create a hybrid visualization that combines advantages of a 2D map and a 3D scene. As the camera moves away from, or toward, a point in the scene, and as the camera is angled away from, or toward, a horizontal orientation, the transparency levels, shapes and sizes of the nodes and edges adjust automatically.

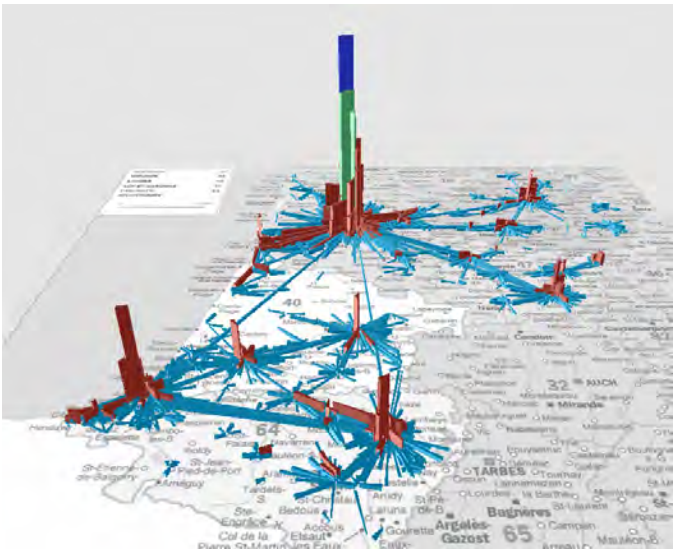

Fig. 1. Screenshot of our prototype, showing the network of daily commuters between cities and towns in Aquitaine, France, in 1999. This network comprises approximately 32 000 nodes and 800 000 edges. Directed links are represented as wedges (trapezoidal prisms) with one trapezoidal side on the ground plane, where the height (and color) of a wedge depends on the number of commuters.

When the user is looking straight down at the data in a vertical camera orientation (Figure 2, bottom), the data are rendered in such a way as to produce a 2D map, with nodes rendered as circles whose radius reflects the number of workers living in the city, and with edges rendered as trapezoids whose shape indicates direction (text labels on the edges can also be displayed to show number of commuters). As the user tilts the camera toward an oblique or side view (Figure 2, middle and top), the nodes are shown as cylinders that gradually become thinner and taller, whose height reflects the number of workers living in the city. At the same time, edges gradually become tall, thin boxes, whose height reflects the number of commuters traveling between cities. Furthermore, as the camera tilts from a vertically-oriented top view, toward an oblique or side view, the sides of the cylindrical nodes and prismatic edges become more opaque, while the tops of the edges si-

- *Yves Chiricota is with Université du Québec à Chicoutimi, Chicoutimi, Canada, E-mail: Yves_Chiricota@uqac.ca.*
- *Michael J. McGuffin is with École de technologie supérieure, Montreal, Canada, E-mail: michael.mcguffin@etsmtl.ca.*
- *Martin Simard is with Université du Québec à Chicoutimi, Chicoutimi, Canada, E-mail: Martin-G_Simard@uqac.ca.*

multaneously become more transparent. The user can seamlessly transition between the 2D map and the 3D scene simply by tilting the camera. Finally, as the user moves the camera closer to a point of interest in the scene, the heights of edges are scaled down, to reduce occlusion and allow the user to see more detail (Figure 3).

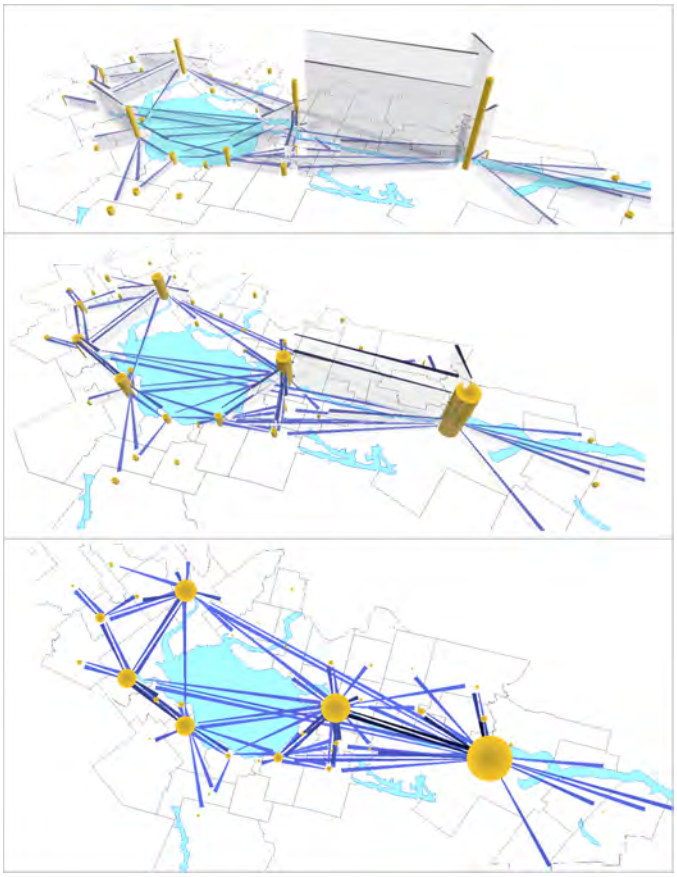

Fig. 2. This figure illustrates the transformation from 3D view to 2D map, as controlled by the view angle ω between the camera's axis and a vertical line. From top to bottom, this angle is $60°, 45°$, and $0°$, respectively. As ω decreases toward zero, the form of the edges morphs from flat boxes to trapezoidal prisms, the tops of the edges become more opaque, the sides of the edges become more transparent, and the cylinders representing nodes becomes more fat and short.

The advantage of the 2D "map" view seen from above is that there is no occlusion due to 3D structures, and navigation simply involves panning and zooming. The advantage of the 3D view is that metric values (number of workers) are shown as heights rather than as thicknesses or colors (judgments about relative differences in quantities are easier to make when the quantities are shown as lengths rather than colors or areas [4], and there is more spatial resolution available to show heights in the 3D view than to show thicknesses in the 2D view). By allowing the user to seamlessly transition between 2D and 3D by tilting the camera, we allow the user to take advantage of the benefits of both views with a single user interface.

Our system also allows metric values to be visualized as a surface, as in Figure 4. When viewed from above (Figure 4, right), the opacity and color of the surface is a function of the metric value, making higher values stand out. When viewed from an oblique angle ((Figure 4, left and middle), the surface becomes more uniformly opaque, with the metric value shown by the height and color of the surface. As with the other camera state transforms in our systems, this transition is gradual and seamless, and occurs as the user tilts the camera.

We can compare the camera state transforms in our system to previous visualizations where the rendering depends on the camera state

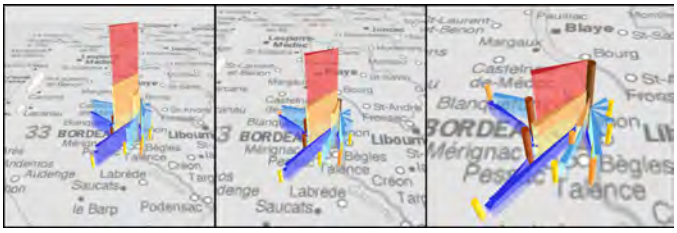

Fig. 3. Z-scaling of the edge heights shown in three screen shots. From far away (*Left*), z-scaling increases the differences between rectangle heights to make important edges more apparent. As the camera approaches a point of interest (*Middle and Right*), in this case the Bordeaux area, the scaling is reduced. The height and color of each edge reflects the number of workers moving between cities.

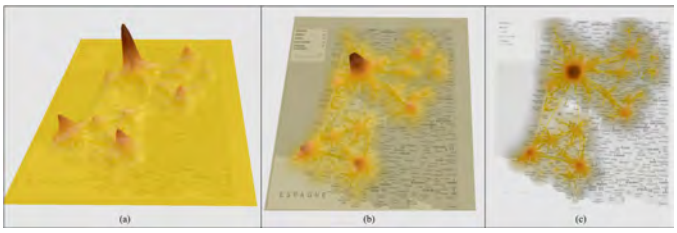

Fig. 4. A surface representation of a metric value, at three camera angles ($\omega = 45°$, $20°$, and $0°$, respectively).

in some way, resulting in the below taxonomy:

		INPUT: camera parameters		
		Angle between line-of-sight and surface	Distance to camera	Presence of objects along line-of-sight
OUTPUT: rendering parameters	Transparency	Our system (transparency depending on camera angle)		Importance-driven rendering [5]
	Graphical representation		Semantic zooming [1]	
	Embedding; i.e., location, size, or form of geometry	Our system (nodes and edges that morph depending on camera angle)	Our system (edges that change size depending on camera distance)	distortion viewing [2]

Our system is implemented using OpenGL and uses GPU shaders to accelerate the computations involved in applying camera state transforms, to maintain interactive frameworks.

Acknowledgments

This work was supported by NSERC (Canada). Our data sets come from the national censuses in France and Canada.

References

[1] B. B. Bederson and J. D. Hollan. Pad++: A zooming graphical interface for exploring alternate interface physics. In *Proceedings of ACM Symposium on User Interface Software and Technology (UIST)*, pages 17–26, 1994.

[2] M. S. T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia. Extending distortion viewing from 2D to 3D. *IEEE Computer Graphics and Applications (CG&A)*, 17(4):42–51, 1997.

[3] Y. Chiricota, G. Melançon, T. T. P. Quang, and P. Tissandier. Visual exploration of (French) commuter networks. In *Geovis*, Gerone, Spain, 2008.

[4] J. D. Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics (TOG)*, 5(2):110–141, April 1986.

[5] I. Viola, A. Kanitsar, and M. E. Gröller. Importance-driven volume rendering. In *Proceedings of IEEE Visualization (VIS)*, pages 139–145, 2004.