An Empirical Evaluation of Force-Directed Graph Layout

by

Roman Klapaukh

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the requirements for the degree of
Doctor of Philosophy

Victoria University of Wellington
2014

Abstract

Force-directed graph layout is a widely used algorithm for the automatic layout of graphs. Little experimental work has been done exploring the behaviour of the algorithm under a variety of conditions. This thesis carries out three large-scale metric-based experiments. The first explores how the core algorithm behaves under changes to initial conditions. The second looks at extending the force-directed layout algorithm with additional forces to reduce overlaps. The third develops a novel symmetry metric for graphs and uses that to explore the symmetries of graphs. This thesis also carries out a user study to show that the differences reported by metrics in the graphs are reflected in a difference in user performance when using graphs for a free-form selection task.

Acknowledgements

I would like to thank my supervisors David Pearce and Stuart Marshall. I would like to thank Michael Homer, Paley Li, and Tim Jones who shared an office with me and helped me check both writing and ideas. I would like to thank the other people who helped me proof read work and sanity check ideas: Sophie Barclay, Thomas Kern, Ryan Chard, Samuel Hindmarsh, Daniel Gibbs, Aysser Al-Janabi, and Craig Anslow. I appreciate all the help and support I have received from my family and friends during the course of the PhD. And thank you to any people I have forgotten, and the other numerous people I have interacted with during my PhD who have helped me in various ways.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many different kinds of data can be represented visually by graphs. Automatic graph layout allows for the visualisation of systems ranging from social networks to mind maps. These graphs range in size from having thousands of vertices, to under 100 for a small network, or a small mind map. These ideas are supported by a wide range of real world programs and libraries intended for the layout of graphs such as Springy.js [1], GraphViz [2], and Prefuse [3]. Many of these systems implement the force-directed layout [4] as one of their options.

The force-directed layout algorithm simulates the graph as a physical system in order to lay out the graph. Edges, like springs, pull vertices together. Vertices, like charged particles, repel each other. The system uses friction to prevent dynamic equilibrium (i.e. perpetual motion) so it tends towards a fixed state. The result of the computation is then used as a visualisation of the graph. A user can interact with the algorithm while it is running without requiring modifications.

While force-directed layout is widely implemented and has been widely researched (discussed in Section 2.1.2 on page 11), there are still aspects which are not well understood. Previous work by Purchase [5] has challenged widely held beliefs such as that force-

directed layout promotes symmetry [4]. However, there has been little large-scale experimental evaluation of the algorithms, without which it is not possible to accurately predict the effect of changing input parameters on the final layout.

The more the parameter effects on the final layout are understood, the less guess-and-check is required in actually laying out a graph. The aesthetics of the final layout are an important factor in graph layout. Consider a user reading the example small graphs shown in Figure 1.1. The figure shows an extract from a social network where vertices contain the name of the person they represent. The graph is laid out in two different ways. In the layout on the right two of the vertices are overlapping, while on the left they are all drawn separately. In the layout on the right, it is hard to determine the names of the two overlapping vertices. In the left layout there is no such confusion.

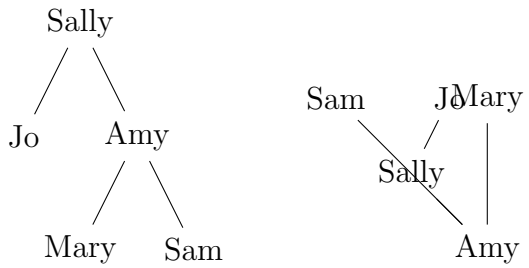


Figure 1.1: Two different layouts of a network.

Consider vertex overlaps as a simple metric. Historical assumptions about graphs have made even this metric difficult to optimise. Most of the core layout algorithms assume that vertices and edge labels take up no space when drawn [4, 6, 7]. This assumption is not upheld in many practical applications with small graphs. Furthermore, previous suggestions about how to resolve this issue have not been evaluated on a large-scale (e.g. [8, 9, 10, 11, 12, 13]). This thesis tests its hypotheses on large-scale data sets of small real-world

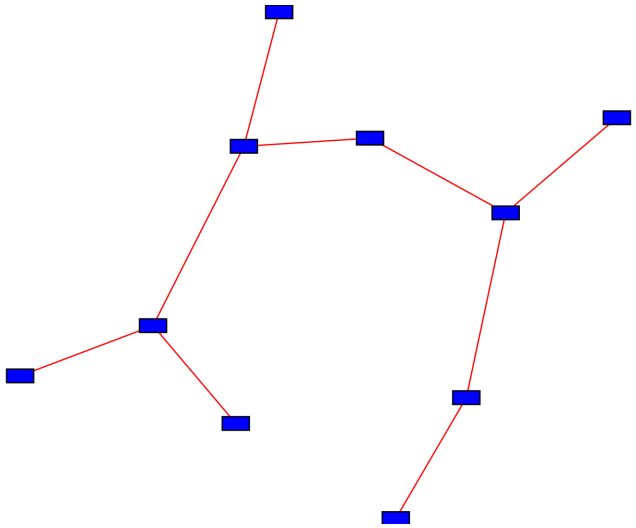


Figure 1.2: An example graph laid out as part of the experiments in this thesis.

graphs where vertices take up space.

To add evidence for the importance of overlaps this thesis attempts to gauge the effect of the reduced overlaps on human performance in selection tasks. This is done by means of a user experiment where users need to perform vertex selection tasks on graphs laid out in different ways.

This thesis looks at predicting the output of force-directed layout and shows that different outputs have different usability. It looks at changing the forces used to achieve reduced overlaps and edge crossings, how changes to the input parameters affect the layout, and how changes in the values of metrics are reflected in differences in user performance. This thesis carries out a large-scale experimental evaluation of force-directed layout across a real-world data set using a suite of metrics and user tests. An example graph laid out during the course of these experiments can be seen in Figure 1.2.

1.1 Contributions

The major contributions of this thesis focus on predicting the results of force-directed layout and showing that the predicted differences are practically useful. The four major contributions are:

1. Identifying the best force combinations to reduce edge crossings and overlaps in the final layout. This is determined by two large-scale experiments to evaluate the performance of different combinations of forces in force-directed layout.

2. Providing basic guidelines on how certain input parameters affect the final layout. These are determined by means of a large-scale experiment.

3. Showing a difference in performance between users using graphs laid out by a control force-directed layout algorithm and one of the best performing force combinations identified in the first contribution.

4. A novel symmetry algorithm for evaluating how symmetrical a graph layout is. It evaluates reflective, rotational and translational symmetries individually and has been extended to detect multiple axes of symmetry. The algorithm is evaluated with a user study and is used on the results of the parameter selection experiment to show that symmetry is affected by both the simulation granularity and the length of edges.

Work done as part of this thesis has been published previously [14]. This work was done with the assistance of the other named authors, but the implementation and the majority of the writing was done by the author.

1.2 Organisation

The remainder of this thesis is organised as follows. Chapter 2 on page 7 discusses the background and related work. It covers graphs, their layout, how to evaluate a layout, gesture controllers, and the technologies used to perform the large-scale experiments. Chapter 3 on page 35 discusses modifications to the force-directed layout algorithm to reduce overlaps in the final layout. Chapter 4 on page 59 discusses a large-scale study on parameters to force-directed layout, showing that parameter selection is an important part of the process. Chapter 5 on page 79 discusses a user study which demonstrates that the differences shown by the metrics in Chapter 3 are reflected in user performance. Chapter 6 on page 99 discusses novel extensions to a symmetry-evaluation algorithm allowing it to work with graphs. It uses this algorithm to extend the results from Chapter 4 to include symmetry, and performs a user study to compare the algorithm to human judgements of symmetry. Chapter 7 on page 129 concludes the thesis and explores avenues for future work.

Chapter 2

Background

This thesis focuses on graphs, their layout, and interactions. This chapter looks at what a graph is and different approaches to graph layout, with a strong focus on the force-directed graph layout algorithm. It then explores what makes a layout "good" and how those properties can be automatically measured and assigned a numeric value. The properties looked at in this chapter focus primarily on readability. It then looks at related work concerning human-computer interaction. Finally, this chapter examines the different high performance computing resources used in this thesis.

2.1 Graphs

A graph consists of a set of vertices and edges. Each edge connects exactly two vertices, or one vertex to itself. In most applications vertices are considered the objects, with edges representing connections or relationships. A wide variety of data can be represented in this way, such as social networks [15] and scene graphs [16]. In other cases, such as road maps, edges represent concrete objects (e.g. roads) and vertices represent their connections (e.g intersections).

This section now defines some terminology for graphs:

Directed Graph A graph where edges have a direction. In directed graphs edges are often drawn as arrows.

Undirected Graph A graph where the edges do not have a direction.

Labelled Graph A graph where edges have labels.

Path A path between two vertices in a graph is a sequence of edges such that the first edge starts at the first vertex, the last edge ends at the last vertex, and every intermediate edge starts where the previous one ended.

Connected Graph A graph is connected if there is a path between every pair of vertices.

Disconnected Graph A graph that is not connected.

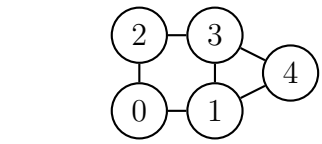
Simple Graph A graph that has no edges from a vertex to itself, and has at most one edge directly connecting any two vertices.

Degree The degree of a vertex is the number of edges connected to that vertex.

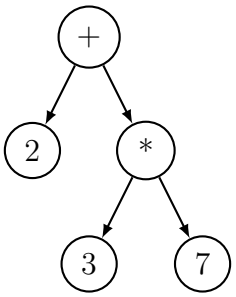
Planar Graph A graph that can be laid out with no edge crossings.

Graph Density The graph density is the proportion of edges the graph has compared to the maximum possible. For simple graphs the maximum possible is the number of edges it would have if every vertex had an edge to every other vertex.

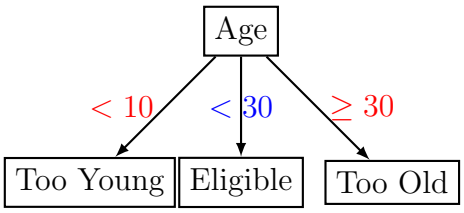
Some example simple graphs to illustrate these definitions can be seen in Figure 2.1 on the facing page.



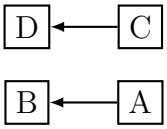
(a) An undirected graph. All the edges are lines.



(b) A directed graph. Note how the edges are drawn as arrows.



(c) A directed labelled graph.



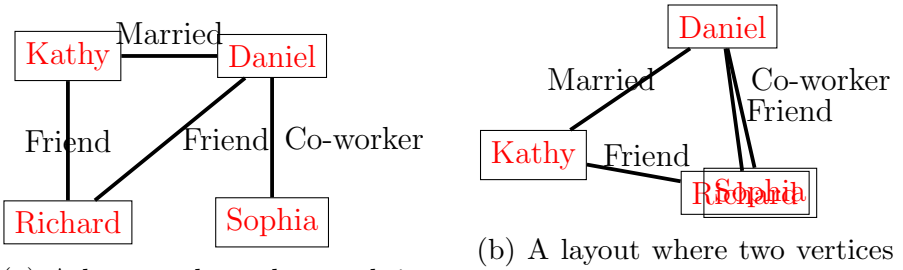
(d) A disconnected directed graph. There is no path that connects C and A.

Figure 2.1: Some example simple graphs.

2.1.1 Graph Layout

Many graphs, unlike maps, do not have inherent locations associated with vertices. The task of graph layout is to assign locations to the vertices, creating a useful visualisation. Identifying a bad visualisation is often easy for a human. Consider Figure 2.2 on the next page which shows the same graph twice but in one version has overlapping vertices. However, assigning a value to the quality of a graph layout is difficult for a human, and even harder algorithmically. Graph layout quality is discussed in detail in Section 2.2 on page 20.

There are a wide range of graph layout algorithms. In many algorithms there is an attractive relationship between vertices connected by edges, and a repulsive one between all vertices. If these equations are of the right form they can be simplified to an eigenvector problem [6] and solved directly. This is called spectral layout. If



(a) A layout where the graph is well spaced increasing readability.

(b) A layout where two vertices are overlapping decreasing readability.

Figure 2.2: A social network laid out with and without overlapping vertices. This illustrates one of the more obvious problems that can impact the readability of a graph layout.

they cannot be solved directly, solving the equations numerically with the Newton-Raphson method results in the Kamada and Kawai method [17]. However, this requires computing derivatives. Directly simulating the rules of the system incrementally results in force-directed layout (of which the original algorithm was Eades' Spring Embedder [4]) which is used in this thesis.

Alternatively, graph layout can be done with search techniques such as genetic algorithms [18]. A range of other automatic graph layout algorithms are discussed by Battista et al. [7]. Graph layout can be done manually [19], but this is difficult for large graphs due to the number of vertices and edges to consider. To complicate matters, some graphs require human input to get a good layout. Biochemical pathways (large graphs of reactions in organic chemistry) can be laid out automatically for small networks [20], but the algorithms cannot follow all the standard drawing conventions or deal with large networks well [21]. For this reason hybrid computer-human approaches have been developed such as those by Drost et al. [21] for metabolic networks, or by Yuan et al. [22] for general graphs.

Laying out a graph does not necessarily mean being able to see all the vertices and edges at the same time. When laying out large graphs

there are two main approaches. The first involves trying to find a layout that places all the vertices on the screen at once. This may involve adjusting vertex positions to unclutter dense areas [12, 23] or recursively drawing subgraphs [24, 25]. In contrast it is also possible to only draw the important features of the graph. This can be done by allowing the users to change which part and how much of the graph they are looking at [26, 27], or by grouping vertices with certain properties into single vertices [28]. For small graphs, it is possible to show all the data on the screen at once, reducing the need to use simplifying techniques. This is the approach taken in this thesis.

2.1.2 Force-directed Layout

The force-directed layout algorithm has seen wide use, being implemented in a number of real-world systems [2, 3, 1]. Additionally the lack of large-scale experimentation with graph layout algorithms means that there is no compelling reason to choose one layout algorithm over another. While force-directed layout is not the most computationally efficient algorithm, laying out small graphs is very fast, and it has found wide use in the layout of large graphs. For example, it can be used for the graph layout component of multi-dimensional scaling algorithms, which take higher dimensional data sets and render them in 2D [29]. GreenMax, which is used to visualise graphs with on the order of 1 million vertices, uses a multi-level graph layout algorithm with force-directed layout [26]. Multi-level layout involves merging edges and vertices to make a small graph, laying that out, then gradually adding edges and vertices back until the whole graph is laid out. Moreover, force-directed layout has a very simple formulation making it easy to modify, extend, and apply human physical intuition to.

The force-directed layout algorithm simulates the graph as a physical system (Figure 2.3 on the next page). Edges, like springs,

pull vertices together. Vertices, like charged particles, repel each other. The system uses friction to prevent dynamic equilibrium (i.e. perpetual motion) so it tends towards a fixed state. At every point of the computation every vertex has a position on the screen. These positions can be used as the graph layout. A user can interact with the algorithm while it is running without requiring modifications.

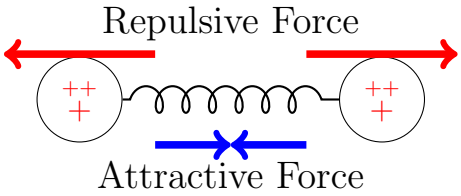


Figure 2.3: Basic forces in force-directed layout.

Various forces can be used to model attraction and repulsion. Almost all the papers surveyed during the course of this thesis used electrostatic repulsion (Coulomb's Law) as the repulsive force between vertices [4, 30, 7, 31]. In contrast, two different forces are used for the attractive force. Some papers, including Eades', use a logarithmic attractive force [4, 31, 32]. However, Eades describes edges as springs, so Hooke's Law (the physical law for ideal springs) is a natural alternative. It is not clear how they compare, though Battista et al. claim that from their experiences the logarithmic variant does not provide sufficiently better results given the extra computation to compute the logarithm [7]. Chapter 3 on page 35 experimentally compares both attractive forces to see if there is a difference.

The algorithm terminates either when a certain number of iterations have finished or when the total kinetic energy is low enough. The kinetic energy is a measure of the activity of the system. Given the mass (m) and the velocity (v) of a vertex, its kinetic energy is $\frac{1}{2}mv^2$. The total kinetic energy is the sum of the kinetic energy for

each vertex. Low values for the kinetic energy suggest that very little movement is happening so the algorithm can be stopped.

The general force-directed algorithm is shown in Algorithm 1. Note the use of the timestep parameter when calling the `move` method. This parameter is often omitted in algorithm formulations (which is equivalent to assuming that it takes the value 1). Some implementations treat the force directly as velocity, rather than using acceleration, to avoid the need for friction [30]. This simplifies the algorithm as fewer forces are required, but loses the flexibility that manual tuning of friction provides. Generating the initial layout involves placing all the vertices in a random location inside the allowed space, and setting all velocities to zero.

Algorithm 1: General force-directed layout algorithm.

```
generateInitialLayout()
for 0 .. maxIterations do
    totalEnergy = 0
    foreach Vertex v do
        tempForce = (0,0)
        foreach Edge {v,w} do
            tempForce += hookesLaw(v,w)
        foreach Vertex w do
            if v ≠ w then
                tempForce += coulombsLaw(v,w)
        tempForce += friction(v)
        tempForce += drag(v)
        v.move(tempForce, timeStep)
        totalEnergy += v.kineticEnergy()
    if totalEnergy ≤ energyCutOff then
        break
```

Forces

Each force has a particular function in the simulation with respect to the final layout desired. The algorithm requires a range of parameters to specify the strengths of the various forces. Eades' original work recommends a set of parameters that it found to work well [4]. Many later papers simply ignore the issue altogether [33, 10, 34, 31, 20, 26]. A small number of papers use a heuristic method to judge parameter selection [30, 17, 35]. An intuitive heuristic is to estimate the edge length the user wishes to achieve by assuming a system with only two vertices connected by a single edge. While this is easy to solve, and sounds reliable, there have been no experimental studies to show whether this distance actually predicts edge length when there are more than two vertices present. This thesis carries out this analysis as part of Chapter 4 on page 59.

The following sections go through each component of the algorithm and its associated parameters. x is used as the distance between two vertices.

Coulomb's Law Coulomb's law causes vertices to repel other vertices, resulting in the layout spreading out across the screen. The repulsion gets weaker as the vertices get further apart. It has two parameters that need to be decided in advance: the global strength of the repulsion (K_e), and the repulsion of each individual vertex (q_i).

$$F_{Coulomb} = K_e \frac{q_1 q_2}{x^2} \tag{2.1}$$

Hooke's Law Hooke's law is intended to introduce order into the layout. By attracting connected vertices it groups them together. If the graph is highly connected this can cause difficulties. However, in that case it is hard to know how to group things. Hooke's law

requires two parameters: the strength of the spring (K_h), and the natural length of the spring (N). The natural length of the spring is the length that the spring would be if there was nothing trying to stretch it. In this thesis this is twice the diagonal length of a vertex, so that the spring force will never try move vertices so close together that they overlap. While it is not necessary for all springs to have the same strength, in this thesis they do (K_h is constant for all edges).

$$F_{Hooke} = -K_h(x - N) \tag{2.2}$$

Logarithmic Spring Force (L) The logarithmic spring force serves the same purpose as Hooke's law, and is a common alternative used in Eades' original paper, and other work [4, 7, 20]. It requires two constants: the strength of the spring (k_l), and the natural length of the spring (N) as described for Hooke's Law. It has a behaviour similar in shape to Coulomb's Law, making it a logical, if more computationally expensive, alternative to Hooke's Law [7].

$$F_{logarithmic} = k_l \log\left(\frac{x}{N}\right) \tag{2.3}$$

Friction Friction serves two purposes. The first is to prevent settled vertices from starting to move again too easily. The second is to ensure that the simulation will eventually settle to a stable state. F_{static} is the amount of force that needs to be applied before a stationary vertex will start moving. $F_{kinetic}$ is the amount of friction that is applied while an object is moving. The formulas used are based on friction for dragging objects along the ground [36]. This requires the following constants: μ_s is the resistance to starting to move; μ_k is the friction while moving; m is the mass of a vertex; and g the strength of gravity.

$$F_{static} = \mu_s mg \tag{2.4}$$

$$F_{kinetic} = \mu_k mg \tag{2.5}$$

Drag Drag serves as an aid to friction. While friction is constant as an object is moving, drag affects vertices proportionally to their velocity. It has a parameter A,which represents the size of the vertex. v is the current velocity of the vertex.

$$D = \frac{1}{4} A v^2 \tag{2.6}$$

Newton's Law Newton's Laws of motion are used to convert the forces acting on a vertex to a displacement on the screen. An estimate of vertex mass is required, as for friction. These also require a timestep parameter (referred to as the simulation granularity earlier) to determine how fine grained the simulation is. The smaller the timestep, the more realistic the simulation, but also the longer it takes to run. Many papers, including Eades' original work, ignore this parameter [4]. This is equivalent to setting the value to 1.

This thesis explores the effect of timestep on the resulting graph layout, and attempts to evaluate the common effective choice of the value 1. It hypothesises that it is an important parameter that has a significant effect on the resulting layout. This is despite the fact that in terms of the conceptual model timestep has little effect. Note that a timestep parameter is needed even if Newton's laws are not used and force is converted directly to velocity.

The values v_i and d_i are the velocity and location at the start of each loop iteration. Similarly v_f and d_f are the velocity and location at the end of each loop iteration. In this thesis, initial velocities are zero and locations are random.

$$a = \frac{F}{m} \tag{2.7}$$

$$v_f = v_i + a * timestep \tag{2.8}$$

$$d_f = d_i + v_f * timestep \tag{2.9}$$

2.1.3 Extensions to Force-directed Layout

The original definitions used by Eades do not consider vertices as having any size, or edges as being labelled [4]. In most graph applications, vertices have a dimension to consider, as they are represented on the screen, and edges may be labelled. For the sake of simplicity, this thesis will refer to everything that is drawn except edges (but including edge labels) as images (e.g., text, raster images, etc). Having images leads to occluded pixels and overlaps. An occluded pixel is a pixel of an image which is covered by a pixel from a different image. An overlap occurs when two images are drawn so that one occludes some pixels of the other. This results in parts of the graph not being visible, reducing clarity and information retrieval from the visualisation, and should be minimised. Chapter 3 on page 35 focuses primarily on minimising occluded pixels and overlaps.

Some previous works on reducing overlaps are modifications to the layout algorithm. Wang and Miyamoto implemented modifications that cancel out attraction of occluded vertices, vary constants to account for vertex size, and integrate a constraint solver [35]. They did not do an experimental analysis, except to time the layout of a single graph, and to generate six figures for the paper. Harel and Koren claim that naïve extensions to layout algorithms to deal with drawn vertices often have negative repercussions. They proposed changes to the Kamada and Kawai method and modifications to the spring method [8] that they claim do not have these limitations, but

tested their ideas on only seven graphs. Kumar et al. reduce clutter by giving certain vertices a stronger repulsive force in directed acyclic graphs [31], but tested their algorithm on just two graphs. This thesis extends this approach to create its degree-based charge force (discussed in Section 3.1.1 on page 39). Lin et al. add vertex rotation to allow vertices to pack better [32], but this allows vertex images to end up at arbitrary angles, potentially decreasing readability. Additionally, their statistical analysis only contained six graphs.

Other works apply a post-processing step to 'fix' a layout. Force Transfer [13] and Force Scan [20] are two common algorithms to iteratively move vertices apart until they no longer overlap. Each of their experimental evaluations involved looking at just seven graphs. Frishman and Tal propose an algorithm to unclutter an existing layout [12], by mapping from the existing layout to one with a better information density. This algorithm is designed for huge graphs, where details on individual vertices are not visible, but was tested on just five graphs. Gansner and North use Voronoi diagrams to move vertex centres away from each other [34], and notably introduce curved edges. Their experiment consisted of only nine graphs. Dwyer et al. use constraint solving to spread the vertices [37]. They tested their performance on some randomly generated graphs, but only tested the layout quality on a single graph. The ePRISM algorithm uses a proximity stress model to move elements apart to remove edge and vertex overlaps from the layout [11]. After being used it requires a second post processing step with the PRISM algorithm. The evaluation consists of two graphs which are shown in the paper.

None of the works above performed any large-scale testing of their algorithms, with the largest test set containing just 12 graphs. This makes it hard to generalise their results. Of all the algorithms surveyed, only the ePRISM [11] algorithm explicitly considers edge labels, despite their common use in practice. Additionally, these

papers did not look at graph metrics or user studies to confirm the benefit of their systems. For this reason, this thesis performs a large-scale evaluation on 13,740 realistic graphs, many of which are anonymised graphs from AT&T. This thesis also includes a user study to confirm that the differences shown by metrics are reflected in human use.

2.1.4 Graphs Used

This thesis focuses on small graphs. While large graphs are interesting particularly with the rise of big data, small graphs are still an interesting area. In particular the availability of giant datasets does not reduce the number of small datasets there are. Additionally, small graphs are interesting from an interaction perspective, as the graph is small enough to be comprehended by a user.

This thesis uses graphs from the VAST challenge 2009 [38] and GraphDrawing.org [39]. All these graphs are simple graphs, and are treated as undirected graphs for the purposes of this thesis. Almost all the graphs are connected. Any directed graph can be laid out as an undirected graph and have the directed edges added after the layout, since there are no special considerations for edge direction. The VAST challenge dataset contains a fake social network. It contains 6000 individuals, their connections and cities of origin. All data is fabricated, and the place names are fictitious. It is represented by individual plain text files that specify the people (vertices), and their connections (edges).

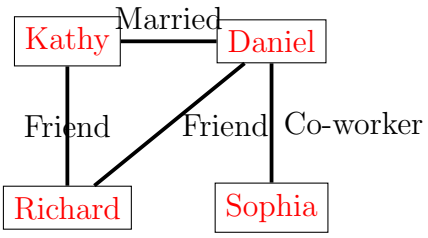
In contrast the GraphDrawing.org dataset is largely based on anonymised real graphs. It contains 1,277 anonymised graphs from AT&T (called the North graphs), 11,534 sample graphs from the GDToolkit [40] (called the Rome graphs), and 909 randomly generated directed acyclic graphs (called random-dag). Overall it contains 13,720 graphs. All the graphs are small, containing 10 to 110 vertices

and 9 to 241 edges. All but three graphs are connected (all three are in the Rome set) and 70% of the graphs are non planar (Rome: 72%, North: 33%, Random-Dag: 100%). The graphs span a variety of graph densities, from 0.86% to 85.45%. However, 75% of the graphs have densities less than 8.3%. In this dataset graphs are represented using GraphML, an XML-based format for describing graphs [41]. One of the experiments in this thesis uses a subset of this data set. 10% of the graphs were randomly selected to be included in this dataset. Figures showing various properties of the graphs of both the complete and sampled data sets can be found in Appendix A on page 135. Importantly the distributions of graph properties are similar between both the sample and the original.

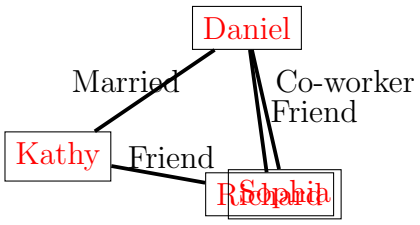
2.2 Graph Layout Evaluation

Simply laying out a graph is insufficient. It is important for the layout quality to be appropriate for the task. Consider the two social network graphs shown in Figure 2.4 on the facing page. Figure 2.4a is clear and allows the reader to identify all the relevant features. In contrast in Figure 2.4b vertices are overlapping, preventing them from being read. These overlaps are only an issue in small graphs, where there is enough space to actually show information on vertices. As this thesis looks at small graphs, overlaps are relevant to this work.

There are a number of existing metrics that are used to evaluate graphs [5, 42]. Moreover, graph layout quality has been shown to have an effect on how a user can and will use the visualisation [43, 44]. Purchase formally defined a number of these metrics [5]. These metrics quantify different properties of the graph, such as number of edge crossings, angle of separation between edges connected to the same vertex, or enforce certain rules such as keeping edges as straight



(a) A layout where the graph is well spaced, increasing readability.



(b) A layout where two vertices are overlapping, decreasing readability.

Figure 2.4: A graph showing a simple social network laid out with and without overlapping vertices. This illustrates one of the more obvious problems that can impact the readability of a graph layout.

lines, or promoting symmetry. Some metrics, such as maximising the similarity of edge lengths, involve solving NP-Complete problems [4, 45]. As there are many metrics, this thesis selects a small number on which to focus. The first is the number of overlaps, which is examined closely in Chapter 3. This thesis looks at this metric as it clearly impedes readability for small graphs where each vertex can be examined by the user. The second is the number of edge crossings. This metric is widely known, and is used by people performing manual graph layout [42]. Additionally, it is used as a gauge of how this thesis's changes affect other properties of the graph layout. Later experiments also measure symmetry to provide more evidence for or against the claim that force-directed layout produces symmetrical layouts [4], which has been experimentally disputed by Purchase [5].

2.2.1 Symmetry

One of the original claims made about force-directed layout was that it promoted symmetry, a trait that is believed to be helpful in reading graphs [4, 46]. Purchase defined a way of computing the symmetry of a graph [5]. This algorithm allowed her to experimentally compare

the symmetry exhibited by various graphs and layout algorithms. Her results showed that force-directed layout did not exhibit as much symmetry as expected relative to other algorithms. However, her symmetry algorithm has certain limitations. It considers only reflective symmetries, ignoring rotational and translational symmetries. It also considers only symmetries generated by the vertices, but, in connected graphs that are not trees, edges will outnumber vertices and be responsible for the majority of the structure that a user can see. For this reason this thesis develops a new symmetry metric for graphs, and looks at under which circumstances the force-directed layout generates symmetries in Chapter 6.

There are a number of symmetry detection algorithms for a variety of different tasks. Symmetries can be found for 3D objects [47], or images [48, 49]. This thesis is interested in finding symmetries of a 2D object with known structure. For this reason it extends the technique described by Loy and Eklundh which focuses on identifying symmetries between points representing objects of interest [48]. This makes heavy use of the Hough transform [50], a technique for uniquely identifying straight lines of infinite length (in this case an axis of symmetry). Their system represents objects of interest as scale-invariant feature transform (SIFT) features which have a position, orientation, location, and scale [51]. In the context of graphs, edges (as they are lines) are symmetric after a 180 degree rotation instead of requiring 360 degrees as in the original algorithm.

2.3 Human Computer Interaction

This thesis focuses on game controllers as the interaction medium as they are widely used. Interaction can be broadly divided up into 5 categories: Navigation, Selection, Manipulation, System Control and Symbolic Input [52, 53]. Since this thesis uses game controllers

to interact with small graphs, it focuses on selection tasks. As the graphs are small, navigation is assumed to not be necessary. This means that the whole graph is expected to fit on one screen. It also limits the results as it does not address the issue of navigation and selection gestures potentially clashing. Manipulation requires selection to identify what to manipulate. System control and symbolic input are not tasks that are generally done with a graph and so also are omitted.

In terms of types of gestures, this thesis looks exclusively at what Karam and Schraefel [54] classify as deictic (pointing) gestures. Deictic gestures are used under the assumption that they will be familiar given the wide spread of touch devices which use direct touch selection.

2.3.1 Multi-Touch Interaction

Graph interactions have been explored with multi-touch devices. Multi-touch devices can detect where a number of points are being touched on their surface. Schmidt et al. [55] created a set of gestures that facilitate viewing graphs by moving edges around. Kristensson et al. [56] developed InfoTouch which also enables graph exploration, but does it though vertex interaction. Dwyer et al. [19] compared manual graph layout using a multi-touch table against a mouse, with generally positive results in terms of user preferences.

Gesture controllers remove some of the limitations of multi-touch tables. Both multi-touch tables and gesture controllers have limited accuracy compared with a computer mouse [57]. However, this is only a problem if the error is on the same order as the spacing between and size of elements on the screen. In a multi-touch system, selecting (by touching) an element involves occluding the element with the user's hand. While there are some techniques to avoid this such as pointer offsetting, fish-eye lensing, or showing the covered area

specially [57, 58], gesture controllers allow a different approach by not requiring physical proximity. Additionally, multi-touch requires users to be able to physically reach the area they wish to interact with. With large displays this forced proximity to the device restricts the user's field of view, while the screen's size can limit the scope of interaction due to a user's limited reach. With gesture controllers the user can easily sit at a comfortable distance to see the whole display, and there are no issues relating to reach as it is based on angles, although this does cause problems with exaggerated movements.

2.3.2 Gesture-Based Interaction

There is a large bulk of work looking at interaction in virtual reality (VR) environments before commodity 3D tracking hardware was available. Historical research looking at selection tasks in VR environments looked at 3D environments, which encouraged the use of 3D tracking.

The simplest form of selection involves touching or framing the object the user wishes to select [59]. This is difficult to implement due to needing to accurately determine which eye the participant is looking through. If the wrong eye is used parallax may cause the wrong object to be selected. The ray casting technique solves this problem, having a user point at the target with a single finger [60]. This avoids the problem of working out which eye is being used, resulting in a unique point being selected (assuming only one finger is extended).

As much of this interaction was happening in 3D, some of the techniques have special characteristics to assist them. The Go-Go technique allowed users to grab objects with their hand, with the arm length growing faster the further it got from their body [61]. This allowed users to reach arbitrarily distant objects, as well as reaching occluded objects. Pausch et al. used miniatures of the world in front

of the user, like a doll house, where objects could be interacted with via their doll house versions [62]. The doll house was interacted with by direct touch. In this work the graph layouts are 2D and aim to avoid occlusion. Therefore selection is based on ray casting.

Using hand-held controllers introduces a few extra difficulties. Bowman et al. [63] found that pushing buttons on a controller causes the pointer to move, reducing accuracy. They called this the Heisenberg effect. Constant use of gesture interfaces can cause fatigue in users if the poses require users to keep their arms extended [64]. This is a particular concern in pointing systems such as the one described in this thesis, as users may decide to keep their arm fully extended while interacting with the system. In Chapter 5 the experiments have intentionally been kept short to mitigate this problem.

Gesture Controllers

Rather than using typical VR controllers, this thesis uses a controller that was already on the mass market and in common circulation to ensure that the work had a concrete application in the current state of hardware. Three gesture controllers have come out with games consoles on the mass market that were considered for this work: the Nintendo Wii remote, the Microsoft Kinect, and the Sony PlayStation Move. Due to their earlier release dates more work has been done with the Nintendo Wii remote and Kinect [65, 66, 67]. The experiments use the PlayStation Move [68] over the Nintendo Wii remote as the Move is more modern and has an accessible API to work with provided by Sony's Move.Me software [69]. This took care of tracking the controller at high speed using a PlayStation 3, removing any errors that could be from misuse of the tracking system. The Microsoft Kinect system was not considered as it provided no clear means either to disengage from the system, or to easily "click"

at a given coordinate due to its lack of buttons. This would make it complicated to switch between the different tasks the user would be doing (selecting, deselecting, no selection). It would also require a mechanism for selecting answer boxes on the question screens as there is no button to push when the cursor is over the desired option. Many Kinect games use a point and hold system, where the user puts the cursor over the option they wish to select and then holds it there. While this does work in practice, this was in conflict with the goal of making the experiment short to reduce fatigue.

The PlayStation Move system tracks the location and orientation of a special controller in 3D space using a camera. The Move controller (shown in Figure 2.5) has a sphere on the top which is uniformly illuminated by internal lights. Direct tracking of this glowing sphere combined with information from sensors within the controller allows it to be used as a mouse replacement, provided that the sphere can be kept within view of the camera.

Figure 2.5: A PlayStation Move controller. Note the sphere (on the left) which can be lit up in a wide range of solid colours.

There are two practical difficulties with the system. The first is that the camera must be set up with a clear view of the participant. This requires the participant be sufficiently far from the screen so that they fit in the camera's view. The second is that the camera has to be connected to a PlayStation 3 console in order to do the processing. The console then sends UDP packets to a listening computer over the network with the current status of the controller.

Alternatives The Wii remote was the first to be released in 2006 [70]. It has an internal infrared camera and a number of internal sensors. The Wii remote comes with a special bar containing infrared lights which the internal camera and sensors can use to work out its location and orientation. Due to its early release date it has generally lower tracking accuracy than the PlayStation Move. Additionally, unlike the Move, there is no official general use API for it.

In 2010 Microsoft released the Kinect sensor [71]. It is a commodity depth camera which provides a distance from the camera for each pixel as well as a colour. Using the depth information, it is possible to extract the pose of the user(s) interacting with the system, and use this information to control the system. In 2012 a second version was released for use with computers [72]. However, this does not provide an easy way to track simple pointing. The system tracks the whole body, and the low resolution camera has difficulty identifying the shape of a user's hand. This makes it difficult to detect interactions with the Kinect, as it cannot detect a gesture such as a closed fist reliably, while the Move can always detect button presses.

At this time there are also PC-specific controllers that have been released. The Razer Hydra [73], which uses magnetic tracking of two hand held controllers, does not have the same wide adoption as the afore mentioned controllers. The Leap Motion which uses two cameras to track a user's fingers was not released at the time this study was being conducted [74].

Game Controller Interaction

More recent work has looked at integrating commodity 3D tracking systems like the Nintendo Wii remote or Microsoft Kinect into the research field. The Wii remote has been used for a wide variety of tasks. It has been used to control robots [75], work out the orientation

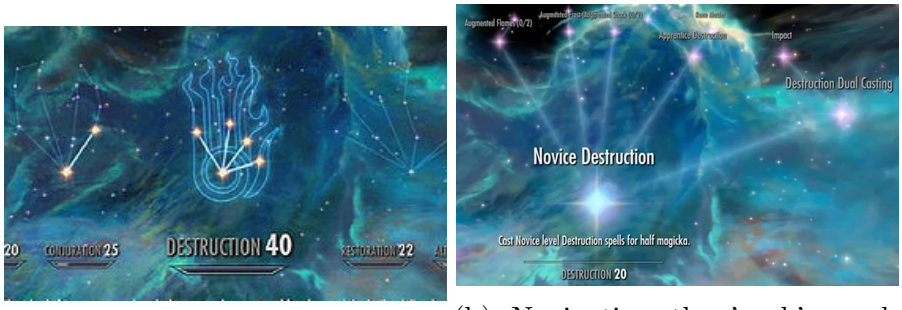
of a head-mounted display [76], validate the steering law [66], and control a football character in video games [65], among other things. In fact, using the Wii remote for interaction by pointing is essentially how the Nintendo Wii main menu works. The main difference is that their menu is a regular table, and a user simply needs to point at any spot within the cell they wish to select. In this work, the task the user must complete is more complex and it involves navigating around a set of obstacles without a regular pattern for the data.

Use of Gestures in Games These different devices have different modes of interaction in games. However, these interactions are based quite strongly around the game context, and do not necessarily generalise to other tasks. Consider two games for the Xbox 360 that use the Kinect: Kinect Adventures and Dance Central. In both these games the user's body is directly mapped to the body of the character they are playing. The user can make the character jump and move individual limbs by performing the appropriate action in the real world. In other games like Carnival Games: Monkey See, Monkey Do it may only track part of the body. But it is always based around a mapping from body position to input. This is particularly useful in medical contexts because there is no contact with any device, and so no risk of contamination [77]. However, the experiment described in this thesis does not consider interactions with data to be somehow mapped to body actions, and does not require the sterility conditions.

The Move and Wii games that that the author has seen either make heavy use of buttons and directional thumbpads on the controllers (Super Mario Galaxy, inFAMOUS 2), or they use the gesture controller to represent an object being held by the user (Sports Champions, Carnival Island). Both of these interactions are sensible in the context of a game, but less so for graph selection tasks.

There have been games that have required simple graph selection

using a D-pad. Bethesda Softworks' Elder Scrolls V: Skyrim (a single-player role playing game) requires users to navigate through a graph when getting 'perks' as part of levelling up their character. These graphs are laid out prettily (see Figure 2.6a), but when a user is navigating through them to find a vertex to select, the game zooms in so far that only a small selection of the vertices in any given graph can be seen (see Figure 2.6b). When played on the console the graph is navigated using a D-pad. This involves pushing the direction pad in the direction of the edge the user wants to follow. Pushing left or right will jump horizontally in the graph even if there is no connecting edge.



(a) Selecting a 'perk' graph.

(b) Navigating the 'perk' graph shown on the left.

Figure 2.6: Screen shots from the 'perk' systems in Bethesda Softworks' Elder Scrolls V: Skyrim.

This interface has a number of problems. The first is that there may be multiple edges going in a similar direction, and it can be hard to make the selection move in the right direction. If the user gets it wrong they can always jump left / right, but without seeing what is there. Consider Figure 2.6b, where "Novice Destruction" is currently selected. The user can, at this point, move left; however, there is no way of knowing from that screen which element will be selected as it is not currently displayed. Moreover, the user can actually jump from one graph to a different graph with no warning. In fact, in the

example before that is what will happen if the user were to move left. This thesis assumes that a pointing interface with the whole graph visible at once is easier to interact with, and therefore designs the experiments around such a system.

In this thesis the gesture consists of holding down a button (the trigger on the Move controller), while moving the controller around to enclose the desired objects in a bounding path. This sort of interaction is not sensible with a D-pad controller, where this would involve holding down a button, and using the D-pad controls to trace a path. The Move controller on the other hand makes this a natural interaction as it is similar to pointing, but one that is not common in video games.

Speed of Interaction

Research on the speed of interaction tasks has resulted in rules like the Steering Law [78] which predicts that the narrower and longer the path that a user needs to take for an action is, the longer the action will take. More recently this has been disputed for screen pointing tasks similar to the experiment in Chapter 5 using the PlayStation Move [79], and recent literature suggests that two part models which also consider gain are better predictors [80]. Gain is the ratio of the distance moved in the real world to the distance moved on the screen. For large displays such as televisions with gesture controllers, values of gain may vary dramatically from values when using a computer mouse with a standard monitor. The conclusion that narrower and longer courses are harder to navigate still holds, but with the added consideration that it is affected by how far a user must move their hand to move across the display.

Such a model predicts that gesture controllers should find navigating on a screen easy, as there are no boundaries the user needs to stay inside when moving between two options. This is also the case when

pointing at a single object in a graph. It is only when performing a more complex interaction, like lasso selection, that speed constraints for avoiding elements become a factor. This is the condition tested in the study described in Chapter 5.

2.4 High Performance Computing

This thesis performs large-scale experiments requiring many tens of thousands of trials. In order to make these experiments run within a reasonable amount of time they were run on various high performance computing platforms. The various platforms used in this thesis are described in this section.

2.4.1 Grid Computing

Grid computing utilises a large number of computers which split the task of performing a computation between them [81]. The only thing that grid computers require is a common mechanism to assign them tasks. Each task submitted to the grid is a program that runs and produces some output. This thesis uses two different types of grid: cycle-stealing and dedicated. Note that both grids are heterogeneous — not all of the machines have the same hardware / software.

Cycle-Stealing Grid

A cycle-stealing grid is one where the grid job is not the only task running on the computer (i.e. a physically present user may also use the machine). The machines used in this thesis run the Sun Grid Engine 6.2 to control the distribution of work [82, 83]. Specifically, the cycle-stealing grid contains approximately 250 machines running primarily Arch Linux, but also a small number of NetBSD machines. Many of the machines that make up the grid are the lab machines used

by the School of Engineering and Computer Science and the School of Mathematics, Statistics and Operations Research at Victoria University of Wellington.

In order to prevent grid jobs from affecting users physically using a computer, any job running via the grid will pause if it may disturb a physically logged-in user. Moreover, if a machine is restarted or shut down while a grid job is running, the job will be lost. Additionally to physically logged-in users, grid jobs compete for resources with users who have remotely logged into a particular machine. While this is unlikely to be a major factor, it may still affect a number of jobs.

Dedicated Grid

A dedicated grid is one where each computer only runs the tasks assigned to it by the grid. Specifically, the one used in this thesis runs the Open Grid Engine [84, 85]. It has a small number of high performance machines, each with a large number of cores and RAM, that are shared between all users of the system (details of the machines can be found online [85]). The only way to gain access to the machines is via the grid interface, reducing the amount of competing for resources that a job has to do as there is no unexpected load on the CPU or RAM, unlike the cycle stealing grid described above.

2.4.2 CUDA

This thesis used CUDA 5.0, a programming language for programming modern Nvidia General Purpose Graphics Processing Units (GPGPUs — graphics cards) [86]. Unlike a CPU which has around 1–8 cores, a GPGPU has hundreds (in this case exactly 240 [87]). This allows for many more computations to be done in parallel. This comes with an extra restriction: that all the cores need to be doing

the same task. Fortunately, force-directed graph layout is amenable to being programmed in a way compatible with GPGPUs.

Chapter 3

Layout Algorithm

Consider the example small graph laid out in two different ways shown in Figure 3.1. This shows an extract from a social network where vertices contain the name of the person they represent, and edges have the type of relationship between people. In the right layout two of the vertices are overlapping, while on the left they are all drawn separately. In the right layout, it is hard to determine the names of the two overlapping vertices, and to determine which edge is connected to which. In the left layout there is no such confusion.

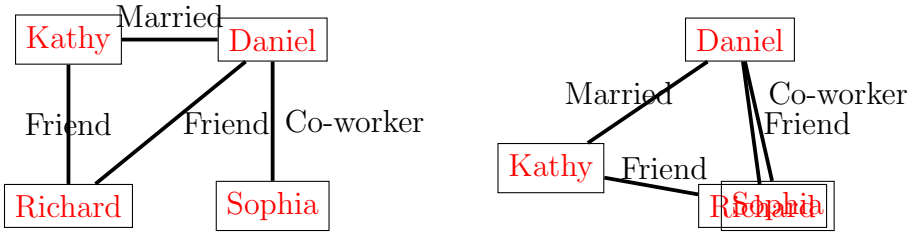


Figure 3.1: A good and a bad layout of a social network. In the right layout two of the vertices are overlapping and it is hard to read what they say or discern which edges connect to them.

This chapter attempts to reduce the number of overlaps in graphs laid out by force-directed layout. This is done by adding extra forces to the layout algorithm. This chapter considers a number of

additional forces, either by adapting forces from previous works, or by means of a novel force.

This chapter describes a large-scale experiment across 13,720 realistic graphs. The experiment measures how the numbers of overlaps and edge crossings changes as different combinations of forces are used. This experiment is followed by a second experiment to validate the results by testing every combination of forces on a random sample of 10% of the graphs in the first experiment. The graphs ranged in size from 10 to 110 vertices and come from a graph drawing benchmark containing real-world data from a major US corporation [39].

This chapter shows that:

1. Using charged walls, degree-based charge and charged edge labels reduce overlaps by an order of magnitude over the base algorithm.

2. Using Hooke's Law instead of the standard logarithmic attractive force tends to give layouts with fewer edge crossings.

3. The novel force — wrap-around forces — is not effective at reducing overlaps or edge crossings.

Overall this chapter finds that the force-directed layout algorithm without modifications performs significantly worse than the best modified version with respect to both overlaps and edge crossings. These findings have been previously published [14].

3.1 Algorithm Design and Variants

This chapter uses the general force-directed algorithm as described in Section 2.1.2 on page 11 as the basic algorithm. Additional details

are described below. The system was implemented using Java 1.6, and is available for download [88].

Initially vertices are assigned random locations. The natural length for the attractive forces is set as the minimum distance such that the two connected vertices do not overlap [20, 24]. Edge labels are placed one pixel away from the centre of the drawn edge, to prevent the edge hiding the label. All vertices have the same charge, shown as q_{vertex} in the parameters table (Tables 3.2 and 3.4). The plane boundaries in this system are an impenetrable barrier (bouncy walls) representing the edges of the screen to keep the vertices contained. Any vertex that hits a wall has its component of velocity in the direction of the collision reversed.

3.1.1 Variable Forces

This section describes the different forces that it experimentally investigated. Recall the basic configuration of forces from Chapter 2. Coulombs law, Hooke's Law (H), and the Logarithmic Spring Force (L) are the same as described.

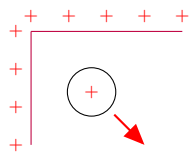


Figure 3.2: Charged Walls (W).

Charged Walls (W). Davidson and Harel proposed (but did not implement) charged walls as a mechanism for keeping vertices inside fixed boundaries [9]. Wall charge is just Coulomb's Law applied to a line the length of the boundary which will repel vertices from the walls as shown in Figure 3.2. This serves several purposes: it prevents unconnected components from moving infinitely far from each other; it prevents layouts settling close to the boundary where their ability to move is limited, impairing their ability to move into a minimal energy configuration; and it centres the resulting image. Wall charge in Tables 3.2 and 3.4 shows the charge of each wall as

used in this chapter.

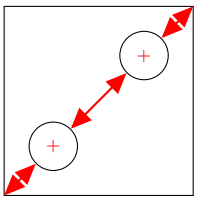


Figure 3.3: Wrap-Around Forces (A).

Wrap-Around Forces (A). Wrap-around forces allow the repulsive forces between vertices to act as if the layout is on a torus as seen in Figure 3.3. This does not change how walls affect the vertices, but Coulomb's Law calculations are performed in both directions. Note that the walls mean that vertices themselves cannot move past the edge of the screen, only their repulsive forces can. Given any two vertices, this force pushes them to be equal distances from each other both ways around the plane, which is hypothesised to group vertices together, and prevent disconnected components from drifting off. This chapter hypothesises that combining wrap-around forces with charged walls will centre all the graphs neatly. This is a novel force which this chapter tests.

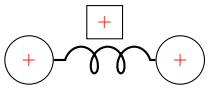


Figure 3.4: Charged Edge Labels (E).

Charged Edge Labels (E). Edge labels can be occluded by other images. To prevent that, charged edge labels makes labels charged in the same way that vertices already are as shown in Figure 3.4. This force is based on the obvious extension to force-directed layout where edge labels are treated as special vertices. Each label is charged just like a vertex (with magnitude q_{label}), and so repels vertices to which it is not connected. Labels are placed near the edge, but not over the top of it (as described earlier) which may cause interesting behaviour. Labels are unable to move independently so forces applied to a label instead affect the two vertices which it is connected to. This is analogous to pushing a bar with weights on it: it is the weights that roll along the ground, moving the bar with

them. This avoids moving the label independently, while allowing labels to affect the layout.

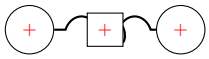


Figure 3.5: Charged Edge Centres (G).

Charged Edge Centres (G). Charged edge centres is an alternative to charged edge labels where the repulsion comes from the centre of the edge, rather than the edge label which is placed near the edge. This is shown in Figure 3.5. The force is exactly the same as charged edge labels, and each edge centre has charge q_{label}. Once again, edge centres cannot move independently and instead cause the vertices they are connected to to move instead. Note that only one of E or G can be active at any one time. Charged edge centres is not used in the first experiment, as it was only introduced in the second experiment in response to feedback from anonymous conference paper reviewers who were concerned about the labels being offset from the edge.

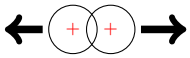


Figure 3.6: Collisions (C).

Collisions (C). Collisions between vertices is a trivial extension to force-directed layout as it is based on physical simulation. The collisions are ideal billiard ball collisions using the coefficient of restitution (Figure 3.6). Collisions are only implemented for vertices.

This implementation computes the velocity changes due to the collision and sets the new velocities appropriately. While it is possible to move the vertices apart until they are no longer overlapping, this is not implemented as there may not always be a sensible layout possible where there are no overlaps. Some graphs may have very dense regions, or so many vertices that allowing some amount of overlapping is actually beneficial.

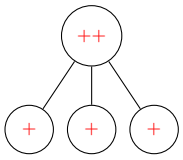


Figure 3.7: Degree-Based Charge (D).

Degree-Based Charge (D). Kumar et al. proposed to increase the charge on certain vertices to give them more space [31]. Their technique relies on the graph being a directed acyclic graph. This chapter generalises their technique to general graphs, giving high degree vertices higher charge. The higher the degree of a vertex the more space it needs, therefore the more it should repel other images as in Figure 3.7. Thus the system multiplies the standard Coulomb repulsion between the vertices by the maximum of 1 and $\frac{\text{degree}(v_1)*\text{degree}(v_2)}{4}$. It uses the product of degrees to increase the strength of the repulsion quickly, as the repulsive force falls off quickly with respect to distance. The original formula of Kumar et al. cannot be used directly as it requires directed acyclic graphs. The constant 4 is chosen for the denominator so that pairs of vertices of degree two are not affected.

3.2 Experiment One

An experiment was carried out comparing different combinations of forces to see how they affect the number of overlaps, and edge crossings. To measure overlaps, the experimental framework recorded both the number of overlapping vertices and the number of occluded pixels. It also recorded the time taken to lay out each graph, to see how the modifications affect performance, and the number of edge crossings as a secondary goal. Coulomb's Law is used as the repulsive force in all the experiments as it is used in Eades' original work [4], and makes intuitive sense as the relevant physical law.

Each layout was run until either the kinetic energy (a measure of the activity of the system) dropped below a given threshold (this was not expected to happen), or the maximum number of iterations was reached. Short names comprised of a letter for each active force

(excluding Coulomb's Law) are used in figures for compactness.

This experiment involved running each of the fourteen combinations shown in Table 3.1 once on each of the 13,720 graphs in the dataset. These combinations were chosen by manually testing to determine which looked the most promising.

Experiment	1	2	3	4	5	6	7	8	9	10	11	12	13	14
H (Hooke's Law)	✓	✓	✓	✓	✓	✓	✓							
L (Logarithmic Attraction)								✓	✓	✓	✓	✓	✓	✓
W (Charged Walls)		✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	
E (Charged Edge Labels)		✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓
C (Collisions)			✓							✓				
D (Degree Based Charge)				✓		✓	✓				✓		✓	✓
A (Wrap-Around Forces)					✓	✓	✓					✓	✓	✓

Table 3.1: Combinations of forces tested in Experiment One.

3.2.1 Data Set

The various algorithms were tested on a data set from GraphDrawing.org [39]. These graphs were chosen as they mostly contain graphs from real-world applications and are small - containing 10 to 110 vertices, and 9 to 241 edges. There are a total of 13,720 graphs. These graphs are described in more detail in Section 2.1.4 on page 19.

3.2.2 Choice of Constants

The constants for Experiment One can be found in Table 3.2 on the next page. The screen size grows with the size of the graph, but is limited to 8000 x 8000 pixels as beyond that it becomes too small to see clearly. This is based on the assumption that the larger a graph is, the more space it needs to be laid out. Image size is based on a personal social network context such that they are small, but still easily recognisable. The kinetic energy cut off was set so low that it was unlikely that the system would ever terminate from that

condition. This was done to try ensure that all graphs would run for a similar length of time, but wouldn't keep going if they did actually stop.

The constants were all initially set by looking at parameters used in papers that the author surveyed. They were then adjusted by trial and error, such that they looked reasonable on several data sets. Choice of constants is a difficult exercise, as small changes can have unforeseen effects. Many papers have simply ignored the issue entirely [7, 4, 20, 33, 34, 20, 31, 10]. A standard practice is to use a heuristic method similar to the one used in this chapter, such as optimising for a simple case [30, 35, 17].

Constant	Value	Constant	Value
K_e	50,000	Coeff. of Restitution	0.9
K_l	-60	E_k Cut Off	3
K_h	0.2	Edge Label Length	2 – 4 characters
q_{label}	1	Natural Spring Length	2 × vertex size
q_{vertex}	3	Dampening	0.9
Wall Charge	1000	Max Iterations	10,000
Vertex Mass	2	timestep	0.01
w, h	400px $\leq$ #Vertices $\times$ 100 $\leq$ 8000px		
Vertex Image Size	107x87 pixels		

Table 3.2: Constants used in Experiment One.

3.2.3 Architecture

As the experiment involved laying out a large number of graphs, the experiments were run across a range of machines using a grid as described in Section 2.4.1 on page 31. The experimental framework was not optimised for speed, but rather simplicity in order to mitigate programming errors. However, this came with a cost. Some of the forces, such as collisions, are implemented using a naïve algorithm, resulting in a much longer runtime than necessary. The framework

recorded the final embeddings of about half of the experiments to allow for manual validation. A random subset of the recorded graphs were manually inspected.

As the experiment ran on the grid, it had to use Java with the `-Djava.awt.headless = true` setting. This was due to not all machines running an X-Server, and the framework using classes that used X (this is to calculate the width of text labels, as this requires specifying a font to use, which references the X libraries). As it was a simulation (there was no image being rendered to a screen), an actual instantiation of the X-Server was unnecessary and that parameter allows Java to use the libraries for computation.

As the framework uses images, there is a wait for the image to be loaded. This is a problem as Java loads the image in a separate thread, so the image may not have been loaded before the framework tries to determine the image width. This entails the use of the `MediaTracker` class additionally to all the standard image loading classes which pauses until all images are loaded, and as such timings are affected. In hindsight, this feature of the program is unnecessary and should not have been present.

The experiment was done on the cycle-stealing heterogeneous grid running Sun Grid Engine 6.2 described in Section 2.4.1 on page 31. While this has disadvantages compared to the dedicated grid, it was the only grid the author had access to at the time. Because the layout tasks were competing for resources with user tasks the experimental framework could not get accurate results about how long the algorithm took to run. A number of grid jobs did not finish successfully, but they were not re-run, as fewer than 5 graphs were lost for any given combination of forces.

3.2.4 Results

R version 2.11.1 (2010-05-31) [89] was used to analyse the results of both experiments. The experimental framework recorded the number of pixels drawn, the total number of pixels, the number of overlapping images, and the number of edge crossings. The number of pixels drawn compared to the total tells us how many pixels were occluded and therefore how much is hidden. The number of overlaps gives an indication of how crowded the layout is. While a layout can have no overlaps and still be crowded, this chapter only looks at crowding that results in occluded pixels. Edge crossings were recorded as minimising them is considered to increase goodness [5, 42].

As the resulting data does not seem to follow a normal distribution, the Wilcoxon rank-sum test was used for significance testing as opposed to the t-test. For this reason this chapter reports medians rather than means [90]. This chapter holds that the difference is significant at 95% significance ($p < 0.05$).

The list of medians for each force in the first experiment is shown in Table 3.3 on the facing page. The entries in bold blue are minima. All reported values have been rounded to 2 decimal places.

With regards to overlaps, H and L ($p = 0.12$, 0.32 respectively) are not significantly different and the best equal performers are LWED and HWED. In terms of occlusion the best performer is LWED, though HWED is not far behind (difference of 0.01%). With respect to the proportion of edge crossings compared to an estimated upper bound (calculation by Purchase [5]), and also as a raw count, HWED is the best performer. It is interesting to note that all of the forces that use Hooke's Law (H) have fewer edge crossings than forces that use the logarithmic attractive force (L). An example graph generated using HWED can be seen in Figure 3.8.

Median	Crossings		Overlaps		Occluded Pixels
	#	%	#	%	%
H	40.00	2.26	13.00	0.20	1.21
HWE	38.00	2.27	3.00	0.04	0.11
HWEC	39.00	2.26	3.00	0.04	0.10
HWED	**34.00**	**1.98**	**1.00**	**0.01**	0.03
HWEA	80.00	5.34	85.00	1.95	16.46
HWEDA	73.00	4.75	53.00	1.28	12.21
HEDA	71.00	4.52	49.00	1.14	11.34
L	41.00	2.37	13.00	0.20	1.19
LWE	42.00	2.45	2.00	0.03	0.09
LWEC	41.00	2.46	2.00	0.04	0.09
LWED	38.00	2.23	**1.00**	**0.01**	**0.02**
LWEA	92.00	6.15	103.00	2.27	18.40
LWEDA	91.00	6.05	69.00	1.74	15.34
LEDA	89.00	5.75	63.00	1.58	14.30

Table 3.3: Medians by force combination from Experiment One.

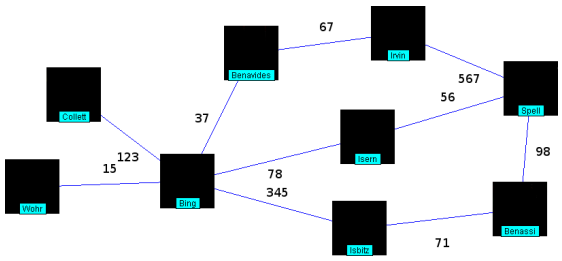


Figure 3.8: A graph produced by HWED. There are no overlaps or edge crossings allowing a human to see details on the vertices as well as the graph structure.

3.3 Experiment Two

The second experiment aims to address some of the limitations of the first. In the first experiment only a small subset of the possible force combinations were tested. This resulted in missing potentially good force combinations. To address this problem the second experiment tests all possible combinations for forces. However, due to the large number of combinations, not all of the graphs were used. Instead a random sample of 10% of the graphs was taken and used for all the tests (see Section 2.1.4 on page 19 and Appendix A on page 135). This experiment was run on a dedicated grid, allowing for a picture of the computation cost of the algorithm, which was not practical with the first experiment's data set.

Hypothesis. This experiment set out to confirm a number of hypotheses from the results of Experiment One.

$\mathcal{H}_1$ HWED and LWED are the best performing forces.

$\mathcal{H}_2$ Changing between Hooke's Law (H) and the logarithmic spring force (L) reduces the number of edge crossings.

$\mathcal{H}_3$ Switching between charged edge labels (E) and charged edge centres (G) has no effect.

3.3.1 Choice of Constants

In this experiment the width and height of the screen were fixed to 1920 x 1080 to mimic a normal screen, and the image size was changed to be square rather than rectangular. These changes are there to see if small changes to the setup would affect the relative performance of the best combination of modifications. The full combination of parameters is shown in Table 3.4 on the next page.

Constant	Value	Constant	Value
K_e	50,000	Coeff. of Restitution	0.9
K_l	-60	E_k Cut Off	3
K_h	0.2	Edge Label Length	2-4 characters
q_{label}	1	Natural Spring Length	2 × vertex size
q_{vertex}	3	Dampening	0.9
Wall Charge	1000	Max Iterations	10,000
Vertex Mass	2	Timestep	0.01
$w \times h$	1920 x 1080	Vertex Image Size	80x80 pixels

Table 3.4: Constants used in Experiment Two.

3.3.2 Architecture

This experiment was run on a dedicated grid running the Open Grid Scheduler using Java 6 under Red Hat Enterprise Linux 5 described in Section 2.4 on page 31. Running on a dedicated grid enabled the collection of accurate timing information, since the layout would not be suspended while running or competing for resources with unpredictable user programs.

3.3.3 Results

The list of medians for each force in the second experiment is shown in Table 3.5. The entries in bold blue are minima. All reported values have been rounded to two decimal places.

LD was the best performing combinations with respect to edge crossings. LWED was best in percentage of overlaps and occluded pixels, and best equal (with HWED) in count of overlaps. In 35 (72%) of the combinations of forces, using H rather than L resulted in fewer edge crossings.

Table 3.5: Medians by force from Experiment Two. The minimum value for each metric is highlighted in blue.

Median	Crossings		Overlaps		Occluded Pixels
	#	%	#	%	%
H	34.50	2.28	13.00	0.24	0.91
HA	276.00	18.20	375.50	7.61	80.38
HC	48.00	3.39	19.00	0.34	1.31
HCA	298.00	19.66	172.00	3.12	29.67
HD	31.00	2.09	10.00	0.22	0.82
HDA	294.50	20.57	411.50	8.61	83.47
HDC	32.00	2.25	10.00	0.22	0.83
HDCA	312.00	20.46	172.50	3.10	29.34
HE	38.00	2.72	6.00	0.13	1.16
HEA	95.00	6.94	16.00	0.32	5.72
HEC	41.00	2.87	8.00	0.18	2.02
HECA	55.00	4.18	23.00	0.46	5.76
HED	40.00	2.85	6.00	0.12	1.22
HEDA	176.50	12.02	128.00	2.55	45.02

Continued on next page...

Table 3.5 continued from previous page.

Median	Crossings		Overlaps		Occluded Pixels
	#	%	#	%	%
HEDC	41.50	2.89	6.00	0.14	1.57
HEDCA	93.00	6.51	41.00	0.72	8.96
HG	39.00	2.74	8.00	0.19	1.37
HGA	98.00	7.09	21.00	0.41	6.15
HGC	42.00	3.09	11.00	0.23	2.04
HGCA	60.00	4.57	28.00	0.54	6.15
HGD	40.00	2.88	9.00	0.18	1.38
HGDA	171.00	11.94	100.00	2.00	38.88
HGDC	43.00	3.14	10.00	0.21	1.68
HGDCA	99.00	7.11	44.50	0.77	9.34
HW	55.00	3.36	35.00	0.63	2.42
HWA	284.00	19.04	388.50	7.99	82.60
HWC	123.00	8.57	65.50	1.07	8.99
HWCA	306.50	20.66	177.00	3.34	30.74
HWD	43.00	2.62	22.00	0.40	1.54
HWDA	312.00	21.02	434.50	8.83	84.34
HWDC	48.00	3.46	24.50	0.51	1.74
HWDCA	323.00	21.09	176.00	3.33	30.69
HWE	34.00	2.17	4.00	0.08	0.33
HWEA	146.00	9.80	34.00	0.59	10.54
HWEC	39.00	2.74	9.00	0.18	1.48

Continued on next page...

Table 3.5 continued from previous page.

Median	Crossings		Overlaps		Occluded Pixels
	#	%	#	%	%
HWECA	83.50	6.08	44.00	0.81	10.31
HWED	34.00	2.22	**2.00**	0.05	0.13
HWEDA	192.00	13.20	165.00	3.37	52.25
HWEDC	38.00	2.59	5.00	0.11	0.69
HWEDCA	125.50	8.52	61.00	1.09	13.19
HWG	35.00	2.19	7.00	0.15	0.49
HWGA	143.00	9.94	40.00	0.70	10.62
HWGC	44.00	3.03	13.00	0.27	1.75
HWGCA	98.00	6.98	51.00	0.89	10.75
HWGD	35.00	2.25	6.00	0.13	0.39
HWGDA	197.00	12.89	137.00	2.71	46.65
HWGDC	40.00	2.87	9.00	0.20	0.95
HWGDCA	133.50	9.65	64.00	1.15	12.89
L	33.00	2.09	13.00	0.24	0.91
LA	266.50	18.08	369.00	7.42	79.89
LC	42.00	2.81	17.00	0.30	1.16
LCA	285.50	19.60	172.00	3.03	29.73
LD	**27.00**	**1.87**	9.00	0.20	0.78
LDA	292.50	19.94	418.50	8.40	82.26
LDC	29.00	1.98	9.00	0.20	0.75
LDCA	315.50	20.60	176.00	3.20	29.54

Continued on next page. . .

Table 3.5 continued from previous page.

Median	Crossings		Overlaps		Occluded Pixels
	#	%	#	%	%
LE	37.00	2.65	4.00	0.10	0.86
LEA	89.00	6.60	12.00	0.26	4.89
LEC	40.00	2.80	6.00	0.12	1.20
LECA	52.00	3.91	20.00	0.41	5.21
LED	39.00	2.77	4.00	0.08	0.73
LEDA	178.00	11.47	106.00	2.06	40.06
LEDC	40.00	2.87	5.00	0.10	1.18
LEDCA	95.00	6.36	40.00	0.71	8.80
LG	36.00	2.72	7.00	0.15	0.99
LGA	95.00	6.74	17.00	0.36	5.89
LGC	42.00	3.03	9.00	0.18	1.54
LGCA	60.00	4.45	24.00	0.50	5.34
LGD	39.00	2.85	7.00	0.16	1.01
LGDA	170.50	11.31	81.00	1.64	34.29
LGDC	40.50	3.02	7.00	0.17	1.22
LGDCA	100.50	7.27	43.00	0.76	9.11
LW	50.00	3.07	36.00	0.63	2.44
LWA	288.00	19.23	401.50	8.04	82.48
LWC	110.00	7.68	65.00	1.04	8.33
LWCA	311.50	20.93	179.00	3.32	30.71
LWD	35.00	2.28	19.50	0.38	1.41

Continued on next page...

...Concluded from previous page.					
Median	Crossings		Overlaps		Occluded Pixels
	#	%	#	%	%
LWDA	315.50	20.97	442.50	8.73	84.21
LWDC	40.00	2.85	22.00	0.45	1.57
LWDCA	322.50	21.50	179.00	3.37	30.78
LWE	32.00	2.10	3.00	0.06	0.25
LWEA	141.50	9.74	32.00	0.53	9.92
LWEC	36.00	2.53	6.00	0.13	0.90
LWECA	84.00	6.03	43.00	0.78	10.14
LWED	34.00	2.16	**2.00**	**0.04**	**0.09**
LWEDA	185.00	12.72	145.00	2.96	49.16
LWEDC	36.00	2.54	3.00	0.07	0.36
LWEDCA	118.00	8.56	61.00	1.08	12.97
LWG	33.00	2.11	6.00	0.14	0.41
LWGA	145.50	9.92	38.00	0.64	10.65
LWGC	41.00	2.95	10.00	0.22	1.15
LWGCA	96.00	6.72	49.00	0.85	10.63
LWGD	35.00	2.23	5.00	0.12	0.37
LWGDA	195.00	12.47	112.00	2.35	43.16
LWGDC	37.00	2.73	7.00	0.16	0.58
LWGDCA	133.00	9.51	66.00	1.14	13.11

3.4 Discussion

The LWED algorithm was the most effective at reducing the number of hidden pixels in both Experiment One and Two, and was best equal with HWED in reducing overlaps. It also fared well with respect to edge crossings in the first experiment, coming second equal. HWED had the least edge crossings in the first experiment and came second with regards to minimising occluded pixels. This partially confirms

$\mathcal{H}_1$ that LWED and HWED would be the best, with only LWED being the best.

All tests run with wrap-around forces (labelled A) fared poorly. This was a surprising result, as in a simple graph with two vertices connected by a single edge, this force resulted in the vertices spreading out. Upon running some additional simulations with larger graphs, it was found that while wrap-around forces did push all the vertices together, it did so too much, leading to poor performance. It seemed to perform particularly poorly on sparse non-planar graphs.

Hypothesis $\mathcal{H}_2$, that changing from using Hooke's Law (H) to the logarithmic spring force (L) generally increases the number of edge crossings, was not confirmed. It is true for 72% of the combinations, however, the force that minimised edge crossings the most is LD.

To test $\mathcal{H}_3$, this chapter compares the difference between charged edge labels (E) and charged edge centres (G). Table 3.6 shows the difference between the mean percentage of crossings and percentage of overlaps between each force that contains one of these forces. Each table entry is the difference in percentages, labels - centres (E-G). Even the maximum difference is below 1%, a value which is sufficiently low that there is little practical difference between the two options either in terms of edge crossings or in terms of overlaps. Note that in almost all cases the values are positive for edge crossings and negative for overlaps. This implies that having the charges on the centre of the edges reduces edge crossings, having the charge slightly offset reduces overlaps, though not by enough to matter in either case. This suggests that charged edge labels (E) and charged edge centres (G) are more or less interchangeable as expected. Note that in the best performing algorithms charged edge labels (E) was always used in place of charged edge centres (G) (as it was used to reduce overlaps, neither was used when minimising crossings).

Using the more consistent results from the second experiment it

Force	% Crossings	% Overlaps
LW[E|G]DC	0.51	-0.09
L[E|G]	0.08	-0.05
HW[E|G]	0.03	-0.06
HW[E|G]CA	**0.95**	-0.12
HW[E|G]DA	-0.19	0.60
LW[E|G]CA	0.79	-0.08
H[E|G]A	0.11	-0.08
L[E|G]D	0.04	-0.06
L[E|G]C	0.42	-0.06
HW[E|G]C	0.59	-0.09
HW[E|G]DCA	0.82	-0.08
H[E|G]	0.05	-0.05
LW[E|G]A	0.22	-0.11
LW[E|G]DCA	0.70	-0.07
LW[E|G]C	0.57	-0.08
L[E|G]DCA	0.66	-0.06
H[E|G]C	0.46	-0.06
HW[E|G]A	0.07	-0.07
LW[E|G]DA	-0.24	**0.62**
LW[E|G]D	0.02	-0.07
HW[E|G]DC	0.47	-0.09
L[E|G]DA	-0.21	0.51
LW[E|G]	0.03	-0.07
H[E|G]DCA	0.56	-0.07
H[E|G]D	0.09	-0.06
H[E|G]DC	0.44	-0.06
L[E|G]DC	0.33	-0.06
L[E|G]A	0.07	-0.08
H[E|G]CA	0.58	-0.07
HW[E|G]D	0.00	-0.07
L[E|G]CA	0.63	-0.07
H[E|G]DA	-0.18	0.56

Table 3.6: Differences between mean percentage of crossings and overlaps when using charged edge labels (E) or charged edge centres (G). The maximum value for each column is highlighted blue and bold.

was possible to measure the runtime required for graph layout and compare that between different combinations of forces. The average runtime for a single iteration of the algorithm for graphs of different sizes can be seen in Figure 3.9 on the next page. The single iteration runtime is averaged over the 10,000 iterations done in laying out each graph. Each data point in the figure is a single graph being laid out by a single combination of forces.

There are four forces which visibly affect program runtime - charged edge labels (E), charged edge centres (G), wrap-around forces (A) and collisions (C). The figure shows that the charged edge labels / centres (E/G) forces incur a clearly noticeable time increase. This is due to this modification requiring an extra inner loop running over all the edges. The presence of both collisions (C) and wrap-around forces (A) increases the run time more than having both independently would suggest. This chapter hypothesises that this happens because the wrap-around forces (A) force promotes collisions, creating more work for the collisions (C) force. This is also coupled with a highly inefficient implementation of all the forces, slowing down the overall system.

Validity The most notable omission of the experiments is that this chapter does not explore how different combinations of constants affect the output of the program. Both experiments in this chapter use an expected edge length of 165, which is not so long as to cause problems. They use a timestep of 0.01. Based on the results of the following chapter, that should provide minimal variability in the results collected.

While this study used a large test set of graphs, they do not encompass all possible graphs. They span a variety of graph densities, from 0.8608% to 85.45%, though 75% are less than 8.3%, and all but three are connected. Nevertheless, this chapter hypothesises that the

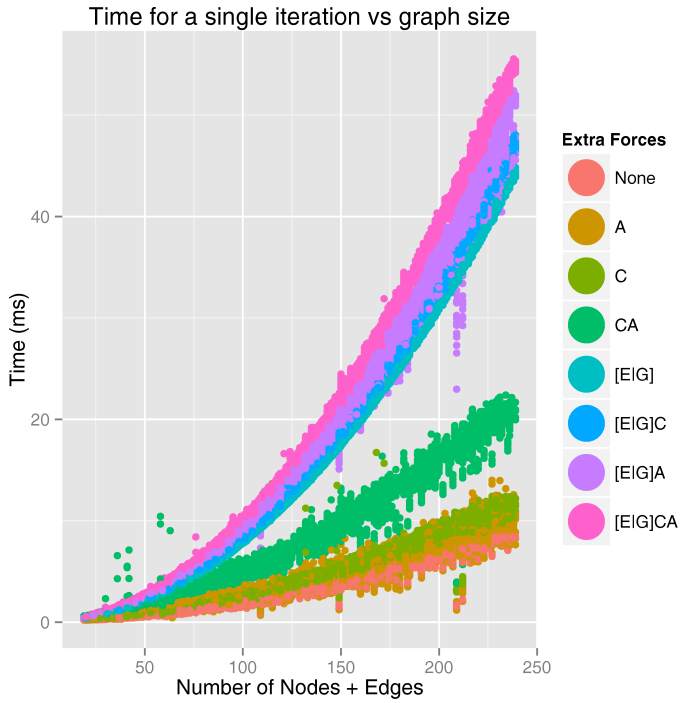


Figure 3.9: Time for an iteration vs number of vertices and edges in the graph coloured by forces which reduce performance. Times are an average over the whole runtime of the program with all combinations of forces. There are three well defined strata. The top one consists of combinations which contain charged edge labels / centres (E / G). Within this there are further divisions where either collisions (C) or wrap-around forces (A) are also present. The second stratum consists solely of combinations containing both collisions and wrap-around forces (CA) in the force. The final stratum shows that wrap-around forces (A) and collisions (C) slow down the algorithm, and having none of charged edge labels / centres, collision, or wrap-around forces (G,E,C or A) is the fastest.

set of graphs tested here is representative of many small real-world graphs as it is sampled from a real data set. This chapter hypothesises that changes and variances in image size can be accounted for by linked changes to physical constants for each vertex. This is supported

by the second experiment in this chapter where images were smaller, but the best performing forces were the same.

3.5 Conclusion

This chapter looked at modifications to the force-directed algorithm to avoid overlaps in small graph layout. It described a range of different forces which can be used with layout algorithms. Two experiments were carried out to find which set of forces would produce a layout with the least number of overlaps and occluded pixels. The experimental results show that:

- Adding in edge label charges, charged boundaries, and increasing vertex charge proportionally to its degree results in layouts that minimise the number of overlaps and occluded pixels.
- Using Hooke's law reduces the number of edge crossings for most force combinations.
- While there is a time cost for using charged edge labels, it is not so high as to make the modification too expensive.
- There is little practical difference between putting edge charges next to the edge or on the centre of the edge.

As a result, this chapter concludes that adding in additional forces is a viable way of preventing occluded pixels for graphs with large vertices and edge labels.

Chapter 4

Parameters

Recall the force-directed layout algorithm from Section 2.1.2 on page 11. The algorithm requires a range of parameters and an input graph in order to create a layout. The effect of input parameters on the resulting layout has not been sufficiently explored in the literature to allow users of the algorithm to predict how their choice of parameters is likely to affect the layout properties.

This chapter describes an experiment which compares how different sets of parameters affect a variety of graph layout properties. The experiment considers only the basic force-directed graph layout algorithm. The experiment varies the strength of spring attraction (Hooke's law), vertex-vertex repulsion (Coulomb's law), the simulation granularity (timestep), and whether there are bounding walls or not. Bounding walls can be used to ensure the graph is laid out entirely on the screen. If bounding walls are not used, the image has to be be scaled to fit the screen. There are two simple ways of scaling the graph. The first is shortening all the edges until the graph fits on the screen. This effectively increases the size of all the vertices, potentially creating lot of overlapping vertices and very short edges. This may have a significant affect on how the layout looks. The second is to scale the whole layout. However, this can

make the vertices so small that they are no longer visible.

This chapter divides the forces into two categories. Forces which motivate the vertices to move, and forces which motivate them to stop. The attractive and repulsive forces are the forces which cause movement. Other forces only cause the vertices to stop moving, and on their own will not cause vertices to change their positions. Therefore, they may have less effect on the final layout. For this reason, this chapter focuses on the parameters for the attractive and repulsive forces.

Walls are included to measure the effects of constraining the layout to the inside of a box, which seems reasonable to do given that the layout is often happening on the screen. The timestep parameter is a stand in for the forces which cause vertices to stop. This chapter assumes that its importance will be an indication of the importance of these stopping forces. This assumption is based on the author's experience, where changes to forces like friction seemed to have similar affects to changing timestep.

Overall this chapter evaluates 250 combinations of four input parameters over 13,720 graphs each. While more parameters and combinations could have been evaluated, there are already 3,430,000 data points in this experiment which require a large amount of computation to evaluate.

Contributions The main contributions of this chapter are:

1. A large-scale experiment that evaluated the effects of input parameters on a range of graph metrics.

2. Demonstrating that the mean edge length in a graph is not accurately predicted by a simple two vertex graph.

3. Demonstrating that increased values of timestep result in less predictable graph layouts.

4. Showing that there is a relationship between the mean edge length in a graph layout and the mean distance between vertices for all graphs.

5. Showing that as the mean edge length increases, the area occupied by the graph layout as a whole increases. The number of vertices in the graph is not seen to have a significant effect. This relationship is suggested to be non-linear.

As a result of this research this chapter shows that the simple 2-vertex graph does not provide very much information about the final layout. In general a large increase in the predicted edge length will often result in only a small change in the actual mean edge length. This chapter also shows that the simulation granularity affects the probability of the graph layout algorithm not converging to a good solution, resulting in considerably more variable results. Walls seem to have little effect beyond imposing some limits on how long edges and distances can be.

4.1 Experiment

The experiment was run over the GraphDrawing.org set of graphs. This includes 13,720 real world graphs. To restrict the scope of the experiment this chapter considers only four parameters: spring attraction, vertex-vertex repulsion strength, timestep, and walls. Together, the spring attraction and vertex-vertex repulsion can be combined to give an *Expected Edge Length* (EEL), which is the edge length as calculated analytically for the two vertices connected by a single edge.

Hypothesis: This chapter sets out to test the following hypotheses:

$\mathcal{H}_1$ The expected edge length is linearly related to the mean edge length in the resulting layout.

$\mathcal{H}_2$ Adding in walls has no effect on any of the metrics tested beyond limiting how much space the graph can occupy.

$\mathcal{H}_3$ The timestep parameter, serving as a proxy for the forces which cause vertices to stop moving, will have no effect on any of the metrics tested.

$\mathcal{H}_4$ Edge crossings are independent of the expected edge length, timestep, or the presence of walls.

4.1.1 Parameters

The four parameters looked at in this chapter are: K_h, the strength of the attractive force; K_e, the strength of the repulsive force; timestep, the granularity of the simulation; and the presence of walls.

K_h and K_e are the only values necessary to work out the expected edge length. This is done by solving the equation setting Hooke's law equal to Coulomb's law using the definitions given in Section 2.1.2 on page 14. The result is how long an edge should be if the graph has exactly two vertices connected by a single edge.

This chapter tested every combination of five values of K_h and K_e against five values of timestep and having walls present or not. This gives 250 combinations and 9 distinct EEL values. The values of parameters tested can be seen in Table 4.1 on the facing page. The mapping of different K_h and K_e values to expected edge lengths can be seen in Table 4.3 on page 64. The experiment used values across a range of orders of magnitude to evaluate a wide range of values. The values were picked such that the values used in laying out tests graphs successfully were included in the range.

Since this chapter only varies four parameters, there were a number of fixed parameters. All the remaining parameters had their values fixed across all the trials and have their values listed in Table 4.2 on the following page. While not varying these parameters limits the results, this chapter could not cover all the parameters due to limitations in time and resources. It is likely that these parameters have an affect on the final layout, and it is assumed that timestep will provide an indication of how important they may be.

K_h	K_e	timestep	Walls
0.0005	0.5	0.1	Bouncy
0.005	5	0.3	None
0.05	50	0.5	
0.5	500	0.7	
5	5000	0.9	

Table 4.1: Variable experimental parameters.

4.1.2 Test Data

This chapter used three test data sets from GraphDrawing.org [39]. It used these graphs as they mostly contain graphs from real world applications and are small - containing 10 to 110 vertices, and 9 to 241 edges. There are a total of 13,720 graphs, of which all were used for each combination of parameters. Additional discussion can be found in Section 2.1.4 on page 19.

4.1.3 Architecture

The graph layout algorithm was implemented using CUDA 5.0 [86], to enable the layout of multiple graphs simultaneously providing they use the same parameters. The implementation is open source and available online [91]. The layout was run on a headless 64 bit Ubuntu

Parameter	Value	Parameter	Value
E_K cut off	3	Max Iterations	10,000
Screen width	1920 px	Screen Height	1080 px
Vertex width	20 px	Vertex Height	10 px
μ_s	0.3	μ_k	0.04
Vertex Mass	1	Coeff. of Restitution	0.9
A	0.2	g	9.8
Max Vertex Speed Y	540 $\frac{\text{px}}{\text{iter}}$	Max Vertex Speed X	960 $\frac{\text{px}}{\text{iter}}$
q	3		

Table 4.2: Constant experimental parameters.

K_e	K_h	EEL	K_e	K_h	EEL
0.5	5	1	0.5	0.0005	26
0.5	0.5	3	5000	5	26
5	5	3	5	0.0005	56
5	0.5	6	50	0.005	56
0.5	0.05	6	500	0.05	56
50	5	6	5000	0.5	56
5	0.05	12	5000	0.05	122
0.5	0.005	12	50	0.0005	122
500	5	12	500	0.005	122
50	0.5	12	5000	0.005	262
500	0.5	26	500	0.0005	262
5	0.005	26	5000	0.0005	565
50	0.05	26			

Table 4.3: K_e (repulsive force) and K_h (attractive force) values that make up each EEL value.

12.04.3 system with an NVIDIA C1060 Tesla card [87]. Each set of parameters was evaluated on all 13,720 graphs as a single CUDA kernel invocation.

A separate Java program was written (and released open source) to evaluate graph properties [92]. This was run on a 64-bit ArchLinux workstation with 4GB of RAM and a quad core Intel® Core™ i5-2400 CPU (3.10GHz) running the Oracle JVM version 1.7.0_25. The analysis program ran over the output graphs and exported the relevant information to a `csv` file so that it could be analysed. Analysis was performed using R versions 3.0.2 (2013-09-25) and 3.0.1 (2013-05-16) [89]. The use of multiple R versions was a result of the size of the dataset. As the dataset is very large, some of the analysis needed to be run on a high memory machine. To get access to high memory machines, the dedicated grid was used [85] along with standard Victoria University of Wellington lab machines for low memory computations.

4.1.4 Validity

While the graphs span a range of sizes (10 – 110 vertices, 9 – 241 edges) and densities (0.86% – 85%) they still only represent graphs that occurred in the context of the original graph creators. These graphs will not cover the full range of graph structures possible. However, this thesis assumes that their variety is sufficient to give a strong indication of how other graphs may be expected to behave.

This chapter only explored four of the algorithm's parameters. There are more parameters that were ignored. As the other parameters were kept constant throughout the experiment this chapter assumes that these results will still hold, especially as most of them are not directly responsible for the strength of attractive and repulsive forces in the system.

Timestep has been used as a proxy for forces that do not cause

vertices to start moving such as friction and drag. It is assumed that similar effects can be produced by varying friction and drag as timestep. This is because all of these can be used to change the speed of the simulation in different ways. Therefore, they may have similar effects on the layout. This assumption can be tested in future experiments but was beyond the scope of this thesis.

4.2 Results

The system recorded the final state of all the trials and analysed the results using R. Since multiple graphs were laid out simultaneously it is hard to determine how long it took to lay out a single graph, however, laying out 13,720 graphs took approximately 5 minutes (≈21 milliseconds per graph if they were sequential). Distributions of the results factored by every variable can be found in Figure B.3 on page 146 for reference. Some example outputs from the experiment can be seen in Figure 4.1 on the next page.

Proportion of Expected Edge Length In the experiments we measure the mean edge length of the graphs. However, for different combinations of parameters we expect to layout graphs with different expected length edges. In order to simplify the presentation of results this chapter introduces a new metric: the proportion of expected edge length. The proportion of expected edge length is the ratio of the mean edge length to the expected edge length. It is calculated by:

$$\text{Proportion of expected edge length} = \frac{\text{Mean edge length}}{\text{Expected edge length}}$$

The proportion of expected edge length takes the value 1 when

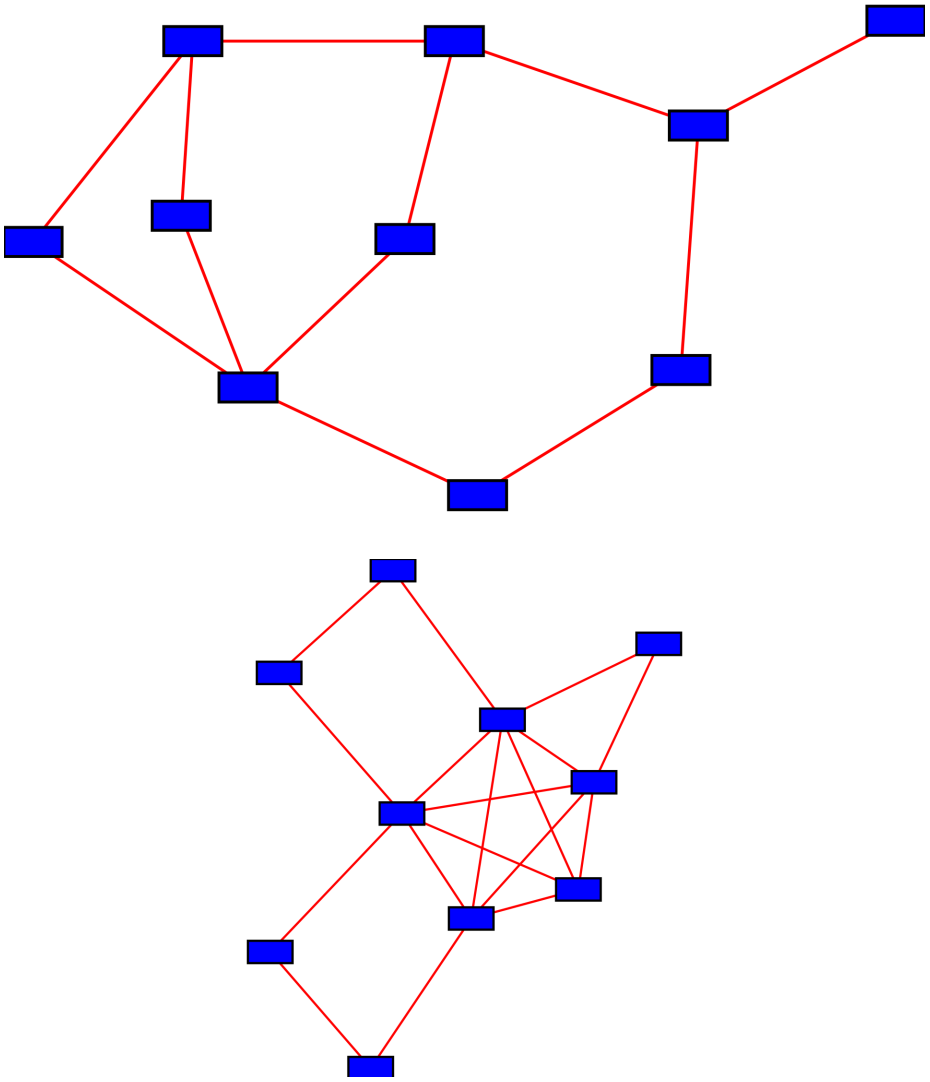


Figure 4.1: Example outputs from the experiment with no walls, timestep: 0.3, K_e: 500, K_h: 0.05.

the expected edge length matches the mean edge length. Otherwise it is the number of times longer that the mean edge length is than the expected. For example, a value of 0.5 implies the mean edge length is half the expected length. A value of 2 implies that the mean edge length was twice the expected.

4.2.1 Mean Edge Length

The first hypothesis ($\mathcal{H}_1$) is that the expected edge length will be linearly related to the mean edge length. In terms of proportions of expected edge length, it means that we expect values to be pretty close to 1.

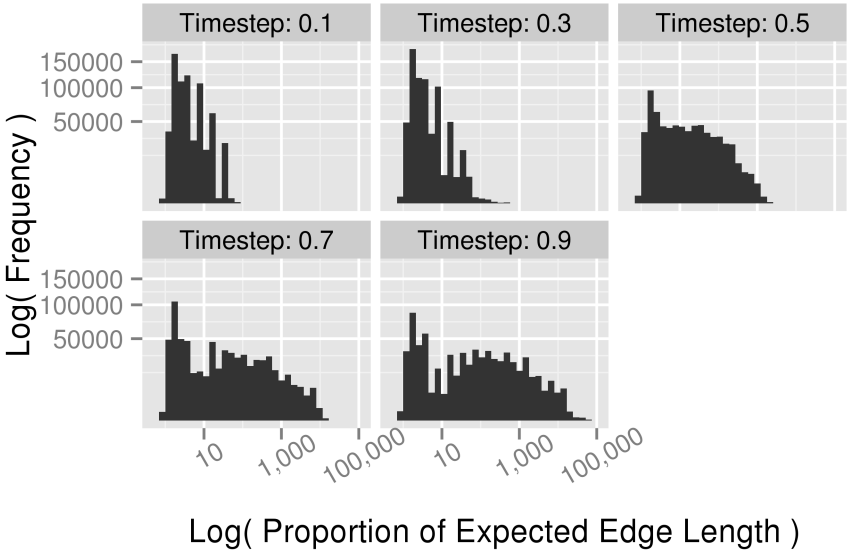


Figure 4.2: Distribution of proportions of the expected edge length by timestep. This shows that as the timestep increases edges get much longer relative to their expected edge length.

Figure 4.2 shows the distribution of proportions of expected edge length by timestep value. This shows three things:

- in general edges are longer than expected (lots of values are > 1);
- the range of proportions of expected edge length values is several orders of magnitude;
- and very long edges only occur for high values of timestep.

This disproves both $\mathcal{H}_1$ and $\mathcal{H}_3$. Note that this does not suggest that a high timestep will result in long edge lengths. A high timestep increases the range of edge lengths produced, but does not reliably produce edges of any given edge length.

Further exploration shows that for low values of timestep, specific values of K_e and K_h have essentially no effect. While this is not visible in the graphs shown, it is visible in the complete results (Figure B.3 on page 146).

Further splitting by walls has has little effect, except where edges become too long in the presence of a high timestep. This disproves $\mathcal{H}_1$, provides support for $\mathcal{H}_2$ and strongly disproves $\mathcal{H}_3$ that timestep will have no effect.

In general this suggests that EEL values are a reasonable way of thinking about the parameters since their component values do not have a strong individual effect. Further examination of the values suggests that in general EEL values are an underestimate of the true mean edge length.

Relationship with Final Kinetic Energy

In force-directed layout the kinetic energy of the system provides information about how settled the system is. The more vertices are moving around quickly, the higher the kinetic energy is. The kinetic energy is also used as a termination condition. Once the energy is low enough, the vertices are no longer moving a lot, then the layout

algorithm is stopped. This prevents the algorithm from running when it would no longer be making any useful changes to the layout.

Since the kinetic energy is used as a termination condition, there is a certain implication that the kinetic energy provides information about the layout. That if the algorithm terminates by running out of time before the kinetic energy gets low enough, then that may be a worse layout then if it was allowed to run until the kinetic energy threshold was reached. An intuitive argument would be that if the kinetic energy is low then the edges length are not changing much, and have stabilised at their ideal values. In contrast, if the kinetic energy is high, the edge lengths are changing a lot, and so may not reflect their length in a more settled state.

Following that reasoning, it is possible that lower kinetic energies at the end of layout have shorter mean edge lengths. Moreover, early pilot tests of the system supported this hypothesis. Additionally, the previous section showed that long mean edge lengths occur at high values of timestep. Therefore it is also possible that high values of timestep result in more graphs with high values for final kinetic energy. This section explores both of these options.

Correlation with Proportion of Expected Edge Length Figure 4.3 on the facing page shows the final kinetic energy against the proportion of expected edge length. This shows virtually no relationship between the final kinetic energy and how much longer than expected the mean edge length is, irrespective of the presence of walls or the value of timestep.

Closer examination of the data shows that this effect is independent of graph size or density (see Figures B.1 and B.2).

Correlation with Timestep Figure B.3 suggests that the longer edge lengths (comparatively to the expected edge length) tend to

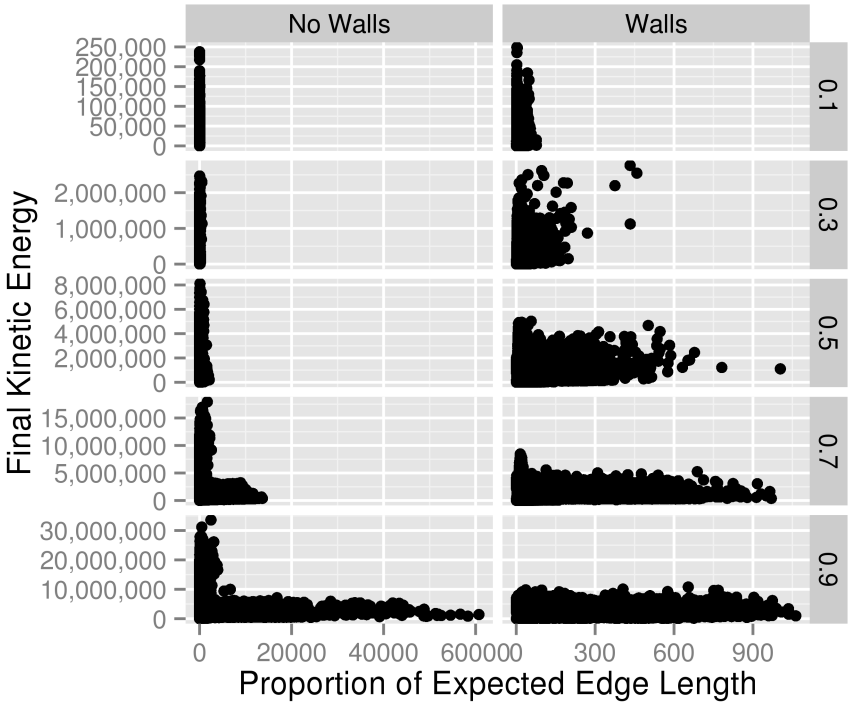


Figure 4.3: The relationship between the proportion of expected edge length and the kinetic energy separated by whether there were walls. It is clear that there is no useful relationship between the two values from the rectangular shape of the data in most cases.

occur more often for higher values of timestep. Figure 4.4 shows that higher values of final kinetic energy are more likely at higher values of timestep. However, this is not strong evidence for the final kinetic energy affecting edge length as it does not show a direct relationship.

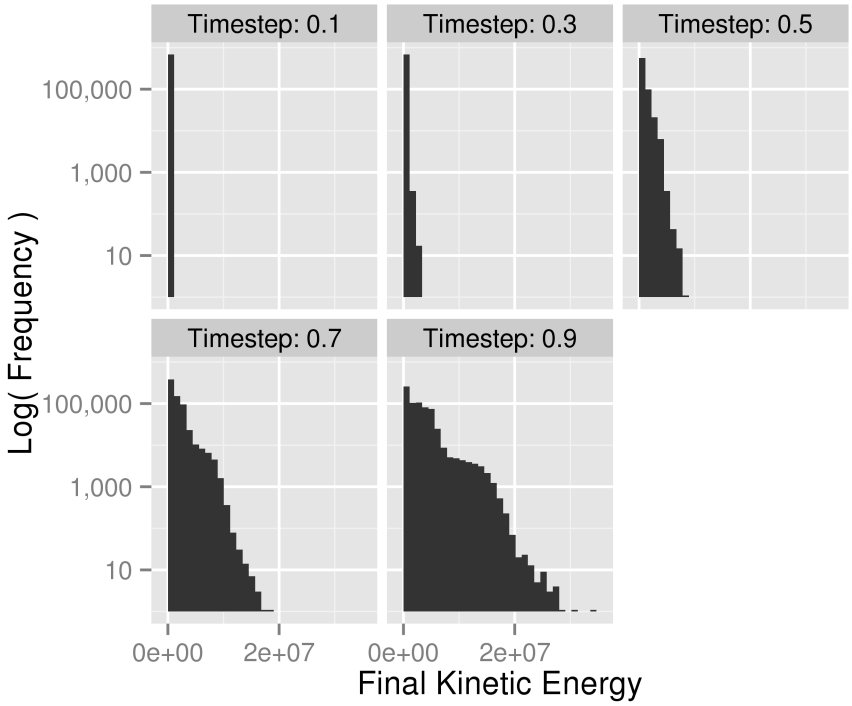


Figure 4.4: Distribution of final kinetic energy by timestep. It shows that there are more graphs with high values of final kinetic energy the higher the value of timestep.

4.2.2 Edge Crossings

Hypothesis $\mathcal{H}_4$ said that edge crossings would be independent of the parameters that were varied. To test this percentages for edge crossings (using Purchase's crossings metric [5]) can be seen in Figure 4.5.

The data do not suggest a strong relationship between edge crossings and expected edge length or timestep. However, there is

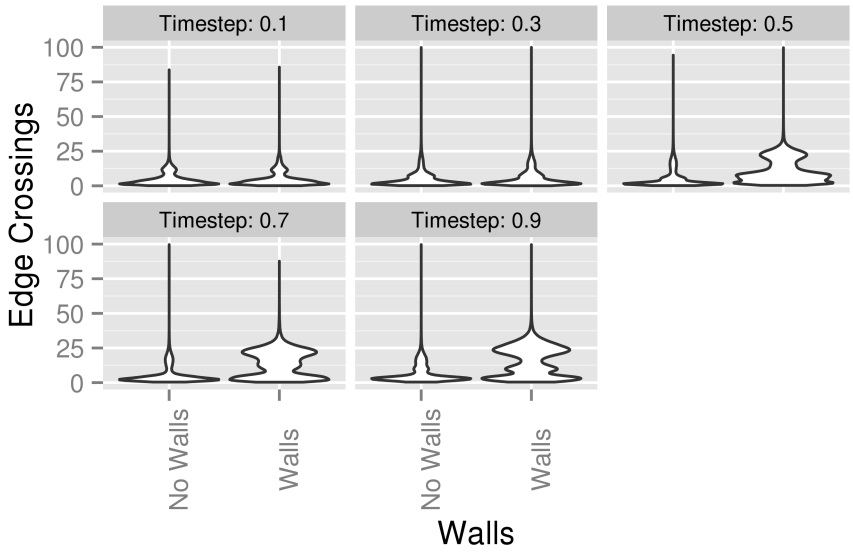


Figure 4.5: Percentage of the edge crossings in a graph (compared to the total possible) split by timestep and the presence of walls.

a suggestion that having walls results in more edge crossings. In general, all combinations of input parameters seem to have resulted in low numbers of edge crossings. This is largely consistent with $\mathcal{H}_4$.

4.2.3 Exploratory Analysis

This section looks at two factors which were recorded by the system, but are not important to any of the claims in this chapter. In particular it explored the size of the bounding box of the graph layout, and the mean distance between all vertices (as an extension of edge length).

Bounding Box In practice a user may wish to know in advance how much space a graph layout will take up, or would take up if it had more space. There is a correlation between the mean edge length and the area of the bounding box necessary to enclose the

graph as shown in Figure 4.6. This suggests that the graph tries to spread out as the edges get longer (though in a restrained way if there are walls). This is particularly interesting as there is no strong effect from the number of vertices in the graph (see Figure B.5 on page 149) as was expected. The relationship seems to be close to linear as shown by the blue regression line. The diagnostic plots for the linear regression shown in Figure 4.6 can be found in Figure B.4 on page 148. These show that the linear model is not quite right. Quadratic and exponential models were expected to provide a better fit, but did not.

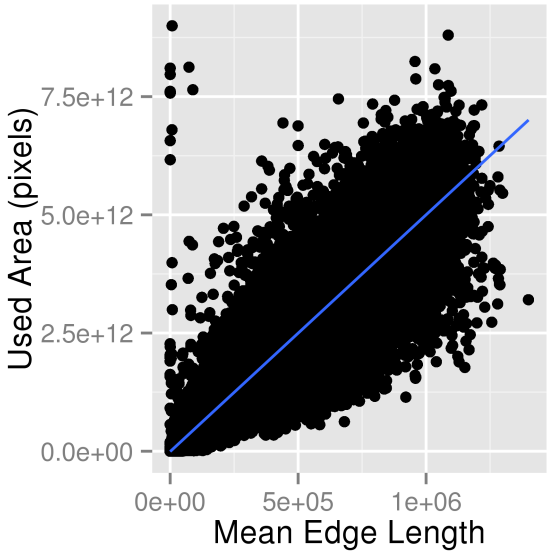


Figure 4.6: Mean edge length vs area of enclosing bounding box. The longer the edges are the more space the graph takes up if it is available. Note that the maximum area available with the walls present is 2×10^6 pixels^2. The blue line is a linear regression.

Vertex Spacing The previous section looked at how much space on the screen a graph occupies. This can be seen a way of looking at the distribution of vertices on the screen. However, this does not

answer questions like: are most of the vertices packed together? An alternative view of vertex distribution over the screen is provided by the vertex-vertex distances between all vertices. This provides a model of how large vertices can be and still avoid overlaps, as well as a notion of crowding. The minimum vertex-vertex distance is essentially uncorrelated to the mean edge length (Figure B.6 on page 150). However, Figure 4.7 shows that the mean vertex-vertex distances are linearly strongly correlated to the mean edge length. This is not an unexpected result as both vertex-vertex distance and edge length provide information about how spread out a graph layout is.

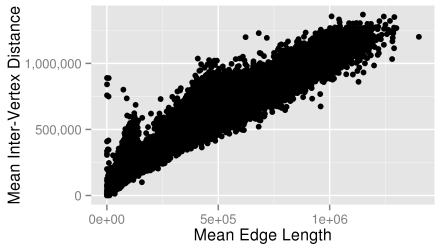


Figure 4.7: The mean vertex-vertex distance and the mean edge length are shown to be strongly correlated.

4.2.4 Individual Parameter Effects

From the data that we collected, we can make some general conclusions about how the parameters that were evaluated in this chapter affect the resulting layout.

The most interesting result is that individual values of K_e and K_h are not important. It is only the balance between them which is important - the EEL value that they get. This is a practically useful result as it shows that you can tune edge lengths by changing either K_e or K_h independently.

The timestep parameter has been shown to increase the variability in results. In particular, layouts with large timesteps are liable to have some graphs (at random) be laid out with much longer edge lengths than would otherwise be expected.

The walls were not found to have any interesting effect on the layout. While they do contain the graph, they do not seem to change the layout in any other way. This suggests that walls can safely be added to graph layout to constrain the layout to fit on a given output medium.

4.3 Parameter Confirmation Experiment

This experiment replicates the second experiment from Chapter 3 using a sample of 38 graphs on all 96 force combinations. This is done because the exact set of parameters used that chapter are not among the combinations tested in this chapter. While only a small number of graphs were tested so that the experiment would be quick to run, all force combinations were tested.

The parameters used in this experiment were the parameters in Table 4.2 on page 64 and $K_e : 5000, K_h : 0.05$ from Table 4.3 on page 64. Some parameters from the previous chapter (e.g. K_l) do not have an analogue in this chapter and so use their original values. This is done to provide additional evidence that the results from the previous chapter hold in light of the current results. In other respects this ran in the same manner as previously described.

Note that the size of the vertices is much reduced between the chapters. This may reduce the difference in performance between different force combinations as the number of overlaps is likely to decrease significantly. As a result, the hypotheses being tested are: HWED is the best equal force with respect to reducing overlaps; and HWED has fewer overlaps than H. These hypotheses are chosen to

validate the use of the H and HWED forces in Chapter 5 on page 79.

4.3.1 Results

The results of this experiment were analysed in the same manner as above. Due to the smaller vertex sizes used in this chapter there are fewer overlaps in the resulting graphs. The full table of results can be found in Table C.1 on page 151.

The interesting results are that HWED is still the best equal performing force with respect to overlaps and occlusions generating graphs with 0 overlaps or occlusions. The base force combinations (H and L) both still generate graphs with overlaps and occluded pixels. HWED also generates graphs with low numbers of edge crossings (1.94% of the possible edge crossings compared with 1.98% in Experiment One and 2.22% in Experiment Two).

This validates our selection of the HWED force as the best performer. This also confirms that HWED performs better than H, which further validates the choice of the two forces to compare in Chapter 5.

4.4 Conclusion

This chapter describes an experiment to explore how parameter selection affects the edge lengths of graphs laid out using force-directed layout. The results invalidate the first three hypotheses stated earlier in the chapter, and confirm the fourth. The results suggests a number of conclusions:

- Increasing the expected edge length will increase the mean edge length, but will only do so reliably for low values of timestep.
- Edge crossings do not seem to follow any clear pattern with respect to either timestep or expected edge length.

- The individual values from the parameters K_e and K_h do not matter. However, the edge length that results from their combination may have a significant effect on the layout. Ideally the edge length should be kept as short as practical, which will usually involve keeping both K_e and K_h as low as possible.

- The timestep parameter has a significant effect on graph layout and should be kept as low as possible.

- The parameter which controlled the presence of walls did not have an observed effect on the resulting graphs. It may, however, significantly affect the aesthetics of the layout as it changes how the graph can move and spread out.

Chapter 5

User Study

The previous chapters have shown that graph layout can be affected in significant ways by modifying the input parameters or the forces used. All of these evaluations are based purely on assessment by metrics. While the metrics that were used are based on well-known ideas, there is no clear relationship between the values that a metric takes and a measurable output with respect to human user performance.

For this reason, this chapter conducts a user study where two graph layouts with different metric results are compared under free-form multi-selection tasks. Specifically, this chapter compares the H and HWED algorithms from Chapter 3. These two combinations of forces were found to have significant differences with respect to their metric values.

An experiment was carried out in the context of a personal social network on an entertainment console. Entertainment consoles have become a standard household good, owned by over 50% of households in North America alone [93, 94]. Since the mid 2000s, consumer entertainment consoles have been releasing gesture based controllers [70, 71, 68], which are now becoming commonplace. These controllers allow users (or a point controlled by the user) to be tracked as they move around. Alongside more traditional gaming functions,

these controllers are also used for tasks such as interacting with various forms of media and social networks.

Many users use consoles as a social interaction platform [95, 96, 97]. Using a graph layout to display friends can cluster friends who know each other into coherent groups. This can simplify the process of finding and selecting a group of users to interact with. Entertainment console media (i.e. video games, movies, music) have a natural graph representation based on both similarity and purchasing recommendations [98].

While many new games target gesture based controllers, free-form multi-selection is not a common console gameplay mechanic. In contrast, selection is a core component of interactive systems [52]. It also differs from many gestures (both game and other) in that it is sensitive to location. Small changes in physical hand position are exaggerated and result in large cursor movements. Many gestures are either location agnostic, or partition space into a small number of coarse sections. In contrast, free-form multi-selection follows the exact path the user traces, and a few centimetres in the wrong direction can result in an incorrect selection.

Contributions. This chapter describes a user experiment where 74 participants interacted with graphs laid out by two different algorithms to see if there was a difference in user performance. The experiment was carried out in a controlled lab environment, with a within-subjects experimental design. The chapter also looks at users' perceptions of the system and how practical it was to use. Overall it finds that:

- The HWED and H (control) layout algorithms did not have a significant difference in time taken to complete a selection.
- When using graphs laid out by HWED the participants made less errors than when using the control algorithm.

- HWED produced layouts with a lower steering complexity than the control.
- Multi-selection using free-form polygons allows for greater error tolerance than repeated single selection.

5.1 Experiment

This section describes an experiment to evaluate the difference in user performance and preference when performing graph selection tasks using a single PlayStation Move controller, on graphs laid out by the H (the control as it is a standard implementation) and the HWED algorithm. The experiment was approved by the Victoria University of Wellington Human Ethics Committee. Each participant had to select a similar set of vertices in a pair of graphs (actually the same graph laid out with a different layout algorithms). They then answered three questions about their preferences and perceived speed. The system recorded all user interactions allowing the system to determine the true time spent on each graph.

5.1.1 Graph Layout Algorithms

The two graph layout algorithms compared are the standard force-directed graph layout algorithm by Eades [4] (H), and the HWED algorithm from Chapter 3. Chapter 3 shows that graphs produced by HWED have a reduced probability of overlaps, and potentially increased space between the vertices. It was hypothesised that these properties would make the selection tasks easier in two ways. First by decreasing the steering complexity that results from needing to not select extraneous vertices in the experiment. And second by compensating for the noisy signal from the Move controller. The purpose of this experiment is to explore the effect of differences in

metrics on human performance. While the metrics in the previous chapter showed significant differences in the layout, it is unclear whether those differences will affect a user interacting with the graph.

5.1.2 Graph Interaction

Choosing to work in the context of an entertainment console, limited the common devices to either a gesture controller or D-pad. However, the D-Pad is quite unsuitable for this task. Using the D-pad to jump from vertex to vertex by moving along edges may be difficult as there may be more than one edge going in a single direction. Additionally, it is hard to draw free form polylines around an area using the D-pad (directional or analog), due to both awkwardness of interaction (using the D-pad to control a cursor) as well as the slow speed of interaction that is required to keep control of the cursor. Moreover, consoles are moving more and more towards gesture controllers, and it is important to not ignore the reality of the situation in HCI settings. The PlayStation 4 D-pad controller has the ability to be tracked in 3D space much like the Move controllers used in this work [99].

5.1.3 Data Set

Rather than using the graphs from the first study, this experiment uses subsets of the VAST challenge data set [38] which contained a social network. Each graph was made by taking the friend networks of each user out to two degrees. It is hypothesised that these graphs are more likely to to be structured like real social networks than randomly generated data, and they are publicly available. This also helps validate the findings in Chapter 3 by testing on a different set of graphs. This validation is very limited, particularly in the number of graphs tested and that they were all likely to have similar structure.

The experiment used a random subset of 20 graphs that had no more than 50 vertices. There was no limitation on the number of edges. The clusters of vertices to select were manually assigned. Each cluster was assigned such that the entire cluster can be selected in one selection.

A limitation of this work is that the vertices to select were chosen after the layout stage, and did not have to be the same for each pair. Instead the vertices were in a similar location in both layouts. This was because there was no way to guarantee that any set of vertices would be laid out close to each other. This experiment only tested selections that could be selected in a single free-form selection with up to 5 vertices, as more complex selections are just repeated single free-form selections. A number of graph pairs used in the experiment can be seen in Figure 5.1 on the next page. The nodes which need to be selected are shown in blue.

5.1.4 Test Participants

Participants were recruited by asking students attending Victoria University of Wellington to participate and to invite their friends. Participants were informed the study would take 15 – 30 minutes, and would be put into a draw to win gift vouchers for participation, as well as a prize for the best performance. Participants had to be at least 14 years of age, with any below 18 requiring parental consent.

5.1.5 Setup

The display was a 40" Panasonic TV with a resolution of 1920x1080 pixels. The user was seated on a computer chair which they could adjust as they chose. The chair was positioned opposite the camera which was at the bottom centre of the screen. It was raised on some books, as the table was both too wide (the camera saw a lot of the

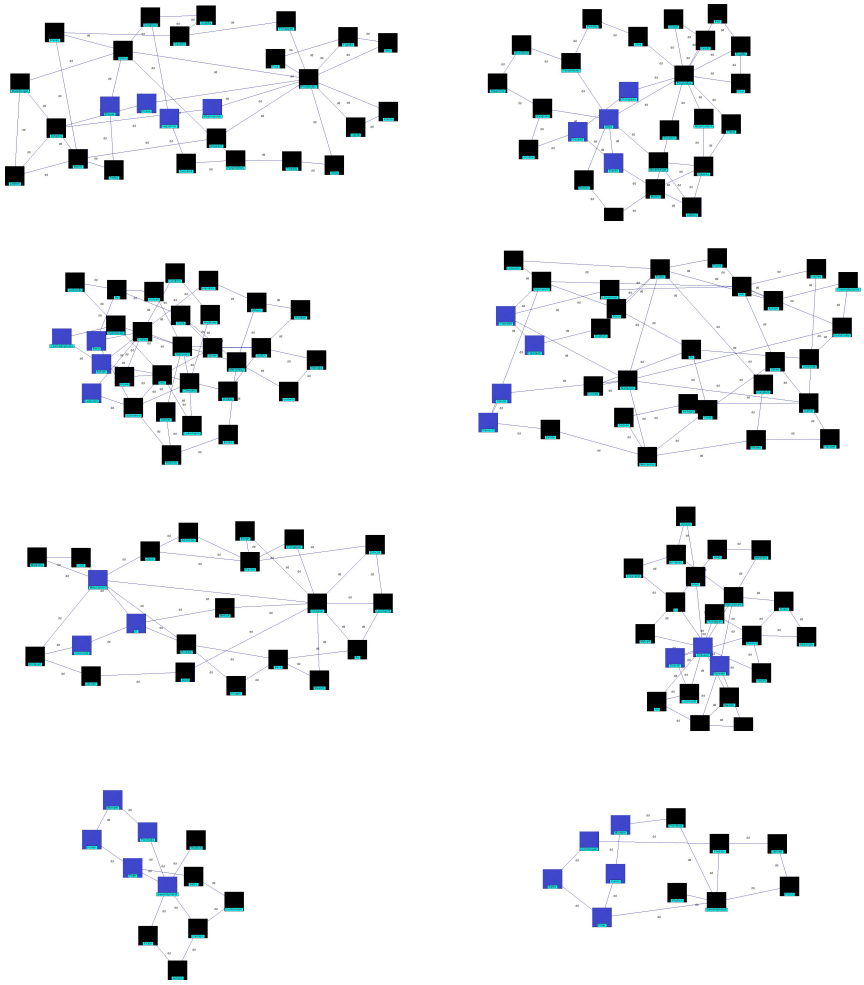


Figure 5.1: Pairs of graphs laid out by different algorithms used in the experiment. Each row is the same logical graph. Nodes which need to be selected are blue.

table in its field of view), and the user was quite high relative to it. A post-it note was hung on each side of the screen with one of the buttons on the PlayStation controller. The entire set-up can be seen in Figure 5.2.

Figure 5.2: Experimental setup. The PlayStation eye camera is propped up on a book as the desk is too low. The vertical column of post-it notes contains measurements that participants could use to estimate their height.

As the room being used was quite narrow, and the table quite wide, participants could not move around much without the camera losing track of the controller. For some participants, their arm's reach could extend out of the camera viewing angle. Fortunately this did not prove to be an issue with only 1 participant performing large enough gestures for this to manifest, and only in the practice graphs. The participants were all initially seated on a standard office chair in a fixed location in order to minimise differences in gain (see Section 2.3.2 on page 30).

The system from Chapter 3 was used to lay out the graphs, and was extended to support and record interaction with the graphs [88]. Additionally, a Java library was developed (and released open source) to allow communication with the Sony Move.Me system [100].

5.1.6 Tasks

The experiment was divided up into four parts, taking 15 – 30 minutes in total. Questionnaires in the study used a seven point Likert scale. Initially each user filled out a pre-questionnaire to get some demographic information as well as previous experience. They then calibrated the system, and performed four practice tasks. Upon completion of the practice tasks the experiment began.

Users were presented with two graphs, one after another, where they had to perform a similar selection task. They were supervised by the author. Unknown to the users the two graphs were logically the same, but had been laid out using different algorithms. Once the correct selection was completed the application automatically took the user to the next graph. When both graphs had been completed they were presented with a question screen about the two previous graphs as in Figure 5.3 on the facing page. To leave this screen they had to both answer all the questions and press the `x` button on the controller (this allowed users to change their mind about their answers). Each user was presented with the same 20 pairs. The order of the graph pairs and the algorithms within each pair were shuffled for each user. Upon completion of the selection tasks users answered a post-questionnaire, completing the experiment.

5.1.7 Interaction

The PlayStation Move was set up in laser mode, in which it attempts to work out where the controller is pointing like a laser pointer (similar to ray casting). This takes care of the issue of reachability as it becomes an issue of angle rather than reach. To calibrate the controller, the participant pointed the controller at each of four post-it notes around the TV (Figure 5.2 on the previous page) and pushed the button shown. These could be done in any order.

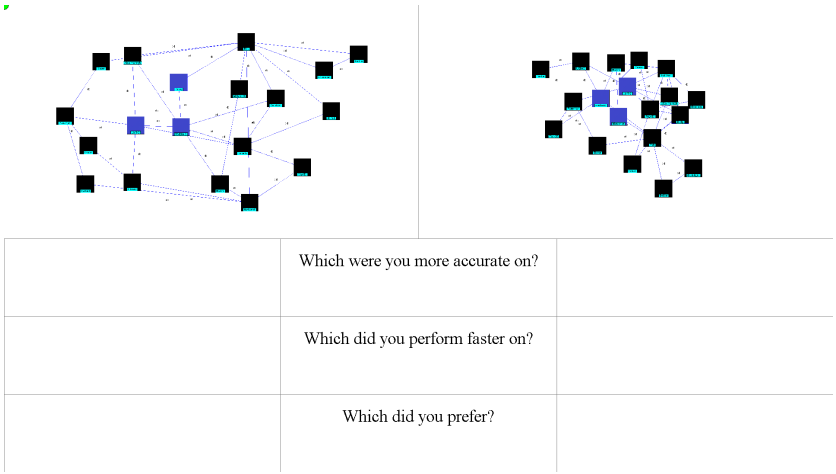


Figure 5.3: A screen asking the three questions about the two tasks that have been completed immediately prior. The green dot in the top left hand corner is the cursor.

A green dot with a black outline and 20 pixel diameter was shown as the cursor. This was the raw position as read from the PlayStation Move Software. If the controller went out of sight of the camera an error message was shown at the top left of the screen, but the system would continue to attempt tracking.

In order to enter the selection mode, users had to hold down the trigger (this was under their index finger). This started drawing a self-closing pink polyline that showed in real-time which vertices would be selected when the button was released, finalising the selection. The polygon obeyed the inside-outside rule, which meant that items could be deselected partway through a selection (Figure 5.4 on the following page). Selected vertices were shown with a large green tick on them. Holding the Move button (under the user's thumb) drew a blue polyline with yellow fill that deselected vertices, in real time, with the same rules as selection.

Allowing the user to begin their selection at an arbitrary point avoids Bowman et al.'s Heisenberg effect. All precise movements can

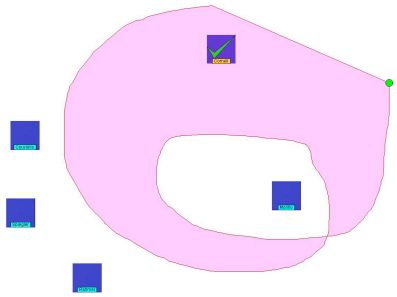


Figure 5.4: The polygon is self-intersecting and can have holes in it. This is a result of the inside-outside rule that says that containment requires a ray going out to infinity to pass through an even number of walls to be inside the polygon.

be done while the button is being held down, avoiding the issue.

5.2 Results

The experiment had 74 participants, many of whom were students studying computer science related topics at Victoria University. R version 2.14.2 (2012-02-29) was used for all the data analysis [89]. 13 participants were female, the mean age was 24 years old, and the mean height was 176 cm. Charts showing the distribution of participant age and height can be seen in Figures 5.5 and 5.6. 75% of participants reported that their exposure to the PlayStation Move controller or similar controllers such as the Nintendo Wii remote was "rarely" (3 out of 7) or less.

The binomial test was used to check that the ordering of which layout came first for each pair of graphs was random (to verify the implementation). It gave a 95% confidence interval of $[0.48, 0.53]$ suggesting that it was in fact random.

The participants answered three questions after each pair of graphs. The binomial test was used to see if the order mattered. For the two questions relating to performance - speed and accuracy

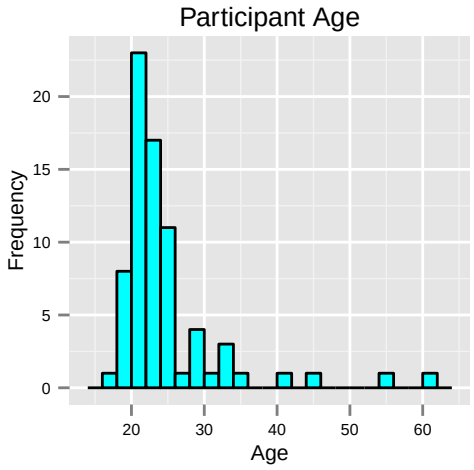


Figure 5.5: Distribution of participant ages. Most are young university students.

- there was no bias based on position at 95% confidence. However, user preference slightly favoured the second layout shown with a 95% confidence interval of $[0.445, 0.498]$. The binomial test was used to determine if users correctly identify which layout they performed faster on. The 95% confidence interval is $[0.69, 0.74]$, suggesting that participants could usually identify which layout they had performed faster on.

Performing the binomial test on the answers users gave provided the same results. Speed and accuracy have no significant difference based on algorithm ($p = 0.30$ and $p = 0.51$ respectively). Preferences show a significant difference of very small magnitude in favour of the modified algorithm ($p = 0.036$, confidence interval $= [0.50, 0.56]$).

The next tests looked at speed differences based on different layout. The Shapiro-Wilk normality test showed that the distribution for times to complete a graph is not normal ($p < 2.2 \times 10^{-16}$ in both cases). Hence the Wilcoxon rank sum test was used to compare the times per graph based on different layouts, and did not find a

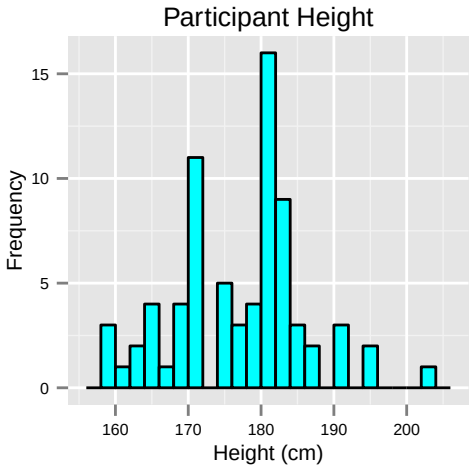
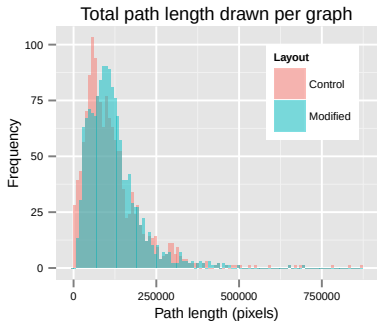


Figure 5.6: Distribution of participant heights. Most participants were in a fairly narrow height range.

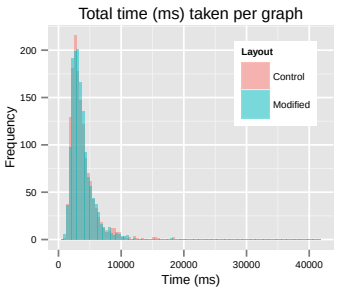
significant difference ($p = 0.80$). The system also determined the total path length drawn to complete each task. Again, the Shapiro-Wilk normality test found the distribution to be not normal ($p < 2.2{\times}10^{-16}$ in both cases), so the Wilcoxon rank sum test was used to show that the values were significantly different at 95% ($p = 0.00043$). The difference in path lengths can be seen in Figure 5.7a on the next page, and the similarity in times in Figure 5.7b on the facing page.

As a measure of how often participants made mistakes, the system recorded the number of times participants used the deselection tool per graph. The Wilcoxon rank sum test showed a significant difference at 95% ($p < 2.2 \times 10^{-16}$) in the use of deselection. Looking at the histogram of the log of deselection tool usage frequencies (Figure 5.8 on page 92), shows that users with the modified algorithm were less likely to use the deselection tool. As users could not advance to the next graph without completing the task correctly, this means that these users were less likely to make an error in their selection.

The feedback from users was generally positive. Each participant



(a) The total path length used to make a selection.



(b) The total time taken to make a selection.

Figure 5.7: These charts show the frequencies of path length for a selection, and time taken for a selection. There is no significant difference between the times, but the path lengths for the modified algorithm are longer suggesting that the user's arm is moving faster.

answered four main questions, evaluated on a Likert scale from 1 to 7, asking their agreement with the following statements:

- The system was fun to use.
- The system was novel to use.
- I would use this system again.
- The system did what I wanted.

Bar-charts of feedback for each of the questions can be seen in Figure 5.9 on page 93. Values of 1–3 indicate disagreement, 4 neutrality and 5–7 agreement. These clearly show that users largely chose ratings which agreed with all of the sentences, indicating positive feeling.

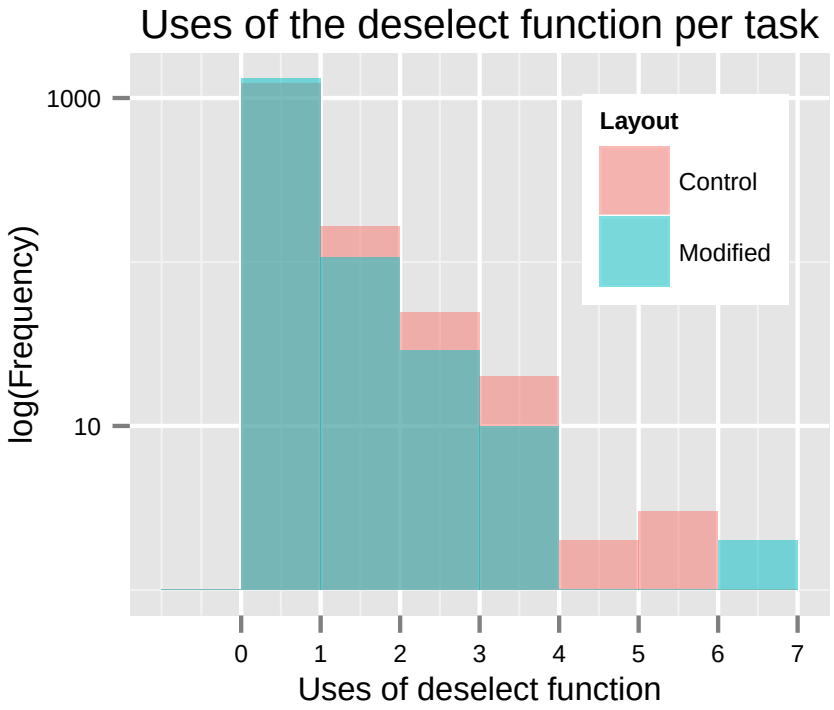
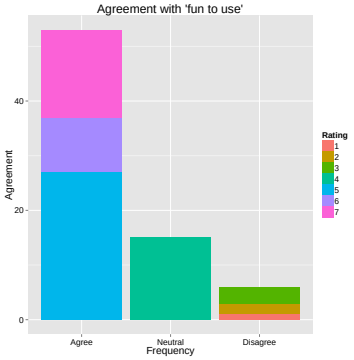
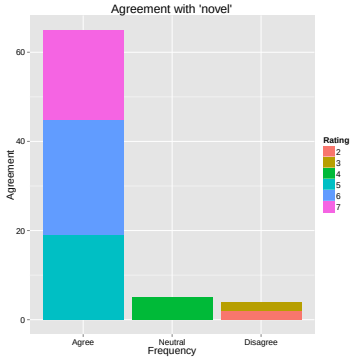


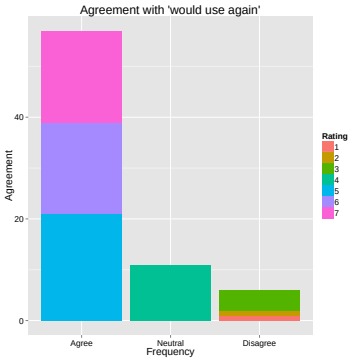
Figure 5.8: Number of times the deselection functionality was used. Note that the frequencies are shown as logarithms, and that for non-zero uses the control algorithm generally has higher incidence. This difference is supported by significance testing.



(a) User ratings for “fun”.



(b) User ratings for “novel”.



(c) User ratings for “use again”.

(d) User ratings for “did what I wanted”.

Figure 5.9: User ratings for questions. 1 is disagree, 4 is neutral, 7 is agree. For all questions the majority of results are at 4 and above suggesting that users had a favourable reaction to vertex selection using the PlayStation Move Controller.

5.3 Discussion

The feedback on the system was positive. The majority of users indicated both that the system was fun and that they would use it again (Figure 5.9 on the previous page). This is good support for the concept of integrating such controls into the standard user interfaces of consoles. Moreover, only a small proportion of participants particularly disliked the system. It is possible that these participants could be swayed to a more positive reaction if the system could be shown to directly improve their interaction with the device.

A number of users attempted to use the controller with their arm fully extended at the beginning of the experiment. As discussed in the related work, this can cause excess fatigue and render the system unusable for long term interaction. While the participants who continued to hold their arm extended for the entire duration of the experiment (6 minutes on average) did report feeling tired at the end, the majority either started off holding the controller comfortably or moved into a more comfortable position as the experiment progressed.

A number of users started out doing repeated single item selection rather than drawing free-form selections around the entire set of vertices. All participants who started out doing selections in this way changed to free-form multi-selection by the end of the study. No participants changed from multi-selection to single selection. This suggests that free-form multi-selection is something that the participants would use, and so is desirable in a real system.

Users had clear difficulty remembering choices that they had made previously. A number of users commented that they felt like they always preferred the first graph. This is contrary to the analysis which shows that the second graph was preferred, and was also not noticed by the experimenter who was supervising the participants.

The tendency to prefer the second graph, while not very strong,

goes against the assumption that users prefer graphs based on the layout algorithm rather than their ordering. Statistical tests suggest that the graphs were in fact shuffled, so this effect cannot be explained by layout algorithm. A possible reason is that slight bias may be caused by the similarity between each pair of graphs. For each pair, the vertices that need to be selected are in the same area on the screen (as much as possible), and the users may have learnt the location of the selection, and therefore considered the second task slightly easier than the first. Alternatively the first graph may not have been remembered clearly, and the second picked for that reason. However, the magnitude of this difference is small enough to be of little practical interest.

The experimental analysis showed that the modified layout (HWED) makes selection tasks easier for the users. There are two main factors that contribute to this conclusion. The first is that there was no difference in timing between the different layouts, but the paths drawn by the user were longer in the modified case. Results on the steering law [78] and the modifications due to gain [79, 80], imply that, at a constant gain, if the difficulty of the paths is the same, the longer path should take more time. In this case, the longer path took less time. As the gain is constant, this means that the longer paths in the modified layout must have a lower index of difficulty. This also provides additional evidence that the modified layout algorithm results in sparser layouts than the control.

The second is that participants used the deselect functionality less when they were performing selection tasks on graphs laid out with the modified algorithm. This means that they generally made fewer mistakes, while taking no more time to do it. While this does not consider the case of expert users, it still shows that there is a difference for some people, re-enforcing the results from the metric evaluation. Furthermore, this creates a safety net for new users so

they can make fewer mistakes without having to go slower to do it.

It may be that these results are strongly biased towards novice users. As the PlayStation Move controller is not used widely for graph selection tasks, it is unlikely that the participants would have had a large amount of expertise before the experiment. While a number of them had previously used gesture controllers, they would have been unlikely to have done a similar task with them. This is further supported by participant comments to the effect that they felt that they liked the more spacious layouts at the beginning, but started to prefer denser layouts as they got more confident and familiar with the system.

5.3.1 Adjustments to the Controller

The experiment used the unfiltered pointing information from the PlayStation Move.Me system. This gave a noisy signal, making it hard for users to hold the cursor steady, and giving the impression that they had very shaky hands. It was suggested by a number of participants that the system would be improved by adding in a filter over the input signal to reduce the amount of jitter that was visible on the screen. Unfortunately this was not picked up in pilot testing, and so was not implemented in the experiment. Moreover, adding such filtering would add in additional delay between the user performing an action and it being reflected in the program cursor state. This should not make a difference to the overall result, as it affected all participants the same way, but it may have emphasised the effect if it made the task harder to complete.

5.3.2 Participants

The sample of user participants is drawn from quite a narrow range. It is largely young men studying computer science or engineering at

university. This is only a single demographic amongst a variety that use entertainment consoles. While some participants are members of other demographics, there are not enough of them to say that they would definitely exhibit the same behaviours, although in this experiment no differences were observed. However, this chapters hypothesises that these results provide a strong indication about the sorts of people who did participate, and believes that further research on other demographic groups would find similar results.

5.3.3 Future Work

This work limited itself to selection tasks that could be accomplished with a single free-form selection. It would be interesting to see how this extended to more complicated selections, where a single contiguous selection lasso would not be a practical mechanism. It would also be interesting to see if factors like decreasing the pointer sensitivity or having previous experience with the system changed users' perceptions of the system.

5.4 Conclusion

As 3D gesture controllers enter the home as part of commodity gaming systems, people get used to their presence. This mass circulation allows them to start being integrated into people's day to day lives and to be used for tasks beyond gaming. This chapter looks at how different graph layouts explored in previous chapters affect users performing a vertex selection task using the PlayStation Move controller.

Overall this chapter finds that the HWED algorithm from Chapter 3 performs better than its matched control algorithm H. Using HWED reduces the number of errors users make when selecting ver-

tices, as well as reducing the steering complexity of the graph. This provides evidence that the changes that were expressed by means of metrics in previous chapters have a real world effect on how users interact with graphs.

Chapter 6

Graph Symmetry Detection

One of the original claims made about force-directed layout was that it promoted symmetry, a trait that is believed to be helpful in reading graphs [4, 46]. Both of these claims have been tested by Purchase whose findings did not support the claim that force-directed layout increases symmetry [5], and were either inconclusive on [46] or weakly confirming of [43] the benefits of symmetry. However, the symmetry comparison done by Purchase has three important limitations which this chapter attempts to address:

1. The symmetry algorithm focused on the symmetries of the vertices, ignoring edges.

2. The study only considered reflective symmetries, ignoring rotational and translational symmetries.

3. The symmetry algorithm did not have an evaluation to assess its performance.

An important limitation of Purchase's algorithm is that it does not consider edges when calculating a symmetry score (i.e the first limitation). In all connected graphs which are not trees, edges will outnumber vertices. Additionally, edges are drawn as lines connecting

the vertices. This results in the edges having a lot of influence on symmetry within the graph. This is not to say that vertices play no part, but edges only exist where vertices do and so implicitly include their effect (excluding orphan vertices and vertex shape). Consider the graph shown in Figure 6.1. If the edges are ignored, then the dotted lines are two potential axes of reflective symmetry. However, if the edges are also considered then the horizontal axis is the only axis of symmetry.

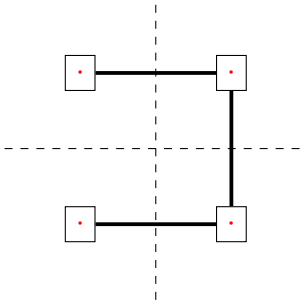


Figure 6.1: A simple graph containing two potential reflective axes of symmetry (black dashed lines). While both of these are equally valid with respect to the vertices, the horizontal line is the only choice when edges are considered. Additionally there is rotational symmetry around the point where the two axes of reflective symmetry meet.

The first two limitations of Purchase's study are addressed here by developing a novel algorithm for measuring symmetry in graphs. The third limitation is addressed by the experiments in this chapter. This chapter extends the symmetry detection algorithm described by Loy and Eklundh [48]. Their algorithm detects reflective, rotational and translational symmetries. The algorithm is modified to handle undirected edges, so that an edge is considered the same after a 180 degree rotation rather than the 360 degree rotation required by Loy and Eklundh.

An important feature of Loy and Eklundh's algorithm is that it can detect when multiple different detected symmetry axes /

centres are actually very similar (i.e. rather than only when they are identical), and groups them together into a single axis. Consider Figure 6.2 on the next page which has a laid out graph with its axis of reflective symmetry shown as a dashed yellow line. The ring of black edges around the outside of the layout are all shown as sharing a single common axis of symmetry (the dashed yellow line). However, close inspection of the image will show that the yellow line is not an exact axis of symmetry for many of those edges. It is good enough that it looks about right, while not being perfect. It is exactly these sort of approximate axes / centres of symmetry that the grouping allows the algorithm to detect.

The chapter uses the new symmetry metric to analyse graphs laid out using the force-directed layout algorithm to gain an idea of how symmetrical the produced layouts are. Finally a user study is carried out to determine whether the symmetry judgements made by the algorithm are similar to symmetry judgements made by humans.

Contributions The main contributions of this chapter are:

- A novel symmetry measuring algorithm for graphs that considers edges and measures reflective, rotational and translational symmetries. This is based on the work of Loy and Eklundh [48].
- Applying the symmetry algorithm to the graphs generated in the experimental study done in Chapter 4 on page 59. The results give an indication as to the variability in symmetry expressed in layouts produced by force-directed graph layout.
- A follow up experimental study over a small number of small graphs which shows that graphs with long mean edge lengths exhibit less symmetry than graphs with short mean edge lengths.
- A user study comparing the symmetry algorithm to human

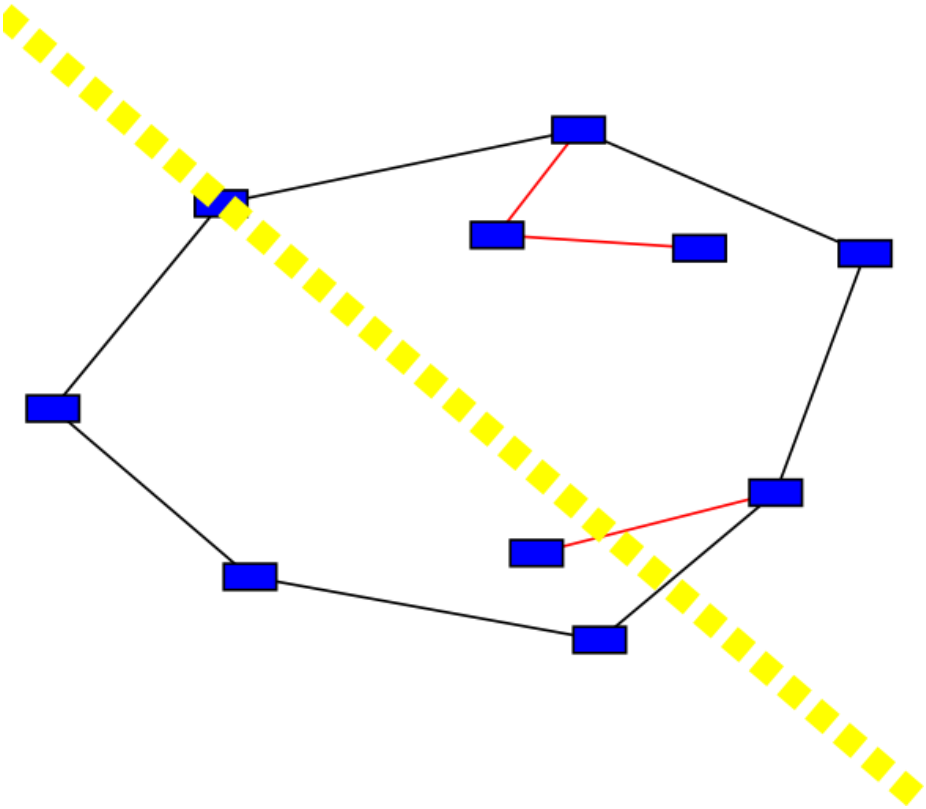


Figure 6.2: Graph with best axis of reflective symmetry marked as a yellow dotted line. The black edges are ones that are mirrored across the yellow mirror line.

judgement showing that the algorithm agrees with humans 92% of the time that agreement is possible.

6.1 Algorithm

The extended algorithm developed in this chapter is described in Algorithm 2 on the next page. The rest of this section describes each part of the algorithm in detail. The implementation can be found as part of the open source graph analysis library released as part of this thesis [92].

Algorithm 2: Symmetry detection algorithm.

```
Data: symmetryType ∈ {reflective,rotational,translational}
      and N the number of axes to look for
Result: score ∈ [0..1]
axes = empty list
features = Convert all edges to SIFT features
for f_i ∈ features do
    if symmetryType == reflective then
        axes.add(perpendicularBisector(f_i), 1)
        axes.add(parallelAxis(f_i), 1)
    else if symmetryType == rotational then
        axes.add(getLocation(f_i), 1)
foreach f_i, f_j ∈ features do
    axis = find symmetry axis(symmetryType, f_i, f_j)
    quality = find symmetry quality(symmetryType, f_i, f_j)
    axes.add(axis, quality)
axes = quantiseAxes(axes)
bestAxes = pickBest(axes, N)
score = score(bestAxes)
return score
```

6.1.1 Initialisation and Storage

```
axes = empty list
```

The algorithm enumerates all the possible axes of symmetry and assigns them a quality score. The initial state is an empty list. Note that the elements of the list are going to be pairs consisting of an axis and its quality score (a number between 0 and 1).

Each kind of symmetry has a different sort of axis:

Rotational Symmetry requires a centre of rotation. This is a point in the 2D plane and can be stored as a pair of floating point numbers.

Translational Symmetry requires a vector. This is a distance and direction and can also be stored as a pair of floating point

numbers.

Reflective Symmetry requires a line of symmetry. The standard representation using m, c from $y = mx + c$ doesn't work as it cannot represent vertical lines, and the difference between two lines based on their m and c values is hard to understand. As a result, this chapter follows the method used by Loy and Eklundh [48] of using the Hough transform [50]. This represents each line as the angle and distance from the origin to the closest point on the line as in Figure 6.3. This always hits the line at a right angle, is unique, is bounded by $[0 \ldots 360]$ in angle, and is bounded by the layout size in radius (as the axis of symmetry is always on screen). As a result the reflective symmetry can also be represented as a pair of floating point numbers (i.e. the angle and radius).

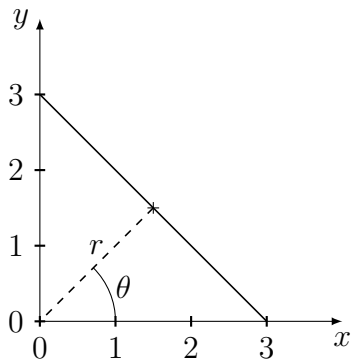


Figure 6.3: A Hough transformed line.

Generic Representation Based on the previous paragraphs a unified representation was created where each axis is represented as a pair of floating point numbers and each quality score as a single floating point number.

6.1.2 Graph to SIFT Features

In order to enumerate all the possible axes of symmetry the edges are converted into a standard format, as they will be used to generate the axes.

```
features = Convert all edges to SIFT features
```

The original algorithm of Loy and Eklundh [48] used Scale Invariant Feature Transform (SIFT) [51] features as inputs. Each SIFT feature is a four-tuple consisting of a location, orientation, scale, and identifying characteristics. This algorithm uses the same technique. Each edge in the graph is turned into a SIFT feature. The location is the centre of the edge, the orientation is the orientation of the edge in degrees, the scale is the length of the edge in pixels, and the identifying characteristic is an `int` which is the same for all edges.

6.1.3 Identify Single Edge Axes

The simplest axes of symmetry to enumerate are those that are generated by a single edge. Note that these single edge axes of symmetry are not present in Loy and Eklundh's algorithm.

```
for f_i ∈ features do
    if symmetryType == reflective then
        axes.add(perpendicularBisector(f_i), 1)
        axes.add(parallelAxis(f_i), 1)
    else if symmetryType == rotational then
        axes.add(getLocation(f_i), 1)
```

For reflective symmetry there are two axes that can be generated from a single edge: the perpendicular bisector of the edge, and the line that runs along the edge. For rotational symmetry there is one: the centre of the edge, as an edge can be spun around its centre 180 degrees to get the same edge.

In all of the cases described above the axes symmetry is perfect, i.e., the edge lines up perfectly after the symmetry transformation. As a result the quality score for all single edge axes of symmetry is 1.

6.1.4 Identify Edge Pair Axes

All other possible axes of symmetry can be generated by calculating the axes of symmetry between every pair of features (edges).

```
foreach f_i, f_j ∈ features do
    axis = find symmetry axis(symmetryType, f_i, f_j)
    quality = find symmetry quality(symmetryType, f_i, f_j)
    axes.add(axis, quality)
```

Every pair of features (edges) generates one or more axes of symmetry. The next few sections will cover how to find the axes of symmetry for a pair of edges.

This section will also show how to calculate each axes' quality score as unlike single edge axes they may not be perfect. This may be because of differences in edge length (for all symmetry types) or orientation (only for reflective and translational symmetry).

The quality score for each axes is the product of its scale quality (S_{ij}) and orientation quality (Φ_{ij}) scores. Each score is bound by $[0 \dots 1]$ and so their product (the quality score) is as well.

The scale quality (S_{ij}) is the same for all symmetry types and is the same as Loy and Eklundh's original paper [48]. In the equation S_{ij} is the scale similarity, s_k is the length of edge k, and σ_s is a scaling factor (tuning the sensitivity). The value of S_{ij} is 1 when both edges are the same length, and decreases monotonically as the difference in edge length (scaled by the total length of the edges) increases.

$$S_{ij} = \left(e^{\frac{-\left|s_i - s_j\right|}{\sigma_s\left(s_i + s_j\right)}} \right)^2 \tag{6.1}$$

Reflective Symmetry

The axis of reflective symmetry is the perpendicular bisector of the line between the SIFT features (edge centres). However, while this guarantees that the mid-points of the edges will match, the orientations of the edges may not line up.

The orientation quality (Φ_{ij}), which measures how closely the edges line up after reflection, is specially adapted from Reisfeld et al. [49] similarly to Loy and Eklundh [48], however, with special consideration for edges being symmetrical after 180 degree rotations. The following equation shows how to compute this value (see Figure 6.4 to see what the different labels are).

$$\Phi_{ij} = \left|\cos\left(\theta_i + \theta_j - 2 * \theta_{ij}\right)\right| \tag{6.2}$$

The following paragraph illustrates how the equation works. Figure 6.4 shows the initial state.

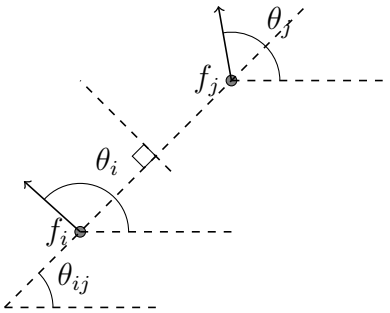
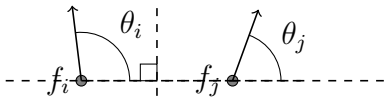
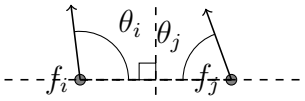


Figure 6.4: Angles needed to compute the orientation metric.

Subtracting θ_{ij} from each of θ_i, θ_j in effect lowers the dashed diagonal line to line up with the x-axis as below.



The reflection will then reflect one of the angles.



If the edges are now parallel then the sum of the two resulting angles will be 0 (mod 180). The less parallel they are the closer to 90 (mod 180) the sum will be. Based on Loy and Eklundh's formulation, the absolute value of the cosine function is applied as it maximises values near 0 (mod 180), and minimises values near 90 (mod 180). This results in a value in the range $[0 \ldots 1]$, which has its minimum at 90 degrees (maximally different angle).

Translational Symmetry

For translational symmetry the required translation is the vector difference in the position of the features. Visually this looks like a repeated structure in the layout as in Figure 6.5 on the next page. This needs to be normalised (multiplied by -1 if dy < 0) to ensure that order that edges are paired up in (which is i and which is j) does not matter.

For translational symmetry the orientation quality has to be changed to deal with the edges not being mirrored. Therefore Equation (6.3) is used.

$$\Phi_{ij} = |\cos(\theta_i - \theta_j)| \tag{6.3}$$

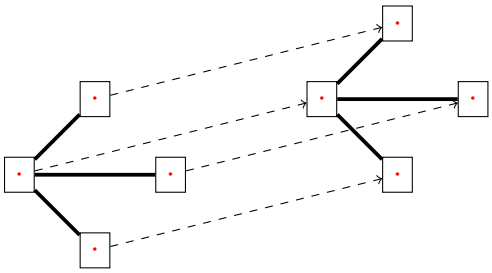
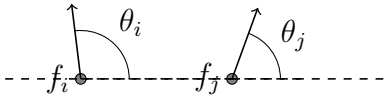


Figure 6.5: An example of translational symmetry. Note how the arrow shape is repeated on the left and right side of the layout. The dashed lines show the translational symmetry.

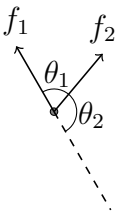
Note that this equation is similar to the equation for reflective symmetry. The only difference is that none of the angles are mirrored so when the connecting line is lowered to match the x-axis you get the following.



However, in this case the edges should already be parallel, so the difference between the two angles is used directly.

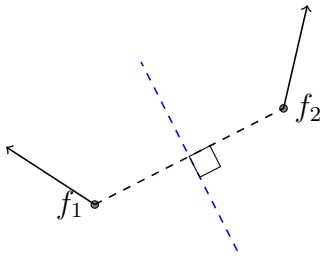
Rotational Symmetry

For each pair of edges there can be up to two centres of rotational symmetry. This is because there are two ways to line up the feature orientations: head to head and head to tail. I.e. you can add 180 degrees to the angle you rotate by and get another correct rotation, however, this rotation may require a different centre of rotation. This is shown below.

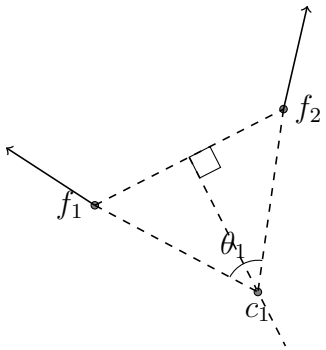


This is different from the original paper where there was only one possible centre, and is a result of an edge being indistinguishable after a 180 degree rotation (because only undirected edges are considered). Note that the orientation of the edges will always match after rotation so an orientation metric is not required (always has the value 1) [48].

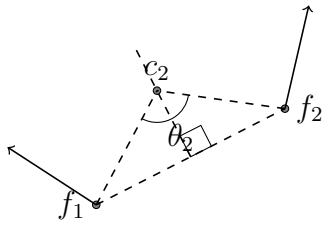
The centres of rotation are always on the perpendicular bisector of the line that joins the two features. This is because the distance of the two edges from the centre of rotation has to be the same for both edges.



We can then use θ_1, the rotation to line the edges up head to head, to find the first centre of rotation.



Similarly centre two can be found using θ_2, the head to tail rotation, as shown in the following diagram. Note that when $\theta_1 = \theta_2 = 180$ both centres will be the same.



6.1.5 Finding the Best Axes of Symmetry

Having enumerated all of the axes of symmetry, the quality scores are now used to vote to find the N best axes. Recall that the value of N is provided by the user at the start of the algorithm.

```
axes = quantiseAxes(axes)
bestAxes = pickBest(axes, N)
```

At this point of the algorithm the axes list contains every possible symmetry axis (with duplicates) and their quality scores. Each axis of symmetry votes for itself. It uses its own quality score as its vote, so a good axis of symmetry (where the edges after the symmetry line up almost perfectly) will have a larger vote than a very poor one (where the difference between the edges after the symmetry is still quite large). Each axis may be in the list more than once, as it may have been generated by more than one pair of edges (or single edge). In this case the votes are summed. The N axes with the most votes are the chosen axes.

In order to deal with noise in the layout — slight deviations from the ideal — the space is quantised. In the implementation used the space is quantised in increments of a chosen integer. This allows votes for two almost identical axes to count for the same axis.

In the original paper by Loy and Eklundh, the voting space is Gaussian blurred rather than quantised. However, that is not ideal in this context as it does not preserve the exact vote count and reduces the number of votes for axes whose neighbours have fewer votes. Quantising the space addresses these issues, as it sums the votes while preserving their number, and does not decrease the vote for poor neighbours. This is important as a common axis of symmetry does not have to have any potential alternatives.

6.1.6 Calculating a Symmetry Score

The final stage of the algorithm is to turn the set of N best axes found into a number that can be used as a metric.

```
score = score(bestAxes)
```

To calculate a symmetry score for a graph: count how many of the edges voted for each of the N axes, and divide that by N times the number of edges (i.e. the maximum possible score):

$$\left(\frac{\sum_{axes} \text{number of edges that voted for this axis}}{N \times \text{number of edges}} \right)$$

In the case of 'perfect' symmetry, all edges are symmetrical with respect to all the axes. This will result in a symmetry score of 1. The score decreases as the number of edges which are symmetrical with respect to the chosen axes decreases. This chapter calculates a separate score for each of reflective, translational and rotational symmetry. It does not combine these values as it is interested in them individually.

6.2 Experiment

This section reports the results of extending the results from Chapter 4 with the symmetry scoring algorithm. The algorithm was run over each of the approximately 3 million graphs to see how the force-directed layout algorithm behaves with respect to symmetry. This allows for the comparison of symmetry across a range of different parameters and a large sample of distinct graphs.

6.3 Results

The results can be seen in Figure 6.7 on the next page. They show that symmetry decreases as the expected edge length increases. They also show a decrease as the timestep increases, which (in Chapter 4 on page 59) was shown to increase edge lengths. Walls seem to increase symmetry in the presence of high timesteps, but less so for long expected edge lengths. Note that the minimum symmetry that can be exhibited by a graph is $\frac{2}{N}$ where N is the number of edges. This is because every pair of edges generates an axis of symmetry.

This suggests that mean edge length and symmetry may be related. This can be seen in Figure 6.6 on page 116. No graphs with long mean edge lengths exhibit large amounts of symmetry. The converse does not apply. Graphs with short mean edge lengths exhibit the full range of symmetry scores.

The effect where long mean edge lengths result in low symmetry scores may be an issue with the algorithm, where it cannot pick up on the symmetry because the error is not quantised proportionally to edge length. It may also simply be that longer mean edge lengths result in less symmetry. To test this theory this chapter carries out a second experiment where the results are compared manually.

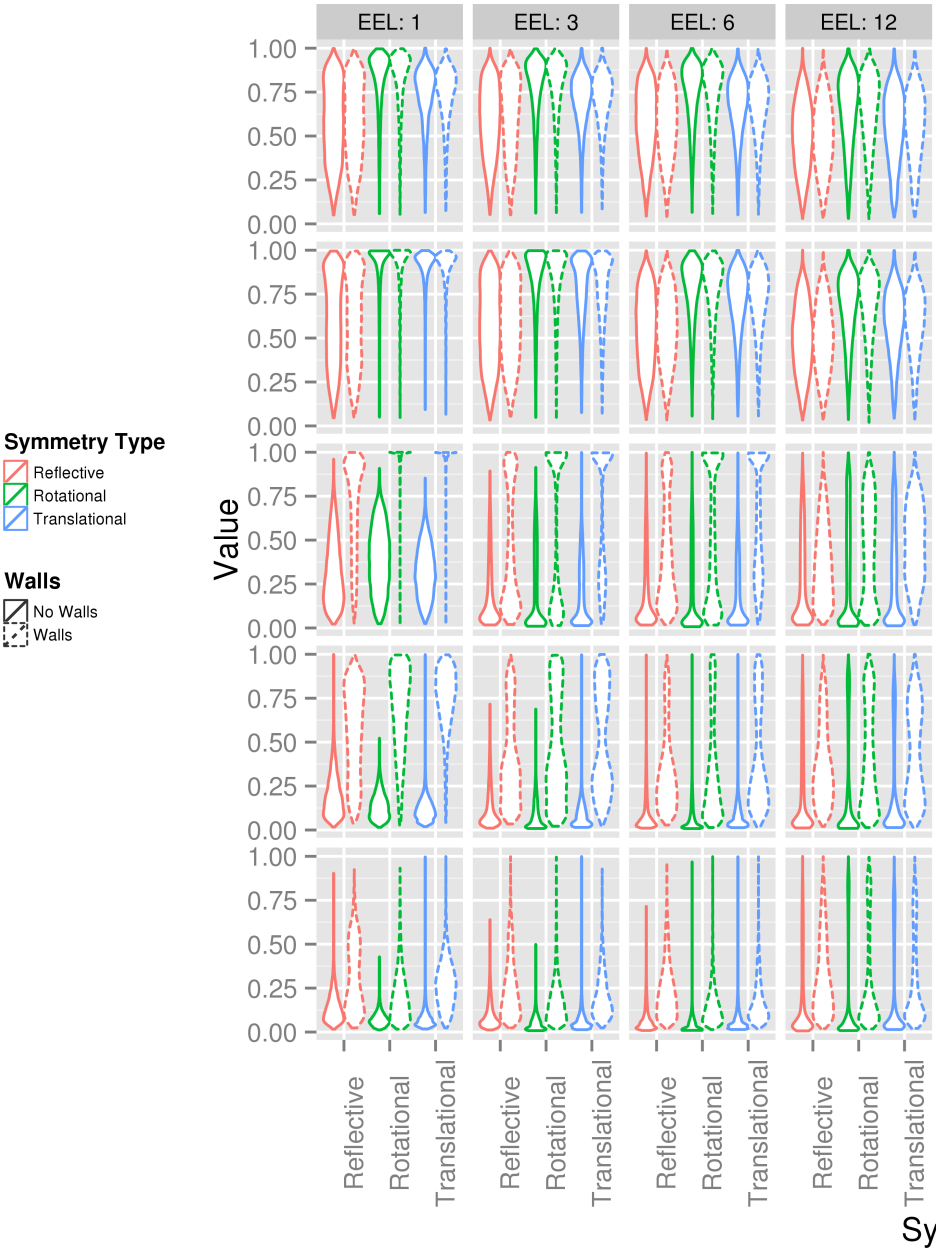
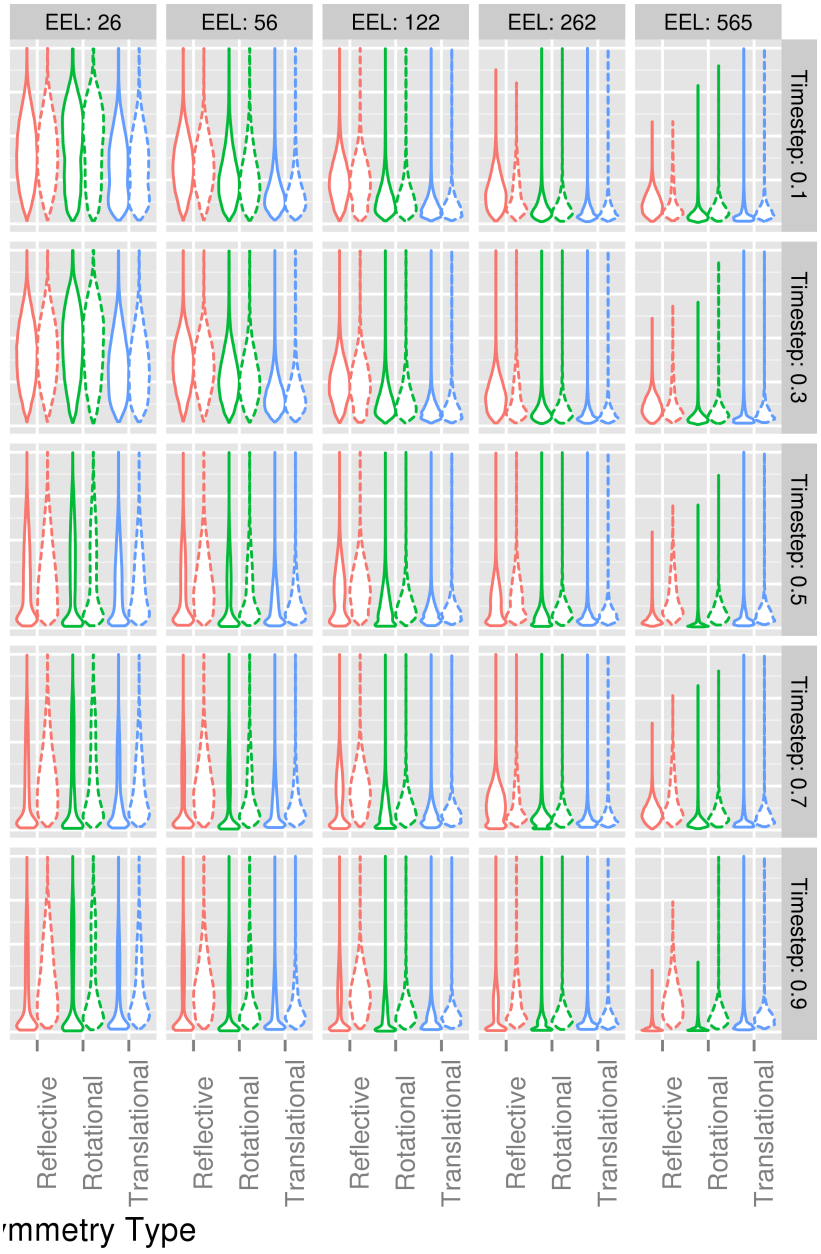


Figure 6.7: This two page violin plot shows the distribution of symmetry score for each type of symmetry separated by expected edge length (EEL) (columns), timestep (rows) and the presence of walls (dotted or solid lines). Colours are used as well as x-axis labels to make the type of symmetry easier to distinguish.


EEL: 26
EEL: 56
EEL: 122
EEL: 262
EEL: 565
Timestep: 0.1
Timestep: 0.3
Timestep: 0.5
Timestep: 0.7
Timestep: 0.9
Reflective
Rotational
Translational
mmetry Type

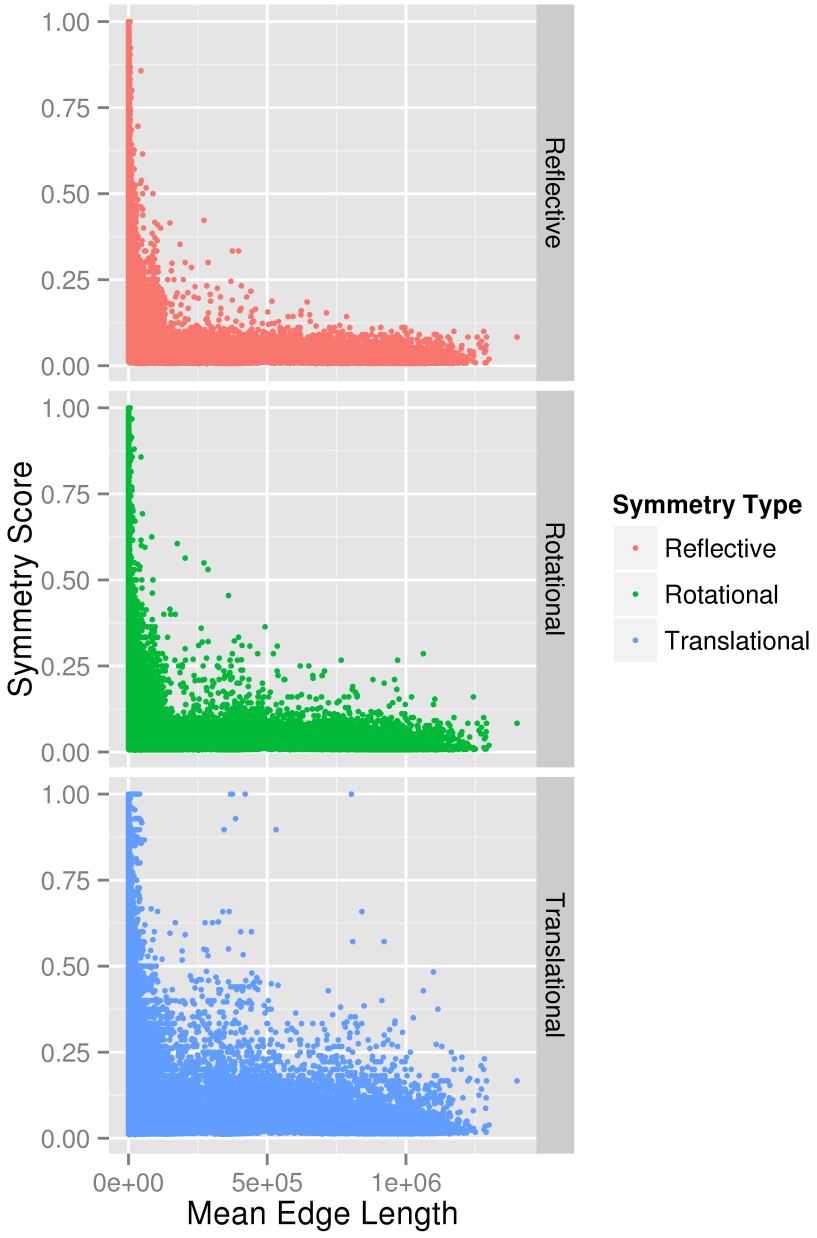


Figure 6.6: Mean edge length vs symmetry present.

6.4 Small Symmetry Experiment

In order to better understand the differences seen between long and short edges in the results, a second experiment was carried out. In this experiment the graphs were generated specially to be laid out symmetrically. Each graph was laid out twice: once with short edges, and once with long edges. The differences in symmetry were then compared both by the symmetry metric and by manual inspection of all the graphs by the author. There were three types of graphs generated:

1. Circle graphs — Each vertex is connected to exactly two other vertices so that the graph can be laid out approximating a circle.

2. Grid graphs — The graph can be laid out as a regular grid.

3. Binary Tree — The graph is a binary tree.

Both circle graphs and binary trees can be created with any given number of vertices. However, grid graphs are always complete square grids, and only their width can be specified. This was done to ensure that they could actually be drawn in a very symmetrical fashion. This was not done for binary trees as it was felt that even incomplete binary trees would be sufficiently symmetrical.

As a test set the system generated circle graphs and binary trees with 3 – 50 vertices and grids with widths from 3 – 7.

The parameters for the experiment can be found in Table 6.1 on the following page. The expected edge lengths were 565 ($K_e = 5000,\ K_h = 0.0005$) and 56 ($K_e = 500,\ K_h = 0.05$). There were no walls to allow the graph to spread out as much as necessary.

Parameter	Value	Parameter	Value
E_K cut off	3	Max Iterations	100,000
Screen width	1920 px	Screen Height	1080 px
Vertex width	20 px	Vertex Height	10 px
μ_s	0.3	μ_k	0.04
Vertex Mass	1	Coeff. of Restitution	0.9
A	0.2	g	9.8
Max Vertex Speed Y	540 $\frac{\text{px}}{\text{iter}}$	Max Vertex Speed X	960 $\frac{\text{px}}{\text{iter}}$
q	3	timestep	0.1

Table 6.1: Constant experimental parameters.

6.4.1 Results

Figure 6.8 on the next page shows the distribution of each type of symmetry for the two different edge lengths. This clearly shows that the layouts with a longer expected edge length result in less symmetry. However, this is not sufficient to rule out an error in the formulation of the algorithm as the cause of the discrepancy.

Unfortunately it is not practical to include all the graphs that were generated in the thesis for comparison. However, the author looked at all the generated images after the experiment. The differences between laying out a circle graph with 8 vertices with the two different edge lengths can be seen in Figure 6.9 on page 120. It can be clearly seen that the layout with short edges has settled to a neat circle, while the longer edges have resulted in a strange asymmetric shape (which is likely to be a local minimum). To mitigate accuracy problems the layout was run for 100,000 iterations with a timestep of 0.1. This should result in the best performing layouts (based on the results from Chapter 4).

This same pattern occurred for all the graphs used in this experiment. In general the layouts with longer edges ended up losing the symmetry that they had with short edges. This shows that simply scaling the layout by adjusting the edge lengths does accurately

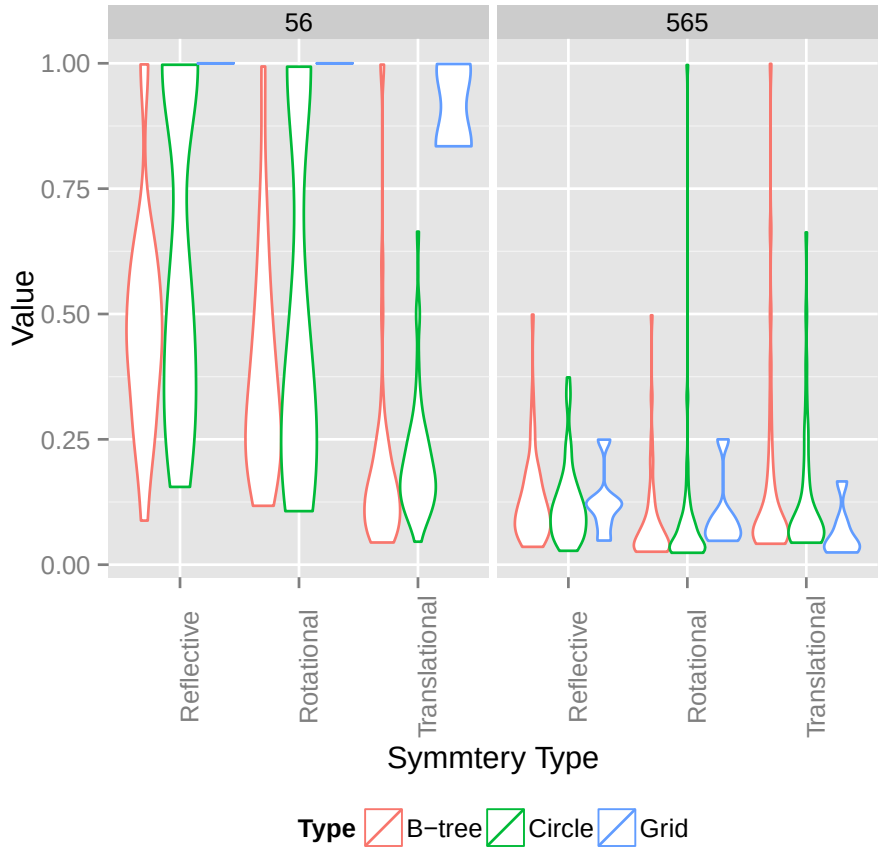
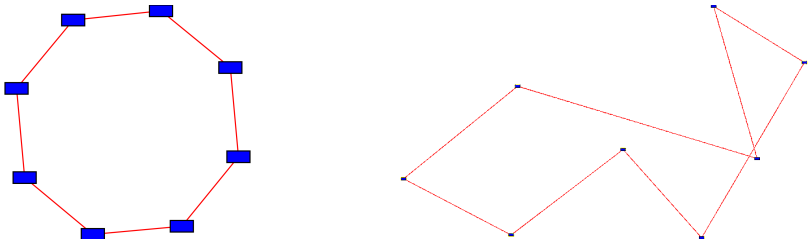


Figure 6.8: The distribution of each type of symmetry by edge length and graph type. Note that for the longer edge lengths the symmetry values are in general lower for all symmetry types.

preserve behaviour with respect to symmetry.

6.5 Symmetry User Study

The operation of the symmetry algorithm is an entirely mechanical process. This section carries out an experiment to compare the symmetry rankings produced by the algorithm with those that would be produced by a human. The goal is to test whether the judgements



(a) EEL: 56. (b) EEL: 565.

Figure 6.9: An 8 vertex circle laid out with two different expected edge lengths. Note how the layout does not form a neat circle for the longer expected edge length.

made by the algorithm when comparing two graphs are similar to the judgements that a human would make. For this purpose the relative symmetries of nine graphs are compared by means of an online survey. The experiment was approved by the Victoria University of Wellington Human Ethics Committee.

The graphs used can be found in Table 6.2 on page 127. The graphs have been sorted based on how symmetrical the algorithm rated the layouts. A graph A is more symmetrical than graph B if:

> A has at least two symmetry scores which are larger (or equal) than those for B.

Each user was presented with two randomly selected graphs and asked to click on the one they consider more symmetrical. Figure 6.10 on the facing page shows the user interface that was presented. Clicking on the chosen image advances the user to the next page. No information was collected about users as the study was completely anonymous. Participants were collected by advertising in classes and tutorials at Victoria University of Wellington (primarily Computer Science and Engineering) as well as through the authors friend network.

Each user was presented with ten screens. The first nine feature a distinct random pair of graphs. The tenth (and final) screen is just

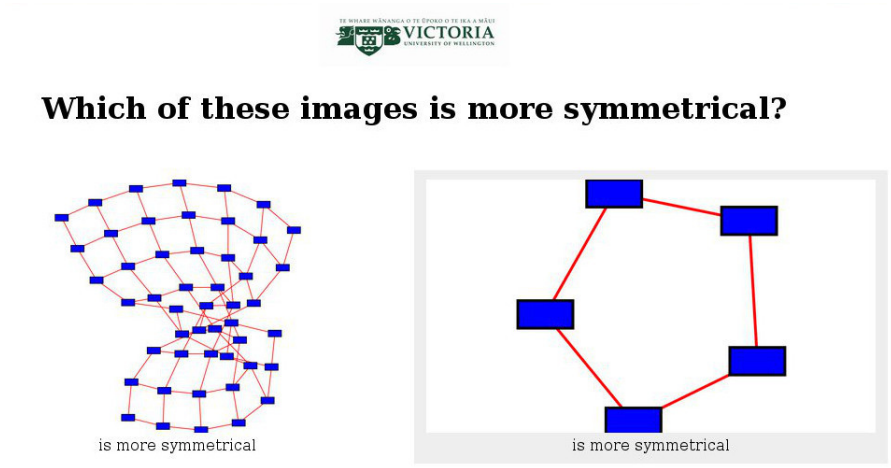


Figure 6.10: The interface shown to the user.

the third pair repeated with the left and right swapped. This can be used to test for consistency in the users' answers.

6.5.1 Results

This user study had 547 participants. Of these 508 completed the entire survey.

Integrity Checks

Before the user study results were used to evaluate the algorithm, they were sanity checked to see if they makes sense. There were two primary concerns about the collected data.

1. Do humans agree with themselves when looking at a repeated question?

2. Do humans agree with each other?

These questions are important as the algorithm cannot do better than the general human consensus.

Self Agreement To address the first concern the first check tests to see if the if users respond to questions 10 and 3 (which are the same question) the same way. The data shows that 430 of 508 (85%) of respondents answered both questions the same way. The binomial test gives a 95% confidence interval of 81% – 88% for the proportion of users that get the same result twice.

General Agreement To address the second concern — "how much do humans agree with each other?" — Figure 6.11 shows how often humans agreed with the algorithm for each question. If humans agree with each other then the agreement should be around either 0 (humans disagree with the algorithm) or 100 (humans agree with the algorithm) percent for each question. However, the graph clearly shows that there are no graphs with 0 or 100 percent agreement, and only a small number that are even close. This suggests that humans did in fact provide different results from each other. Moreover, it also shows that there are number of questions which humans answered essentially randomly (i.e. those where 50% is inside the confidence interval).

Given that the human results are not consistent, the human answers are used to create a model solution that will result in the maximum possible agreement. This model solution is approximate, it will not achieve 100% agreement with the human results, a limitation caused by the inconsistency in human responses. To determine which of two graphs is more symmetric the model solution will return the most popular answer given by humans.

The model solution was then evaluated against the full human

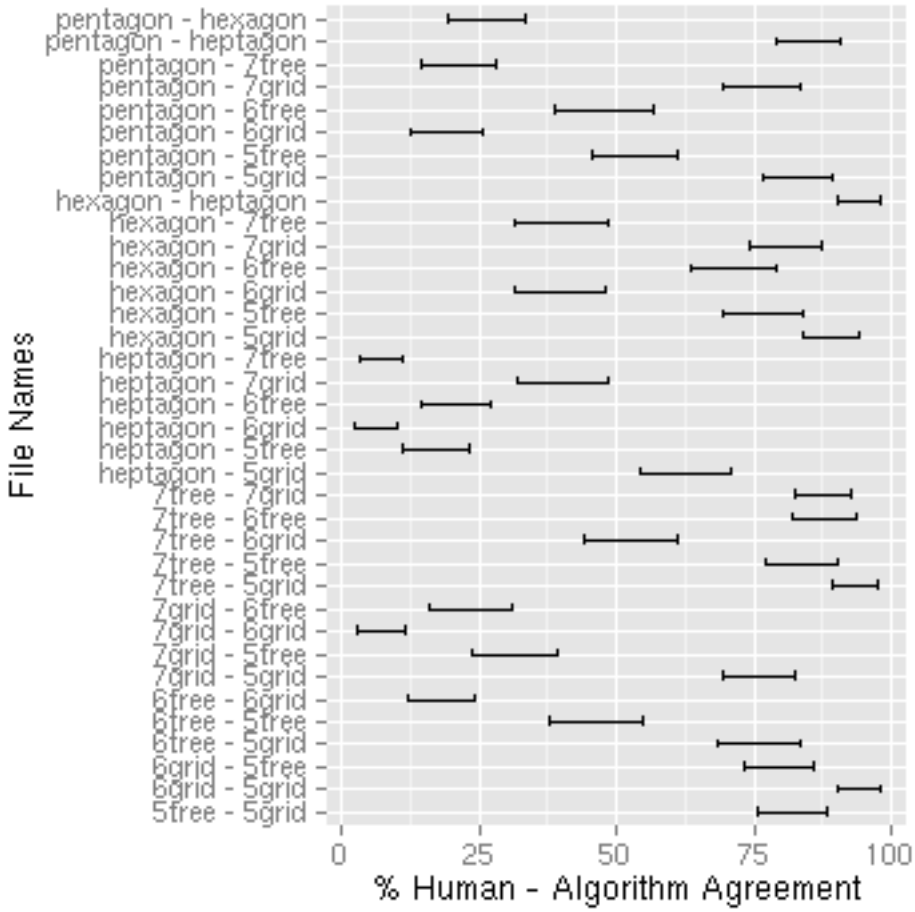


Figure 6.11: How often users agreed with the algorithm for each pair of graphs. The width of the error bar shows the 95% confidence interval from the binomial test. The closer the bar is to 50% the more humans disagreed with each other.

survey results which shows that it agrees with users 78% of the time with a 95% confidence of 77% – 79%. This is the upper bound of how well a fixed ranking can do (e.g. a ranking generated by an algorithm), given the inherent inconsistency of humans. Note that the ranking which is generated this way is not transitive, a property which is necessary for a sensible symmetry ranking. This shows one of the advantages of using an algorithm over humans for symmetry ranking: the algorithm will give results which are internally consistent, but a human may not.

Performance of the Symmetry Algorithm

Having determined that the data is reasonable, and found an upper bound on the maximum user agreement possible, the symmetry algorithm results were compared with the full human survey results. The binomial test showed that humans agreed with the algorithm 71% of the time with a 95% confidence interval of 70% – 73%. Since the maximum possible agreement is 78% (as given by the model solution), this means that the algorithm achieved 92% of the maximum possible agreement.

Closer examination of the results shows that while there have been a number of participants who have disagreed with the algorithm significantly, many participants agree on about 7–9 questions. The distribution of per-user agreement with the algorithm can be found in Figure D.1.

Looking at the specific responses of participants suggests an additional anomaly. The graph ‘heptagon’ was considered by humans (but not the algorithm) to be more symmetric than graphs such as ‘5tree’ and ‘7tree’. It is possible that this is partially because many users (unlike the algorithm) did not consider the axis of symmetry that splits vertices along a diagonal in the case of the tree graphs. Since vertices in the graph are rectangular there are only two axes

of symmetry which split them: vertical and horizontal lines passing through their centre. Had vertices been circular, this would not have been the case. However, this shows a limitation of the symmetry algorithm: it does not account for the shape of vertices and how they may be affected by axes of symmetry.

6.6 Conclusion

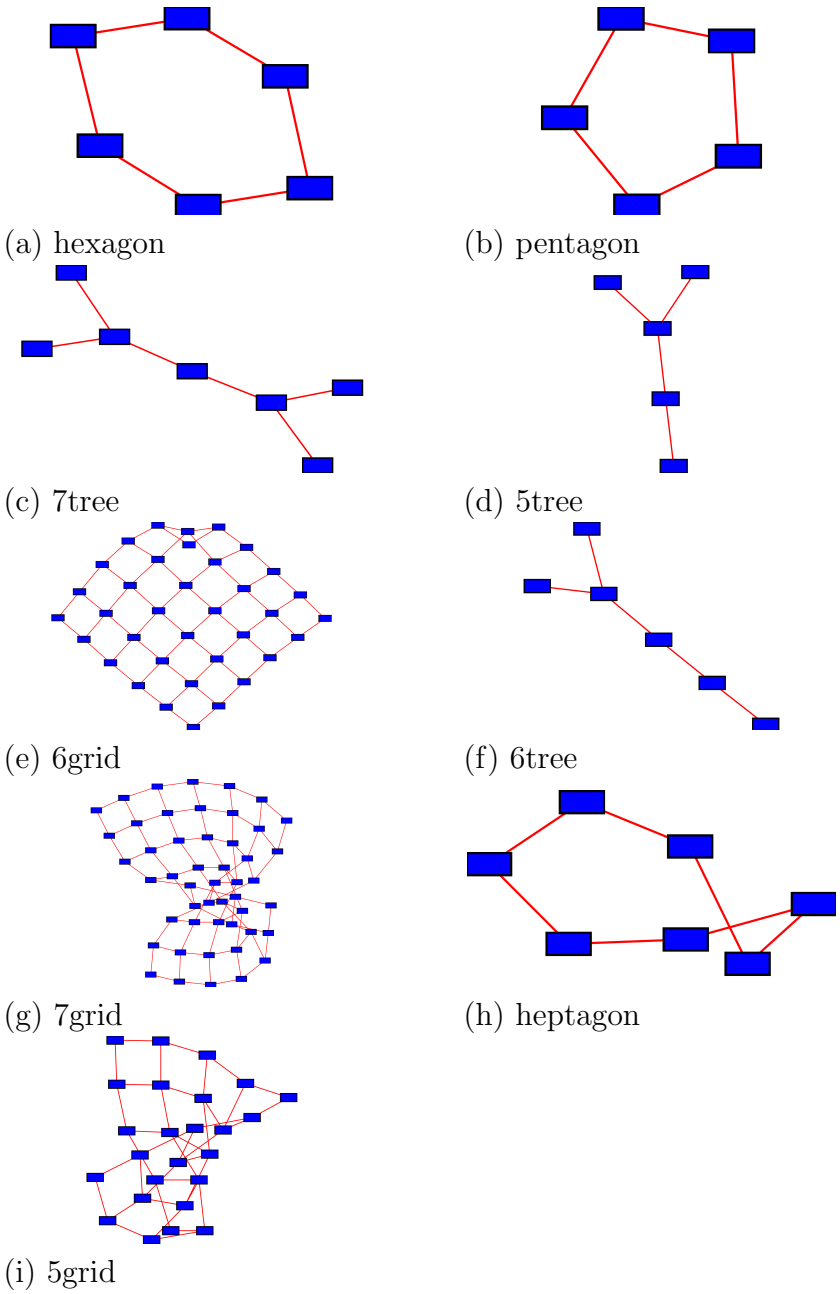
This chapter developed a novel metric to evaluate how symmetrical a given graph layout is with respect to reflective, rotational and translational symmetries. It finds that having shorter edges results in more symmetrical graph layouts. It also shows that the variability in symmetry with only base force-directed layout spans the full range of symmetry values.

One of the limitations of this chapter is that it does not say anything about whether or not it was possible to lay out the graphs used in the large experiment symmetrically. While the graphs laid out for different combinations of expected edge length and timestep were all the same, inside of those groups it is not clear how symmetrical the graphs could have potentially been. As such the results of this chapter only make claims about symmetry relative to itself.

While the symmetry algorithm is not perfect, it is shown to have 92% of the best possible performance. While it is not able to agree with humans in all cases, neither are humans able to agree with each other. This gives the algorithm a significant advantage over humans in being consistent, predictable and producing a transitive ranking.

With the large range of graphs used, this chapter looks not at what the specific values of the symmetry metric are, but rather at the distribution of values. The rationale behind this is that this allows for comparison between the distributions of symmetry for different sets of parameters. The actual values of the metric are not important.

They only serve to allow multiple graph layouts to be compared in terms of their symmetry. This means that the value of the metric cannot always be used to judge the quality of a layout without a reference layout to compare against.



(a) hexagon (b) pentagon

(c) 7tree (d) 5tree

(e) 6grid (f) 6tree

(g) 7grid (h) heptagon

(i) 5grid

Table 6.2: Graphs (and their names) used in the symmetry user study sorted by symmetry as determined by the algorithm described in this chapter. Starting from the most symmetrical in the top left corner, then decreasing going right and then row by row.

Chapter 7

Conclusion

Many different data can be represented by graphs. In many cases, like social networks, the vertices (people) have no inherent position. Graph layout is the process of finding a position to put every vertex. Ideally these layouts will also be good.

This thesis explores force-directed graph layout. It carries out four experimental studies (eight experiments) exploring the algorithm by means of metric evaluations and user experiments. Each metric based study was run over a subset of 13,720 real world graphs, many of which are anonymised graphs from AT&T. The two user experiments used different data sets. On was run over 20 graphs derived from the VAST challenge 2009 dataset, while the second was run over specially created graphs.

Experimental Study 1 The first experimental study consists of two experiments which explore the use of additional forces to reduce overlaps and edge crossings in the resulting layout. The first experiment tests a sample of 14 of the force combinations on the full set of graphs. The second experiment tests all the force combinations on a subset of 10% of the graphs. The experiments show that:

- Adding in edge label charges, charged boundaries, and increas-

ing vertex charge proportionally to its degree results in layouts that minimise the number of overlaps and occluded pixels.

- Using Hooke's law reduces the number of edge crossings for most force combinations.
- While there is a time cost for using charged edge labels, it is not so high as to make the modification too expensive.
- There is little practical difference between putting edge charges next to the edge or on the centre of the edge.

Experimental Study 2 The second experimental study looks at the effect of four input parameters — K_e, K_h, timestep and the presence of walls — on the generated layouts. It then describes a follow on experiment to show that the new results are consistent with the results from the first study. The results show that:

- Increasing the expected edge length will increase the mean edge length, but will only do so reliably for low values of timestep.
- Edge crossings do not seem to follow any clear pattern with respect to either timestep or expected edge length.
- The individual values from the parameters K_e and K_h do not matter. However, the edge length that results from their combination may have a significant effect on the layout. Ideally the edge length should be kept as short as practical, which will usually involve keeping both K_e and K_h as low as possible.
- The timestep parameter has a significant effect on graph layout and should be kept as low as possible.
- The parameter which controlled the presence of walls did not have an observed effect on the resulting graphs. It may, however,

significantly affect the aesthetics of the layout as it changes how the graph can move and spread out.

Experimental Study 3 The third experimental study is a user study. It shows that the differences in overlaps from the second study are reflected in user performance. Users had to select vertices in graphs laid out with a control and modified algorithm. The selection mechanism was free-form multi-selection. Selection was done using a PlayStation Move controller in laser mode. A 40" TV was used as the display. The results showed that the modified algorithm results in users performing less errors with no difference in speed of interaction.

Experimental Study 4 The fourth experimental study explores the range of symmetry present in graphs laid out by the force-directed algorithm. This study also involved the development of a novel metric to measure graph symmetry. The metric measures reflective, translational, and rotational symmetry.

Three experiments were carried out. The first experiment showed that the main factor in symmetry was edge length. The second experimental study showed that this was not an artefact of the symmetry metric, but rather a property of the layout algorithm. Symmetry is present for short mean edge lengths, but disappears as edges get longer. The third experiment was a user study which showed that the symmetry algorithm agreed with humans 92% of the time that agreement is possible.

Overall Overall, the thesis has the following contributions:

1. Providing basic guidelines on how certain input parameters affect the final layout. These are determined by means of a large-scale experiment.

2. Identifying the best force combinations to reduce edge crossing and overlaps in the final layout. This is determined by two large-scale experiments to evaluate the performance of different combinations of forces in force-directed layout.

3. Showing a difference in performance between users using graphs laid out by a control force-directed layout algorithm and one of the best performing force combinations identified in the previous contribution.

4. A novel symmetry algorithm for evaluating how symmetrical a graph layout is. It evaluates reflective, rotational, and translational symmetries individually. It is extended to detect multiple axes of symmetry. The algorithm is then used on the results of the parameter selection experiment to show that symmetry is affected by both the simulation granularity and the length of edges.

These contributions can be applied to the practical graph layout as a set of suggestions on how to layout graphs:

- Keep the expected edge length quite short. Exact parameters for contributing forces do not matter as much as what the resulting value is.

- Expose a timestep parameter to the algorithm and make sure that the value used is low enough that graph layout is predictable. Practically this may require guess-and-check and be implementation specific.

- Adding in extra forces can have a positive effect on the graph layout. This thesis recommends adding charged walls, vertex charge proportional to the degree and charged edge labels.

7.1 Future Work

There is a large amount of follow-up work that would be interesting to do following on from the results in this thesis. This section shall explore interesting follow-on studies by topic.

Graph Types The graphs used in this thesis do not span all possible types of graphs. Many of these experiments could be usefully repeated and have the results separated by structures inherent in the graph. This would allow for better prediction of the final outcome as more of the relevant factors could be understood. In particular, wider exploration of the effect of graph connectivity would be a possible starting point.

Evaluating Parameters This thesis only looked at the effect of four parameters. There are also a number of other parameters that this thesis kept constant that could be explored. Moreover, the additional forces described in Section 3.1.1 on page 37 could have each of their parameters explored.

This could then be extended to look at other graph layout algorithms. Ideally it would be possible to create a measure of stability of a graph layout algorithm. The stability value would indicate how important choosing the right parameters is to this algorithm. This would help users select a graph layout algorithm to use, as algorithms sensitive to parameters are less suitable for use with client systems where the graph layout should 'just work' with minimal tuning.

User Study The user experiment in Chapter 5 only looked at single selection, on a small set of graphs. Extending this study to look at other kinds of interaction tasks would provide a broader range feedback on what aspects of graph layout are important for human computer interaction.

Symmetry Detection The symmetry detection in particular has a number of limitations and weaknesses. The notable weaknesses are that:

- It cannot detect symmetry when something is almost symmetrical but beyond its error detection limits.
- It compares on an edge-by-edge basis, and cannot do abstractions where two edges take the place of a single edge.
- It does not account for the shape of a vertex.

Many of these limitations come from limits in time and scope of the project done for the thesis. Future work can address each of these problems.

Another interesting avenue with the symmetry metric is to compare force-directed layout to other graph layout algorithm to provide more evidence for or against the claim that force-directed layout promotes symmetry.

7.2 Summary

This thesis explores force-directed graph layout. It carries out four experimental studies looking at the force-directed layout and its behaviour under changes to the forces used and parameters selected, and provides additional evidence that the metric values reflect user performance in selection tasks.

The thesis provides insight into which input parameters to the algorithm are important for controlling the output. Moreover, it shows that changes to metric values can be reflected in user performance on real tasks.

Appendix A

Distributions of Graph Properties

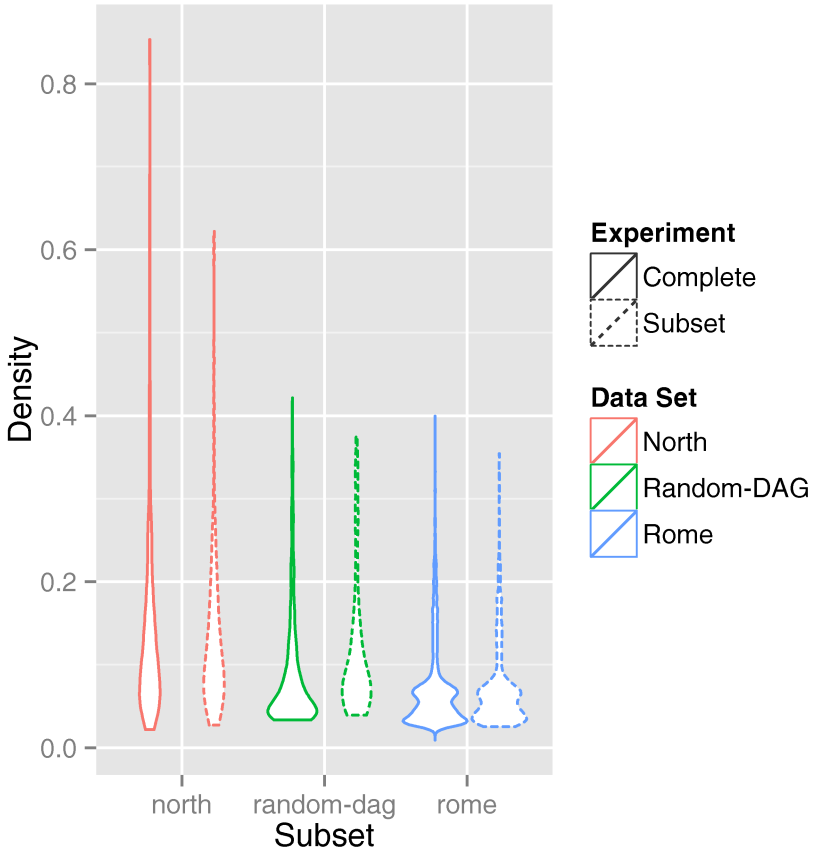


Figure A.1: Distribution of graph densities.

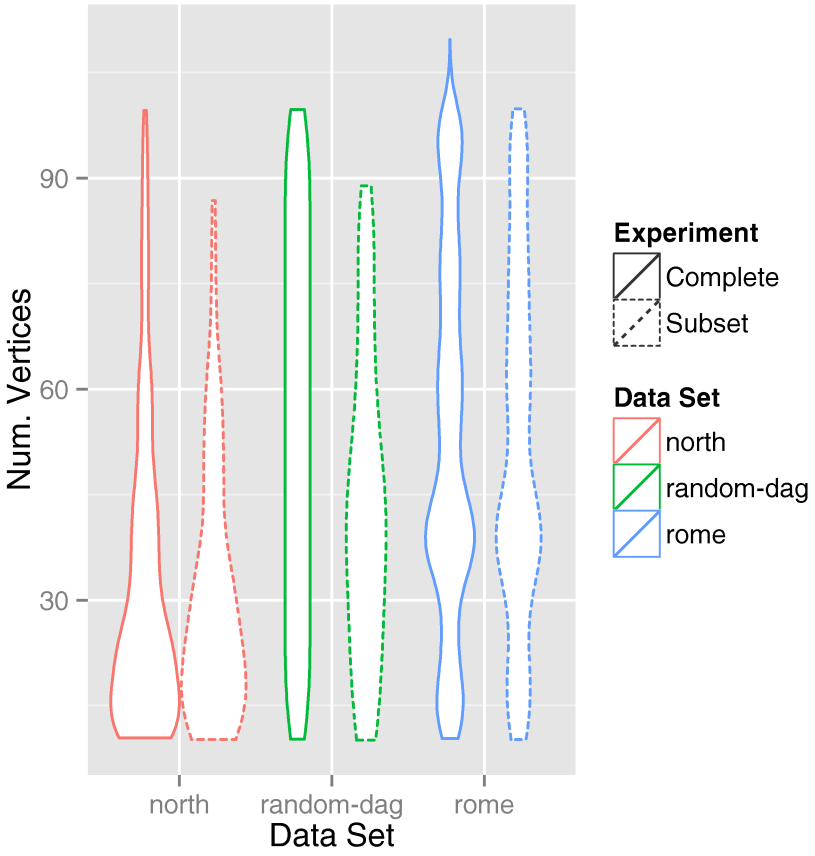


Figure A.2: Distribution of number of vertices per graph.

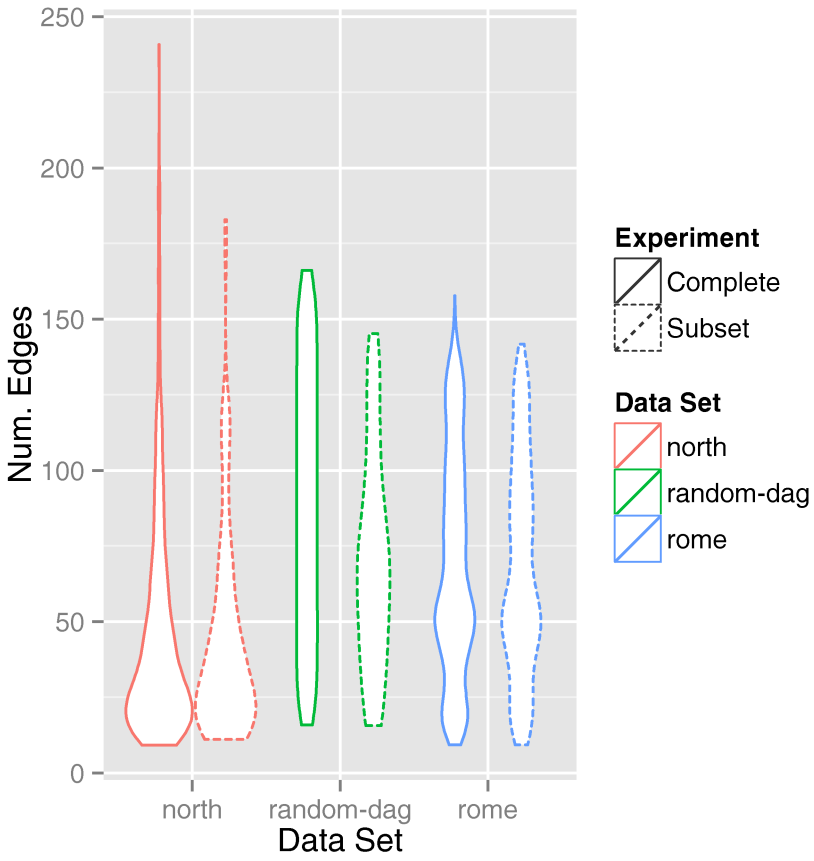


Figure A.3: Distribution of number of edges per graph.

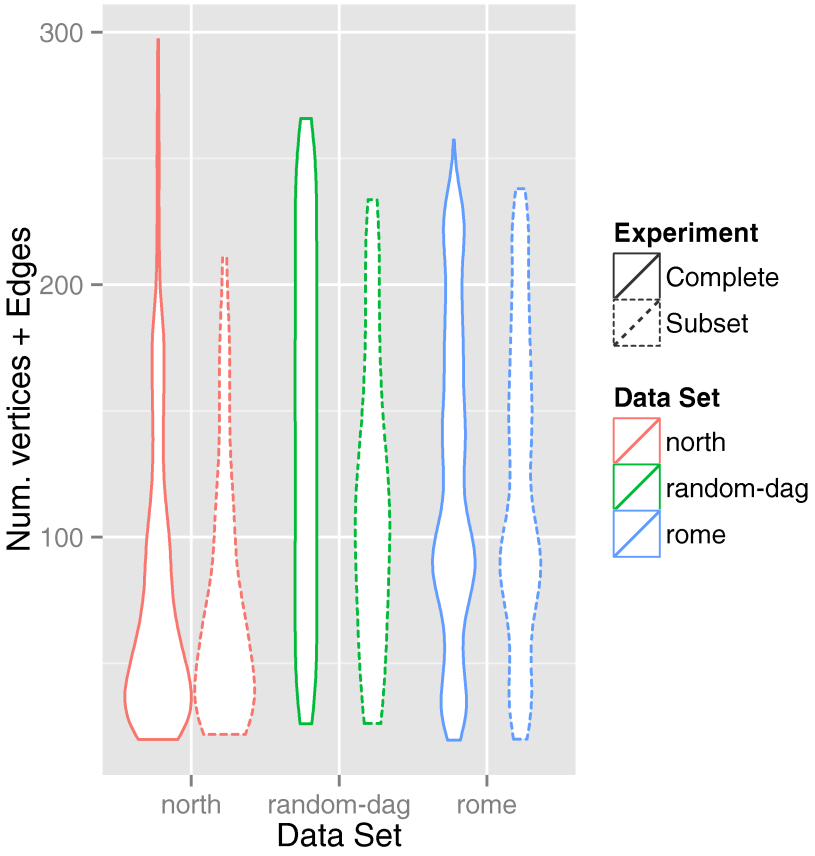


Figure A.4: Distribution of number of vertices and edges per graph.

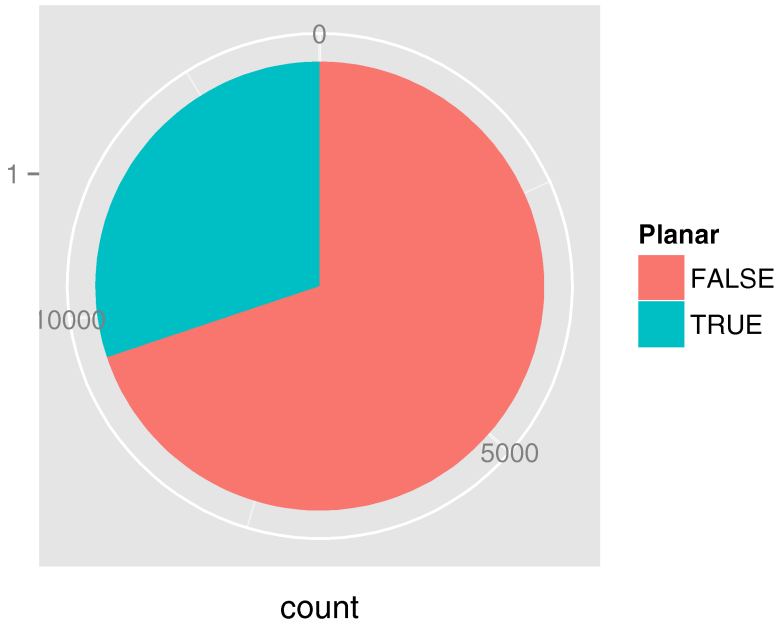


Figure A.5: Proportion of planar graphs in the complete graphdrawing.org data set.

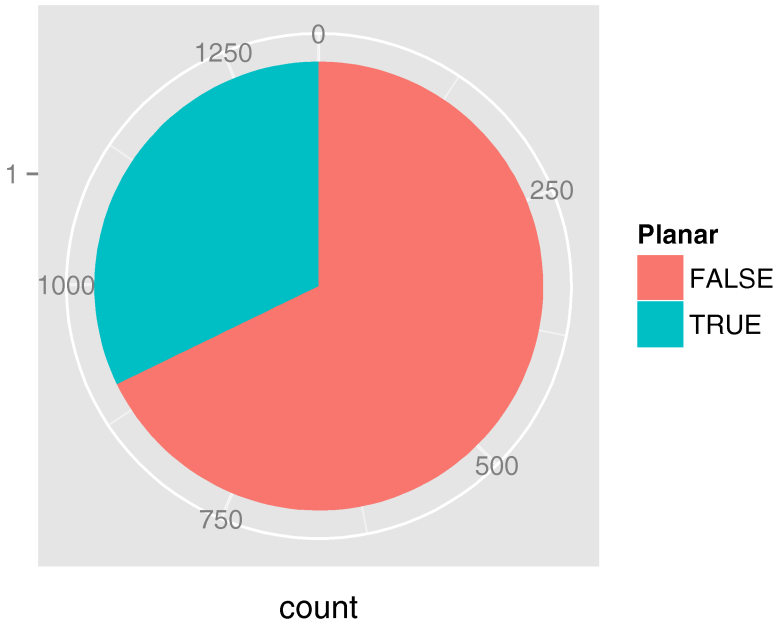


Figure A.6: Proportion of planar graphs in the subset of the graph-drawing.org data set.

Appendix B

Additional Parameter Experiment Graphs

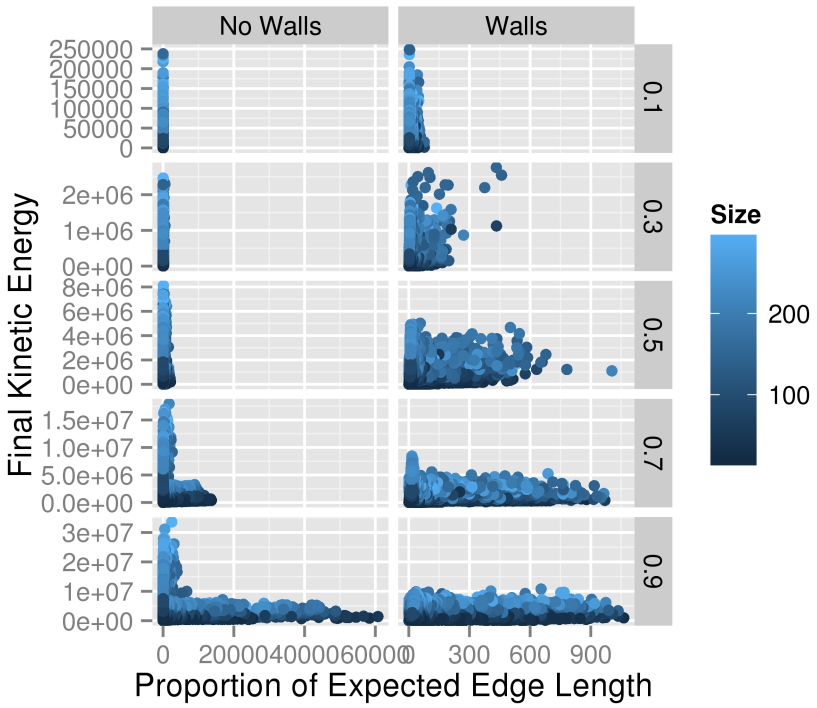


Figure B.1: The relationship between how much longer the mean edge length is over the expected edge length and the kinetic energy separated by whether there were walls. The colour shows the size of the graph. It is clear that there is no useful relationship between the two values from the largely rectangular shape of the data.

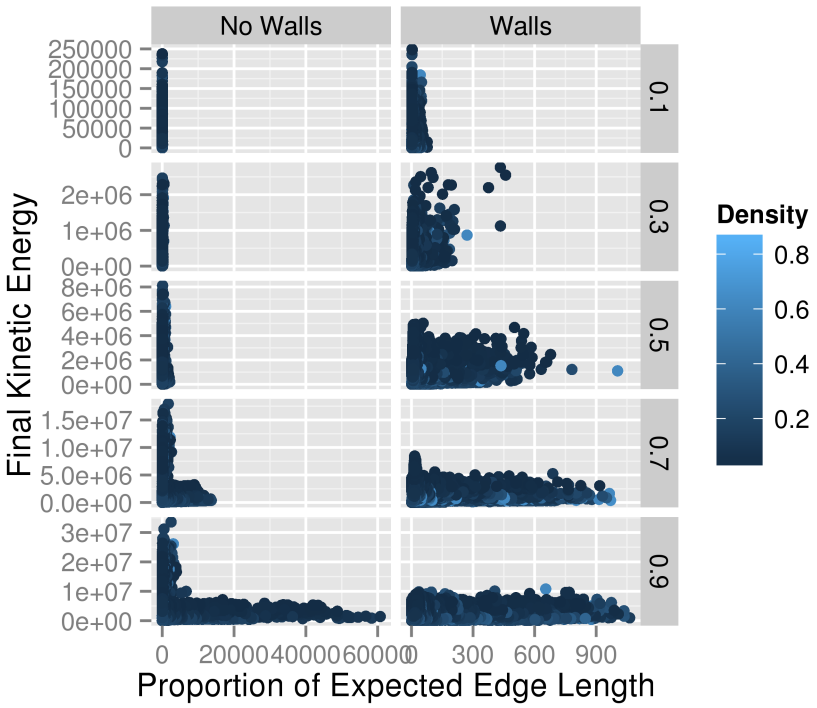


Figure B.2: The relationship between how much longer the mean edge length is over the expected edge length and the kinetic energy separated by whether there were walls. The colour shows the density of the graph. It is clear that there is no useful relationship between the two values from the largely rectangular shape of the data.

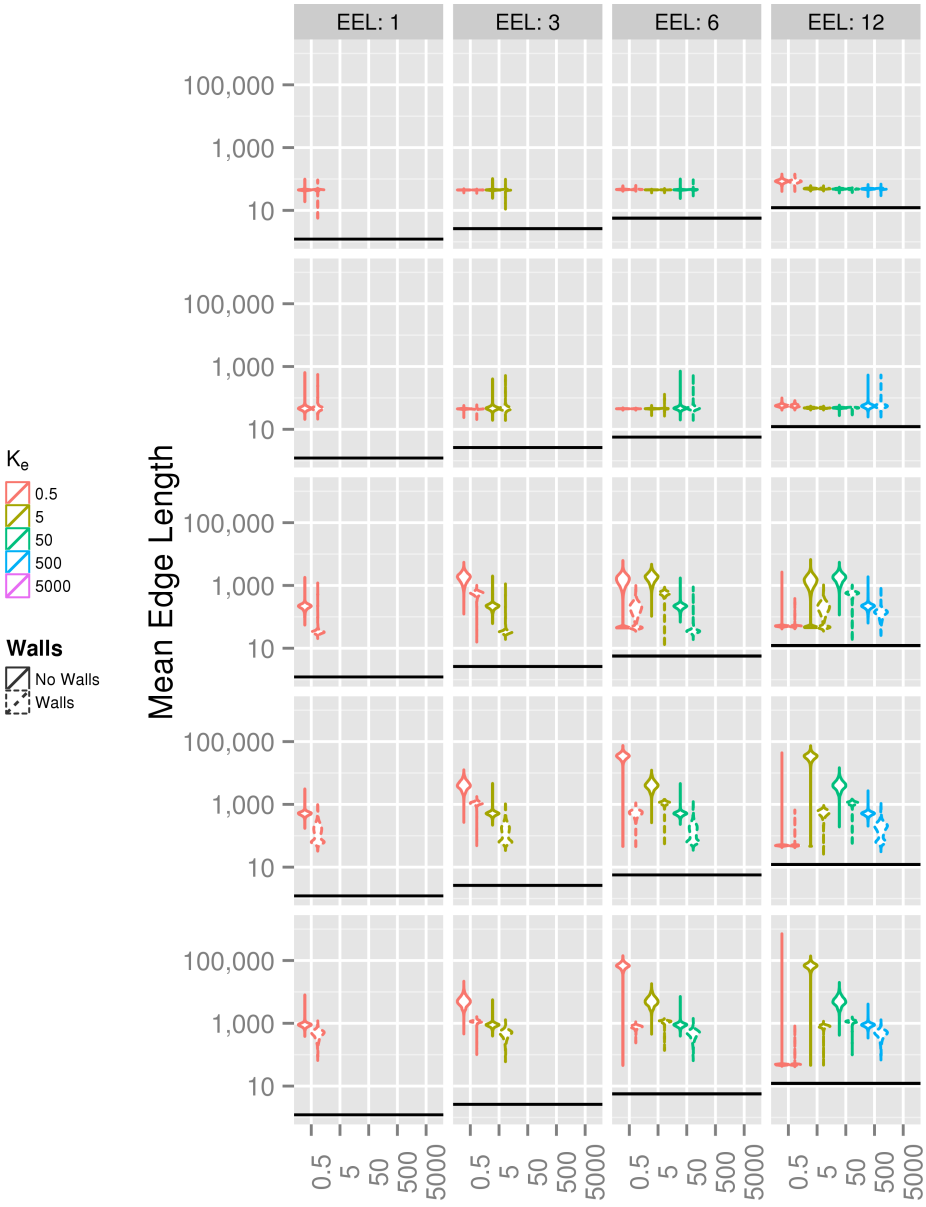
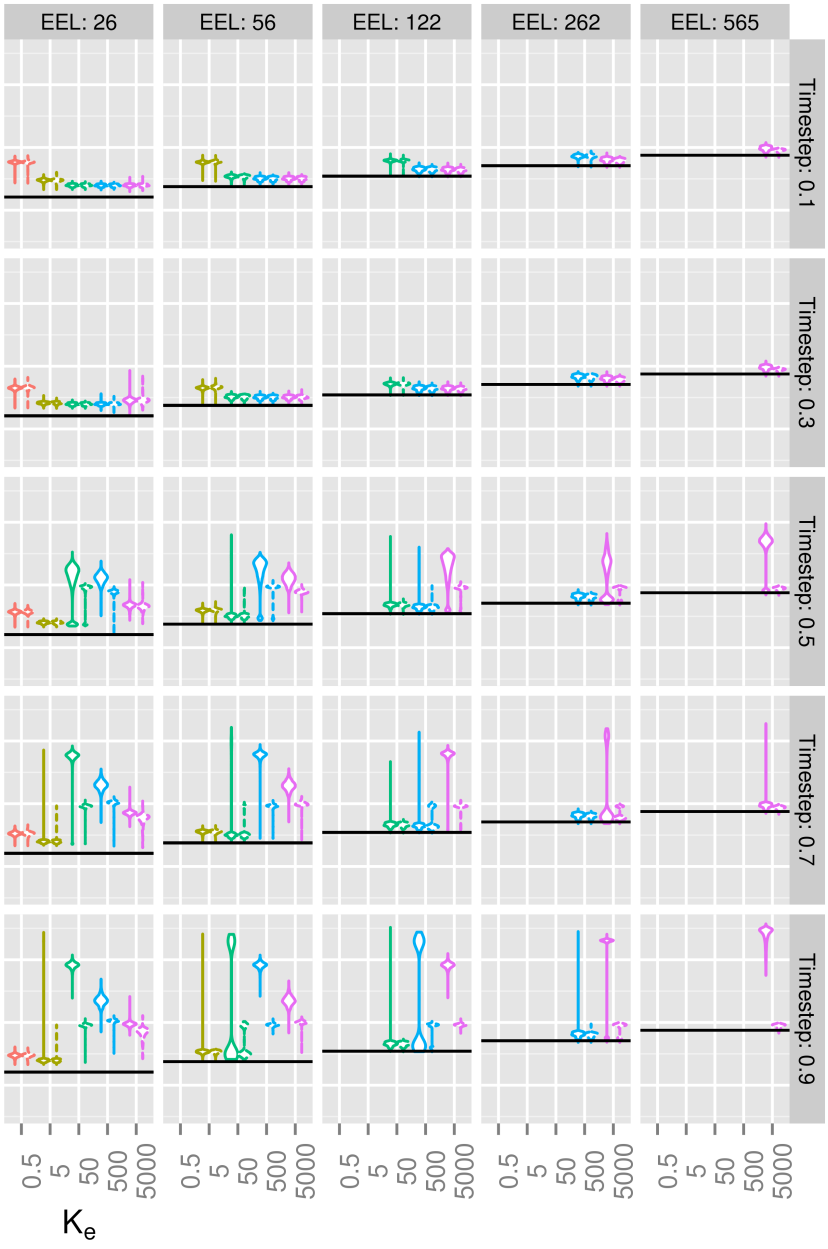


Figure B.3: This two page violin plot shows the distribution of mean edge lengths (pixels) separated by expected edge length (EEL), timestep, K_e and the presence of walls. Each value of K_e that makes up each expected edge length is shown separately and in different colours to make them easier to distinguish. Violins with dotted lines are for graphs with walls. The wider a violin is, the more graphs have that mean edge length. The horizontal black lines are the expected edge length. Note that both the x and y axes are log scales.


EEL: 26
EEL: 56
EEL: 122
EEL: 262
EEL: 565
Timestep: 0.1
Timestep: 0.3
Timestep: 0.5
Timestep: 0.7
Timestep: 0.9
0.5
5
50
500
5000
K_e

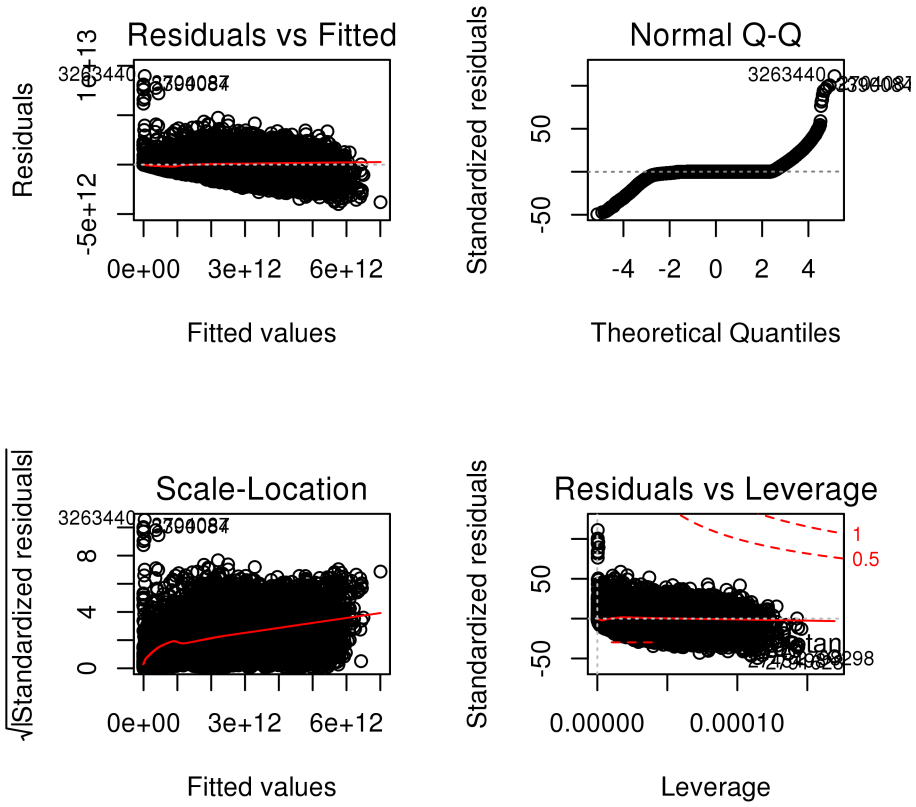


Figure B.4: Diagnostic plots for the linear model shown in Figure 4.6 on page 74. They show that the linear model does not account for all of the trend in the data, as there is still a clear pattern in the top left hand plot. The top right plot shows that the errors from the model are not normally distributed as it does not form a diagonal line. The bottom plots do not suggest any serious outliers with high leverage.

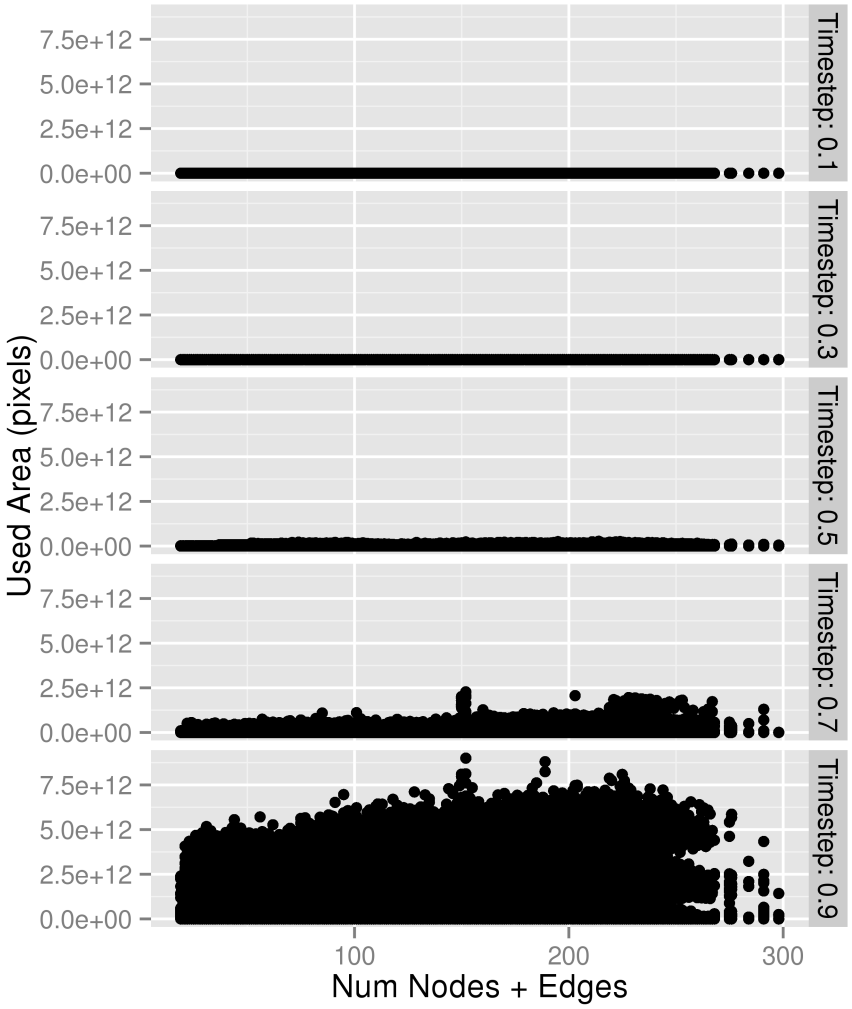


Figure B.5: The number of vertices and edges in the graph and the used area in pixels. Splitting by timestep makes it clear that timestep is a significant factor in the total area that the layout used. However, the size of the graph itself was not significant.

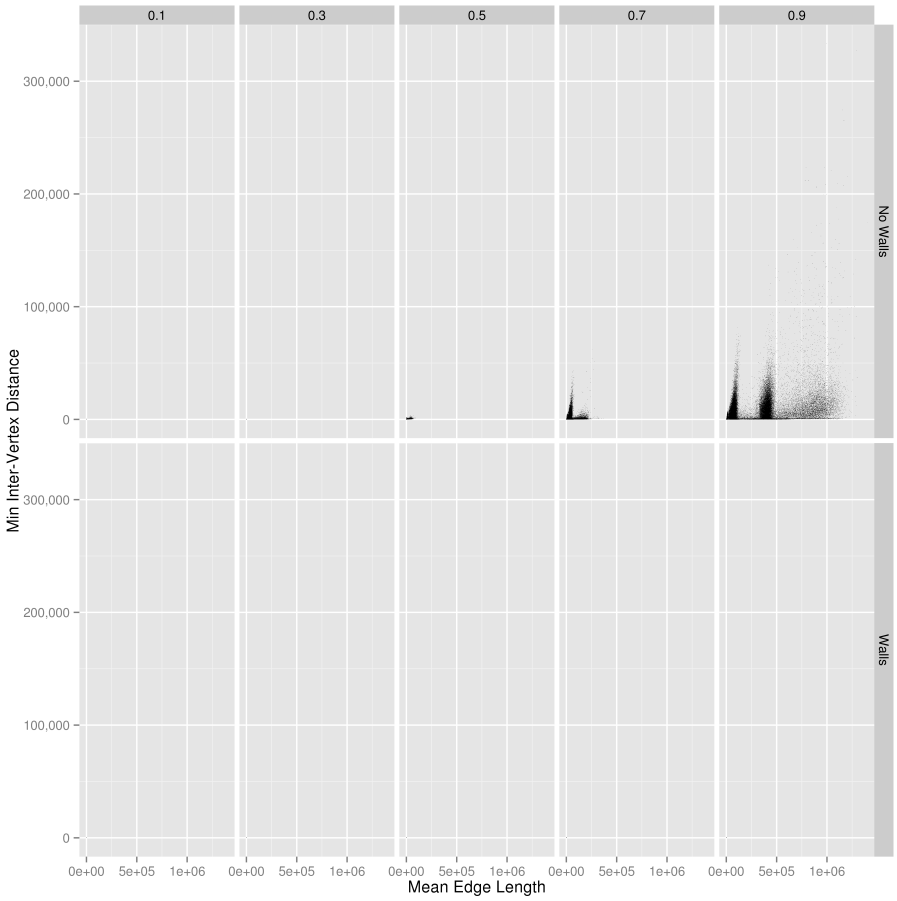


Figure B.6: The minimum vertex-vertex distance and the mean edge length are shown to be largely uncorrelated.

Appendix C

Parameter Confirmation Results

Table C.1: Medians by force from the parameter confirmation experiment in Chapter 4 on page 59.

Median	Crossings		Overlaps		Occluded Pixels
	#	%	#	%	%
H	15.00	1.89	0.93	0.08	2.00
HA	142.50	15.28	5.83	0.60	19.00
HC	19.00	2.25	1.60	0.15	5.00
HCA	98.00	12.59	10.26	1.07	27.50
HD	12.50	1.52	0.50	0.06	1.00
Continued on next page...					

Table C.1 continued from previous page.

Median	Crossings		Overlaps		Occluded Pixels
	#	%	#	%	%
HDA	124.00	14.87	3.33	0.33	10.00
HDC	15.50	1.78	0.73	0.09	2.50
HDCA	89.00	14.77	12.75	1.18	32.50
HE	11.50	1.38	0.00	0.00	0.00
HEA	13.00	1.99	0.00	0.00	0.00
HEC	12.00	1.57	0.00	0.00	0.00
HECA	15.50	2.19	0.21	0.03	1.00
HED	11.50	1.76	0.00	0.00	0.00
HEDA	40.00	4.35	0.00	0.00	0.00
HEDC	14.50	1.92	0.00	0.00	0.00
HEDCA	26.00	3.16	1.12	0.11	3.00
HG	10.00	1.57	0.00	0.00	0.00
HGA	11.00	1.82	0.00	0.00	0.00
HGC	11.50	1.89	0.00	0.00	0.00
HGCA	24.00	2.92	0.46	0.08	1.00
HGD	13.50	1.63	0.00	0.00	0.00
HGDA	29.50	3.78	0.14	0.04	1.00
HGDC	13.50	2.19	0.11	0.02	1.00
HGDCA	23.00	3.45	1.88	0.16	5.50
HW	23.00	3.49	5.29	0.43	11.50
HWA	147.50	17.71	8.87	0.88	24.00

Continued on next page...

Table C.1 continued from previous page.

Median	Crossings		Overlaps		Occluded Pixels
	#	%	#	%	%
HWC	45.50	6.95	6.22	0.64	17.50
HWCA	180.50	20.59	17.79	2.11	55.50
HWD	25.00	3.23	3.03	0.29	8.00
HWDA	135.50	16.65	3.67	0.45	12.50
HWDC	33.50	4.44	4.01	0.41	10.50
HWDCA	186.00	21.00	18.23	2.03	61.50
HWE	13.50	1.77	0.00	0.00	0.00
HWEA	55.00	6.31	0.23	0.06	2.00
HWEC	17.00	2.54	0.00	0.00	0.00
HWECA	58.50	7.30	3.65	0.37	8.00
HWED	14.50	1.94	0.00	0.00	0.00
HWEDA	94.50	12.31	0.47	0.10	3.00
HWEDC	18.00	2.07	0.00	0.00	0.00
HWEDCA	70.00	9.33	4.14	0.38	11.00
HWG	14.00	1.78	0.52	0.04	1.00
HWGA	65.50	7.20	1.52	0.14	4.50
HWGC	14.00	2.81	0.99	0.06	2.00
HWGCA	61.00	6.17	4.07	0.40	12.00
HWGD	14.50	1.94	0.24	0.03	1.00
HWGDA	81.00	11.54	2.07	0.19	6.00
HWGDC	20.00	2.63	1.04	0.07	2.00

Continued on next page...

Table C.1 continued from previous page.

Median	Crossings		Overlaps		Occluded Pixels
	#	%	#	%	%
HWGDCA	69.50	8.42	5.55	0.59	14.50
L	10.50	1.58	5.46	0.56	17.50
LA	50.50	6.67	11.62	1.38	35.00
LC	23.00	2.42	7.56	0.78	22.50
LCA	52.00	6.44	13.97	1.48	46.00
LD	8.50	1.23	3.31	0.34	7.50
LDA	65.00	8.28	4.95	0.61	16.50
LDC	16.50	2.17	6.47	0.63	17.50
LDCA	91.50	10.00	15.25	1.47	33.00
LE	9.00	1.08	0.00	0.00	0.00
LEA	9.50	1.40	0.29	0.06	1.00
LEC	12.00	1.72	0.36	0.04	1.00
LECA	12.00	2.13	1.87	0.22	4.00
LED	7.00	0.98	0.00	0.00	0.00
LEDA	14.50	1.85	0.03	0.02	1.00
LEDC	9.50	1.19	0.05	0.03	1.00
LEDCA	19.50	3.09	3.04	0.22	6.50
LG	7.00	1.12	0.95	0.10	3.00
LGA	9.50	1.26	1.23	0.10	4.00
LGC	14.00	1.74	2.14	0.20	4.50
LGCA	12.50	2.27	3.86	0.35	8.00

Continued on next page...

Table C.1 continued from previous page.

Median	Crossings		Overlaps		Occluded Pixels
	#	%	#	%	%
LGD	8.00	1.13	0.79	0.09	3.00
LGDA	13.50	1.74	1.53	0.12	3.50
LGDC	10.00	1.50	1.59	0.18	4.50
LGDCA	21.00	3.46	4.74	0.48	9.00
LW	12.00	1.59	6.91	0.81	21.50
LWA	118.50	14.37	14.42	1.91	43.00
LWC	32.00	4.06	9.87	0.98	28.50
LWCA	63.50	10.21	15.96	2.00	50.50
LWD	9.50	1.20	5.12	0.50	14.50
LWDA	103.50	14.60	6.56	0.78	21.50
LWDC	22.00	3.04	9.38	0.81	23.50
LWDCA	110.50	12.99	16.64	2.15	53.50
LWE	11.00	1.17	0.01	0.01	0.50
LWEA	36.00	4.42	1.40	0.28	7.50
LWEC	14.00	1.36	0.90	0.10	3.00
LWECA	24.50	3.40	3.99	0.46	13.00
LWED	9.50	1.26	0.04	0.02	1.00
LWEDA	76.50	8.27	1.94	0.29	8.50
LWEDC	11.00	1.53	0.59	0.08	2.00
LWEDCA	62.50	6.40	5.67	0.60	16.50
LWG	11.50	1.54	1.76	0.21	5.00

Continued on next page...

...Concluded from previous page.

Median	Crossings		Overlaps		Occluded Pixels
	#	%	#	%	%
LWGA	28.00	3.00	4.30	0.43	12.00
LWGC	15.50	2.19	3.26	0.27	7.00
LWGCA	32.50	3.53	4.54	0.53	14.00
LWGD	10.00	1.31	1.68	0.16	4.00
LWGDA	64.00	7.16	4.31	0.48	11.00
LWGDC	12.00	1.36	2.29	0.24	5.00
LWGDCA	34.50	4.56	7.42	0.69	21.50

Appendix D

Symmetry

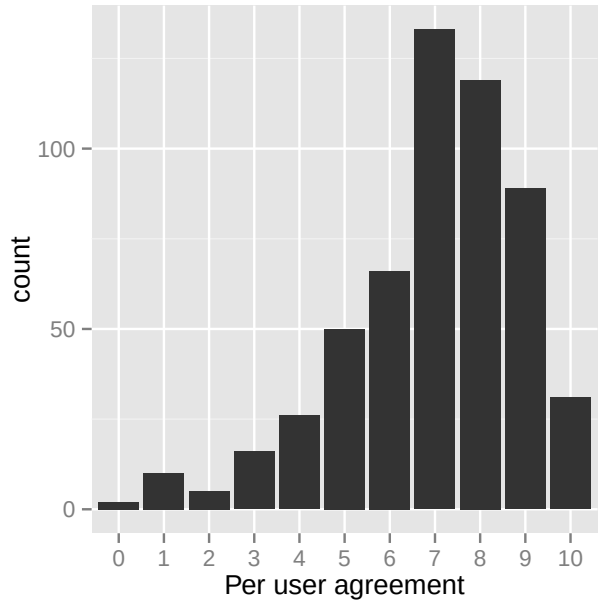


Figure D.1: How many questions each user answered the same as the algorithm. Note that the peak is at around 7 – 8.

Appendix E

Human Ethics Committee Application

Appendix D

TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

HUMAN ETHICS COMMITTEE

Application for Approval of Research Projects

Please write legibly or type if possible. Applications must be signed by supervisor (for student projects) and Head of School

Note: The Human Ethics Committee attempts to have all applications approved within three weeks but a longer period may be necessary if applications require substantial revision.

1. NATURE OF PROPOSED RESEARCH:

(a) Staff Research/ Student Research (delete one)

(b) If Student Research ..... Degree .....PhD..... Course Code ...690.....

(c) Project Title: .....Graph Layout for Human Interaction.....

.....

2. INVESTIGATORS:

(a) Principal Investigator

Name .....Roman Klapaukh .....

Email address .....roma@ecs.vuw.ac.nz.....

School/Dept/Group .....ECS.....

(b) Other Researchers Name Position

.....

.....

(c) Supervisor (in the case of student research projects)

Stuart Marshall and David Pearce .....

3. DURATION OF RESEARCH

(a) Proposed starting date for data collection 19 September 2012 .....

(**Note:** that NO part of the research requiring ethical approval may commence prior to approval being given)

(b) Proposed date of completion of project as a whole .....28 February 2015 .....

4. PROPOSED SOURCE/S OF FUNDING AND OTHER ETHICAL CONSIDERATIONS

(a) Sources of funding for the project

Please indicate any ethical issues or conflicts of interest that may arise because of sources of funding e.g. restrictions on publication of results

Small grant from the ECS for prize vouchers .....

.....

(b) Is any professional code of ethics to be followed **Y ✓ N □**

If yes, name

Association for Computing Machinery.....

(c) Is ethical approval required from any other body **Y □ N ✓**

If yes, name and indicate when/if approval will be given

.....

5. DETAILS OF PROJECT

Briefly Outline:

(a) The objectives of the project

The objectives are to compare graph layout algorithms for use with NUI (natural user interface) game controllers .....

(b) Method of data collection

We will present each participant with a paper copy of the pre and post questionnaire and a number of common graph interaction tasks where all movements and actions (of the NUI device) are recorded. There will be 20 pairs of graphs, and 5 training graphs. During the interaction they will also be prompted to answer questions about the last two graphs that they saw using the NUI device. All experiments will be conducted in CO 241, which is a software engineering research lab run by ECS.

(c) The benefits and scientific value of the project

We are investigating what properties of graph layout will improve interaction via NUI controllers.....

(d) Characteristics of the participants

We are looking to find about 70 participants. All participants will be of sound mind and able to consent (or with parental consent -14years or over) to participating in the experiment, computer literate, and aware of the task and its purpose .....

(e) Method of recruitment

Email, online forums and lists associated with ECS courses, and advertising via friend and family networks .....

(f) Payments that are to be made/expenses to be reimbursed to participants

There will be 2 -3 prizes of gift vouchers amounting to no more than $50 per person to the best performing participants. And may be confectionary for participants .....

(g) Other assistance (e.g. meals, transport) that is to be given to participants

None.....

.....

(h) Any special hazards and/or inconvenience (including deception) that participants will encounter

None.....

.....

(i) State whether consent is for (delete where not applicable):

(i) the collection of data

~~(ii) attribution of opinions or information~~

(iii) release of data to others

(iv) use for a conference report or a publication

~~(iv) use for some particular purpose (specify)~~

Any participant is able to retroactively withdraw from the study within 2 weeks of the data collection. Additionally, inclusion in the released data set is opt-in.

Attach a copy of any questionnaire or interview schedule to the application

(j) How is informed consent to be obtained (see sections 4.1, 4.5(d) and 4.8(g) of the Human Ethics Policy)

(i) the research is strictly anonymous, an information sheet is supplied and informed consent is implied by voluntary participation in filling out a questionnaire for example (include a copy of the information sheet) Y ☐ N ✓

(ii) the research is not anonymous but is confidential and informed consent will be obtained through a signed consent form (include a copy of the consent form and information sheet) Y ✓ N ☐

(iii) the research is neither anonymous or confidential and informed consent will be obtained through a signed consent form (include a copy of the consent form and information sheet) Y ☐ N ✓

(iv) informed consent will be obtained by some other method (please specify and provide details) Y ☐ N ✓

.....

.....

With the exception of anonymous research as in (i), if it is proposed that written consent will not be obtained, please explain why

Not Applicable .....

.....

(k) If the research will not be conducted on a strictly anonymous basis state how issues of confidentiality of participants are to be ensured if this is intended. (See section 4..1(e) of the Human Ethics Policy). (E.g. who will listen to tapes, see questionnaires or have access to data). Please ensure that you distinguish clearly between anonymity and confidentiality. Indicate which of these are applicable.

(i) access to the research data will be restricted to the investigator Y □ N ✓

(ii) access to the research data will be restricted to the investigator and their supervisor (student research) Y □ N ✓

(iii) all opinions and data will be reported in aggregated form in such a way that individual persons or organisations are not identifiable Y ✓ N □

(iv) Other (please specify)

We will remove all personally identifiable information, but allow other scientist to access the data.....

(l) Procedure for the storage of, access to and disposal of data, both during and at the conclusion of the research. (see section 4.12 of the Human Ethics Policy). Indicate which are applicable:

(i) all written material (questionnaires, interview notes, etc) will be kept in a locked file and access is restricted to the investigator Y ✓ N □

(ii) all electronic information will be kept in a password-protected file and access will be restricted to the investigator Y □ N ✓

(iii) all questionnaires, interview notes and similar materials will be destroyed:

(a) at the conclusion of the research Y □ N ✓

(b).....years after the conclusion of the research; or Y □ N ✓

(iv) any audio or video recordings will be returned to participants and/or electronically wiped Y □ N ✓

(v) other procedures (please specify):

All personally identifiable information will be kept in a secure online location only accessible by the supervisors and investigator. An anonymised version of the data will be publically available. There will be no audio or video recording

If data and material are not to be destroyed please indicate why and the procedures envisaged for ongoing storage and security

The physical copies that have personally identifiable tags will be destroyed after the proposed date of completion of the project as a whole. Only anonymised election copies will remain.

.....

.....

.....

(m) Feedback procedures (See section 7 of Appendix 1 of the Human Ethics Policy). You should indicate whether feedback will be provided to participants and in what form. If feedback will not be given, indicate the reasons why.

On the consent form participants will be able to provide an email address to which we will email a soft copy of the most relevant publication to come out of the experiment .....

(n) Reporting and publication of results. Please indicate which of the following are appropriate. The proposed form of publications should be indicated on the information sheet and/or consent form.

(i)	publication in academic or professional journals	Y ✓ N □
(ii)	dissemination at academic or professional conferences	Y ✓ N □
(iii)	deposit of the research paper or thesis in the University Library (student research)	Y ✓ N □
(iv)	other (please specify)	Y □ N ✓

.....

.....

Signature of investigators as listed on page 1 **(including supervisors) and Head of School**.

NB: All investigators and the Head of School must sign before an application is submitted for approval

..... Date.....

..... Date.....

..... Date.....

Head of School:

..... Date.....

APPLICATIONS FOR HUMAN ETHICS APPROVAL

CHECKLIST

- Have you read the Human Ethics Policy?
- Is ethical approval required for your project?
- Have you established whether informed consent needs to be obtained for your project?
- In the case of student projects, have you consulted your supervisor about any human ethics implications of your research?
- Has your supervisor read and signed the application?
- Have you included an information sheet for participants which explains the nature and purpose of your research, the proposed use of the material collected, who will have access to it, whether the data will be kept confidential to you, how anonymity or confidentiality is to be guaranteed?
- Have you included a written consent form?
- If not, have you explained on the application form why you do not need to get written consent?
- Are you asking participants to give consent to:
  - collect data from them
  - attribute information to them
  - release that information to others
  - use the data for particular purposes
- Have you indicated clearly to participants on the information sheet or consent form how they will be able to get feedback on the research from you (e.g. they may tick a box on the consent form indicating that they would like to be sent a summary), and how the data will be stored or disposed of at the conclusion of the research?
- Have you included a copy of any questionnaire or interview checklist you propose using?
- Has your application been seen by the head of your school or department (or the person given responsibility to consider applications on behalf of the head (see section 4.5(b) of the Human Ethics Policy).

PLEASE FORWARD YOUR COMPLETED APPLICATION FORM TO THE SECRETARY, HUMAN ETHICS COMMITTEE OR, IN THE CASE OF APPLICATIONS FROM SCHOOLS OR DEPARTMENTS WITH AN APPROVED ETHICS SUB-COMMITTEE, TO THE CONVENER OF THAT SUB-COMMITTEE

Appendix F

User Study Documents

Participant Information Sheet

Principal Investigator	**Investigator**	**Investigator**
Roman Klapaukh	Dr. Stuart Marshall	Dr. David Pearce
PhD Student	Senior Lecturer	Senior Lecturer
roma@ecs.vuw.ac.nz	stuart@ecs.vuw.ac.nz	djp@ecs.vuw.ac.nz
CO 254	CO 261	CO 231

School of Engineering and Computer Science
Victoria University of Wellington

This experiment is being undertaken towards Roman Klapaukh's PhD in the Software Engineering group at Victoria University of Wellington. The Victoria University Human Ethics Committee has granted ethics approval for this experiment. The research looks at interaction with data using NUI (natural user interface) controllers such as the Sony PlayStation Move. Participants will be asked to interact with a number of graphs using the PlayStation Move controller. The experiment is also a race, with the fastest participant receiving a prize.

Any person above 14 years of age is free to participate in the study, however, any participants below 18 years of age will require parental consent. Participants will be asked to fill out a pre- and post-test questionnaire as well as carrying out the experimental tasks. Participants are able to refuse to answer any given question, and are able to withdraw from the study without question within a fortnight of the data collection.

While the experiment will be recording the user interaction, only the location of the controller and its state will be recorded. There will be no video or audio recording of the participants. Participants will be timed and the information gathered will be assessed for accuracy, but your completed data will only be reported in aggregate form. The final dataset will also be made available in an anonymised form for other researchers, but will only contain information from participants who opt in to this dataset. The entire process should take approximately thirty to forty minutes.

If you have any questions or would like to receive further information about the project, please contact us via the details supplied above.

Thank you for your participation in the study.

Participant Consent Form

Principal Investigator	**Investigator**	**Investigator**
Roman Klapaukh	Dr. Stuart Marshall	Dr. David Pearce
PhD Student	Senior Lecturer	Senior Lecturer
roma@ecs.vuw.ac.nz	stuart@ecs.vuw.ac.nz	djp@ecs.vuw.ac.nz
CO 254	CO 261	CO 231

School of Engineering and Computer Science
Victoria University of Wellington

Please tick and sign

I have read the information sheet supplied and the researchers have satisfactorily answered any questions I may have had.

I consent to taking part in this study and understand that I have the right to withdraw from this experiment within two weeks of data collection by emailing the researchers accordingly.

I consent to the researchers using movement and selection data and answers from my participation in a non-identifiable way in a PhD thesis and related conference / journal papers

☐ I consent to my data being present in a publicly available anonymised version of the final results.

☐ I would like to be emailed a softcopy of the report at .....

Signed: .....

Signed (parent or guardian if under 18): .....

Date : .....

Initial Questionnaire

Principal Investigator	**Investigator**	**Investigator**
Roman Klapaukh	Dr. Stuart Marshall	Dr. David Pearce
PhD Student	Senior Lecturer	Senior Lecturer
roma@ecs.vuw.ac.nz	stuart@ecs.vuw.ac.nz	djp@ecs.vuw.ac.nz
CO 254	CO 261	CO 231

School of Engineering and Computer Science
Victoria University of Wellington

Name : .....

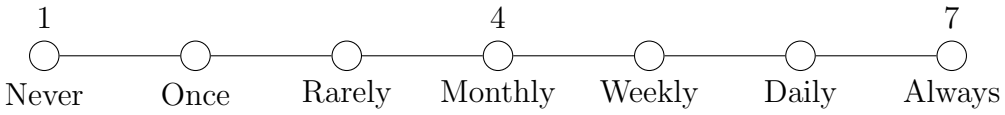
Age: .....

Gender: .....

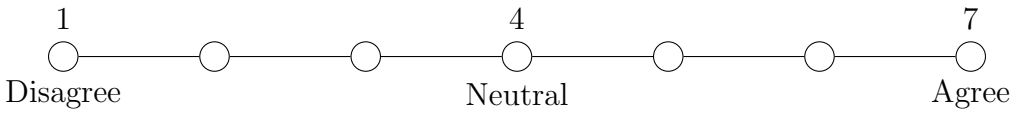
Height: .....

Please tick the appropriate circle:

How often do you use a PlayStation Move or Nintendo Wii remote:



I feel comfortable using the PlayStation Move or Nintendo Wii remote:



Final Questionnaire

Principal Investigator	**Investigator**	**Investigator**
Roman Klapaukh	Dr. Stuart Marshall	Dr. David Pearce
PhD Student	Senior Lecturer	Senior Lecturer
roma@ecs.vuw.ac.nz	stuart@ecs.vuw.ac.nz	djp@ecs.vuw.ac.nz
CO 254	CO 261	CO 231

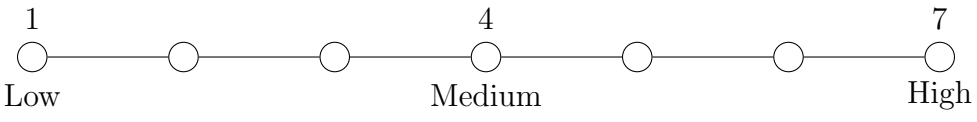
School of Engineering and Computer Science
Victoria University of Wellington

For each pair circle which of the two options was the biggest problem:

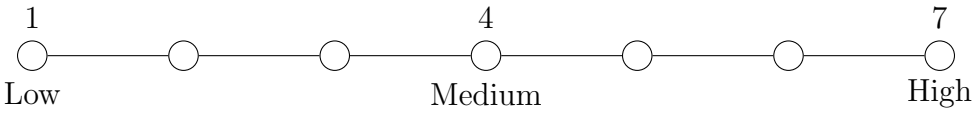
Effort / Getting it right	Time / Frustration
Time / Effort	Physical demand / Frustration
Getting it right / Frustration	Physical Demand / Time
Physical Demand / Getting it right	Time / Mental Demand
Frustration / Effort	Getting it right / Mental Demand
Getting it right / Time	Mental Demand / Effort
Mental Demand / Physical Demand	Effort / Physical demand
Frustration / Mental Demand	

Please tick the appropriate circle:

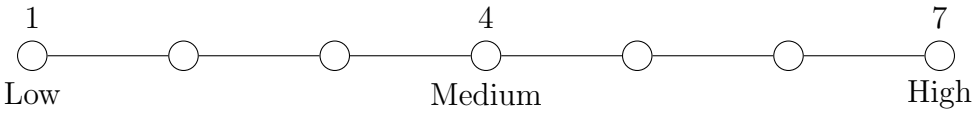
How mentally demanding was the task:



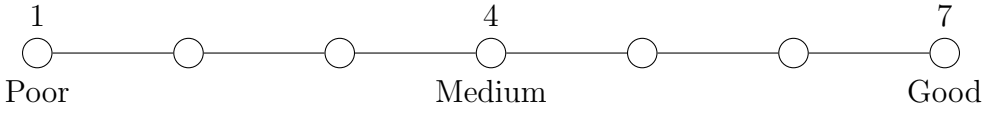
How physically demanding was the task:



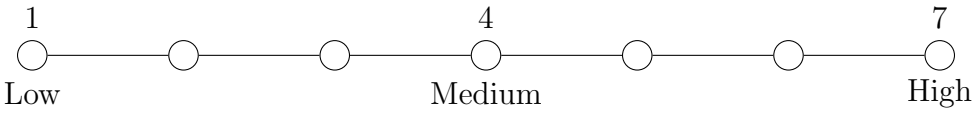
Rate the time stress:



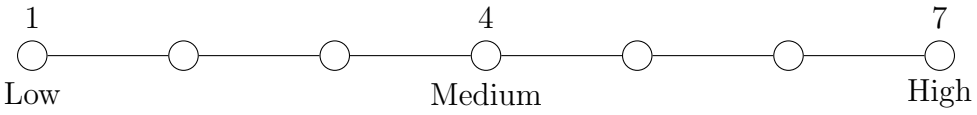
How would you rate your performance:



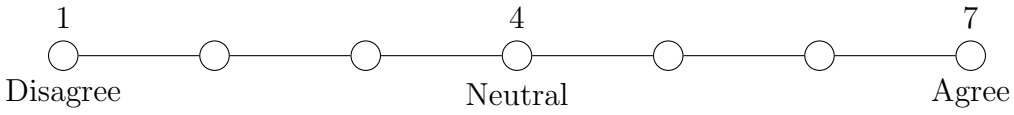
How much total effort did this require:



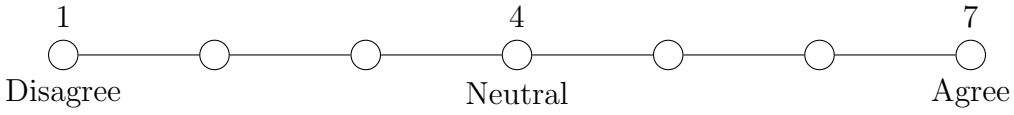
How frustrating did you find the task:



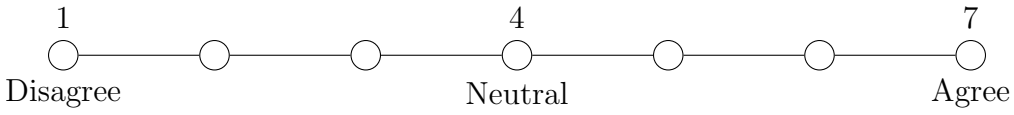
The system was fun to use:



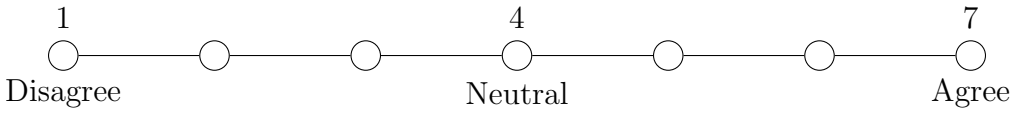
The system was novel to use:



I would use this system again:



The system did what I wanted:



If you have any comments you wish to make, please write them here:

Bibliography

[1] D. Hotson, "Springy.js." http://getspringy.com/ accessed: 20 Nov 2012. Cited on pages 1 and 11.

[2] AT&T, "Graphviz." www.graphviz.org/ accessed: 20 Nov 2012. Cited on pages 1 and 11.

[3] J. Heer, S. K. Card, and J. A. Landay, "Prefuse: a toolkit for interactive information visualization," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, (New York, NY, USA), pp. 421–430, ACM, 2005. Cited on pages 1 and 11.

[4] P. Eades, "A Heuristic for Graph Drawing," *Congressus Numeratum*, vol. 42, pp. 149–160, 1984. Cited on pages 1, 2, 10, 12, 14, 15, 16, 17, 21, 40, 42, 81, and 99.

[5] H. C. Purchase, "Metrics for Graph Drawing Aesthetics," *JVLC*, vol. 13, no. 5, pp. 501–516, 2002. Cited on pages 1, 20, 21, 44, 72, and 99.

[6] K. M. Hall, "An r-Dimensional Quadratic Placement Algorithm," *Management Science*, vol. 17, no. 3, pp. pp. 219–229, 1970. Cited on pages 2 and 9.

[7] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999. Cited on pages 2, 10, 12, 15, and 42.

[8] D. Harel and Y. Koren, "Drawing graphs with non-uniform vertices," in *Proc. of the Working Conf. on AVI*, AVI 2002, (New York), pp. 157–166, ACM, 2002. Cited on pages 2 and 17.

[9] R. Davidson and D. Harel, "Drawing Graphs Nicely Using Simulated Annealing," *ACM TOG*, vol. 15, no. 4, pp. 301–331, 1996. Cited on pages 2 and 37.

[10] J.-H. Chuang, C.-C. Lin, and H.-C. Yen, "Drawing graphs with nonuniform nodes using potential fields," in *Proc. GD, LNCS*, pp. 460–465, 2004. Cited on pages 2, 14, and 42.

[11] Y. Hu, "Visualizing Graphs with Node and Edge Labels," *CoRR*, vol. abs/0911.0626, 2009. Cited on pages 2 and 18.

[12] Y. Frishman and A. Tal, "Uncluttering Graph Layouts Using Anisotropic Diffusion and Mass Transport," *IEEE TVCG*, vol. 15, no. 5, pp. 777–788, 2009. Cited on pages 2, 11, and 18.

[13] X. Huang, A. S. M. Sajeev, and W. Lai, "A Scalable Algorithm for Adjusting Node-Node Overlaps," in *CGIV*, pp. 43–48, IEEE Computer Society, 2006. Cited on pages 2 and 18.

[14] R. Klapaukh, D. J. Pearce, and S. Marshall, "Towards a vertex and edge label aware force directed layout algorithm," in *Thirty-Seventh Australasian Computer Science Conference (ACSC 2014)* (B. Thomas and D. Parry, eds.), vol. 147 of *CRPIT*, (Auckland, New Zealand), pp. 29–37, ACS, 2014. Cited on pages 4 and 36.

[15] U. Brandes and D. Wagner, "Analysis and Visualization of Social Networks," in *Graph Drawing Software* (M. Jünger and P. Mutzel, eds.), Mathematics and Visualization, pp. 321–340, Springer Berlin Heidelberg, 2004. Cited on page 7.

[16] Autodesk, Inc., "Autodesk Maya 2013 API Documentation: Querying the Scene Graph," 2013. http://docs.autodesk.com/MAYAUL/2013/ENU/Maya-API-Documentation/index.html?url=files/GUID-0B85C721-C3C6-47D7-9D85-4F27B787ABB6.htm,topicNumber=d30e5360 accessed: 03/04/2013. Cited on page 7.

[17] T. Kamada and S. Kawai, "An Algorithm for Drawing General Undirected Graphs," *Information Processing Letters*, vol. 31, no. 1, pp. 7–15, 1989. Cited on pages 10, 14, and 42.

[18] L. J. Groves, Z. Michalewicz, P. V. Elia, and C. Z. Janikow, "Methodologies for intelligent systems, 5," ch. Genetic algorithms for drawing directed graphs, pp. 268–276, New York, NY, USA: Elsevier North-Holland, Inc., 1990. Cited on page 10.

[19] T. Dwyer, B. Lee, D. Fisher, K. Quinn, P. Isenberg, G. Robertson, and C. North, "A Comparison of User-Generated and Automatic Graph Layouts," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 15, no. 6, pp. 961–968, 2009. Cited on pages 10 and 23.

[20] W. Li, P. Eades, and N. S. Nikolov, "Using Spring Algorithms to Remove Node Overlapping," in *APVIS*, vol. 45 of *CRPIT*, pp. 131–140, 2005. Cited on pages 10, 14, 15, 18, 37, and 42.

[21] P. Droste, W. Wiechert, and K. Nh, "Semi-automatic drawing of metabolic networks," *Information Visualization*, vol. 11, no. 3, pp. 171–187, 2012. Cited on page 10.

[22] X. Yuan, L. Che, Y. Hu, and X. Zhang, "Intelligent Graph Layout Using Many Users' Input," *IEEE Trans. Vis. Comput. Graph.*, vol. 18, no. 12, pp. 2699–2708, 2012. Cited on page 10.

[23] X. Huang, W. Lai, A. S. M. Sajeev, and J. Gao, "A new algorithm for removing node overlapping in graph visualization," *Information Sciences*, vol. 177, no. 14, pp. 2821–2844, 2007. Cited on page 11.

[24] D. Archambault, T. Munzner, and D. Auber, "TopoLayout: Multilevel Graph Layout by Topological Features," *IEEE TVCG*, vol. 13, pp. 305–317, Mar. 2007. Cited on pages 11 and 37.

[25] G. Bartel, C. Gutwenger, K. Klein, and P. Mutzel, "An experimental evaluation of multilevel layout methods," in *Graph Drawing* (U. Brandes and S. Cornelsen, eds.), vol. 6502 of *Lecture Notes in Computer Science*, pp. 80–91, Springer Berlin Heidelberg, 2011. Cited on page 11.

[26] P. C. Wong, H. Foote, P. Mackey, G. C. Jr., H. J. Sofia, and J. Thomas, "A dynamic multiscale magnifying tool for exploring large sparse graphs," *Information Visualization*, vol. 7, no. 2, pp. 105–117, 2008. Cited on pages 11 and 14.

[27] P. C. Wong, K. Schneider, P. Mackey, H. Foote, G. C. Jr., R. Guttromson, and J. Thomas, "A novel visualization technique for electric power grid analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 3, pp. 410–423, 2009. Cited on page 11.

[28] C. Dunne and B. Shneiderman, "Motif simplification: improving network visualization readability with fan, connector, and clique glyphs," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, (New York, NY, USA), pp. 3247–3256, ACM, 2013. Cited on page 11.

[29] A. Morrison, G. Ross, and M. Chalmers, "Fast multidimensional scaling through sampling, springs and interpolation," *Information Visualization*, vol. 2, no. 1, pp. 68–77, 2003. Cited on page 11.

[30] T. M. J. Fruchterman and E. M. Reingold, "Graph Drawing by Force-directed Placement," *Software - Practice and Experience*, vol. 21, no. 11, pp. 1129–1164, 1991. Cited on pages 12, 13, 14, and 42.

[31] P. Kumar, K. Zhang, and Y. Wang, "Visualization of Clustered Directed Acyclic Graphs without Node Overlapping," in *IV*, pp. 38–43, IEEE Comp. Soc., 2008. Cited on pages 12, 14, 18, 40, and 42.

[32] C.-C. Lin, H.-C. Yen, and J.-H. Chuang, "Drawing graphs with nonuniform nodes using potential fields," *J. Vis. Lang. Comput.*, vol. 20, no. 6, pp. 385–402, 2009. Cited on pages 12 and 18.

[33] U. Brandes, V. Kääb, A. Löh, D. Wagner, and T. Willhalm, "Dynamic WWW Structures in 3D," *JGAA*, vol. 4, no. 3, pp. 183–191, 2000. Cited on pages 14 and 42.

[34] E. R. Gansner and S. C. North, "Improved Force-Directed Layouts," in *GD* (S. Whitesides, ed.), vol. 1547 of *LNCS*, pp. 364–373, Springer, 1998. Cited on pages 14, 18, and 42.

[35] X. Wang and I. Miyamoto, "Generating Customized Layouts," in *GD* (F.-J. Brandenburg, ed.), vol. 1027 of *LNCS*, pp. 504–515, Springer, 1995. Cited on pages 14, 17, and 42.

[36] R. D. Knight, *Physics For Scientists and Engineers: A Strategic Approach*. Pearson — Addison-Wesley, 2004. Cited on page 15.

[37] T. Dwyer, K. Marriott, and P. J. Stuckey, "Fast Node Overlap Removal," in *GD* (P. Healy and N. S. Nikolov, eds.), vol. 3843 of *LNCS*, pp. 153–164, 2005. Cited on page 18.

[38] IEEE, "VAST Challenge," 2009. www.cs.umd.edu/hcil/VASTchallenge09 accessed: 24 April 2013. Cited on pages 19 and 82.

[39] "graphdrawing.org." http://graphdrawing.org/data.html accessed: 18 May 2011. Cited on pages 19, 36, 41, and 63.

[40] P. F. Cortese, "GDToolkit." http://www.dia.uniroma3.it/~gdt/gdt4/index.php accessed: 11 Oct 2013. Cited on page 19.

[41] U. Brandes, M. Eiglsperger, I. Herman, M. Himsolt, and M. Marshall, "GraphML progress report structural layer proposal," in *Graph Drawing* (P. Mutzel, M. Jünger, and S. Leipert, eds.), vol. 2265 of *Lecture Notes in Computer Science*, pp. 501–512, Springer Berlin Heidelberg, 2002. Cited on page 20.

[42] H. C. Purchase, C. Pilcher, and B. Plimmer, "Graph Drawing Aesthetics - Created by Users, Not Algorithms," *IEEE Trans. Vis. Comput. Graph.*, vol. 18, no. 1, pp. 81–92, 2012. Cited on pages 20, 21, and 44.

[43] H. C. Purchase, D. Carrington, and J.-A. Allder, "Empirical Evaluation of Aesthetics-based Graph Layout," *Empirical Soft-*

ware Engineering, vol. 7, pp. 233–255, Sept. 2002. Cited on pages 20 and 99.

[44] J. Blythe, C. McGrath, and D. Krackhardt, "The effect of graph layout on inference from social network data," in *Graph Drawing* (F. Brandenburg, ed.), vol. 1027 of *Lecture Notes in Computer Science*, pp. 40–51, Springer Berlin Heidelberg, 1996. Cited on page 20.

[45] D. S. Johnson, "The NP-completeness column: An ongoing guide," *J. of Alg.*, vol. 3, no. 2, pp. 182–195, 1982. Cited on page 21.

[46] H. Purchase, R. Cohen, and M. James, "Validating graph drawing aesthetics," in *Graph Drawing* (F. Brandenburg, ed.), vol. 1027 of *Lecture Notes in Computer Science*, pp. 435–446, Springer Berlin Heidelberg, 1996. Cited on pages 21 and 99.

[47] M. Ben-Chen, A. Butscher, J. Solomon, and L. J. Guibas, "On discrete killing vector fields and patterns on surfaces," *Comput. Graph. Forum*, vol. 29, no. 5, pp. 1701–1711, 2010. Cited on page 22.

[48] G. Loy and J.-O. Eklundh, "Detecting symmetry and symmetric constellations of features," in *Computer Vision – ECCV 2006* (A. Leonardis, H. Bischof, and A. Pinz, eds.), vol. 3952 of *Lecture Notes in Computer Science*, pp. 508–521, Springer Berlin Heidelberg, 2006. Cited on pages 22, 100, 101, 104, 105, 106, 107, and 110.

[49] D. Reisfeld, H. Wolfson, and Y. Yeshurun, "Context-free attentional operators: The generalized symmetry transform," *International Journal of Computer Vision*, vol. 14, no. 2, pp. 119–130, 1995. Cited on pages 22 and 107.

[50] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, pp. 11–15, Jan. 1972. Cited on pages 22 and 104.

[51] D. G. Lowe, "Object recognition from local scale-invariant features," in *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157, 1999. Cited on pages 22 and 105.

[52] D. A. Bowman, E. Kruijff, J. J. LaViola, and I. Poupyrev, *3D User Interfaces: Theory and Practice*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 2004. Cited on pages 22 and 80.

[53] J. J. J. LaViola and R. L. Marks, "An introduction to 3D spatial interaction with video game motion controllers," in *ACM SIGGRAPH 2010 Courses*, SIGGRAPH '10, (New York, NY, USA), pp. 2:1–2:78, ACM, 2010. Cited on page 22.

[54] M. Karam and M. M. Schraefel, "A taxonomy of gestures in human computer interactions," technical report, University of Southampton, 2005. Cited on page 23.

[55] S. Schmidt, M. A. Nacenta, R. Dachselt, and S. Carpendale, "A set of multi-touch graph interaction techniques," in *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, (New York, NY, USA), pp. 113–116, ACM, 2010. Cited on page 23.

[56] P. O. Kristensson, O. Arnell, A. Björk, N. Dahlbäck, J. Pennerup, E. Prytz, J. Wikman, and N. Åström, "InfoTouch: an explorative multi-touch visualization interface for tagged photo collections," in *Proceedings of the 5th Nordic conference on Human-computer interaction: building bridges*, NordiCHI '08,

(New York, NY, USA), pp. 491–494, ACM, 2008. Cited on page 23.

[57] H. Benko, A. D. Wilson, and P. Baudisch, "Precise selection techniques for multi-touch screens," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, (New York, NY, USA), pp. 1263–1272, ACM, 2006. Cited on pages 23 and 24.

[58] D. Vogel and P. Baudisch, "Shift: a technique for operating pen-based interfaces using touch," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, (New York, NY, USA), pp. 657–666, ACM, 2007. Cited on page 24.

[59] J. S. Pierce, A. S. Forsberg, M. J. Conway, S. Hong, R. C. Zeleznik, and M. R. Mine, "Image plane interaction techniques in 3D immersive environments," in *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, I3D '97, (New York, NY, USA), pp. 39–ff., ACM, 1997. Cited on page 24.

[60] M. R. Mine, "Virtual environment interaction techniques," tech. rep., University of North Carolina, Chapel Hill, NC, USA, 1995. Cited on page 24.

[61] I. Poupyrev, M. Billinghurst, S. Weghorst, and T. Ichikawa, "The Go-Go interaction technique: Non-linear mapping for direct manipulation in VR," in *ACM Symposium on User Interface Software and Technology*, pp. 79–80, 1996. Cited on page 24.

[62] R. Pausch, T. Burnette, D. Brockway, and M. E. Weiblen, "Navigation and locomotion in virtual worlds via flight into hand-held miniatures," in *Proceedings of the 22nd annual conference*

on Computer graphics and interactive techniques, SIGGRAPH '95, (New York, NY, USA), pp. 399–400, ACM, 1995. Cited on page 25.

[63] D. A. Bowman, C. A. Wingrave, J. M. Campbell, V. Q. Ly, and C. J. Rhoton, "Novel Uses of Pinch Gloves for Virtual Environment Interaction Techniques," *Virtual Reality*, vol. 6, no. 3, pp. 122–129, 2002. Cited on page 25.

[64] K. Pfeil, S. L. Koh, and J. LaViola, "Exploring 3D gesture metaphors for interaction with unmanned aerial vehicles," in *IUI* (J. Kim, J. Nichols, and P. A. Szekely, eds.), pp. 257–266, ACM, 2013. Cited on page 25.

[65] B. Williamson, C. A. Wingrave, and J. J. L. Jr., "RealNav: Exploring natural user interfaces for locomotion in video games," in *3DUI*, pp. 3–10, IEEE, 2010. Cited on pages 25 and 28.

[66] M. Wrzesien, M. J. Rupérez, and M. A. Raya, "Input Devices in Mental Health Applications: Steering Performance in a Virtual Reality Paths with WiiMote," in *INTERACT (2)* (P. Campos, T. C. N. Graham, J. A. Jorge, N. J. Nunes, P. A. Palanque, and M. Winckler, eds.), vol. 6947 of *Lecture Notes in Computer Science*, pp. 65–72, Springer, 2011. Cited on pages 25 and 28.

[67] M. N. Kamel Boulos, B. J. Blanchard, C. Walker, J. Montero, A. Tripathy, and R. Gutierrez-Osuna, "Web GIS in practice X: a Microsoft Kinect natural user interface for Google Earth navigation.," *International Journal of Health Geographics*, vol. 10, no. 1, p. 45, 2011. Cited on page 25.

[68] J. Gallagher, "Everything You Need To Know About PlayStation Move," 2010. http://blog.eu.playstation.com/2010/09/07/

everything-you-need-to-know-about-playstation-move/ accessed: 18 Feb 2013. Cited on pages 25 and 79.

[69] Sony Computer Entertainment America LLC, "Move.me - Software Tool for PS3 System that uses PlayStation Move Technology," 2013. https://us.playstation.com/ps3/playstation-move/move-me/ accessed: 04/04/2013. Cited on page 25.

[70] Nintendo of America Inc., "|Nintendo - Corporate Information | Company History." http://www.nintendo.com/corp/history.jsp accessed: 18 Feb 2013. Cited on pages 27 and 79.

[71] Microsoft, "Kinect for Xbox 360: Science Fiction Comes to Your Living Room," 2010. http://www.microsoft.com/en-us/news/features/2010/nov10/11-03Kinect.aspx accessed: 18 Feb 2013. Cited on pages 27 and 79.

[72] Kinect for Windows Team, "Starting February 1, 2012: Use the Power of Kinect for Windows to Change the World," 2012. http://blogs.msdn.com/b/kinectforwindows/archive/2012/01/09/kinect-for-windows-commercial-program-announced.aspx accessed: 18 Feb 2013. Cited on page 27.

[73] Razer Inc, "Razer Hydra Gaming Controller - PC Motion Sensing Controllers - Razer United States," 2013. http://www.razerzone.com/gaming-controllers/razer-hydra accessed: 04/04/2013. Cited on page 27.

[74] Leap Motion, Inc, "Leap Motion," 2013. https://www.leapmotion.com/product accessed: 04/04/2013. Cited on page 27.

[75] C. Smith and H. Christensen, "Wiimote robot control using human motion models," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 5509–5515, Oct 2009. Cited on page 27.

[76] Y. Chow, "3D Spatial Interaction with the Wii Remote for Head-Mounted Display Virtual Reality," *Engineering and Technology*, pp. 377–383, 2009. Cited on page 28.

[77] K. O'hara, R. Harper, H. Mentis, A. Sellen, and A. Taylor, "On the naturalness of touchless: Putting the "interaction" back into NUI," *ACM Transactions on Computer-Human Interactions*, vol. 20, pp. 5:1–5:25, Apr. 2013. Cited on page 28.

[78] J. Accot and S. Zhai, "Beyond Fitts' law: models for trajectory-based HCI tasks," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '97, (New York, NY, USA), pp. 295–302, ACM, 1997. Cited on pages 30 and 95.

[79] R. Kopper, D. A. Bowman, M. G. Silva, and R. P. McMahan, "A human motor behavior model for distal pointing tasks," *International Journal of Human-Computer Studies*, vol. 68, no. 10, pp. 603–615, 2010. Cited on pages 30 and 95.

[80] G. Shoemaker, T. Tsukitani, Y. Kitamura, and K. S. Booth, "Two-Part Models Capture the Impact of Gain on Pointing Performance," *ACM Trans. Comput.-Hum. Interact.*, vol. 19, pp. 28:1–28:34, Dec. 2012. Cited on pages 30 and 95.

[81] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *Int. J. High Perform. Comput. Appl.*, vol. 15, pp. 200–222, Aug. 2001. Cited on page 31.

[82] W. Gentzsch, "Sun Grid Engine: towards creating a compute power grid," in *Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on*, pp. 35–36, 2001. Cited on page 31.

[83] U. Shankar and A. Schwierskott, *Oracle Grid Engine: User Guide*. Oracle, release 6.2 update 7 ed., February 2012. Cited on page 31.

[84] Grid Engine, "Open grid scheduler: The official open source grid engine." http://gridscheduler.sourceforge.net/ accessed: 24 Oct 2013. Cited on page 32.

[85] K. Buckley, "Science faculty HPC facility." http://ecs.victoria.ac.nz/EResearch/ScienceFacultyHPCFacility accessed: 24 Oct 2013. Cited on pages 32 and 65.

[86] Nvidia, *CUDA C Programming Guide*, pg-02829-001_v5.0 ed., October 2012. Cited on pages 32 and 63.

[87] K. Buckley, "ECS Tesla resource." http://ecs.victoria.ac.nz/EResearch/EcsTeslaResource accessed: 24 Oct 2013. Cited on pages 32 and 65.

[88] R. Klapaukh, "GraphLayout." https://github.com/klapaukh/GraphLayout accessed: 04 Dec 2013. Cited on pages 37 and 85.

[89] R Development Core Team, *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2010. Cited on pages 44, 65, and 88.

[90] M. Hollander and D. A. Wolfe, *Nonparametric Statistical Methods*. Whiley Series in Probability and Mathematical Statistics,

New York, USA: John Whiley & Sons, 1st ed., 1973. Cited on page 44.

[91] R. Klapaukh, "CudaLayout." https://github.com/klapaukh/CudaLayout accessed: 20 Dec 2013. Cited on page 63.

[92] R. Klapaukh, "GraphAnalyser." https://github.com/klapaukh/GraphAnalyser accessed: 20 Dec 2013. Cited on pages 65 and 102.

[93] Entertainment Software Association, "2013 essential facts about the computer and video game industry," 2013. Cited on page 79.

[94] Entertainment Software Association of Canada, "Essential facts 2012," 2012. Cited on page 79.

[95] A. Voida and S. Greenberg, "Wii all play: the console game as a computational meeting place," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, (New York, NY, USA), pp. 1559–1568, ACM, 2009. Cited on page 80.

[96] K. Poels, Y. de Kort, and W. Ijsselsteijn, ""It is always a lot of fun!": exploring dimensions of digital game experience using focus group methodology," in *Proceedings of the 2007 conference on Future Play*, Future Play '07, (New York, NY, USA), pp. 83–89, ACM, 2007. Cited on page 80.

[97] A. Kerr, "Women just want to have fun - a study of adult female players of digital games," in *Level Up Conference Proceedings: Proceedings of the 2003 Digital Games Research Association Conference* (C. Marinka and R. Joost, eds.), (Utrecht), pp. 270–285, University of Utrecht, November 2003. Cited on page 80.

[98] T. Bogers, "Movie Recommendation using Random Walks over the Contextual Graph," in *Second Workshop on Context-Aware Recommender Systems*, 2010. Cited on page 80.

[99] D. Cooper, "Sony reveals how the PlayStation 4 Eye works," 2013. http://www.engadget.com/2013/02/21/sony=playstation-4-eye-works/ accessed: 23 April 2013. Cited on page 82.

[100] R. Klapaukh, "JMoveMe." https://github.com/klapaukh/JMoveMe accessed: 20 Dec 2013. Cited on page 85.