Informatics

Automatisierte Typogramm-Erstellung aus 3D-Modellen

Entwicklung einer Applikation zur automatischen erstellung von Typogrammen

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Andreas Josef Rippl
Matrikelnummer 1183435

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Dr. Renata Georgia Raidou

Wien, 4. September 2023

Andreas Josef Rippl

Renata Georgia Raidou

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Automated Typogram Generation from 3D Models: A Computer-Aided Approach

Developing a Tool for Streamlined Typogram Creation

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Andreas Josef Rippl
Registration Number 1183435

to the Faculty of Informatics

at the TU Wien

Advisor: Dr. Renata Georgia Raidou

Vienna, 4th September, 2023

Andreas Josef Rippl	Renata Georgia Raidou

Erklärung zur Verfassung der Arbeit

Andreas Josef Rippl

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 4. September 2023

Andreas Josef Rippl

Kurzfassung

Diese Arbeit untersucht das Konzept der Typogramme, grafische Designs, die Objekte und Ideen mit Typografie darstellen, wobei der Schwerpunkt auf dem Prozess ihrer Erstellung und dem Potenzial zur Automatisierung dieses Prozesses liegt. Obwohl sie traditionell aufgrund ihrer Komplexität und Arbeitsintensität auf den Bereich der Kunst und des Designs beschränkt waren, könnte die Anwendung von Typogrammen in verschiedenen Bereichen wie Bildung und Wissenschaft revolutionär sein. Allerdings stellen das Fehlen klarer Richtlinien darüber, was ein Typogramm ausmacht, und die mit ihrer Erstellung verbundenen Komplexitäten erhebliche Herausforderungen dar.

Das Ziel dieser Studie ist es, einen computergestützten Workflow zu entwickeln, der die Erstellung von Typogrammen für beliebige 3D-Modelle vereinfacht. Dieser Workflow umfasst mehrere Schritte, von der Modellvorbereitung und der Auswahl des Sichtfensters bis hin zur Festlegung verschiedener Optionen und abschließenden Anpassungen. Die Studie betont die Notwendigkeit eines automatisierten Systems, das in der Lage ist, ein erstes Ergebnis zu generieren, das dann manuell angepasst werden kann. Das vorgeschlagene Tool basiert auf Webtechnologie und verwandelt den komplizierten Prozess der Typogrammerstellung in eine intuitive und explorative Aufgabe.

Die Arbeit ist so strukturiert, dass sie zuerst frühere Arbeiten und Inspirationen überprüft, gefolgt von einer eingehenden Diskussion über die angewandte Methodik und den detaillierten Implementierungsprozess. Sie schließt mit einer Präsentation und Analyse der Ergebnisse sowie Überlegungen für die Zukunft ab. Diese Arbeit dient dazu, die Erstellung von Typogrammen zu vereinfachen und sie für ein breiteres Publikum in verschiedenen Bereichen zugänglich und anpassbar zu machen.

Abstract

This thesis explores the concept of typograms, graphic designs that illustrate objects and ideas using typography, with a focus on the process of their creation and the potential to automate this process. Although traditionally limited to the arts and design sphere due to their complexity and labor-intensity, the application of typograms in diverse fields like education and science could be transformative. However, the lack of definitive guidelines about what constitutes a typogram and the complexities associated with their creation present significant challenges.

The purpose of this study is to devise a computer-aided workflow to streamline the creation of typograms for arbitrary 3D models. This workflow comprises several steps, from model preparation and viewport selection to setting various options and final adjustments. The study emphasizes the need for an automated system, capable of generating an initial result which can then be manually adjusted. The proposed tool is web-based and simplifies the intricate process of typogram creation into an intuitive and explorative task.

The thesis is structured to first review prior work and inspirations, followed by an in-depth discussion on the applied methodology and detailed implementation process. It concludes with a presentation and analysis of the results, along with future considerations. This work serves to democratize typogram creation, making it accessible and adaptable for a wider audience across various domains.

Contents

Kurzfassung **vii**

Abstract **ix**

Contents **xi**

1 Introduction **1**

2 Related Work **3**
- 2.1 Art Historic Context 3
- 2.2 Usages and Experiments 5
- 2.3 Labeling 6

3 Typogram Generator **9**
- 3.1 Typogram Creator Pipeline 9
- 3.2 User Interface 14

4 Implementation **19**
- 4.1 Settings and Helpers Panel 19
- 4.2 SVG Generation 20

5 Results **21**
- 5.1 Performance 22
- 5.2 Limitations and Areas for Improvement 24
- 5.3 Contribution to the Field 24

6 Conclusion **27**
- 6.1 Future Work 27

List of Figures **29**

Bibliography **31**

CHAPTER 1

Introduction

Typograms are illustrations that represent objects and ideas with the use of typography. When representing a shape or object, the object itself gets replaced with a word that is configured typographically within a designated spatial region so that it replicates the shape of the object it should represent.

Typograms have found their unique space within the design landscape due to their ability to represent complex structures in a visually compelling manner. However, the process of creating these typograms is often labor-intensive and requires a high level of skill in typography and design, making them inaccessible for wider application across diverse domains, such as education and science. These are fields where they could be employed as an effective labeling technique to build quick associations between object or concepts and their inherent title

The term ’typogram’ predominantly finds its use in the arts and design sphere, to represent a graphically manipulated version of a word with adjusted typography to better depict its meaning. This idea of a typogram was expanded by different artists (e.g. Xu Bing [Xu02] or Aaron Kuehn [Kue11]) into more than just one word, representing whole systems with this type of visualization.

The boundaries defining what is and is not considered a typogram are somewhat fluid, and no definitive guidelines have been explicitly stated in existing literature. This ambiguity allows the act of labeling three-dimensional or two-dimensional objects through internal labeling to fall within the realm of typograms. Given that labeling is a complex and extensive area of research, this thesis will concentrate on the aspect of internal labeling.

Historically internal labels are omnipresent in visualization and illustration of objects in for example in Medical Visualization (Grays anatomy atlas [GS16]). These illustrations were all hand crafted, either drawn or computer generated rendered and then again edited with photo editing software e.g. Adobe Photoshop or illustration tools e.g. Figma, Adobe Illustrator.

The aim of this thesis is to develop a computer-aided workflow that makes it quicker and more straightforward to create an appropriate typogram for any arbitrary three-dimensional model with labeled parts. The result of this tool is a customizable vector graphic that can be further adjusted manually with an SVG (Scalable Vector Graphics) illustration tool if necessary.

The workflow for creating typograms through the proposed computer-aided system can be broken down into several steps. The process begins with preparing the models. If multiple objects are involved, they can be combined into a single GLB/GLTF (GL Binary / GL Transmission Format) file using Blender or other comparable tools. Each object within this consolidated file should be separately named with the label that should be used for generating athe typogram.

Once the model is ready, the next step is to choose the viewport. Previously, due to the time-consuming nature of typogram creation, a meticulous process of deciding on a viewport was necessary, and changing it after the fact was not an option. However, as our proposed workflow is much quicker, it encourages exploration with different viewports.

Following the viewport selection, the third step involves setting various options. The variety of three-dimensional models and the potential types of typogram adjacent imagery one might want to create necessitates this. Options that can be configured include borders, fill, text-stretch, and text-size-variation, among others, depending on the desired output.

With these prerequisites completed, the system is ready to generate the initial result. However, given the range of potential model topologies, errors cannot be entirely avoided in this automated process. This is why a final step is necessary: finetuning the result. After the automatic process generates the typogram, it is essential to have the flexibility for manual adjustments to fine-tune the produced result. This way, the end product fully aligns with the expectations and needs of the user.

This tool, readily accessible in any modern browser (`typograms.rippl.at`), transforms the complex, time-consuming task of typogram creation into a more intuitive and explorative process. It simplifies what once required hours of manual adjustments, making the world of graphic design more approachable. Furthermore, this tool democratizes typogram creation, removing the need for in-depth understanding of design programs. It offers flexibility and ease of use, turning the tedious process into a platform for creative exploration.

The structure of this thesis is as follows: Chapter 2 offers a brief review of prior work and sources of inspiration in this field. Chapter 3 delves into the methodology employed in this study. The detailed implementation process is laid out in chapter 4. In chapter 5, the results obtained are presented, along with a discussion on the limitations of the current workflow. Finally, chapter 6 wraps up the thesis with a concise conclusion and considerations for potential future developments.

CHAPTER 2

Related Work

This chapter will explore two principal categories. First, it will explore the origin of typograms, providing an examination of their definition, as well as an exploration of their artistic and historical context. Secondly, the chapter will undertake a comparative study of work conducted in the field of labeling or annotations, with an emphasis on internal labeling due to its strong resemblance to the visual aesthetic characteristics of a contemporary typogram.

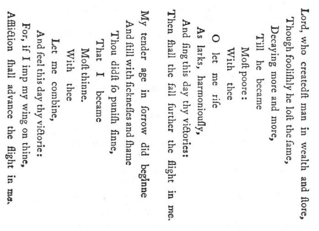
2.1 Art Historic Context

The use of typograms in art, while not being a new concept, has been inadequately explored, and the term itself remains under-defined and non-standardized. Several art forms rooted in typography have been known to demonstrate elements of what can be considered typograms today. One of the primary distinctions to note when discussing these various forms lies in the way they connect text, form and meaning.

- **Micrography (microcalligraphy)**, an art form originated in the 9th century within Jewish communities, uses minuscule hebrew letters to construct intricate patterns and images [Hal16]. In this discipline, the text flows along a path that forms the contours of the shapes. In essence, the visual form is dictated by the flow of the text (Figure 2.1a).
- **Carmina Figurata** or pattern poems, takes a somewhat contrasting approach to the use of text within the artwork. Here, the form is created not by the contour of the text line but by arranging written lines of different lengths one under the other within the overall contour of the artwork [Qui84]. In other words, the contours are not text-derived but are rather filled in with text of varying lengths to form the desired visual (Figure 2.1b).

- **Calligram** was popularized by Guillaume Apollinaire in France around the early 20th Centurywith his work "Calligrammes" [Apo66] (Figure 2.1c) created from 1913 until 1916 and José Juan Tablada with the work "Li-Po y otros Poemas" [TM17] around 1920. The calligram itself forms a recognizable shape that can be enhanced in meaning through the words or letters. It is defined as any design where letters are rearranged so that they form a decorative pattern [MW23].

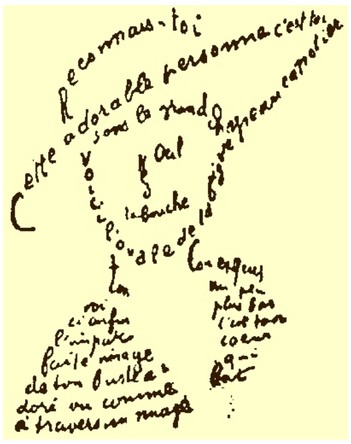
(a) The Sana'a Pentateuch, Carpet pages with verses from the Psalms in micrography, Yemen, Sana'a, 1469 CE, Grammatical Introduction (Makhberet ha-Tigan) (Or 2348, ff. 38v-39r)



Lord, who createdſt man in wealth and ſtore,
Though fooliſhly he loſt the ſame,
Decaying more and more,
Till he became
Moſt poore:
With thee
O let me riſe
As larks, harmoniouſly,
And ſing this day thy victories:
Then ſhall the fall further the flight in me.

My tender age in ſorrow did beginne
And ſtill with ſickneſſes and ſhame
Thou didſt ſo puniſh ſinne,
That I became
Moſt thinne.
With thee
Let me combine,
And feel this day thy victorie:
For, if I imp my wing on thine,
Affliction ſhall advance the flight in me.

(b) Pattern poem / Carmina Figurata "Easter Wings" by George Herbert. [Her38]



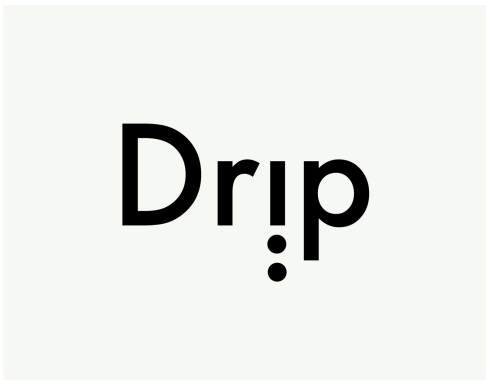
(c) A Page from Guillaume Apollinaire's "Calligrammes" [Apo66]

Figure 2.1: Origins of typograms

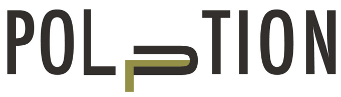
In a typogram the text itself and what it describes are the main focus, usually taking precedence over any shape or image it might form. Typograms in their emergence were used as visual puns more than anything, playing with a word and its meaning with clever typography to convey more meaning than before as shown in Figure 2.2.

Typically, the most common typograms are comprised of single-word instances. These primarily consist of noun that is attached to an adjective that expresses a state such as "hang" ("hanging") or "drip" ("dripping"), where each can be transformed into a typographical depiction that bears resemblance to its intended signification as shown in Figure 2.2a. Alternatively, they may involve a theme or entity such as 'pollution' or 'moon', which can be emphasized or enriched through typographical modifications or the incorporation of supplementary details.

The development of typograms has expanded from being based on single words or phrases to entire systems, as demonstrated in the works of Xu Bing (Figure 2.3b) and Aaron Kuehn (Figure 2.3a). These works are prominent examples of this progression. Even as they have grown in complexity, they still adhere to the same principles as the single-word typogram, just on a broader scale. They make use of methods that were previously used in calligrams to produce shapes that are similar to real objects, and the typography also discusses the objects they are meant to represent.

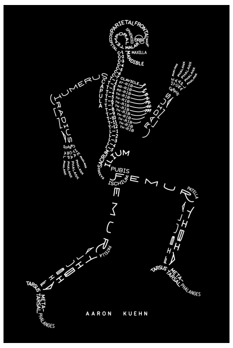


(a) Drip by Kekeli H. Sackey [Sac18]

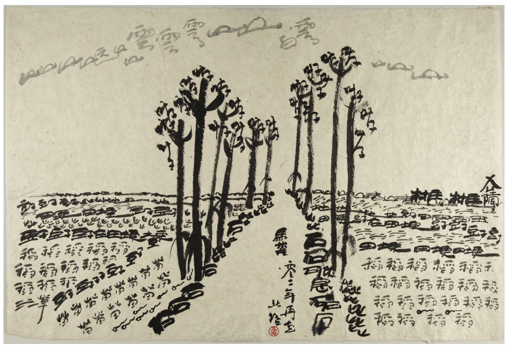


(b) Pollution by Brian Walton

Figure 2.2: Typical single-word typograms making a pun with typographical elements



(a) **Skeleton** by Aaron Kuehn [Kue11]



(b) **Landscript** by Xu Bing [Xu02]

Figure 2.3: Complex typogram systems

2.2 Usages and Experiments

Typogram creations are used in architectural education to help students visualize and explore different things [FM15]. Lipitz et al. developed a "Typographical Imaging System" [Lip21] capable of scrutinizing images via an algorithmic process, yielding a segmented rendition of the original image. These individual segments are further processed through object and face detection systems to gather a comprehensive set of descriptors for each distinct region, all of which are stored in a text file.

This elaborate procedure culminates in the generation of word clouds, crafted to fit perfectly into the separated parts of the image. In the final stages of the process, the original colors from the underlying image are applied to the text, creating a unique typographical interpretation.

When viewed from a distance, the completed works bear a resemblance to pixelated versions of the original images, a result of the merging text and distinctive color application. Upon closer inspection, the text content becomes more apparent, with the most prominent

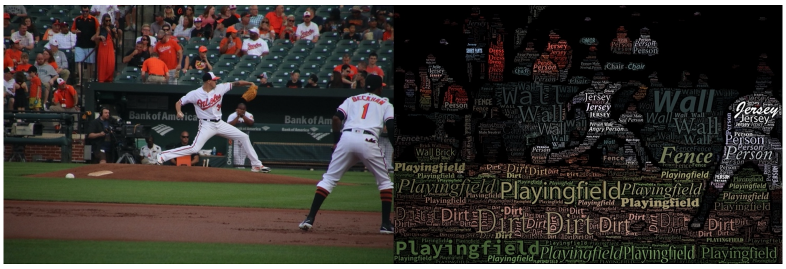


Figure 2.4: Image from Max Lipitz processed through the TIS [Lip21]

words in the word cloud (such as the name of a person) visible first, followed by less dominant words, such as adjectives. These descriptors potentially provide insights about the person or object depicted, depending on the information the system was able to generate using its suite of neural networks.

The primary method involves the extensive use of diverse text in multiple fonts. This technique achieves a representation of the original image through the sheer magnitude and amalgamation of information. Fonts are not arranged according to a predetermined layout or pattern. Instead, they completely fill the designated area. Every small part is filled with a font, and therefore, a color that corresponds to a particular section of the image. This thorough dispersion of text and color effectively encapsulates the entirety of the original visual.

2.3 Labeling

Annotations or labels of three-dimensional or two-dimensional objects, play a vital role in a wide variety of fileds, particularly in the educational and medical domain, but also architecture, design and data visualization in general. Oeltze et al. reviewed a number of different popular techniques for labeling and proposed a classification system based on the labeling technique employed. [OJP14] They listed the following types of labeling as the most significant:

- **Internal labels** are superimposed onto the object in question and horizontal if possible, if not enough space along the centerline. (Figure 2.5.a)
- **External labels** are next to the object in empty space connected to it via a line that attaches from the label to the object. (Figure 2.5.b)
- **Boundary labeling** could be described as a form of external labeling with the extra limitation of ordering the external labels along a rectangle around the object (Figure 2.5.c)
- **Excentric labeling**, instead of the previous methods which try to label the whole structure this technique by Fekete and Plaisant [FP03], is based on a movable focus

region where the labels are displayed differently depending on the location of the region (Figure 2.5.d)

- **Necklace maps** eliminate the classic notion of labels that need to be associated through a line or an internal position instead creating for each object a symbol that are organized along a curve around the object with proximity, size and color resembling to the actual object (Figure 2.5.e)

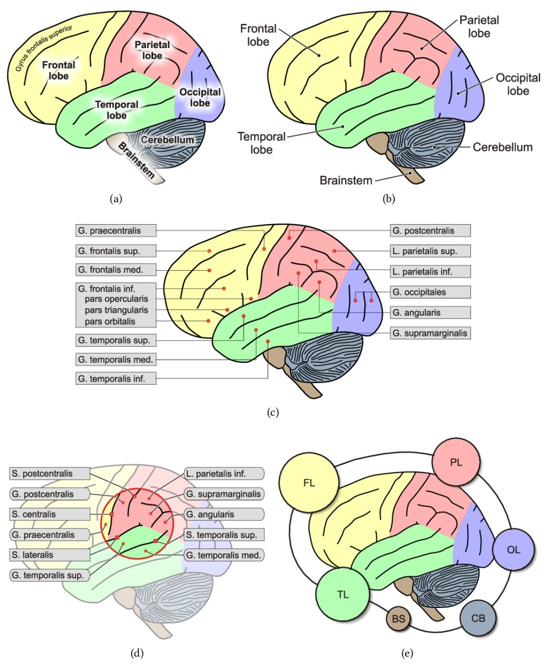


Figure 2.5: Survey of labeling techniques [OJP14] alphabetically indicated: (a) internal labels, (b) external labels, (c) boundary labeling, (d) excentric labeling, (e) necklace maps

The labeling technique closest to related with the task of this thesis is internal labeling as it places texts inside of the object rather than marking it from outer bounds or using other elements than text to convey the connection between object and annotation.

2.3.1 View-Management

Bell at al. [BFH01] introduced a view-management component for virtual environments. In this case view-management describes the process of adding labels to a 3D environment (a 3D model of a city) while making sure that no labels are occluding objects or conflicting with other objects or labels. This idea of view-management gets tested and expanded by Azuma et al. [AF03] where 4 different label placement algorithms are compared to each other and ranked based on performance and other features. Hartmann et al. [HAS04] is also concerned with view-management but primarily focused on the occlusion part of the equasion as they use three-dimensional landscapes as their input. Maass et al. [MD06] developed a way to place three-dimensional labels into a model of a city that get placed at different heights and sizes depending on the occlusion situation between the labels.

2.3.2 Contemporary Internal labeling

Multiple works in the field of internal labelling are related to the topic of the study. As mentioned previously [GAHS05] uses three-dimensional models as input. The internal labels produced by Hartmann et al. are limited to two-dimensional labels. Ropinski et al. [RPRH07] introduced three-dimensional shape cues for internal labels that try to convey the shape of the object through applying perspective transformations onto the labels so that they appear to be attached to the model directly. Jian et al. [JNH$^+$13] expands on the idea of three-dimensional shape cues and proposes a method to annotate vascular structures in volume rendered images. The work limits the possible annotations to vascular structures only but through that achieves a performant way of rendering such annotations onto a vascular structure in the same time as the volume rendered image would take anyway. Cipriano et al. [CG08] take the approach of projecting the labels onto the surface of the object but provide a text scaffold, a surface around the object that gets flattened if needed for the sake of readability. Kouril et al. [K$^+$19] also describes a form of internal labelling as the structures they are trying to label crowded 3D environments where a normal form of labelling does not suffice. Regions get segmented and labeled throughout instead of labelling a specific object exactly.

CHAPTER 3

Typogram Generator

The web application developed for this thesis encompasses a file upload field, a three-dimensional model viewer, and a control panel. This panel houses various user interface (UI) elements, organized into distinct folders that facilitate the customization of the SVG typogram rendering process.

Upon uploading of a model in a compatible format, the model is displayed in the three-dimensional model viewer. Within this viewer, the viewport can be adjusted according to the preference of the user. The resulting SVG typogram can be tailored via the UI options, while the user is simultaneously provided a preview of aspects that will be present in the final output.

Following the generation of the SVG, this can be further modified using the UI options and subsequently downloaded. The download options include the standalone SVG or accompanied with an image of the current viewport for context and subsequent editing. Additionally, an option is provided to convert the created SVG text elements back into paths, enabling its use in any subsequent editing software. This functionality is particularly useful given that certain editing tools (such as Figma, Adobe, and Affinity) do not fully support text path elements, leading to potential loss of text if opened in these platforms.

3.1 Typogram Creator Pipeline

The process outlined in Figure 3.6 details the transformation of raw data, either labeled or unlabeled, into a typogram designed for various applications. The software developed in this thesis covers the central segment of this pipeline, while other open-source tools handle the initial and final stages of the process.

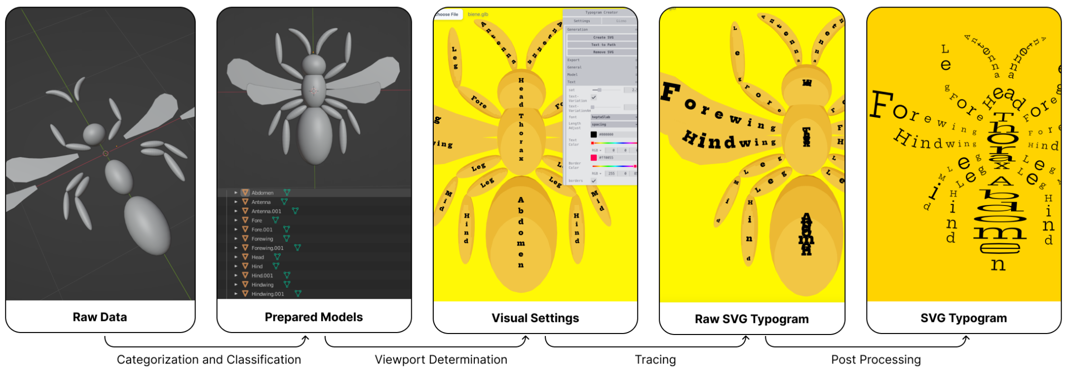


Figure 3.1: Overview of the Typogram Creator pipeline, illustrating a five-stage workflow. The process starts with raw data, which is then categorized and classified to create prepared models. These models are used to determine the visual settings. Tracing is applied to generate a raw SVG typogram, which is further refined through post-processing to produce the final SVG typogram.

3.1.1 Model Preparation

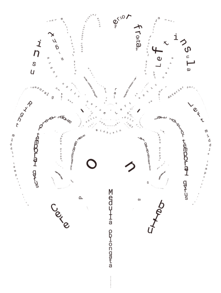
To ensure an effective outcome from the web application, it is important to pre-process the model, tailoring it to the specific use case. Often, three-dimensional models obtained from online sources are not designed with a focus on applications that heavily rely on labeling, such as the one under consideration. The model should encompass distinct components, each associated with an appropriate title to accurately represent the object. This responsibility of ensuring accurate labeling falls upon the early stages of three-dimensional model creation.

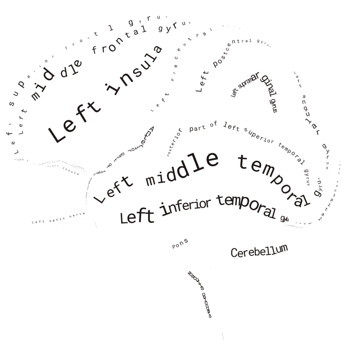
Although the process of creating such a model might seem complex initially, it has proven to be significantly less challenging then expected, especially for medical applications. This ease comes from the availability of scientific three-dimensional model libraries, for example the Life Science Database Archive (LSDB) Brain Pharmacology 3D (BP3D) Database [Lif23]. These resources offer a range of three-dimensional models illustrating different body parts at various anatomical levels (skeletal, muscular, organs, and so forth). These models, downloadable as .obj files, can subsequently be seamlessly amalgamated into more extensive .glb files, integrating multiple labelled structures. Such a process can be effectively executed using open-source software, such as Blender.

3.1.2 Viewport Determination

The issue of viewport determination was initially envisioned to be resolved programmatically at the outset of the project. However, through subsequent development and testing of the application, it became apparent that algorithmic setting of viewports would not confer substantial benefits to the user, given the creative and exploratory process inherent in designing a typogram. Unlike a traditional labeling process where "goodness for recognition" (highlighting the most prominent features while avoiding significant occlusion) [BFS+18] is the primary objective, the creation of a typogram necessitates a considerable degree of aesthetic consideration.

Minimizing the occlusion of sub-objects might seem like an intuitive approach for automatic viewport setting; however, testing revealed this not to be the case. Many objects are only visible in somewhat unconventional viewport choices, which are not conducive to presenting a typogram (Figure 3.2a). The objective of this thesis was to represent the true object, without the use of any borders to provide shape cues to the user (Figure 3.2b). This led to the conclusion that the most familiar or aesthetically pleasing viewport choices would be superior, as they can accurately represent the object even when the provided information is minimal, which is the characteristic of a typogram.


(a) Automatic viewport selection based on occlusion minimization of sub-objects



(b) Manual viewport choice after exploration

Figure 3.2: Comparison of automatic and manual viewport selections: The difference between algorithmic occlusion minimization and a subjective choice grounded in typogram aesthetics.

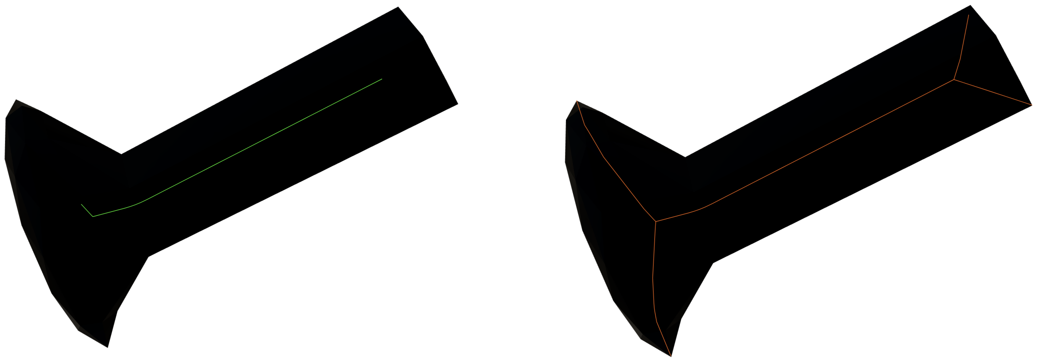
To help the user visualize where which text is going to be located in the real typogram, after the tracing process, small labels are already shown in the real-time view when the preview flag is enabled (Figure 3.8).

3.1.3 Tracing and Labeling with Visual Settings

In order to trace an SVG typogram within the selected viewport, users are only required to activate a button, given that the default settings align with the requirements of the majority of use cases. Subsequently, the web application undertakes the process of individually rendering an image for each visible object, incorporating occlusion with other objects to isolate the visible structure of each item.

These images are then traced using an algorithm known as Potrace, an abbreviation for "polygon tracer" [Sel03]. This algorithm is designed to yield high-quality results for any given high quality black and white bitmap. The completion of this process results in refined, closed SVG paths in the majority of tested scenarios, with each SVG path representing a part of the model that requires labeling.

To appropriately locate this label, the centerline of each object is computed using the scale axis transformation [GMPW09]. This technique mitigates the occurrence of rogue ends that typically arise when applying a medial axis transformation [CCMW97] to arbitrary two-dimensional shapes. The scale axis transformation can be controlled by the user with a parameter and is at default set to 2.5 which yielded acceptable results for most paths, as shown in Figure 3.3a. If this parameter is increased more of the rogue ends are cut off but there could also be mistakes in generation if the parameter is too high. Setting it to 1 results in the standard medial axis transform output shown in Figure 3.3b.



(a) Scale axis transformation of arbitrary object

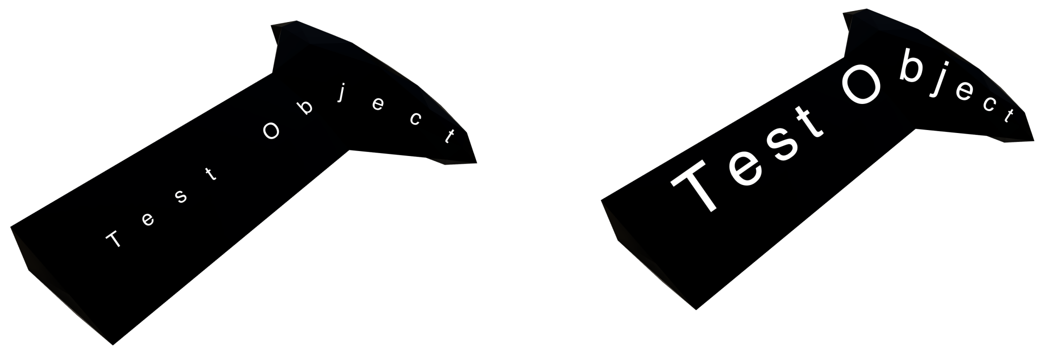
(b) Medial axis transformation of arbitrary object

Figure 3.3: Scale axis transformation compared to medial axis transformation

Following the determination of this general path and the initial placement of the text, an algorithm was implemented that assesses the length of the text and evaluates the space within the path, taking into account the placement location of the letters. Subsequently, each letter is sized differently based on its surrounding environment (Figure 3.4).

However, this method may lead to individual letters occupying excessive space in the direction of the text path, especially if label names are long (Figure 3.5a). To alleviate this issue, a process of gradual scaling is employed, which proportionally adjusts all letters to ensure that variations in size at different points of the object continue to be represented through the individual sizes of each character (Figure 3.5b).

This happens regardless of visual settings. In this step of the pipeline there are a lot of different settings to be configured by the user if needed that can be categorized:



(a) Uniform font size for each character

(b) Variation between font size in each character

Figure 3.4: Comparison between text placement on path with or without variation in font sizes depending on each letters position inside the object

Long TestObjectTitle
Long Test Object Title

(a) Long label without gradual scaling of characters

(b) Long label with gradual scaling of characters

Figure 3.5: Comparison with and without gradual scaling process to alleviate characters overlapping each other

- **General**
  - *Background Color:* Allows modification of the color for the downloadable background image.
- **Model**
  - *Show Textured Model:* Enables the display of the model with its intrinsic texture. Note: tracing becomes disabled when this feature is activated.
  - *Material Type:* Permits alteration by selecting from an array of available materials.
  - *Colour:* Defines the hue of the selected material.
- **Text**

  - *Scale Axis Transform Threshold Level:* Decides the extent to which the extremities of a Medial Axis Transformation are addressed. Overemphasizing this may result in no path recognition.
  - *Text Variation Activation:* Determines whether the text variation algorithm is employed.
  - *Text Variation Degree:* Dictates the level of text variation and its adaptability to the size of the encompassing path along different points of the Axis.
  - *Length Adjust:* Specifies the SVG lengthAdjust property preferred by the user - options include solely spacing or spacingAndGlyphs.
  - *Text Color:* Alters the hue of the SVG text generated during tracing.
  - *Font Family:* Grants the user the liberty to select from an array of font families provided by Google Fonts [Fon].

- **Borders**
  - *Tracing Borders (Boolean):* Decides if borders are to be traced and visible.
  - *Border Colour:* In instances where borders are traced, this setting determines their color.

3.1.4 Post-Processing

Once the SVG is downloaded, there are multiple options for further refining the results. Minor generation errors can be corrected using tools that handle SVG textpaths and tspan elements. it is important to choose tools that recognize these specific SVG features. Many leading tools have their own formats for aligning text along paths, which can lead to issues displaying the generated SVGs. However, when a compatible tool is selected, manual adjustments are usually minor, leading to a satisfactory result.

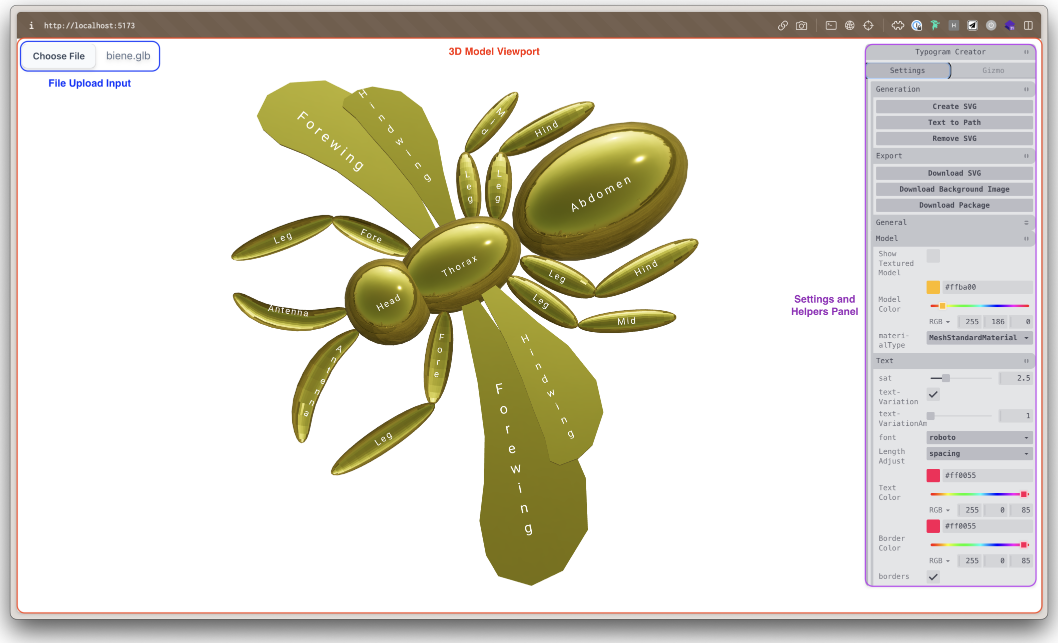
Given the limited software that correctly renders and edits SVG text-path elements, a recommended post-processing step is to generate non-overlapping text-paths. Tools similar to Boxy SVG [Fok23], can then be used to convert the characters in these text paths into path elements. This process turns all text into distinct path elements.

With this transformation, the resulting SVG can be edited in a wide range of software applications, given that standard paths are universally recognized and supported across popular editing platforms.

3.2 User Interface

The designed interface for this thesis aims to provide flexibility and freedom in settings for various use-cases. While the focus remains on offering maximum customization, the platform also emphasizes presenting straightforward defaults. This ensures the application remains accessible and not overly complex for first-time users.

The Interface comprises three primary sections:



Show Textured Model: ☐
Model Color: #ffba00, RGB 255 186 0
materialType: MeshStandardMaterial
sat: 2.5
textVariation: ✓
textVariationAm: 1
font: roboto
Length Adjust: spacing
Text Color: #ff0055, RGB 255 0 85
Border Color: #ff0055, RGB 255 0 85
borders: ✓

Figure 3.6: Web application interface of the typogram creator

- **File Upload Input**: Positioned prominently at the top left of the screen, this input allows users to select a file from their system for display within the software. It supports both .glb and .gltf file formats.

- **3D Model Viewport**: After file selection, the chosen model appears in this viewport, which spans the entire screen, ensuring maximum visibility. The design of the viewport is intuitive, enabling users to manipulate the model (e.g., rotate, zoom, drag) in a manner consistent with most software featuring similar interaction capabilities.

- **Settings and Helpers Panel**: Serving as the primary interactive hub of the software, this panel houses functions to tailor the SVG output. Users can navigate through its multiple folders based on their immediate requirements, ensuring a streamlined experience. Moreover, the panel comprises two distinct tabs, further categorizing its functionalities.

3.2.1 Settings

Within the settings section, users can control and produce desired visual outputs.

- **Generation**

  - *Create SVG (Button)*: Launches the tracing process, pausing all previews and related settings to ensure a seamless operation.
  - *Remove SVG (Button)*: Resets the SVG, reinstating interactivity to the 3D viewport, making it ready for any further tracing operations.

- **Export**

  - *Download SVG (Button)*: Saves the SVG as it appears in the viewport to the users designated download location.
  - *Download Background Image (Button)*: Saves the presented model (textured or otherwise) as a PNG, offering a choice between a transparent or selected color background.
  - *Download Package SVG + Image (Button)*: Saves the simultaneously the SVG and its background image.

- **Model Settings**: Detailed in section 3.1.3, this part encompasses choices related to model color, material, and texture.

- **Text Settings**: Detailed in section 3.1.3, this section covers configurations such as text variation, length adjustment, font, and color.

- **Border Settings**: Detailed in section 3.1.3, users can determine border render specifications, including thickness and color.

3.2.2 Helpers

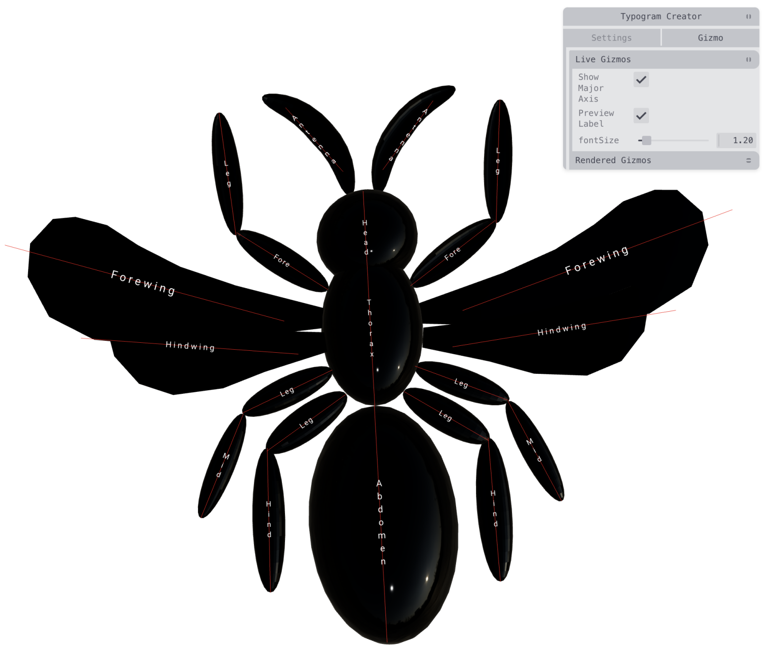
Referred to as "gizmos", these are designed tools meant to assist users during typogram development by providing distinct visual indications and labels.

- Live gizmos

  - *Show 3D Principle Axis (Boolean)*: Renders the deduced 3D principle axis overlaid on the model components.
  - *Show Preview Label (Boolean)*: Appends dynamic annotations to the 3D Model in alignment with the principle axis of each section.
  - *Preview Font Size (Slider)*: Modifies the font size of the annotations, using the measurements of the oriented bounding box combined with a user-specified multiplier to ensure size coherence.

- Traced gizmos

  - *Visible Scale Medial Axis (Boolean)*: Exposes the textpath element created by the scale medial axis conversion.



Typogram Creator

Settings | Gizmo

Live Gizmos

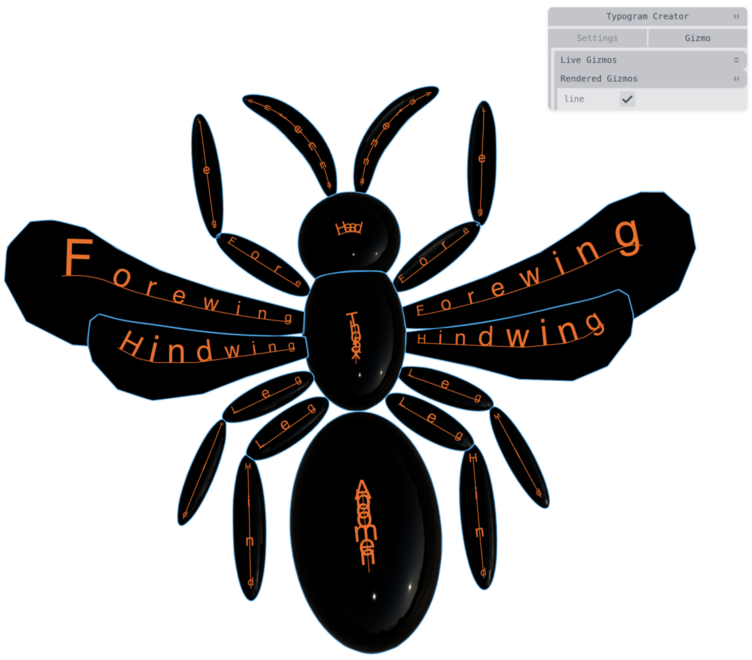
Show Major Axis ✓

Preview Label ✓

fontSize 1.20

Rendered Gizmos

Figure 3.7: Live gizmos shown in a simplified model of a bee. 3D principle axis in red, preview labels shown in white aligned at the principle axis.



Typogram Creator

Settings | Gizmo

Live Gizmos

Rendered Gizmos

line ✓

Figure 3.8: Traced scale medial axis shown in orange in a simplified model of a bee.

CHAPTER 4

Implementation

This research work introduces a web application developed using Svelte, TypeScript, and the 3D Framework, Threlte. Threlte acts as a wrapper for the widely acknowledged Three.js library [Thr23], offering integration with the Svelte Ecosystem. Within the application, users can upload a 3D model directly from their file system. Upon selection, the model displays in full screen within the viewport, offering interaction capabilities such as zooming, rotating, and translating through the three.js OrbitControls component. This interaction component is versatile, supporting various input mechanisms including trackpads, mice, and ball mice.

Once a model is uploaded, it is decomposed into its discrete components. Each of these components is individually analyzed and processed. Additional helper variables are computed for every model since they are not inherently provided by the three.js Framework.

The centroid of each sub-object is deduced by examining the buffer geometry. This approach provides a more effective starting point for internal labeling, harmonizing better with object shapes. The primary axis in the 3D domain is deduced through a series of transformations, calculations involving the covariance matrix, and subsequent rotations and translations.

To effectively size live labels, an Oriented Bounding Box is generated, leveraging previously calculated eigenvectors, centroid, and original vector positions of the buffer geometry.

4.1 Settings and Helpers Panel

One of the key interaction interfaces is its so called GUI (Graphical User Interface) panel, facilitated using the Tweakpane GUI library [Kok23]. This library offers an assortment of built-in input types, all of which were instrumental for this research. The application predominantly employs buttons, sliders, and boolean inputs, structured into two main tabs: **Settings** 3.2.1 and **Helpers** 3.2.2. Within the settings tab, users can access five distinct folders, a design choice to streamline the user experience by organizing features.

4.2 SVG Generation

To generate an SVG typogram, the application first renders an image for each labeled sub-component of the uploaded object. Concurrently, it ensures that all other components are rendered as transparent yet occluding entities. Each distinct image is then processed using the potrace algorithm, as described by Selinger [Sel03]. The implementation used is a Node.js-compatible version accessible at `https://github.com/tooolbox/node-potrace`, initially translated to JavaScript by @kilobtye `https://github.com/kilobtye/potrace`. This algorithm yields closed paths that are incorporated into an SVG canvas equivalent in size to the viewport.

Every closed path undergoes a Scale Medial Axis Transform (SAT) [GMPW09], aiming to identify its centerline. The transformation is facilitated by the medial axis transform (MAT) library, conceptualized by Floris Steenkamp. This library amalgamates Choi et al.'s MAT methodology [CCMW97] and Giesen et al.'s SAT technique [GMPW09], selected for its efficacy in minimizing undesirable terminations. Multiple path elements are merged into one, organized in the SVG, and grouped appropriately. Text is split character-wise, with each character represented as a TSPAN element, aligned with the path as an SVG text-path. The text orientation, either vertical or horizontal, is contingent on the cumulative angle of all sub paths.

When the letter variation mode is active, it employs an algorithm to utilize data generated by the MAT library. This involves mapping maximum disks at specific line points with two contact points. For each character, the nearest maximum disk acts as a font sizing reference. Given that this process may modify the central position of a character when aligned on the path, an iterative re-calibration mechanism is deployed to ensure no overlaps occur among characters.

CHAPTER 5

Results

This chapter details the results obtained from the development and application of the proposed computer-aided workflow for typogram creation. The assessment of the tool encompasses its performance, effectiveness, and usability across various three-dimensional models and design scenarios.

The primary outcome of this thesis was the development of a web based application to optimize the workflow of creating a typogram graphic starting from a 3D Model without any limitations on size or amount of objects and also without any constraint on which type of object can be used with the application. This application can be found at `https://typograms.rippl.at`.

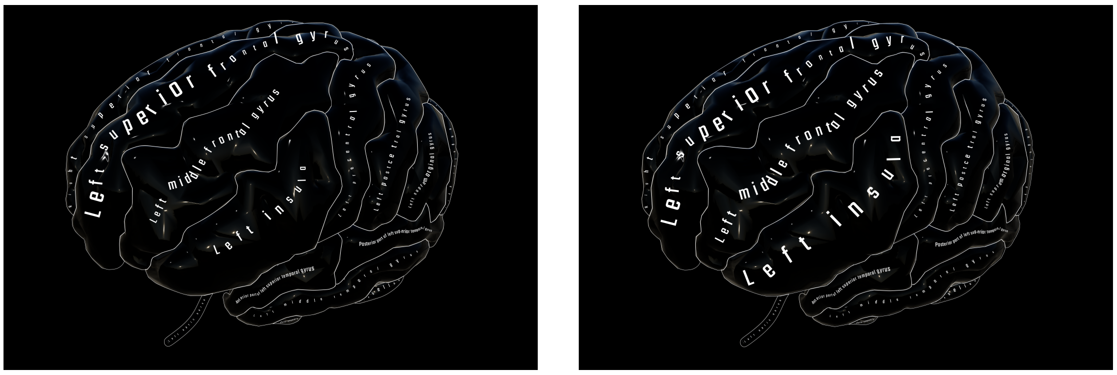
To evaluate the performance of the web application across different object types, we conducted several experiments. These tests helped identify the conditions under which the application excels and pinpointed specific shapes that currently require manual adjustments.

Organic shapes, for example a model of a brain (Figure 5.1b) or also a skeleton (Figure 5.2b), behaved well in the conditions and resulted in acceptable SVG files after the first run. With some manual post-processing these results can be easily improved further aesthetically.

Completely round shapes and square shapes the medial axis transformation performed very poorly as finding the centerline in objects that are very conform to those shapes results in no result at all or unusable results at best. (Figure 5.5)

In our model, we designed the wall behind the window as multiple segments rather than one solid object (Figure 5.3). This approach improves accuracy by facilitating better text placement. If the wall were modeled as a single object, its centerline would be circular, making effective placement of the short label 'WALL' challenging.

To compare the technical workflow against the hand made typograms an effort was made to recreate one of the typograms developed by Artist Aaron Kuehn (Figure 5.4a). A model representing the bee was created and adjusted with appropriate label names. The model was loaded into the application and traced and labelled automatically. Afterwords the head, thorax and abdomen were tweaked by hand as they performed worse than the other elements and a adjusted to fit the style of Kuehn's artwork. (Figures 5.1 and 5.2)



(a) Automatically generated through workflow (b) After minor manual adjustments

Figure 5.1: A brain model from the Life Science Database Archive (LSDB) [Lif23] containing 75 objects stitched together and relabeled

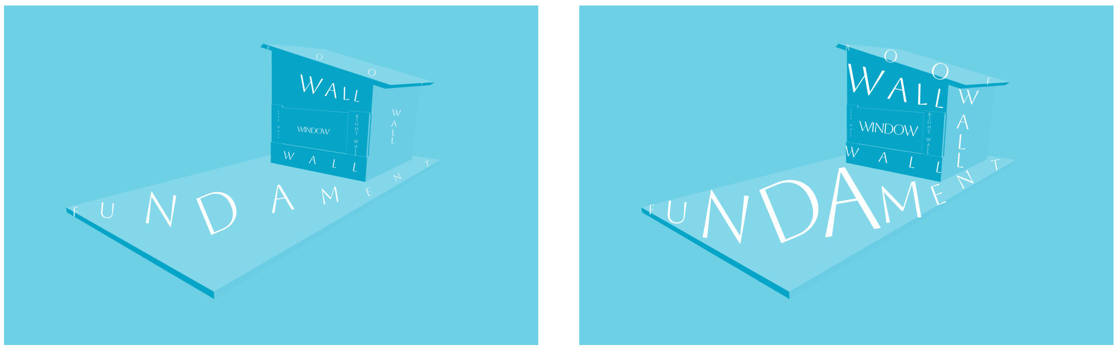
radius
second metacarpal
proximal phalanx
Middle phalanx
Distal phalanx

(a) Automatically generated through workflow (b) After minor manual adjustments

Figure 5.2: A skeleton model of a human hand from the Life Science Database Archive (LSDB) [Lif23] containing 29 objects stitched together and relabeled

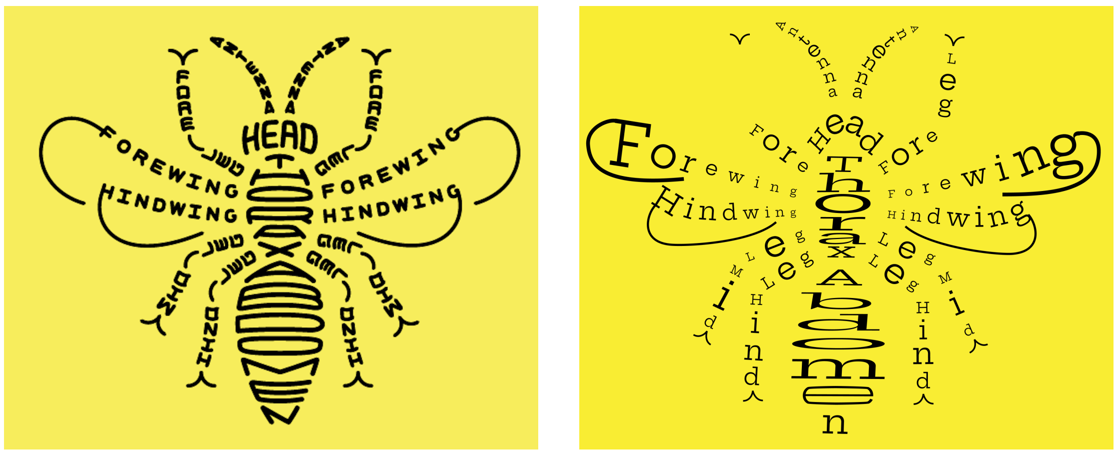
5.1 Performance

The computer-aided workflow significantly reduced the time required to produce typograms compared to traditional methods. On average, there were noticeable time savings across diverse models and complexity levels as the first setup of doing a typogram without a the tool can take a up a long amount of time. The automated process suc-



(a) Automatically generated through workflow (b) After minor manual adjustments

Figure 5.3: A custom built house model for this use-case



(a) Handmade typogram by Aaron Kuehn (b) Automatically generated and readjusted through the proposed workflow

Figure 5.4: A comparison between the by hand created typograms of Aaron Kuehn and the typogram created through the automatic computer aided workflow with readjustments

cessfully generated starting points for typograms for a wide variety of three-dimensional models.

The proposed system offered a wide range of options for customization, including borders, fill, text-stretch, and text-size-variation. The ability to experiment with different viewports and configurations fostered creativity and allowed users to tailor the typograms to specific needs and aesthetics.

The SVG output formats compatibility with popular illustration tools facilitated further manual adjustments if required. This feature maintained the continuity with existing design workflows and preserved the human touch in the final products.

In this system, the time required to process data varies significantly depending on the type of input and selected viewport. As a result, any real world performance measurement is inherently biased and cannot serve as a reliable estimation for any arbitrary object. To nonetheless provide users with a rough estimate of what to expect when using the application with a complex model, the skeletal hand model depicted in Figure 5.2 from the Life Science Database Archive (LSDB) was duplicated within a single file to reach a certain number of objects. It was then tested through the application using standard settings. The resulting processing times are listed in Table 5.1 below

Table 5.1: Processing time estimates based on object count in the skeletal hand model from LSDB [Lif23], shown in Figure 5.2

Amount of Objects	Processing Time (ms)
10	1600
50	16700
100	35600
500	207200
1000	376700

5.2 Limitations and Areas for Improvement

While the system performed admirably in most cases, certain limitations were identified. Simple topologies occasionally led to errors that required substantial manual correction. As the generation of centerlines often failed to yield any result in equidistant shapes such as circles, squares or triangles (Figure 5.5). This does not pose a major problem because most of the shapes where it failed to produce an acceptable result where simple shapes that can be done easily and quickly by hand in the post-processing step, but should be adjusted for future versions of the application. To further evaluate the areas of improvement, usability studies with artists and general audiences must be conducted. These should focus on the ease of use of the system, the learning curve, and how well it integrates into the workflow of artists. Additionally, the current implementation might not accommodate all possible design styles or preferences, and further customization options could enhance its versatility.

5.3 Contribution to the Field

The computer-aided workflow developed in this thesis represents a significant advancement in the automation of typogram design. By transforming a labor-intensive and complex process into an accessible and efficient platform, it opens new avenues for the application of typograms across various domains, such as education and visualization.

The results of this work demonstrate the potential of the proposed tool to not only simplify typogram creation but also inspire new creative possibilities by reducing the barriers to entry and fostering an explorative design approach.

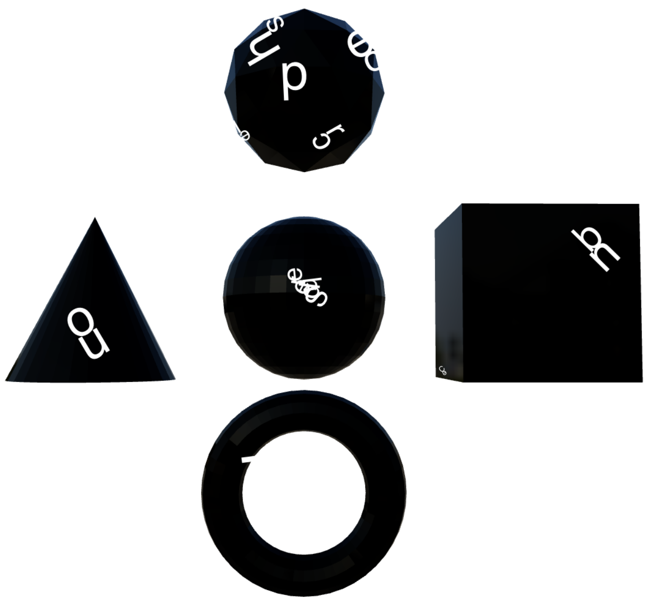


Figure 5.5: Scale medial axis failures on simple symmetrical shapes

CHAPTER 6

Conclusion

This thesis aimed to explore the potential of a computer-aided workflow for typogram creation. The outcome was the development of a web-based application designed to optimize the creation process of typograms from 3D models. The results showcased the efficiency, versatility, and compatibility of this new approach.

The main benefit of this work is the notable reduction in time required to produce typograms. The automated system provides a solid foundation for designers, allowing them to initiate their designs with minimal manual adjustments, particularly for intricate organic shapes. The systems integration with existing design tools through the SVG output format further ensures its practicality and utility for contemporary designers.

However, the current system does have limitations. Issues were encountered with some geometric shapes, particularly in the medial axis transformation phase. These areas of concern, primarily associated with simpler shapes, can be addressed with manual corrections.

Considering its broader implications, this thesis contributes to making typogram design more approachable and streamlined. By simplifying the creation process, it enables a larger audience to experiment and contribute to this design realm.

6.1 Future Work

The findings and developments presented in this thesis pave the way for further enhancements:

- **Improvement for simple geometric shapes**: Given the challenges with certain geometric shapes, future versions of the tool can focus on implementing a different method than medial axis transformation for equidistant shapes such as spheres, triangles and cubes as the scale medial axis transform does not produce satisfying results for these.

- **Enhanced customization options**: To cater to a wider range of design preferences, it would be beneficial to introduce more customization options in subsequent iterations, such as a wider selection of fonts, the addition of custom fonts or a wider selection of materials with customizable properties.
- **Integrated editing options**: Instead of transferring the typogram to another software to make manual adjustemts the goal for the next version of the application would be to integrate path editing and conversion from text to path directly into the application
- **User experience enhancements**: The interface and overall user experience can be further refined to ensure intuitiveness and efficiency by conducting tests and reviews with artists and designers that are working in the field of typogram creation.

In summary, this thesis offers a new approach to typogram design, and with continued refinement and feedback, there is potential for it to play a significant role in the evolution of typogram design.

List of Figures

2.1 Origins of typograms 4
2.2 Typical single-word typograms making a pun with typographical elements 5
2.3 Complex typogram systems 5
2.4 Image from Max Lipitz processed through the TIS [Lip21] 6
2.5 Survey of labeling techniques [OJP14] alphabetically indicated: (a) internal labels, (b) external labels, (c) boundary labeling, (d) excentric labeling, (e) necklace maps 7

3.1 Overview of the Typogram Creator pipeline, illustrating a five-stage workflow. The process starts with raw data, which is then categorized and classified to create prepared models. These models are used to determine the visual settings. Tracing is applied to generate a raw SVG typogram, which is further refined through post-processing to produce the final SVG typogram. . . . 10
3.2 Comparison of automatic and manual viewport selections: The difference between algorithmic occlusion minimization and a subjective choice grounded in typogram aesthetics. 11
3.3 Scale axis transformation compared to medial axis transformation 12
3.4 Comparison between text placement on path with or without variation in font sizes depending on each letters position inside the object 13
3.5 Comparison with and without gradual scaling process to alleviate characters overlapping each other 13
3.6 Web application interface of the typogram creator 15
3.7 Live gizmos shown in a simplified model of a bee. 3D principle axis in red, preview labels shown in white aligned at the principle axis. 17
3.8 Traced scale medial axis shown in orange in a simplified model of a bee. . 17

5.1 A brain model from the Life Science Database Archive (LSDB) [Lif23] containing 75 objects stitched together and relabeled 22
5.2 A skeleton model of a human hand from the Life Science Database Archive (LSDB) [Lif23] containing 29 objects stitched together and relabeled . . . 22
5.3 A custom built house model for this use-case 23
5.4 A comparison between the by hand created typograms of Aaron Kuehn and the typogram created through the automatic computer aided workflow with readjustments 23

5.5 Scale medial axis failures on simple symmetrical shapes 25

Bibliography

[AF03] R. Azuma and C. Furmanski. Evaluating label placement for augmented reality view management. In *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.*, pages 66–75, 2003.

[Apo66] Guillaume Apollinaire. *Calligrammes: Poemes de la Paix et de la Guerre.* 1966.

[BFH01] Blaine Bell, Steven Feiner, and Tobias Höllerer. View management for virtual and augmented reality. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, UIST '01, page 101–110, New York, NY, USA, 2001. Association for Computing Machinery.

[BFS$^+$18] Xavier Bonaventura, Miquel Feixas, Mateu Sbert, Lewis Chuang, and Christian Wallraven. A survey of viewpoint selection methods for polygonal models. *Entropy*, 20(5):370, May 2018.

[CCMW97] Hyeong In Choi, Sung Woo Choi, Hwan Pyo Moon, and Nam-Sook Wee. New algorithm for medial axis transform of plane domain. *Graphical Models and Image Processing*, 59(6):463–483, 1997.

[CG08] Gregory Cipriano and Michael Gleicher. Text scaffolds for effective surface labeling. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1675–1682, 2008.

[FM15] Thomas Fischer and Marian Macken. Architectural typograms in a cross-language architectural foundation class. In D Fassi, editor, *Emerging Practices: Inquiry into the Developing*, pages 130–140, Shanghai, 10 2015. Tongji University Press. Copyright: Tongji University Press.

[Fok23] Jarosław Foksa. Boxy svg. `https://boxy-svg.com/`, 2023. Accessed: 2023-09-03.

[Fon] Google Fonts. Google Fonts Library. `https://fonts.google.com/`.

[FP03] Jean-Daniel Fekete and Catherine Plaisant. Excentric labeling: Dynamic neighborhood labeling for data visualization. In BENJAMIN B. BEDERSON and BEN SHNEIDERMAN, editors, *The Craft of Information Visualization*, Interactive Technologies, pages 316–323. Morgan Kaufmann, San Francisco, 2003.

[GAHS05] Timo Götzelmann, Kamran Ali, Knut Hartmann, and Thomas Strothotte. Form Follows Function: Aesthetic Interactive Labels. In Laszlo Neumann, Mateu Sbert, Bruce Gooch, and Werner Purgathofer, editors, *Computational Aesthetics in Graphics, Visualization and Imaging*. The Eurographics Association, 2005.

[GMPW09] Joachim Giesen, Balint Miklos, Mark Pauly, and Camille Wormser. The scale axis transform. *Proceedings of the Annual Symposium on Computational Geometry*, 06 2009.

[GS16] Henry Gray and Susan Standring. *Gray's Anatomy: The Anatomical Basis of Clinical Practice*. Elsevier, London, 41 edition, 2016.

[Hal16] Dalia-Ruth Halperin. Micrography - a jewish art. Web Article, Feb 2016. Published in: British Library.

[HAS04] Knut Hartmann, Kamran Ali, and Thomas Strothotte. Floating labels: Applying dynamic potential fields for label layout. In Andreas Butz, Antonio Krüger, and Patrick Olivier, editors, *Smart Graphics*, pages 101–113, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[Her38] G. Herbert. *The Temple: Sacred Poems and Private Ejaculations*. Works of George Herbert. Pickering, 1838.

[JNH$^+$13] Zhengang Jiang, Yukitaka Nimura, Yuichiro Hayashi, Takayuki Kitasaka, Kazunari Misawa, Michitaka Fujiwara, Yasukazu Kajita, Toshihiko Wakabayashi, and Kensaku Mori. Anatomical annotation on vascular structure in volume rendered images. *Computerized Medical Imaging and Graphics*, 37(2):131–141, 2013. Special Issue on Mixed Reality Guidance of Therapy - Towards Clinical Implementation.

[Kok23] Hiroki Kokubun. Tweakpane: Compact gui for fine-tuning parameters and monitoring value changes. `https://github.com/cocopon/tweakpane`, 2023.

[Kue11] Aaron Kuehn. Skeleton typogram. `https://aaronkuehn.com/art/skeleton-typogram`, 2011. Modified: Mar 17, 2020.

[K$^+$19] David Kouřil, Ladislav Čmolík, Barbora Kozlíková, Hsiang-Yun Wu, Graham Johnson, David S. Goodsell, Arthur Olson, M. Eduard Gröller, and Ivan Viola. Labels on levels: Labeling of multi-scale multi-instance and crowded 3d biological environments. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):977–986, 2019.

[Lif23] Life Science Database Archive (LSDB). Brain Pharmacology 3D (BP3D) Database. `https://lifesciencedb.jp/bp3d/`, 2023.

[Lip21] Maxmillian Lipitz. The typographical imaging system. tis art it is. 05 2021. Preprint available at `https://www.researchgate.net/publication/351334796_The_Typographical_Imaging_System_TIS_Art_It_Is`.

[MD06] Stefan Maass and Jürgen Döllner. Efficient view management for dynamic annotation placement in virtual landscapes. In Andreas Butz, Brian Fisher, Antonio Krüger, and Patrick Olivier, editors, *Smart Graphics*, pages 1–12, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[MW23] Merriam-Webster. Calligram, 2023. Accessed: 2023-07-26, `https://www.merriam-webster.com/dictionary/calligram`.

[OJP14] Steffen Oeltze-Jafra and Bernhard Preim. Survey of labeling techniques in medical visualizations. In *Proceedings of the 4th Eurographics Workshop on Visual Computing for Biology and Medicine*, pages 199–208, 2014.

[Qui84] William A. Quinn. "the windhover" as "carmen figuratum". *The Hopkins Quarterly*, 10(4):127–143, 1984.

[RPRH07] Timo Ropinski, Jörg-Stefan Praßni, Jan Roters, and Klaus H. Hinrichs. Internal labels as shape cues for medical illustration. In *International Symposium on Vision, Modeling, and Visualization*, 2007.

[Sac18] Kekelinbsp; H. Sackey. Typogram no. 2, Oct 2018. Dribbble `https://dribbble.com/shots/5395201-Typogram-No-2`.

[Sel03] Peter Selinger. Potrace : a polygon-based tracing algorithm. 2003. Avaliable at `https://potrace.sourceforge.net/potrace.pdf`.

[Thr23] Three.js Contributors. Three.js Library. `https://threejs.org/`, 2023.

[TM17] Jose Juan Tablada and Rodolfo Mata. *Li-Po y Otros Poemas*. Universidad Nacional Autonoma de Mexico, 2017.

[Xu02] Bing Xu. Landscript. `https://artmuseum.princeton.edu/collections/objects/42723`, 2002. Medium: Sheet; ink on paper. Dimensions: Painting/Calligraphy: 51 x 76.2 cm. (20 1/16 x 30 in.). Gift of Henry and Patricia Tang, in honor of Wen C. Fong, Class of 1951 and Graduate School Class of 1958, and Constance Tang Fong. Culture: Chinese. Place Made: Asia, China. Signature: "Xu Bing" written in "square word calligraphy".