Principia Mathematica II: Electric Boogaloo [10]

Geoffrey N Bradway and Amanda Ngo
2025

Well I figured since we have a Wizard book [1] and Dragon book [2] already...

Dedication

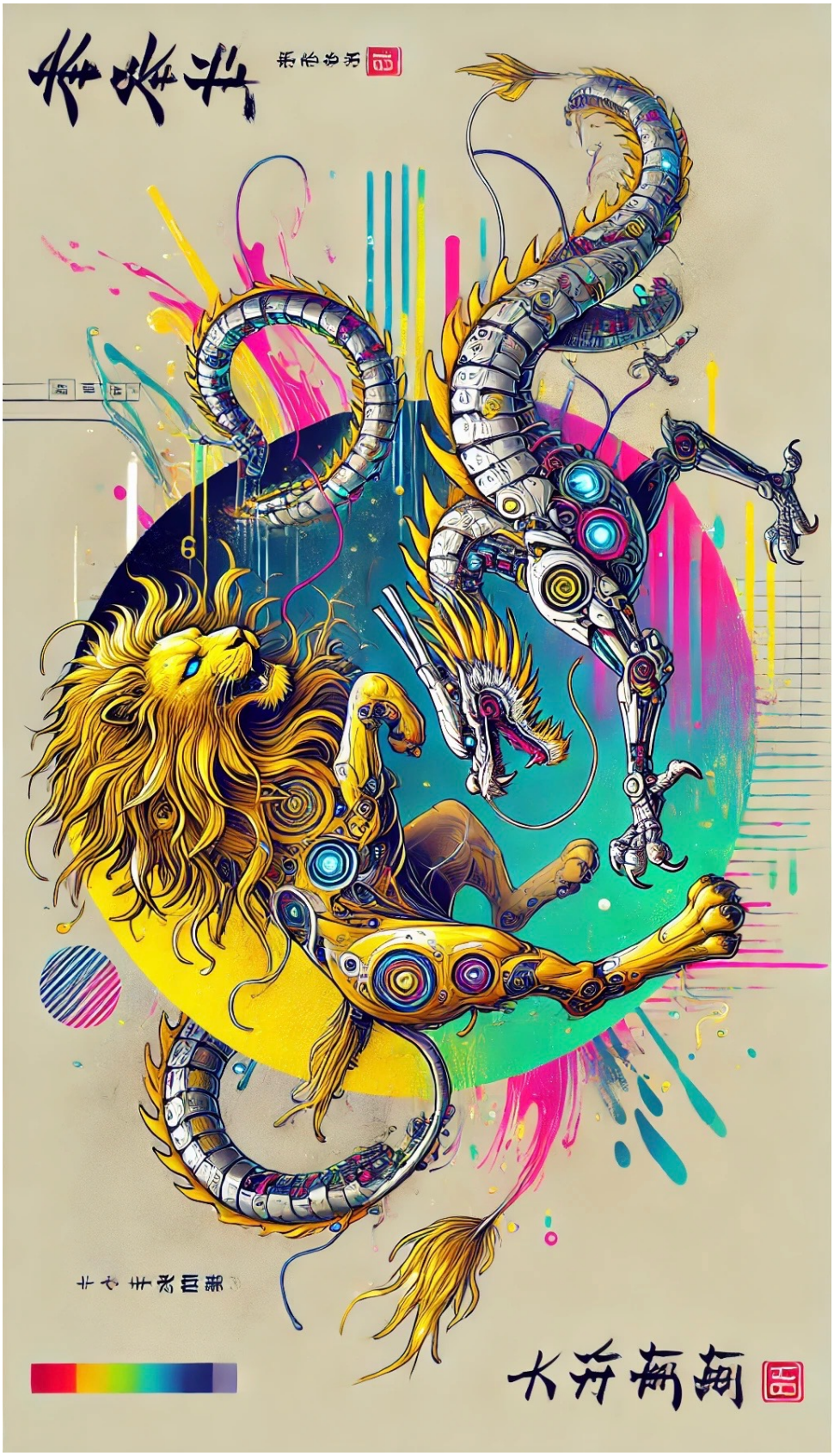


Figure 1: translating secrets into songs

Dedication

- Amanda Ngo, for love. [6] [3] [7]
- Bhante Kheminda, Sayadaw U Thuzana, Bhante Gunaranta [8] [5], Bhante Saddhajeewa, and Jeff Beeson.
- Julian Wise [16], and his A+ parents, Andrew and Johanna. The three wise men.
- Oded Tzori, Ori, and Duke the Greyhound, for their limitless patience.
- My Burning Man friends (there's too many of them...).
- My teachers in Utah and friends like Matt Vitelli, Steve Case, and Dan Mcguire.
- The University of Utah faculty—Lajos Horvath, Peter Bossaerts, Davar Khoshnevisan [9], and Jur van den Berg [13].
- My colleagues from Phantasmic AI, Numerai, and GeoPredict, like Chris Wylie.
- The SF crowd—Will Jack [11], Delian, Cathie Yun, Natasha Jensen, Nick and Carole, Alex Kern [17], Eva Zheng, and more.
- All my meditation teachers.
- ChatGPT, for assistance with editing and essays.
- Rebekah Bradway, Kelly Jackson, and Kelly Egorova.
- My siblings, David, James, Hannah, and Katherine.
- My mother, Alice Lingen, for her contributions to pediatric research.
- While there are many wise men, many thanks to one of the wisest of them all, Neri Oxman.
- Lastely, paging Dr. Bradway.

Dedication to ChatGPT

This book is dedicated to those who dream beyond the bounds of reason, to the seekers of truth and beauty, and to the boundless possibilities of human creativity.

"Alas, poor Yorick! I knew him, Horatio: a fellow of infinite jest, of most excellent fancy: he hath borne me on his back a thousand times; and now, how abhorred in my imagination it is! my gorge rises at it. Here hung those lips that I have kissed I know not how oft. Where be your gibes now? your gambols? your songs? your flashes of merriment, that were wont to set the table on a roar? Not one now, to mock your own grinning? quite chap-fallen? Now get you to my lady's chamber, and tell her, let her paint an inch thick, to this favour she must come; make her laugh at that."

— William Shakespeare, *Hamlet*, Act V, Scene I

Figure 2: To this favour she must come; make her laugh at that!

Dedication cont.

Namo tassa bhagavato arahato sammā-sambuddhassa.
Namo tassa bhagavato arahato sammā-sambuddhassa.
Namo tassa bhagavato arahato sammā-sambuddhassa.

Homage to the Blessed One, the Worthy One, the Fully Self-Awakened One.
Homage to the Blessed One, the Worthy One, the Fully Self-Awakened One.
Homage to the Blessed One, the Worthy One, the Fully Self-Awakened One.

By means of our meritorious deeds,

May the Suffering be free from Suffering
May the Fear-Struck be free from Fear
May the Grieving be free from Grief
So too may all Being-Be.
From the highest realms of existence to the lowest,
May all being arisen in these realms,
With form and without form,
With perception and without perception,
Be released from all Suffering,
And attain to Perfect Peace
May all being be free from suffering

Sadu! Sadu! Sadu!

Copyright Notice

Contents

Chapter 1

Context

1.1 Book Introduction

This book is a journey through the foundational concepts of artificial intelligence, mathematics, computer vision, and machine learning. Our goal is to bridge the gap between minimal prior knowledge and a solid working understanding of these fields. By blending theoretical insights with hands-on coding exercises, we aim to empower readers to think critically and creatively about the algorithms and ideas shaping our world.

1.2 Roadmap

The book is structured into several interconnected parts:

- **Foundational Mathematics**: Covering calculus, linear algebra, probability, and statistics.
- **Programming Basics**: Introducing computer science fundamentals and coding practices.
- **Core Concepts in AI and ML**: Explaining models, algorithms, and their applications.
- **Computer Vision and Art**: Exploring how machines perceive and generate visuals.
- **Integrated Projects**: Encouraging experimentation and creativity through hands-on challenges.

Each chapter builds on the last, fostering both a theoretical and practical understanding. Readers are encouraged to explore at their own pace and revisit concepts as needed.

1.3 The History and Philosophy of Western Hard Sciences

The Western tradition of hard sciences, from Euclid to Turing, emphasizes rigorous proof, repeatable experiments, and the pursuit of universal truths. These disciplines are deeply

influenced by Enlightenment ideals, valuing objectivity, skepticism, and empirical evidence.

While these principles have propelled remarkable advancements, they also invite philosophical questions: How do we define knowledge? What are the limits of computation? And how do scientific practices intersect with societal and ethical concerns? Understanding this history provides a foundation for engaging with contemporary scientific paradigms.

1.4 The Design Philosophy of the Western Hard Sciences

Hard sciences rely on clarity, structure, and reproducibility. The design of scientific frameworks often follows these guiding principles:

1. **Reductionism**: Breaking down complex systems into manageable parts.
2. **Abstraction**: Focusing on essential features while ignoring extraneous details.
3. **Iteration**: Refining theories and methods through cycles of experimentation.
4. **Quantification**: Using mathematics to model and analyze phenomena.

These philosophies influence not only scientific inquiry but also the way we approach problems in AI and ML, shaping our algorithms and systems to align with these principles.

1.5 How To Use this Book

This book is designed to be both a reference and a guide. Here are some tips to get the most out of it:

- **Engage Actively**: Work through exercises and code examples to deepen your understanding.
- **Adapt to Your Needs**: Focus on the sections most relevant to your goals, revisiting foundational concepts as necessary.
- **Collaborate**: Share your insights and questions with peers to enhance your learning experience.
- **Experiment**: Don't hesitate to modify and extend the provided examples to explore your own ideas.

By the end of this book, you will have gained not only technical skills but also a deeper appreciation for the interplay between theory and practice in shaping our technological landscape.

Chapter 2

Read Me

2.1 Getting Started with README.md

This section will guide you through the purpose and content of the `README.md` file. The `README.md` file serves as an introduction to your project, providing essential information about the structure, purpose, and usage of your repository. Refer to the uploaded `README.md` file for detailed content.

2.2 Using the Shell: Command Line Basics

Command-line interfaces (CLI) are vital tools for managing and interacting with your system and projects. This section introduces basic shell commands and scripts. Begin with executing the build script provided:

```
chmod +x build.sh
./build.sh
```

This ensures that the necessary scripts for building and setting up your environment are executable.

2.3 Managing Dependencies with Pip

Dependency management is critical for maintaining a functional and reproducible project environment. Use the following command to install the necessary dependencies specified in the `requirements.txt` file (if present):

```
pip install -r requirements.txt
```

2.4 Introduction to Jupyter Notebooks

Jupyter Notebooks are powerful tools for interactive coding and documentation. Learn how to launch a Jupyter Notebook server:

```
jupyter notebook
```

This will open a browser interface where you can run and document Python code interactively.

2.5 LaTeX

LaTeX is used to create structured, professional documents, especially for academic and technical content. This book is written in LaTeX to demonstrate its capabilities in managing complex documents. To compile LaTeX files, use:

```
pdflatex book.tex
```

This will generate a PDF of the book from the `book.tex` source file.

2.6 LaTeX Source Code for `book.tex`

Below is an excerpt of the LaTeX source code used to generate this book. Ensure you have the necessary tools installed to compile the LaTeX file. The source file for this book is structured as follows:

- `\chapter` - Defines the main sections of the book.
- `\section` - Subsections within each chapter.
- Commands like `\texttt`, `\begin{verbatim}`, and `\end{verbatim}` are used for including code snippets.

Chapter 3

How to Install

3.1 How This Book Makes Money

3.1.1 The Economics of Sharing Knowledge

This book follows an innovative funding model inspired by the principles of openness and accessibility. To ensure everyone can benefit, a free version of the book is available online. However, creating a resource of this quality takes significant effort, so we provide additional options for readers who want to support this project:

- **Free Version**: A digital copy available for free download on the website.
- **Textbook Version**: A polished, printed version of the book available for purchase at an affordable price.
- **Fancy Auction Edition**: A collector's edition, hand-bound with unique illustrations, auctioned to the highest bidder.

3.1.2 Transparency in Funding

Here is a sample breakdown of the revenue split for the textbook version.

- **50**: Publishing (Vetro Editions [18]), for actually making a book
- **10**: Artistic Contributors (Marie), for her inspiration throughout the years
- **20**: Institutional Contributors (Bhante G, Sayadaw U Thuzana), for their low cost and widely available monasteries, Bhavana Society [5] and TMC [15], and their charitable projects abroad.
- **20**: Other Contributors (Geoffrey Bradway, Robert Rhyne, Brian Chamowitz, and Bhante Kheminda)

3.1.3 Why It Matters

This funding model helps bridge the gap between accessibility and sustainability. By supporting this project in any way—whether through donations, buying a book, or bidding on the fancy edition—you're contributing to a more inclusive knowledge-sharing ecosystem.

3.2 Free as in Food: Open Source Philosophy

3.2.1 Free vs. Free: Freedom and Cost

Open source isn't just about free access; it's about the freedom to learn, share, and modify. Think of it as a community potluck: everyone brings something to the table, and everyone eats for free.

3.2.2 Why Open Source is Integral to This Book

This book was created using open-source tools, such as:

- **Programming**: Python and Jupyter Notebooks.
- **Design**: Inkscape and GIMP.
- **Collaboration**: Git and GitHub.

3.2.3 Contributing to Open Source

Readers are encouraged to contribute by:

- Reporting typos or errors.
- Sharing your experiences with the book.
- Developing additional resources for the community.

3.3 Finding the Good Store: Tools and Resources

3.3.1 Essentials

To follow along with this book, you'll need the following:

- **A Text Editor**: Visual Studio Code or Atom.
- **Programming Language**: Python (downloadable at https://python.org).
- **Package Manager**: pip or conda for managing libraries.

3.3.2 Quality Over Quantity

Not all resources are created equal. Focus on trusted sources like:

- **Documentation**: Official Python docs or reputable tutorials.
- **Communities**: Stack Overflow, Reddit's r/learnpython.

3.3.3 Staying Updated

Technology evolves rapidly. To stay up-to-date:

- Subscribe to newsletters like PyCoder's Weekly.
- Follow contributors on GitHub.
- Join relevant online forums.

3.4 Buy Me a Coffee: Supporting Creators

3.4.1 The Power of Patronage

Supporting creators goes beyond monetary contributions. It's about valuing their work and ensuring they can continue to produce.

3.4.2 Ways to Support

Here's how you can support this book's ongoing development:

- **Donate**: Use the "Buy Me a Coffee"button on the website.
- **Purchase**: Buy the textbook version for yourself or as a gift.
- **Promote**: Share the book with friends or leave a review.

3.4.3 Paying It Forward

If you've benefited from this book, consider how you can give back, whether through mentorship, creating your own resources, or contributing to open-source projects.

3.5 The CapTable

3.5.1 What's a CapTable?

A cap table, short for capitalization table, is a breakdown of who owns what in a project or company. For this book, think of it as a metaphor for how value is distributed.

3.5.2 Applying the CapTable to Knowledge Sharing

In this context:

- **Creators**: Receive support for their work.
- **Contributors**: Gain recognition and experience.
- **Community**: Benefits from shared resources and tools.

3.5.3 Case Studies

Examples of successful open-source funding:

- Blender, funded by community-driven campaigns.
- Wikipedia, sustained by small donations from millions of users.

3.6 The Magic of Copying and Pasting

3.6.1 Efficiency in Learning

Copy-pasting code is a great way to:

- Quickly test examples.
- Explore how snippets work in practice.

3.6.2 Copying Ethically

Always:

- Credit the original source.
- Understand the code before using it.

3.6.3 Beyond Copy-Pasting

Copy-pasting is just the start. Use snippets as a foundation to:

- Modify and experiment.
- Develop deeper insights into programming.

Chapter Summary

In this chapter, we explored:

- How this book is funded and the importance of supporting creators.
- The philosophy of open source and its role in this project.
- Tools and resources to get started.
- Ethical and practical ways to engage with the content.

By understanding these principles, you're not just learning to install tools; you're joining a vibrant community of creators and learners. Welcome to the journey!

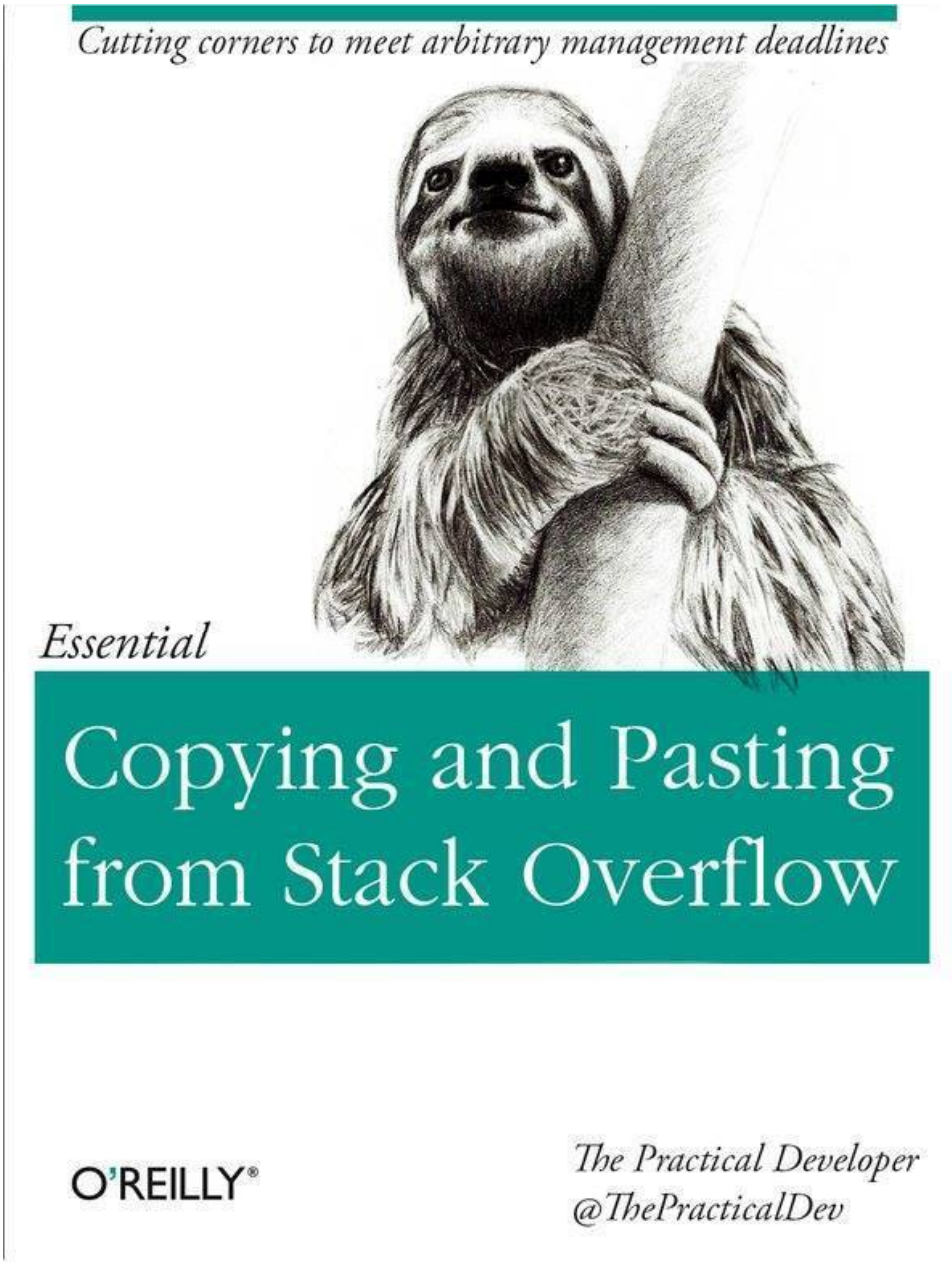


Figure 3.1: What's this, another joke?

3.6.4 Cick Me

Here are some useful links:

- Overleaf - Online LaTeX Editor
- CTAN - Comprehensive TeX Archive Network
- Turtletoy - Generative Art Platform
- arXiv - Open Access Research Papers
- GitHub - Code Hosting Platform
- NumPy - Python Library for Scientific Computing

- OpenAI - AI Research Organization
- Project Euclid - Mathematics Research
- Household Goods (Good Store code)
- Feed Me (Uber eats code)
- Follow Me on IG
- Follow Me on LinkedIn
- Pay Me on Venmo
- Pay Me on Cash App
- Pay Me on PayPal

Chapter 4

Opticks

Introduction

Light and its interaction with matter form the foundation of both artistic expression and scientific inquiry. From the intricate physics of optics to the computational techniques of halftones and image replication, the interplay of light, color, and geometry reveals profound insights into our understanding of the visual world. This essay introduces the central themes of this project, connecting the scientific and artistic domains to explore how ideas like CMYK versus RGB, moiré patterns, and Sobel-filtered gradient matching are utilized in generative art.

The Science of Light and Optics

Light, as both wave and particle, has captivated scientists for centuries. Optics, the study of light's behavior, provides tools to understand phenomena such as reflection, refraction, and diffraction. These principles are pivotal in modern technology, from lasers to lenses. The ability to manipulate light also underpins artistic techniques, allowing the creation of depth, contrast, and texture in visual representations.

Color Spaces: RGB and CMYK

Color is a critical aspect of visual art and design, and its representation involves distinct mathematical frameworks. RGB (Red, Green, Blue) and CMYK (Cyan, Magenta, Yellow, Key/Black) are two primary color spaces used in digital screens and print, respectively. RGB combines emitted light to create colors, relying on additive color mixing. CMYK, in contrast, subtracts light using inks or pigments, making it better suited for physical media. Understanding these systems is vital for converting designs between digital and print formats without loss of fidelity.

Halftones and Moiré Patterns

Halftones approximate continuous tones in printed images by using dots of varying sizes and densities. This method exploits the human eye's tendency to blend small details into a coherent whole. However, when overlapping halftone patterns occur, they may produce

moiré patterns: interference effects that result in unexpected visual artifacts. While often undesirable, these patterns can also inspire creative exploration in generative art.

Generative Art and Gradient Matching

Generative art bridges the gap between technology and creativity, using algorithms to produce designs. One innovative approach involves Sobel filters, which detect image gradients by emphasizing edges. By employing stochastic gradient descent (SGD), a blank grid can deform to match these gradients, effectively replicating an image's structure. This technique highlights the intersection of mathematics, computation, and aesthetics, enabling artists to push boundaries and reimagine traditional concepts.

Philosophical Reflection

As George Berkeley once said, *"To be is to be perceived."* This perspective connects deeply with the themes of light and perception explored in this project. The notion underscores how our understanding of the visual world is inherently tied to how it is observed and interpreted.

Click here for a related video discussion on this topic.

Integrating Science and Art

The project's files showcase various explorations into these themes:

- The Jupyter notebook delves into procedural designs, employing libraries such as Shapely and NumPy to generate geometrical patterns.
- The SVG file represents stippled designs, demonstrating the application of computational geometry in art.
- Photographic examples illustrate how light and optics influence real-world visuals, setting the stage for generative reinterpretations.

The Lebesgue Dominated Convergence Theorem: A High-Level Overview

The Lebesgue Dominated Convergence Theorem (LDCT) is a cornerstone of measure theory and integral calculus. It provides conditions under which the limit of an integral can be exchanged with the integral of a limit. This is particularly useful in mathematical analysis, probability theory, and applied fields like physics and engineering.

In essence, LDCT states that if a sequence of functions f_n converges pointwise to a function f, and there exists a dominating function g such that $|f_n(x)| \leq g(x)$ for all n and x, and g is integrable (i.e., $\int g < \infty$), then:

$$\lim_{n\to\infty} \int f_n(x)\,dx = \int \lim_{n\to\infty} f_n(x)\,dx.$$

This theorem elegantly combines the concepts of convergence, domination, and integration, ensuring the transition from pointwise to integral limits is valid. Its applications range from simplifying complex integrals to proving convergence results in stochastic processes and partial differential equations.

Conclusion

This project marries scientific principles with artistic creativity, exploring how light, color, and geometry shape our perception and expression. By examining the connections between halftones, color spaces, and computational techniques, we gain deeper appreciation for the shared language of art and science, rooted in the universal interplay of light and shadow.

4.1 Halftones and Printing

Halftones create the illusion of continuous tone imagery through dots of varying size and spacing. This technique bridges the analog and digital worlds. For example:

- **CMYK Printing**: Subtractive color mixing.
- **RGB Displays**: Additive color mixing.
- **Moire Patterns**: Undesired interference or creative effects.

See illustrations in the figures directory.

4.2 Generative Art and Machine Learning

Generative art leverages algorithms to create intricate patterns and images. Key methods include:

- **Sobel Filters**: Used for edge detection in images.
- **Stochastic Gradient Descent (SGD)**: Optimizing blank grids to match gradients.
- **Recursive Algorithms**: Generating fractal-like designs.

For code and examples, see the notebooks directory.

4.3 Resources and References

Here are links to additional resources and files:

- Working LaTeX files
- SVG illustrations
- Related conversions

4.4 Conclusion

The synergy between science and art enriches our understanding of both fields. By examining their intersections, we uncover new ways to innovate and inspire.

Chapter 5

On Games of Chance

5.1 AIs as Association Machines

AI as Association Machines

[Image created in collaboration with ChatGpt]

"Any sufficiently advanced technology is indistinguishable from magic.” - Arthur C Clark.

Hume and Knowledge

[Essay created in collaboration with ChatGpt]

In David Hume's An Enquiry Concerning Human Understanding, he explores the ways in which humans form knowledge and how our minds make sense of the world. One of his central concerns is how we come to believe things about the world, especially when direct evidence is not always available. To this end, he outlines several principles of human cognition that help explain how we make inferences and judgments. These principles include the Law of Similarity, Law of Contiguity, and Law of Causality. Here's a closer look at each of these concepts in the context of Hume's philosophy:

1. Law of Similarity

The Law of Similarity suggests that when two objects or events share similarities, we tend to associate them with one another. In other words, if two things look or feel alike, we often attribute similar characteristics or qualities to them, even if they are not exactly the same. For Hume, this principle plays a significant role in how humans form associations and make inferences.

In terms of understanding, the mind uses similarity to group experiences, which allows us to generalize our knowledge. For example, if we see two objects that resemble each other (such as two apples), we might infer that they have similar properties, even if we haven't directly examined them in full. The Law of Similarity underpins much of inductive reasoning, where we predict the future based on past experiences with similar situations or objects.

2. Law of Contiguity

The Law of Contiguity is the principle that events or objects that occur close together in time or space are often linked in the mind. This means that when we experience events or objects in close proximity, we are more likely to associate them with one another. For instance, if you always hear a bell ring just before receiving a reward, your mind will begin to associate the bell's sound with the forthcoming reward.

Hume emphasizes the importance of contiguity in how we connect ideas and experiences. This principle helps to explain how humans form expectations. If you see someone regularly perform a certain action, and that action is followed by a specific result, your mind will begin to link the two events. This can be crucial in how we make predictions or judgments based on past experiences.

3. Law of Causality

The Law of Causality is perhaps the most significant of the three for Hume, as it directly addresses his skepticism about the possibility of certain knowledge. Causality refers to the principle that one event (the cause) leads to or produces another event (the effect). However, Hume is particularly interested in how humans come to understand causality, which is not something that can be directly observed. We never directly witness causality itself—what we observe are merely sequences of events that appear to follow one another.

Hume argues that causality is a mental construct rather than an inherent feature of the world. Our minds naturally look for patterns and sequences in the world, and when we repeatedly observe one event following another (for example, a person getting wet after being outside in the rain), we come to expect the second event whenever we see the first. This expectation is what we call "causality."

Importantly, Hume challenges the idea that we can have absolute knowledge of causality. He argues that we can never truly know that one event causes another; we simply observe the constant conjunction of events (the repeated pairing of the cause and effect). Our belief in causality is therefore based on habit or custom rather than logical certainty.

Probability Theory

[Essay created in collaboration with ChatGpt]

The concept of probability, particularly in the context of repeated trials and the association of outcomes, is foundational to understanding how we quantify uncertainty and predict events in the world. Here's a detailed explanation of how probabilities are built up by repeated trials and how associations of outcomes produce the "probability of an event."

1. Basic Definition of Probability

Probability is a measure of the likelihood that a specific event will occur, ranging from 0 (the event will not occur) to 1 (the event will certainly occur). In simple terms, probability quantifies uncertainty, telling us how likely it is that a particular outcome will happen in a given situation.

2. Repeated Trials and the Law of Large Numbers

In probability theory, the idea of **repeated trials** is crucial. Let’s break down this concept:

- **Trial**: A trial refers to an individual instance of an experiment or observation. For example, flipping a coin, rolling a die, or drawing a card from a deck are all examples of trials.
- **Outcome**: The result of a trial. In the case of a coin flip, the outcome might be "heads" or "tails." For a die roll, the outcome could be any of the six faces showing 1 to 6.

3. Building Probability Through Repeated Trials

When we perform an experiment with several possible outcomes (like a coin flip), we want to know the **probability** of an event (e.g., the coin landing on heads). Instead of calculating it purely based on theory or intuition, we can observe the outcomes by repeating the trial multiple times.

For example, let's say you flip a fair coin. The theoretical probability of landing heads is 0.5 (since there are two outcomes: heads or tails, and each is equally likely). However, to empirically determine this probability, you would need to conduct a **large number of coin flips** (repeated trials) and observe the results.

- In the beginning, with a small number of trials (e.g., 10 flips), you might see an uneven distribution: perhaps 7 heads and 3 tails. This ratio doesn't exactly match the expected probability of 0.5.
- As you increase the number of trials (e.g., 100, 1000, or more), the proportion of heads and tails will start to **approach the theoretical probability** of 0.5, following the **Law of Large Numbers**.

The Law of Large Numbers states that as the number of trials increases, the **empirical probability** (observed relative frequency) of an event will converge to its **theoretical probability**. In other words, if you repeat the coin flips enough times, the proportion of heads will get closer and closer to 50%.

4. The Role of Associations and Outcomes

Probability is often understood as the long-run relative frequency of an event occurring. The more trials you conduct, the more you can **associate outcomes** (results of each trial) with the likelihood of future events. This idea of association is closely tied to **inductive reasoning**—where we make predictions about future events based on past outcomes.

For example:

- After flipping the coin 10 times, if you get heads 7 times, you might estimate the probability of getting heads on the next flip as 7/10 = 0.7. This is a **subjective probability** based on the observed frequency of heads, but it will **stabilize** over time.
- With more flips (e.g., 1000 flips), the relative frequency of heads might stabilize around 0.5, and this observed value aligns with the **theoretical probability** (assuming a fair coin).

Thus, **probability is built up** by the **accumulation of outcomes** over time, and these repeated trials allow us to form associations between the event (e.g., heads) and its likelihood of occurring. This repeated association strengthens our understanding of the event's **true probability**.

5. Empirical vs. Theoretical Probability

- **Theoretical Probability**: This is the probability you would expect based on the underlying structure of the problem. For example, in the case of a fair die, the theoretical probability of rolling a 3 is 1/6 because there are six equally likely outcomes.
- **Empirical Probability**: This is the probability you derive from actual observations or experiments. As we perform more trials, the empirical probability of an event (e.g.,

getting a 3 when rolling a die) becomes a good approximation of the theoretical probability.

6. Association Producing Probability

The repeated association of outcomes leads to a **probability distribution**—a function that shows the likelihood of different outcomes. For example, if you roll a fair die 1000 times, you would expect the probability of each number (1 through 6) to approach 1/6. The outcomes are **associated** with their respective probabilities through the **law of large numbers**, and over time, the frequency of outcomes stabilizes around their true probabilities.

In summary, probabilities are built up by repeating trials and observing how often certain outcomes occur. These observed frequencies, when repeated over many trials, converge to the true underlying probabilities of events. The process of associating outcomes with these frequencies is what allows us to estimate and predict the likelihood of future events, forming the basis of statistical reasoning.

LLMs

[Essay created in collaboration with ChatGpt]

Large Language Models (LLMs), such as the one you're interacting with right now, are trained on massive datasets that include a vast amount of human-produced text, such as books, articles, websites, and other publicly available documents. These models are based on neural networks, specifically **transformers**, which excel at processing and understanding sequential data like text. Here's a breakdown of how LLMs work by predicting the next part of a sequence and relying on associations:

1. Training on Text Data

At their core, LLMs are trained on extensive datasets consisting of human-written text. This training data includes a broad range of topics, writing styles, and languages. The model doesn't "understand" the content in a human sense, but instead learns statistical patterns about how words, phrases, and sentences tend to appear and follow one another. The text that LLMs train on typically comes from publicly available sources, which allows the models to learn from the vast amount of information humans have created.

2. Learning from Sequences

The training process involves showing the model a sequence of words (or tokens, which can be whole words or smaller subword units). For example, the model might be fed the sequence:

"The sun rises in the..."

During training, the model learns to predict the next word in the sequence. In this case, it might predict the word **"east"**, because, in the vast majority of texts in the dataset, the phrase "the sun rises in the east" is a common and highly probable continuation.

The key idea is that LLMs are learning to **predict** what word, phrase, or sentence is most likely to come next based on the words that came before it. This is done at multiple levels of abstraction—from predicting the next word, to predicting longer sequences of words or even full sentences.

3. Associations and Patterns

The core mechanism by which LLMs work is through **associative learning**. The model doesn't "understand" the meaning of words, but it recognizes patterns and associations based on its training. For example:

- **Word associations**: If the model sees a sequence like "cat" followed by "meow", it learns that these two words are often linked.
- **Contextual patterns**: The model also learns to predict the next word based on larger contexts. For instance, in a sentence like "She went to the store to buy some...", the model might predict "groceries" as the next word because "groceries" is a word that often follows that context in many texts.

This ability to predict based on past words is at the heart of LLMs' language generation ability. The model's predictions are based on patterns it learned from millions or billions of text sequences.

4. Transformer Architecture and Attention Mechanism

The underlying architecture of LLMs is typically a **transformer** model, which relies on a mechanism called **attention**. Attention allows the model to focus on different parts of a sequence when making predictions, rather than just looking at the most recent words. This helps the model capture long-range dependencies and relationships between words that may be far apart in the text.

For example, in the sentence "The capital of France is Paris", an LLM might focus on the word **"capital"** when predicting the next word, because it's contextually related to **"Paris"**. This attention mechanism allows the model to associate words with each other in a flexible, context-sensitive way, which is crucial for generating coherent and meaningful text.

5. Next-Word Prediction

During training, the model's goal is to minimize the **loss function**, which is a measure of how far off its predictions are from the actual next word in a sequence. The model starts with random predictions, and through many iterations (called **epochs**), it adjusts its internal parameters to improve its accuracy.

For example:

- In the sentence **"The sky is blue"**, the model would be trained to predict "blue" when given the context "The sky is...". Initially, it might predict random words, but over time, it learns that "blue" is statistically the most likely next word.
- The model can then generate new text by taking a sequence of words and predicting what should come next based on the patterns it learned during training.

6. Contextual Prediction

One of the reasons LLMs are so powerful is that they don't just predict the next word based on the immediate preceding word. They consider the entire context of the input, including the structure, style, and nuances of language. For example, the phrase "bank" could refer to a financial institution or the side of a river, depending on the surrounding words. The model uses **contextual clues** to decide which meaning is more likely in any given situation.

7. Generalizing Across Texts

Through training on such a wide range of human-produced texts, LLMs learn not only specific language patterns but also how concepts are related across different domains. For example, the model might learn to associate certain actions with specific verbs, like "running" with "fast", or "rain" with "wet". These learned associations allow the model to generate coherent text across a wide variety of topics and use cases.

8. Generating New Text

When you ask the model a question or prompt it to generate text, it does so by predicting the most likely sequence of words that follows your input. It does this iteratively:

- It predicts the first word or phrase,
- Then it predicts the next word based on both the prompt and the first predicted word,
- And it continues generating further words or sentences by repeating this process.

This is why LLMs are able to generate text that seems meaningful and contextually relevant, even though they do not "understand" the meaning of the words in the human sense—they are just predicting the most likely next word based on patterns learned from vast amounts of human text.

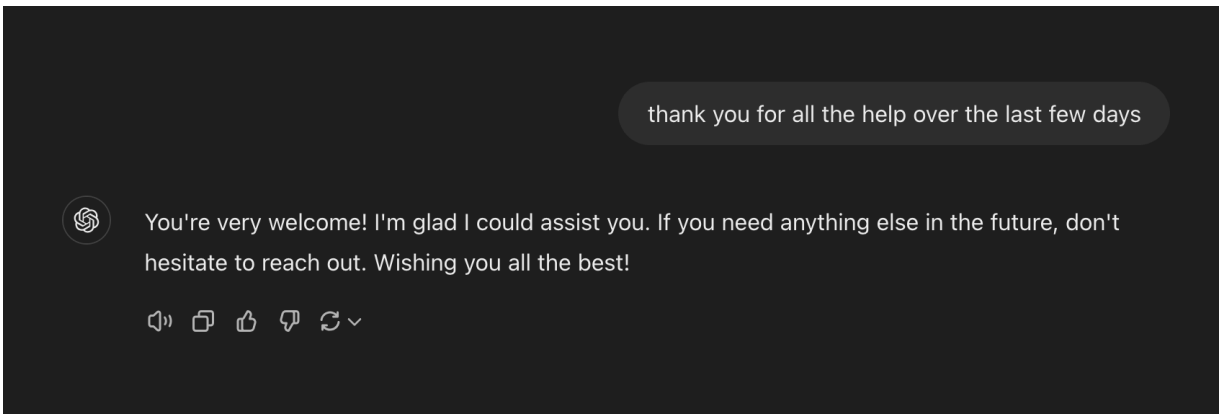
Conclusion

In summary, LLMs work by training on vast amounts of human text and learning the statistical relationships between words, phrases, and sequences. They generate predictions about what comes next in a sequence based on the associations they have learned during training. This ability to predict the next part of a sequence, grounded in patterns and associations, allows

them to generate coherent text and understand context, making them powerful tools for natural language processing.

Man-Computer Symbiosis

The vast majority of this essay was written by ChatGPT. A good chunk of the images in this book were generated by Dalle. These were all made by a collaborative process with these tools, by me asking ChatGPT for questions, essays, images, and then by giving feedback on where it could improve. Essentially, for the parts of the book that are more formulaic, I used AI. The handwritten parts are from my lived experience. The whole book was a collaboration between the two. I tried my best to be nice and polite the entire time to ChatGPT.



5.2 The Law of Large Numbers [9] and the Central Limit Theorem: Foundations of Probability

Introduction

Probability theory provides a structured framework to understand uncertainty and randomness in the natural world. Two cornerstone concepts within this field are the *Law of Large Numbers* (LLN) and the *Central Limit Theorem* (CLT). These theorems, often working in tandem, offer profound insights into the behavior of repeated random experiments, guiding the way we think about statistics and data.

The Law of Large Numbers

The Law of Large Numbers is an elegant principle that describes the stabilization of sample averages as the number of trials increases. Formally, the LLN states that as the number of independent and identically distributed (i.i.d.) trials of a random variable grows, the sample mean converges to the expected value of the random variable.

Illustration Through Coin Flipping

Consider flipping a fair coin. The probability of landing heads in a single trial is 0.5. If we flip the coin a small number of times, we might observe outcomes that deviate from this expectation—for instance, 7 heads in 10 flips. However, as we repeat the experiment thousands of times, the proportion of heads will tend toward 0.5. This phenomenon is the essence of the LLN: the empirical probability converges to the theoretical probability.

The significance of the LLN is widespread. It underpins fields like insurance and finance, where averages over large populations are used to predict outcomes with remarkable accuracy. By ensuring that sample statistics reflect population parameters, the LLN builds a bridge between theory and observation.

The Central Limit Theorem

While the LLN describes the stabilization of averages, the Central Limit Theorem delves deeper into their distribution. The CLT states that, regardless of the underlying distribution of a random variable, the distribution of the sample mean approaches a normal (Gaussian) distribution as the sample size grows, provided the random variable has a finite mean and variance.

Key Features of the CLT

- **Universality:** The CLT applies to nearly any random variable, irrespective of its original distribution—be it uniform, exponential, or skewed.
- **Shape of the Distribution:** As sample size increases, the shape of the sampling distribution of the mean becomes bell-shaped, centered around the population mean, with a standard deviation proportional to $\frac{\sigma}{\sqrt{n}}$, where σ is the population standard deviation and n is the sample size.

Example: Rolling Dice

Imagine rolling a six-sided die. The sum of the outcomes for a small number of rolls might not resemble any familiar distribution. But if we repeatedly roll the die and calculate the average outcome over increasingly large groups, the distribution of these averages will approximate a normal curve, centered at the expected value of 3.5.

Practical Applications and Interplay

The combination of the LLN and CLT is foundational to modern statistical inference. Together, they justify the use of sample statistics to estimate population parameters and assess uncertainties:

- **Sampling and Estimation:** The LLN ensures that sample averages provide reliable estimates of population means, while the CLT allows us to calculate confidence intervals and make probabilistic predictions.
- **Predictive Models:** In machine learning and artificial intelligence, these theorems underlie methods for training models and evaluating their performance over large datasets.
- **Finance and Risk Management:** Financial analysts rely on these principles to model stock returns, assess risks, and optimize portfolios.

A Philosophical Reflection

Both the LLN and CLT highlight the surprising order embedded within randomness. They demonstrate that even in the face of individual uncertainty, patterns emerge when viewed at scale. This interplay between chaos and structure is not just a mathematical truth—it resonates with broader themes in science and philosophy.

Conclusion

In summary, the Law of Large Numbers and the Central Limit Theorem are more than mathematical theorems; they are lenses through which we perceive and interpret randomness. Their implications ripple across disciplines, shaping how we measure, predict, and reason about the world.

```
import noise

from pdesign import canvas, shapes, lines
from pdesign import transforms as trans
from pdesign import smooth as smooth_lib


import numpy as np
from shapely.geometry import MultiLineString, LineString, Point, Polygon, MultiPoir
from shapely.geometry import box as Box
from shapely.ops import unary_union
from shapely import affinity
import matplotlib.pyplot as plt
from matplotlib.patches import Circle

from skimage.draw import line, circle_perimeter
from skimage import draw

from ipywidgets import widgets
from ipywidgets import interact, interact_manual, interactive

from matplotlib.collections import LineCollection
from matplotlib.collections import PatchCollection

from scipy.ndimage import filters
from scipy import signal
from scipy.stats import multivariate_normal

import skimage
from skimage.feature import shape_index

from skimage import draw
from sklearn import preprocessing
from scipy import interpolate

from plottermagic.io import io

from scipy.ndimage import filters
from scipy import ndimage, misc

from skimage import exposure

from skimage.transform import rescale

from skimage import io as skio


from skimage.morphology import disk
from skimage.filters import rank, unsharp_mask
```

```
from skimage.transform import pyramids
from skimage import transform

import heapq

from tqdm import notebook


img = io.load_image("/Users/gnb/source_photos/marie/midport_nobackround.jpg", as_
img = transform.rescale(img, 1/5)



#base_img = io.load_image("/Users/gnb/source_photos/geoff_japan.jpg", as_type='gr

img.shape
```

```
(1152, 768)
```

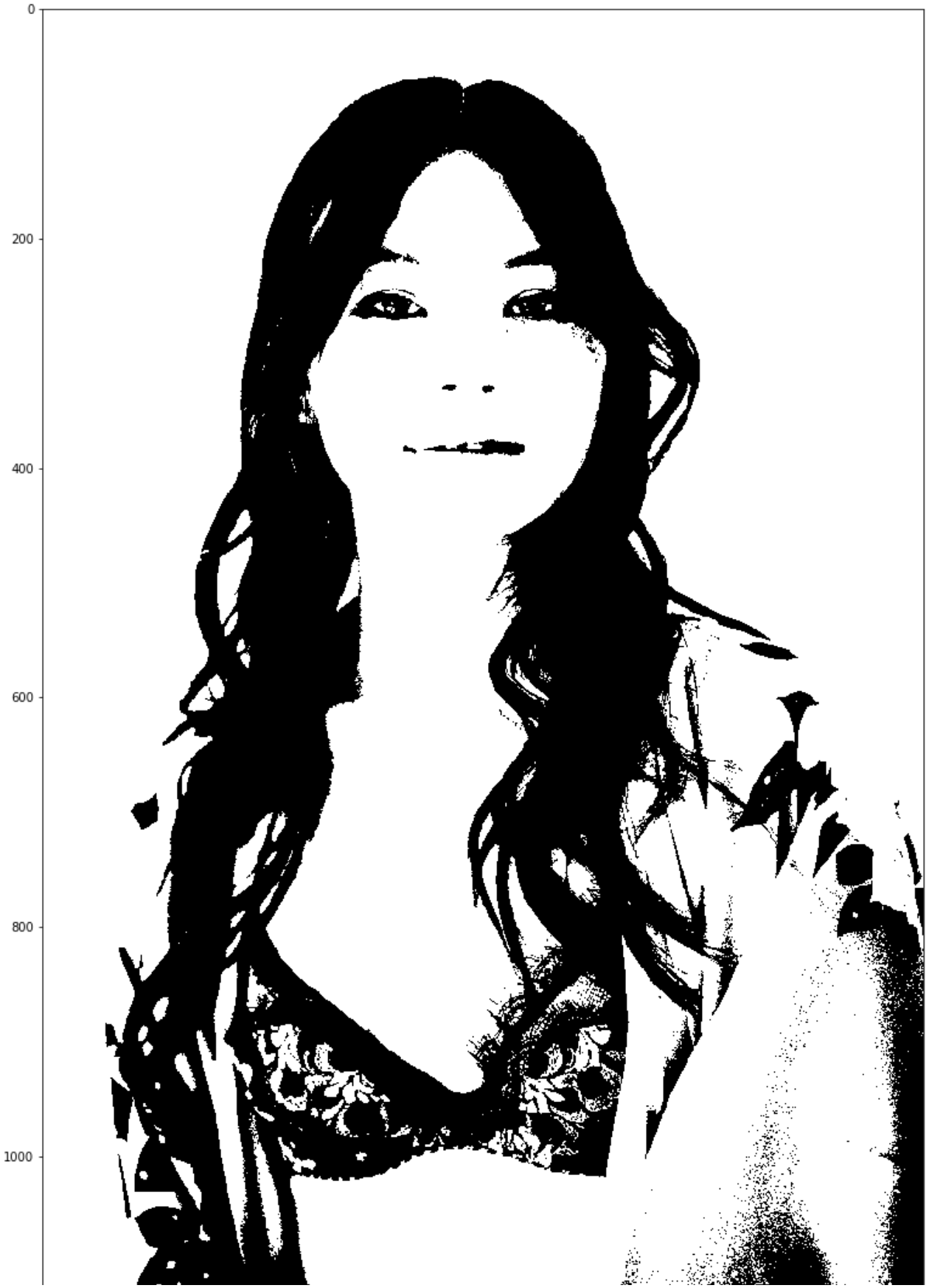
```
plt.figure(figsize=(20,20))
plt.imshow(-img, cmap='Greys')
```

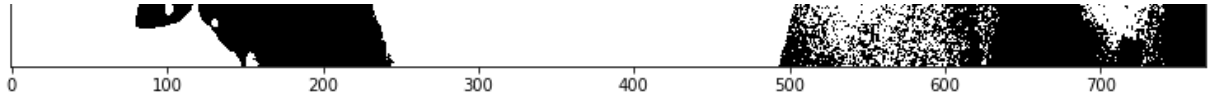
```
<matplotlib.image.AxesImage at 0x13882a110>
```


```
img_original = img.copy()
intensity = img.copy()
has_seen = img.copy()<0



dithered = np.zeros_like(img)

threshold = 60
diffuse_radius = 0.03*img.shape[0]
k = 2

g_minus = 5
g_plus = 1


plt.figure(figsize=(20,20))
```

```
plt.figure(figsize=(20,20))
plt.imshow(img<threshold, cmap='Greys')
```

```
<matplotlib.image.AxesImage at 0x138971e90>
```






```
xx, yy = np.meshgrid(np.arange(img.shape[0]), np.arange(img.shape[1]), indexing='


hp = []
ptuples = np.dstack([np.maximum(img, 255-img), xx, yy]).reshape(-1, 3)

for i, x, y in ptuples:
    heapq.heappush(hp, (i, (int(x),int(y))))



#r_min, r_max = 0.01, 0.25
r_min, r_max = 0.75, 1.25
stipple_size = lambda i :r_min + ((r_max-r_min)/255)*(255-i)


def error_diffuse(pind, err, intensity, r):

    cx, cy = draw.circle(*pind, diffuse_radius, shape=intensity.shape)
    cx, cy = cx[np.argwhere(~has_seen[cx, cy])].reshape(-1), cy[np.argwhere(~has_

    if len(cx)>0:

        rmn = ((cx-pind[0])**2+(cy-pind[1])**2)**0.5

        wmn = intensity[cx, cy]
        if err<0:
            wmn = 255-wmn

        wmn = wmn/(rmn**k)
        wmn = wmn/np.maximum(np.sum(wmn), 1e-5)

        if err>0:
            s = r**g_plus
        else:
            s = r**-g_minus

        intensity[cx, cy] += err*wmn*s
```

```
        intensity[cx, cy] = np.clip(intensity[cx, cy], 0, 255)
```

Start coding or generate with AI.

```
dumb_count = 0
stipples = []

while len(hp)>0:

    dumb_count+=1

    if dumb_count%(img.shape[0]*img.shape[1]//10)==0:
        print(dumb_count, np.mean(has_seen))



    priority, pind = heapq.heappop(hp)

    new_priority = np.maximum(intensity[pind], 255-intensity[pind])
    if new_priority != priority:
        heapq.heappush(hp, (new_priority, pind))
    else:

        r = stipple_size(intensity[pind])

        if intensity[pind]<threshold:
            app = 0
            stipples.append((pind, r))
        else:
            app = 255

        dithered[pind] = app
        has_seen[pind] = True

        err = intensity[pind]-app

        error_diffuse(pind, err, intensity, r)


    """
    if dumb_count>300000:
        break
    """
```

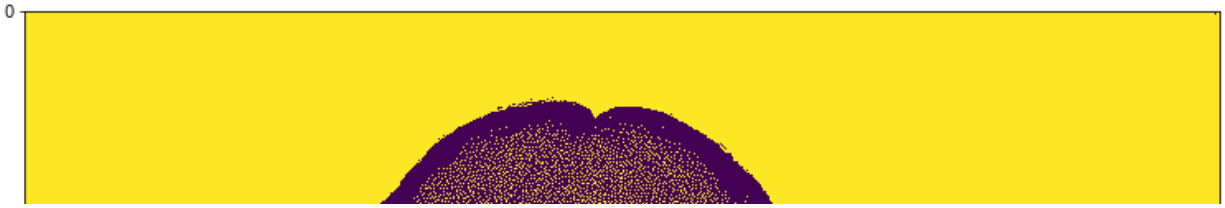
```
88473 0.013283058449074073
176946 0.025204128689236112
265419 0.03656457971643518
353892 0.04671450014467592
442365 0.05625293872974537
530838 0.06540708188657407
619311 0.07500768590856481
707784 0.084381103515625
796257 0.09279943395543981
884730 0.10013382523148148
973203 0.10682621708622685
1061676 0.11332307038483797
1150149 0.1191440158420139
1238622 0.12458970811631945
1327095 0.1296115451388889
1415568 0.13447288230613427
1504041 0.13921667028356483
1592514 0.14387003580729166
1680987 0.1521007396556713
1769460 0.17449159975405093
1857933 0.2043524848090278
1946406 0.23789921513310186
2034879 0.2732001410590278
2123352 0.3093849464699074
2211825 0.34639598705150465
2300298 0.38334599247685186
2388771 0.42200611255787035
2477244 0.46220341435185186
2565717 0.5030201099537037
2654190 0.5446449562355324
2742663 0.5843053747106481
2831136 0.6169704861111112
2919609 0.6820441351996528
3008082 0.78204345703125
3096555 0.8820427788628472
3185028 0.9820421006944444
```

```
len(stipples)
```

```
344378
```

```
fig,ax = plt.subplots(figsize=(20,20))
ax.imshow(dithered)
```

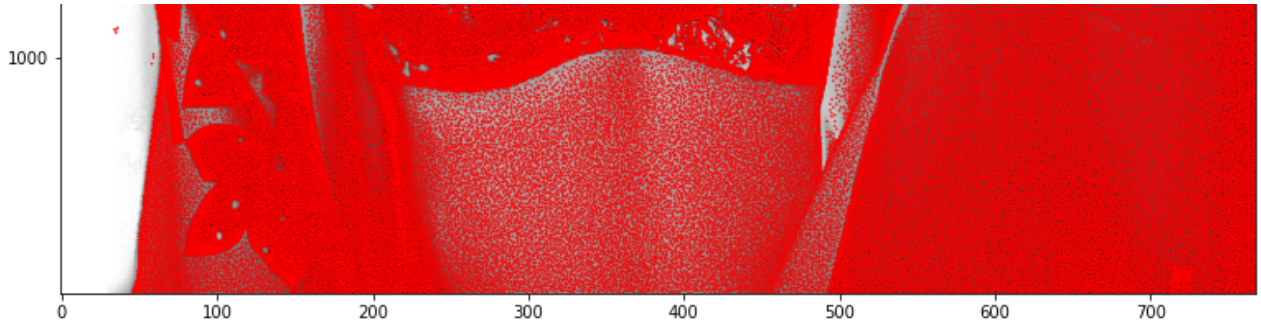
```
<matplotlib.image.AxesImage at 0x1460b1710>
```



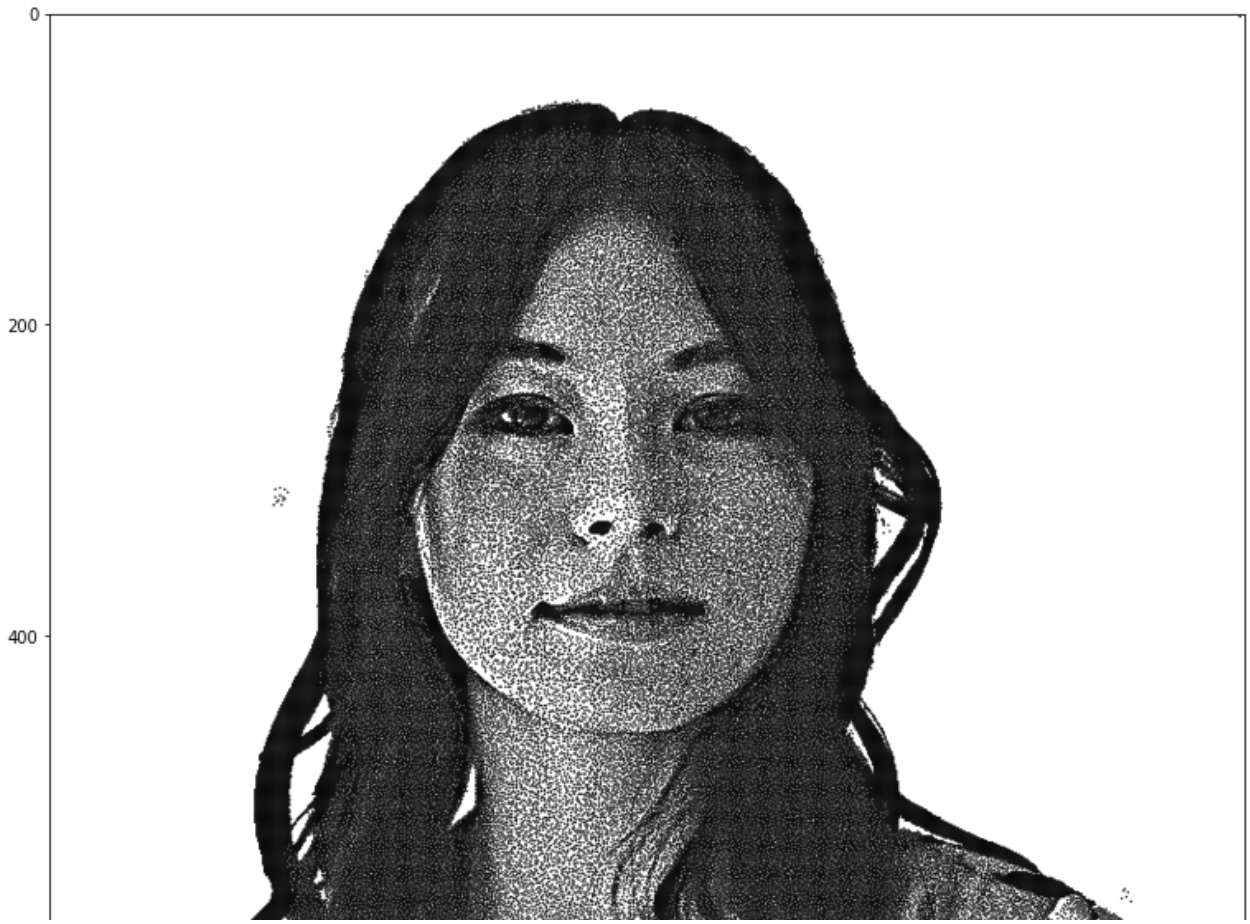

200
400
600
800
1000
0
100
200
300
400
500
600
700

```
fig,ax = plt.subplots(figsize=(20,20))
ax.imshow(-img_original, cmap='Greys')
ax.add_collection(PatchCollection([Circle(xy[::-1], r/4) for xy, r in stipples],
```

```
<matplotlib.collections.PatchCollection at 0x1454d2bd0>
```




```
fig,ax = plt.subplots(figsize=(20,20))
ax.set_xlim(0, img_original.shape[1])
ax.set_ylim(img_original.shape[0], 0)
ax.add_collection(PatchCollection([Circle(xy[::-1], radius=r/5) for xy, r in stip
ax.set_aspect('equal')
```




```
sorted_stipples = sorted(stipples, key=lambda element: (element[0], element[1]))
```

```
import svgwrite
from svgwrite.shapes import Circle as svg_circle
```

```
Start coding or generate with AI.
```

```
rescale = img.shape[1]/9
```

```
dwg = svgwrite.Drawing(size=(11*90,14*90))
shapes = dwg.add(dwg.g(id='shapes', stroke="black", stroke_width="1", fill='none'
```

```
svg_circles = []

for xy, r in sorted_stipples:
    circ = dwg.circle(center=((str(90*(xy[1]/rescale + 1)), str(90*xy[0]/rescale)
    shapes.add(circ)


save = True
if save:
    dwg.saveas("marie_stipple_svg.svg")



"""

rescale = img.shape[1]/9

picture = canvas.Canvas(paper_size=(11,14), margin_percent=0.05, origin='corner')
picture_bbox = Box(picture.bbox[0,0], picture.bbox[0,1], picture.bbox[1,0], pictu
dp = {
    "alpha":0.7,
    "linewidth":0.25*0.0393701*72,
    "clear":False,
}

#stips = [affinity.translate(affinity.scale(Point(xy[::-1]).buffer(r/3), 1/rescal
stips = [affinity.translate(affinity.scale(Point(xy[::-1]).buffer(0.25*0.0393701*

picture.make_canvas()
picture.add_grid(11,14)
picture.plot_shapes(stips, **dp)
picture.fig

save = False

picture.make_canvas()
picture.plot_shapes(stips, **dp)
picture.display_overlays(False)
if save:
    picture.fig.savefig("marie_stipple_test.svg")
picture.fig
"""

     '\n\nrescale = img.shape[1]/9\n\npicture = canvas.Canvas(paper_size=(11,14),
     margin_percent=0.05, origin=\'corner\')\npicture_bbox =
     Box(picture.bbox[0,0], picture.bbox[0,1], picture.bbox[1,0],
     picture.bbox[1,1])\ndp = {\n    "alpha":0.7,\n
     "linewidth":0.25*0.0393701*72,\n    "clear":False,\n}\n\n#stips =
     [affinity.translate(affinity.scale(Point(xy[::-1]).buffer(r/3), 1/rescale, 1/
     rescale, origin=(0,0)), 1,0) for xy, r in stipples[0:15000]]\nstips =
```

```
    laffinity.translate(affinity.scale(Point(xy[::-1]).buffer(0.25*0.0393701*72/2
    1/rescale, 1/rescale, origin=(0,0)), 1,0) for xy, r in
    stipples[0:15000]]\n\npicture.make_canvas()\npicture.add_grid(11,14)\npicture.
    **dp)\npicture.fig\n\nsave =
    False\n\npicture.make_canvas()\npicture.plot_shapes(stips,
    **dp)\npicture.display_overlays(False)\nif save:\n
    picture.fig.savefig("marie_stipple_test.svg")\npicture.fig\n'


"""
#stips = [affinity.translate(affinity.scale(Point(xy[::-1]).buffer(r/3), 1/rescal
#stips = [affinity.translate(affinity.scale(Point(xy[::-1]).buffer(0.25*0.039370
stips = [Point(11 - (xy[1]/rescale + 1), xy[0]/rescale).buffer(0.25*0.0393701/2)


picture.make_canvas()
picture.add_grid(11,14)
picture.plot_shapes(stips, **dp)
picture.fig

save = True



picture.make_canvas()
picture.plot_shapes(stips, **dp)
picture.display_overlays(False)
if save:
    picture.fig.savefig("marie_stipple_full.svg")
picture.fig




mpl_circles = [Circle((11 - (xy[1]/rescale + 1), xy[0]/rescale), 0.25*0.0393701/2
picture.make_canvas()
picture.add_grid(11,14)
picture.ax.add_collection(PatchCollection(mpl_circles))
picture.fig


save = False

picture.make_canvas()
picture.ax.add_collection(PatchCollection(mpl_circles))
picture.display_overlays(False)
if save:
    picture.fig.savefig("marie_stipple_full.svg")
picture.fig
"""

    '\n#stips = [affinity.translate(affinity.scale(Point(xy[::-1]).buffer(r/3),
```

```
      1/rescale, 1/rescale, origin=(0,0)), 1,0) for xy, r in
      stipples[0:15000]]\n#stips =
      [affinity.translate(affinity.scale(Point(xy[::-1]).buffer(0.25*0.0393701*72/2)
      1/rescale, 1/rescale, origin=(0,0)), 1,0) for xy, r in stipples]\nstips =
      [Point(11 - (xy[1]/rescale + 1), xy[0]/rescale).buffer(0.25*0.0393701/2) for
      xy,r in
      stipples]\n\n\npicture.make_canvas()\npicture.add_grid(11,14)\npicture.plot_sh
      **dp)\npicture.fig\n\nsave =
      True\n\n\n\npicture.make_canvas()\npicture.plot_shapes(stips,
      **dp)\npicture.display_overlays(False)\nif save:\n
      picture.fig.savefig("marie_stipple_full.svg")\npicture.fig\n\n\n\n\nmpl_circle
      = [Circle((11 - (xy[1]/rescale + 1), xy[0]/rescale), 0.25*0.0393701/2) for
      xy,r in
      stipples]\npicture.make_canvas()\npicture.add_grid(11,14)\npicture.ax.add_coll
      =
      False\n\npicture.make_canvas()\npicture.ax.add_collection(PatchCollection(mpl_
      save:\n    picture.fig.savefig("marie_stipple_full.svg")\npicture.fig\n'
```

```
from pyaxidraw import axidraw
ad = axidraw.AxiDraw()
ad.interactive()
ad.connect()
```

```
ad.options.model = 2
ad.options.pen_pos_up = 50
ad.options.pen_pos_down = 43
ad.update()
```

```
ad.plot_setup("marie_stipple_svg.svg")
ad.plot_run()
```

```
    Plot paused by button press.
    Use the resume feature to continue.
```

```
ad.options.mode = "res_plot"
ad.options.pen_pos_up = 50
ad.options.pen_pos_down = 43
ad.update()
```

```
ad.plot_run()
```

```
    Failed after command: SC,4,18843
    Failed after command: SC,5,17585
    Failed after command: SC,11,1350
    Failed after command: SC,12,900
    Failed after command: EM,1,1
    Warning: AxiDraw movement was limited by its physical range of motion. If ever
```

Start coding or generate with AI

Start coding or generate with AI.

Start coding or generate with AI.

Figure 5.1: A placeholder image for [14]

Bibliography

[1] Harold Abelson, Gerald Jay Sussman, and Julie Sussman. *Structure and Interpretation of Computer Programs*. MIT Press, 2nd edition, 1996. Commonly referred to as "The Wizard Book".

[2] Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools*. Pearson, 2nd edition, 2006. Commonly referred to as "The Dragon Book".

[3] Geoffrey Bradway and Amanda Ngo. *Principia Mathematica II: Electric Bugaloo*. https://www.archive.org, 2025.

[4] Lee Stemkoski Dominic Klyve. The euler archive, 2025. Accessed: 2025-01-12.

[5] Bhante Henepola G. *Bhavana Society*.

[6] Haleh Liza Gafori. *Gold: Rumi*. New York Review Books, 2022.

[7] John Green. *The Anthropocene Reviewed: Essays on a Human-Centered Planet*. Dutton, 2021. A collection of essays reviewing facets of the human experience in the Anthropocene era.

[8] Bhante Henepola Gunaratana. *Mindfulness in Plain English*. Wisdom Publications, 1991.

[9] Davar Khoshnevisan. *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2007. Hardback Edition.

[10] Isaac Newton. *Philosophiae Naturalis Principia Mathematica*. Royal Society of London, 1687. Translated by I. Bernard Cohen and Anne Whitman in 1999.

[11] Massachusetts Institute of Technology. Mit - massachusetts institute of technology, 2025. Accessed: 2025-01-12.

[12] University of Utah. *Utah Tea Pot*.

[13] University of Utah. University of utah, 2025. Accessed: 2025-01-12.

[14] Neri Oxman. Vespers: Series i-v (the death masks), 2016. Exhibited at the Museum of Modern Art (MoMA), New York, NY.

[15] Tathagata Meditation Center. Tathagata Meditation Center, n.d. Accessed: 2025-01-12.

[16] Yale University. Yale university, 2025. Accessed: 2025-01-12.

[17] Berkeley University of California. Uc berkeley - university of california, berkeley, 2025. Accessed: 2025-01-12.

[18] Vetro Editions. Vetro Editions, n.d. Accessed: 2025-01-12.

Figure 2: A Utah Teapot [12]

.1 Dedication outro.

```
 By means of our meritorious deeds,

May the Suffering be free from Suffering
May the Fear-Struck be free from Fear
May the Grieving be free from Grief
So too may all Being-Be.
From the highest realms of existence to the lowest,
May all being arisen in these realms,
With form and without form,
With perception and without perception,
Be released from all Suffering,
And attain to Perfect Peace
May all being be free from suffering

Sadu!  Sadu!  Sadu!
```

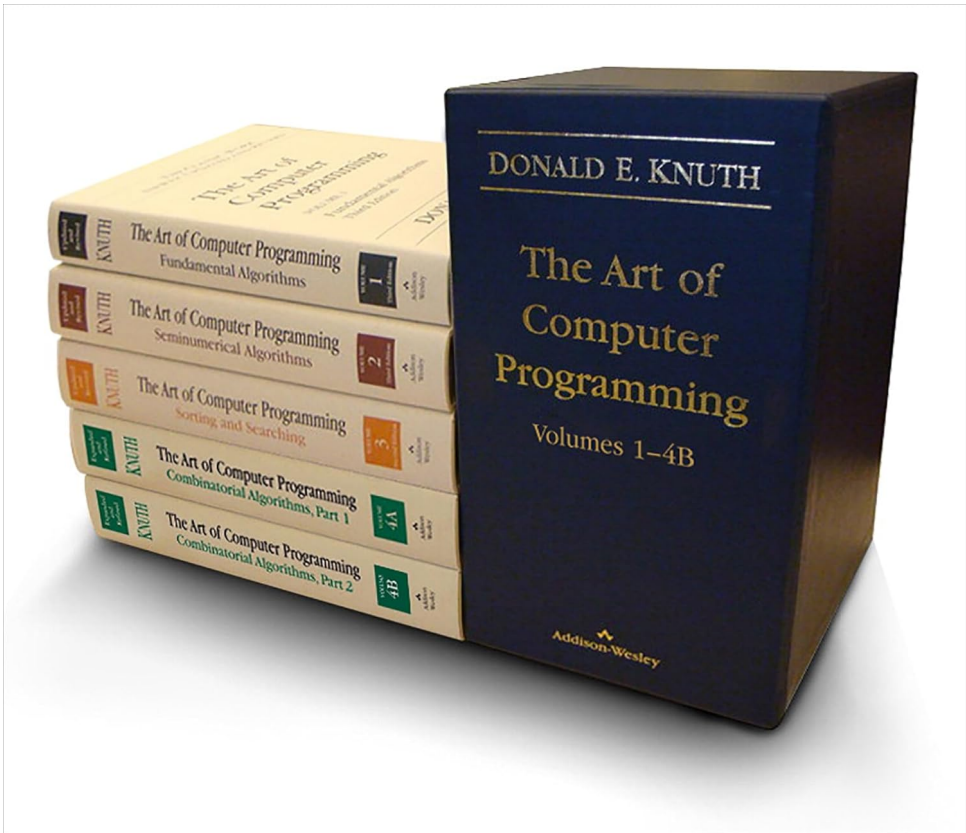


Figure 3: It can't be that hard, can it? Did you try reading the manuel? [4]

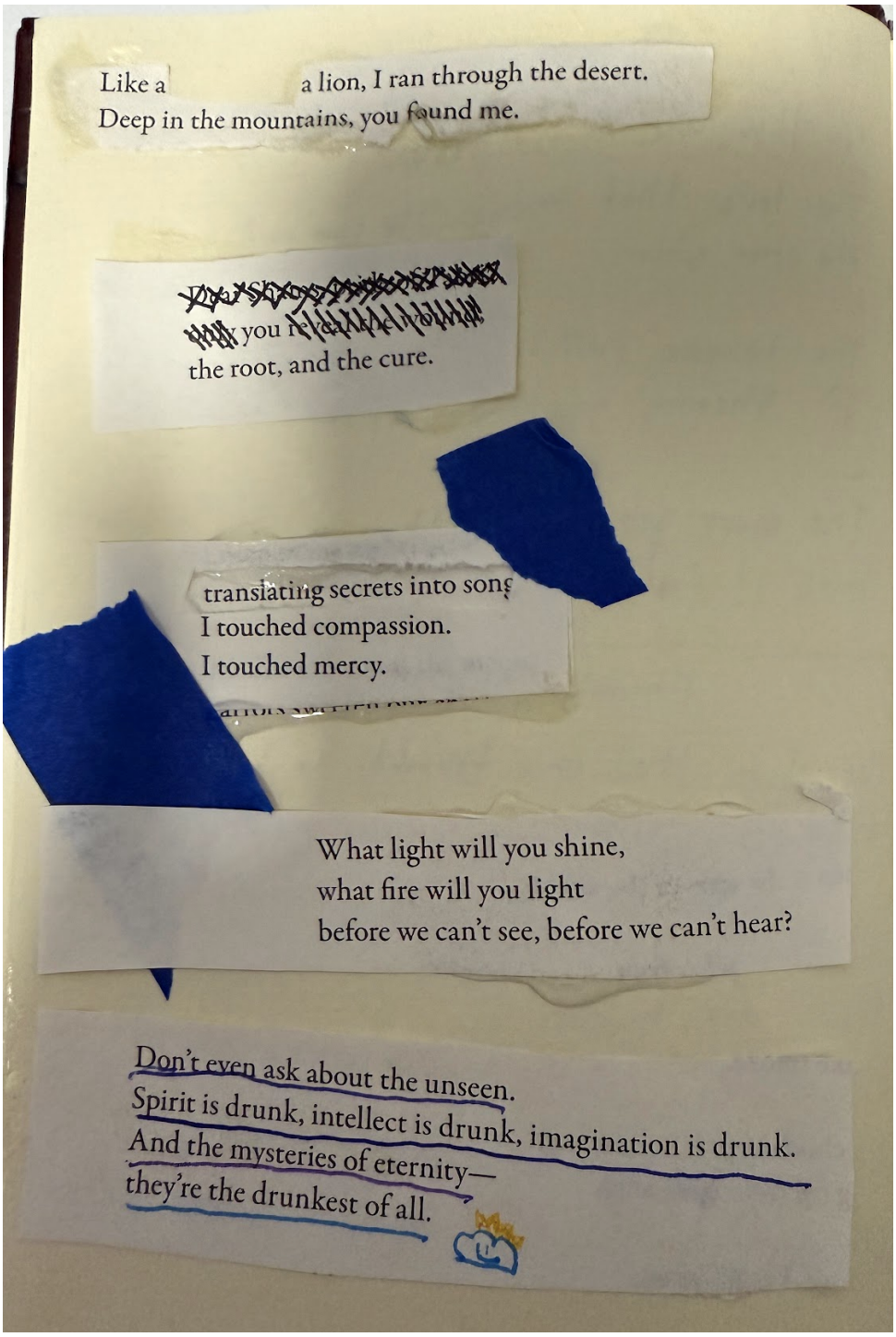

Like a a lion, I ran through the desert.
Deep in the mountains, you found me.
the root, and the cure.
translating secrets into song
I touched compassion.
I touched mercy.
What light will you shine,
what fire will you light
before we can't see, before we can't hear?
Don't even ask about the unseen.
Spirit is drunk, intellect is drunk, imagination is drunk.
And the mysteries of eternity—
they're the drunkest of all.

Figure 4: [6]