Safety Critical 2D Pen Plotter

Arran Jones
945187

Project Dissertation submitted to Swansea University in Partial Fulfilment
for the Degree of Master of Science

Swansea University
Prifysgol Abertawe

Department of Computer Science
Swansea University

September 2020

Declaration

This work has not been previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed ..... (candidate)

Date .....

Statement 1

This dissertation is the result of my own independent investigations, except where otherwise stated. Other sources are acknowledged by giving explicit references. A bibliography is appended.

Signed ..... (candidate)

Date .....

Statement 2

I hereby give consent for my dissertation, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ..... (candidate)

Date .....

Abstract

This document is a report on the progress and findings of a dissertation to implement a 2D pen plotter using safety critical design and implementation techniques. This project is split into 4 major sections. The conversion of DXF files into GCODE, the conversion of rasterized images (such as a PNG file) into GCODE, a simulator for a 2d pen plotter robot and a robust GUI for controlling the entire system. The original plan was to create a robot out of lego mindstorms which would move a pen around a sheet of paper to draw the product, but unfortunately COVID-19 caused the country to go into lockdown and obtaining the lego mindstorms kit became problematic. It was then that a simulator would be necessary to complete the project.

This project will require rigorous testing, multiple fail-safes and emergency stop buttons for each section in order to produce a safe and reliable product. In order to ensure this is all complete on time, a strict time schedule will be required. This will be maintained by using a modified version of scrumban, where the idea of sprints will be implemented along side the time management systems of kanban. This will need to be modified as scrumban is usually used for a team of developers, whereas this will be a solo project.

Contents

List of Tables

List of Figures

Chapter 1

Sprint 1 - Background Research

Before starting the project a lot of background research is required, as there is no prior knowledge of GCODE, DXF files or safety critical systems. This chapter outlines researched topics required to gain enough understanding to begin the project and what is learnt during the research.

1.1 Safety Critical Systems

The first section which requires research is safety critical systems. This is the main topic of the dissertation and there is no prior knowledge of the topic going into the project. This means that a lot of prior research is required in order to ensure the system follows a safety critical level design, implementation and testing.

Safety is a not usually talked about in conventional computer science. It isn't related to reliability or security but is its own quality. Safety is the act of preventing of accidents or loss. These can come in the forms of injuries, deaths or damage to equipment. Alongside the accidents, the main concepts to consider with safety are the risks, hazards, failures, errors and faults. The risk is a combination of probability of an accident occurring and the severity of the accident should it occur. The equation $risk = p(a) * s(a)$ is usually used as a guideline for the risk of an accident, although quantifying severity can be difficult in some unforeseen cases. Hazards are the criteria that must be fulfilled for an accident to take place. When an accident occurs, the hazard is usually very minor, but minor hazards have the ability to cause accidents of major severity. A failure is when a component of the system fails randomly due to a fault. An error is when a component fails due to a predictable fault in the system. A fault is either an

error or a failure.

Safety is far more expensive to add to a system as an after thought than it is to incorporate into the design and implementation phase during the entire development of the system. There is an eight-step guide to safety design and implementation:

1. Identify Hazards
2. Determine Risks
3. Define Safety Measures
4. Create Safe Requirements
5. Create Safe Designs
6. Implement Following Safe Designs
7. Review Previous Safety Steps
8. Test Extensively

Stages one through 3 are known as the safety analysis. These are the steps which require most forethought, as all following steps will be ensuring the chance of an accident occurring from all hazards stated in these steps are minimised. As the project is now using a simulation, instead of a robot, the hazards have already been reduced. However, all forethought will be done with an actual robot in mind, so some hazards may seem unnecessary to the final product.

I will be addressing each of the eight stages in their respective sprint, before beginning any implementation.

1.2 GCODE

The second topic that requires research is GCODE. Prior to beginning the project, GCODE was an unfamiliar language as we had minimal knowledge of CAD. When looking through the many instructions the language consisted of, some instructions can be chosen as relevant to this project:

- G0 - Rapid Movement to specified co-ordinates. This will be useful for moving the pen when it isn't meant to be drawing. This command is used in the format: G0 X1 Y1

- G1 - Linear Movement to specified co-ordinates. This will be useful for moving the pen when it's meant to be drawing. This command is used in the format: G1 X1 Y1
- G2 - Linear movement to specified co-ordinates in a clockwise arc. The arc is centred around 2 other co-ordinates. This command is used in the format: G2 X1 Y1 I1 J1
- G3 - Linear movement to specified co-ordinates in a counter-clockwise arc. The arc is centred around 2 other co-ordinates. This command is used in the format: G3 X1 Y1 I1 J1
- M0 - Program Stop
- M1 - Optional Stop
- M2 - End of Program
- F - Feed Rate, or in the case of the pen plotter, Movement Speed.

There are many more GCODE instructions, however they seem unnecessary to the project at this point in time.

In order to ensure the GCODE created is always created in an industry standard format we will be using a python library called pygcode.

1.3 DXF

At a basic level, DXF files are separated into features called entities. These entities define many things, such as the shape of an object, the thickness, the start and end co-ordiantes, the angle at which it begins and ends, and much more.

As we are only creating a 2D pen plotter, we will be minimising the number of entities we use in order to maintain simplicity and minimise errors. The entities we will use for this project will be "LINE" which is a straight line from one co-ordinate to another, "ARC" which is a curved line, "POLYLINE" which is a combination of the 2 previously stated entities and "CIRCLE" which is a 360 degree arc defined by its centre point and radius.

In order to deal with these entities, we will be using a library called ezdxf. This will help simplify the code we produce and means that any changes in the DXF structure can be updated in the program by updating the library version.

1.4 Rasterized Images

In order to covert rasterized images into GCODE, the image will first need to be converted into grayscale in order to allow for a single value to govern what is considered a desired part of the image and what is not. This will be the black value of the image, a number between 0 and 1. The user will then be given the option of drawing the contours of the image, or the entire image of black value greater than the given threshold. In order to find the contours of the image, the marching squares algorithm will be used.

1.4.1 Contours - Marching Squares

Marching squares is an image analysis algorithm, used to determine the contours. This is done by splitting the image into small squares and finding the black value of the corner of each square, these values are then compared to the threshold provided by the user to check if the corner is within an object or not. These squares can then be categorized into 16 unique cases, defining how the contours travel through each square. Once the case of each square is determined, a form of interpolation is required to approximate the co-ordinate at which the contour crosses the boundaries of the squares. This is usually done, and will also in the case of this project, using linear interpolation. The main issue with this algorithm is the large amount of approximation and that 2 of the cases contain ambiguity as to the orientation of the contour. This would not cause any real-world problem should the program require the objects remain close, but may cause issues when drawing the images.

1.4.2 Entire Image

In order to draw the entire image, an alrorithm will be required to read the image row by row and convert the lines of pixels with a black value within the threshold of the user to line GCODE. This will probably be done with a self-defined algorithm.

Chapter 2

Sprint 2 - DXF to GCODE

After completing the background research, the first thing to be worked on was the conversion of a DXF file into basic GCODE instructions.

2.1 Safety Critical Systems

The first section which required research was safety critical systems. This was the main topic of the dissertation and there was no prior knowledge of the topic going into the project. This meant that a lot of prior research was required in order to ensure the system followed a safety critical level design, implementation and testing.

2.2 GCODE

The internet is big [1]. Knowing how to phrase a question to a search engine is therefore an invaluable skill. If the request is simple enough, even a poorly structured query will likely return usable results. For more difficult to find resources you can leverage the language of the search engine to gather relevant papers and resources for your research more efficiently.

`https://www.gwern.net/Search`

"Internet Search Tips" [2] provides an excellent review of methods and tips for scouring the internet for hard to find resources. You will also be less likely to get caught behind journal paywalls when working remotely without a tunnel as your queries can be made to look for raw pdfs that are often released by the authors directly.

2.3 Organizing your citations in BibTeX

BibTeX is a language for specifying resource citations. Every time you access and read an academic paper, take code from an online repository, or source the media such as images from existing works you should create a BibTeX entry in a file that you keep throughout your research. Software such as Mendeley [3] can help automate the process of building your BibTeX library of citations.

```
@INPROCEEDINGS{kaj86,
    author    = {Kajiya, James T.},
    title     = {The Rendering Equation},
    booktitle = {Proceedings of the 13th Annual Conference on Computer Graphics
        and Interactive Techniques},
    year      = {1986},
    series    = {SIGGRAPH '86},
    pages     = {143--150},
    address   = {New York, NY, USA},
    publisher = {ACM},
    isbn      = {0-89791-196-2},
    numpages  = {8},
    acmid     = {15902}
}
```

Listing 2.1: An example BibTeX entry for an academic paper published in conference proceedings [4].

The BibTeX code listing above (listing 2.1) shows an example of how to cite an academic paper, in this case one of the central papers in Computer Graphics research. The key **kaj86** is an arbitrary name chosen as a meaningful identifier for the resource. In the document text we can call on this resource as an inline citation using the LaTeX command `\cite{kaj86}` which produces [4] at the location it is called. As long as a citation has been used at least once somewhere within the document then a formatted full citation will be created in the bibliography at the end of the document with the same citation number that is shown inline.

It is considerably easier to be disciplined in methodically taking note of the resources you access and make use of as you access them, than it is to try and hunt them all down again at the time you need to write about them in your document. Invest time in being organized and consistent up front and it will be easier when you come to write up.

2.4 DXF

Usually you would not put the URL of the resource you are citing directly in the text like is done previously in section 2.2. The citation for the resource [2] is sufficient to reference it within the text given that full details of its location are then kept neatly within the bibliography at the end of the document.

In normal usage the purpose of a citation is not to direct the reader away from your thesis, but to justify and back up assertions you are making about the state of the domain. If a reader questions your assertions then they can follow the rabbit hole of papers which will likely also make and justify assertions with even earlier papers from the literature.

In the above case the intention is for the reader of this template to actually go to that resource and read what it has to say directly. The link is therefore shown clearly within the main text to indicate that the reader should visit it. This as opposed to wanting the reader to purely acknowledge that the facts which are within the resource legitimize the points made in this document, in which case a simple inline citation is the best way to back up your assertions. Section 3.3.7 specifically touches on the best practice for how to cite images which you are importing from existing work.

Chapter 3

Typesetting your thesis

This document is intended as both a LaTeX thesis template and as a tutorial on structuring and typesetting your thesis in the LaTeX programming language.

The following are some powerful online resources for learning about LaTeX:

- **Overleaf Documentation for LaTeX**

  Overleaf [5] is an online browser-based LaTeX IDE which stores your document in the cloud and provides live recompilation as you type. The documentation on Overleaf's website has a good knowledge base of examples for how to typeset things cleanly and simply in LaTeX code.

  See: `https://www.overleaf.com/learn`

- **TeX StackExchange, the StackOverflow site dedicated to TeX questions**

  TeX StackExchange [6] is sub-community of the StackOverflow network dedicated to questions about the TeX family of typesetting tools including LaTeX, BibTeX and others. A vast majority of the time it is unlikely that the question or issue you are facing is one that has not been encountered before, and this site more than likely to be able to point you in the correct direction.

  See: `https://tex.stackexchange.com`

3.1 Referencing items within this document

In section 2.3 we saw examples of how to typeset citations for resources we had stored in an external BibTeX file. However, often we would like to accurately refer to the location of a resource or region of text stored somewhere else within this document[1]. To do this we need to annotate our LaTeX code with `\label{key}` statements which will take on the numeric (or otherwise formatted) identifier for the current chapter, section, figure, table, equation, ect where they are directly defined. To insert an inline reference to the label you can use the `\ref{key}` command which works similarly to the `\cite{key}` used for external references. In the event we chose to reorder or add additional content to the document, which would change the section numbering, the document will still compile to a pdf with the correct references inserted for each `\ref{key}` command.

3.2 Equations

Typesetting equations is one of the things that LaTeX does best. It has packages for different fonts and symbols for many different mathematical notations. However, to person learning how to typeset in LaTeX for the first time it can be a daunting and unwieldy user experience. Almost all LaTeX packages have documentation available in pdf format online, and documentation for packages specifically relating to fonts and symbols usually have tables enumerating the names and codes for all of the fonts symbols, organized by intended usage.

3.2.1 Inline equations

Small equations like $x = 0$ can be written directly within the text by using LaTeX's maths mode shorthand controlled by dollar signs `$ math mode $`. As long as it is not becoming cumbersome to the reader, equations such as $\mathbb{P}(A \cap B) = \mathbb{P}(B \cap A)$ are quite neatly displayed in this fashion.

[1] Like at the beginning of the last sentence when we referred to section 2.3.

3.2.2 Block equations

For long equations it is best to provide a break in the main text of the document and format the equation using a `\begin{equation}...\end{equation}` environment.

$$|a| = \left|\left[\begin{array}{c} a_0 \\ a_1 \\ \vdots \\ a_n \end{array}\right]\right| = \sqrt{a_0^2 + a_1^2 + \ldots + a_n^2} \tag{3.1}$$

Equation 3.1 demonstrates formatting a larger equation and uses an `\begin{array}...\end{array}` environment to structure a column vector of sub-equations. Block equations should be located at a relevant point directly as they are being referred to in the text. When referred to from other locations in the document you should use the `\ref{key}` command to insert the correct equation number.

3.2.2.1 Aligning multi-line block equations

When equations become even larger they may need cross over multiple new lines. When this happens it is desirable to align relevant parts of the equation on each line to one another for aesthetic reasons and to help imply structure to the reader.

$$\begin{aligned} \mathscr{L}_o(x,\omega_o,\lambda,t) &= \mathscr{L}_e(x,\omega_o,\lambda,t) \\ &\quad + \int_\Omega f(x,\omega_i,\omega_o,\lambda,t)\,\mathscr{L}_i(x,\omega_i,\lambda,t)\,(\omega_i \bullet n)\,d\omega_i \\ &\quad \text{where} \quad \mathscr{L}_i(x,\omega_i,\lambda,t) = \mathscr{L}_o\left(x',-\omega_i,\lambda,t\right) \end{aligned} \tag{3.2}$$

Equation 3.2, known as Kajiya's Rendering Equation [4] demonstrates the use of the `\begin{split}...\end{split}` environment which uses a single un-escaped & symbol placed on each line of the equations LaTeX code to indicate where each line should be co-aligned. In this example the &'s were placed on the =, +, and w (in where) characters.

3.2.3 A masochistic approach to learning to typeset mathematics in LaTeX

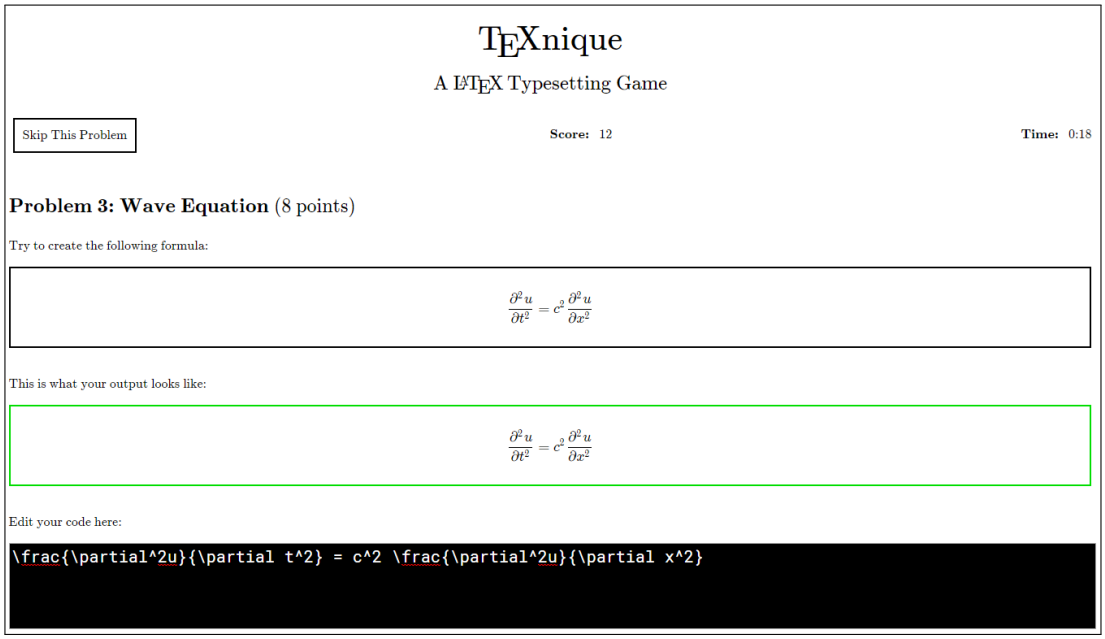


Figure 3.1: TeXnique, a game about typesetting equations [7]. (Top) The game presents you with a rendered equation, (Bottom) the task is to enter LaTeX code that produces the same rendered equation. The green border on the lower rendering indicates it is a valid solution.

TeXnique [7] is web-browser based game for practising how to typeset equations in LaTeX. The game will present you with a rendered equation and your task is to type LaTeX code into the box below it such that your code produces the same (or closely matching / pixel equivalent) rendered equation. Figure 3.1 shows the game during play, the bottom rendered equation is bordered in green to indicate it is a valid match with the target.

`https://texnique.xyz`

This is one of the more painful parts of typesetting a document, so it really takes a special kind of sadism to come up with such a game. Least to say, graduate students and researchers can be an odd bunch, and when we found this it was surprisingly addictive to compete over.

3.3 Figures

In this template figures are numbered starting with the current chapter number followed by a figure number that resets to 1 each new chapter. As you can see below, the first figure is

labelled Figure 3.2 because we are in Chapter 3.

Figures in LaTeX are defined using a `\begin{figure}...\end{figure}` environment and often immediately begin rendering in centre aligned mode by calling `\centering`. Listing 3.1 below shows the LaTeX code used to typeset figure 3.2. Figures 3.3 and 3.4 are defined similarly and make additional use of the `\subfloat` command to position multiple images within a single figure environment, each with their own automatically incremented labels and individual captions.

```
\begin{figure}[H]
  % [H] means put the figure HERE, directly when you input this code.
  \centering

  % We set the width of the figure based on the width of one line
  % of text on the page. The value can be tuned to any value in
  % [0.0, 1.0] to scale the image while maintaining its aspect ratio.
  \includegraphics[width=1.0\linewidth]{./graphics/dragon.png}

  % Caption is defined with a short and long version. The short
  % version is shown in the List of Figures section, and the long
  % version is used directly with the figure.
  \caption[Short caption.]{Long caption and citation \cite{whittle15_dragons}.}

  % For figures, \label should be defined after the caption to ensure
  % proper figure numbering.
  \label{fig:dragon}
\end{figure}
```

Listing 3.1: An example LaTeX excerpt demonstrating how to typeset figure 3.2 with a simple caption.

3.3.1 Consistent presentation throughout the document

Figures work best in a document when you use a consistent style for formatting and captioning them and make sure that figures always actively support the content of the main text.

3.3.2 Justified use of space in the document

All figures must be referred to directly in the main text of the document and discussed with meaningful and in depth critical analysis. If you don't need to use the figure to leverage and support your discussion then it is just taking up space and padding out the document. For example, you can use a command like `\ref{fig:dragon}` to automatically get the figure number for Figure 3.2.

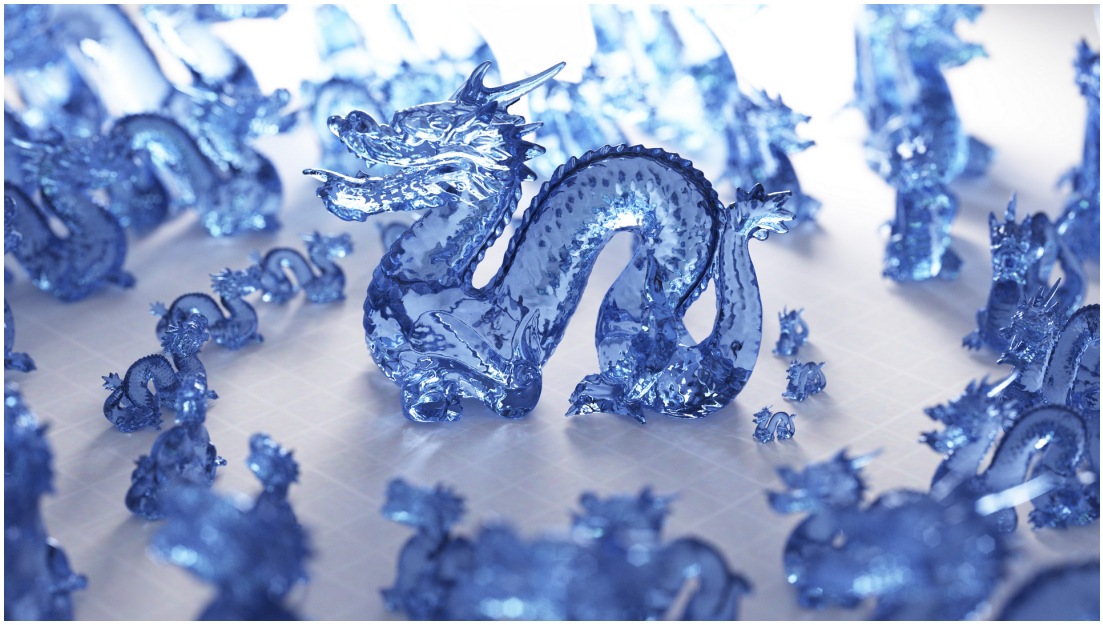


Figure 3.2: A good caption should be sufficient enough to put the figure in context even if the reader has randomly flicked to the current page and looked only at the figure in isolation. All figures should also be referred to directly within the main text of your document. You can use the LaTeX **ref{key}** command to insert the correct figure number when you refer to it in the main text. By the very logic of this caption, this is a very poor caption because we still don't know why on earth is there an picture of glass dragons here. Image of glass dragons rendered using Path Tracing [8].

3.3.3 Placement that supports and enhances the flow of the document

All figures shown in your document should be displayed in relevant locations, ideally just after that have been alluded to in the main text. Although there are many times where it is best to force a figure to the top or bottom of a nearby page.

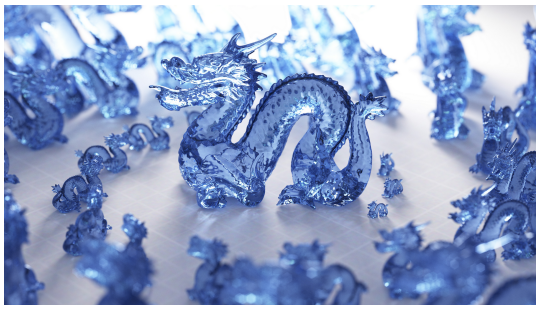
3.3.4 Avoid directly importing other peoples images

You should avoid using other peoples figures whenever possible, and instead create your own figures for visualizing the specific methods and data you are working with in a way directly relevant to your project.

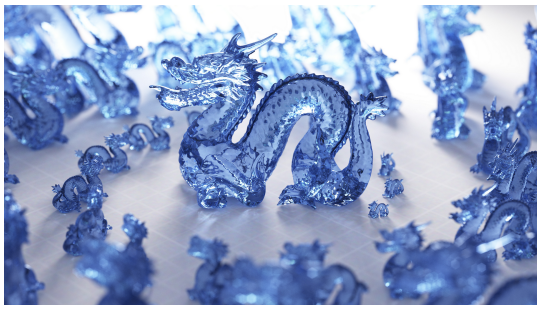
3.3.5 Format sub-figures in LaTeX, not in the image itself

Construct sub-figures from multiple image files in LaTeX not in the image file itself. This allows you to tweak the positioning and layout without having to modify the images. It also

allows for automatic formatting and numbering of captions and sub-captions. Figures 3.3 and 3.4 show examples of side-by-side and quad layouts respectively.

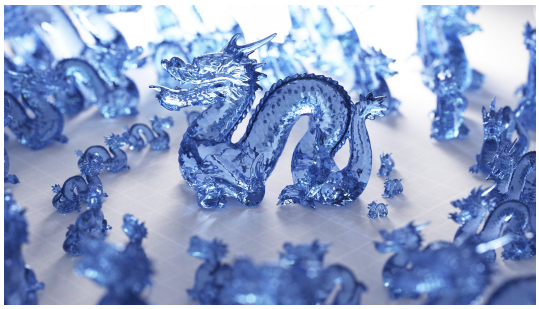


A. Left image sub-caption.

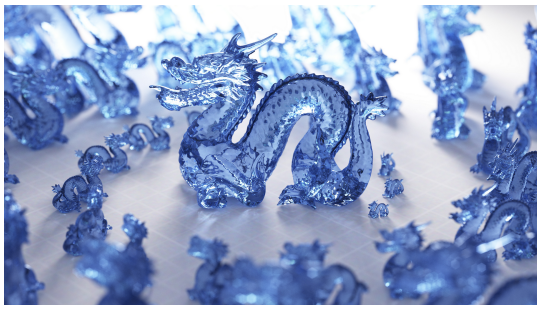


B. Right image sub-caption.

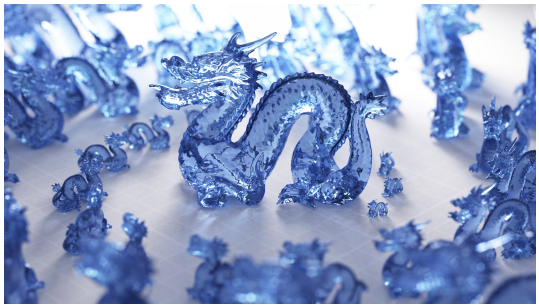
Figure 3.3: Construct sub-figures from multiple image files in LaTeX not in the image file itself. This allows you to tweak the positioning and layout without having to modify the images. It also allows for automatic formatting and numbering of captions and sub-captions. Image of glass dragons rendered using Path Tracing [8].



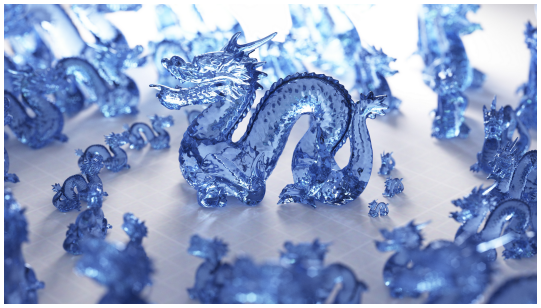
A. Top-Left image sub-caption.



B. Top-Right image sub-caption.



C. Bottom-Left image sub-caption.



D. Bottom-Right image sub-caption.

Figure 3.4: A demonstration of a 2x2 sub-figure layout. Between A-B and C-D we use tilde symbols and between B-C we use a new line. Image of glass dragons rendered using Path Tracing [8].

3.3.6 Robust captions that can stand in isolation

Figures need to be captioned such that they can be viewed in isolation and still be meaningful to the viewer. There will likely be some duplication of information that is written in the main text, but this is intended.

3.3.7 Proper attribution and citation of images

If an image does not belong to you it **must** be cited directly in the figure caption. **It is not correct to put a URL in the figure caption directly.** A URL in isolation is not an accurate or reliable way of directing a future reader to the exact content you are referencing. Instead make a new entry in your `citations.bib` file and then reference that citation in the caption using the `\cite{key}` command. Figures 3.2, 3.3, and 3.4 each include a statement in the caption stating "Image of glass dragons rendered using Path Tracing [8].". When adding the BibTeX entry, try to find the proper information about the original author and source document to strengthen the citation in case the URL changes.

3.4 Code Listings

Code listings should be formatted in the same style as figures and inline equations. It is important to use a monospace font so that characters line up vertically. Syntax highlighting is also extremely important for effectively displaying complicated code segments. To format inline code listings you can use the `\lstinline|the_code|` command[2].

```
#include <stdio.h>

int main(int argc, char *argv[]) {
  printf("Hello world.\n");
  return 0;
}
```

Listing 3.2: An implementation of an important algorithm from our work.

In LaTeX the "Listings" package can be used to properly format code and provide basic syntax highlighting, line numbering, and captioning of embedded code excerpts. Listing 3.3 shows examples of how to properly format code using the listings package.

[2] So meta.

```
% The lstinline command can be used to insert monospace formatted code directly
% inline within the documents main text. You can optionally specify a programming
% language to enable syntax highlighting.
\lstinline|the_code|
\lstinline[language={the_language}]|the_code|

% The lstinputlisting command is used to insert an external file containing
% code into the document formatted in the same manner as a figure or table.
% All stand alone listings should have a label and caption. You can optionally
% specify a programming language to enable syntax highlighting.
\lstinputlisting[label={lst:my_label_name}, caption={The caption.}]{the_file}
\lstinputlisting[language={the_language}, label={lst:the_label}, caption={The
    caption.}]{the_file}

% An example showing how Listing 3.1 is formatted in LaTeX code.
% The C code is stored in its own file as C code, allowing it to be modified
% and prepared separately using a dedicated code IDE to ensure correctness and
% proper formatting.
\lstinputlisting[language={c}, label={lst:c_hello_world}, caption={An
    implementation of an important algorithm from our work.}]{./listings/
    hello_world.c}
```

Listing 3.3: Examples of methods for typesetting code listings within a LaTeX document.

3.5 Tables

Tables are also quite predictably captioned and formatted the same way. It is important to decide on a style for how you will organize your data and apply that style consistently for all of your tables. Table 3.1 shows one possible way of styling your data but is by no means the only way of doing so neatly. Consistency is the key.

Table 3.1: An example of a table formatted with caption.

Some	Relevant	Fields	From	Your	Data
0	0	0	0	0	0
1	1	1	1	1	1
2	2	2	2	2	2

Chapter 4

Conclusions and Future Work

In this document we have demonstrated the use of a LaTeX thesis template which can produce a professional looking academic document.

4.1 Contributions

The main contributions of this work can be summarized as follows:

- **A LaTeX thesis template**

  Modify this document by adding additional top level content chapters. These descriptions should take a more retrospective tone as you include summary of performance or viability.

- **A typesetting guide of useful primitive elements**

  Use the building blocks within this template to typeset each part of your document. Aim to use simple and reusable elements to keep your document neat and consistently styled throughout.

- **A review of how to find and cite external resources**

  We review techniques and resources for finding and properly citing resources from the prior academic literature and from online resources.

4.2 Future Work

Future editions of this template may include additional references to Futurama.

Bibliography

[1] Internet Live Stats. (2020). [Online]. Available: https://www.internetlivestats.com

[2] G. Branwen. (2020) Internet search tips. [Online]. Available: https://www.gwern.net/Search

[3] RELX Group. (2019) Mendeley. [Online]. Available: https://www.mendeley.com

[4] J. T. Kajiya, "The rendering equation," in *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '86. New York, NY, USA: ACM, 1986, pp. 143–150.

[5] Overleaf. (2020) Overleaf documentation. [Online]. Available: https://www.overleaf.com/learn

[6] Stack Overflow. (2008) Tex stackexchange. [Online]. Available: https://tex.stackexchange.com

[7] A. Ravikumar. (2019) Texnique. [Online]. Available: https://texnique.xyz

[8] J. Whittle. (2015) Path traced glass dragons.

Appendix A

Implementation of a Relevant Algorithm

```
#include <stdio.h>

int main(int argc, char *argv[]) {
  printf("Hello world.\n");
  return 0;
}
```

Listing A.1: An implementation of an important algorithm from our work.

Appendix B

Supplementary Data

The results of large ablative studies can often take up a lot of space, even with neat visualization and formatting. Consider putting full results in an appendix chapter and showing excerpts of interesting results in your chapters with detailed analysis. You can use labels and references to refer the reader here for the full data.