Preserving the Hand-drawn Appearance of Graphs

Beryl Plimmer[1], Helen Purchase[2], Hong Yul Yang[1], Laura Laycock[1]

[1]Department of Computer Science
University of Auckland
Private Bag 92019
Auckland, New Zealand

[2] Computing Science Department
University of Glasgow
17 Lilybank Gardens
Glasgow, Scotland

ABSTRACT

When using a sketching tool to draw graphs, the edges need to appear hand-drawn. This is particularly the case after edges have been repositioned – if the action of moving a node results in its edges appearing as straight lines, the graph drawing will not retain its informal, hand-drawn appearance. The method for preserving the hand-drawn appearance of graphs described here is based on user observations and takes into account the context of the edge. The effectiveness of this algorithm was validated with a user study which suggests that people cannot distinguish the generated edges from hand-drawn edges.

1. Introduction

Using hand-drawn diagrams is a proven, easy and helpful technique in the early stages of design [6][12]. The process of sketching aids the communication of ideas, analysis of design, and creativity, while allowing alternative designs to be easily explored without concern for the cost of changes. Sketches are often used to create initial diagrammatic models of objects and processes: for example, UML diagrams, circuit diagrams and ER diagrams. Many of these diagrammatic forms are graphs that consist of a set of nodes with edges indicating the relationships between these nodes.

There are physical limitations to pencil and paper graph sketching. During design a diagram can often become convoluted and hard to understand as edges and nodes are added to the graph or are altered (Figure 1). To overcome this, the sketcher may need to go through a messy process of erasing and redrawing nodes and edges, or may need to restart the diagram altogether, as it is not possible to drag nodes around on paper.

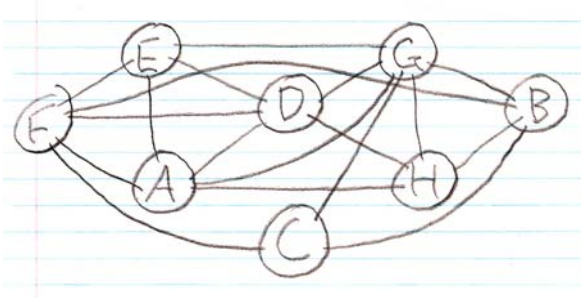

Figure 1: A hand drawn graph can quickly become messy without the ability to reposition nodes and edges

Computer-based sketch editing tools which allow the sketcher to reposition nodes manually by dragging them or which can apply an automatic layout algorithm can assist with the problem of tidying messy hand-drawn graphs.

Repositioning nodes in a sketched diagram introduces a new problem. How should an edge connected to a repositioned node appear after the node has been repositioned? This is the problem of *edge reflow*. Edge reflow is a common problem when editing a graph (e.g. [5]); here we consider the specific problem of edge reflow in when the graph is drawn using a sketch tool.

There are three particular challenges that need to be addressed when implementing sketch edge reflow:

(1) *The hand-drawn appearance of the diagram.* Hand-drawing is a simple and intuitive design process that is useful for brainstorming or communication. Crucial to the process of design is the hand-drawn appearance of a sketch. Sketch-editing graph tools should maintain the hand-drawn appearance of the graph, so as to best represent the creative ideas of the designer.

Designers' performance changes when working with designs of different visual fidelity [14]. Hand-drawn sketches permit designers to place emphasis on some areas while leaving others hazy and ambiguous; this helps exploration of alternative design ideas [6][12]. Bailey and Konstan compared a stylus based system against pencil and paper sketching and against Authorware [4]. They suggest that designers find hand-drawn designs most useful in the process of creation and that pencil and paper is most effective for exploring and communicating designs.

A sketched edge, once reflowed, should therefore leave the sketch with a natural hand-drawn appearance so that the advantages of sketching remain [14].

(2) *Intelligent reflow.* An edge does not exist in isolation: it is part of a graph of other nodes and edges. While previous implementations of graph sketch tools have used several techniques to retain the hand-drawn appearance of a reflowed edge (as described in Section 2 below), none have considered its context and its interaction with other visible elements of the diagram. For example, a reflowed edge in a graph should be adjusted so that does not pass through other nodes in the graph (Figure 2).

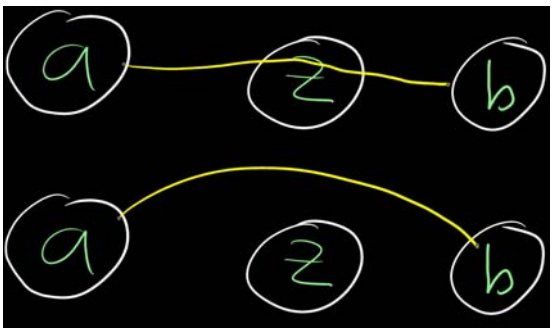

Figure 2: Intelligent reflow: The edge between a and b is best reflowed around z rather than going through it.

Such intelligent reflow is particularly important if automatic layout algorithms are to be applied to a hand-drawn diagram.

(3) *User-centered design and validation.* A reflow algorithm could be designed based on the intuitions of its designer. However, defining a reflow algorithm that does not take into account current sketching practices may result in a diagram that looks awkward or unnatural, or which follows conventions not typically used by sketchers.

A reflow algorithm is best designed after the graph drawing practices of sketchers have been observed. The results of this algorithm should be validated empirically so as to confirm that the reflowed edges are indistinguishable from hand-drawn edges, this proving its effectiveness.

In this paper, we present a new algorithm for determining the look of an edge after either its source or destination node has been repositioned in a graph sketch-editing tool. The advantages of this algorithm are:

- Its design was based on observations of users creating graphs;
- It is simple, being based on a library of hand-drawn edges;
- It preserves the hand-drawn appearance of an edge;
- It takes into account the context of the edge within the structure of the graph as a whole;
- Its output has been compared with hand-drawn edges, and its effectiveness validated with user studies.

Edge reflow is used in both the manual repositioning of a connected node and in the application of an automatic layout algorithm to the whole graph: the algorithm presented here can be used in both contexts.

2. Related Work

Various techniques have been explored to maintain the appearance of lines. Igarashi et al. [7] describe an 'As-rigid-as-possible' curve editing approach to hand-drawn line editing for cartoon characters. This process involved stretching a curve, by either scaling it to its new width or by a process of scale adjustment the curve acts as though it is a rigid object being pulled outward. These techniques are intuitive ways to morph an inflated graphic, but are not intended to preserve the hand-drawn appearance and may cause the curve to react in an unnatural way.

Arvo and Novins [2] explored edge reflow and preservation of sketch appearance within their blackboard style graphing system. They suggested techniques of edge reflow to preserve the hand-drawn appearance while also having the ink reflow in a predictable and intuitive manner. Their approach varied depending on whether: the new baseline (Figure 3) is shorter than the original baseline (compression); the new baseline is shorter than the original stroke length but is larger than the original baseline (stretch) the new baseline is longer than the original stroke length (over stretch).

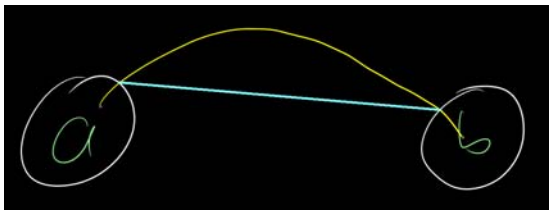

Figure 3: The straight line is the baseline of the arching connector stroke.

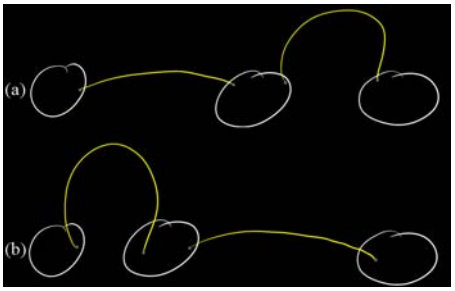

Figure 4: An illustration of the compression and stretch method [2]. When the centre node in drawing (a) is moved to the left, drawing (b) is created, with one connection stretched and interpolated with a straight line, the other compressed and interpolated with an ellipse.

Their metaphor is a piece of string. To compress the stroke points are interpolated with an ellipse (Figure 4a). To stretch a stroke (Figure 4b) it is interpolated with a straight line of the same length. The ratio of original stroke to straight line used for interpolation is decided by the amount of stretch: the greater the stretch the less the original stroke and the more of the straight line. This provides a smooth stretch and maintains the stroke length as though the stroke is a string being pulled and all the curves are slowly pulled out. When the total length of the stroke is less than the length of the straight line (over-stretched) the edge becomes perfectly straight. Arvo and Novins also presumed that it was important to maintain the relative position of the start point of an edge to the midpoint and boundary of the node and perform complex calculations to preserve this.

Reid et al [10] took a simpler approach to reflow maintaining more of the original appearance but avoiding straight edges. The stroke is rotated so its

baseline lies flat, then scaled to the appropriate length along the horizontal and the height halved if it is above a given threshold. The problem with this approach is that strokes when heavily compressed look unnatural and stressed (Figure 5).

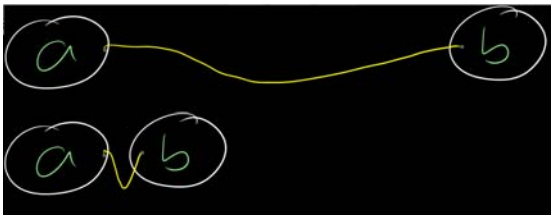

Figure 5: The top segment is the original; when node b is brought close to a, an unnatural looking edge is created

Ao et al. [3] considered appearance preservation in their network structure graph diagram sketching tool. When a container (node) is translated the associated connectors (edges) are moved with it. Two methods are reported: an uncomplicated scale to the new dimensions similar to [10]; and "stretching" by morphing of the stroke in a similar manner to [2]. This approach works in the majority of situations but in cases of extreme angles (elbows) it can cause loss of shape, in which case the scaling performs better.

All the above approaches are limited if we want to maintain the hand-drawn appearance in all situations: Arvo and Novins' [2] can result in perfectly straight edges while unnaturally curved edges occur in some cases with the Reid et al. [10] approach. Ao et al. [3] algorithms can result in both over-curved and over-straightened edges. Furthermore, none of these algorithms consider the context of the edge. This can result in edges cutting though nodes and intersecting with other edges; this compromises the hand-drawn appearance of the diagram.

3. Our edge reflow algorithm

3.1 Observational study

Before designing new algorithmic approaches to edge reflow we undertook a small informal observational study of how people create hand drawn graphs. Eight participants were asked to create two graphs each from a description and to then "tidy" the graph by repositioning the nodes.

Our observations of this process revealed that:

- When connecting two nodes, participants usually drew an approximately straight edge lying on the virtual line between the mid-points of the nodes, except when:
- (i) a third node lay on this virtual line (as between nodes p and o in Figure 6): in this case, the edge was routed around this node;
- (ii) there were two edges between a pair of nodes (as between nodes m and f in Figure 6): in this case, the edges were separated either side of this virtual line.
- Few participants could draw perfectly straight edges.
- The attachment points for an edge were close to the boundaries of the end nodes.
- People often crossed edges during construction but would eliminate crossings when tidying the graph. There were no incidences of edges crossing a node.

We did not observe any individual variation in edge style or the placement of node attachment points, and our observations and informal discussions with the participants suggest that they focus on the logical relationships between the nodes rather than the visual appearance of the edges.

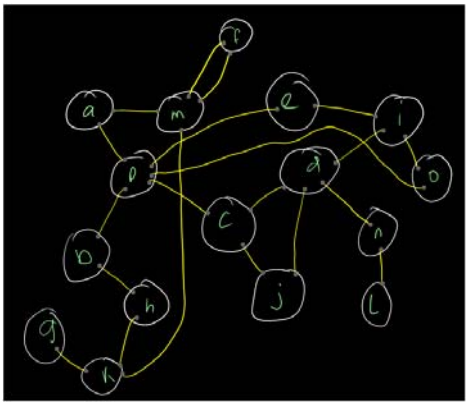

Figure 6*:* A typical initial construction of a graph

These observations formed the basis of the design of our new edge reflow algorithm to be used when nodes are repositioned either manually or automatically. Having observed no obvious user edge-drawing variations, the algorithm is designed to be generic, rather than specific to users' drawing styles.

3.2 Algorithm Design

Based on our observations, we suggest that, in order to maintain the hand-drawn appearance of the graph drawing, a reflowed edge should

- be a more-or-less straight edge (but not an exactly straight edge);
- lie approximately on the straight line between the centre points of the nodes.

This should be the case unless:

- there are multiple edges between a pair of nodes (in which case the edges should repel each other slightly), or
- the straight line between the nodes intersects with other nodes (in which case the edge should flow smoothly around the other nodes).

Our proposed solution is simple: when a node is repositioned, each of its connected edges is replaced by a generated edge.

The attachment points of these new edges are positioned at a small random offset from the point where the virtual straight line between two node centers crosses the node boundary, and at a small random offset within or outside the edge boundary.

Where there are multiple edges between a pair of nodes the new edges are repelled either side of the straight line and from each other so that there is visual separation.

Where the virtual straight line between an edge's nodes crosses other nodes, the generated edge is *flowed* around the node. Simply repelling the edge from the centre of the node results in an unnaturally curved edge with a flawless curve path. We explored a number of different functions to smooth the flow of the edge around a node, including a Gauss function [1], and a cosine function. We chose the cosine function (which we call *Context Reflow*), as it seems to give the most natural appearance, being not too sharp, and allowing multiple areas of force repulsion to be used without the stroke appearing too warped or disfigured. Figure 7 shows an example of each of these reflow approaches.

3.3 Implementation

There are four steps to reflowing an edge: the end points are established, a 'straightish' edge of appropriate length is generated, other node intersections are detected, and repulsion is used to push the edge away from other nodes.

The endpoints are established by identifiying the points of intersection between a straight line between the node centre points and the node boundaries (Figure 8, a and b) and then offsetting each by a small random amount.

The edge is generated from a small library of hand-drawn 'straightish' strokes of different lengths: this library was created and has been extended by many different users over the past two years. The new unique edge is generated from this library by random morphing between random pairs. The new edge is placed on the graph (Figure 8, edge ab).

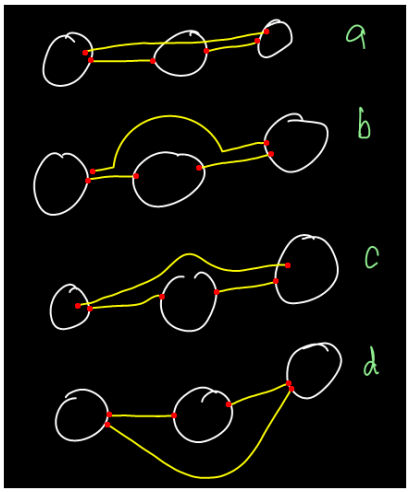

Figure 7: Node avoidance: a) no reflow, b) simple force, c) Gauss function force, d) Cosine function force (*Context Reflow).*

The new edge is then checked to see if it intersects with any other nodes. If there are no intersections the edge creation is complete. If intersections are detected the context reflow is applied.

The first step in the context reflow is to establish the exclusion zone. To do this all nodes that the edge intersects with are grouped and their bounding box found. The maximum distance for the displacement of the edge is calculated as the distance from the point on the edge nearest the midpoint of this exclusion zone to the most distant edge of the bounding box plus a padding value (Figure 8, c).

The maximum displacement is applied to this point. All other points are moved in relation to this point using formula (1). This allows the reflow to bend the edge over all the points in one arc. On a crowded graph the reflowed edge may cut through the corners of the bounding box; these cases are rare and cause few problems. A more sophisticated reflow would have to consider the individual bounding boxes of all the interfering nodes.

$$displacement \cdot Cos\left(\frac{\pi \cdot dist}{2\,max\,Dist}\right) \qquad (1)$$

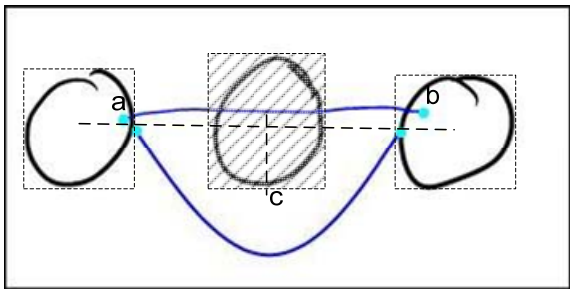

Figure 8: Context reflow: the edge is first placed between the connection points a b, and then reflowed around the bounding box of the intersecting nodes.

Where there are two edges between a pair of nodes the endpoints are pushed away from each other and the edges recreated between the new endpoints. This results in two 'straightish', and separated, edges between the nodes. Future enhancements to the system will include curving the edges away from each other.

4. Evaluation

Our aim was to produce reflowed edges that maintain the hand-drawn appearance of the graph drawing. Success can be determined by seeing whether the generated edges are indistiguishable from hand-drawn edges.

12 drawings of the same graph containing 10 nodes and 15 edges (referred to as D1-D12) were created by one of the authors. All the edges in D1 were hand-drawn (Figure 9). All the edges in D12 were system-generated, (Figure 10). The remaining D2-D10 drawings each had between 6 and 9 generated edges, with the remaining edges being hand-drawn (Figure 11).[1]

[1] A minor oversight meant that three of the drawings (D5, D6 and D7) were each missing one edge. This does not affect the validity of our experiment as these three edges are a small proportion of the total number of edges that each participant made judgements on.

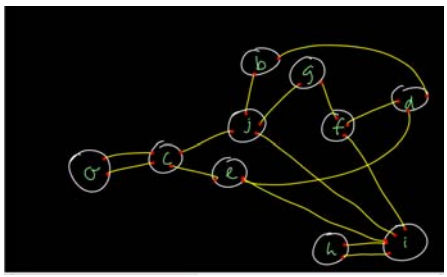

Figure 9: D1: All the edges are hand-drawn

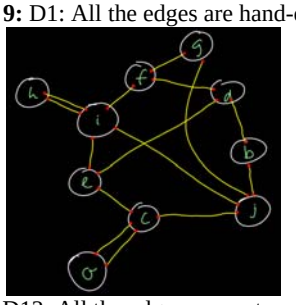

Figure 10: D12: All the edges are system-generated

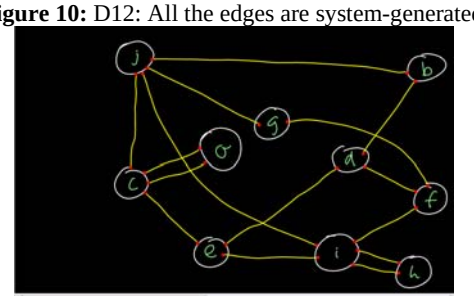

Figure 11: D8: Six edges are hand-drawn; nine edges are system-generated.

The following constraints were taken into account when creating these graph drawings:

- It is well-known that edge crossings can negatively affect participants' view and use of a graph drawing [8][9]. We did not want participants to be biased towards choosing edges that cross as the generated edges simply because they appeared awkward or anomalous. We therefore chose a non-planar graph which can be drawn with a minimum of two crossings. All 12 graph drawings had not more then four edge crossings: some of these were hand-drawn while some were generated.
- The nodes were all labelled, so that we could keep a record of which edges were system-generated. The same labels were used on all graph drawings, but different edges were chosen to be the ones that were system-generated.
- Some of the hand-drawn edges were deliberately drawn as curves (Figure 12).

Ten participants were shown all 12 graph drawings and asked to distinguish between hand-drawn and system-generated edges. They marked on the drawing with a pen, indicating those edges that they thought were system-generated with a C and those that they thought hand-drawn with a P. There was no time limit, and participants were encouraged to mark all the edges on all graph drawings.

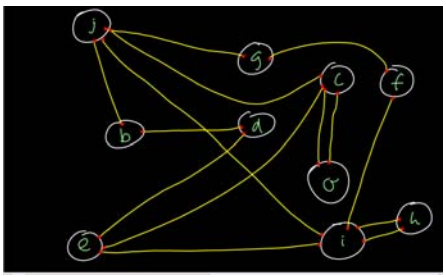

Figure 12: D5: The edge between nodes g and f has been hand-drawn.

The data collected was the number of correctly identified hand-drawn edges (HD), the number of correctly identified system-generated edges (SG), the number of hand-drawn edges incorrecly identified as system-generated edges (HDSG) and the number of system-generated edges incorrectly identified as hand-drawn (SGHD). Any informal comments made by the participants were also noted.

5. Results

Overall, 50.5% of the hand-drawn edges were classified as system-generated, and 52.6% of the system-generated edges were classified hand drawn.[2]

The data for the 'mixed-edge' drawings (D2-D11) was aggregated; the data for the 'control' drawings (D1, D12) was analysed separately (Table 1).

		D2-D11	D1	D12
HD (correct)	Mean	52.29	60.00	*NA*
	Max	75.00	86.67	*NA*
	Min	29.31	26.67	*NA*
HDSG (incorrect)	Mean	51.01	40.00	*NA*
	Max	75.41	73.33	*NA*
	Min	29.17	13.33	*NA*
SG (correct)	Mean	46.71	*NA*	49.63
	Max	59.72	*NA*	66.67
	Min	34.48	*NA*	26.67
SGHD (incorrect)	Mean	53.04	*NA*	50.37
	Max	65.52	*NA*	73.33
	Min	38.46	*NA*	33.33

Table 1: Mean, max and min percent of correct and incorrect classifications over all participants.

Paired t-tests were used to determine whether there was any significant difference between (a) the percentage of hand-drawn edges correctly identified (HD) and those incorrectly identified (HDSG) and (b) the percentage of system-generated edges correctly identified (SG) and those incorrectly identified (SGHD). In both cases, the probability of the classifications having been made simply by chance was high, and over the traditional p-value of 0.05 used for testing statistical significance (HD/HDSG: p=0.452; SG/SGHD: p=0.117). This proves that

[2] Although all participants were encouraged to classify all the edges, 10 of the 120 marked-up graphs had at least one missing label: these drawings were removed from the analysis.

participants could not distinguish between system-generated and hand-drawn edges, and that their decisions were as good as if they had been random.

We note, however, that the difference in the averages for the HD/HDSG comparison (1.28%) is less than that for SG/SGHD comparison (6.33%) by a factor of almost 5, suggesting that it was easier to correctly classify system-generated edges than hand-drawn ones. This observation is also shown in the narrower range between maximum and minimum percentages for identification of system generated edges than hand-drawn ones.

However, the two control drawings show that there was greater success with a completely hand-drawn diagram (mean 60.00%) than with a drawing with all edges system-generated (mean 49.63%).

The qualitative questionnaire data revealed that the following features were considered important for classification:

- The 'curviness' of the edge. Many participants said that curved edges were hand-drawn and straight edges system-generated.
- The 'kinkyness' of the edges. Smooth edges were considered as computer-generated, while those with 'bumps and curves' were classified as hand-drawn.
- The edge connectors. Edges with connectors close to the node boundary were labelled system-generated as 'the human dots are less accurate'.

General comments made it clear that the participants found this a very difficult task; e.g. "Nigh on impossible to make a decision. They all look the same", "For the majority of the lines, I just guessed."

6. Discussion

Our goal in this project is to retain the hand-drawn appearance of edges in a graph drawing sketching system when the manual or automatic repositioning of nodes requires that they be redrawn.

Our implementation of a *Context Reflow* algorithm is based on our observations of how people draw graphs. It is both simple and flexible, and in this paper we have demonstrated that its results are indistinguishable from hand-drawn edges.

Our results indicate further improvements to the edge-reflow algorithm; in particular, the placement of the edge connectors appears to be too precise to be comparable to hand-drawn edges.

We also need to consider how to adapt this algorithm to deal with several interfering nodes, rather than one. In addition, we are keen on investigating whether using a library of hand-drawn curved edges which may be morphed and adjusted at real time may be useful in edge reflow.

We have implemented other reflow algorithms and chose the *Context Reflow* one for this first empirical study based on our own intuition. An empirical comparison with the Gaussian method (Figure 7 (c)) and with enhanced versions of our *Context Reflow* method is needed in order to determine the best reflow algorithm for maintaining the hand-drawn appearance of the graph.

7. Conclusion

Many algorithms for edge reflow have been developed: the strength of the one reported here is that its design is based on observations of humans creating graphs, and that its success has been empirically validated.

Sketch design tools that do not preserve hand-drawn appearance do not fully utilize the stylus and interfere with one of the known advantages of pencil and paper. Our reflow algorithm is a validated approach to ensuring that the hand-drawn appearance of a graph is maintained after modification.

References

[1] Anton, H and R. C. Busby, Contemporary Linear Algebra, John Wiley & Sons, Inc, 2003.

[2] Arvo, C and K. Novins, Appearance-preserving manipulation of hand-drawn graphs, Graphite, ACM, 2006, pp. 61-68.

[3] Ao, X., Wang, X,. Jiang, W. and G. Dai, Structuring and Manipulating Hand-Sketched Diagrams, Sketch Based Interfaces and Modeling, Eurographics, Riverside, CA, USA, 2007

[4] Bailey, B. P. and J. A. Konstan, Are Informal Tools Better? Comparing DEMAIS, Pencil and Paper, and Authorware for Early Multimedia Design, CHI 2003, ACM, Ft Lauderdale, 2003, pp. 313-320.

[5] Dwyer, T., Marriott, K., and Wybrow, M. Dunnart: A Constraint-Based Network Diagram Authoring Tool. In GD 2008, pp 420-431.

[6] Goldschmidt, G., The backtalk of self-generated sketches, in J. S. Gero and B. Tversky, eds., Visual and spatial reasoning in design, Key Centre, University of Sydney, Sydney, 1999, pp. 163-184.

[7] Igarashi, T., Moscovich T, .and J. F. Hughes, As-Rigid-As-Possible Shape Manipulation, International Conference on Computer Graphics and Interactive Techniques, ACM New York, NY, USA, Los Angeles, California, 2005, pp. 1134 - 1141.

[8] Purchase, H, Carrington, D. and J.-A. Allder, Empirical Evaluation of Aesthetics-based Graph Layout Empirical Software Engineering, 7, 2002, pp. 233-255.

[9] Purchase, H., Which aesthetic has the greatest effect on human understanding? , Graph Drawing, Springer Berlin / Heidelberg, 1997, pp. 248-261.

[10] Reid, P., Hallett-Hook, F., Plimmer, B. and H. Purchase, Applying Layout Algorithms to Hand-drawn Graphs, OzCHI 2007: Entertaining User Interfaces ACM, Adelaide, 2007, pp. 203-206.

[11] Soni, BK., Thompson, JF., and N. P. Weatherill, Computer-Aided Geometric Design, Handbook of Grid Generation, CRC Press LLC, 1999.

[12] Tversky, B., What does drawing reveal about thinking, Visual and spatial reasoning in design, Cambridge, Mass., 1999, pp. 75-81.

[13] Ware, C. Purchase, H., Colpoys, L., and M. McGill, Cognitive measurements of graph aesthetics, Information Visualization 2002, pp. 103 – 110.

[14] Yeung, L., B. Plimmer, Lobb, B. Elliffe, D., 2008. Effect of Fidelity in Diagram Presentation HCI 2008. D. England. Liverpool, BCS. 1: 35-45.