University of Glasgow | School of Computing Science

Honours Individual Project Dissertation

Level 4 Project Report Template

Pieter van Tuijl

February 21, 2025

Abstract

Every abstract follows a similar pattern. Motivate; set aims; describe work; explain results.

“XYZ is bad. This project investigated ABC to determine if it was better. ABC used XXX and YYY to implement ZZZ. This is particularly interesting as XXX and YYY have never been used together. It was found that ABC was 20% better than XYZ, though it caused rabies in half of subjects.”

Acknowledgements

Education Use Consent

Contents

1 Introduction

1.1 Context

Algorithms drive much of our digital world. They are used in everything from search engines to social media to self-driving cars. Algorithms operate on data and both the data and the operations can be complex and difficult to visualise without good tools. This is particularly the case for spatial data.

Graph data structures are a common way to represent spatial data, where locations are modelled as nodes and connections between the locations as edges. Trees are a type of graph that connect all nodes in a single, connected network. In other words, for each node in the network, there exists exactly one path to all other nodes (Wikipedia contributors 2025).
Here, we will discuss the visualisation of two types of trees: minimum spanning trees and Steiner minimal trees. Additionally, in this project, we only consider the Euclidian variants of these trees. This means that the Euclidian distance between two nodes is used as the edge weight.

1.1.1 Minimum Spanning Tree

A minimum spanning tree (MST) is a tree that connects all nodes in a network with the least amount of total distance. There exist well-established algorithms for finding the MST of a graph, such as Kruskal's or Prim's algorithm. These algorithms run in polynomial time and work by building up the MST one edge at a time.

1.1.2 Steiner Minimal Tree

A Steiner minimal tree (SMT) is a variant of the MST. Apart from connecting all nodes in the network with minimal total distance, other nodes may be added to potentially further reduce the total distance. These additional nodes are called Steiner points. A SMT with zero Steiner points is equivalent to an MST. In contrast to the MST algorithm, finding the optimal SMT is a NP-hard problem, meaning, there exists no polynomial time algorithm for finding the optimal SMT (more on this later).

TODO: show example of MST vs SMT

1.2 Aims

Many tools exist for visualising minimum spanning trees. However, few tools exist for visualising Steiner minimal trees. This is related to the earlier-mentioned fact that the optimal SMT for a given set of nodes is computationally hard to find. This has historically imposed serious limits on the instance sizes that can be solved on an average computer, thus limiting the practicality and usefulness of SMT visualisation tools. Advances in computer hardware and efficient algorithm implementations, however, have made it possible to solve instances of several orders of magnitude larger than previously possible in reasonable time (Juhl et al. 2018).

The few existing visualisation tools lack flexibility and a user-friendly interface. (see background) We aim to fill this gap by developing a user-friendly interface that allows users to visualise SMTs alongside MSTs for graph instances of arbitrary sizes. These instances can be generated randomly or imported from a file.

The interface should be able to display the MST and SMT of a graph simultaneously, allowing for direct comparison of their structures and total length.
Comparison of length will be helpful for building intuition for the *Steiner Ratio*, which is defined as the least upper bound (supremum) of the ratio between the length of the MST and the SMT. Gilbert and Pollak (1968) conjectured that this ratio is

$$\frac{2}{\sqrt{3}} \approx 1.1547$$

In other words, the *Steiner Ratio* states that the MST is at most 15% longer than the SMT in the worst case.

Lastly, the interface should be able to dynamically update the solution when the user modifies the graph.

1.3 Dissertation Outline

TODO: write at the end of the project

2 Background

Let's start with the theoretical background. Then we will proceed to talk about existing tools and how they compare to our aims.

Start with explaining why the SMT is a NP hard problem. State what algorithms exist and why they are not polynomial time algorithms.

2.1 Existing visualisation tools

During the research phase, we found three related projects on Github whose features and limitations will be discussed in this section.

2.1.1 Steiner-Tree-Visualisation (STV)

STV is a Python-based GUI tool developed by Keydrain (2015). It provides a simple interface for visualising Euclidian MSTs and SMTs, displaying their lengths and allowing for direct comparison. However, the tool is limited by a few factors.
First, a brute-force approximation algorithm is used to find the SMT, and despite the $O(n^4 \log(n))$ complexity, the tool freezes for instances larger than 40 nodes. Additionally, the GUI is not very flexible. For example, you cannot overlay the MST and SMT simultaneously and the canvas does not support zooming or resizing. It is also not possible to import a graph from an external file or export the results. Lastly, the tool does not work out of the box and requires code patching to run in modern Python environments.

2.1.2 ESteiner-3D (E3D)

E3D is another Python-based tool developed by Abd (2024). It is a program that can be used to find the Euclidian SMT of a graph and supports 2D and 3D graphs. However, the tool is limited due to a lack of an interface, whether it a GUI or CLI. It also does not support the simultaneous visualisation of MSTs and SMTs and the comparison of their lengths.

2.1.3 Steiner-Tree (ST)

ST is a Javascript-based tool developed by Dawkey (2019). It is a web page that can be used to find the Rectilinear SMT of a graph. Although the tool has a nice interface and provides a flexible canvas, it does not support MSTs or SMTs in the Euclidian plane (L2-norm). Furthermore, it is not possible to import a graph from an external file or export the results.

2.2 Theoretical background

2.3 Algorithms

2.4 Problem definition

Given a graph $G = (V, E)$ and a set of Steiner points $S \subseteq V$, find the SMT T of G with respect to S.‘

3 Analysis/Requirements

What is the problem that you want to solve, and how did you arrive at it?

3.1 Guidance

Make it clear how you derived the constrained form of your problem via a clear and logical process.

The analysis chapter explains the process by which you arrive at a concrete design. In software engineering projects, this will include a statement of the requirement capture process and the derived requirements.

In research projects, it will involve developing a design drawing on the work established in the background, and stating how the space of possible projects was sensibly narrowed down to what you have done.

4 Design

How is this problem to be approached, without reference to specific implementation details?

4.1 Guidance

Design should cover the abstract design in such a way that someone else might be able to do what you did, but with a different language or library or tool. This might include overall system architecture diagrams, user interface designs (wireframes/personas/etc.), protocol specifications, algorithms, data set design choices, among others. Specific languages, technical choices, libraries and such like should not usually appear in the design. These are implementation details.

5 Implementation

What did you do to implement this idea, and what technical achievements did you make?

5.1 Guidance

You can't talk about everything. Cover the high level first, then cover important, relevant or impressive details.

5.2 General guidance for technical writing

These points apply to the whole dissertation, not just this chapter.

5.2.1 Figures

Always refer to figures included, like Figure 5.1, in the body of the text. Include full, explanatory captions and make sure the figures look good on the page. You may include multiple figures in one float, as in Figure 5.2, using `subcaption`, which is enabled in the template.

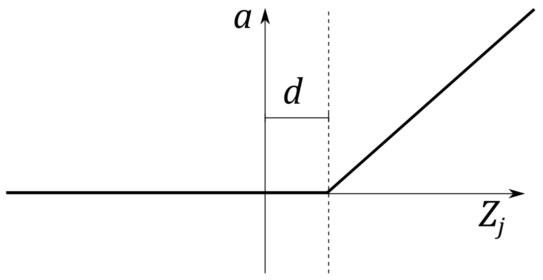
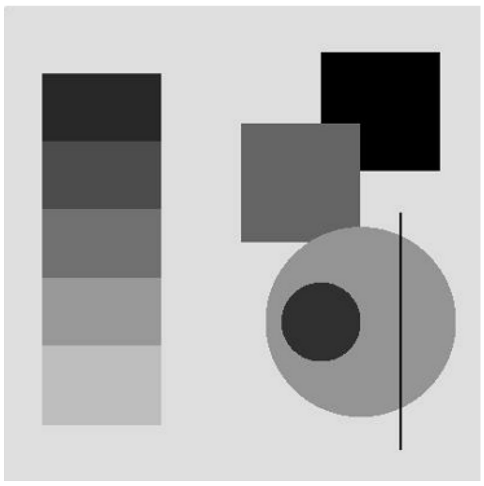
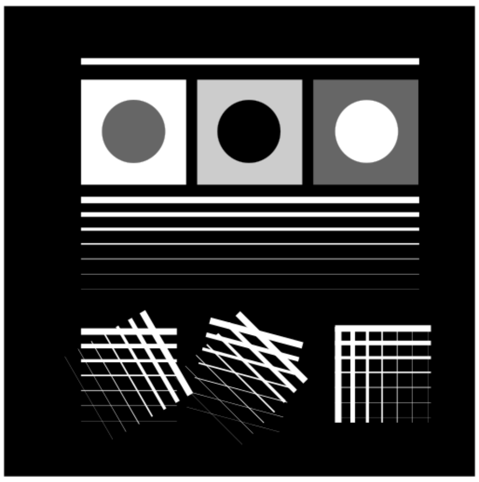


Figure 5.1: *In figure captions, explain what the reader is looking at: "A schematic of the rectifying linear unit, where* a *is the output amplitude,* d *is a configurable dead-zone, and* Z_j *is the input signal", as well as why the reader is looking at this: "It is notable that there is no activation* at all *below 0, which explains our initial results."* ***Use vector image formats (.pdf) where possible****. Size figures appropriately, and do not make them over-large or too small to read.*



(a) *Synthetic image, black on white.*



(b) *Synthetic image, white on black.*

Figure 5.2: *Synthetic test images for edge detection algorithms. (a) shows various gray levels that require an adaptive algorithm. (b) shows more challenging edge detection tests that have crossing lines. Fusing these into full segments typically requires algorithms like the Hough transform. This is an example of using subfigures, with* `subref`*s in the caption.*

5.2.2 Equations

Equations should be typeset correctly and precisely. Make sure you get parenthesis sizing correct, and punctuate equations correctly (the comma is important and goes *inside* the equation block). Explain any symbols used clearly if not defined earlier.

For example, we might define:

$$\hat{f}(\xi) = \frac{1}{2}\left[\int_{-\infty}^{\infty} f(x)e^{2\pi i x \xi}\right], \tag{5.1}$$

where $\hat{f}(\xi)$ is the Fourier transform of the time domain signal $f(x)$.

5.2.3 Algorithms

Algorithms can be set using `algorithm2e`, as in Algorithm 1.

```
Data: $f_X(x)$, a probability density function returing the density at $x$.
$\sigma$ a standard deviation specifying the spread of the proposal distribution.
$x_0$, an initial starting condition.
Result: $s = [x_1, x_2, \ldots, x_n]$, $n$ samples approximately drawn from a distribution with PDF $f_X(x)$.
begin
    $s \longleftarrow []$
    $p \longleftarrow f_X(x)$
    $i \longleftarrow 0$
    while $i < n$ do
        $x' \longleftarrow \mathcal{N}(x, \sigma^2)$
        $p' \longleftarrow f_X(x')$
        $a \longleftarrow \frac{p'}{p}$
        $r \longleftarrow U(0, 1)$
        if $r < a$ then
            $x \longleftarrow x'$
            $p \longleftarrow f_X(x)$
            $i \longleftarrow i + 1$
            append $x$ to $s$
        end
    end
end
```

Algorithm 1: The Metropolis-Hastings MCMC algorithm for drawing samples from arbitrary probability distributions, specialised for normal proposal distributions $q(x'|x) = \mathcal{N}(x, \sigma^2)$. The symmetry of the normal distribution means the acceptance rule takes the simplified form.

5.2.4 Tables

If you need to include tables, like Table 5.1, use a tool like https://www.tablesgenerator.com/ to generate the table as it is extremely tedious otherwise.

5.2.5 Code

Avoid putting large blocks of code in the report (more than a page in one block, for example). Use syntax highlighting if possible, as in Listing 5.1.

Table 5.1: *The standard table of operators in Python, along with their functional equivalents from the* `operator` *package. Note that table captions go above the table, not below. Do not add additional rules/lines to tables.*

Operation	**Syntax**	**Function**
Addition	`a + b`	`add(a, b)`
Concatenation	`seq1 + seq2`	`concat(seq1, seq2)`
Containment Test	`obj in seq`	`contains(seq, obj)`
Division	`a / b`	`div(a, b)`
Division	`a / b`	`truediv(a, b)`
Division	`a // b`	`floordiv(a, b)`
Bitwise And	`a & b`	`and_(a, b)`
Bitwise Exclusive Or	`a ^b`	`xor(a, b)`
Bitwise Inversion	`~a`	`invert(a)`
Bitwise Or	`a | b`	`or_(a, b)`
Exponentiation	`a ** b`	`pow(a, b)`
Identity	`a is b`	`is_(a, b)`
Identity	`a is not b`	`is_not(a, b)`
Indexed Assignment	`obj[k] = v`	`setitem(obj, k, v)`
Indexed Deletion	`del obj[k]`	`delitem(obj, k)`
Indexing	`obj[k]`	`getitem(obj, k)`
Left Shift	`a <<b`	`lshift(a, b)`
Modulo	`a % b`	`mod(a, b)`
Multiplication	`a * b`	`mul(a, b)`
Negation (Arithmetic)	`- a`	`neg(a)`
Negation (Logical)	`not a`	`not_(a)`
Positive	`+ a`	`pos(a)`
Right Shift	`a >>b`	`rshift(a, b)`
Sequence Repetition	`seq * i`	`repeat(seq, i)`
Slice Assignment	`seq[i:j] = values`	`setitem(seq, slice(i, j), values)`
Slice Deletion	`del seq[i:j]`	`delitem(seq, slice(i, j))`
Slicing	`seq[i:j]`	`getitem(seq, slice(i, j))`
String Formatting	`s % obj`	`mod(s, obj)`
Subtraction	`a - b`	`sub(a, b)`
Truth Test	`obj`	`truth(obj)`
Ordering	`a <b`	`lt(a, b)`
Ordering	`a <= b`	`le(a, b)`

```python
def create_callahan_table(rule="b3s23"):
   """Generate the lookup table for the cells."""
   s_table = np.zeros((16, 16, 16, 16), dtype=np.uint8)
   birth, survive = parse_rule(rule)

   # generate all 16 bit strings
   for iv in range(65536):
      bv = [(iv >> z) & 1 for z in range(16)]
      a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p = bv

      # compute next state of the inner 2x2
      nw = apply_rule(f, a, b, c, e, g, i, j, k)
      ne = apply_rule(g, b, c, d, f, h, j, k, l)
      sw = apply_rule(j, e, f, g, i, k, m, n, o)
      se = apply_rule(k, f, g, h, j, l, n, o, p)

      # compute the index of this 4x4
      nw_code = a | (b << 1) | (e << 2) | (f << 3)
      ne_code = c | (d << 1) | (g << 2) | (h << 3)
      sw_code = i | (j << 1) | (m << 2) | (n << 3)
      se_code = k | (l << 1) | (o << 2) | (p << 3)

      # compute the state for the 2x2
      next_code = nw | (ne << 1) | (sw << 2) | (se << 3)

      # get the 4x4 index, and write into the table
      s_table[nw_code, ne_code, sw_code, se_code] = next_code

   return s_table
```

Listing 5.1: *The algorithm for packing the 3×3 outer-totalistic binary CA successor rule into a $16 \times 16 \times 16 \times 16$ 4 bit lookup table, running an equivalent, notionally 16-state 2×2 CA.*

6 Evaluation

How good is your solution? How well did you solve the general problem, and what evidence do you have to support that?

6.1 Guidance

- Ask specific questions that address the general problem.
- Answer them with precise evidence (graphs, numbers, statistical analysis, qualitative analysis).
- Be fair and be scientific.
- The key thing is to show that you know how to evaluate your work, not that your work is the most amazing product ever.

6.2 Evidence

Make sure you present your evidence well. Use appropriate visualisations, reporting techniques and statistical analysis, as appropriate. The point is not to dump all the data you have but to present an argument well supported by evidence gathered.

If you use numerical evidence, specify reasonable numbers of significant digits; don't state "18.41141% of users were successful" if you only had 20 users. If you average *anything*, present both a measure of central tendency (e.g. mean, median) *and* a measure of spread (e.g. standard deviation, min/max, interquartile range).

You can use `siunitx` to define units, space numbers neatly, and set the precision for the whole LaTeX document.

For example, these numbers will appear with two decimal places: 3.14, 2.72, and this one will appear with reasonable spacing 1 000 000.00.

If you use statistical procedures, make sure you understand the process you are using, and that you check the required assumptions hold in your case.

If you visualise, follow the basic rules, as illustrated in Figure 6.1:

- Label everything correctly (axis, title, units).
- Caption thoroughly.
- Reference in text.
- **Include appropriate display of uncertainty (e.g. error bars, Box plot)**
- Minimize clutter.

See the file `guide_to_visualising.pdf` for further information and guidance.

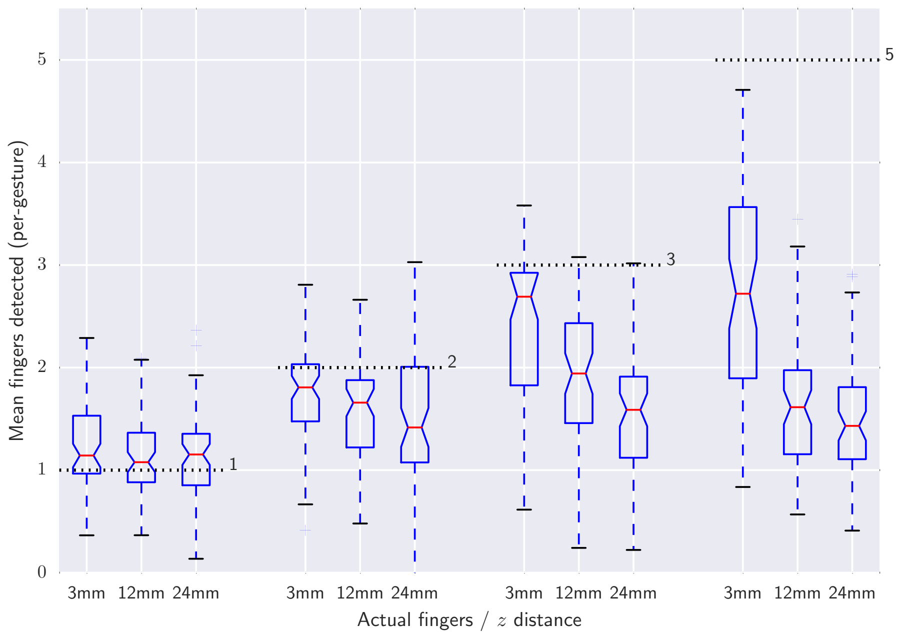


Figure 6.1: *Average number of fingers detected by the touch sensor at different heights above the surface, averaged over all gestures. Dashed lines indicate the true number of fingers present. The Box plots include bootstrapped uncertainty notches for the median. It is clear that the device is biased toward undercounting fingers, particularly at higher z distances.*

7 Conclusion

Summarise the whole project for a lazy reader who didn't read the rest (e.g. a prize-awarding committee). This chapter should be short in most dissertations; maybe one to three pages.

7.1 Guidance

- Summarise briefly and fairly.
- You should be addressing the general problem you introduced in the Introduction.
- Include summary of concrete results ("the new compiler ran 2x faster")
- Indicate what future work could be done, but remember: **you won't get credit for things you haven't done**.

7.2 Summary

Summarise what you did; answer the general questions you asked in the introduction. What did you achieve? Briefly describe what was built and summarise the evaluation results.

7.3 Reflection

Discuss what went well and what didn't and how you would do things differently if you did this project again.

7.4 Future work

Discuss what you would do if you could take this further – where would the interesting directions to go next be? (e.g. you got another year to work on it, or you started a company to work on this, or you pursued a PhD on this topic)

A Appendices

Use separate appendix chapters for groups of ancillary material that support your dissertation. Typical inclusions in the appendices are:

- Copies of ethics approvals (you must include these if you needed to get them)
- Copies of questionnaires etc. used to gather data from subjects. Don't include voluminous data logs; instead submit these electronically alongside your source code.
- Extensive tables or figures that are too bulky to fit in the main body of the report, particularly ones that are repetitive and summarised in the body.
- Outline of the source code (e.g. directory structure), or other architecture documentation like class diagrams.
- User manuals, and any guides to starting/running the software. Your equivalent of `readme.md` should be included.

Don't include your source code in the appendices. It will be submitted separately.

Bibliography

Abd, K. (2024), ‘Esteiner-3d’.
URL: *https://github.com/Kallel-Abd/ESteiner-3D/tree/main*

Dawkey (2019), ‘Steiner-tree’.
URL: *https://github.com/Dawkey/Steiner-Tree/*

Gilbert, E. N. and Pollak, H. O. (1968), ‘Steiner minimal trees’, *Siam Journal on Applied Mathematics* **16**, 1–29.
URL: *https://api.semanticscholar.org/CorpusID:123196263*

Juhl, D., Warme, D., Winter, P. and Zachariasen, M. (2018), ‘The geosteiner software package for computing steiner trees in the plane: an updated computational study’, *Mathematical Programming Computation* **10**(4), 487–532.

Keydrain (2015), ‘Steiner-tree-visualisation’.
URL: *https://github.com/Keydrain/Steiner-Tree-Visualisation*

Wikipedia contributors (2025), ‘Tree (graph theory) — Wikipedia, the free encyclopedia’.
URL: *https://en.wikipedia.org/w/index.php?title=Tree_(graph_theory)&oldid=1270850413*