Interactive Poster: Using Edge Plucking for Interactive Graph Exploration

Nelson Wong* and Sheelagh Carpendale*

Department of Computer Science, University of Calgary

ABSTRACT

Excessive edge density in graphs can cause serious readability issues. Edge Plucking, an interactive tool that clarifies node-edge relationships by bending edges while preserving node positions, extends previous exploration tools that have been designed to address edge congestion.

CR Categories: I.3.6[Computer Graphics]:Interaction Techniques

Keywords: information visualization, graph layout, navigation, interaction, edge congestion

1 INTRODUCTION

Graphs are useful for representing information that has entities that can be mapped to nodes and relationships that can be mapped to edges. However, graphs can be complex to lay out and difficult to read. For instance, high edge density may cause edges that are close to one another to overlap, or to pass underneath nodes. We call these phenomena *edge congestion* [3]. One research direction to address edge congestion is to improve the graph layout [1]. However, this is a difficult task and involves node repositioning, which for some information is not suitable. This includes graphs representing road maps, airline routes, and telecommunication networks, and other information where nodes carry either geographical information or other spatially explicit information.

To address this issue, we developed EdgeLens [5] which is an interactive technique that pushes edges away from its centre while preserving the locations of nodes. Subsequently we have expanded on this direction, developing Edge Plucking [4] which is another interactive technique that can clarify the confusion caused by edge congestion without moving any node positions. In stead of pushing edges like EdgeLens, Edge Plucking pulls edges. We will describe the concept of Edge Plucking, followed by its interaction and algorithm.

2 EDGE PLUCKING

Edge Plucking is intended to provide users with the ability to temporarily move edges (one at a time or as a group); in so doing, the technique helps clarify graph structure, node-edge relationships, and associated information. Our plucking metaphor is taken from everyday life. For example, as in Figure 1, when one wants to peek through a set of closed Venetian blinds, one may run a finger down then peek through it. Also, when one releases the blinds, they will return to their original shape. Edge Plucking simulates this plucking action for use in graph exploration.

To pluck edges, one drags the cursor across one or more edges. Moving the cursor across edges will pull them to the direction that the cursor is moving (Figure 2). As long as the mouse button is held, edges will be plucked when the cursor moves across them. Once the mouse button is released, all edges will return to their original positions.

*e-mail: {yw,sheelagh}@cs.ucalgary.ca

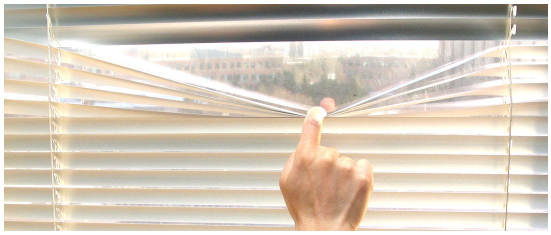

Figure 1. Running a finger down a set of closed Venetian blinds

Edge Plucking includes a variant visual response depending on the location of the point on the edge from which plucking is initiated. This variation in visual response appears quite natural in that the longer part of the edge appears more stretched and the shorter part of the edge seems to have somewhat less tension. Also, the shape of the curve gets tighter, in that the bend gets much sharper, as the pluck point approaches a node. While this does introduce the possibility of a sharp bend, it also keeps the shape of the edge more neatly within the boundaries set by the edge's nodes. This is illustrated in Figure 3: a) is the untouched edge; b) through d) show different shapes of a plucked edge from smooth to a sharper curve.

To support exploration at different areas of a graph, the plucked edges can be pinned by a right mouse click. The edges that are pinned keep the exact shape they had the moment they were pinned. Other edges can be plucked while some edges are pinned. Pins can be removed after exploration, and then edges will return to their original shape.

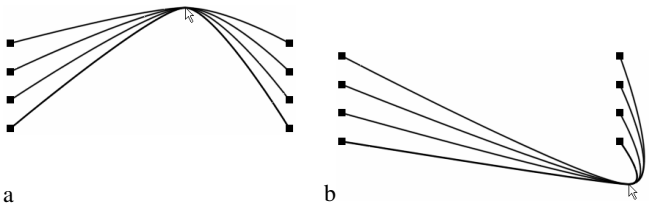

Figure 2. Edges are plucked in different directions

2.1 Edge Plucking Algorithm

A plucked edge is composed of two connecting cubic Bézier curves. They are controlled by seven control points (*cp*), where *cp1*, *cp2*, *cp3*, and *cp4* control one curve, and *cp4*, *cp5*, *cp6*, and *cp7* control the other curve. Control points *cp1* and *cp2* have the same location as node *n1*, and control points *cp6* and *cp7* are both located at node *n2*. The middle control point, *cp4*, is shared by both curves, and is located at the mouse cursor *mc*.

When the mouse cursor touches the edge at *mc*, the edge is considered as two line segments joined at *mc*. The first step is to assess which line segment *n1* to *mc* or *mc* to *n2* is shorter. The control point on the shortest line segment will be calculated first.

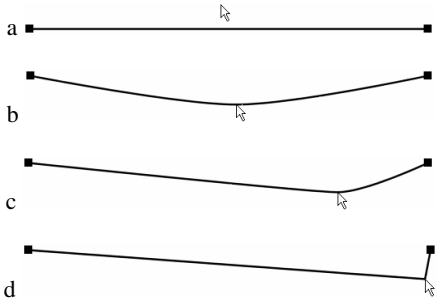

Figure 3. An edges is being plucked in different ways: a) the original graph; b) the edge is plucked in the middle; c) the edge is plucked on the right side of the edge; d) the edge is plucked very close to the right node

In Figure 4 this segment *n1* to *mc* is shorter and, as a result, *cp3* will be calculated first using the formula:

$$dc = dn * r$$

where *dc* is the distance from *cp3* to *mc*, *dn* is the distance from *n1* to *mc*, and *r* is a number between 0 and 1. While *r* can be interactively adjusted, the current default for *r*, as used in the illustrations in this paper, was arrived at through observation and is *0.3*. In Figure 4, the control point for the longer line segment, *cp5*, is placed between *mc* and *n2* using the same distance *dc* just calculated for the shorter line segment. This places the two calculated control points equidistant from *mc* and all three are on the original edge and thus are co-linear.

At this point the mouse has touched the edge and we are ready for the edge to be plucked. When the mouse cursor moves, and consequently *mc*, the co-linear relationship between *cp3*, *mc*, and *cp5* is maintained. The three points move in the same direction and distance as *mc* moves, maintaining a parallel relationship to the original edge (Figure 5). Maintaining this co-linearity and using cubic Bezier splines, in spite of the fact that the end control points must be doubled at the nodes in order to get four control points per segment, does provide us with C^2 continuity at the location of the mouse cursor. This visual continuity seems important to provide the feedback that the edge is still a single edge and is responding as a unit.

2.2 Applying Edge Plucking

Similar to EdgeLens, Edge Plucking moves edges apart, revealing underlying information and disambiguating node-edge relationships. However, Edge Plucking does offer particular advantages when clarifying graph structure. Since one can use

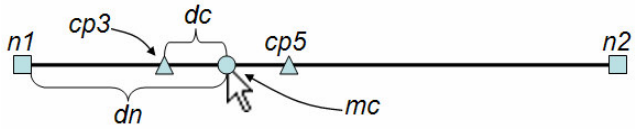

Figure 4. Illustrate how *cp3* and *cp5* are calculated

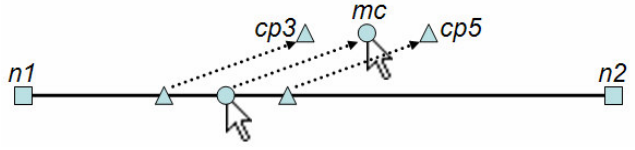

Figure 5. *cp3* and *cp5* move in the same direction and distance as the cursor

Edge Plucking to interact with one or more edges and pin them separately, it is able to aid exploration of a graph (or portion of). Figure 6 shows an example with both EdgeLens and Edge Plucking separately applied on the same portion of a graph. In Figure 6a there are seven nodes connected by several edges that are all lying along the slight circular arc. Applying an EdgeLens to the region of interest does reveal the structure (Figure 6b); however, it can be difficult to choose just the right position to achieve a reasonable spread to the edges and the structure is still somewhat difficult to decipher. Instead, one can use Edge Plucking (Figure 6c to 6h) more deliberately. Following through from Figure 6c each edge is plucked and pinned to the side, gradually revealing the structure. Pinning all six edges clarifies the relationships between nodes. While Edge Plucking can be effective with larger graphs, we have illustrated this discussion with a small graph so that the static images will be clear. Much of the effectiveness is a result of the motion that is caused by plucking. For instance, the motion caused by plucking an edge or group of edges makes long edges easy to follow as suggested by Ware and Bobrow [2].

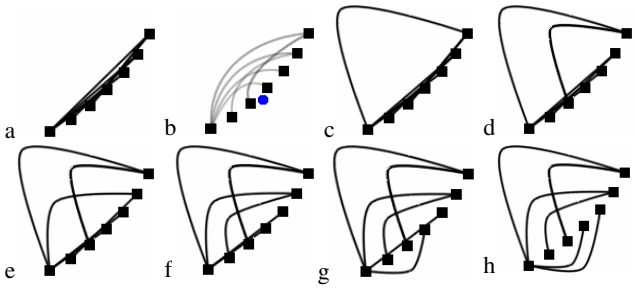

Figure 6. Exploring a cluster of nodes in a graph: a) the original graph; b) EdgeLens is applied; c) through h) Edge Plucking is used separate the edges within the cluster of nodes

3 Conclusion

In this paper we described Edge Plucking, an interactive technique that clarifies graph structures. One can use it to temporarily pluck edges apart to reveal hidden structures while preserving node position.

Acknowledgements

This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

Reference

[1] Di Battista, G., Eades, P., Tamassia, R., and Tollis, I. 1999. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall.

[2] Ware, C. and Bobrow, R. 2004. Motion to Support Rapid Interactive Queries on Node-Link Diagrams. *ACM Transactions on Applied Perception*, Vol. 1(1), 3-18.

[3] Wong, N. 2005. *EdgeLens: An Interactive Technique for Mitigating Edge Congestion in Graphs*. MSc Thesis, Dept. of Computer Science, University of Calgary, Calgary, AB, Canada T2N 1N4.

[4] Wong, N., and Carpendale, S. 2005. Supporting Interactive Graph Exploration with Edge Plucking. *Technical Report 2005-808-01*, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada, Aug.

[5] Wong, N., Carpendale, S., and Greenberg, S. 2003. EdgeLens: An Interactive Method for Managing Edge Congestion in Graphs. *Proceedings of the IEEE Symposium on Information Visualization (InfoVis 2003)*, 51-58.